Documentation 5.2

ZABBIX

01.07.2025

Contents

Zabbix Manual **4**

Copyright notice . . . 4

附录 . . . 4

1 常见问题/疑难解答赵 . . . 4

2 安装赵 . . . 5

3 后台进程配置 . . . 16

4 各种协议 . . . 112

5 监控项 . . . 148

6 触发器赵 . . . 179

7 宏赵 . . . 196

8 单位符号说明 . . . 236

9 时间段配置 . . . 237

10 命令执行 . . . 238

11 监控方案 . . . 239

12 Performance tuning . . . 241

13 Version compatibility . . . 243

14 Database error handling . . . 244

15 Zabbix sender dynamic link library for Windows . . . 244

16 Issues with SELinux . . . 245

1. 简介 . . . 245

1 手册结构 . . . 245

2 Zabbix 介绍 . . . 246

3 Zabbix 功能 . . . 246

4 Zabbix 概述 . . . 247

2. 定义 . . . 248

3. 进程 . . . 249

1 Server . . . 249

2 Agent . . . 252

3 Proxy . . . 256

4 Java gateway . . . 259

5 Sender . . . 261

6 Get . . . 261

4. 安装 . . . 262

1 获取 Zabbix . . . 262

2 安装要求 . . . 262

3 从源代码包安装 . . . 276

4 从二进制包安装 . . . 287

5 从容器中安装 . . . 291

6 升级步骤 . . . 300

7 已知问题 . . . 313

8 模板变更 . . . 315

5. 快速入门 . . . 315

1 登陆和配置用户 . . . 315

2 新建主机 . . . 319

3 新建监控项 . . . 320

4 新建触发器 . . . 322

5 获取问题通知 . . . 324

6 新建模版 . . . 328

6. Zabbix 应用 . . . 330

7. 配置 . . . 334

7. Configuration . . . 334
1 主机和主机组 . . . 338
2 监控项 . . . 363
3 触发器 . . . 522
4 事件 . . . 545
5 事件关联 . . . 548
7 模板 . . . 556
8 模板开箱即用 . . . 557
9 事件通知（ZX） . . . 560
10 宏 . . . 658
11 用户和用户组 . . . 665
8. Service 服务监控 [ZeMing] . . . 685
8. Service monitoring . . . 685
9. Web 监控 [ZeMing] . . . 695
9. Web monitoring . . . 695
1 Web 监控项（items） . . . 730
2 Web 场景监控 . . . 737
10. 虚拟机监控 . . . 744
1 虚拟机发现 key 值字段 . . . 748
11. 维护 . . . 749
12. 正则表达式 [杨青] . . . 755
13. 问题确认 . . . 770
14. 配置导出/导入 . . . 776
1 主机组 . . . 777
2 模板 . . . 777
4 网络拓扑图 . . . 818
5 聚合图形 . . . 837
15. 发现 . . . 849
1 网络发现 . . . 849
2 Active agent 自动注册 . . . 861
3 Low-level discovery . . . 866
16. Distributed monitoring . . . 878
1 Proxies . . . 879
17. Encryption . . . 882
1 Using certificates . . . 884
2 Using pre-shared keys . . . 891
3 Troubleshooting . . . 893
18. Web 界面 . . . 895
1 前端 . . . 895
2 User profile 用户资料 . . . 1038
3 Global search 全局搜索 . . . 1043
4 Frontend maintenance mode 前端维护模式 . . . 1045
5 Page parameters 页面参数 . . . 1046
6 Definitions 定义 . . . 1046
7 Creating your own theme 制定自己的主题 . . . 1048
8 Debug mode 调试模式 . . . 1049
19. API . . . 1050
Method reference . . . 1055
Appendix 1. Reference commentary . . . 1713

Zabbix manpages **1714**

zabbix_agentd . . . 1714
NAME . . . 1715
SYNOPSIS . . . 1715
DESCRIPTION . . . 1715
OPTIONS . . . 1715
FILES . . . 1716
SEE ALSO . . . 1716
AUTHOR . . . 1716
Index . . . 1716
zabbix_get . . . 1716
NAME . . . 1717
SYNOPSIS . . . 1717

DESCRIPTION . . . 1717
OPTIONS . . . 1717
EXAMPLES . . . 1718
SEE ALSO . . . 1718
AUTHOR . . . 1718
Index . . . 1718
zabbix_proxy . . . 1718
NAME . . . 1718
SYNOPSIS . . . 1719
DESCRIPTION . . . 1719
OPTIONS . . . 1719
FILES . . . 1719
SEE ALSO . . . 1719
AUTHOR . . . 1720
Index . . . 1720
zabbix_sender . . . 1720
NAME . . . 1720
SYNOPSIS . . . 1720
DESCRIPTION . . . 1721
OPTIONS . . . 1721
EXIT STATUS . . . 1723
EXAMPLES . . . 1723
SEE ALSO . . . 1723
AUTHOR . . . 1723
Index . . . 1724
zabbix_server . . . 1724
NAME . . . 1724
SYNOPSIS . . . 1724
DESCRIPTION . . . 1724
OPTIONS . . . 1724
FILES . . . 1725
SEE ALSO . . . 1725
AUTHOR . . . 1725
Index . . . 1725

Zabbix Manual

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

Copyright notice

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

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

附录

请使用侧栏访问附录部分中的内容。

1 常见问题/疑难解答赵

常见问题

1. Q：可以更新或清空队列（如菜单 “管理”→“队列” 中所展示的队列）？
   A: 不可以。
2. Q: 如何从一个数据库迁移到另一个数据库？
   A: 只需要转存数据 (对于 MySQL, 使用参数 -t 或 --no-create-info), 用 Zabbix 的 schema 文件创建新的数据库，并导入数据。
3. Q: 想用下划线替换监控项 key 中的所有空格，因为空格只在老版本中起作用，而在 3.0 版本的监控项 key 中，空格不是一个有效的标示符（或者因为其它需要大量修改监控项 key 的场景），应该如何做以及有哪些注意事项？
   A: 可以使用数据库更新语句用下划线替换所有出现的空格:
   update items set key_=replace(key_,' ','_');
   触发器可以使用这些监控项而不需要额外的改动，但是需要修改以下位置的监控项引用:
   * Notifications (actions)
   * Map element and link labels
   * Calculated item formulas
4. Q: 我的图形中有一些点而不是线或者有一些空白区域，为什么会这样?
   A: 数据丢失，这种情况的发生有多种原因——Zabbix 数据库的性能问题、Zabbix 服务器问题、网络问题、监控设备问题...
5. Q: Zabbix 守护进程无法启动消息监听器，错误信息为*: socket() for [[-]:10050] failed with error 22: Invalid argument.*
   A: 当在一个内核 2.6.26 或更低内核版本的操作系统上，试图运行编译的版本为 2.6.27 或更高版本的 Zabbix agent 时会产生该错误。注意，在这种情况下，静态链接不会起作用，因为早期操作系统内核版本中不支持带 SOCK_CLOEXEC 标志的 socket() 系统调用。ZBX-3395
6. Q: 尝试使用一个位置参数（如 $1）去设置一个命令中灵活的用户参数，但它不起作用。怎么解决这个问题？
   A: 使用两个 $$ 符合，像这样 **$$1**
7. Q: 在 Opera11 中，所有的下拉菜单都有一个滚动条，看起来不太美观，为什么会这样呢?
   A: 对于 Opera11.00 和 11.01 操作系统来说，这是一个 bug; 更多信息请访问 Zabbix 问题跟踪。
8. Q: 如何更改自定义主题中的图形背景颜色?
   A: 参照数据库中的 graph_theme 表和主题帮助。
9. Q: 调试等级为 4 时，在 zabbix server/proxy 日志中出现”Trapper got [] len 0” 信息，这是什么原因?
   A: 很有可能是前端有问题，连接并检查服务是否仍在运行。
10. Q: 系统时间设置为将来的某一时间，导致没有数据出现。这个问题怎么解决?
    A: 清除数据库中的字段 hosts.disable_until*, drules.nextcheck, httptest.nextcheck 的值，并重启 zabbix server/proxy。
11. Q: 在前端使用 *{ITEM.VALUE}* 宏或者在其他情况下，item 的文本类型值无论多大都会被修剪为 20 个字符，这种情况正常吗?
    A: 是正常的, 在 include/items.inc.php 中有一个硬编码限制，长度最大仅为 20 个字符。

另见

* zabbix 官方问题解决版块

1 Frequently asked questions / Troubleshooting Frequently asked questions or FAQ.

1. Q: Can I flush/clear the queue (as depicted in *Administration → Queue*)?
   A: No.
2. Q: How do I migrate from one database to another?
   A: Dump data only (for MySQL, use flag -t or --no-create-info), create the new database using schema files from Zabbix and import the data.
3. Q: I would like to replace all spaces with underscores in my item keys because they worked in older versions but space is not a valid symbol for an item key in 3.0 (or any other reason to mass-modify item keys). How should I do it and what should i beware of?
   A: You may use a database query to replace all occurrences of spaces in item keys with underscores:
   update items set key_=replace(key_,' ','_');
   Triggers will be able to use these items without any additional modifications, but you might have to change any item references in these locations:
   * Notifications (actions)
   * Map element and link labels
   * Calculated item formulas
4. Q: My graphs have dots instead of lines or empty areas. Why so?
   A: Data is missing. This can happen for a variety of reasons - performance problems on Zabbix database, Zabbix server, network, monitored devices...
5. Q: Zabbix daemons fail to start up with a message *Listener failed with error: socket() for [[-]:10050] failed with error 22: Invalid argument.*
   A: This error arises at attempt to run Zabbix agent compiled on version 2.6.27 or above on a platform with a kernel 2.6.26 and lower. Note that static linking will not help in this case because it is the socket() system call that does not support SOCK_CLOEXEC flag on earlier kernels. ZBX-3395
6. Q: I try to set up a flexible user parameter (one that accepts parameters) with a command that uses a positional parameter like $1, but it doesn't work (uses item parameter instead). How to solve this?
   A: Use a double dollar sign like **$$1**
7. Q: All dropdowns have a scrollbar and look ugly in Opera 11. Why so?
   A: It's a known bug in Opera 11.00 and 11.01; see Zabbix issue tracker for more information.
8. Q: How can I change graph background colour in a custom theme?
   A: See graph_theme table in the database and theming guide.
9. Q: With DebugLevel 4 I'm seeing messages "Trapper got [] len 0" in server/proxy log - what's that?
   A: Most likely that is frontend, connecting and checking whether server is still running.
10. Q: My system had the time set in the future and now no data is coming in. How could this be solved?
    A: Clear values of database fields hosts.disable_until*, drules.nextcheck, httptest.nextcheck and restart the server/proxy.
11. Q: Text item values in frontend (when using *{ITEM.VALUE}* macro and in other cases) are cut/trimmed to 20 symbols. Is that normal?
    A: Yes, there is a hardcoded limit in include/items.inc.php currently.

See also

* Troubleshooting page on zabbix.org

2 安装赵

2 Installation

1 数据库创建

1 Database creation

概述

Overview

Zabbxi 数据库必须在 Zabbix server 或 proxy 安装的时候创建。A Zabbix database must be created during the installation of Zabbix server or proxy.

本节提供有关创建 Zabbix 数据库的说明。每个受支持的数据库都有对应的创建命令。This section provides instructions for creating a Zabbix database. A separate set of instructions is available for each supported database.

<note 注意 >`schema.sql`, `images.sql` and `data.sql` 这些文件在 Zabbix 的子目录 *database* 下. 如果 Zabbix 是通过分发包安装的, 参考分发包相关文档. :::

<note 重要 > 对于 Zabbix proxy 数据库, 只需要导入 schema.sql (不需要导入 images.sql 和 data.sql) :::

Note:
schema.sql, images.sql and data.sql files are located in the *database* subdirectory of Zabbix sources. If Zabbix was installed from distribution packages, refer to the distribution documentation.

Attention:
For a Zabbix proxy database, **only** schema.sql should be imported (no images.sql nor data.sql)

MySQL

```
shell> mysql -uroot -p<password>
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> grant all privileges on zabbix.* to zabbix@localhost identified by '<password>';
mysql> quit;
```

如果使用 Zabbix 程序包按照手册Debian/Ubuntu 或RHEL/CentOS 导入数据到数据库。

If you use Zabbix packages continue with instructions for Debian/Ubuntu or RHEL/CentOS to import the data into the database.

```
shell> cd database/mysql
shell> mysql -uzabbix -p<password> zabbix < schema.sql
# 下面步骤当创建Zabbix proxy数据库时不需要执行
shell> mysql -uzabbix -p<password> zabbix < images.sql
shell> mysql -uzabbix -p<password> zabbix < data.sql
```

PostgreSQL

需要使用有权限的用户去创建数据库对象。以下 shell 命令将创建 zabbix 用户。在提示下请输入密码并再次确认密码。(注意，可能首先要求输入 sudo 命令对应的用户密码)：

You need to have database user with permissions to create database objects. The following shell command will create user zabbix. Specify password when prompted and repeat password (note, you may first be asked for sudo password):

```
shell> sudo -u postgres createuser --pwprompt zabbix
```

现在将以先前创建的用户作为数据库所有者（参数：-O zabbix）设置数据库 zabbix（最后一个参数）并导入 initial schema 和数据(假设当前目录位于 Zabbix sources 的根目录中)：

Now we will set up the database zabbix (last parameter) with the previously created user as the owner (-O zabbix) and import initial schema and data (assuming you are in the root directory of Zabbix sources):

```
shell> sudo -u postgres createdb -O zabbix zabbix
```

如果使用 Zabbix 程序包按照手册Debian/Ubuntu 或RHEL/CentOS导入数据到数据库。

If you use Zabbix packages continue with instructions for Debian/Ubuntu or RHEL/CentOS to import the data into the database.

```
shell> cd database/postgresql
shell> cat schema.sql | sudo -u zabbix psql zabbix
# 下面步骤当创建Zabbix proxy数据库时不需要执行
shell> cat images.sql | sudo -u zabbix psql zabbix
shell> cat data.sql | sudo -u zabbix psql zabbix
```

Attention:
上面的命令仅作为例子提供参考，它可以在大多数 GNU / Linux 安装中使用。可以使用不同的命令，例如："psql -U <user-name>"，这取决于系统/数据库的配置方式。如果在设置数据库时遇到麻烦，请咨询数据库管理员。

Attention:
The above commands are provided as an example that will work in most of GNU/Linux installations. You can use different commands, e. g. "psql -U <username>" depending on how your system/database are configured. If you have troubles setting up the database please consult your Database administrator.

Oracle

假设在 Oracle 服务器 host 上存在有权限创建数据库对象的用户（用户名为 zabbix，密码为 password), 并且该用户具有/tmp 目录的写入权限。Zabbix 数据库需要使用 UTF8 字符集。检查当前设置：

We assume that a *zabbix* database user with *password* password exists and has permissions to create database objects in ORCL service located on the *host* Oracle database server with a *user* shell user having write access to /tmp directory. Zabbix requires a Unicode database character set and a UTF8 national character set. Check current settings:

```
sqlplus> select parameter,value from v$nls_parameters where parameter='NLS_CHARACTERSET' or parameter='NLS
```

需要将 Zabbix 数据库安装介质拷贝到 Oracle 服务器上的*/tmp/zabbix_images* 目录下：

If you are creating a database for Zabbix server you need to have images from Zabbix sources on the host where Oracle is running. Copy them to a directory */tmp/zabbix_images* on the Oracle host:

```
shell> cd /path/to/zabbix-sources
shell> ssh user@oracle_host "mkdir /tmp/zabbix_images"
shell> scp -r misc/images/png_modern user@oracle_host:/tmp/zabbix_images/
```

现在开始创建数据库：Now prepare the database:

```
shell> cd /path/to/zabbix-sources/database/oracle
shell> sqlplus zabbix/password@oracle_host/ORCL
sqlplus> @schema.sql
# 下面步骤当创建Zabbix proxy数据库时不需要执行
sqlplus> @images.sql
sqlplus> @data.sql
```

然后删掉介质存放的临时目录：Now the temporary directory can be removed:

```
shell> ssh user@oracle_host "rm -rf /tmp/zabbix_images"
```

IBM DB2

```
shell> db2 "create database zabbix using codeset utf-8 territory us pagesize 32768"
shell> cd database/ibm_db2
shell> db2batch -d zabbix -f schema.sql
# 下面步骤当创建Zabbix proxy数据库时不需要执行
shell> db2batch -d zabbix -f images.sql
shell> db2batch -d zabbix -f data.sql
```

Note:
Zabbix server,Zabbix proxy,Zabbix web server 必须将字符集设置为 UTF-8。否则，来自 Zabbix 的文本信息将被 IBM DB2 服务器解析为非 UTF-8 字符集，并在 Zabbix 与数据库之间互相转换。数据库将存储有问题的非 ASCII 字符。

Note:
It is important to set UTF-8 locale for Zabbix server, Zabbix proxy and the web server running Zabbix frontend. Otherwise text information from Zabbix will be interpreted by IBM DB2 server as non-UTF-8 and will be additionally converted on the way from Zabbix to the database and back. The database will store corrupted non-ASCII characters.

Zabbix 前端使用了 SQL 中的 OFFSET 和 LIMIT 语句。为了 SQL 正常工作，IBM DB2 服务器必须设置环境变量 DB2_COMPATIBILITY_VECTOR 的值为 3。在数据库启动之前执行下面的命令：

Zabbix frontend uses OFFSET and LIMIT clauses in SQL queries. For this to work, IBM DB2 server must have DB2_COMPATIBILITY_VECTOR variable be set to 3. Run the following command before starting the database server:

```
shell> db2set DB2_COMPATIBILITY_VECTOR=3
```

SQLite

只有为 **Zabbix proxy** 创建数据库的时候才能使用 SQLite！

Using SQLite is supported for **Zabbix proxy** only!

Note:
如果使用 SQLite 作为 Zabbix proxy 的数据库，创建时如果数据库不存在，将自动创建。

Note:
If using SQLite with Zabbix proxy, database will be automatically created if it does not exist.

```
shell> cd database/sqlite3
shell> sqlite3 /var/lib/sqlite/zabbix.db < schema.sql
```

返回安装部分。Return to the installation section.

2 Windows 下的 Zabbix agent

2 Zabbix agent on Microsoft Windows

agent 配置

Configuring agent

Zabbix agent 作为 Windows 服务运行。Zabbix agent runs as a Windows service.

在一台 Windows 主机上可以运行一个或多个 Zabbix agent 实例。如果安装一个实例可以使用默认的配置文件 `C:\zabbix_agentd.conf` 或者在命令中指定配置文件路径。如果安装多个实例，每一个 agent 必须有自己的配置文件 (其中一个实例可以使用默认的配置文件)。

You can run a single instance of Zabbix agent or multiple instances of the agent on a Microsoft Windows host. A single instance can use the default configuration file `C:\zabbix_agentd.conf` or a configuration file specified in the command line. In case of multiple instances each agent instance must have its own configuration file (one of the instances can use the default configuration file).

在 Zabbix 源文件目录有一个配置文件样例 `conf/zabbix_agentd.win.conf`。

An example configuration file is available in Zabbix source archive as `conf/zabbix_agentd.win.conf`.

关于 Zabbix Windows agent 更多详细信息，参考配置文件 。

See the configuration file options for details on configuring Zabbix Windows agent.

主机名参数

Hostname parameter

主机执行active checks 时，Zabbix agent 需要定义主机名。而且,agent 端的主机名必须和前端配置的主机名”Host name” 完全一致。

To perform active checks on a host Zabbix agent needs to have the hostname defined. Moreover, the hostname value set on the agent side should exactly match the ”Host name” configured for the host in the frontend.

agent 端的主机名可以通过配置文件configuration file中的 **Hostname** 或 **HostnameItem** 参数定义 - 如果不指定参数值将使用默认的主机名字。

The hostname value on the agent side can be defined by either the **Hostname** or **HostnameItem** parameter in the agent configuration file - or the default values are used if any of these parameters are not specified.

参数 **HostnameItem** 的默认值即 agent 端 key 值为”system.hostname” 的监控项返回值，对于 Windows 平台返回的是 NetBIOS 的主机名。

The default value for **HostnameItem** parameter is the value returned by the ”system.hostname” agent key and for Windows platform it returns the NetBIOS host name.

参数 **Hostname** 默认值为 **HostnameItem** 参数的返回值。所以，实际上，如果这两个参数都是未指定的，实际的主机名将是主机 NetBIOS 名称; Zabbix agent 将使用 NetBIOS 主机名从 Zabbix server 获取 active checks 列表，并将检查结果发送给它。

The default value for **Hostname** is the value returned by the **HostnameItem** parameter. So, in effect, if both these parameters are unspecified the actual hostname will be the host NetBIOS name; Zabbix agent will use NetBIOS host name to retrieve the list of active checks from Zabbix server and send results to it.

<note important>**system.hostname**key 始终返回限制为 15 个字符的 NetBIOS 主机名，并且全为大写字符 - 而不管实际主机名中的长度和字符大小写。:::

> **Attention:**
> The **system.hostname** key always returns the NetBIOS host name which is limited to 15 symbols and in UPPERCASE only - regardless of the length and lowercase/uppercase characters in the real host name.

从 Windows Zabbix agent 1.8.6 版本开始，”system.hostname” key 支持可选参数 -名称的类型。此参数的默认值为”netbios” (用于向后兼容) 另一个可能的值是”host”.

Starting from Zabbix agent 1.8.6 version for Windows the ”system.hostname” key supports an optional parameter - *type* of the name. The default value of this parameter is ”netbios” (for backward compatibility) and the other possible value is ”host”.

<note important>**system.hostname[host]** 键总是返回完整的，实际的（区分大小写的）Windows 主机名。:::

Attention:
The **system.hostname[host]** key always returns the full, real (case sensitive) Windows host name.

因此，为了简化 zabbix_agentd.conf 文件的配置并使其统一起来，可以使用两种不同的方法。

1. 不定义 **Hostname** 或者 **HostnameItem** 参数，Zabbix agent 将使用 NetBIOS 主机名作为主机名;
2. 不定义 **Hostname** 参数，定义 **HostnameItem** 如:
   HostnameItem=system.hostname[host]
   Zabbix agent 将使用完整的，实际的（区分大小写的）Windows 主机名作为主机名。

So, to simplify the configuration of zabbix_agentd.conf file and make it unified, two different approaches could be used.

1. leave **Hostname** or **HostnameItem** parameters undefined and Zabbix agent will use NetBIOS host name as the hostname;
2. leave **Hostname** parameter undefined and define **HostnameItem** like this:
   HostnameItem=system.hostname[host]
   and Zabbix agent will use the full, real (case sensitive) Windows host name as the hostname.

主机名也用作 Windows 服务名称的一部分，用于安装，启动，停止和卸载 Windows 服务。例如，如果 Zabbix agent 配置文件指定 `Hostname=Windows_db_server`, 那么 agent 将作为 Windows 服务"`Zabbix Agent [Windows_db_server]`" 安装。因此，如果要每个 Zabbix agent 实例拥有不同的 Windows 服务名称，则每个实例都必须使用不同的主机名。

Host name is also used as part of Windows service name which is used for installing, starting, stopping and uninstalling the Windows service. For example, if Zabbix agent configuration file specifies `Hostname=Windows_db_server`, then the agent will be installed as a Windows service "`Zabbix Agent [Windows_db_server]`". Therefore, to have a different Windows service name for each Zabbix agent instance, each instance must use a different host name.

将代理安装为 Windows 服务

Installing agent as Windows service

使用默认配置文件 `c:\zabbix_agentd.conf` 安装 Zabbix agent 的单个实例:

To install a single instance of Zabbix agent with the default configuration file `c:\zabbix_agentd.conf`:

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

<note important> 在 64 位系统上，运行 64 位进程相关的所有检查都正常工作需要 64 位的 Zabbix agent 版本。:::

Attention:
On a 64-bit system, a 64-bit Zabbix agent version is required for all checks related to running 64-bit processes to work correctly.

如果希望使用除 `c:\zabbix_agentd.conf` 以外的配置文件, 应该使用以下命令进行服务安装：

If you wish to use a configuration file other than `c:\zabbix_agentd.conf`, you should use the following command for service installation:

```
zabbix_agentd.exe --config <your_configuration_file> --install
```

应指定配置文件的全路径。A full path to the configuration file should be specified.

Zabbix agent 多实例作为服务安装的命令如下: Multiple instances of Zabbix agent can be installed as services like this:

```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --install --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --install --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --install --multiple-agents
```

现在在控制面板中可以看到安装的服务。The installed service should now be visible in Control Panel.

启动 agent

Starting agent

启动 agent 服务，可以使用控制面板或通过命令行方式。To start the agent service, you can use Control Panel or do it from command line.

启动使用默认配置文件的单实例 Zabbix agent 命令如下：To start a single instance of Zabbix agent with the default configuration file:

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

启动使用自定义配置文件的单实例 Zabbix agent 命令如下：To start a single instance of Zabbix agent with another configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --start
```

启动多实例 Zabbix agent 中的一个实例命令如下: To start one of multiple instances of Zabbix agent:

```
zabbix_agentd.exe --config <configuration_file_for_this_instance> --start --multiple-agents
```

停止 agent

Stopping agent

停止 agent 服务，可以使用控制面板或通过命令行方式。To stop the agent service, you can use Control Panel or do it from command line.

停止使用默认配置文件的单实例 Zabbix agent 命令如下：To stop a single instance of Zabbix agent started with the default configuration file:

```
zabbix_agentd.exe --stop
```

停止使用自定义配置文件的单实例 Zabbix agent 命令如下：To stop a single instance of Zabbix agent started with another configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --stop
```

停止多实例 Zabbix agent 中的一个实例命令如下: To stop one of multiple instances of Zabbix agent:

```
zabbix_agentd.exe --config <configuration_file_for_this_instance> --stop --multiple-agents
```

卸载 agent Windows 服务

Uninstalling agent Windows service

卸载使用默认配置文件的单实例 Zabbix agent 服务命令如下：To uninstall a single instance of Zabbix agent using the default configuration file:

```
zabbix_agentd.exe --uninstall
```

卸载使用自定义配置文件的单实例 Zabbix agent 服务命令如下：To uninstall a single instance of Zabbix agent using a non-default configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --uninstall
```

卸载多实例 Zabbix agent 服务命令如下: To uninstall multiple instances of Zabbix agent from Windows services:

```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --uninstall --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --uninstall --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --uninstall --multiple-agents
```

3 Elasticsearch 配置

3 Elasticsearch setup

> **Attention:**
> Elasticsearch 的支持是实验性的！本节中考虑的配置步骤适用于以下 Elasticsearch 版本：** 5.0.x - > 6.1.x **。如果使用早期或更高版本的 Elasticsearch，某些功能可能无法按预期工作。

> **Attention:**
> Elasticsearch support is experimental!
> Setup procedure considered in this section is applicable to the following Elasticsearch versions: **5.0.x -→ 6.1.x**. In case an earlier or later version of Elasticsearch is used, some functionality may not work as intended.

Zabbix 最近开始支持通过使用 Elasticsearch 而不是数据库来存储历史数据。现在，用户可以在兼容数据库和 Elasticsearch 之间选择历史数据的存储位置。

Zabbix has recently started to support storage of historical data by means of Elasticsearch instead of a database. Users are now given the possibility to choose the storage place for historical data between a compatible database and Elasticsearch.

配置

Configuration

正确的设置 Zabbix server 配置文件和前端配置文件中的参数，以保证所有元素之间的正确通信。To ensure proper communication between all elements involved make sure server configuration file and frontend configuration file parameters are properly configured.

Zabbix server 和前端

Zabbix server and frontend

已经更新参数的 Zabbix server 配置文件示例如下：Zabbix server configuration file draft with parameters to be updated:

```
### Option: HistoryStorageURL
# History storage HTTP[S] URL.
#
# Mandatory: no
# Default:
# HistoryStorageURL=
### Option: HistoryStorageTypes
# Comma separated list of value types to be sent to the history storage.
#
# Mandatory: no
# Default:
# HistoryStorageTypes=uint,dbl,str,log,text
```

用于设置 Zabbix server 配置文件的示例参数值如下：Example parameter values to fill the Zabbix server configuration file with:

```
HistoryStorageURL=http://test.elasticsearch.lan:9200
HistoryStorageTypes=str,log,text
```

此配置文件使 Zabbix Server 将相应数据库中的数值类型和文本类型的历史数据存储到 Elasticsearch 中。This configuration forces Zabbix Server to store history values of numeric types in the corresponding database and textual history data in Elasticsearch.

Elasticsearch 支持以下几种监控项类型：Elasticsearch supports the following item types:

```
uint,dbl,str,log,text
```

支持的监控项类型说明如下：Supported item type explanation:

Item value type	Database table	Elasticsearch type
Numeric (unsigned)	history_uint	uint
Numeric (float)	history	dbl
Character	history_str	str
Log	history_log	log
Text	history_text	text

已经更新参数的 Zabbix 前端配置文件 (`conf/zabbix.conf.php`) 示例如下：Zabbix frontend configuration file (`conf/zabbix.conf.php`) draft with parameters to be updated:

```
// Elasticsearch url (can be string if same url is used for all types).
$HISTORY['url']   = [
      'uint' => 'http://localhost:9200',
      'text' => 'http://localhost:9200'
];
// Value types stored in Elasticsearch.
$HISTORY['types'] = ['uint', 'text'];
```

用于设置 Zabbix 前端配置文件的示例参数值如下：Example parameter values to fill the Zabbix frontend configuration file with:

```
$HISTORY['url']   = 'http://test.elasticsearch.lan:9200';
$HISTORY['types'] = ['str', 'text', 'log'];
```

此配置文件将文本、字符、日志类型的历史数据存储到 Elasticsearch 中。This configuration forces to store `Text`, `Character` and `Log` history values in Elasticsearch.

还需要在 `conf / zabbix.conf.php` 文件中配置 $HISTORY 为全局参数，以确保一切正常工作(请参阅 `conf / zabbix.conf.php.exampl` 以了解如何配置)：

It is also required to make $HISTORY global in `conf/zabbix.conf.php` to ensure everything is working properly (see `conf/zabbix.conf.php.example` for how to do it):

```
// Zabbix GUI configuration file.
global $DB, $HISTORY;
```

Elasticsearch 配置和创建映射

Installing Elasticsearch and creating mapping

正常配置包括安装 Elasticsearch 和创建映射两个步骤。Final two steps of making things work are installing Elasticsearch itself and creating mapping process.

安装 Elasticsearch 请参考Elasticsearch 安装指南

To install Elasticsearch please refer to Elasticsearch installation guide.

Note:
映射是 Elasticsearch 中的数据结构（类似于数据库中的表）。此处提供了所有历史数据类型的映射：`database / elasticsearch / elasticsearch.map`。

Note:
Mapping is a data structure in Elasticsearch (similar to a table in a database). Mapping for all history data types is available here: `database/elasticsearch/elasticsearch.map`.

Warning:
必须创建映射。如果未按照要求创建映射，则某些功能将无法正常使用。

Warning:
Creating mapping is mandatory. Some functionality will be broken if mapping is not created according to the instruction.

创建 `text` 类型的映射可以发送如下请求到 Elasticsearch：To create mapping for `text` type send the following request to Elasticsearch:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/text \
 -H 'content-type:application/json' \
 -d '{
   "settings" : {
      "index" : {
         "number_of_replicas" : 1,
         "number_of_shards" : 5
      }
   },
   "mappings" : {
      "values" : {
         "properties" : {
            "itemid" : {
               "type" : "long"
            },
            "clock" : {
               "format" : "epoch_second",
               "type" : "date"
            },
            "value" : {
               "fields" : {
                  "analyzed" : {
                     "index" : true,
                     "type" : "text",
                     "analyzer" : "standard"
                  }
               },
               "index" : false,
               "type" : "text"
            }
         }
      }
   }
}'
```

对于创建 “字符” 和 “日志” 类型的历史数据映射，需要执行类似的请求，请求内容需要进行相应的修改。Similar request is required to be executed for `Character` and Log history values mapping creation with corresponding type correction.

> **Note:**
> 要使用 Elasticsearch，请参阅Requirement page以获取更多信息。

> **Note:**
> To work with Elasticsearch please refer to Requirement page for additional information.

> **Note:**
> Housekeeper 不会删除任何 Elasticsearch 中的数据

> **Note:**
> Housekeeper is not deleting any data from Elasticsearch.

历史数据存储于多个基于时间的索引

Storing history data in multiple date-based indices

本节介绍使用 pipeline 和 ingest 节点所需的其他配置步骤。This section describes additional steps required to work with pipelines and ingest nodes.

首先必须为索引创建一个模板。创建 uint 模板的请求示例如下：To begin with, you must create templates for indices. The following example shows a request for creating uint template:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_template/uint_template \
 -H 'content-type:application/json' \
 -d '{
   "template": "uint*",
   "index_patterns": ["uint*"],
   "settings" : {
      "index" : {
         "number_of_replicas" : 1,
         "number_of_shards" : 5
      }
   },
   "mappings" : {
      "values" : {
         "properties" : {
            "itemid" : {
               "type" : "long"
            },
            "clock" : {
               "format" : "epoch_second",
               "type" : "date"
            },
            "value" : {
               "type" : "long"
            }
         }
      }
   }
}'
```

要创建其他模板，用户应更改请求 URL（最后一部分是模板名称），更改 “template” 和 “index_patterns” 字段以匹配索引名称并设置可从 “database / elasticsearch / elasticsearch” 中获取的有效映射。例如，下面的命令能为一个文本索引创建一个模板：

To create other templates, user should change the URL (last part is the name of template), change ”template” and ”index_patterns” fields to match index name and to set valid mapping that can be taken from `database/elasticsearch/elasticsearch.map`. For example, the following command can be used to create a template for text index:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_template/text_template \
```

```
 -H 'content-type:application/json' \
 -d '{
   "template": "text*",
   "index_patterns": ["text*"],
   "settings" : {
      "index" : {
         "number_of_replicas" : 1,
         "number_of_shards" : 5
      }
   },
   "mappings" : {
      "values" : {
         "properties" : {
            "itemid" : {
               "type" : "long"
            },
            "clock" : {
               "format" : "epoch_second",
               "type" : "date"
            },
            "value" : {
               "fields" : {
                  "analyzed" : {
                     "index" : true,
                     "type" : "text",
                     "analyzer" : "standard"
                  }
               },
               "index" : false,
               "type" : "text"
            }
         }
      }
   }
}'
```

这是允许 Elasticsearch 为自动创建的索引设置有效的映射所必需做的。然后需要创建 pipeline 定义。在将数据放入索引之前，pipeline 能对数据进行多种预处理操作。以下命令可用于为 uint 索引创建 pipeline：

This is required to allow Elasticsearch to set valid mapping for indices created automatically. Then it is required to create the pipeline definition. Pipeline is some sort of preprocessing of data before putting data in indices. The following command can be used to create pipeline for uint index:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_ingest/pipeline/uint-pipeline \
 -H 'content-type:application/json' \
 -d '{
  "description": "daily uint index naming",
  "processors": [
    {
      "date_index_name": {
        "field": "clock",
        "date_formats": ["UNIX"],
        "index_name_prefix": "uint-",
        "date_rounding": "d"
      }
    }
  ]
}'
```

用户可以修改参数（“date_rounding”）来设置特定的索引循环周期。要创建其他 pipeline，用户应更改请求 URL（最后一部分是 pipeline 名称）并更改 “index_name_prefix” 字段以匹配索引名称。

User can change the rounding parameter (”date_rounding”) to set a specific index rotation period. To create other pipelines, user should change the URL (last part is the name of pipeline) and change ”index_name_prefix” field to match index name.

可以参考 Elasticsearch 文档.

See also Elasticsearch documentation.

另外，可以通过 Zabbix server 配置文件中新加的参数来配置将历史数据存储于基于时间的多个索引。Additionally, storing history data in multiple date-based indices should also be enabled in the new parameter in Zabbix server configuration:

```
### Option: HistoryStorageDateIndex
# Enable preprocessing of history values in history storage to store values in different indices based on
# 0 - disable
# 1 - enable
#
# Mandatory: no
# Default:
# HistoryStorageDateIndex=0
```

故障诊断

Troubleshooting

以下步骤可帮助您解决 Elasticsearch 的配置问题：The following steps may help you troubleshoot problems with Elasticsearch setup:

1. 检查映射是否正确 (通过 URL 的发送 GET 请求获取索引信息，例如：`http://localhost:9200/uint`)。
2. 检查 shards 状态是否正常 (不正常时重启 Elasticsearch 可能解决问题)。
3. 检查 Elasticsearch 配置文件，配置文件应允许从 Zabbix 前端主机和 Zabbix server 主机进行访问。
4. 检查 Elasticsearch 日志。

1. Check if the mapping is correct (GET request to required index URL like `http://localhost:9200/uint`).
2. Check if shards are not in failed state (restart of Elasticsearch should help).
3. Check the configuration of Elasticsearch. Configuration should allow access from the Zabbix frontend host and the Zabbix server host.
4. Check Elasticsearch logs.

如果您仍然遇到配置问题，请创建一个错误报告，其中包含映射，错误日志，配置，版本等信息。If you are still experiencing problems with your installation then please create a bug report with all the information from this list (mapping, error logs, configuration, version, etc.)

4 实时导出事件，监控项采集值，趋势数据

4 Real-time export of events, values, trends

概述

Overview

可以配置使用换行符分隔的 JSON 格式实时导出触发器事件，监控项采集值，趋势数据。

It is possible to configure real-time exporting of trigger events, item values and trends in a newline-delimited JSON format.

导出完成后的文件中，每一行都是 JSON 对象。值映射不被应用。Exporting is done into files, where each line of the export file is a JSON object. Value mappings are not applied.

如果无法将数据写入导出文件或无法重命名导出文件或重命名后无法创建新文件，则 Zabbix 将以 10 秒的间隔进行重试，直到成功为止。In case data cannot be written to the export file or the export file cannot be renamed or new one cannot be created after renaming it, Zabbix will retry with 10 second interval until success.

有关导出数据的详细信息，请参见导出协议页面。For precise details on what information is exported, see the export protocol page.

请注意，如果在收到数据后，在服务器导出数据之前删除了主机/监控项，主机/监控项可能没有元数据（例如：主机组，主机名，监控项名称）。Note that host/item can have no metadata (host groups, host name, item name) if the host/item was removed after the data was received, but before server exported data.

配置

Configuration

实时导出触发器事件，监控项采集值，趋势数据可以通过 Zabbix server 配置文件中的 `ExportDir` 参数来指定数据导出目录。Real-time export of trigger events, item values and trends is configured by specifying a directory for the export files - see the `ExportDir` parameter in server configuration.

另外一个参数 `ExportFileSize` 可用于设置单个导出文件的最大允许大小。Another parameter - `ExportFileSize` may be used to set the maximum allowed size of an individual export file.

当进程需要写入文件时，它首先检查文件的大小。如果超出配置的大小限制，则重命名该文件 (在原文件名后加入.old 后缀)，并创建具有原文件名的新文件。

When a process needs to write to a file it checks the size of the file first. If it exceeds the configured size limit, the file is renamed by appending .old to its name and a new file with the original name is created.

3 后台进程配置

3 Daemon configuration

1 Zabbix server

> **Note:**
> 本节中参数的默认值只是代表守护进程的默认使用值，实际运行时应以你所使用的配置文件中定义的参数值为准。

> **Note:**
> The default values reflect daemon defaults, not the values in the shipped configuration files.

以下参数可以在 Zabbix server 配置文件中配置：The parameters supported in a Zabbix server configuration file:

参数名称必须配	范围默	值	描述信息	
Parameter	Mandatory	Range	Default	Description
AlertScriptsPath	否		usr/local/share/zabbix/alertscripts	定义报警脚本位置(依赖编译安装时的参数设置*datadir*)。

参数名称必须配		范围默	值	描述信息
AlertScriptsPath		no		/usr/local/share/zabbix/alertscripts Location of custom alert scripts (depends on compile-time installation variable *datadir*).

AllowRoot	否			许服务以’root’ 身份运行。如果该参数配置为禁止，并且服务仍以 root 身份启动，服务会切换到使用’zabbix’ 用户启动。对于以普通用户启动的，该参数没有影响。 0 - 禁止 1 - 允许 Zabbix 2.2.0。以后的版本都支持这个参数
AllowRoot	no		0	Allow the server to run as ’root’. If disabled and the server is started by ’root’, the server will try to switch to the ’zabbix’ user instead. Has no effect if started under a regular user. 0 - do not allow 1 - allow This parameter is supported since Zabbix 2.2.0.

CacheSize	否	28K-8G	M	存大小, 单位为字节。用于存储主机、监控项、触发器数据的共享内存大小。 Zabbix2.2.3 以前的版本最大可配置值为 2GB。

CacheSize	no	128K-8G	8M	Size of configuration cache, in bytes. Shared memory size for storing host, item and trigger data. Upper limit used to be 2GB before Zabbix 2.2.3.

CacheUpdateFrequency	否	-3600	0	abbix 缓存更新频率，单位为秒. 另外参考runtime control 选项。
CacheUpdateFrequency	no	1-3600	60	How often Zabbix will perform update of configuration cache, in seconds. See also runtime control options.

DBHost	否		ocalhost	据库主机名。 如果是 MySQL localhost 或空字符串会导致使用套接字。如果是 PostgreSQL 只有空字符串会使用套接字。
DBHost	no		localhost	Database host name. In case of MySQL localhost or empty string results in using a socket. In case of PostgreSQL only empty string results in attempt to use socket.

DBName	是		据库名称。
DBName	yes		Database name.

DBPassword	否			据库登录密码。 如果数据库没有密码，请注释掉此参数。
DBPassword	no			Database password. Comment this line if no password is used.

DBPort	否	024-65535	地套接字链接时不使用该参数。
DBPort	no	1024-65535	Database port when not using local socket.

DBSchema	否	据库 Schema 名字。仅 IBM DB2 和 PostgreSQL 使用。
DBSchema	no	Schema name. Used for IBM DB2 and PostgreSQL.

DBSocket	否	ySQL 套接字文件的路径。
DBSocket	no	Path to MySQL socket file.

DBUser	否	据库用户名。
DBUser	no	Database user.

DebugLevel	否	-5		定调试等级: 0 - Zabbix 进程的起停基本信息 1 - 重要信息 2 - 错误信息 3 - 警告信息 4 - 调试信息 (产生大量信息) 5 - 扩展调试 (产生更多信息) 另外可参考runtime control 选项。
DebugLevel	no	0-5	3	Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information) See also runtime control options.

ExportDir	否	换行符分隔的 JSON 格式实时导出事件, 历史数据和趋势数据到这个目录。如果设置，则启用实时导出数据到这个目录。 此参数从 Zabbix 4.0.0 开始支持。
ExportDir	no	Directory for real-time export of events, history and trends in newline-delimited JSON format. If set, enables real-time export. This parameter is supported since Zabbix 4.0.0.

ExportFileSize	否	M-1G	G	个导出文件的最大限制，单位为字节。仅当 ExportDir 参数设置后才使用。 此参数从 Zabbix 4.0.0 开始支持。
ExportFileSize	no	1M-1G	1G	Maximum size per export file in bytes. Only used for rotation if ExportDir is set. This parameter is supported since Zabbix 4.0.0.

ExternalScripts	否		usr/local/share/zabbix/externalscripts	脚本位置（依赖编译安装时的环境变量 *datadir*）。
ExternalScripts	no		/usr/local/share/zabbix/externalscripts	Location of external scripts (depends on compile-time installation variable *datadir*).

Fping6Location	否		usr/sbin/fping6	ping6 程序的路径。 确保 fping6 程序的所有者是 root 用户，并且设置了 SUID 标记。 如果需要 fping 程序处理 IPv6 地址，就置空（”Fping6Location=”）参数。
Fping6Location	no		/usr/sbin/fping6	Location of fping6. Make sure that fping6 binary has root ownership and SUID flag set. Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses.

FpingLocation	否		usr/sbin/fping	ping 程序的路径。 确保 fping 程序的所有者是 root 用户，并且设置了 SUID 标记。
FpingLocation	no		/usr/sbin/fping	Location of fping. Make sure that fping binary has root ownership and SUID flag set!

HistoryCacheSize	否	28K-2G	6M	史缓存数据大小, 单位为字节。

HistoryCacheSize	no	128K-2G	16M	Size of history cache, in bytes. Shared memory size for storing history data.

HistoryIndexCacheSize	否	28K-2G	M	史索引缓存大小, 单位为字节。\\缓存一个 item 大概需要大小为 100 字节的空间。 该参数从 Zabbix 3.0.0 开始支持。
HistoryIndexCacheSize	no	128K-2G	4M	Size of history index cache, in bytes. Shared memory size for indexing history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item. This parameter is supported since Zabbix 3.0.0.

HistoryStorageDateIndex	否			用历史数据预处理，可以将数据存储到不同的基于时间的索引: 0 - 禁止 1 - 允许
HistoryStorageDateIndex	no		0	Enable preprocessing of history values in history storage to store values in different indices based on date: 0 - disable 1 - enable

HistoryStorageURL	否		史数据存储 HTTP[S] URL。 这个参数参考Elasticsearch进行配置。	
HistoryStorageURL	no			History storage HTTP[S] URL. This parameter is used for Elasticsearch setup.

HistoryStorageTypes	否	int,dbl,str,log,text	逗号分隔的列表配置哪些类型的历史数据需要存储到 Elasticsearch。这个参数参考Elasticsearch 进行配置。
HistoryStorageTypes	no	uint,dbl,str,log,text	Comma separated list of value types to be sent to the history storage. This parameter is used for Elasticsearch setup.

HousekeepingFrequency	否	-24	abbix 执行 housekeeping 的频率 (单位为小时)。 housekeeping 负责从数据库中删除过期的信息。 注意: 为了防止 housekeeper 负载过大 (例如, 当历史和趋势周期大大减小时), 对于每一个监控项，不会在一个 housekeeping 周期内删除超过 4 倍 HousekeepingFrequency 的过期数据。因此, 如果 HousekeepingFrequency 是 1 小时, 一个周期内不会删除超过 4 小时的过期信息 (从最旧的数据开始)。 备注: 为降低 server 压力，housekeeping 将在 server 启动以后，延迟 30 分钟执行。因此, 如果 HousekeepingFrequency 是 1 小时,serverg 启动 30 分后执行第一次 housekeeping，然后按 1 小时为周期重复执行。从 Zabbix 2.4.0 以后有了这种延迟行为。从 Zabbix 3.0.0 开始，可以设置 HousekeepingFrequency 为 0 来禁止自动 housekeeping。此时 housekeeping 只能通过 *housekeeper_execute* 启动，在一个 housuekeeping 周期内删除的过期信息时长为从最后一次 housekeeping 以来到配置周期的 4 倍, 不少于 4 小时且不大于 4 天。 也可参见运行控制 选项。

HousekeepingFrequency	no	0-24	1

How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database.
Note: To prevent housekeeper from being overloaded (for example, when history and trend periods are greatly reduced), no more than 4 times HousekeepingFrequency hours of outdated information are deleted in one housekeeping cycle, for each item. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated information (starting from the oldest entry) will be deleted per cycle.
Note: To lower load on server startup housekeeping is postponed for 30 minutes after server start. Thus, if HousekeepingFrequency is 1, the very first housekeeping procedure after server start will run after 30 minutes, and will repeat with one hour delay thereafter. This postponing behavior is in place since Zabbix 2.4.0.
Since Zabbix 3.0.0 it is possible to disable automatic housekeeping by setting HousekeepingFrequency to 0. In this case the housekeeping procedure can only be started by *housekeeper_execute* runtime control option and the period of outdated information deleted in one housekeeping cycle is 4 times the period since the last housekeeping cycle, but not less than 4

Include	否			以在配置文件中指定单个文件或者指定一个目录（所有文件在该目录中）。 只有在指定的目录中包含相关文件, 才可以使用正则匹配的通配符。 例如: `/absolute/path/to/config/fil` Zabbix 2.4.0 以后都支持模式匹配。 参看关于限制条件特例.
Include	no			You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `/absolute/path/to/config/fil` Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.

JavaGateway	否			abbix Java 网关的 IP 地址 (或主机名) 。 Java 轮询器启动时才需要该参数。 Zabbix 2.0.0 以后的所有版本都支持该参数.
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started. This parameter is supported since Zabbix 2.0.0.

JavaGatewayPort	否	024-32767	0052	abbix Java 网关监听端口。 Zabbix 2.0.0 以后的所有版本都支持该参数。
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on. This parameter is supported since Zabbix 2.0.0.

ListenIP	否	.0.0.0	rapper 监听的 Ip 地址，多个 Ip 用逗号分开。 如果没有设置该参数，会监听所有网络接口。 从 Zabbix 1.8.3 开始支持多 Ip 地址。
ListenIP	no	0.0.0.0	List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing. Multiple IP addresses are supported since Zabbix 1.8.3.

ListenPort	否	024-32767	0051	rapper 监听端口。
ListenPort	no	1024-32767	10051	Listen port for trapper.

LoadModule	否	erver 端启动时加载的模块，这些模块用来扩展 server 的功能。 格式: LoadModule=<module.so> 这些模块必须在 LoadModulePath 参数指定的路径中。 允许多个 LoadModule 参数。
LoadModule	no	Module to load at server startup. Modules are used to extend functionality of the server. Format: LoadModule=<module.so> The modules must be located in directory specified by LoadModulePath. It is allowed to include multiple LoadModule parameters.

LoadModulePath	否	erver 模块的绝对路径。默认值在编译时指定。
LoadModulePath	no	Full path to location of server modules. Default depends on compilation options.

LogFile	是, 如果 LogType 设置为 *file*, 否则为 日志否	件名	。	
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.

LogFileSize	否	-1024		志文件大小，单位 MB。 0 - 禁止日志文件自动回滚. 注意: 如果日志文件达到限定的大小，文件回滚失败, 不管是什么原因, 现有的日志会被截断，并重新记录日志。
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. *Note*: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

LogType	否		ile	志输出类型: *file* - 写入 LogFile 参数指定的日志文件中, *system* - 写入 syslog, *console* - 控制台输出. 从 Zabbix 3.0.0 开始支持该参数。
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log to syslog, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.

LogSlowQueries	否	-3600000		据库查询消耗时间，大于该时间将会记入日志(毫秒)。 0 - 不记录慢查询日志。 DebugLevel=3 时该选项可用。 从 Zabbix 1.8.2 开始支持该参数

LogSlowQueries	no	0-3600000	0	How long a database query may take before being logged (in milliseconds). 0 - don't log slow queries. This option becomes enabled starting with DebugLevel=3. This parameter is supported since Zabbix 1.8.2.

MaxHousekeeperDelete	否	-1000000	000	个 housekeeping 周期内，一个任务删除的最大行数（相应的表名，字段名，值）。 如果设置为 0，不限制删除的行数，这种情况，你必须清楚这样做的影响! 从 Zabbix 1.8.2 开始支持该参数，仅在对已经被删除的监控项进行历史和趋势数据删除操作时有效。
MaxHousekeeperDelete	no	0-1000000	5000	No more than 'MaxHousekeeperDelete' rows (corresponding to [tablename], [field], [value]) will be deleted per one task in one housekeeping cycle. If set to 0 then no limit is used at all. In this case you must know what you are doing! This parameter is supported since Zabbix 1.8.2 and applies only to deleting history and trends of already deleted items.

PidFile	否	tmp/zabbix_server.pid	ID 文件名称。
PidFile	no	/tmp/zabbix_server.pid	Name of PID file.

ProxyConfigFrequency	否	-604800	600	abbix server 多少秒向 Zabbix proxy 发送一次配置数据，用于被动模式的 proxy 。 从 Zabbix 1.8.3 开始支持该参数。

ProxyConfigFrequency	no	1-604800	3600	How often Zabbix server sends configuration data to a Zabbix proxy in seconds. Used only for proxies in a passive mode. This parameter is supported since Zabbix 1.8.3.

ProxyDataFrequency	否	-3600		abbix server 多少秒向 Zabbix proxy 请求一次历史数据，用于被动模式的 proxy。\\从 Zabbix 1.8.3 开始支持该参数。
ProxyDataFrequency	no	1-3600	1	How often Zabbix server requests history data from a Zabbix proxy in seconds. Used only for proxies in a passive mode. This parameter is supported since Zabbix 1.8.3.

SNMPTrapperFile	否		tmp/zabbix_traps.tmp	时文件，用于传递 SNMP trap 守护进程的数据给 server. 必须和 zabbix_trap_receiver.pl 或 SNMPTT 配置文件中的配置保持一致。 从 Zabbix 2.0.0 开始支持该参数。
SNMPTrapperFile	no		/tmp/zabbix_traps.tmp	Temporary file used for passing data from SNMP trap daemon to the server. Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file. This parameter is supported since Zabbix 2.0.0.

SocketDir	否		tmp	abbix 内部服务使用的用于存储 IPC sockets 的目录。从 Zabbix 3.4.0 开始支持该参数。

SocketDir	no	/tmp	Directory to store IPC sockets used by internal Zabbix services. This parameter is supported since Zabbix 3.4.0.

SourceIP	否	外连接的源 IP 地址。
SourceIP	no	Source IP address for outgoing connections.

SSHKeyLocation	no	SSH 检查和操作的公钥和私钥的位置。
SSHKeyLocation	no	Location of public and private keys for SSH checks and actions

SSLCertLocation	否	于客户端身份验证的 SSL 证书文件的位置。该参数只用于 web 监控，从 Zabbix 2.4 开始支持该参数。
SSLCertLocation	no	Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.

SSLKeyLocation	否	于客户端身份验证的 SSL 私钥文件的位置。该参数只用于 web 监控，从 Zabbix 2.4 开始支持该参数。
SSLKeyLocation	no	Location of SSL private key files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.

SSLCALocation	否			SSL 服务器证书验证覆盖证书颁发机构 (CA) 文件的位置。如果不设置，系统范围的目录将被使用。 注意，这个参数的值将被设置为 libcurl 选项 curloptcapath，在 7.42.0 之前的 libcurl 版本中，只有使用 OpenSSL 编译 libcurl 才会有效。更多信息见 cURL 网页. 这个参数从 Zabbix 2.4.0 开始的 web 监控和自从 Zabbix 3.0.0 开始的 SMTP 身份验证中使用。
SSLCALocation	no			Override the location of certificate authority (CA) files for SSL server certificate verification. If not set, system-wide directory will be used. Note that the value of this parameter will be set as libcurl option CURLOPT_CAPATH. For libcurl versions before 7.42.0, this only has effect if libcurl was compiled to use OpenSSL. For more information see cURL web page. This parameter is used in web monitoring since Zabbix 2.4.0 and in SMTP authentication since Zabbix 3.0.0.

StartDBSyncers	否	-100		据库进程的初始实例数量。 在版本 1.8.5 之前，上限是 64。 这个参数从 Zabbix 1.8.3 开始得到了支持。
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.

StartAlerters	否	-100		警进程的初始实例数量。 从 Zabbix 3.4.0 开始支持该参数。
StartAlerters	no	1-100	3	Number of pre-forked instances of alerters. This parameter is supported since Zabbix 3.4.0.
StartDiscoverers	否	-250		现进程的初始实例数量。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers. The upper limit used to be 255 before version 1.8.5.
StartEscalators	否	-100		scalators 进程的初始实例数量。 从 Zabbix 3.0.0 开始支持该参数。
StartEscalators	no	1-100	1	Number of pre-forked instances of escalators. This parameter is supported since Zabbix 3.0.0.
StartHTTPPollers	否	-1000		TTP 轮询进程的初始实例数量。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers[1]. The upper limit used to be 255 before version 1.8.5.
StartIPMIPollers	否	-1000		PMI 轮询进程的初始实例数量。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartIPMIPollers	no	0-1000	0	Number of pre-forked instances of IPMI pollers. The upper limit used to be 255 before version 1.8.5.

StartJavaPollers	否	-1000		ava 轮询子进程的初始实例数量。 从 Zabbix 2.0.0 开始支持该参数。
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers[1]. This parameter is supported since Zabbix 2.0.0.

StartPingers	否	-1000		CMP pingers 进程的初始实例数量 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers[1]. The upper limit used to be 255 before version 1.8.5.

StartPollersUnreachable	否	-1000		可达主机 (包括 IPMI 和 Java) 的轮询进程的初始实例数量。 从 Zabbix 2.4.0 开始，如果 IPMI 或 Java 轮询器启动，那么至少有一个针对不可访问主机的轮询进程必须运行。 \\在 Zabbix 1.8.5 版本之前，最大能设置为 255。 这个参数从 Zabbix 1.8.3 开始得到了支持。
StartPollersUnreachable	no	0-1000	1	Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java)[1]. Since Zabbix 2.4.0, at least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started. The upper limit used to be 255 before version 1.8.5. This option is missing in version 1.8.3.

StartPollers	否	-1000		询进程的初始实例数量。\\注意如果要内部，聚合，计算的监控项能正常工作，这个参数值必须大于 0。
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers[1]. *Note* that a non-zero value is required for internal, aggregated and calculated items to work.

StartPreprocessors	否	-1000		处理工作进程的初始实例数量。\\预处理管理进程将跟随预处理工作进程启动。 从 Zabbix 3.4.0 开始支持该参数。
StartPreprocessors	no	1-1000	3	Number of pre-forked instances of preprocessing workers[1]. The preprocessing manager process is automatically started when a preprocessor worker is started. This parameter is supported since Zabbix 3.4.0.

StartProxyPollers	否	-250		动 proxy 的轮询进程初始实例数量。\\在 Zabbix 1.8.5 版本之前，最大能设置为 255。 从 Zabbix 1.8.3 开始支持该参数。
StartProxyPollers	no	0-250	1	Number of pre-forked instances of pollers for passive proxies[1]. The upper limit used to be 255 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.

StartSNMPTrapper	否	-1		置为 1, SNMP trapper 进程将启动。 从 Zabbix 2.0.0 开始支持该参数。

StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started. This parameter is supported since Zabbix 2.0.0.

StartTimers	否	-1000		时器进程的初始实例数量。 计时器进程处理基于时间的触发器和维护期功能。 只有第一个计时器进程处理维护期。 从 Zabbix 2.2.0 开始支持该参数。
StartTimers	no	1-1000	1	Number of pre-forked instances of timers. Timers process time-based trigger functions and maintenance periods. Only the first timer process handles the maintenance periods. This parameter is supported since Zabbix 2.2.0.

StartTrappers	否	-1000		rapper 进程的初始实例数量。 Trapper 接收来自 Zabbix 发送者、主动 agent 和主动 proxies 的数据。 至少要运行一个 trapper 进程用于在 web 前端展示服务器可用性和队列视图。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers[1]. Trappers accept incoming connections from Zabbix sender, active agents and active proxies. At least one trapper process must be running to display server availability and view queue in the frontend. The upper limit used to be 255 before version 1.8.5.

StartVMwareCollectors	否	-250		mware 采集器进程的初始实例数量。\\从 Zabbix 2.2.0 开始支持该参数。
StartVMwareCollectors	no	0-250	0	Number of pre-forked vmware collector instances. This parameter is supported since Zabbix 2.2.0.

Timeout	否	-30		gent, SNMP 设备或外部检查的超时时长 (单位为秒)。
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).

TLSCAFile	否			含用于对等证书验证的顶级 CA（s）证书的文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSCertFile	否			含服务器证书或证书链文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSCertFile	no			Full pathname of a file containing the server certificate or certificate chain, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSCRLFile	否		含已吊销证书文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSCRLFile	no		Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSKeyFile	否		含私钥文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSKeyFile	no		Full pathname of a file containing the server private key, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TmpDir	否	tmp	时目录。
TmpDir	no	/tmp	Temporary directory.

TrapperTimeout	否	-300	00	义 trapper 处理数据的超时时间。
TrapperTimeout	no	1-300	300	Specifies how many seconds trapper may spend processing new data.

TrendCacheSize	否	28K-2G	M	势数据缓存大小，单位字节。 用于存储趋势数据的共享内存大小。
TrendCacheSize	no	128K-2G	4M	Size of trend cache, in bytes. Shared memory size for storing trends data.

UnavailableDelay	否	-3600	0	资源不可用期间，Zabbix 多少秒检查一次资源是否可用。
UnavailableDelay	no	1-3600	60	How often host is checked for availability during the unavailability period, in seconds.

UnreachableDelay	否	-3600	5	资源不可达期间，Zabbix 多少秒检查一次资源是否可达。
UnreachableDelay	no	1-3600	15	How often host is checked for availability during the unreachability period, in seconds.

UnreachablePeriod	否	-3600	5	主机不可用多少秒后，即视为主机不可用。
UnreachablePeriod	no	1-3600	45	After how many seconds of unreachability treat a host as unavailable.

User	否		abbix	了降低权限使用普通用户。 仅当以’root’ 身份运行且 AllowRoot 参数设置为禁止时，该参数才起作用。 从 Zabbix 2.4.0 开始支持该参数。
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as ’root’ and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.

ValueCacheSize	否	,128K-64G	M	史数据缓存大小, 单位为字节。 0 即禁止缓存 (不建议). 当缓存大小超过共享内存时，每 5 分钟会向服务器日志写入一条警告信息。 从 Zabbix 2.2.0 开始支持该参数。

ValueCacheSize	no	0,128K-64G	8M	Size of history value cache, in bytes. Shared memory size for caching item history data requests. Setting to 0 disables value cache (not recommended). When value cache runs out of the shared memory a warning message is written to the server log every 5 minutes. This parameter is supported since Zabbix 2.2.0.

VMwareCacheSize	否	56K-2G	M	储 VMware 数据的共享内存大小。 VMware 内部检查 [vmware,buffer,...] 可以用来监控 VMware 缓存使用情况 (参见内部检查)。 注意，如果没有配置并启动 vmware 收集器实例，那么共享内存就不会被分配。\\从 Zabbix 2.2.0 开始支持该参数。
VMwareCacheSize	no	256K-2G	8M	Shared memory size for storing VMware data. A VMware internal check zabbix[vmware,buffer,...] can be used to monitor the VMware cache usage (see Internal checks). Note that shared memory is not allocated if there are no vmware collector instances configured to start. This parameter is supported since Zabbix 2.2.0.

VMwareFrequency	否	0-86400	0	隔多少秒从单个 VMware 服务收集数据。\\任何 VMware 监控项的最小更新周期都大于或等于该时间。 从 Zabbix 2.2.0 开始支持该参数。

VMwareFrequency	no	10-86400	60	Delay in seconds between data gathering from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item. This parameter is supported since Zabbix 2.2.0.

VMwarePerfFrequency	否	0-86400	0	隔多少秒从单个 VMware 服务检索性能计数器统计数据。 该时间为任一 VMware 监控项（使用 VMware 性能计数器）的最小更新间隔。 从 Zabbix 2.2.9, 2.4.4 开始支持该参数。
VMwarePerfFrequency	no	10-86400	60	Delay in seconds between performance counter statistics retrieval from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item that uses VMware performance counters. This parameter is supported since Zabbix 2.2.9, 2.4.4

VMwareTimeout	否	-300	0	mware 采集器等待 VMware 服务 (vCenter or ESX 管理程序) 响应的最大时长。 从 Zabbix 2.2.9, 2.4.4 开始支持该参数。
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor). This parameter is supported since Zabbix 2.2.9, 2.4.4

附注

Footnotes

Warning:
(1) 请注意，大量的数据收集进程（pollers, unreachable pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers）加上 IPMI 管理程序，SNMP trapper 和预处理程序可能使预处理管理器的进程超过其文件描述符的限制。这将会引起 Zabbix server 停止服务（通常启动后很快会出现，但有时候可能需要很长时间才出现）。应修改 Zabbix 配置文件或提高进程文件描述符限制来避免这种情况。

Warning:
(1) Note that too many data gathering processes (pollers, unreachable pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers) together with IPMI manager, SNMP trapper and preprocessing workers can exhaust the per-process file descriptor limit for the preprocessing manager. This will cause Zabbix server to stop (usually shortly after the start, but sometimes it can take more time). The configuration file should be revised or the limit should be raised to avoid this situation.

Note:
Zabbix 配置文件仅仅支持 UTF-8 编码但不支持BOM。
注释信息必须在行的开头使用”#” 来标识。

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

Comments starting with ”#” are only supported in the beginning of the line.

2 Zabbix proxy

2 Zabbix proxy

Note:
本节中参数的默认值只是代表守护进程的默认使用值，实际运行时应以你所使用的配置文件中定义的参数值为准。

Note:
The default values reflect daemon defaults, not the values in the shipped configuration files.

以下参数可以在 Zabbix proxy 配置文件中配置：The parameters supported in a Zabbix proxy configuration file:

参数名称必须配	范围默	值描述	息	
Parameter	Mandatory	Range	Default	Description
AllowRoot	否			许服务以’root’ 身份运行。如果该参数配置为禁止，并且服务仍以 root 身份启动，服务会切换到使用’zabbix’ 用户启动。对于以普通用户启动的，该参数没有影响。 0 - 禁止 1 - 允许 Zabbix 2.2.0。以后的版本都支持这个参数

AllowRoot	no		0	Allow the proxy to run as 'root'. If disabled and the proxy is started by 'root', the proxy will try to switch to the 'zabbix' user instead. Has no effect if started under a regular user. 0 - do not allow 1 - allow This parameter is supported since Zabbix 2.2.0.

CacheSize	否	28K-8G	M	存大小, 单位为字节。用于存储主机、监控项数据的共享内存大小。Zabbix2.2.3 以前的版本最大可配置值为 2GB。
CacheSize	no	128K-8G	8M	Size of configuration cache, in bytes. Shared memory size, for storing host and item data. Upper limit used to be 2GB before Zabbix 2.2.3.

ConfigFrequency	否	-604800	600	隔多少秒 proxy 从 Zabbix server 获取配置数据。 该参数只有主动 proxy 才会使用，proxy 工作模式由参数 ProxyMode 决定。
ConfigFrequency	no	1-604800	3600	How often proxy retrieves configuration data from Zabbix server in seconds. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

DBHost	否		ocalhost	据库主机名。 如果是 MySQL localhost 或空字符串会导致使用套接字。如果是 PostgreSQL 只有空字符串会使用套接字。

DBHost	no	localhost	Database host name. In case of MySQL localhost or empty string results in using a socket. In case of PostgreSQL only empty string results in attempt to use socket.

DBName	是		据库名称。 对于 SQLite3 必须提供数据库文件路径 (Zabbix 的多进程架构不允许使用内存数据库，例如 `:memory:`, `file::memory:?cache=shared` 或 `file:memdb1?mode=memory&cache=shared`)。 警告: 不要与 Zabbix server 使用同一个数据库。
DBName	yes		Database name or path to database file for SQLite3 (multi-process architecture of Zabbix does not allow to use in-memory database, e.g. `:memory:`, `file::memory:?cache=shared` or `file:memdb1?mode=memory&cach` Warning: Do not attempt to use the same database Zabbix server is using.

DBPassword	否		据库登录密码。此参数 SQLite 不使用。 如果数据库没有密码，请注释掉此参数。
DBPassword	no		Database password. Ignored for SQLite. Comment this line if no password is used.

DBSchema	否	据库 Schema 名字。仅 IBM DB2 和 PostgreSQL 使用。
DBSchema	no	Schema name. Used for IBM DB2 and PostgreSQL.

DBSocket	否		306	ySQL 套接字文件的路径。 本地套接字链接时不使用数据库端口参数。此参数 SQLite 不使用。
DBSocket	no		3306	Path to MySQL socket. Database port when not using local socket. Ignored for SQLite.

DBUser	否		据库用户名。此参数 SQLite 不使用。
DBUser			Database user. Ignored for SQLite.

DataSenderFrequency	否	-3600		roxy 将采集到的数据以一定的时间间隔（单位为秒）发送给 Zabbix server。 该参数只有主动 proxy 才会使用，proxy 工作模式由参数 ProxyMode 决定。
DataSenderFrequency	no	1-3600	1	Proxy will send collected data to the server every N seconds. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

DebugLevel	否	-5		定调试等级: 0 - Zabbix 进程的起停基本信息 1 - 重要信息 2 - 错误信息 3 - 警告信息 4 - 调试信息 (产生大量信息) 5 - 扩展调试 (产生更多信息)
DebugLevel	no	0-5	3	Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)

EnableRemoteCommands	否		否允许 Zabbix server 远程执行命令。 0 - 禁止 1 - 允许 从 Zabbix 3.4.0 开始支持该参数。
EnableRemoteCommands	no	0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed This parameter is supported since Zabbix 3.4.0.

ExternalScripts	否	usr/local/share/zabbix/externalscripts	脚本位置（依赖编译安装时的环境变量 *datadir*）。
ExternalScripts	no	/usr/local/share/zabbix/externalscripts	Location of external scripts (depends on compile-time installation variable *datadir*).

Fping6Location	否	usr/sbin/fping6	ping6 程序的路径。 确保 fping6 程序的所有者是 root 用户，并且设置了 SUID 标记。 如果需要 fping 程序处理 IPv6 地址，就置空（”Fping6Location=”）参数。
Fping6Location	no	/usr/sbin/fping6	Location of fping6. Make sure that fping6 binary has root ownership and SUID flag set. Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses.

FpingLocation	否	usr/sbin/fping	ping 程序的路径。 确保 fping 程序的所有者是 root 用户，并且设置了 SUID 标记。
FpingLocation	no	/usr/sbin/fping	Location of fping. Make sure that fping binary has root ownership and SUID flag set!

HeartbeatFrequency	否	-3600	0	跳信息发送频率，单位为秒。 用于监视 proxy 的可用性。 0 - 禁止 该参数只有主动 proxy 才会使用，proxy 工作模式由参数 ProxyMode 决定。
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring availability of proxy on server side. 0 - heartbeat messages disabled. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

HistoryCacheSize	否	28K-2G	6M	史缓存数据大小, 单位为字节。
HistoryCacheSize	no	128K-2G	16M	Size of history cache, in bytes. Shared memory size for storing history data.

HistoryIndexCacheSize	否	28K-2G	M	史索引缓存大小, 单位为字节。\\缓存一个 item 大概需要大小为 100 字节的空间。 该参数从 Zabbix 3.0.0 开始支持。
HistoryIndexCacheSize	no	128K-2G	4M	Size of history index cache, in bytes. Shared memory size for indexing history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item. This parameter is supported since Zabbix 3.0.0.

Hostname	否		参数 HostnameItem 设置唯一的大小	敏感的 Proxy 名称。确保 Server 端能正确解析这个 proxy 名称。 允许的字符: 字母, '.', ' ', '_' 和'-'. 最大长度: 64

Hostname	no	Set by HostnameItem	Unique, case sensitive Proxy name. Make sure the proxy name is known to the server! Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 64

HostnameItem	否	ystem.hostname	参数 Hostname 没有定义时使用这个参数设置主机名。 该参数不能用于 UserParameters, performance counters or aliases, 但能用于 system.run[]。 从 Zabbix 1.8.6 开始支持该参数。
HostnameItem	no	system.hostname	Item used for setting Hostname if it is undefined (this will be run on the proxy similarly as on an agent). Does not support UserParameters, performance counters or aliases, but does support system.run[]. Ignored if Hostname is set. This parameter is supported since Zabbix 1.8.6.

HousekeepingFrequency	否	-24	abbix 执行 housekeeping 的频率 (单位为小时)。 housekeeping 负责从数据库中删除过期的信息。 注意: 为了防止 housekeeper 负载过大 (例如, 当历史和趋势周期大大减小时), 对于每一个监控项，不会在一个 housekeeping 周期内删除超过 4 倍 HousekeepingFrequency 的过期数据。因此, 如果 HousekeepingFrequency 是 1 小时, 一个周期内不会删除超过 4 小时的过期信息 (从最旧的数据开始)。 备注: 为降低 server 压力，housekeeping 将在 server 启动以后，延迟 30 分钟执行。因此, 如果 HousekeepingFrequency 是 1 小时,serverg 启动 30 分后执行第一次 housekeeping，然后按 1 小时为周期重复执行。从 Zabbix 2.4.0 以后有了这种延迟行为。从 Zabbix 3.0.0 开始，可以设置 HousekeepingFrequency 为 0 来禁止自动 housekeeping。此时 housekeeping 只能通过 *housekeeper_execute* 启动，在一个 housuekeeping 周期内删除的过期信息时长为从最后一次 housekeeping 以来到配置周期的 4 倍, 不少于 4 小时且不大于 4 天。

HousekeepingFrequency	no	0-24	1

How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database.
Note: To prevent housekeeper from being overloaded (for example, when configuration parameters ProxyLocalBuffer or ProxyOfflineBuffer are greatly reduced), no more than 4 times HousekeepingFrequency hours of outdated information are deleted in one housekeeping cycle. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated information (starting from the oldest entry) will be deleted per cycle.
Note: To lower load on proxy startup housekeeping is postponed for 30 minutes after proxy start. Thus, if HousekeepingFrequency is 1, the very first housekeeping procedure after proxy start will run after 30 minutes, and will repeat every hour thereafter. This postponing behavior is in place since Zabbix 2.4.0.
Since Zabbix 3.0.0 it is possible to disable automatic housekeeping by setting HousekeepingFrequency to 0. In this case the housekeeping procedure can only be started by *housekeeper_execute* runtime control option and the period of outdated information deleted in one housekeeping cycle is 4 times the period since the last housekeeping cycle,

Include	否			以在配置文件中指定单个文件或者指定一个目录（所有文件在该目录中）。 只有在指定的目录中包含相关文件, 才可以使用正则匹配的通配符。 例如: `/absolute/path/to/config/fil` Zabbix 2.4.0 以后都支持模式匹配。 参看关于限制条件特例。
Include	no			You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `/absolute/path/to/config/fil` Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.

JavaGateway	否			abbix Java 网关的 IP 地址 (或主机名) 。 Java 轮询器启动时才需要该参数。 Zabbix 2.0.0 以后的所有版本都支持该参数.
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started. This parameter is supported since Zabbix 2.0.0.

JavaGatewayPort	否	024-32767	0052	abbix Java 网关监听端口。 Zabbix 2.0.0 以后的所有版本都支持该参数。
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on. This parameter is supported since Zabbix 2.0.0.

ListenIP	否	.0.0.0	rapper 监听的 Ip 地址，多个 Ip 用逗号分开。 如果没有设置该参数，会监听所有网络接口。 从 Zabbix 1.8.3 开始支持多 Ip 地址。
ListenIP	no	0.0.0.0	List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing. Multiple IP addresses are supported since Zabbix 1.8.3.

ListenPort	否	024-32767	0051	rapper 监听端口。
ListenPort	no	1024-32767	10051	Listen port for trapper.

LoadModule	否	erver 端启动时加载的模块，这些模块用来扩展 server 的功能。 格式: LoadModule=<module.so> 这些模块必须在 LoadModulePath 参数指定的路径中。 允许多个 LoadModule 参数。
LoadModule	no	Module to load at proxy startup. Modules are used to extend functionality of the proxy. Format: LoadModule=<module.so> The modules must be located in directory specified by LoadModulePath. It is allowed to include multiple LoadModule parameters.

LoadModulePath	否	erver 模块的绝对路径。 默认值在编译时指定。
LoadModulePath	no	Full path to location of proxy modules. Default depends on compilation options.

LogFile LogFile	是, 如果 LogType 设置为 *file*, 否则为 日志否 yes, if LogType is set to *file*, otherwise no	件名	。	Name of log file.

LogFileSize	否	-1024		志文件大小，单位 MB。 0 - 禁止日志文件自动回滚. 注意: 如果日志文件达到限定的大小，文件回滚失败, 不管是什么原因, 现有的日志会被截断，并重新记录日志。
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. *Note*: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

LogRemoteCommands	否			执行 shell 命令时可以记录日志。 0 - 禁止 1 - 允许 从 Zabbix 3.4.0 开始支持该参数。
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled This parameter is supported since Zabbix 3.4.0.

LogType	否		ile	志输出类型: *file* - 写入 LogFile 参数指定的日志文件中, *system* - 写入 syslog, *console* - 控制台输出. 从 Zabbix 3.0.0 开始支持该参数。

LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log to syslog, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.

LogSlowQueries	否	-3600000		据库查询消耗时间，大于该时间将会记入日志(毫秒)。 0 - 不记录慢查询日志。 DebugLevel=3 时该选项可用。 从 Zabbix 1.8.2 开始支持该参数
LogSlowQueries	no	0-3600000	0	How long a database query may take before being logged (in milliseconds). 0 - don’t log slow queries. This option becomes enabled starting with DebugLevel=3. This parameter is supported since Zabbix 1.8.2.

PidFile	否	tmp/zabbix_server.pid	ID 文件名称。	
PidFile	no	/tmp/zabbix_proxy.pid	Name of PID file.	

ProxyLocalBuffer	否	-720		roxy 将在本地保留数据 N 小时，即使数据已与 server 同步。\\如果第三方应用程序将使用本地数据，则可以使用此参数。
ProxyLocalBuffer	no	0-720	0	Proxy will keep data locally for N hours, even if the data have already been synced with the server. This parameter may be used if local data will be used by third party applications.

ProxyMode	否	-1		roxy 工作模式。 0 - 主动模式 1 - 被动模式 从 Zabbix 1.8.3 开始支持该参数。 注意当使用 Active proxy 时，敏感的 proxy 配置数据可供有权访问 Zabbix server trapper 端口的应用使用。因为第三方应用可以假装是活动 proxy 并请求配置数据而不会进行身份验证。
ProxyMode	no	0-1	0	Proxy operating mode. 0 - proxy in the active mode 1 - proxy in the passive mode This parameter is supported since Zabbix 1.8.3. *Note* that (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data; authentication does not take place.

ProxyOfflineBuffer	否	-720		果无法连接 Zabbix server，proxy 将保留数据 N 小时。\\旧数据将丢失。
ProxyOfflineBuffer	no	1-720	1	Proxy will keep data for N hours in case of no connectivity with Zabbix server. Older data will be lost.

ServerPort	否	024-32767	0051	abbix server 上 trapper 使用的端口. 该参数只有主动 proxy 才会使用，proxy 工作模式由参数 ProxyMode 决定。

ServerPort	no	1024-32767	10051	Port of Zabbix trapper on Zabbix server. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

Server	是			ProxyMode 参数设置为 主动模式:\\可以通过 Zabbix server 的 IP 地址或 DNS 名称获取配置数据并将数据发送给 Zabbix server。 当 ProxyMode 参数设置为被动模式:\\逗号分隔的 IP 地址列表，可选以 CIDR 表示法或 Zabbix server 的 DNS 名称。仅接受来自此处配置地址的传入连接。如果启用了 IPv6 支持，则 “127.0.0.1”，“:: 127.0.0.1”，“:: ffff：127.0.0.1” 将被同等对待，“:: / 0” 将允许任何 IPv4 或 IPv6 地址。’0.0.0.0/0’ 可用于允许任何 IPv4 地址。 示例：Server = 127.0.0.1,192.168.1.0 / 24，:: 1,2001：db8 :: / 32，zabbix.example.com

Server	yes		If ProxyMode is set to *active mode*: IP address or DNS name of Zabbix server to get configuration data from and send data to. If ProxyMode is set to *passive mode*: List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix server. Incoming connections will be accepted only from the addresses listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. *Example*: Server=127.0.0.1,192.168.1.0/24,::1

SNMPTrapperFile	否	tmp/zabbix_traps.tmp	时文件，用于传递 SNMP trap 守护进程的数据给 server. 必须和 zabbix_trap_receiver.pl 或 SNMPTT 配置文件中的配置保持一致。 从 Zabbix 2.0.0 开始支持该参数。
SNMPTrapperFile	no	/tmp/zabbix_traps.tmp	Temporary file used for passing data from SNMP trap daemon to the proxy. Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file. This parameter is supported since Zabbix 2.0.0.

SocketDir	否	tmp	abbix 内部服务使用的用于存储 IPC sockets 的目录。从 Zabbix 3.4.0 开始支持该参数。
SocketDir	no	/tmp	Directory to store IPC sockets used by internal Zabbix services. This parameter is supported since Zabbix 3.4.0.

SourceIP	否	外连接的源 IP 地址。
SourceIP	no	Source IP address for outgoing connections.

SSHKeyLocation	no		SSH 检查和操作的公钥和私钥的位置。
SSHKeyLocation	no		Location of public and private keys for SSH checks and actions

SSLCertLocation	否		于客户端身份验证的 SSL 证书文件的位置。该参数只用于 web 监控，从 Zabbix 2.4 开始支持该参数。
SSLCertLocation	no		Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.0.

SSLKeyLocation	否		于客户端身份验证的 SSL 私钥文件的位置。该参数只用于 web 监控，从 Zabbix 2.4 开始支持该参数。
SSLKeyLocation	no		Location of SSL private key files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.0.

SSLCALocation	否			SSL 服务器证书验证覆盖证书颁发机构 (CA) 文件的位置。如果不设置，系统范围的目录将被使用。 注意，这个参数的值将被设置为 libcurl 选项 curloptcapath，在 7.42.0 之前的 libcurl 版本中，只有使用 OpenSSL 编译 libcurl 才会有效。更多信息见 cURL 网页. 这个参数从 Zabbix 2.4.0 开始的 web 监控和自从 Zabbix 3.0.0 开始的 SMTP 身份验证中使用。
SSLCALocation	no			Location of certificate authority (CA) files for SSL server certificate verification. Note that the value of this parameter will be set as libcurl option CURLOPT_CAPATH. For libcurl versions before 7.42.0, this only has effect if libcurl was compiled to use OpenSSL. For more information see cURL web page. This parameter is used in web monitoring since Zabbix 2.4.0 and in SMTP authentication since Zabbix 3.0.0.

StartDBSyncers	否	-100		据库进程的初始实例数量。 在版本 1.8.5 之前，上限是 64。 这个参数从 Zabbix 1.8.3 开始得到了支持。
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.

StartDiscoverers	否		-250		现进程的初始实例数量。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartDiscoverers	no		0-250	1	Number of pre-forked instances of discoverers. The upper limit used to be 255 before version 1.8.5.

StartHTTPPollers	否	-1000		TTP 轮询进程的初始实例数量。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers.

StartIPMIPollers	否		-1000		PMI 轮询进程的初始实例数量。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartIPMIPollers	no		0-1000	0	Number of pre-forked instances of IPMI pollers. The upper limit used to be 255 before version 1.8.5.

StartJavaPollers	否		-1000		ava 轮询子进程的初始实例数量。 从 Zabbix 2.0.0 开始支持该参数。
StartJavaPollers	no		0-1000	0	Number of pre-forked instances of Java pollers. This parameter is supported since Zabbix 2.0.0.

StartPingers	否		-1000		CMP pingers 进程的初始实例数量 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartPingers	no		0-1000	1	Number of pre-forked instances of ICMP pingers. The upper limit used to be 255 before version 1.8.5.

StartPollersUnreachable	否	-1000		可达主机 (包括 IPMI 和 Java) 的轮询进程的初始实例数量。 从 Zabbix 2.4.0 开始，如果 IPMI 或 Java 轮询器启动，那么至少有一个针对不可访问主机的轮询进程必须运行。 \\在 Zabbix 1.8.5 版本之前，最大能设置为 255。 这个参数从 Zabbix 1.8.3 开始得到了支持。
StartPollersUnreachable	no	0-1000	1	Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java). Since Zabbix 2.4.0, at least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started. The upper limit used to be 255 before version 1.8.5. This option is missing in version 1.8.3.

StartPollers	否	-1000		询进程的初始实例数量。\\在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers. The upper limit used to be 255 before version 1.8.5.

StartSNMPTrapper	否	-1		置为 1, SNMP trapper 进程将启动。 从 Zabbix 2.0.0 开始支持该参数。
StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started. This parameter is supported since Zabbix 2.0.0.

StartTrappers	否	-1000		rapper 进程的初始实例数量。 Trapper 接收来自 Zabbix 发送者、主动 agent 的数据。 至少要运行一个 trapper 进程用于在 web 前端展示服务器可用性和队列视图。 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers. Trappers accept incoming connections from Zabbix sender and active agents. The upper limit used to be 255 before version 1.8.5.

StartVMwareCollectors	否	-250		mware 采集器进程的初始实例数量。\\从 Zabbix 2.2.0 开始支持该参数。
StartVMwareCollectors	no	0-250	0	Number of pre-forked vmware collector instances. This parameter is supported since Zabbix 2.2.0.

Timeout	否	-30		gent, SNMP 设备或外部检查的超时时长 (单位为秒)。
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).

TLSAccept	是, 如果 TLS certificate 或 PSK 参数都进行了定义 (即使是未加密的连接), 否则为否 Zabbix server 能接受哪些	接方	。	参数仅用于被动 proxy。
				可以指定多个值，以逗号分隔：
				未加密接受无加密的连接（默认）
				// psk // - 接受与 TLS 的连接和预共享密钥（PSK）
				// cert // - 接受与 TLS 和证书的连接
				从 Zabbix 3.0.0 开始支持该参数。

TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no	What incoming connections to accept from Zabbix server. Used for a passive proxy, ignored on an active proxy. Multiple values can be specified, separated by comma: *unencrypted* - accept connections without encryption (default) *psk* - accept connections with TLS and a pre-shared key (PSK) *cert* - accept connections with TLS and a certificate This parameter is supported since Zabbix 3.0.0.

TLSCAFile	否	含用于对等证书验证的顶级 CA（s）证书的文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSCAFile	no	Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSCertFile	否	含服务器证书或证书链文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。

TLSCertFile	no			Full pathname of a file containing the proxy certificate or certificate chain, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSConnect	是, 如果 TLS certificate 或 PSK 参数进行了定义 (即使是未加密的连接), 否则为否该参数仅用于主动 proxy 连接 Za	bi	s	rver。仅可以选择一种方式:
				未加密接受无加密的连接（默认）
				// psk // - 接受与 TLS 的连接和预共享密钥（PSK）
				// cert // - 接受与 TLS 和证书的连接
				从 Zabbix 3.0.0 开始支持该参数。

TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			How the proxy should connect to Zabbix server. Used for an active proxy, ignored on a passive proxy. Only one value can be specified: *unencrypted* - connect without encryption (default) *psk* - connect using TLS and a pre-shared key (PSK) *cert* - connect using TLS and a certificate This parameter is supported since Zabbix 3.0.0.

TLSCRLFile	否			含已吊销证书文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSCRLFile	no			Full pathname of a file containing revoked certificates.This parameter is used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSKeyFile	否	含私钥文件的完整路径名，用于 Zabbix 组件之间的加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSKeyFile	no	Full pathname of a file containing the proxy private key, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

TLSPSKFile	否	含 proxy 与 Zabbix server 加密通信所使用的预共享密钥文件的完整路径名。 从 Zabbix 3.0.0 开始支持该参数。
TLSPSKFile	no	Full pathname of a file containing the proxy pre-shared key. used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.

TLSPSKIdentity	否	共享密钥身份字符串，用于与 Zabbix server 进行加密通信。 从 Zabbix 3.0.0 开始支持该参数。
TLSPSKIdentity	no	Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.

TLSServerCertIssuer	否	许 server 证书颁发者。 从 Zabbix 3.0.0 开始支持该参数。
TLSServerCertIssuer	no	Allowed server certificate issuer. This parameter is supported since Zabbix 3.0.0.

TLSServerCertSubject	否				许的 server 证书主题。 从 Zabbix 3.0.0 开始支持该参数。
TLSServerCertSubject	no				Allowed server certificate subject. This parameter is supported since Zabbix 3.0.0.

TmpDir	否	tmp	时目录。
TmpDir	no	/tmp	Temporary directory.

TrapperTimeout	否	-300	00	义 trapper 处理数据的超时时间。
TrapperTimeout	no	1-300	300	Specifies how many seconds trapper may spend processing new data.

User	否		abbix	了降低权限使用普通用户。 仅当以’root’ 身份运行且 AllowRoot 参数设置为禁止时，该参数才起作用。 从 Zabbix 2.4.0 开始支持该参数。
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as ’root’ and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.

UnavailableDelay	否	-3600	0	资源不可用期间，Zabbix 多少秒检查一次资源是否可用。
UnavailableDelay	no	1-3600	60	How often host is checked for availability during the unavailability period, in seconds.

UnreachableDelay	否	-3600	5	资源不可达期间，Zabbix 多少秒检查一次资源是否可达。
UnreachableDelay	no	1-3600	15	How often host is checked for availability during the unreachability period, in seconds.

UnreachablePeriod	否	-3600	5	主机不可用多少秒后，即视为主机不可用。
UnreachablePeriod	no	1-3600	45	After how many seconds of unreachability treat a host as unavailable.

VMwareCacheSize	否	56K-2G	M	储 VMware 数据的共享内存大小。 VMware 内部检查 [vmware,buffer,...] 可以用来监控 VMware 缓存使用情况 (参见内部检查)。 注意，如果没有配置并启动 vmware 收集器实例，那么共享内存就不会被分配。\\从 Zabbix 2.2.0 开始支持该参数。
VMwareCacheSize	no	256K-2G	8M	Shared memory size for storing VMware data. A VMware internal check zabbix[vmware,buffer,...] can be used to monitor the VMware cache usage (see Internal checks). Note that shared memory is not allocated if there are no vmware collector instances configured to start. This parameter is supported since Zabbix 2.2.0.

VMwareFrequency	否	0-86400	0	隔多少秒从单个 VMware 服务收集数据。\\任何 VMware 监控项的最小更新周期都大于或等于该时间。 从 Zabbix 2.2.0 开始支持该参数。

VMwareFrequency	no	10-86400	60	Delay in seconds between data gathering from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item. This parameter is supported since Zabbix 2.2.0.

VMwarePerfFrequency	否	0-86400	0	隔多少秒从单个 VMware 服务检索性能计数器统计数据。 该时间为任一 VMware 监控项（使用 VMware 性能计数器）的最小更新间隔。 从 Zabbix 2.2.9, 2.4.4 开始支持该参数。
VMwarePerfFrequency	no	10-86400	60	Delay in seconds between performance counter statistics retrieval from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item that uses VMware performance counters. This parameter is supported since Zabbix 2.2.9, 2.4.4

VMwareTimeout	否	-300	0	mware 采集器等待 VMware 服务 (vCenter or ESX 管理程序) 响应的最大时长。 从 Zabbix 2.2.9, 2.4.4 开始支持该参数。
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor). This parameter is supported since Zabbix 2.2.9, 2.4.4

Note:
Zabbix 配置文件仅仅支持 UTF-8 编码但不支持BOM。
注释信息必须在行的开头使用”#” 来标识。

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

Comments starting with "#" are only supported in the beginning of the line.

3 Zabbix agent (UNIX)

Note:
The default values reflect daemon defaults, not the values in the shipped configuration files. 默认值反映守护程序默认值，而不是写在配置文件中的值。

The parameters supported in a Zabbix agent configuration file (*zabbix_agentd.conf*): Zabbix agent 配置文件 (*zabbix_agentd.conf*) 支持的参数:

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
Alias	no			Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are allowed. Different *Alias* keys may reference the same item key. Aliases can be used in

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
AllowRoot	no		0	Allow the agent to run as 'root'. If disabled and the agent is started by 'root', the agent will try to switch to user 'zabbix' instead. Has no effect if started under a regular user. 0 - do not allow 1 - allow
BufferSend	no	1-3600	5	Do not keep data longer than N seconds in buffer.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
BufferSize	no	2-65535	100	Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
DebugLevel	no	0-5	3	Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
EnableRemoteCommands			0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used only at host autoregistration process (active agent). If not defined, the value will be acquired from HostMetadataItem. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string. This option is

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
HostMetadataItem	no			Optional parameter that defines a *Zabbix agent* item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports *system.run[]* regardless of *EnableRemoteCommands* value. Host metadata is used only at host auto-registration pro-

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
Hostname	no		Set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, ’.’, ’ ’, ’_’ and ’-’. Maximum length: 64

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
HostnameItem	no		system.hostname	Optional parameter that defines a *Zabbix agent* item used for getting host name. This option is only used when Hostname is not defined. Does not support UserParameters or aliases, but does support *system.run[]* regardless of *EnableRemoteCommands* value. This option is supported in ver-

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
Include	no			You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `/absolute/path/` Pattern matching is supported since

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
ListenIP	no		0.0.0.0	List of comma de-lim-ited IP ad-dresses that the agent should lis-ten on. Multiple IP ad-dresses are sup-ported in ver-sion 1.8.3 and higher.
ListenPort	no	1024-32767	10050	Agent will lis-ten on this port for con-nec-tions from the server.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
LoadModule	no			Module to load at agent startup. Modules are used to extend functionality of the agent. Format: LoadModule=<module.so> The modules must be located in directory specified by LoadModulePath. It is allowed to include multiple LoadModule parameters.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
LoadModulePath	no			Full path to location of agent modules. Default depends on compilation options.
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. *Note*: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log to syslog, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
LogRemoteCommands			0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
MaxLinesPerSecond		1-1000	20	Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' or 'eventlog' item key. *Note*: Zabbix will process 10 times more new

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
PidFile	no		/tmp/zabbix_agentd.pid	Name of PID file.
RefreshActiveChecks	no	60-3600	120	How often list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
Server	yes, if StartAgents is not explicitly set to 0			List of comma de-lim-ited IP ad-dresses, op-tion-ally in CIDR nota-tion, or host-names of Zab-bix servers and Zab-bix prox-ies. Incoming con-nec-tions will be ac-cepted only from the hosts listed here. If IPv6 sup-port is en-abled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 ad-

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
ServerActive	no			IP:port (or host-name:port) of Zab-bix server or Zab-bix proxy for ac-tive checks. Multiple comma-delimited ad-dresses can be pro-vided to use sev-eral inde-pen-dent Zab-bix servers in par-allel. No spaces al-lowed. If port is not spec-ified, de-fault port is used. IPv6 ad-dresses must be en-closed in square brack-ets if

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
SourceIP	no			Source IP address for outgoing connections.
StartAgents	no	0-100	3	Number of preforked instances of zabbix_agentd that process passive checks. If set to 0, disables passive checks and the agent will not listen on any TCP port. The upper limit used to be 16 before version 1.8.5.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
Timeout	no	1-30	3	Spend no more than Time-out sec-onds on pro-cess-ing

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma: *unencrypted* - accept connections without encryption (default) *psk* - accept connections with TLS and a pre-shared key (PSK) *cert* - accept connections with TLS and a

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSCAFile	no			Full path-name of a file con-tain-ing the top-level CA(s) cer-tifi-cates for peer cer-tifi-cate veri-fica-tion, used for en-crypted com-mu-nica-tions be-tween Zab-bix com-po-nents. This pa-ram-eter is sup-ported since Zab-bix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: *unencrypted* - connect without encryption (default) *psk* - connect using TLS and a pre-shared key (PSK) *cert* - connect using TLS and a certificate This parameter is supported

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSCRLFile	no			Full path-name of a file con-tain-ing re-voked cer-tifi-cates. This pa-ram-eter is used for en-crypted com-mu-nica-tions with Zab-bix com-po-nents. This pa-ram-eter is sup-ported since Zab-bix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSKeyFile	no			Full path-name of a file con-tain-ing the agent pri-vate key used for en-crypted com-mu-nica-tions with Zab-bix com-po-nents. This pa-ram-eter is sup-ported since Zab-bix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSPSKFile	no			Full path-name of a file con-tain-ing the agent pre-shared key used for en-crypted com-mu-nica-tions with Zab-bix com-po-nents. This pa-ram-eter is sup-ported since Zab-bix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSPSKIdentity	no			Pre-shared key iden-tity string, used for en-crypted com-mu-nica-tions with Zab-bix server. This pa-ram-eter is sup-ported since Zab-bix 3.0.0.
TLSServerCertIssuer	no			Allowed server (proxy) cer-tifi-cate is-suer. This pa-ram-eter is sup-ported since Zab-bix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
TLSServerCertSubject				Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
UnsafeUserParameters		0,1	0	Allow all characters to be passed in arguments to user-defined parameters. Supported since Zabbix 1.8.2. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ Additionally, newline characters are not allowed.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.

Parameter 参数 M	ndatory 必填 Ran	e 范围 Defau	t 默认 Descrip	ion 描述
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,wh -l

Note:
In Zabbix agent 2.0.0 version configuration parameters related to active and passive checks have been changed.
See the "See also" section at the bottom of this page to read more details about these changes.

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

Comments starting with "#" are only supported in the beginning of the line.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0

4 Zabbix agent (Windows)

Note:
The default values reflect daemon defaults, not the values in the shipped configuration files.

The parameters supported in a Zabbix agent (Windows) configuration file:

Parameter	Mandatory	Range	Default	Description
Alias	no			Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are allowed. Different *Alias* keys may reference the same item key. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* or *PerfCounter* parameters. Examples: 1. Retrieving paging file usage in percents from the server. Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage] Now shorthand key **pg_usage** may be used to retrieve data. 2. Getting CPU load with default and custom parameters. Alias=cpu.load:system.cpu.load Alias=cpu.load[*]:system.cpu.load[*] This allows use **cpu.load** key to get CPU utilisation percentage with default parameters as well as use **cpu.load[percpu,avg15]** to get specific data about CPU load. 3. Running multiple low-level discovery rules processing the same discovery items. Alias=vfs.fs.discovery[*]:vfs.fs.discovery Now it is possible to set up several discovery rules using **vfs.fs.discovery** with different parameters for each rule, e.g., **vfs.fs.discovery[foo]**, **vfs.fs.discovery[bar]**, etc.

Parameter	Mandatory	Range	Default	Description
BufferSend	no	1-3600	5	Do not keep data longer than N seconds in buffer.
BufferSize	no	2-65535	100	Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full.
DebugLevel	no	0-5	3	Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used only at host auto-registration process (active agent). If not defined, the value will be acquired from HostMetadataItem. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string. This option is supported in version 2.2.0 and higher.

Parameter	Mandatory	Range	Default	Description
HostMetadataItem	no			Optional parameter that defines a *Zabbix agent* item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters, performance counters and aliases. Supports *system.run[]* regardless of *EnableRemoteCommands* value. Host metadata is used only at host auto-registration process (active agent). During an auto-registration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. The value returned by the item must be a UTF-8 string otherwise it will be ignored. This option is supported in version 2.2.0 and higher.
Hostname	no		Set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 64
HostnameItem	no		system.hostname	Optional parameter that defines a *Zabbix agent* item used for getting host name. This option is only used when Hostname is not defined. Does not support UserParameters, performance counters or aliases, but does support *system.run[]* regardless of *EnableRemoteCommands* value. This option is supported in version 1.8.6 and higher. See also a more detailed description.

Parameter	Mandatory	Range	Default	Description
Include	no			You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `/absolute/path/to/config/files/*.` Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.
ListenIP	no		0.0.0.0	List of comma-delimited IP addresses that the agent should listen on. Multiple IP addresses are supported since Zabbix 1.8.3.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no		/tmp/zabbix_agentd.log	Name of the agent log file.
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. *Note*: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log Windows Event Log, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled
MaxLinesPerSecond	no	1-1000	20	Maximum number of new lines the agent will send per second to Zabbix server or proxy processing 'log', 'logrt' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log', 'logrt' or 'eventlog' item keys. *Note*: Zabbix will process 10 times more new lines than set in *MaxLinesPerSecond* to seek the required string in log items.

Parameter	Mandatory	Range	Default	Description
PerfCounter	no			Syntax: <parameter_name>,”<perf_counter_path>”,<period> Defines new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). For example, if you wish to receive average number of processor interrupts per second for last minute, you can define new parameter ”interrupts” as following: PerfCounter = interrupts,”\Processor(0)\Interrupts/sec”,60 Please note double quotes around performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating average value will be taken every second. You may run ”typeperf -qx” to get list of all performance counters available in Windows.
RefreshActiveChecks	no	60-3600	120	How often list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.
Server	yes, if StartAgents is not explicitly set to 0			List of comma delimited IP addresses, optionally in CIDR notation, or hostnames of Zabbix servers. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then ’127.0.0.1’, ’::127.0.0.1’, ’::ffff:127.0.0.1’ are treated equally and ’::/0’ will allow any IPv4 or IPv6 address. ’0.0.0.0/0’ can be used to allow any IPv4 address. Note, that ”IPv4-compatible IPv6 addresses” (0000::/96 prefix) are supported but deprecated by RFC4291. Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:d Spaces are allowed since the Zabbix 2.2 version.

Parameter	Mandatory	Range	Default	Description
ServerActive	no	(*)		IP:port (or hostname:port) of Zabbix server or Zabbix proxy for active checks. Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. No spaces allowed. If port is not specified, default port is used. IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional. If this parameter is not specified, active checks are disabled.
SourceIP	no			Source IP address for outgoing connections.
StartAgents	no	0-63 (*)	3	Number of pre-forked instances of zabbix_agentd that process passive checks. If set to 0, disables passive checks and the agent will not listen on any TCP port. The upper limit used to be 16 before version 1.8.5.
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma: *unencrypted* - accept connections without encryption (default) *psk* - accept connections with TLS and a pre-shared key (PSK) *cert* - accept connections with TLS and a certificate This parameter is supported since Zabbix 3.0.0.
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.

Parameter	Mandatory	Range	Default	Description
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: *unencrypted* - connect without encryption (default) *psk* - connect using TLS and a pre-shared key (PSK) *cert* - connect using TLS and a certificate This parameter is supported since Zabbix 3.0.0.
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.

Parameter	Mandatory	Range	Default	Description
UnsafeUserParameters	no	0-1	0	Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ Additionally, newline characters are not allowed.
UserParameter				User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Example: UserParameter=system.test,echo 1

Note:
(*) The number of active servers listed in ServerActive plus the number of pre-forked instances for passive checks specified in StartAgents must be less than 64.

Note:
In Zabbix agent 2.0.0 version configuration parameters related to active and passive checks have been changed.
See the "See also" section at the bottom of this page to read more details about these changes.

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

Comments starting with "#" are only supported in the beginning of the line.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0.

5 Zabbix Java 网关

5 Zabbix Java gateway

如果使用 `startup.sh` 和 `shutdown.sh` 脚本启动和停止Zabbix Java 网关, 那么就可以在 `settings.sh` 文件中指定必要的配置参数。startup 和 shutdown 脚本以配置文件为输入源，并且将 shell 变量 (第一列) 转换为相应的 Java 属性 (第二列)。If you use `startup.sh` and `shutdown.sh` scripts for starting Zabbix Java gateway, then you can specify the necessary configuration parameters in file `settings.sh`. The startup and shutdown scripts source the settings file and take care of converting shell variables (listed in the first column) to Java properties (listed in the second column).

如果通过手动运行 `java` 命令来起动 Zabbix Java 网关, 可以通过命令行方式来指定 Java 属性. If you start Zabbix Java gateway manually by running `java` directly, then you specify the corresponding Java properties on the command line.

变量参	必须配	范围默	值描述	息	
Variable	Property	Mandatory	Range	Default	Description
LISTEN_IP	zabbix.listenIP	否	.0.0.0	听 IP。	

LISTEN_IP	zabbix.listenIP	no	0.0.0.0	IP address to listen on.

LISTEN_PORT	zabbix.listenPort	否	024-32767	0052	听端口。
LISTEN_PORT	zabbix.listenPort	no	1024-32767	10052	Port to listen on.

PID_FILE	zabbix.pidFile	否		tmp/zabbix_java.pid	ID 文件的名称。如果省略，Zabbix Java 网关将作为控制台应用程序启动。
PID_FILE	zabbix.pidFile	no		/tmp/zabbix_java.pid	Name of PID file. If omitted, Zabbix Java Gateway is started as a console application.

START_POLLERS	zabbix.startPollers	否	-1000		动多少个轮询线程。
START_POLLERS	zabbix.startPollers	no	1-1000	5	Number of worker threads to start.

TIMEOUT	zabbix.timeout	否	-30		络超时时间。从 Zabbix 2.0.15, 2.2.10 和 2.4.5 开始支持该参数。
TIMEOUT	zabbix.timeout	no	1-30	3	How long to wait for network operations. This parameter is supported since Zabbix 2.0.15, 2.2.10 and 2.4.5.

Warning:
端口 10052 没有IANA 注册.

Warning:
Port 10052 is not IANA registered.

6 关于 **“Include”** 参数的特别说明

6 Special notes on ”Include” parameter

如果 Include 参数用来包含一个文件，该文件必须可读。If an Include parameter is used for including a file, the file must be readable.

如果 Include 参数用来包含一个目录: If an Include parameter is used for including a directory:

- 该目录下所有文件必须可读。
- 不考虑包含的特定顺序（例如：文件不按字母顺序包含）。因此，不要在几个''Include''文件中定义一个相同参数(
- 该目录下的所有文件都包含在配置文件中。
- 注意一些文本编辑器会自动创建文件备份。 如，如果编辑 ''include/my_specific.conf'' 会产生一个副本 ''incl
- 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 '

如果 Include 参数使用模式来匹配包含的文件：If an Include parameter is used for including files using a pattern:

- 与模式匹配的所有文件都必须是可读的。
- 不考虑包含的特定顺序（例如：文件不按字母顺序包含）。因此，不要在几个''Include''文件中定义一个相同参数(
- All files matching the pattern must be readable.
- No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order)

4 各种协议

4 Protocols

1 Server-proxy 数据交换协议

概述

Server-proxy 数据交换基于 JSON 格式。

请求和响应消息必须以header and data length开头

被动代理

代理配置请求

proxy config 请求由服务器发送以提供代理配置数据。每次发送此请求 ProxyConfigFrequency （服务器配置参数）秒。

name	value type	description
server→proxy:		
request	*string*	'proxy config'
<table>	*object*	one or more objects with <table> data
fields	*array*	array of field names
-	*string*	field name
data	*array*	array of rows
-	*array*	array of columns
-	*string,number*	column value with type depending on column type in database schema
proxy→server:		
response	*string*	the request success information ('success' or 'failed')
version	*string*	the proxy version (<major>.<minor>.<build>)

例：

server→proxy:

```
{
    "request": "proxy config",
    "globalmacro":{
        "fields":[
            "globalmacroid",
            "macro",
            "value"
        ],
        "data":[
            [
                2,
                "{$SNMP_COMMUNITY}",
                "public"
            ]
        ]
    },
    "hosts":{
```

```
        "fields":[
            "hostid",
            "host",
            "status",
            "ipmi_authtype",
            "ipmi_privilege",
            "ipmi_username",
            "ipmi_password",
            "name",
            "tls_connect",
            "tls_accept",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Template OS Linux",
                3,
                -1,
                2,
                "",
                "",
                "Template OS Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10050,
                "Template App Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Template App Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
```

```
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```


proxy→server:


```
{
  "response": "success",
  "version": "3.4.0"
}
```


代理请求

`proxy data` request 用于从代理获取主机可用性，历史，发现和自动注册数据。每次发送此请求 `ProxyDataFrequency` （服务器配置参数）秒。

name	value type	description
server→proxy:		
request	*string*	'proxy data'
proxy→server:		
host availability	*array*	*(optional)* array of host availability data objects
hostid	*number*	host identifier
available	*number*	Zabbix agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
error	*string*	Zabbix agent error message or empty string
snmp_available	*number*	SNMP agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
snmp_error	*string*	SNMP agent error message or empty string

name	value type	description
	ipmi_available *number*	IPMI agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	ipmi_error *string*	IPMI agent error message or empty string
	jmx_available *number*	JMX agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	jmx_error *string*	JMX agent error message or empty string
history data	*array*	*(optional)* array of history data objects
	itemid *number*	item identifier
	clock *number*	item value timestamp (seconds)
	ns *number*	item value timestamp (nanoseconds)
	value *string*	*(optional)* item value
	timestamp *number*	*(optional)* timestamp of log type items
	source *string*	*(optional)* eventlog item source value
	severity *number*	*(optional)* eventlog item severity value
	eventid *number*	*(optional)* eventlog item eventid value
	state *string*	*(optional)* item state **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
	lastlogsize *number*	*(optional)* last logs ize of log type items
	mtime *number*	*(optional)* modify time of log type items
discovery data	*array*	*(optional)* array of discovery data objects
	clock *number*	the discovery data timestamp
	druleid *number*	the discovery rule identifier
	dcheckid *number*	the discovery check indentifier or null for discovery rule data
	type *number*	the discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
	ip *string*	the host IP address
	dns *string*	the host DNS name
	port *number*	*(optional)* service port number
	key_ *string*	*(optional)* the item key for discovery check of type **9** *SVC_AGENT*
	value *string*	*(optional)* value received from the service, can be empty for most of services
	status *number*	*(optional)* service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN

name	value type	description
auto registration	*array*	*(optional)* array of auto registration data objects
	clock *number*	the auto registration data timestamp
	host *string*	the host name
	ip *string*	*(optional)* the host IP address
	dns *string*	*(optional)* the resolved DNS name from IP address
	port *string*	*(optional)* the host port
	host_metadata *string*	*(optional)* the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks	*array*	*(optional)* array of tasks
	type *number*	the task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
	status *number*	the remote command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - the remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - the remote command failed
	error *string*	*(optional)* the error message
	parent_taskid *string*	the parent task id
more	*number*	*(optional)* 1 - there are more history data to send
clock	*number*	data transfer timestamp (seconds)
ns	*number*	data transfer timestamp (nanoseconds)
version	*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	*string*	the request success information (’success’ or ’failed’)
tasks	*array*	*(optional)* array of tasks
	type *number*	the task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
	clock *number*	the task creation time
	ttl *number*	the time in seconds after which task expires
	command_type *number*	the remote command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
	command *string*	the remote command to execute
	execute_on *number*	the execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
	port *number*	*(optional)* the port for telnet and ssh commands

name	value type	description
authtype	*number*	*(optional)* the authentication type for ssh commands
username	*string*	*(optional)* the user name for telnet and ssh commands
password	*string*	*(optional)* the password for telnet and ssh commands
publickey	*string*	*(optional)* the public key for ssh commands
privatekey	*string*	*(optional)* the private key for ssh commands
parent_taskid	*number*	the parent task id
hostid	*number*	target hostid

例如:

server→proxy:

```
{
  "request": "proxy data"
}
```

proxy→server:

```
{
    "host availability":[
        {
            "hostid":10106,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        },
        {
            "hostid":10107,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "value":"52956612"
        },
        {
            "itemid":"12346",
            "clock":1478609647,
            "ns":330690279,
            "state":1,
            "value":"Cannot find information for this network interface in /proc/net/dev."
        }
    ],
    "discovery data":[
```

```
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":3,
            "type":12,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        },
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":null,
            "type":-1,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        }
    ],
    "auto registration":[
        {
            "clock":1478608371,
            "host":"Logger1",
            "ip":"10.3.0.1",
            "dns":"localhost",
            "port":"10050"
        },
        {
            "clock":1478608381,
            "host":"Logger2",
            "ip":"10.3.0.2",
            "dns":"localhost",
            "port":"10050"
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "clock":1478609648,
    "ns":157729208,
    "version":"3.4.0"
}
```


server→proxy:


```
{
  "response": "success",
  "tasks":[
      {
         "type": 1,
         "clock": 1478608371,
         "ttl": 600,
         "commandtype": 2,
```

```
            "command": "restart_service1.sh",
            "execute_on": 2,
            "port": 80,
            "authtype": 0,
            "username": "userA",
            "password": "password1",
            "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
            "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
            "parent_taskid": 10,
            "hostid": 10070
        },
        {
            "type": 1,
            "clock": 1478608381,
            "ttl": 600,
            "commandtype": 1,
            "command": "restart_service2.sh",
            "execute_on": 0,
            "authtype": 0,
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "parent_taskid": 20,
            "hostid": 10084
        }
    ]
}
```


主动代理

代理心跳请求

`proxy heartbeat` 请求由代理发送以报告代理正在运行。每次发送此请求 `HeartbeatFrequency` （代理配置参数）秒。

name	value type	description
proxy→server:		
request	*string*	'proxy heartbeat'
host	*string*	the proxy name
version	*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	*string*	the request success information ('success' or 'failed')

proxy→server:


```
{
    "request": "proxy heartbeat",
    "host": "Proxy #12",
    "version": "3.4.0"
}
```


server→proxy:


```
{
    "response": "success"
}
```


代理配置请求

`proxy config` 请求由代理发送以获取代理配置数据。每次发送此请求 `ConfigFrequency` （代理配置参数）秒。

name	value type	description
proxy→server:		
request	*string*	'proxy config'

name	value type	description
host	*string*	proxy name
version	*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:		
request	*string*	'proxy config'
<table>	*object*	one or more objects with <table> data
fields	*array*	array of field names
-	*string*	field name
data	*array*	array of rows
-	*array*	array of columns
-	*string,number*	column value with type depending on column type in database schema
proxy→server:		
response	*string*	the request success information ('success' or 'failed')

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version":"3.4.0"
}
```

server→proxy:

```
{
    "globalmacro":{
        "fields":[
            "globalmacroid",
            "macro",
            "value"
        ],
        "data":[
            [
                2,
                "{$SNMP_COMMUNITY}",
                "public"
            ]
        ]
    },
    "hosts":{
        "fields":[
            "hostid",
            "host",
            "status",
            "ipmi_authtype",
            "ipmi_privilege",
            "ipmi_username",
            "ipmi_password",
            "name",
            "tls_connect",
            "tls_accept",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
```

```
                10001,
                "Template OS Linux",
                3,
                -1,
                2,
                "",
                "",
                "Template OS Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10050,
                "Template App Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Template App Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
```

```
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```

proxy→server:

```
{
  "response": "success"
}
```

代理数据请求

`proxy data` 请求由代理发送，以提供主机可用性，历史记录，发现和自动注册数据。每次发送此请求 `DataSenderFrequency` （代理配置参数）秒。

name	value type	description
proxy→server:		
request	*string*	'proxy data'
host	*string*	the proxy name
host availability	*array*	*(optional)* array of host availability data objects
hostid	*number*	host identifier
available	*number*	Zabbix agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
error	*string*	Zabbix agent error message or empty string
snmp_available		SNMP agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
snmp_error	*string*	SNMP agent error message or empty string
ipmi_available		IPMI agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
ipmi_error	*string*	IPMI agent error message or empty string
jmx_available		JMX agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
jmx_error	*string*	JMX agent error message or empty string
history data	*array*	*(optional)* array of history data objects
itemid	*number*	item identifier
clock	*number*	item value timestamp (seconds)
ns	*number*	item value timestamp (nanoseconds)
value	*string*	*(optional)* item value
timestamp	*number*	*(optional)* timestamp of log type items
source	*string*	*(optional)* eventlog item source value

name	value type	description
	severity *number*	*(optional)* eventlog item severity value
	eventid *number*	*(optional)* eventlog item eventid value
	state *string*	*(optional)* item state **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
	lastlogsize *number*	*(optional)* last logs ize of log type items
	mtime *number*	*(optional)* modify time of log type items
discovery data	*array*	*(optional)* array of discovery data objects
	clock *number*	the discovery data timestamp
	druleid *number*	the discovery rule identifier
	dcheckid *number*	the discovery check indentifier or null for discovery rule data
	type *number*	the discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
	ip *string*	the host IP address
	dns *string*	the host DNS name
	port *number*	*(optional)* service port number
	key_ *string*	*(optional)* the item key for discovery check of type **9** *SVC_AGENT*
	value *string*	*(optional)* value received from the service, can be empty for most of services
	status *number*	*(optional)* service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
auto registration	*array*	*(optional)* array of auto registration data objects
	clock *number*	the auto registration data timestamp
	host *string*	the host name
	ip *string*	*(optional)* the host IP address
	dns *string*	*(optional)* the resolved DNS name from IP address
	port *string*	*(optional)* the host port
	host_metadata *string*	*(optional)* the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks	*array*	*(optional)* array of tasks
	type *number*	the task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result

name	value type	description
	status *number*	the remote command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - the remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - the remote command failed
	error *string*	*(optional)* the error message
	parent_taskid *number*	the parent task id
more	*number*	*(optional)* 1 - there are more history data to send
clock	*number*	data transfer timestamp (seconds)
ns	*number*	data transfer timestamp (nanoseconds)
version	*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	*string*	the request success information (’success’ or ’failed’)
tasks	*array*	*(optional)* array of tasks
	type *number*	the task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
	clock *number*	the task creation time
	ttl *number*	the time in seconds after which task expires
	command_type *number*	the remote command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
	command *string*	the remote command to execute
	execute_on *number*	the execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
	port *number*	*(optional)* the port for telnet and ssh commands
	authtype *number*	*(optional)* the authentication type for ssh commands
	username *string*	*(optional)* the user name for telnet and ssh commands
	password *string*	*(optional)* the password for telnet and ssh commands
	publickey *string*	*(optional)* the public key for ssh commands
	privatekey *string*	*(optional)* the private key for ssh commands
	parent_taskid *number*	the parent task id
	hostid *number*	target hostid

例如:

proxy→server:

```
{
  "request": "proxy data",
  "host": "Proxy #12",
```

```
    "host availability":[
        {
            "hostid":10106,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        },
        {
            "hostid":10107,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "value":"52956612"
        },
        {
            "itemid":"12346",
            "clock":1478609647,
            "ns":330690279,
            "state":1,
            "value":"Cannot find information for this network interface in /proc/net/dev."
        }
    ],
    "discovery data":[
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":3,
            "type":12,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        },
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":null,
            "type":-1,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        }
    ],
    "auto registration":[
        {
            "clock":1478608371,
```

```
            "host":"Logger1",
            "ip":"10.3.0.1",
            "dns":"localhost",
            "port":"10050"
        },
        {
            "clock":1478608381,
            "host":"Logger2",
            "ip":"10.3.0.2",
            "dns":"localhost",
            "port":"10050"
        }
    ],
    "tasks":[
        {
            "type": 2,
            "clock":1478608371,
            "ttl": 600,
            "commandtype": 2,
            "command": "restart_service1.sh",
            "execute_on": 2,
            "port": 80,
            "authtype": 0,
            "username": "userA",
            "password": "password1",
            "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
            "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
            "parent_taskid": 10,
            "hostid": 10070
        },
        {
            "type": 2,
            "clock":1478608381,
            "ttl": 600,
            "commandtype": 1,
            "command": "restart_service2.sh",
            "execute_on": 0,
            "authtype": 0,
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "parent_taskid": 20,
            "hostid": 10084
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "clock":1478609648,
    "ns":157729208,
    "version":"3.4.0"
```

```
}
```

server→proxy:

```
{
  "response": "success",
  "tasks":[
      {
         "type": 1,
         "clock": 1478608371,
         "ttl": 600,
         "commandtype": 2,
         "command": "restart_service1.sh",
         "execute_on": 2,
         "port": 80,
         "authtype": 0,
         "username": "userA",
         "password": "password1",
         "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
         "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
         "parent_taskid": 10,
         "hostid": 10070
      },
      {
         "type": 1,
         "clock": 1478608381,
         "ttl": 600,
         "commandtype": 1,
         "command": "restart_service2.sh",
         "execute_on": 0,
         "authtype": 0,
         "username": "",
         "password": "",
         "publickey": "",
         "privatekey": "",
         "parent_taskid": 20,
         "hostid": 10084
      }
  ]
}
```

向后兼容性

服务器通过接受旧的支持部分向后兼容 `host availability`, `history data`, `discovery data` and `auto registration` 请求.

1 Server-proxy data exchange protocol

Overview

Server - proxy data exchange is based on JSON format.

Request and response messages must begin with header and data length.

Passive proxy

Proxy config request

The `proxy config` request is sent by server to provide proxy configuration data. This request is sent every `ProxyConfigFrequency` (server configuration parameter) seconds.

name		value type	description
server→proxy:			
request		*string*	'proxy config'
<table>		*object*	one or more objects with <table> data
	fields	*array*	array of field names
	-	*string*	field name
	data	*array*	array of rows

name	value type	description
-	*array*	array of columns
-	*string,number*	column value with type depending on column type in database schema
proxy→server:		
response	*string*	the request success information ('success' or 'failed')
version	*string*	the proxy version (<major>.<minor>.<build>)

Example:

server→proxy:

```
{
    "request": "proxy config",
    "globalmacro":{
        "fields":[
            "globalmacroid",
            "macro",
            "value"
        ],
        "data":[
            [
                2,
                "{$SNMP_COMMUNITY}",
                "public"
            ]
        ]
    },
    "hosts":{
        "fields":[
            "hostid",
            "host",
            "status",
            "ipmi_authtype",
            "ipmi_privilege",
            "ipmi_username",
            "ipmi_password",
            "name",
            "tls_connect",
            "tls_accept",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Template OS Linux",
                3,
                -1,
                2,
                "",
                "",
                "Template OS Linux",
                1,
                1,
                "",
                "",
                "",
                ""
```

```
            ],
            [
                10050,
                "Template App Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Template App Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
```

```
}
```

proxy→server:

```
{
  "response": "success",
  "version": "3.4.0"
}
```

Proxy request

The `proxy data` request is used to obtain host availability, historical, discovery and autoregistration data from proxy. This request is sent every `ProxyDataFrequency` (server configuration parameter) seconds.

name	value type	description
server→proxy:		
request	*string*	'proxy data'
proxy→server:		
host availability	*array*	*(optional)* array of host availability data objects
hostid	*number*	host identifier
available	*number*	Zabbix agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
error	*string*	Zabbix agent error message or empty string
snmp_available	*number*	SNMP agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
snmp_error	*string*	SNMP agent error message or empty string
ipmi_available	*number*	IPMI agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
ipmi_error	*string*	IPMI agent error message or empty string
jmx_available	*number*	JMX agent availability
		0, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
jmx_error	*string*	JMX agent error message or empty string
history data	*array*	*(optional)* array of history data objects
itemid	*number*	item identifier
clock	*number*	item value timestamp (seconds)
ns	*number*	item value timestamp (nanoseconds)
value	*string*	*(optional)* item value
timestamp	*number*	*(optional)* timestamp of log type items
source	*string*	*(optional)* eventlog item source value
severity	*number*	*(optional)* eventlog item severity value
eventid	*number*	*(optional)* eventlog item eventid value
state	*string*	*(optional)* item state
		0, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
lastlogsize	*number*	*(optional)* last logs ize of log type items
mtime	*number*	*(optional)* modify time of log type items
discovery data	*array*	*(optional)* array of discovery data objects
clock	*number*	the discovery data timestamp
druleid	*number*	the discovery rule identifier
dcheckid	*number*	the discovery check indentifier or null for discovery rule data

name		value type	description
	type	*number*	the discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
	ip	*string*	the host IP address
	dns	*string*	the host DNS name
	port	*number*	*(optional)* service port number
	key_	*string*	*(optional)* the item key for discovery check of type **9** *SVC_AGENT*
	value	*string*	*(optional)* value received from the service, can be empty for most of services
	status	*number*	*(optional)* service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
auto registration		*array*	*(optional)* array of auto registration data objects
	clock	*number*	the auto registration data timestamp
	host	*string*	the host name
	ip	*string*	*(optional)* the host IP address
	dns	*string*	*(optional)* the resolved DNS name from IP address
	port	*string*	*(optional)* the host port
	host_metadata	*string*	*(optional)* the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks		*array*	*(optional)* array of tasks
	type	*number*	the task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
	status	*number*	the remote command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - the remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - the remote command failed
	error	*string*	*(optional)* the error message
	parent_taskid	*number*	the parent task id
more		*number*	*(optional)* 1 - there are more history data to send
clock		*number*	data transfer timestamp (seconds)
ns		*number*	data transfer timestamp (nanoseconds)
version		*string*	the proxy version (<major>.<minor>.<build>)

server→proxy:

name	value type	description
response	*string*	the request success information ('success' or 'failed')
tasks	*array*	*(optional)* array of tasks
	type *number*	the task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
	clock *number*	the task creation time
	ttl *number*	the time in seconds after which task expires
	commandtype *number*	the remote command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
	command *string*	the remote command to execute
	execute_on *number*	the execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
	port *number*	*(optional)* the port for telnet and ssh commands
	authtype *number*	*(optional)* the authentication type for ssh commands
	username *string*	*(optional)* the user name for telnet and ssh commands
	password *string*	*(optional)* the password for telnet and ssh commands
	publickey *string*	*(optional)* the public key for ssh commands
	privatekey *string*	*(optional)* the private key for ssh commands
	parent_taskid *number*	the parent task id
	hostid *number*	target hostid

Example:

server→proxy:

```
{
  "request": "proxy data"
}
```

proxy→server:

```
{
    "host availability":[
        {
            "hostid":10106,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
```

```
            "jmx_error":""
        },
        {
            "hostid":10107,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "value":"52956612"
        },
        {
            "itemid":"12346",
            "clock":1478609647,
            "ns":330690279,
            "state":1,
            "value":"Cannot find information for this network interface in /proc/net/dev."
        }
    ],
    "discovery data":[
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":3,
            "type":12,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        },
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":null,
            "type":-1,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        }
    ],
    "auto registration":[
        {
            "clock":1478608371,
            "host":"Logger1",
            "ip":"10.3.0.1",
            "dns":"localhost",
            "port":"10050"
        },
        {
            "clock":1478608381,
            "host":"Logger2",
            "ip":"10.3.0.2",
            "dns":"localhost",
```

```
            "port":"10050"
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "clock":1478609648,
    "ns":157729208,
    "version":"3.4.0"
}
```


server→proxy:


```
{
  "response": "success",
  "tasks":[
      {
         "type": 1,
         "clock": 1478608371,
         "ttl": 600,
         "commandtype": 2,
         "command": "restart_service1.sh",
         "execute_on": 2,
         "port": 80,
         "authtype": 0,
         "username": "userA",
         "password": "password1",
         "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
         "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
         "parent_taskid": 10,
         "hostid": 10070
      },
      {
         "type": 1,
         "clock": 1478608381,
         "ttl": 600,
         "commandtype": 1,
         "command": "restart_service2.sh",
         "execute_on": 0,
         "authtype": 0,
         "username": "",
         "password": "",
         "publickey": "",
         "privatekey": "",
         "parent_taskid": 20,
         "hostid": 10084
      }
  ]
}
```


Active proxy

Proxy heartbeat request

The `proxy heartbeat` request is sent by proxy to report that proxy is running. This request is sent every `HeartbeatFrequency`

(proxy configuration parameter) seconds.

name	value type	description
proxy→server:		
request	*string*	'proxy heartbeat'
host	*string*	the proxy name
version	*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	*string*	the request success information ('success' or 'failed')

proxy→server:

```
{
   "request": "proxy heartbeat",
   "host": "Proxy #12",
   "version": "3.4.0"
}
```

server→proxy:

```
{
  "response": "success"
}
```

Proxy config request

The `proxy config` request is sent by proxy to obtain proxy configuration data. This request is sent every `ConfigFrequency` (proxy configuration parameter) seconds.

name			value type	description
proxy→server:				
request			*string*	'proxy config'
host			*string*	proxy name
version			*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:				
request			*string*	'proxy config'
<table>			*object*	one or more objects with <table> data
	fields		*array*	array of field names
		-	*string*	field name
	data		*array*	array of rows
		-	*array*	array of columns
			- *string,number*	column value with type depending on column type in database schema
proxy→server:				
response			*string*	the request success information ('success' or 'failed')

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version":"3.4.0"
}
```

server→proxy:

```
{
    "globalmacro":{
        "fields":[
            "globalmacroid",
```

```
            "macro",
            "value"
        ],
        "data":[
            [
                2,
                "{$SNMP_COMMUNITY}",
                "public"
            ]
        ]
    },
    "hosts":{
        "fields":[
            "hostid",
            "host",
            "status",
            "ipmi_authtype",
            "ipmi_privilege",
            "ipmi_username",
            "ipmi_password",
            "name",
            "tls_connect",
            "tls_accept",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Template OS Linux",
                3,
                -1,
                2,
                "",
                "",
                "Template OS Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10050,
                "Template App Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Template App Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
```

```
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```

proxy→server:

```
{
  "response": "success"
}
```

Proxy data request

The `proxy data` request is sent by proxy to provide host availability, history, discovery and auto registation data. This request is sent every `DataSenderFrequency` (proxy configuration parameter) seconds.

name	value type	description
proxy→server:		
request	*string*	'proxy data'
host	*string*	the proxy name
host availability	*array*	*(optional)* array of host availability data objects
	hostid*number*	host identifier

name	value type	description
	available *number*	Zabbix agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	error *string*	Zabbix agent error message or empty string
	snmp_available *number*	SNMP agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	snmp_error *string*	SNMP agent error message or empty string
	ipmi_available *number*	IPMI agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	ipmi_error *string*	IPMI agent error message or empty string
	jmx_available *number*	JMX agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	jmx_error *string*	JMX agent error message or empty string
history data	*array*	*(optional)* array of history data objects
	itemid *number*	item identifier
	clock *number*	item value timestamp (seconds)
	ns *number*	item value timestamp (nanoseconds)
	value *string*	*(optional)* item value
	timestamp *number*	*(optional)* timestamp of log type items
	source *string*	*(optional)* eventlog item source value
	severity *number*	*(optional)* eventlog item severity value
	eventid *number*	*(optional)* eventlog item eventid value
	state *string*	*(optional)* item state **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
	lastlogsize *number*	*(optional)* last logs ize of log type items
	mtime *number*	*(optional)* modify time of log type items
discovery data	*array*	*(optional)* array of discovery data objects
	clock *number*	the discovery data timestamp
	druleid *number*	the discovery rule identifier
	dcheckid *number*	the discovery check indentifier or null for discovery rule data
	type *number*	the discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check

name		value type	description
	ip	*string*	the host IP address
	dns	*string*	the host DNS name
	port	*number*	*(optional)* service port number
	key_	*string*	*(optional)* the item key for discovery check of type **9** *SVC_AGENT*
	value	*string*	*(optional)* value received from the service, can be empty for most of services
	status	*number*	*(optional)* service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
auto registration		*array*	*(optional)* array of auto registration data objects
	clock	*number*	the auto registration data timestamp
	host	*string*	the host name
	ip	*string*	*(optional)* the host IP address
	dns	*string*	*(optional)* the resolved DNS name from IP address
	port	*string*	*(optional)* the host port
	host_metadata	*string*	*(optional)* the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks		*array*	*(optional)* array of tasks
	type	*number*	the task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
	status	*number*	the remote command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - the remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - the remote command failed
	error	*string*	*(optional)* the error message
	parent_taskid	*number*	the parent task id
more		*number*	*(optional)* 1 - there are more history data to send
clock		*number*	data transfer timestamp (seconds)
ns		*number*	data transfer timestamp (nanoseconds)
version		*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:			
response		*string*	the request success information (’success’ or ’failed’)
tasks		*array*	*(optional)* array of tasks
	type	*number*	the task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
	clock	*number*	the task creation time
	ttl	*number*	the time in seconds after which task expires
	command_type	*number*	the remote command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)

name	value type	description
command	*string*	the remote command to execute
execute_on	*number*	the execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
port	*number*	*(optional)* the port for telnet and ssh commands
authtype	*number*	*(optional)* the authentication type for ssh commands
username	*string*	*(optional)* the user name for telnet and ssh commands
password	*string*	*(optional)* the password for telnet and ssh commands
publickey	*string*	*(optional)* the public key for ssh commands
privatekey	*string*	*(optional)* the private key for ssh commands
parent_taskid	*number*	the parent task id
hostid	*number*	target hostid

Example:

proxy→server:

```
{
  "request": "proxy data",
  "host": "Proxy #12",
    "host availability":[
        {
            "hostid":10106,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        },
        {
            "hostid":10107,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "value":"52956612"
        },
        {
            "itemid":"12346",
            "clock":1478609647,
```

```
            "ns":330690279,
            "state":1,
            "value":"Cannot find information for this network interface in /proc/net/dev."
        }
    ],
    "discovery data":[
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":3,
            "type":12,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        },
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":null,
            "type":-1,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        }
    ],
    "auto registration":[
        {
            "clock":1478608371,
            "host":"Logger1",
            "ip":"10.3.0.1",
            "dns":"localhost",
            "port":"10050"
        },
        {
            "clock":1478608381,
            "host":"Logger2",
            "ip":"10.3.0.2",
            "dns":"localhost",
            "port":"10050"
        }
    ],
    "tasks":[
        {
            "type": 2,
            "clock":1478608371,
            "ttl": 600,
            "commandtype": 2,
            "command": "restart_service1.sh",
            "execute_on": 2,
            "port": 80,
            "authtype": 0,
            "username": "userA",
            "password": "password1",
            "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
            "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
            "parent_taskid": 10,
            "hostid": 10070
        },
        {
            "type": 2,
            "clock":1478608381,
            "ttl": 600,
```

```
            "commandtype": 1,
            "command": "restart_service2.sh",
            "execute_on": 0,
            "authtype": 0,
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "parent_taskid": 20,
            "hostid": 10084
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "clock":1478609648,
    "ns":157729208,
    "version":"3.4.0"
}
```


server→proxy:


```
{
  "response": "success",
  "tasks":[
      {
          "type": 1,
          "clock": 1478608371,
          "ttl": 600,
          "commandtype": 2,
          "command": "restart_service1.sh",
          "execute_on": 2,
          "port": 80,
          "authtype": 0,
          "username": "userA",
          "password": "password1",
          "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
          "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
          "parent_taskid": 10,
          "hostid": 10070
      },
      {
          "type": 1,
          "clock": 1478608381,
          "ttl": 600,
          "commandtype": 1,
          "command": "restart_service2.sh",
          "execute_on": 0,
          "authtype": 0,
          "username": "",
          "password": "",
          "publickey": "",
          "privatekey": "",
```

```
        "parent_taskid": 20,
        "hostid": 10084
    }
  ]
}
```

Backwards compatibility

Server supports partial backwards compatibility by accepting old `host availability`, `history data`, `discovery data` and `auto registration` requests.

2 Zabbix 代理协议

有关详细信息，请参阅被动和主动代理检查页面。

3 Zabbix sender 协议

有关详细信息，请参阅Trapper items页面。

4 Header and data length

概述

Zabbix 组件之间的响应和请求消息中存在标头和数据长度。需要确定消息的长度。

```
<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, 64 bit num
```

为了不耗尽内存（可能），Zabbix 协议仅限于在一个连接中仅接受 128MB。

实施

以下是显示如何将 Zabbix 协议标头添加到的代码段 `data` 你 想要发送以获取 `packet` 你 必须发送到 Zabbix 以便正确解释。

Language	Code
bash	`printf -v LENGTH '%016x' "${#DATA}"PACK=`
Java	`byte[] header = new byte[] {'Z', 'B', 'X`
PHP	`$packet = "ZBXD\1" . pack('P', strlen($d`
Perl	`my $packet = "ZBXD\1" . pack('<Q', lengt`
Python	`packet = "ZBXD\1" + struct.pack('<Q', le`

4 Header and data length

Overview

Header and data length are present in response and request messages between Zabbix components. It is required to determine the length of message.

```
<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, 64 bit num
```

To not exhaust memory (potentially) Zabbix protocol is limited to accept only 128MB in one connection.

Implementation

Here are code snippets showing how to add Zabbix protocol header to the `data` you *want* to send in order to obtain `packet` you *should* send to Zabbix so it is interpreted correctly.

Language	Code
bash	`printf -v LENGTH '%016x' "${#DATA}"PACK=`
Java	`byte[] header = new byte[] {'Z', 'B', 'X`
PHP	`$packet = "ZBXD\1" . pack('P', strlen($d`
Perl	`my $packet = "ZBXD\1" . pack('<Q', lengt`
Python	`packet = "ZBXD\1" + struct.pack('<Q', le`

5 实时导出协议

此部分描述real-time export 协议以换行符分隔的 JSON 格式：

- trigger events
- item values
- trends

所有文件都有.ndjson 扩展名。导出文件的每一行都是 JSON 对象。

触发事件

导出问题事件的以下信息：

Field			Type	Description
hosts			array	List of hosts involved in the trigger expression; there should be at least one element in array.
	-		string	Visible host name.
groups			array	list of host groups of all hosts involved in the trigger expression; there should be at least one element in array.
	-		string	Host group name.
tags			array	List of problem tags (can be empty).
	-		object	
		tag	string	Tag name.
		value	string	Tag value (can be empty).
name			string	Problem event name.
clock			number	Number of seconds since Epoch to the moment when problem was detected (integer part).
ns			number	Number of nanoseconds to be added to `clock` to get a precise problem detection time.
eventid			number	Problem event ID.
value			number	1 (always).

将为恢复事件导出以下信息：

Field	Type	Description
clock	number	Number of seconds since Epoch to the moment when problem was resolved (integer part).
ns	number	Number of nanoseconds to be added to `clock` to get a precise problem resolution time.
eventid	number	Recovery event ID.
p_eventid	number	Problem event ID.
value	number	0 (always).

例子

Problem:


```
{"hosts":["Host B","Zabbix Server"],"groups":["Group X","Group Y","Group Z","Zabbix servers"],"tags":[{"ta
```


Recovery:


```
{"clock":1519304345,"ns":987654321,"eventid":43,"p_eventid":42,"value":0}
```


Problem (multiple problem event generation):


```
{"hosts":["Host B","Zabbix Server"],"groups":["Group X","Group Y","Group Z","Zabbix servers"],"tags":[{"ta

{"hosts":["Host B","Zabbix Server"],"groups":["Group X","Group Y","Group Z","Zabbix servers"],"tags":[{"ta
```


Recovery:


```
{"clock":1519304346,"ns":987654321,"eventid":44,"p_eventid":43,"value":0}

{"clock":1519304346,"ns":987654321,"eventid":44,"p_eventid":42,"value":0}
```

监控项值

将为收集的项目值导出以下信息：

Field		Type	Description
host		string	Visible host name of the item host.
groups		array	List of host groups of the item host; there should be at least one element in array.
	-	string	Host group name.
applications		array	List of the item applications; empty if there are none.
	-	string	Application name.
itemid		number	Item ID.
name		string	Visible item name.
clock		number	Number of seconds since Epoch to the moment when value was collected (integer part).
ns		number	Number of nanoseconds to be added to `clock` to get a precise value collection time.
timestamp (*Log* only)		number	0 if not available.
source (*Log* only)		string	Empty string if not available.
severity (*Log* only)		number	0 if not available.
logeventid (*Log* only)		number	0 if not available.
value		number (for numeric items) or string (for text items)	Collected item value.

例子

Numeric (unsigned) value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Availability"],"
```

Numeric (float) value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["CPU","Performance"],"itemid":4,
```

Character, text value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Installed softwa
```

Log value:

```
{"host":"Host A","groups":["Group X","Group Y","Group Z"],"applications":["Log files","Critical"],"itemid"
```

趋势

将为计算的趋势值导出以下信息：

Field		Type	Description
host		string	Visible host name of the item host.
groups		array	List of host groups of the item host; there should be at least one element in array.
	-	string	Host group name.
applications		array	List of the item applications; empty if there are none.
	-	string	Application name.
itemid		number	Item ID.
name		string	Visible item name.
clock		number	Number of seconds since Epoch to the moment when value was collected (integer part).
count		number	Number of values collected for a given hour.
min		number	Minimum item value for a given hour.
avg		number	Average item value for a given hour.
max		number	Maximum item value for a given hour.

例子

Numeric (unsigned) value:


```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Availability"],"
```


Numeric (float) value:


```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["CPU","Performance"],"itemid":4,
```


5 Real-time export protocol

This section presents details of the real-time export protocol in a newline-delimited JSON format for:

- trigger events
- item values
- trends

All files have a .ndjson extension. Each line of the export file is a JSON object.

Trigger events

The following information is exported for a problem event:

Field			Type	Description
hosts			array	List of hosts involved in the trigger expression; there should be at least one element in array.
	-		string	Visible host name.
groups			array	list of host groups of all hosts involved in the trigger expression; there should be at least one element in array.
	-		string	Host group name.
tags			array	List of problem tags (can be empty).
	-		object	
		tag	string	Tag name.
		value	string	Tag value (can be empty).
name			string	Problem event name.
clock			number	Number of seconds since Epoch to the moment when problem was detected (integer part).
ns			number	Number of nanoseconds to be added to `clock` to get a precise problem detection time.
eventid			number	Problem event ID.
value			number	1 (always).

The following information is exported for a recovery event:

Field	Type	Description
clock	number	Number of seconds since Epoch to the moment when problem was resolved (integer part).
ns	number	Number of nanoseconds to be added to `clock` to get a precise problem resolution time.
eventid	number	Recovery event ID.
p_eventid	number	Problem event ID.
value	number	0 (always).

Examples

Problem:


```
{"hosts":["Host B","Zabbix Server"],"groups":["Group X","Group Y","Group Z","Zabbix servers"],"tags":[{"ta
```


Recovery:


```
{"clock":1519304345,"ns":987654321,"eventid":43,"p_eventid":42,"value":0}
```


Problem (multiple problem event generation):

{"hosts":["Host B","Zabbix Server"],"groups":["Group X","Group Y","Group Z","Zabbix servers"],"tags":[{"ta

{"hosts":["Host B","Zabbix Server"],"groups":["Group X","Group Y","Group Z","Zabbix servers"],"tags":[{"ta

Recovery:

{"clock":1519304346,"ns":987654321,"eventid":44,"p_eventid":43,"value":0}

{"clock":1519304346,"ns":987654321,"eventid":44,"p_eventid":42,"value":0}

Item values

The following information is exported for a collected item value:

Field		Type	Description
host		string	Visible host name of the item host.
groups		array	List of host groups of the item host; there should be at least one element in array.
	-	string	Host group name.
applications		array	List of the item applications; empty if there are none.
	-	string	Application name.
itemid		number	Item ID.
name		string	Visible item name.
clock		number	Number of seconds since Epoch to the moment when value was collected (integer part).
ns		number	Number of nanoseconds to be added to `clock` to get a precise value collection time.
timestamp (*Log* only)		number	0 if not available.
source (*Log* only)		string	Empty string if not available.
severity (*Log* only)		number	0 if not available.
logeventid (*Log* only)		number	0 if not available.
value		number (for numeric items) or string (for text items)	Collected item value.

Examples

Numeric (unsigned) value:

{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Availability"],"

Numeric (float) value:

{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["CPU","Performance"],"itemid":4,

Character, text value:

{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Installed softwa

Log value:

{"host":"Host A","groups":["Group X","Group Y","Group Z"],"applications":["Log files","Critical"],"itemid"

Trends

The following information is exported for a calculated trend value:

Field		Type	Description
host		string	Visible host name of the item host.
groups		array	List of host groups of the item host; there should be at least one element in array.
	-	string	Host group name.
applications		array	List of the item applications; empty if there are none.
	-	string	Application name.

Field	Type	Description
itemid	number	Item ID.
name	string	Visible item name.
clock	number	Number of seconds since Epoch to the moment when value was collected (integer part).
count	number	Number of values collected for a given hour.
min	number	Minimum item value for a given hour.
avg	number	Average item value for a given hour.
max	number	Maximum item value for a given hour.

Examples

Numeric (unsigned) value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Availability"],"
```

Numeric (float) value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["CPU","Performance"],"itemid":4,
```

5 监控项

5 Items

1 不同平台支持的监控项

下表列出了不同平台支持的 Zabbix agent 监控项:

- 标记为"**X**" 的监控项代表支持, 标记为"**-**" 的监控项代表不支持.
- 如果监控项标记为"**?**", 不确定是否被支持.
- 如果监控项标记为"**r**", 代表该监控项需要 root 权限.
- 中括号 **<**like_this**>** 中的参数为可选项.

Note:
只支持 Windowsd 的Zabbix agent 监控项 不在该表中.

NetBSD OpenBSD Mac OS X Tru64 AIX HP-UX Solaris FreeBSD Linux 2.6 (and later) Linux 2.4 Windows Parameter / sys- tem	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼
▼▼	1	2	3	4	5	6	7	8	9	10	11
agent.hostname	X	X	X	X	X	X	X	X	X	X	X
agent.ping	X	X	X	X	X	X	X	X	X	X	X
agent.version	X	X	X	X	X	X	X	X	X	X	X
kernel.maxfiles	-	X	X	X	-	-	-	?	X	X	X

kernel.maxproc		-	-	X	X	X	-	-	?	X	X	X
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]		X	X	X	X	X	X	X	X	X	X	X
logrt[file_format,<regexp>,<encoding>,<maxlines>,<mode>,<output>]		X	X	X	X	X	X	X	X	X	X	X
net.dns[<ip>,zone,<type>,<timeout>,<count>]		X	X	X	X	X	X	X	X	X	X	X
net.dns.record[<ip>,zone,<type>,<timeout>,<count>]		X	X	X	X	X	X	X	X	X	X	X
net.if.collisions[if]		X	X	X	X	X	-	X	-	X	X	r
net.if.discovery		X	X	X	X	X	X	X	-	-	X	X
net.if.in[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	X	-	-	X	X	r
net.if.out[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	X	-	-	-	-	-	-
net.if.total[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	X	-	-	-	-	-	-
net.tcp.listen[port]		X	X	X	X	X	-	-	-	X	-	-
net.tcp.port[<ip>,port]		X	X	X	X	X	X	X	X	X	X	X
net.tcp.service[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.tcp.service.perf[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.udp.listen[port]		X	X	X	X	X	-	-	-	X	-	-
net.udp.service[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.udp.service.perf[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]		-	X	X	-	X[3]	-	-	-	-	-	-
type	total	-	X	X	-	X	-	-	-	-	-	-
▲	*(default)*											
	user	-	X	X	-	X	-	-	-	-	-	-
	system	-	X	X	-	X	-	-	-	-	-	-
mode	avg1	-	X	X	-	X	-	-	-	-	-	-
▲	*(default)*											
	avg5	-	X	X	-	X	-	-	-	-	-	-
	avg15	-	X	X	-	X	-	-	-	-	-	-
zone	current	-	-	-	-	X	-	-	-	-	-	-
▲	*(default)*											
	all	-	-	-	-	X	-	-	-	-	-	-
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]		X	X	X	X	X[3]	X	X	-	X	X	
mode	sum	-	X	X	X	X	-	X	X	-	X	X
▲	*(default)*											
	avg	-	X	X	X	X	-	X	X	-	X	X
	max	-	X	X	X	X	-	X	X	-	X	X
	min	-	X	X	X	X	-	X	X	-	X	X
memtype		-	X	X	X	X	-	X	-	-	-	-
▲												
proc.num[<name>,<user>,<state>,<cmdline>]		X	X	X	X	X[3]	X	X	X	-	X	X

state ▲	all *(default)*	-	X	X	X	X	X	X	X	-	X	X
	sleep	-	X	X	X	X	X	X	X	-	X	X
	zomb	-	X	X	X	X	X	X	X	-	X	X
	run	-	X	X	X	X	X	X	X	-	X	X
cmdline ▲		-	X	X	X	X	X	X	X	-	X	X
sensor[device,sensor,<mode>]		-	X	X	-	-	-	-	-	-	X	-
system.boottime		-	X	X	X	X	-	-	-	X	X	X
system.cpu.discovery		X	X	X	X	X	X	X	X	X	X	X
system.cpu.intr		-	X	X	X	X	-	X	-	-	X	X
system.cpu.load[<cpu>,<mode>]		X	X	X	X	X	X	X	X	X	X	X
cpu ▲	all *(default)*	X	X	X	X	X	X	X	X	X	X	X
	percpu	X	X	X	X	X	X	X	-	X	X	X
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	X	X	X
	avg5	X	X	X	X	X	X	X	X	X	X	X
	avg15	X	X	X	X	X	X	X	X	X	X	X
system.cpu.num[<type>]		X	X	X	X	X	X	X	-	X	X	X
type ▲	online *(default)*	X	X	X	X	X	X	X	-	X	X	X
	max	-	X	X	X	X	-	-	-	X	-	-
system.cpu.switches		-	X	X	X	X	-	X	-	-	X	X
system.cpu.util[<cpu>,<type>,<mode>]		X	X	X	X	X	X	X	X	-	X	X
type ▲	user *(default)*	-	X	X	X	X	X	X	X	-	X	X
	nice	-	X	X	X	-	X	-	X	-	X	X
	idle	-	X	X	X	X	X	X	X	-	X	X
	system	X	X	X	X	X	X	X	X	-	X	X
	iowait	-	-	X	-	X	-	X	-	-	-	-
	interrupt	-	-	X	X	-	-	-	-	-	X	-
	softirq	-	-	X	-	-	-	-	-	-	-	-
	steal	-	-	X	-	-	-	-	-	-	-	-
	guest	-	-	X	-	-	-	-	-	-	-	-
	guest_nice	-	-	X	-	-	-	-	-	-	-	-
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	-	X	X
	avg5	X	X	X	X	X	X	X	-	-	X	X
	avg15	X	X	X	X	X	X	X	-	-	X	X
		1	2	3	4	5	6	7	8	9	10	11
system.hostname[<type>]		X	X	X	X	X	X	X	X	X	X	X
system.hw.chassis[<info>]		-	X	-	-	-	-	-	-	-	-	-
system.hw.cpu[<cpu>,<info>]		X	X	-	-	-	-	-	-	-	-	-
system.hw.devices[<type>]		-	X	-	-	-	-	-	-	-	-	-
system.hw.macaddr[<interface>,<format>]		-	X	-	-	-	-	-	-	-	-	-
system.localtime[<type>]		X	X	X	X	X	X	X	X	X	X	X
type ▲	utc *(default)*	X	X	X	X	X	X	X	X	X	X	X
	local	X	X	X	X	X	X	X	X	X	X	X
system.run[command,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	wait *(default)*	X	X	X	X	X	X	X	X	X	X	X
	nowait	X	X	X	X	X	X	X	X	X	X	X

system.stat[resource,<type>]		-	-	-	-	-	X	-	-	-	-	-
system.sw.arch		X	X	X	X	X	X	X	X	X	X	X
system.sw.os[<info>]		X	X	-	-	-	-	-	-	-	-	-
system.sw.packages[<package>,<manager>,<format>]		-	X	-	-	-	-	-	-	-	-	-
system.swap.in[<device>,<type>]		X	X	-	X	-	-	-	-	X	-	
(specifying a device is only supported under Linux)												
type ▲ *(pages will only work if device was not specified)*	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-
system.swap.out[<device>,<type>]		-	X	X	-	X	-	-	-	-	X	-
(specifying a device is only supported under Linux)												
type ▲ *(pages will only work if device was not specified)*	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-

system.swap.size[<device>,<type>] *(specifying a device is only supported under FreeBSD, for other platforms must be empty or "all")*		X	X	X	X	X	-	X	X	-	X	-
type ▲	free *(default)*	X	X	X	X	X	-	X	X	-	X	-
	total	X	X	X	X	X	-	X	X	-	X	-
	used	X	X	X	X	X	-	X	X	-	X	-
	pfree	X	X	X	X	X	-	X	X	-	X	-
	pused	-	X	X	X	X	-	X	X	-	X	-
system.uname		X	X	X	X	X	X	X	X	X	X	X
system.uptime		X	X	X	X	X	-	X	?	X	X	X
system.users.num		-	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
vfs.dev.read[<device>,<type>,<mode>] *(defaults are different under various OSes)*		X	X	X	X	X	-	X	-	-	X	-
type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
	operations	-	X	X	X	X	-	X	-	-	X	-
	bytes	-	-	-	X	X	-	X	-	-	X	-
	sps	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dev.write[<device>,<type>,<mode>]		X	X	X	X	X	-	X	-	-	X	-

type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
(defaults are differ-ent under vari-ous OSes)												
	operations	-	X	X	X	X	-	X	-	-	X	-
	bytes	-	-	-	X	X	-	X	-	-	X	-
	sps	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(de-fault)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.file.cksum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.contents[file,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.exists[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.md5sum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regexp[file,regexp,<encoding>,<output>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regmatch[file,regexp,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.size[file]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
vfs.file.time[file,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	modify *(de-fault)*	X	X	X	X	X	X	X	X	X	X	X
	access	X	X	X	X	X	X	X	X	X	X	X
	change	X	X	X	X	X	X	X	X	X	X	X
vfs.fs.discovery		X	X	X	X	X	X	X	-	X	X	X
vfs.fs.inode[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(de-fault)*	-	X	X	X	X	X	X	X	X	X	X
	free	-	X	X	X	X	X	X	X	X	X	X
	used	-	X	X	X	X	X	X	X	X	X	X
	pfree	-	X	X	X	X	X	X	X	X	X	X
	pused	-	X	X	X	X	X	X	X	X	X	X
vfs.fs.size[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(de-fault)*	X	X	X	X	X	X	X	X	X	X	X
	free	X	X	X	X	X	X	X	X	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pfree	X	X	X	X	X	X	X	X	X	X	X
	pused	X	X	X	X	X	X	X	X	X	X	X
vm.memory.size[<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(de-fault)*	X	X	X	X	X	X	X	X	X	X	X
	active	-	-	-	X	-	X	-	-	X	X	X

anon	-	-	-	-	-	-	-	-	-	-	X
buffers	-	X	X	X	-	-	-	-	-	X	X
cached	X	X	X	X	-	-	X	-	-	X	X
exec	-	-	-	-	-	-	-	-	-	-	X
file	-	-	-	-	-	-	-	-	-	-	X
free	X	X	X	X	X	X	X	X	X	X	X
inactive	-	-	-	X	-	-	-	-	X	X	X
pinned	-	-	-	-	-	-	X	-	-	-	-
shared	-	X	-	X	-	-	-	-	-	X	X
wired	-	-	-	X	-	-	-	-	X	X	X
used	X	X	X	X	X	X	X	X	X	X	X
pused	X	X	X	X	X	X	X	X	X	X	X
available	X	X	X	X	X	X	X	X	X	X	X
pavailable	X	X	X	X	X	X	X	X	X	X	X
web.page.get[host,<path>,<port>]				X	X	X	X	X	X	X	X
web.page.perf[host,<path>,<port>]				X	X	X	X	X	X	X	X
web.page.regexp[host,<path>,<port>,<regexp>,<length>,<output>]				X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11

<note tip> 另请参见vm.memory.size 参数说明. :::

脚注

[1] net.if.in，net.if.out 和 net.if.total 项目不提供环回接口的统计信息 (e.g. lo0).

[2] 这些项目的这些值不支持 Solaris 系统上的环回接口（包括 Solaris 10 6/06）作为字节，错误和利用率统计信息不会由内核存储和/或报告。但是，如果您通过 net snmp 监视 Solaris 系统，返回值可能是 net-snmp 携带遗留代码, 但是，如果要通过 net-snmp 监视 Solaris 系统，则可能会返回 net-snmp 携带从 1997 年开始的 cmu-snmp 的旧代码，即在读取接口统计信息字节值之后，返回后分组计数器（它存在于环回接口上）乘以任意值 308。这假设分组的平均长度为 308 个八位字节，这是非常粗略的估计，因为用于环回接口的 Solaris 系统上的 MTU 限制为 8892 字节。这些值不应该被认为是正确的，更不应该被认为是非常准确的。他们是推测值。Zabbix agent 不会做任何猜测的工作, 但是 net-snmp 会返回这些字段的一个值。

[3] Solaris 系统中, /proc/pid/psinfo 获得的命令行限制为 80 字节而且在进程启动时包含命令行。

1 Items supported by platform

The table displays support for Zabbix agent items on various platforms:

- Items marked with "**X**" are supported, the ones marked with "**-**" are not supported.
- If an item is marked with "**?**", it is not known whether it is supported or not.
- If an item is marked with "**r**", it means that it requires root privileges.
- Parameters that are included in angle brackets **<**like_this**>** are optional.

Note:
Windows-only Zabbix agent items are not included in this table.

NetBSD											
OpenBSD											▼▼
Mac										▼▼	
OS X											
Tru64									▼▼		
AIX								▼▼			
HP-UX							▼▼				
Solaris						▼▼					
FreeBSD					▼▼						
Linux				▼▼							
2.6											
(and											
later)											
Linux			▼▼								
2.4											
Windows		▼▼									

Parameter / system	▼▼	1	2	3	4	5	6	7	8	9	10	11
agent.hostname		X	X	X	X	X	X	X	X	X	X	X
agent.ping		X	X	X	X	X	X	X	X	X	X	X
agent.version		X	X	X	X	X	X	X	X	X	X	X
kernel.maxfiles		-	X	X	X	-	-	-	?	X	X	X
kernel.maxproc		-	-	X	X	X	-	-	?	X	X	X
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]		X	X	X	X	X	X	X	X	X	X	X
logrt[file_format,<regexp>,<encoding>,<maxlines>,<mode>,<output>]		X	X	X	X	X	X	X	X	X	X	X
net.dns[<ip>,zone,<type>,<timeout>,<count>]		X	X	X	X	X	X	X	X	X	X	X
net.dns.record[<ip>,zone,<type>,<timeout>,<count>]		X	X	X	X	X	X	X	X	X	X	X
net.if.collisions[if]		X	X	X	X	X	-	X	-	X	X	r
net.if.discovery		X	X	X	X	X	X	X	-	-	X	X
net.if.in[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	X	-	-	X	X	X	r
net.if.out[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	-	X	-	-	-	-	-
net.if.total[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	-	X	-	-	-	-	-
net.tcp.listen[port]		X	X	X	X	X	-	-	-	X	-	-
net.tcp.port[<ip>,port]		X	X	X	X	X	X	X	X	X	X	X
net.tcp.service[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.tcp.service.perf[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.udp.listen[port]		X	X	X	X	X	-	-	-	X	-	-
net.udp.service[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.udp.service.perf[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]		X	X	X	X	X[3]	-	-	-	-	-	-
type	total	-	X	X	-	X	-	-	-	-	-	-
▲	*(default)*											
	user	-	X	X	-	X	-	-	-	-	-	-
	system	-	X	X	-	X	-	-	-	-	-	-
mode	avg1	-	X	X	-	X	-	-	-	-	-	-
▲	*(default)*											
	avg5	-	X	X	-	X	-	-	-	-	-	-
	avg15	-	X	X	-	X	-	-	-	-	-	-
zone	current	-	-	-	-	X	-	-	-	-	-	-
▲	*(default)*											
	all	-	-	-	-	X	-	-	-	-	-	-
proc.mem[<name>,<user>,<mode>,<cmdline><memtype>]		X	X	X	X	X[3]	X	X	-	X	X	

mode ▲	sum *(default)*	-	X	X	X	X	-	X	X	-	X	X
	avg	-	X	X	X	X	-	X	X	-	X	X
	max	-	X	X	X	X	-	X	X	-	X	X
	min	-	X	X	X	X	-	X	X	-	X	X
memtype ▲		-	X	X	X	X	-	X	-	-	-	-
proc.num[<name>,<user>,<state>,<cmdline>]		X	X	X	X	X [3]	X	X	X	-	X	X
state ▲	all *(default)*	-	X	X	X	X	X	X	X	-	X	X
	disk	-	X	X	X	-	-	-	-	-	X	X
	sleep	-	X	X	X	X	X	X	X	-	X	X
	zomb	-	X	X	X	X	X	X	X	-	X	X
	run	-	X	X	X	X	X	X	X	-	X	X
	trace	-	X	X	X	-	-	-	-	-	X	X
cmdline ▲		-	X	X	X	X	X	X	X	-	X	X
sensor[device,sensor,<mode>]		X	X	X	-	-	-	-	-	-	X	-
system.boottime		-	X	X	X	X	-	-	-	X	X	X
system.cpu.discovery		X	X	X	X	X	X	X	X	X	X	X
system.cpu.intr		-	X	X	X	X	-	X	-	-	X	X
system.cpu.load[<cpu>,<mode>]		X	X	X	X	X	X	X	X	X	X	X
cpu ▲	all *(default)*	X	X	X	X	X	X	X	X	X	X	X
	percpu	X	X	X	X	X	X	X	-	X	X	X
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	X	X	X
	avg5	X	X	X	X	X	X	X	X	X	X	X
	avg15	X	X	X	X	X	X	X	X	X	X	X
system.cpu.num[<type>]		X	X	X	X	X	X	X	-	X	X	X
type ▲	online *(default)*	X	X	X	X	X	X	X	-	X	X	X
	max	-	X	X	X	X	-	-	-	X	-	-
system.cpu.switches		-	X	X	X	X	-	X	-	-	X	X
system.cpu.util[<cpu>,<type>,<mode>]		X	X	X	X	X	X	X	X	-	X	X
type ▲	user *(default)*	-	X	X	X	X	X	X	X	-	X	X
	nice	-	X	X	X	-	X	-	X	-	X	X
	idle	-	X	X	X	X	X	X	X	-	X	X
	system	X	X	X	X	X	X	X	X	-	X	X
	iowait	-	-	X	-	X	-	X	-	-	-	-
	interrupt	-	-	X	X	-	-	-	-	-	X	-
	softirq	-	-	X	-	-	-	-	-	-	-	-
	steal	-	-	X	-	-	-	-	-	-	-	-
	guest	-	-	X	-	-	-	-	-	-	-	-
	guest_nice	-	-	X	-	-	-	-	-	-	-	-
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	-	X	X
	avg5	X	X	X	X	X	X	X	-	-	X	X
	avg15	X	X	X	X	X	X	X	-	-	X	X
		1	2	3	4	5	6	7	8	9	10	11
system.hostname[<type>]		X	X	X	X	X	X	X	X	X	X	X
system.hw.chassis[<info>]		-	X	-	-	-	-	-	-	-	-	-
system.hw.cpu[<cpu>,<info>]		X	X	-	-	-	-	-	-	-	-	-
system.hw.devices[<type>]		X	X	-	-	-	-	-	-	-	-	-

system.hw.macaddr[<interface>,<format>]		X	X	-	-	-	-	-	-	-		
system.localtime[<type>]		X	X	X	X	X	X	X	X	X	X	
type ▲	utc *(default)*	X	X	X	X	X	X	X	X	X	X	
	local	X	X	X	X	X	X	X	X	X	X	
system.run[command,<mode>]		X	X	X	X	X	X	X	X	X	X	
mode ▲	wait *(default)*	X	X	X	X	X	X	X	X	X	X	
	nowait	X	X	X	X	X	X	X	X	X	X	
system.stat[resource,<type>]		-	-	-	-	X	-	-	-	-		
system.sw.arch		X	X	X	X	X	X	X	X	X	X	
system.sw.os[<info>]		X	X	-	-	-	-	-	-	-		
system.sw.packages[<package>,<manager>,<format>]		-	X	-	-	-	-	-	-	-		
system.swap.in[<device>,<type>]		X	X	-	X	-	-	-	-	X	-	
(specifying a device is only supported under Linux)												
type ▲ *(pages will only work if device was not specified)*	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-
system.swap.out[<device>,<type>]		-	X	X	-	X	-	-	-	-	X	-
(specifying a device is only supported under Linux)												
type ▲ *(pages will only work if device was not specified)*	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-

	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-
system.swap.size[<device>,<type>] *(specifying a device is only supported under FreeBSD, for other platforms must be empty or "all")*		X	X	X	X	X	-	X	X	-	X	-
type ▲	free *(default)*	X	X	X	X	X	-	X	X	-	X	-
	total	X	X	X	X	X	-	X	X	-	X	-
	used	X	X	X	X	X	-	X	X	-	X	-
	pfree	X	X	X	X	X	-	X	X	-	X	-
	pused	-	X	X	X	X	-	X	X	-	X	-
system.uname		X	X	X	X	X	X	X	X	X	X	X
system.uptime		X	X	X	X	X	-	X	?	X	X	X
system.users.num		-	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
vfs.dev.read[<device>,<type>,<mode>]		X	X	X	X	X	-	X	-	-	X	-
type ▲ *(defaults are different under various OSes)*	sectors	-	X	X	-	-	-	-	-	-	-	-
	operations	-	X	X	X	X	-	X	-	-	X	-
	bytes	-	-	-	X	X	-	X	-	-	X	-
	sps	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dev.write[<device>,<type>,<mode>]		X	X	X	X	X	-	X	-	-	X	-

type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
(defaults are differ-ent under vari-ous OSes)												
	operations	-	X	X	X	X	-	X	-	-	X	-
	bytes	-	-	-	X	X	-	X	-	-	X	-
	sps	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(de-fault)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>]		?	?	?	?	?	?	?	?	?	?	?
vfs.file.cksum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.contents[file,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.exists[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.md5sum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regexp[file,regexp,<encoding>,<output>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regmatch[file,regexp,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.size[file]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
vfs.file.time[file,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	modify *(de-fault)*	X	X	X	X	X	X	X	X	X	X	X
	access	X	X	X	X	X	X	X	X	X	X	X
	change	X	X	X	X	X	X	X	X	X	X	X
vfs.fs.discovery		X	X	X	X	X	X	X	-	X	X	X
vfs.fs.inode[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(de-fault)*	-	X	X	X	X	X	X	X	X	X	X
	free	-	X	X	X	X	X	X	X	X	X	X
	used	-	X	X	X	X	X	X	X	X	X	X
	pfree	-	X	X	X	X	X	X	X	X	X	X
	pused	-	X	X	X	X	X	X	X	X	X	X
vfs.fs.size[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(de-fault)*	X	X	X	X	X	X	X	X	X	X	X
	free	X	X	X	X	X	X	X	X	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pfree	X	X	X	X	X	X	X	X	X	X	X
	pused	X	X	X	X	X	X	X	X	X	X	X
vm.memory.size[<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(de-fault)*	X	X	X	X	X	X	X	X	X	X	X

active	-	-	-	X	-	X	-	-	X	X	X
anon	-	-	-	-	-	-	-	-	-	-	X
buffers	-	X	X	X	-	-	-	-	-	X	X
cached	X	X	X	X	-	-	X	-	-	X	X
exec	-	-	-	-	-	-	-	-	-	-	X
file	-	-	-	-	-	-	-	-	-	-	X
free	X	X	X	X	X	X	X	X	X	X	X
inactive	-	-	-	X	-	-	-	-	X	X	X
pinned	-	-	-	-	-	-	X	-	-	-	-
shared	-	X	-	X	-	-	-	-	-	X	X
wired	-	-	-	X	-	-	-	-	X	X	X
used	X	X	X	X	X	X	X	X	X	X	X
pused	X	X	X	X	X	X	X	X	X	X	X
available	X	X	X	X	X	X	X	X	X	X	X
pavailable	X	X	X	X	X	X	X	X	X	X	X
web.page.get[host,<path>,<port>]	X	X	X	X	X	X	X	X	X	X	X
web.page.perf[host,<path>,<port>]	X	X	X	X	X	X	X	X	X	X	X
web.page.regexp[host,<path>,<port>,<regexp>,<length>,<output>]	X	X	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11

Note:
See also a description of vm.memory.size parameters.

Footnotes

[1] net.if.in, net.if.out and net.if.total items do not provide statistics of loopback interfaces (e.g. lo0).

[2] These values for these items are not supported for loopback interfaces on Solaris systems up to and including Solaris 10 6/06 as byte, error and utilisation statistics are not stored and/or reported by the kernel. However, if you're monitoring a Solaris system via net-snmp, values may be returned as net-snmp carries legacy code from the cmu-snmp dated as old as 1997 that, upon failing to read byte values from the interface statistics returns the packet counter (which does exist on loopback interfaces) multiplied by an arbitrary value of 308. This makes the assumption that the average length of a packet is 308 octets, which is a very rough estimation as the MTU limit on Solaris systems for loopback interfaces is 8892 bytes.

These values should not be assumed to be correct or even closely accurate. They are guestimates. The Zabbix agent does not do any guess work, but net-snmp will return a value for these fields.

[3] The command line on Solaris, obtained from /proc/pid/psinfo, is limited to 80 bytes and contains the command line as it was when the process was started.

2 参数 **vm.memory.size**

- **total** - 总物理内存.
- **free** - 可用内存.
- **active** - 内存当前使用或最近使用，所以它在 RAM 中。
- **inactive** - 未使用内存.
- **wired** - 被标记为始终驻留在 RAM 中的内存，不会移动到磁盘。
- **pinned** - 和'wired' 一样。
- **anon** - 与文件无关的内存 (不能重新读取)。
- **exec** - 可执行代码，通常来自于一个 (程序) 文件。
- **file** - 缓存最近访问文件的目录。
- **buffers** - 缓存文件系统元数据。
- **cached** - 缓存为不同事情。
- **shared** - 可以同时被多个进程访问的内存。
- **used** - active + wired 内存。
- **pused** - active + wired 总内存的百分比。
- **available** - inactive + cached + free 内存。
- **pavailable** - inactive + cached + free memory 占'total' 的百分比。

Attention:
vm.memory.size[used] 和 *vm.memory.size[available]* 的和不是必需等于总内存。例如，在 FreeBSD 中 active, inactive, wired, cached 被认为是使用的内存，因为他们存储一些有用的信息。
同样，inactive, cached, free 也被认为是可用内存，因为这些内存可以立即被分配给需要更多内存的线程。
所以不活动的内存是同时可以是使用和可用的。正因为如此，item *vm.memory.size[used]* 只用来获得信息，监控项 *vm.memory.size[available]* 在触发器中使用。

Note:
参看本页底部" 另外见" 部分关于在不同的操作系统中内存计算的更多详细信息。

特定系统的注意事项

- 在 Solaris 中 **available** and **free** 是一样的。
- 在 Linux 中 **shared** 只在 kernel 2.4 中起作用。

另见

1. 关于不同操作系统内存计算的详细信息

2 vm.memory.size parameters

Overview

This section provides more details and platform-specific information on the parameters of the `vm.memory.size[<mode>]` agent item.

Parameters

The following parameters are possible for this item:

- **active** - memory currently in use or very recently used, and so it is in RAM
- **anon** - memory not associated with a file (cannot be re-read from it)
- **available** - available memory, calculated differently depending on the platform (see the table below)
- **buffers** - cache for things like file system metadata
- **cached** - cache for various things
- **exec** - executable code, typically from a (program) file
- **file** - cache for contents of recently accessed files
- **free** - memory that is readily available to any entity requesting memory
- **inactive** - memory that is marked as not used
- **pavailable** - inactive + cached + free memory as percentage of 'total'
- **pinned** - same as 'wired'
- **pused** - active + wired memory as percentage of 'total'
- **shared** - memory that may be simultaneously accessed by multiple processes
- **slab** - total amount of memory used by the kernel to cache data structures for its own use
- **total** - total physical memory available
- **used** - used memory, calculated differently depending on the platform (see the table below)
- **wired** - memory that is marked to always stay in RAM. It is never moved to disk.

Platform-specific calculation of **available** and **used**:

Platform	**"available"**	**"used"**
AIX	free + cached	real memory in use
FreeBSD	inactive + cached + free	active + wired + cached
HP UX	free	total - free
Linux<3.14	free + buffers	total - free
Linux 3.14+	/proc/meminfo, "Cached":+"MemAvailable:"	total - free
NetBSD	inactive + execpages + file + free	total - free
OpenBSD	inactive + free + cached	active + wired
OSX	inactive + free	active + wired
Solaris	free	total - free
Win32	free	total - free

Attention:
The sum of *vm.memory.size[used]* and *vm.memory.size[available]* does not necessarily equal total. For instance, on FreeBSD:
* Active, inactive, wired, cached memories are considered used, because they store some useful information.
* At the same time inactive, cached, free memories are considered available, because these kinds of memories can be given instantly to processes that request more memory.

So inactive memory is both used and available simultaneously. Because of this, the *vm.memory.size[used]* item is designed for informational purposes only, while *vm.memory.size[available]* is designed to be used in triggers.

See the "See also" section at the bottom of this page to find more detailed information about memory calculation in different OS.

Platform-specific notes

- on Linux **shared** works only on kernel 2.4

See also

1. Detailed information about memory calculation in different OS

3 被动和主动代理检查

概述

本节提供关于 Zabbix 代理执行的被动和主动检查的详细信息。

Zabbix 使用一个基于 JSON 的通信协议来与 Zabbix 代理进行通信。

被动检查

被动检查是一个简单的数据请求。Zabbix 服务器或 proxy 请求一些数据 (例如，CPU 负载)，Zabbix agent 将结果发送回服务器。

Server 请求

```
<item key>\n
```

Agent 响应

```
<DATA>[\0<ERROR>]
```

在上面，方括号中的部分是可选的，只发送到不受支持的项目。

例如, 对于支持的监控项:

1. Server 打开一个 TCP 连接
2. Server 发送 **agent.ping\n**
3. Agent 读取请求并响应 **<HEADER><DATALEN>1**
4. Server 处理数据以获取值, '1' in our case
5. TCP 连接关闭

对于不支持的监控项:

1. Server 打开一个 TCP 连接
2. Server 发送 **vfs.fs.size[/nono]\n**
3. Agent 读取请求并响应 **<HEADER><DATALEN>ZBX_NOTSUPPORTED\0Cannot obtain filesystem information: [2] No such file or directory**
4. Server 处理数据, 更改项目状态为不支持并显示指定的错误消息
5. TCP 连接关闭

主动检查

主动检查需要更复杂的处理，agent 必须首先从 server 端检索监控项的列表以便独立处理。

主动检查的 zabbix server 列表在 agent 配置文件中的'ServerActive' 参数中配置，检查的频率是由相同配置文件中的'RefreshActiveChecks' 参数设置。主动检查失败后会在 60 秒后重试。

agent 定期向 zabbix server 发送新值。

Note:
如果代理位于防火墙后面，则可以考虑仅使用主动检查，因为在这种情况下，不需要修改防火墙以允许初始传入连接。

获取监控项列表

Agent 请求

```
{
    "request":"active checks",
    "host":"<hostname>"
}
```

Server 响应

```
{
    "response":"success",
    "data":[
        {
            "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
            "delay":30,
            "lastlogsize":0,
            "mtime":0
        },
        {
            "key":"agent.version",
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        },
        {
            "key":"vfs.fs.size[/nono]",
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        }
    ]
}
```

服务器必须响应成功。对于每一个返回的监控项, 不管监控项是不是日志监控项，必须存在 **key**, **delay**, **lastlogsize** and **mtime** 。

例如:

1. Agent 打开一个 TCP 连接
2. Agent 请求检查清单
3. Server 响应为监控项列表 (item key, delay)
4. Agent 解析响应
5. TCP 关闭连接
6. Agent 开始定期收集数据

<note important> 注意，在使用主动检查时，有权访问 Zabbix 服务器 trapper 端口的第三方程序可能会获得（敏感）配置数据，因为任何一个程序都可以假装是一个主动 agent，并请求项目配置数据; 除非你使用加密 选项，否则认证不会发生:::

发送收集的数据

Agent 发送

```
{
    "request":"agent data",
    "session": "12345678901234567890123456789012",
    "data":[
        {
            "host":"<hostname>",
            "key":"agent.version",
            "value":"2.4.0",
            "id": 1,
            "clock":1400675595,
            "ns":76808644
        },
        {
            "host":"<hostname>",
            "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
            "lastlogsize":112,
            "value":" 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision
```

```
            "id": 2,
            "clock":1400675595,
            "ns":77053975
        },
        {
            "host":"<hostname>",
            "key":"vfs.fs.size[/nono]",
            "state":1,
            "value":"Cannot obtain filesystem information: [2] No such file or directory",
            "id": 3,
            "clock":1400675595,
            "ns":78154128
        }
    ],
    "clock": 1400675595,
    "ns": 78211329
}
```

为每个值分配一个虚拟 ID。Value ID 是一个简单的递增计数器，在一个数据会话中是唯一的（由会话令牌标识）。此 ID 用于丢弃在连接不良环境中可能发送的重复值。**Server** 响应

```
{
    "response":"success",
    "info":"processed: 3; failed: 0; total: 3; seconds spent: 0.003534"
}
```

<note important> 如果在服务器上发送一些值失败 (例如，因为主机或监控项被禁用或删除)，agnet 将不会重试发送这些值。:::

例如:

1. Agent 打开一个 TCP 连接
2. Agent 发送一个值列表
3. Server 处理数据并将状态返回
4. TCP 连接关闭

注意，上面例子中 vfs.fs.size[/nono] 状态为 not supported，因为”state” 值为 1 和”value” 中的错误消息。

<note important> 在服务器端，错误消息最大长度为 2048 个字符。:::

老的 XML 协议

> **Note:**
> Zabbix 将占用 16 MB 的 XML base64 编码的数据, 但单个解码值应该不超过 64kb，否则，在解码时将被截断到 64 KB。

4 捕捉器监控项

概述

Zabbix 服务器使用基于 JSON 的通信协议，使用trapper item从 Zabbix sender 接收数据。

请求和响应消息必须以header and data length开头。

Zabbix sender 请求

```
{
    "request":"sender data",
    "data":[
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value"
        }
    ]
}
```

Zabbix 服务器响应

```
{
    "response":"success",
    "info":"processed: 1; failed: 0; total: 1; seconds spent: 0.060753"
}
```

Zabbix sender 可以发送带有时间戳的请求

```
{
    "request":"sender data",
    "data":[
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value",
            "clock":1516710794
        },
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value",
            "clock":1516710795
        }
    ],
    "clock":1516712029,
    "ns":873386094
}
```

Zabbix 服务器响应

```
{
    "response":"success",
    "info":"processed: 2; failed: 0; total: 2; seconds spent: 0.060904"
}
```

5 返回值的编码

Zabbix server 期望每个返回的文本值都是 UTF8 编码的，这涉及每一种类型的检查: zabbix agent, ssh, telnet 等等。

不同的监视系统/设备和检查的返回值中可能有非 ascii 字符。对于这种情况，几乎所有的 zabbix keys 都包含一个额外的 item key 参数 * * <encoding> * *。这个关键参数是可选的，但是如果返回的值不是 UTF8 编码，并且它包含非 ascii 字符，则应该指定它。否则，结果可能是出乎意料的和不可预测的。

在这种情况下，对不同数据库后台的行为描述如下。

MySQL

如果一个值在非 UTF8 编码中包含非 ascii 字符，那么当数据库存储此值时，该字符及该字符后的值将被丢弃。没有警告信息写入 *zabbix_server.log*.
Relevant for at least MySQL version 5.1.61

PostgreSQL

如果一个值在非 UTF8 编码中包含非 ascii 字符—这将导致一个失败的 SQL 查询 (PGRES_FATAL_ERROR: 编码的无效字节序列) 和数据将不会被存储。会向 *zabbix_server.log* 中写入一个适当的警告消息.
Relevant for at least PostgreSQL version 9.1.3

5 Encoding of returned values

Zabbix server expects every returned text value in the UTF8 encoding. This is related to any type of checks: zabbix agent, ssh, telnet, etc.

Different monitored systems/devices and checks can return non-ASCII characters in the value. For such cases, almost all possible zabbix keys contain an additional item key parameter - **<encoding>**. This key parameter is optional but it should be specified if the returned value is not in the UTF8 encoding and it contains non-ASCII characters. Otherwise the result can be unexpected and unpredictable.

A description of behavior with different database back-ends in such cases follows.

MySQL

If a value contains a non-ASCII character in non UTF8 encoding - this character and the following will be discarded when the database stores this value. No warning messages will be written to the *zabbix_server.log*.
Relevant for at least MySQL version 5.1.61

PostgreSQL

If a value contains a non-ASCII character in non UTF8 encoding - this will lead to a failed SQL query (PGRES_FATAL_ERROR:ERROR invalid byte sequence for encoding) and data will not be stored. An appropriate warning message will be written to the *zabbix_server.log*.
Relevant for at least PostgreSQL version 9.1.3

6 大文件支持

大型文件支持，通常缩写为 LFS, 这个术语适用于在 32 位操作系统上处理大于 2 GB 的文件的能力。自从 Zabbix 2.0 对大文件的支持已经被添加。该变动会影响log file monitoring 和所有vfs.file.* items. 大文件支持依赖于 Zabbix 编译时系统的性能，但是在 32 位 Solaris 上完全禁用，因为它与 procfs 和 swapctl 不兼容。

6 Large file support

Large file support, often abbreviated to LFS, is the term applied to the ability to work with files larger than 2 GB on 32-bit operating systems. Since Zabbix 2.0 support for large files has been added. This change affects at least log file monitoring and all vfs.file.* items. Large file support depends on the capabilities of a system at Zabbix compilation time, but is completely disabled on a 32-bit Solaris due to its incompatibility with procfs and swapctl.

7 传感器

每个传感器芯片在 sysfs /sys/devices 都有自己的目录。要找到所有的传感器芯片，从/sys/class/hwmon/hwmon* 跟踪设备的符号链接更容易, 这里 * is 是个数字 (0,1,2,...)。

对于虚拟设备，传感器读数在 /sys/class/hwmon/hwmon*/ 目录, 对于非虚拟设备，传感器读数在 /sys/class/hwmon/hwmon*/device 目录。hwmon* 或 hwmon*/device 目录中一个叫 name 的文件包含该芯片的名称, 它对应于传感器芯片所使用的内核驱动程序的名称。

每个文件只有一个传感器读取值。在上面提到的目录中包含传感器读数的文件的命令常用方案是: <type><number>_<item>, 这里

- **type** - 对于传感器芯片：”in” (电压), ”temp” (温度), ”fan” (风扇), 等，
- **item** - ”input” (测量值), ”max” (高阈值), ”min” (低阈值), 等,
- **number** - 总是用于可以不止一次出现的元素 (经常从 1 开始, 除了电压从 0 开始)，如果文件不引用特定的元素，则它们的名称简单，没有数字。

可以通过 **sensor-detect** 和 **sensors** 工具获取主机上可用的传感器信息 (lm-sensors package: http://lm-sensors.org/)。**Sensors-detect** 帮助确定哪些模块对于可用的传感器是必需的。当模块加载 **sensors** 程序时可以用来显示所有传感器芯片的读数。该程序使用的传感器读数的标记可以和常规的命名方案不同 (<type><number>_<item>):

- 如果有一个名为 <type><number>_label 的文件, 那么该文件中的标签会代替 <type><number><item> 名字;
- 如果没有名为 <type><number>_label 的文件, 那么程序会在 /etc/sensors.conf (也许会为/etc/sensors3.conf, 或其他的) 文件中找 name 的替代标签。

这个标签允许用户决定使用什么样的硬件。如果既没有 <type><number>_label 文件，配置文件中也没有 label ，那么硬件的类型可以由分配的名字 (hwmon*/device/name) 决定。zabbix_agent 接受的传感器的实际名称可以通过运行 **sensors** 程序带着 -u 参数 (**sensors -u**).

在 **sensor** 程序中，可用的传感器被总线类型 (ISA 适配器，PCI 适配器，SPI 适配器，虚拟设备，ACPI 接口，HID 适配器) 分开。

Linux 2.4:

(传感器读数从/ proc/sys/dev/sensor 目录获得)

- **device** - 设备名字 (如果使用了 <mode>，则是正则表达式);
- **sensor** - 传感器名字 (如果使用了 <mode>，则是正则表达式);
- **mode** - 可能的值: avg, max, min (如果忽略了这个参数，设备和传感器将逐字处理)。

例子: sensor[w83781d-i2c-0-2d,temp1]

在 Zabbix 1.8.4 之前，使用了 sensor[temp1] 格式。

Linux 2.6+:

(传感器读数从/ sys / class / hwmon 目录获得)

- **device** - 设备名称 (非正则表达式)。设备名称可以是设备的实际名称 (e.g 0000:00:18.3) 或使用传感器程序获取得的名称 (例如:k8temp-pci-00c3)，这由用户决定使用哪个名称;

- **sensor** - 传感器名称 (非正则表达式);
- **mode** - 可能的值: avg, max, min (如果忽略了这个参数，设备和传感器将逐字处理)。

例如:

sensor[k8temp-pci-00c3,temp, max] 或 sensor[0000:00:18.3,temp1]

sensor[smsc47b397-isa-0880,in, avg] 或 sensor[smsc47b397.2176,in1]

获取传感器的名字

传感器标签，由 *sensors* 命令打印，不能总是被直接使用，因为标签的命名对于每个传感器芯片供应商来说可能是不同的。例如, *sensors* 输出可能包含以下几行:

```
$ sensors
in0:          +2.24 V  (min =  +0.00 V, max =  +3.32 V)
Vcore:        +1.15 V  (min =  +0.00 V, max =  +2.99 V)
+3.3V:        +3.30 V  (min =  +2.97 V, max =  +3.63 V)
+12V:        +13.00 V  (min =  +0.00 V, max = +15.94 V)
M/B Temp:     +30.0°C  (low  = -127.0°C, high = +127.0°C)
```

在这些情况下，只有一个标签可以直接使用:

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in0]
2.240000
```

尝试使用其他标签 (像 *Vcore* 或 *+12V*) 是不会起作用的。

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,Vcore]
ZBX_NOTSUPPORTED
```

为了找到实际的 Zabbix 可以使用它来检索读数的传感器名称，运行 *sensors -u* 命令。在输出中，可以看到到以下内容:

```
$ sensors -u
...
Vcore:
  in1_input: 1.15
  in1_min: 0.00
  in1_max: 2.99
  in1_alarm: 0.00
...
+12V:
  in4_input: 13.00
  in4_min: 0.00
  in4_max: 15.94
  in4_alarm: 0.00
...
```

所有 *Vcore* 应该检索 *in1*,*+12V* 应该检索 *in4*.[1]

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in1]
1.301000
```

不止电压 (in), 还有电流 (curr), 温度 (temp) 和风扇转速 (fan) 的读数都可以被 Zabbix 检索到。

7 Sensor

Each sensor chip gets its own directory in the sysfs /sys/devices tree. To find all sensor chips, it is easier to follow the device symlinks from /sys/class/hwmon/hwmon*, where * is a real number (0,1,2,...).

The sensor readings are located either in /sys/class/hwmon/hwmon*/ directory for virtual devices, or in /sys/class/hwmon/hwmon*/device directory for non-virtual devices. A file, called name, located inside hwmon* or hwmon*/device directories contains the name of the chip, which corresponds to the name of the kernel driver used by the sensor chip.

There is only one sensor reading value per file. The common scheme for naming the files that contain sensor readings inside any of the directories mentioned above is: <type><number>_<item>, where

- **type** - for sensor chips is ”in” (voltage), ”temp” (temperature), ”fan” (fan), etc.,
- **item** - ”input” (measured value), ”max” (high threshold), ”min” (low threshold), etc.,

[1] 例如，一个简单的触发器 `{host:item.timeleft(1h,,X)} < 1h` 当监控项值接近 X 时可能进入异常状态，然后一旦达到值 X 就突然恢复。如果异常是监控项值低于 X，请使用：`{host:item.last()} < X or {host:item.timeleft(1h,,X)} < 1h` 如果异常是项目值高于 X，请使用：`{host:item.last()} > X or {host:item.timeleft(1h,,X)} < 1h`

- **number** - always used for elements that can be present more than once (usually starts from 1, except for voltages which start from 0). If files do not refer to a specific element they have a simple name with no number.

The information regarding sensors available on the host can be acquired using **sensor-detect** and **sensors** tools (lm-sensors package: http://lm-sensors.org/). **Sensors-detect** helps to determine which modules are necessary for available sensors. When modules are loaded the **sensors** program can be used to show the readings of all sensor chips. The labeling of sensor readings, used by this program, can be different from the common naming scheme (<type><number>_<item>):

- if there is a file called <type><number>_label, then the label inside this file will be used instead of <type><number><item> name;
- if there is no <type><number>_label file, then the program searches inside the /etc/sensors.conf (could be also /etc/sensors3.conf, or different) for the name substitution.

This labeling allows user to determine what kind of hardware is used. If there is neither <type><number>_label file nor label inside the configuration file the type of hardware can be determined by the name attribute (hwmon*/device/name). The actual names of sensors, which zabbix_agent accepts, can be obtained by running **sensors** program with -u parameter (**sensors -u**).

In **sensor** program the available sensors are separated by the bus type (ISA adapter, PCI adapter, SPI adapter, Virtual device, ACPI interface, HID adapter).

On Linux 2.4:

(Sensor readings are obtained from /proc/sys/dev/sensors directory)

- **device** - device name (if <mode> is used, it is a regular expression);
- **sensor** - sensor name (if <mode> is used, it is a regular expression);
- **mode** - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).

Example key: sensor[w83781d-i2c-0-2d,temp1]

Prior to Zabbix 1.8.4, the sensor[temp1] format was used.

On Linux 2.6+:

(Sensor readings are obtained from /sys/class/hwmon directory)

- **device** - device name (non regular expression). The device name could be the actual name of the device (e.g 0000:00:18.3) or the name acquired using sensors program (e.g. k8temp-pci-00c3). It is up to the user to choose which name to use;
- **sensor** - sensor name (non regular expression);
- **mode** - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).

Example key:

sensor[k8temp-pci-00c3,temp, max] or sensor[0000:00:18.3,temp1]

sensor[smsc47b397-isa-0880,in, avg] or sensor[smsc47b397.2176,in1]

Obtaining sensor names

Sensor labels, as printed by the *sensors* command, cannot always be used directly because the naming of labels may be different for each sensor chip vendor. For example, *sensors* output might contain the following lines:

```
$ sensors
in0:          +2.24 V  (min =  +0.00 V, max =  +3.32 V)
Vcore:        +1.15 V  (min =  +0.00 V, max =  +2.99 V)
+3.3V:        +3.30 V  (min =  +2.97 V, max =  +3.63 V)
+12V:        +13.00 V  (min =  +0.00 V, max = +15.94 V)
M/B Temp:     +30.0°C  (low  = -127.0°C, high = +127.0°C)
```

Out of these, only one label may be used directly:

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in0]
2.240000
```

Attempting to use other labels (like *Vcore* or *+12V*) will not work.

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,Vcore]
ZBX_NOTSUPPORTED
```

To find out the actual sensor name, which can be used by Zabbix to retrieve the sensor readings, run *sensors -u*. In the output, the following may be observed:

```
$ sensors -u
...
Vcore:
```

```
  in1_input: 1.15
  in1_min: 0.00
  in1_max: 2.99
  in1_alarm: 0.00
...
+12V:
  in4_input: 13.00
  in4_min: 0.00
  in4_max: 15.94
  in4_alarm: 0.00
...
```

So *Vcore* should be queried as *in1*, and *+12V* should be queried as *in4*.[2]

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in1]
1.301000
```

Not only voltage (in), but also current (curr), temperature (temp) and fan speed (fan) readings can be retrieved by Zabbix.

8 proc.mem 监控项中 **memtype** 参数类型的注意事项

概述

Linux, AIX, FreeBSD 和 Solaris 都支持 **memtype** 参数。

'memtype' 参数的三个常用值 `pmem`, `rss` 和 `vsize` 在所有系统中都适用。另外, 在一些系统中只支持该系统下的'memtype' 值。

AIX

请参见表中 AIX 上的 "memtype" 参数所支持的值.

支持的参数值描述	Source	n procentry64 structure Tries t	be compatible with
vsize ((- default value))	虚拟内存大小 pi_s	ze	
pmem	实际内存的百分比 pi_prm	ps -o p	em
rss	驻留集大小 pi_	rss + pi_drss ps -	rssize
size	进程大小 (代码 + 数据) pi_dvm	"ps gvw	SIZE column
dsize	数据大小 pi	dsize	
tsize	文本 (代码) 的大小 pi_ts	ze "ps gv	" TSIZ column
sdsize	来自共享库的数据大小 pi_sdsiz		
drss	数据驻留集大小 pi_dr	s	
trss	文本驻留集大小 pi_tr	s	

FreeBSD

请参见表中 FreeBSD 上的 "memtype" 参数支持的值。

Supported value	Description	Source in kinfo_proc structure	Tries to be compatible with
vsize	虚拟内存大小 kp_e	roc.e_vm.vm_map.size or ki_size ps -o	vsz
pmem	实际内存的百分比 calcula	ed from rss ps -o p	em
rss	驻留集大小 kp_	proc.e_vm.vm_rssize or ki_rssize ps -	rss
size ((- default value))	进程 (代码 + 数据 + 堆栈) 大小 tsize + d	ize + ssize	
tsize	文本 (代码) 的大小 kp_ep	oc.e_vm.vm_tsize or ki_tsize ps -o	siz
dsize	数据大小 kp	eproc.e_vm.vm_dsize or ki_dsize ps	o dsiz
ssize	堆栈大小 kp	eproc.e_vm.vm_ssize or ki_ssize ps	o ssiz

[2] 多项式度可以是从 1 到 6, 多项式 *1* 等于线性。然而，谨慎使用更高阶多项式with caution。如果评估周期包含比确定多项式系数所需的更少的点数，则多项式度将降低。(例如请求多项式 *5*，但只有 4 点，因此多项式 *3* 更合适)。

Linux

请参见表中 Linux 上的 “memtype” 参数支持的值。

Supported value	Description	Source in /proc/<pid>/status file
vsize ((- default value))	虚拟内存大小 VmSiz	
pmem	实际内存的百分比 (VmRSS/	otal_memory) * 100
rss	驻留集大小 VmRS	
data	数据段的大小 VmDat	
exe	代码段的大小 VmExe	
hwm	驻留集峰值大小 VmHWM	
lck	锁定内存大小 VmLck	
lib	共享库的大小 VmLib	
peak	虚拟内存峰值大小 VmPeak	
pin	固定的页面大小 VmPin	
pte	页表条目的大小 VmPTE	
size	进程码 + 数据 + 栈段大小 VmExe +	mData + VmStk
stk	堆栈段大小 VmSt	
swap	使用的交换空间大小 VmSwap	

Linux 上注意事项:

1. 一些旧版本 Linux 内核并不是支持所有’memtype’ 值的。例如, Linux 内核版本 2.4 就不支持 `hwm`, `pin`, `peak`, `pte` 和 `swap` 等值。
2. 我们发现 Zabbix agent 主动检查进程参数 `proc.mem[...,...,...,...,data]` 显示的值比 agent 的 /proc/<pid>/status 文件中 `VmData` 行的值大大 4 kB。在 agent 自我监控管理时，agent 的数据碎片增长率 4 kB ，然后又返回到先前的值。

Solaris

请参见表中的 Solaris 上的 “memtype” 参数所支持的值。

支持的参数值描述	Source	n psinfo structure 兼容	
vsize ((- default value))	Size of process image	pr_size	ps -o vsz
pmem	实际内存的百分比 pr_pct	em ps -o p	em
rss	驻留集大小 pr_ 可能会被低估 - 参看”man ps” 中 rss 描述.	ssize ps -	rss

8 Notes on memtype parameter in proc.mem items

Overview

The **memtype** parameter is supported on Linux, AIX, FreeBSD, and Solaris platforms.

Three common values of ’memtype’ are supported on all of these platforms: `pmem`, `rss` and `vsize`. Additionally, platform-specific ’memtype’ values are supported on some platforms.

AIX

See values supported for ’memtype’ parameter on AIX in the table.

Supported value	Description	Source in procentry64 structure	Tries to be compatible with
vsize ((- default value))	Virtual memory size	pi_size	
pmem	Percentage of real memory	pi_prm	ps -o pmem
rss	Resident set size	pi_trss + pi_drss	ps -o rssize
size	Size of process (code + data)	pi_dvm	”ps gvw” SIZE column
dsize	Data size	pi_dsize	
tsize	Text (code) size	pi_tsize	”ps gvw” TSIZ column
sdsize	Data size from shared library	pi_sdsize	
drss	Data resident set size	pi_drss	

Supported value	Description	Source in procentry64 structure	Tries to be compatible with
trss	Text resident set size	pi_trss	

FreeBSD

See values supported for ’memtype’ parameter on FreeBSD in the table.

Supported value	Description	Source in kinfo_proc structure	Tries to be compatible with
vsize	Virtual memory size	kp_eproc.e_vm.vm_map.size or ki_size	ps -o vsz
pmem	Percentage of real memory	calculated from rss	ps -o pmem
rss	Resident set size	kp_eproc.e_vm.vm_rssize or ki_rssize	ps -o rss
size ((- default value))	Size of process (code + data + stack)	tsize + dsize + ssize	
tsize	Text (code) size	kp_eproc.e_vm.vm_tsize or ki_tsize	ps -o tsiz
dsize	Data size	kp_eproc.e_vm.vm_dsize or ki_dsize	ps -o dsiz
ssize	Stack size	kp_eproc.e_vm.vm_ssize or ki_ssize	ps -o ssiz

Linux

See values supported for ’memtype’ parameter on Linux in the table.

Supported value	Description	Source in /proc/<pid>/status file
vsize ((- default value))	Virtual memory size	VmSize
pmem	Percentage of real memory	(VmRSS/total_memory) * 100
rss	Resident set size	VmRSS
data	Size of data segment	VmData
exe	Size of code segment	VmExe
hwm	Peak resident set size	VmHWM
lck	Size of locked memory	VmLck
lib	Size of shared libraries	VmLib
peak	Peak virtual memory size	VmPeak
pin	Size of pinned pages	VmPin
pte	Size of page table entries	VmPTE
size	Size of process code + data + stack segments	VmExe + VmData + VmStk
stk	Size of stack segment	VmStk
swap	Size of swap space used	VmSwap

Notes for Linux:

1. Not all ’memtype’ values are supported by older Linux kernels. For example, Linux 2.4 kernels do not support `hwm`, `pin`, `peak`, `pte` and `swap` values.
2. We have noticed that self-monitoring of the Zabbix agent active check process with `proc.mem[...,...,...,...,data]` shows a value that is 4 kB larger than reported by `VmData` line in the agent’s /proc/<pid>/status file. At the time of self-measurement the agent’s data segment increases by 4 kB and then returns to the previous size.

Solaris

See values supported for ’memtype’ parameter on Solaris in the table.

Supported value	Description	Source in psinfo structure	Tries to be compatible with
vsize ((- default value))	Size of process image	pr_size	ps -o vsz
pmem	Percentage of real memory	pr_pctmem	ps -o pmem

Supported value	Description	Source in psinfo structure	Tries to be compatible with
rss	Resident set size It may be un-derestimated - see rss description in ”man ps”.	pr_rssize	ps -o rss

9 在 **proc.mem** 和 **proc.num** 项目中选择进程的注意事项

Processes modifying their commandline

一些程序使用修改它们的命令行作为显示当前活动的方法。用户可以通过运行 ps 和 top 命令来查看活动。这些程序的例子包括 *PostgreSQL*, *Sendmail*, *Zabbix*.

让我们来看一个 Linux 的例子，假设我们想要监视许多 Zabbix 代理进程。

ps 命令显示的进程如下

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6318     1  0 12:01 ?        00:00:00 sbin/zabbix_agentd -c /home/zabbix/ZBXNEXT-1078/zabbix_age
zabbix    6319  6318  0 12:01 ?        00:00:01 sbin/zabbix_agentd: collector [idle 1 sec]
zabbix    6320  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #1 [waiting for connection]
zabbix    6321  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #2 [waiting for connection]
zabbix    6322  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #3 [waiting for connection]
zabbix    6323  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: active checks #1 [idle 1 sec]
...
```

通过名称和用户选择进程来完成任务:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
6
```

现在让我们将 zabbix_agentd 重命名为 zabbix_agentd_30 并重新启动它。

ps 现在显示为

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6715     1  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30 -c /home/zabbix/ZBXNEXT-1078/zabbix_
zabbix    6716  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: collector [idle 1 sec]
zabbix    6717  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #1 [waiting for connection
zabbix    6718  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #2 [waiting for connection
zabbix    6719  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #3 [waiting for connection
zabbix    6720  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]
...
```

现在根据名称和用户选择进程会产生不正确的结果:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
1
```

为什么将可执行文件重命名为更长的名称会导致完全不同的结果?

Zabbix agent 启动时检查进程名字，/proc/<pid>/status 文件是打开的并且检查 Name 行。我们的例子中 Name 行如下:

```
$ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
/proc/6715/status:Name:   zabbix_agentd_3
/proc/6716/status:Name:   zabbix_agentd_3
/proc/6717/status:Name:   zabbix_agentd_3
/proc/6718/status:Name:   zabbix_agentd_3
/proc/6719/status:Name:   zabbix_agentd_3
/proc/6720/status:Name:   zabbix_agentd_3
```

status 文件中的进程名会被截断为 15 个字符。

ps 命令会产生相似的结果:

```
$ ps -u zabbix
  PID TTY          TIME CMD
...
 6715 ?        00:00:00 zabbix_agentd_3
 6716 ?        00:00:01 zabbix_agentd_3
 6717 ?        00:00:00 zabbix_agentd_3
 6718 ?        00:00:00 zabbix_agentd_3
 6719 ?        00:00:00 zabbix_agentd_3
 6720 ?        00:00:00 zabbix_agentd_3
 ...
```

显然, 跟我们的 proc.num[] name 参数值 zabbix_agentd_30 并不一样。Zabbix agent 从 status 文件中匹配进程名失败后，会转到 /proc/<pid>/cmdline 文件。

agent 如何看待 “cmdline” 文件，可以通过运行一个命令来说明

```
$ for i in 6715 6716 6717 6718 6719 6720; do cat /proc/$i/cmdline | awk '{gsub(/\x0/,"<NUL>"); print};'; d
sbin/zabbix_agentd_30<NUL>-c<NUL>/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf<NUL>
sbin/zabbix_agentd_30: collector [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><
sbin/zabbix_agentd_30: listener #1 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #2 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #3 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL
```

/proc/<pid>/cmdline 文件包含在 *C* 语言中用于终止字符的隐藏的、不可显示的空字符。这个例子中空字符以”<NUL>” 形式出现。

Zabbix agent 检查”cmdline” ，得到 zabbix_agentd_30 值，该值匹配我们的 name 参数值 zabbix_agentd_30。因此, 主进程会被监控项 proc.num[zabbix_agentd_30,zabbix] 计数。

当检查下一进程时, agent 从 cmdline 文件中得到 zabbix_agentd_30: collector [idle 1 sec] ，但不匹配 name 参数值 zabbix_agentd_30。所以，只有不改变命令行的主进程被计数，其他的 agent 进程改变了命令行而被忽略。

这个例子展示了 name 参数不能用在 proc.mem[] 和 proc.num[] 监控项目中来选择进程。

cmdline 参数使用恰当的正则表达式会达到一个正确的结果:

```
$ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
6
```

使用 proc.mem[] and proc.num[] 监控项监控可以修改命令行的程序时要小心。

在给 proc.mem[] 和 proc.num[] 监控项使用 name and cmdline 参数前, 你应该使用 proc.num[] 监控项和 ps 命令测试该参数。

Linux 内核线程

proc.mem[] 和 proc.num[] 监控项中的 cmdline 参数不可以使用线程

让我们以内核线程为例:

```
$ ps -ef| grep kthreadd
root         2     0  0 09:33 ?        00:00:00 [kthreadd]
```

可以用进程 “名称” 参数选择:

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd,root]'
1
```

但使用进程 cmdline 参数就不起作用:

```
$ zabbix_get -s localhost -k 'proc.num[,root,,kthreadd]'
0
```

原因是 Zabbix agent 采用 “cmdline” 参数中指定的正则表达式，并将其应用于进程的内容 /proc/<pid>/cmdline. 对于内核线程的 /proc/<pid>/cmdline 文件是空的，所以, cmdline 参数不会匹配到。

proc.mem[] 和 proc.num[] 监控项中的线程计数

Linux 内核线程通过 proc.num[] 监控项计数，但是 proc.mem[] 监控项并不报告内存。例如:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]
```

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
1

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
ZBX_NOTSUPPORTED: Cannot get amount of "VmSize" memory.
```

但是如果用户线程和内核线程名字相同会发生什么呢? 可能会是这样:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]
zabbix    9611  6133  0 17:58 pts/1    00:00:00 ./kthreadd

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
2

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
4157440
```

proc.num[] 计算内核线程和用户进程。proc.mem[] 只计算用户进程内存，如果为 0 计算内核线程内存。这和上面报告 ZBX_NOTSUPPORTED 的例子不同。

如果程序名恰好匹配其中一个线程，请小心使用 proc.mem[] 和 proc.num[] 监控项。

在给 proc.mem[] 和 proc.num[] 监控项配置参数时, 你应该使用 proc.num[] 监控项和 ps 命令测试该参数。

9 Notes on selecting processes in proc.mem and proc.num items

Processes modifying their commandline

Some programs use modifying their commandline as a method for displaying their current activity. A user can see the activity by running ps and top commands. Examples of such programs include *PostgreSQL*, *Sendmail*, *Zabbix*.

Let's see an example from Linux. Let's assume we want to monitor a number of Zabbix agent processes.

ps command shows processes of interest as

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6318     1  0 12:01 ?        00:00:00 sbin/zabbix_agentd -c /home/zabbix/ZBXNEXT-1078/zabbix_age
zabbix    6319  6318  0 12:01 ?        00:00:01 sbin/zabbix_agentd: collector [idle 1 sec]
zabbix    6320  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #1 [waiting for connection]
zabbix    6321  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #2 [waiting for connection]
zabbix    6322  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #3 [waiting for connection]
zabbix    6323  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: active checks #1 [idle 1 sec]
...
```

Selecting processes by name and user does the job:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
6
```

Now let's rename zabbix_agentd executable to zabbix_agentd_30 and restart it.

ps now shows

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6715     1  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30 -c /home/zabbix/ZBXNEXT-1078/zabbix_
zabbix    6716  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: collector [idle 1 sec]
zabbix    6717  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #1 [waiting for connection
zabbix    6718  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #2 [waiting for connection
zabbix    6719  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #3 [waiting for connection
zabbix    6720  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]
...
```

Now selecting processes by name and user produces an incorrect result:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
1
```

Why a simple renaming of executable to a longer name lead to quite different result ?

Zabbix agent starts with checking the process name. /proc/<pid>/status file is opened and the line Name is checked. In our case the Name lines are:

```
$ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
/proc/6715/status:Name:   zabbix_agentd_3
/proc/6716/status:Name:   zabbix_agentd_3
/proc/6717/status:Name:   zabbix_agentd_3
/proc/6718/status:Name:   zabbix_agentd_3
/proc/6719/status:Name:   zabbix_agentd_3
/proc/6720/status:Name:   zabbix_agentd_3
```

The process name in status file is truncated to 15 characters.

A similar result can be seen with ps command:

```
$ ps -u zabbix
  PID TTY          TIME CMD
...
 6715 ?        00:00:00 zabbix_agentd_3
 6716 ?        00:00:01 zabbix_agentd_3
 6717 ?        00:00:00 zabbix_agentd_3
 6718 ?        00:00:00 zabbix_agentd_3
 6719 ?        00:00:00 zabbix_agentd_3
 6720 ?        00:00:00 zabbix_agentd_3
 ...
```

Obviously, that is not equal to our proc.num[] name parameter value zabbix_agentd_30. Having failed to match the process name from status file the Zabbix agent turns to /proc/<pid>/cmdline file.

How the agent sees the "cmdline" file can be illustrated with running a command

```
$ for i in 6715 6716 6717 6718 6719 6720; do cat /proc/$i/cmdline | awk '{gsub(/\x0/,"<NUL>"); print};'; d
sbin/zabbix_agentd_30<NUL>-c<NUL>/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf<NUL>
sbin/zabbix_agentd_30: collector [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><
sbin/zabbix_agentd_30: listener #1 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #2 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #3 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL
```

/proc/<pid>/cmdline files in our case contain invisible, non-printable null bytes, used to terminate strings in *C* language. The null bytes are shown as "<NUL>" in this example.

Zabbix agent checks "cmdline" for the main process and takes a zabbix_agentd_30, which matches our name parameter value zabbix_agentd_30. So, the main process is counted by item proc.num[zabbix_agentd_30,zabbix].

When checking the next process, the agent takes zabbix_agentd_30: collector [idle 1 sec] from the cmdline file and it does not meet our name parameter zabbix_agentd_30. So, only the main process which does not modify its commandline, gets counted. Other agent processes modify their command line and are ignored.

This example shows that the name parameter cannot be used in proc.mem[] and proc.num[] for selecting processes in this case.

Using cmdline parameter with a proper regular expression produces a correct result:

```
$ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
6
```

Be careful when using proc.mem[] and proc.num[] items for monitoring programs which modify their commandlines.

Before putting name and cmdline parameters into proc.mem[] and proc.num[] items, you may want to test the parameters using proc.num[] item and ps command.

Linux kernel threads

Threads cannot be selected with cmdline parameter in proc.mem[] and proc.num[] items

Let's take as an example one of kernel threads:

```
$ ps -ef| grep kthreadd
root         2     0  0 09:33 ?        00:00:00 [kthreadd]
```

It can be selected with process name parameter:

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd,root]'
1
```

But selection by process cmdline parameter does not work:

```
$ zabbix_get -s localhost -k 'proc.num[,root,,kthreadd]'
0
```

The reason is that Zabbix agent takes the regular expression specified in cmdline parameter and applies it to contents of process /proc/<pid>/cmdline. For kernel threads their /proc/<pid>/cmdline files are empty. So, cmdline parameter never matches.

Counting of threads in proc.mem[] and proc.num[] items

Linux kernel threads are counted by proc.num[] item but do not report memory in proc.mem[] item. For example:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
1

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
ZBX_NOTSUPPORTED: Cannot get amount of "VmSize" memory.
```

But what happens if there is a user process with the same name as a kernel thread ? Then it could look like this:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]
zabbix    9611  6133  0 17:58 pts/1    00:00:00 ./kthreadd

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
2

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
4157440
```

proc.num[] counted both the kernel thread and the user process. proc.mem[] reports memory for the user process only and counts the kernel thread memory as if it was 0. This is different from the case above when ZBX_NOTSUPPORTED was reported.

Be careful when using proc.mem[] and proc.num[] items if the program name happens to match one of the thread.

Before putting parameters into proc.mem[] and proc.num[] items, you may want to test the parameters using proc.num[] item and ps command.

10 net.tcp.service 和 net.udp.service 检查的实现细节

net.tcp.service 和 net.udp.service 检查实现的细节在该页详细介绍，不同的服务指定不同的服务参数。

监控项 net.tcp.service 参数

ftp

创建一个 TCP 连接，并期望响应的前 4 个字符是 “220”，然后发送 “QUIT\r\n”。如果未指定，则使用缺省端口 21。

http

创建一个 TCP 连接，而不需要等待和发送任何东西。如果未指定，则使用缺省端口 80。

https

使用 (并且只使用)libcurl，不验证证书的真实性，不验证 SSL 证书中的主机名，只获取响应头 (HEAD 请求)。如果未指定端口，则使用默认端口 443。

imap

创建一个 TCP 连接，并期望响应的前 4 个字符是 “* OK”，然后发送”a1 LOGOUT\r\n”。如果未指定，则使用缺省端口 143。

ldap

打开到 LDAP 服务器的连接，并使用过滤器集执行 LDAP 搜索操作 (objectClass=*)。期望成功地检索第一个条目的第一个属性。如果未指定，则使用缺省端口 389。

nntp

创建一个 TCP 连接，并期望响应的前 3 个字符是 “200” 或 “201”，然后发送 “QUIT\r\n”。如果未指定，则使用缺省端口 119。

pop

创建一个 TCP 连接，并期望响应的前 3 个字符是 “+OK”，然后发送 “QUIT\r\n”。如果未指定，则使用缺省端口 110。

smtp

创建一个 TCP 连接，并期望响应的前 3 个字符是 “220”，然后是空格、行的结束或虚线。包含一个虚线的行属于多行响应，响应将被重新读取，直到收到一条没有虚线的行。然后发送 “QUIT\r\n”。如果未指定，则使用缺省端口 25。

ssh

创建一个 TCP 连接，如果建立了连接，双方交换一个标识字符串 (SSH-major.minor-XXXX)，其中 major 和 minor 是协议版本，XXXX 是一个字符串。Zabbix 检查是否找到了匹配该指定的字符串，不匹配则返回返回字符串 “SSH-major.minor-zabbix_agent\r\n” 或者 “0\n”。如果未指定，则使用缺省端口 22。

tcp

创建一个 TCP 连接，而不需要等待和发送任何东西。与其他检查需要指定端口参数不同。

telnet

创建一个 TCP 连接，并期望一个登录提示 (’:’ 在最后)。如果未指定，则使用缺省端口 23。

Item net.udp.service parameters

ntp

在 UDP 上发送一个 SNTP 包，并根据 RFC 4330, section 5需要验证响应。如果未指定，则使用默认端口 123。

10 Implementation details of net.tcp.service and net.udp.service checks

Implementation of net.tcp.service and net.udp.service checks is detailed on this page for various services specified in the service parameter.

Item net.tcp.service parameters

ftp

Creates a TCP connection and expects the first 4 characters of the response to be ”220 ”, then sends ”QUIT\r\n”. Default port 21 is used if not specified.

http

Creates a TCP connection without expecting and sending anything. Default port 80 is used if not specified.

https

Uses (and only works with) libcurl, does not verify the authenticity of the certificate, does not verify the host name in the SSL certificate, only fetches the response header (HEAD request). Default port 443 is used if not specified.

imap

Creates a TCP connection and expects the first 4 characters of the response to be ”* OK”, then sends ”a1 LOGOUT\r\n”. Default port 143 is used if not specified.

ldap

Opens a connection to an LDAP server and performs an LDAP search operation with filter set to (objectClass=*). Expects successful retrieval of the first attribute of the first entry. Default port 389 is used if not specified.

nntp

Creates a TCP connection and expects the first 3 characters of the response to be ”200” or ”201”, then sends ”QUIT\r\n”. Default port 119 is used if not specified.

pop

Creates a TCP connection and expects the first 3 characters of the response to be ”+OK”, then sends ”QUIT\r\n”. Default port 110 is used if not specified.

smtp

Creates a TCP connection and expects the first 3 characters of the response to be ”220”, followed by a space, the line ending or a dash. The lines containing a dash belong to a multi-line response and the response will be re-read until a line without the dash is received. Then sends ”QUIT\r\n”. Default port 25 is used if not specified.

ssh

Creates a TCP connection. If the connection has been established, both sides exchange an identification string (SSH-major.minor-XXXX), where major and minor are protocol versions and XXXX is a string. Zabbix checks if the string matching the specification is found and then sends back the string "SSH-major.minor-zabbix_agent\r\n" or "0\n" on mismatch. Default port 22 is used if not specified.

tcp

Creates a TCP connection without expecting and sending anything. Unlike the other checks requires the port parameter to be specified.

telnet

Creates a TCP connection and expects a login prompt (':' at the end). Default port 23 is used if not specified.

Item net.udp.service parameters

ntp

Sends an SNTP packet over UDP and validates the response according to RFC 4330, section 5. Default port 123 is used if not specified.

12 不可达/不可用主机设置

概述

当 agent 检查 (Zabbix, SNMP, IPMI, JMX) 失败并且主机变得不可达时，一些配置参数 定义了 Zabbix server 作何反应。

不可达主机

Zabbix, SNMP, IPMI 或 JMX agents 检查（网络错误，超时）失败后即视主机不可达. 注意，Zabbix agent 主动检查不影响主机可用性。

From that moment **UnreachableDelay** 定义了主机再次检查的频率 is rechecked using one of the items (包括 LLD 规则) in this unreachability situation and such rechecks will be performed already by unreachable pollers. 默认情况下，两次检查时间间隔为 15 秒。

在 Zabbix server 日志中，不可达是通过类似下面的消息表示的:

```
Zabbix agent item "system.cpu.load[percpu,avg1]" on host "New host" failed: first network error, wait for
Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait f
```

注意，失败的监控项和监控项类型（Zabbix agent）列出来了。

> **Note:**
> 在主机不可达期间，*Timeout* 参数也会影响主机再次被检查的时间。如果 Timeout 是 20 秒，但是 UnreachableDelay 是 30 秒，下一次检查在 50 秒后。

UnreachablePeriod 参数定义了不可达的总时长。UnreachablePeriod 应该比 UnreachableDelay 大几倍, 这样在主机变为不可用之前，主机会被检查不止一次。

如果不可达主机再次出现, 监控自动恢复正常:

恢复 Zabbix agent 对主机"New host" 的检查: 连接恢复

不可用主机

主机不可达期结束后主机没有再次出现, 视主机为不可用。

在 server 日志中，不可用是通过类似下面的消息来表示的:

```
temporarily disabling Zabbix agent checks on host "New host": host unavailable
```

在前端 主机可用性图标由绿色（或灰色）变为红色（注意，在鼠标经过时会提示错误描述）:

UnavailableDelay 参数定义了在主机不可用期间，主机被检查的频率。

默认为 60 秒 (所以此时从上面的日志信息来看，”temporarily disabling” 意味着禁用检查一分钟)。

当主机连接恢复时，监控也会自动恢复正常:

```
启用Zabbix agent 对 "New host"主机的检查: 主机变为可达
```

12 Unreachable/unavailable host settings

Overview

Several configuration parameters define how Zabbix server should behave when an agent check (Zabbix, SNMP, IPMI, JMX) fails and a host becomes unreachable.

Unreachable host

A host is treated as unreachable after a failed check (network error, timeout) by Zabbix, SNMP, IPMI or JMX agents. Note that Zabbix agent active checks do not influence host availability in any way.

From that moment **UnreachableDelay** defines how often a host is rechecked using one of the items (including LLD rules) in this unreachability situation and such rechecks will be performed already by unreachable pollers (or IPMI pollers for IPMI checks). By default it is 15 seconds before the next check.

In the Zabbix server log unreachability is indicated by messages like these:

```
Zabbix agent item "system.cpu.load[percpu,avg1]" on host "New host" failed: first network error, wait for
Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait f
```

Note that the exact item that failed is indicated and the item type (Zabbix agent).

> **Note:**
> The *Timeout* parameter will also affect how early a host is rechecked during unreachability. If the Timeout is 20 seconds and UnreachableDelay 30 seconds, the next check will be in 50 seconds after the first attempt.

The **UnreachablePeriod** parameter defines how long the unreachability period is in total. By default UnreachablePeriod is 45 seconds. UnreachablePeriod should be several times bigger than UnreachableDelay, so that a host is rechecked more than once before a host becomes unavailable.

If the unreachable host reappears, the monitoring returns to normal automatically:

```
resuming Zabbix agent checks on host "New host": connection restored
```

Unavailable host

After the UnreachablePeriod ends and the host has not reappeared, the host is treated as unavailable.

In the server log it is indicated by messages like these:

```
temporarily disabling Zabbix agent checks on host "New host": host unavailable
```

and in the frontend the host availability icon for the respective interface goes from green (or gray) to red (note that on mouseover a tooltip with the error description is displayed):

The **UnavailableDelay** parameter defines how often a host is checked during host unavailability.

By default it is 60 seconds (so in this case ”temporarily disabling”, from the log message above, will mean disabling checks for one minute).

When the connection to the host is restored, the monitoring returns to normal automatically, too:

```
enabling Zabbix agent checks on host "New host": host became available
```

6 触发器赵

6 Triggers

1 触发器函数

1 Supported trigger functions

所有触发器表达式支持的函数都在下表中。All functions supported in trigger expressions are listed here.

函数
FUNCTION

描述信息 ** 参 ** ** 备注 *		
Description	**Parameters**	**Comments**
abschange		
	nt, str, text, log 例如: (前一个值; 后一个值 = 绝对值) 1;5=4 3;1=2 0;-2.5=2.5 值类型为 str 型的返回值: 0 - 两个 str 相等 1 - 两个 str 不相等	
后一个值与前一个值变动的绝对值。支持的值类型: floa	,	
	例如: (前一个值; 后一个值 = 绝对值) 1;5=4 3;1=2 0;-2.5=2.5 值类型为 str 型的返回值: 0 - 两个 str 相等 1 - 两个 str 不相等	

avg (sec|#num,<time_shift>)		
		<Examples: => avg(#5) → 最新 5 个值的平均值 => avg(1h) → 最近一小时的平均值 => avg(1h,1d) → 一天前的一小时内的平均值 从 Zabbix 1.8.2 开始支持 `time_shift` 参数。主要用于将当前平均值与偏移若干秒后的平均值进行比较。
指定评估期内的一个 item 的平均值。**sec** or **	#num** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示 支持的值类型: float, int **time_shift** (可选) - 评估时间点相对于当前时间的偏移量	
		Examples: => avg(#5) → 最新 5 个值的平均值 => avg(1h) → 最近一小时的平均值 => avg(1h,1d) → 一天前的一小时内的平均值 从 Zabbix 1.8.2 开始支持 `time_shift` 参数。主要用于将当前平均值与偏移若干秒后的平均值进行比较。

band (sec|#num,mask,<time_shift>)			
	略) or **#num** - 第 N 个最近的价值。 支持的值类型: int **mask** (不可省略) - 64-bit 无符号整数 (0 - 18446744073709551615) **time_shift** (可选) - 参见 avg() 函数请注意 #|<um 在这里的工作方式与其他函数不同 (具体用法参见 last() 函数)。 \\尽管以二进制方式进行比较，但是所有的参数和返回值都是十进制。 示例: => band(,12)=8 or band(,12)=4 → 第三位和第四位被设置，但不是同时设置。 => band(,20)=16 → 第三位没有被设置但是第五位被设置了。 从 Zabbix 2.2.0 开始支持该函数。		
将 item 值与 mask 进行按位与操作。 **sec** (可	**time_shift** (可选) - 参见 avg() 函数请注意 #|um 在这里的工作方式与其他函数不同 (具体用法参见 last() 函数)。 \\尽管以二进制方式进行比较，但是所有的参数和返回值都是十进制。 示例: => band(,12)=8 or band(,12)=4 → 第三位和第四位被设置，但不是同时设置。 => band(,20)=16 → 第三位没有被设置但是第五位被设置了。 从 Zabbix 2.2.0 开始支持该函数。		

change	最近获取值与之前获取值的差。支持值类型: flo	t,	int, str, text, log 例如: (前一个值; 后一个值 = 差) 1;5=+4 3;1=-2 0;-2.5=-2.5 可与abschange函数对照。 值类型为 str 型的返回值: 0 - 两个 str 相等 1 - 两个 str 不相等

	The amount of difference between last and previous values.	Supported value types: float, int, str, text, log For example: (previous value;last value=change) 1;5=+4 3;1=-2 0;-2.5=-2.5 See also: abschange for comparison For strings returns: 0 - values are equal 1 - values differ

count (sec|#num,<pattern>,<operator>,<time_shift>)

|<| 指定评估期内值出现的次数。**sec** or |*#num* **-** 评估期以多少秒或最新值个数（个数跟在 **#** 号后）表示。 支持值类型***: float, integer, string, text,****pattern* ***(***可选***)*** **-** 指定模式 浮点类型的 ***
****operator* ***(***可选***)*** **
** 支持的操作符**:
*eq** -** 等于 *ne* - 不等于
gt - 大于 示 *ge* - 大于等于 =&g*lt* - 小于 *=le* - 小于等于 =&g*like* - 只要包含 (区分大小写) 就被匹配 => count(*band* - 按位与 =&*regexp* - 按 pattern 参数进行正则表达式匹配（大小写敏感） => count(10m,12*iregexp* - 按 pattern 参数进行正则表达式匹配（不区分大小） => count(10m,6/
 注意:
eq (默认), *ne*, *gt*, *ge*, *lt*, *le*, *band*, *regexp*, *iregexp* 仅仅支持整型数据。 #num 参数从 *eq* (默认), *ne*, *gt*, *ge*, *lt*, *le*, *regexp*, *iregexp* 仅仅支持浮点型数据。 time_shift*like* (默认), *eq*, *ne*, *regexp*, *iregexp* 支持 string, text 和 log 类型数据。\ *band* 操作从
time_shift (可选) - 参考 avg() 函数 |log
 据精度为 0.000001。

and* 作为第三个参数时，第二个参数 pattern 可以用'/'分隔的两个数字表示: **number_to_compare_with/mask**。 count() 函数计算值和掩码的按位与，再和 number_to_compare_with 参数进行比较，如果与 *number_to_compare_with* 参数结果相等，则该值被计数。
 如果参数 *number_to_compare_with* 和参数 *mask* 相等时，只需要指定 *mask* 参数，不需要指定 number_to_compare_with 参数以及使用'/'。
egexp* 或 *iregexp* 作为第三个参数时，第二个参数 pattern 可以是一个普通的或以 '@'开头的 [全局](/manual/regular_expressions#global_regular_expressions) 正则表达式。使用全局正则表达式时，是否大小写敏感取决于全局正则表达式的配置。为了进行正则表达式匹配，浮点值将始终用'.'后的 4 位数字表示。另请注意，对于十进制（存储在数据库中）和二进制（由 Zabbix server 使用）表示的数字差异，可能会对 4 位数字有影响。
:
; count(10m) → 过去 10 分钟值的个数。
gt; count(10m,"error",eq) → 过去十分钟值等于'error'的个数。
; count(10m,12) → 过去 10 分钟值等于 12 的个数。
0m,12,gt) → 过去 10 分钟值大于 12 的个数。
t; count(\#10,12,gt) → 最新 10 个值大于 12 的个数。
gt,1d) → 昨天这个时间点前十分钟值大于 12 的个数。
,band) → 过去 10 分钟值最低三个有效位是 '110'（二进制）的个数。
=> count(10m,,,1d) → 昨天这个时间点前十分钟值的个数。
abbix 1.6.1 开始获得支持。
 参数和字符类型操作从 Zabbix 1.8.2 开始获得支持。
abbix 2.2.0 开始获得支持。
regexp, *iregexp* 从 Zabbix 3.2.0 开始获得支持。|

|-|----------|

|<|<|
operator (可选)\
支持的操作符:\
- 等于\ *
- 不等于\ <
- 大于\
ge - 大于等于\
- 小于\ =
- 小于等于\ =
- 只要包含 (区分大小写) 就被匹配\ =>
count(*band* - 按位与\
- 按 pattern 参数进行正则表达式匹配（大小写敏感）\ =>
count(10m,12*iregexp* - 按 pattern 参数进行正则表达式匹配(不区分大小)\
count(10m,6/
 注意:\
(默认), *ne*, *gt*, *ge*, *lt*, *le*, *band*, *regexp*, *iregexp* 仅仅支持整型数据。\#num 参数从 *eq* (默认), *ne*, *gt*, *ge*, *lt*, *le*, *regexp*, *iregexp* 仅仅支持浮点型数据。\time_shift*like* (默认), *eq*, *ne*, *regexp*, *iregexp* 支持 string, text 和 log 类型数据。 *band* 操作从
time_shift (可选) - 参考 avg() 函数 |
and 作为第三个参数时, 第二个参数 *pattern* 可以用'/' 分隔的两个数字表示*:* ***number_to_compare_with/mask***。*count()* 函数计算值和掩码的按位与，再和 *number_to_compare_with* 参数进行比较，如果与 number_to_compare_with 参数结果相等, 则该值被计数。*
* 如果参数 number_to_compare_with 和参数 mask 相等时, 只需要指定 mask 参数，不需要指定 *number_to_compare_with* 参数以及使用'/'。

<*br*>*egexp* 或 *iregexp* 作为第三个参数时，第二个参数 `pattern` 可以是一个普通的或以'@' 开头的全局正则表达式。使用全局正则表达式时，是否大小写敏感取决于全局正则表达式的配置。为了进行正则表达式匹配，浮点值将始终用'.' 后的 4 位数字表示。另请注意，对于十进制（存储在数据库中）和二进制（由 Zabbix server 使用）表示的数字差异，可能会对 4 位数字有影响。
:
; count(10m) → 过去 10 分钟值的个数。
gt; count(10m,"error",eq) → 过去十分钟值等于'error' 的个数。
; count(10m,12) → 过去 10 分钟值等于 12 的个数。
0m,12,gt) → 过去 10 分钟值大于 12 的个数。
t; count(#10,12,gt) → 最新 10 个值大于 12 的个数。
gt,1d) → 昨天这个时间点前十分钟值大于 12 的个数。
,band) → 过去 10 分钟值最低三个有效位是'110' (二进制) 的个数。
=> count(10m„,1d) → 昨天这个时间点前十分钟值的个数。
abbix 1.6.1 开始获得支持。
 参数和字符类型操作从 Zabbix 1.8.2 开始获得支持。
abbix 2.2.0 开始获得支持。
regexp, *iregexp* 从 Zabbix 3.2.0 开始获得支持。|

date			
	当前时间，格式为：YYYYMMDD。支持值类型: Current date in YYYYMMDD format.	*a	y* 例如返回值: 20150731 Supported value types: *any* Example of returned value: 20150731
dayofmonth			
	当前是本月的第几天，取值范围从 1 到 31。 Day of month in range of 1 to 31.	支持值类型: *any*	< 从 Zabbix 1.8.5 开始支持该函数。 Supported value types: *any* This function is supported since Zabbix 1.8.5.
dayofweek			
	当前是本周的第几天，取值范围从 1 到 7 (周一 - 1, 周日 - 7). Day of week in range of 1 to 7 (Mon - 1, Sun - 7).	支持值的类型: *any*	Supported value types: *any*
delta (sec|#num,<time_shift>)			
	指定评估期内最大值和最小值的差 ('max()' 减去'min()')。 **sec** or **#nu	** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示。 支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数	< 从 Zabbix 1.8.2 开始支持 `time_shift` 参数。
	Difference between the maximum and minimum values within the defined evaluation period ('max()' minus 'min()').	**sec** or **#num** - evaluation period in seconds or in latest collected values specified (preceded by a hash mark) **time_shift** (optional) - see avg()	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
diff			
	比较最近获取值与之前获取值是否相同。支持值类型: float,	nt	str, text, log 返回值: 1 - 两值不等 0 - 两值相等

	Checking if last and previous values differ.	Supported value types: float, int, str, text, log Returns: 1 - last and previous values differ 0 - otherwise

forecast
(sec|#num,<time_shift>,time,<fit>,<mode>)

预测 item 未来的最大值，最小值，增量值或平均值。**sec** or **#num**	- 评估期以多少秒或最新值个数（个数跟在 # 号后）表示。支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数 **time** - 需要预测的时间点 如果返回值大于 **fit** (可选) - 用于匹配历史数据的函数 支持的匹配函数: *linear* - 线性函数 Exa*polynomialN* - N 次多项式 (1 <= N <= 6) =&g*exponential* - 指数函数 =&g*logarithmic* - 对数函数 =&g*power* - 幂函数 =& 注意: =*linear* 函数为默认, *polynomial1* 等同于 *linear* **mode** (可选) - 按需输出 参数 t 支持的 modes: *value* - 值 (默认) *max* - 最大值 *min* - 最小值 *delta* - * 最大 *-* 最小 * *avg* - 平均值 注意: *value* 预测 item 值在 now+time 时间点 *max*, *min*, *delta* 和 *avg* 函数在 now 和 now+time 时间段计算 item 值 |<999999999999.9999 或者小于 -999999999999.9999, 返回值相应被设置为 999999999999.9999 或 -999999999999.9999 。 只有在表达式被错误使用时才不可用（错误的项目类型，无效的参数），出现错误时返回-1。 ples: ; forecast(\#10,1h) → 根据最新的十个值预测一小时后的值 ; forecast(1h,30m) → 根据过去一小时的值预测三十分钟后的值 ; forecast(1h,1d,12h) → 根据昨天这个时间点前一个小时的值预测十二个小时后的值 t; fore-cast(1h,10m,exponential) → 根据过去一小时的值并按照指数函数方式预测十分钟后的值 => fore-cast(1h,2h,polynomial3,max) → 根据过去一小时的值并按照三次多项式方式预测两小时后的最大值 gt; forecast(\#2,-20m) → 根据最新的两个值预测二十分钟前的值（比使用 last() 或

	Future value, max, min, delta or avg of the item.	**sec** or **#num** - evaluation period in seconds or in latest collected values specified (preceded by a hash mark) **time_shift** (optional) - see avg() **time** - forecasting horizon in seconds **fit** (optional) - function used to fit historical data Supported `fits`: *linear* - linear function *polynomialN* - polynomial of degree N (1 <= N <= 6) *exponential* - exponential function *logarithmic* - logarithmic function *power* - power function Note that: *linear* is default, *polynomial1* is equivalent to *linear* **mode** (optional) - demanded output Supported `modes`: *value* - value (default) *max* - maximum *min* - minimum *delta* - *max-min* *avg* - average Note that: *value* estimates item value at the moment `now` + `time` *max*, *min*, *delta* and *avg* investigate item value estimate on the interval between `now` and `now` + `time`	Supported value types: float, int If value to return is larger than 999999999999.9999 or less than -999999999999.9999, return value is cropped to 999999999999.9999 or -999999999999.9999 correspondingly. Becomes not supported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => forecast(#10,1h) → forecast of item value after one hour based on last 10 values => forecast(1h,30m) → forecast of item value after 30 minutes based on last hour data => forecast(1h,1d,12h) → forecast of item after 12 hours based on one hour one day ago => forecast(1h,10m,exponential) → forecast of item value after 10 minutes based on last hour data and exponential function => forecast(1h,2h,polynomial3,max) → forecast of maximum value item can reach in next two hours based on last hour data and cubic (third degree) polynomial => forecast(#2,-20m) → estimate the value of an item which was 20 minutes ago based on last two values (this can be more precise than using last() or prev(), especially if item is updated rarely, say, once an hour) This function is supported since Zabbix 3.0.0. Negative `time` values are supported since Zabbix 3.0.6 and 3.2.2. See also additional information on predictive trigger functions.

fuzzytime (sec)	检查 item 的时间戳和 zabbix server 时间之间相差多大。 **sec** - 秒数	支持值类型: float, int	< 返回值: 0 - item 时间戳和 zabbix server 时间戳之间相差超过指定的时间 1 - 其它。 常使用 system.localtime 来检查本地时间是否与 zabbix server 的时间相同。 也可以使用 vfs.file.time[/path/file,modify] 键值检测文件是否长时间未更新。 例如: => fuzzytime(60)=0 → 如果时间差超过 60 秒，就会检测到一个问题
	Checking how much an item timestamp value differs from the Zabbix server time.	**sec** - seconds	Supported value types: float, int Returns: 0 - if difference between item timestamp value and Zabbix server timestamp is over T seconds 1 - otherwise. Usually used with system.localtime to check that local time is in sync with local time of Zabbix server. Can be used also with vfs.file.time[/path/file,modify] key to check that file didn't get updates for long time. Example: => fuzzytime(60)=0 → detect a problem if time difference is over 60 seconds

iregexp (pattern,<sec|#num>)	该函数和 regexp() 类似，只是不区分大小写。参考 regexp() 函数	持值类型: str, log, te	t
	This function is a non case-sensitive analogue of regexp().	see regexp()	Supported value types: str, log, text

last (sec|#num,<time_shift>)			

	最近的值。**se	** (可省略) or **#num** - 最新的第 N 个值 支持值类型: float,**time_shift** (可选) - 参考 avg() 函数	int, str, text, log 注意此处的 #num 参数和在其它函数中的作用不一样。 例如: last() 通常等同于 last(#1) last(#3) - 第三个最新值 (不是三个最新值) 如果在 history 中同一秒有多个值存在，Zabbix 不能保证值的精确顺序。 从 Zabbix 1.6.2 开始支持 #num 参数。 从 Zabbix 1.8.2 开始支持 time_shift 参数。
	The most recent value.	**sec** (ignored) or **#num** - the Nth most recent value **time_shift** (optional) - see avg()	Supported value types: float, int, str, text, log Take note that #num works differently here than with many other functions. For example: last() is always equal to last(#1) last(#3) - third most recent value (*not* three latest values) Zabbix does not guarantee exact order of values if more than two values exist within one second in history. The #num parameter is supported since Zabbix 1.6.2. The time_shift parameter is supported since Zabbix 1.8.2.
logeventid (pattern)			
	检查最近日志记录的 EventID 是否匹配正则表达式。 **pattern** - 正则表达	需要匹配的模式，支持POSIX extended 类型。支持值类型: log	< 返回值: 0 - 不匹配 1 - 匹配 从 Zabbix 1.8.5 开始支持该函数。
	Checking if event ID of the last log entry matches a regular expression.	**pattern** - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style.	Supported value types: log Returns: 0 - does not match 1 - matches This function is supported since Zabbix 1.8.5.
logseverity			

	最近日志记录的日志等级。支持值类型: I	g	< 返回值: 0 - 默认等级 N - 对应的等级 (整数，常用的 Windows 日志等级: 1 - 信息, 2 - 警告, 4 - 错误, 7 - 审计失败, 8 - 审计成功, 9 - 严重错误, 10 - Verbose). Zabbix 从 Windows 日志事件中的 **Information** 字段获取日志等级。
	Log severity of the last log entry.		Supported value types: log Returns: 0 - default severity N - severity (integer, useful for Windows event logs: 1 - Information, 2 - Warning, 4 - Error, 7 - Failure Audit, 8 - Success Audit, 9 - Critical, 10 - Verbose). Zabbix takes log severity from **Information** field of Windows event log.

logsource (pattern)			
	检查最近的日志记录是否匹配参数指定的日志来源。 **pattern** - string 类型	支持值类型: log	< 返回值: 0 - 不匹配 1 - 匹配 通常用于 Windows 日志事件。例如, logsource(”VMware Server”).
	Checking if log source of the last log entry matches parameter.	**pattern** - required string	Supported value types: log Returns: 0 - does not match 1 - matches Normally used for Windows event logs. For example, logsource(”VMware Server”).

max (sec|#num,<time_shift>)			
	指定评估期内一个 item 的最大值。**sec** or *	#num** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示 支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数	< 从 Zabbix 1.8.2 开始支持 `time_shift` 参数。
	Highest value of an item within the defined evaluation period.	**sec** or **#num** - evaluation period in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - see avg()	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.

min (sec|#num,<time_shift>)	指定评估期内一个 item 的最小值。**sec** or *	#num** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示 支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数	< 从 Zabbix 1.8.2 开始支持 `time_shift` 参数。
	Lowest value of an item within the defined evaluation period.	**sec** or **#num** - evaluation period in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - see avg()	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
nodata (sec)	检查评估期内是否接收到数据。**sec** - 评估期	单位为秒。 支持值类型: *any* 评估期不应该少于 30 秒，因为 timer 处理器每 30 秒调用一次该函数。 nodata(0) 不被允许. 1 -	< 返回值: 指定评估期没有接收到数据 0 - 其它 注意，如果在第一个参数指定的时间内出现以下问题，该函数会报错: - Zabbix server 被重启 - 刚出维护期 - item 被添加或重新激活 - 错误显示在触发器配置界面的 *Info* 列。
	Checking for no data received.	**sec** - evaluation period in seconds. The period should not be less than 30 seconds because the timer process calculates this function only every 30 seconds. nodata(0) is disallowed.	Supported value types: *any* Returns: 1 - if no data received during the defined period of time 0 - otherwise Note that this function will display an error if, within the period of the 1st parameter: - there's no data and Zabbix server was restarted - there's no data and maintenance was completed - there's no data and the item was added or re-enabled Errors are displayed in the *Info* column in trigger configuration.
now	距离 Epoch (00:00:00 UTC, January 1, 1970) 的秒数。支持 Number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).	的类	: *any* Supported value types: *any*

percentile (sec|#num,<time_shift>,percentage)	P-th 一段时间的百分值，P (percentage) 做为第三个参数。**sec** or **#n	m** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示 支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数 **percentage** - 0 到 100 (包括) 之间的浮点数，小数点后最多保留四位从 Zabbix 3.0.0 开始支持该函	<。
	P-th percentile of a period, where P (percentage) is specified by the third parameter.	**sec** or **#num** - evaluation period in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - see avg() **percentage** - a floating-point number between 0 and 100 (inclusive) with up to 4 digits after the decimal point	Supported value types: float, int This function is supported since Zabbix 3.0.0.
prev	取前一个值。支持	类型	float, int, str, text, log 返回值和 last(#2) 相同。
	Previous value.		Supported value types: float, int, str, text, log Returns the same as last(#2).
regexp (pattern,<sec|#num>)	检查最近的值是否匹配正则表达式。**pattern** - 正	表达式, 支持POSIX extended 样式。 支持值类型: str, text, log **sec** or **#num** (可选) - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示。这种情况下，可能有多个值被处理。	< 返回值: 1 - 匹配 0 - 不匹配。如果有多个值被处理，其中有一个值匹配也会返回 1。 该函数区分大小写。
	Checking if the latest (most recent) value matches regular expression.	**pattern** - regular expression, Perl Compatible Regular Expression (PCRE) style. **sec** or **#num** (optional) - evaluation period in seconds or in latest collected values (preceded by a hash mark). In this case, more than one value may be processed.	Supported value types: str, text, log Returns: 1 - found 0 - otherwise If more than one value is processed, '1' is returned if there is at least one matching value. This function is case-sensitive.

str (pattern,<sec|#num>)	从最新值中查找一个字符串。**pattern**	string 型 支持值类型: str, t**sec** or **#num** (可选) - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示。这种情况下，可能有多个值被处理。	xt, log 返回值: 1 - 找到 0 - 没找到。如果有多个值被处理，其中有一个值找到也会返回 1。 该函数区分大小写。
	Finding a string in the latest (most recent) value.	**pattern** - required string **sec** or **#num** (optional) - evaluation period in seconds or in latest collected values (preceded by a hash mark). In this case, more than one value may be processed.	Supported value types: str, text, log Returns: 1 - found 0 - otherwise If more than one value is processed, '1' is returned if there is at least one matching value. This function is case-sensitive.
strlen (sec|#num,<time_shift>)	最新值的字符长度（而不是字节数）。**sec** (可省略) or	**#num** - 最新的第 N 个值 支持值类型: str, text, log **time_shift** (可选) - 参考 avg() 函数	< 注意此处的 #num 参数和它在其它函数中的作用不一样。 示例: => strlen()(等同于 strlen(#1)) → 最新值的长度 => strlen(#3) → 最新的第三个值的长度 => strlen(,1d) → 一天前最新值的长度。 从 Zabbix 1.8.4 开始支持该函数。
	Length of the latest (most recent) value in characters (not bytes).	**sec** (ignored) or **#num** - the Nth most recent value **time_shift** (optional) - see avg()	Supported value types: str, text, log Take note that #num works differently here than with many other functions. Examples: => strlen()(is equal to strlen(#1)) → length of the latest value => strlen(#3) → length of the third most recent value => strlen(,1d) → length of the most recent value one day ago. This function is supported since Zabbix 1.8.4.

sum (sec|#num,<time_shift>)			
	指定评估期内 item 值的和。 **sec** o	**#num** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示 支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数	< 从 Zabbix 1.8.2 开始支持 `time_shift` 参数。
	Sum of collected values within the defined evaluation period.	**sec** or **#num** - evaluation period in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - see avg()	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
time			
	当前时间，以 HHMMSS 格式表示。支持值类型: Current time in HHMMSS format.	an	* 返回值如: 123055 Supported value types: *any* Example of returned value: 123055
timeleft (sec|#num,<time_shift>,threshold,<fit>)			

	item 达到定义阀值需要多久时间，单位为秒。**sec** or **#nu	** - 评估期以多少秒或最新值个数（个数跟在 # 号后）表示 支持值类型: float, int **time_shift** (可选) - 参考 avg() 函数 **threshold** - 阀值 如 **fit** (可选) - 参考 forecast() 函数	< 返回值大于 999999999999.9999, 则被设置为 999999999999.9999。 如果达不到阀值也将返回值设置为 999999999999.9999。 只有在表达式错误时才不可用(错误的 item 类型，无效的参数)，出现错误时返回-1。 示例: => timeleft(#10,,0) → 根据最新的十个值计算值达到 0 需要的时间 => timeleft(1h,,100) → 根据过去一小时的值计算值达到 100 需要的时间 => timeleft(1h,1d,0) → 根据昨天当前时间点前一个小时的值计算值达到 0 需要的时间 => timeleft(1h,,200,polynomial2) → 根据过去一小时的值并按照二次多项式方式计算值达到 200 需要的时间 从 Zabbix 3.0.0 开始支持该函数。 从 Zabbix 3.0.6 和 3.2.2 开始支持Unit symbols 的 `threshold` 参数。 通过predictive trigger functions查看扩展信息。

	Time in seconds needed for an item to reach a specified threshold.	**sec** or **#num** - evaluation period in seconds or in latest collected values specified (preceded by a hash mark) **time_shift** (optional) - see avg() **threshold** - value to reach **fit** (optional) - see forecast()	Supported value types: float, int If value to return is larger than 999999999999.9999, return value is cropped to 999999999999.9999. Returns 999999999999.9999 if threshold cannot be reached. Becomes not supported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => timeleft(#10,,0) → time until item value reaches zero based on last 10 values => timeleft(1h,,100) → time until item value reaches 100 based on last hour data => timeleft(1h,1d,0) → time until item value reaches 0 based on one hour one day ago => timeleft(1h,,200,polynomial2) → time until item reaches 200 based on last hour data and assumption that item behaves like quadratic (second degree) polynomial This function is supported since Zabbix 3.0.0. Unit symbols in `threshold` parameter are supported since Zabbix 3.0.6 and 3.2.2. See also additional information on predictive trigger functions.

Attention:

1）所有函数返回值都为数值类型。例如，不可能返回字符型数据。

2）部分函数不能用于非数值类型数据。

3）字符型参数都应该使用双引号。否则，可能会被错误解析。

4）所有 trigger 函数中的 **sec** 和 **time_shift** 参数都必须是数值类型。时间单位后缀与 item 的数据类型完全无关。

Attention:

1) All functions return numeric values only. Comparison to strings is not supported, for example.
2) Some of the functions cannot be used for non-numeric parameters!
3) String arguments should be double quoted. Otherwise, they might get misinterpreted.
4) For all trigger functions **sec** and **time_shift** must be an integer with an optional time unit suffix and has absolutely nothing to do with item's data type.

函数与 unsupported item

Functions and unsupported items

Attention:
从 Zabbix 3.2 开始，**nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **now()** 和 **time()** 函数都支持用于 unsupported item。但其他函数都要求用于 supported item。

Attention:
Starting from Zabbix 3.2, **nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **now()** and **time()** functions are calculated for unsupported items, too. Other functions require that the referenced item is in a supported state.

7 宏赵

7 Macros

1 宏使用场景

1 Macros supported by location

概述

Overview

下表包含 Zabbix 支持宏的完整列表。The table contains a complete list of macros supported by Zabbix.

宏	持场景描述信息	
Macro	Supported in	Description
{ACTION.ID}	→ 基于 Trigger 的通知和命令 *action→ 发现通知 从 2.→ 自动注册通知 → 内部通知 → 故障更新通知	数字标识。* .0 开始支持。
{ACTION.ID}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Numeric ID of the triggered action.* Supported since 2.2.0.
{ACTION.NAME}	→ 基于 Trigger 的通知和命令 *action→ 发现通知 从 2.→ 自动注册通知 → 内部通知 → 故障更新通知	名称。* .0 开始支持。

{ACTION.NAME}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Name of the triggered action.* Supported since 2.2.0.

{ALERT.MESSAGE}	→ 报警脚本参数 * 默认值由	ction 配置。* 从 3.0.0 开始支持。
{ALERT.MESSAGE}	→ Alert script parameters	*'Default message' value from action configuration.* Supported since 3.0.0.

{ALERT.SENDTO}	→ 报警脚本参数 * 值来自于	户报警媒介配置。* 从 3.0.0 开始支持。
{ALERT.SENDTO}	→ Alert script parameters	*'Send to' value from user media configuration.* Supported since 3.0.0.

{ALERT.SUBJECT}	→ 报警脚本参数 * 默认值由	ction 配置。* 从 3.0.0 开始支持。
{ALERT.SUBJECT}	→ Alert script parameters	*'Default subject' value from action configuration.* Supported since 3.0.0.

{DATE}	→ 基于 Trigger 的通知和命令 * 当前时间使用 → 发现通知 → 自动注册通知 → 内部通知 → 故障更新通知	yyyy.mm.dd 格式。*
{DATE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Current date in yyyy.mm.dd. format.*

{DISCOVERY.DEVICE.IPADDRESS}	→ 发现通知 * 被发	设备的 IP 地址。* 不依赖于是否添加设备。
{DISCOVERY.DEVICE.IPADDRESS}	→ Discovery notifications	*IP address of the discovered device.* Available always, does not depend on host being added.

{DISCOVERY.DEVICE.DNS}	→ 发现通知 * 被发	设备的 DNS 名称。* 不依赖于是否添加设备。
{DISCOVERY.DEVICE.DNS}	→ Discovery notifications	*DNS name of the discovered device.* Available always, does not depend on host being added.

{DISCOVERY.DEVICE.STATUS}	→ 发现通知 * 被发	设备的状态。*: 可能是 UP 或 DOWN.

{DISCOVERY.DEVICE.STATUS}	→ Discovery notifications	*Status of the discovered device*: can be either UP or DOWN.
{DISCOVERY.DEVICE.UPTIME}	→ 发现通知 //距	定设备最近一次发现状态改变的时间。<*br*> 例如*: 1h 29m.*\\对于状态为 *DOWN* 的设备，这是其停机时间。| |*{DISCOVERY.DEVICE.UPTIME}* |→ *Discovery notifications* |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} {DISCOVERY.RULE.NAME}	→ 发现通知 * 发现 → Discovery notifications	备或服务是否存在的发现规则名称。* *Name of the discovery rule that discovered the presence or absence of the device or service.*
{DISCOVERY.SERVICE.NAME} {DISCOVERY.SERVICE.NAME}	→ 发现通知 * 被发 → Discovery notifications	服务的名称。*\\例如: HTTP。 *Name of the service that was discovered.* For example: HTTP.
{DISCOVERY.SERVICE.PORT} {DISCOVERY.SERVICE.PORT}	→ 发现通知 * 被发 → Discovery notifications	服务的端口。* 例如: 80。 *Port of the service that was discovered.* For example: 80.
{DISCOVERY.SERVICE.STATUS} {DISCOVERY.SERVICE.STATUS}	→ 发现通知 * 被发 → Discovery notifications	服务的状态。* 可能是 UP 或 DOWN。 //Status of the discovered service:// can be either UP or DOWN.
{DISCOVERY.SERVICE.UPTIME}	→ 发现通知 //距	定服务最近一次发现状态改变的时间。<*br*> 例如*: 1h 29m.*\\对于状态为 *DOWN* 的服务，这是其停服时间。| |*{DISCOVERY.SERVICE.UPTIME}* |→ *Discovery notifications* |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}	→ 基于 Trigger 的通知和命令 * 记录以前发送 → 内部通知 显示先 → 故障更新通知	息的日志。<*br*> 在升级步骤中发送的通知信息，且发送通知状态为：(已发送, 正在发送 * 或 发送失败).

{ESC.HISTORY}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	*Escalation history. Log of previously sent messages.* Shows previously sent notifications, on which escalation step they were sent and their status (*sent*, *in progress* or *failed*).

{EVENT.ACK.STATUS}	→ 基于 Trigger 的通知和命令 * 事件的确认状 → 故障更新通知	。(Yes/No)*.
{EVENT.ACK.STATUS}	→ Trigger-based notifications and commands → Problem update notifications	*Acknowledgement status of the event (Yes/No)*.

{EVENT.AGE}	→ 基于 Trigger 的通知和命令 * 触发动作的事 → 发现通知 对逐步 → 自动注册通知 → 内部通知 → 故障更新通知	持续时间。* 级的消息非常有用。
{EVENT.AGE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Age of the event that triggered an action.* Useful in escalated messages.

{EVENT.DATE}	→ 基于 Trigger 的通知和命令 * 触发动作的事 → 发现通知 → 自动注册通知 → 内部通知 → 故障更新通知	发生日期。*
{EVENT.DATE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Date of the event that triggered an action.*

{EVENT.ID}	→ 基于 Trigger 的通知和命令 * 触发动作的事 → 发现通知 → 自动注册通知 → 内部通知 → 故障更新通知	数字标识。*
{EVENT.ID}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Numeric ID of the event that triggered an action.*

{EVENT.NAME}	→ 基于 Trigger 的通知和命令 * 触发动作的故 → 故障更新通知从 4.0.	或恢复事件的名字。* 开始支持。
{EVENT.NAME}	→ Trigger-based notifications → Problem update notifications	*Name of the problem/recovery event that triggered an action.* Supported since 4.0.0.

{EVENT.NSEVERITY}	→ 基于 Trigger 的通知和命令 * 事件的级别。→ 故障更新通知从 4.0.	可能的值: 0 - 未知, 1 - 信息, 2 - 警告, 3 - 普通, 4 - 高, 5 - 灾难。 开始支持。
{EVENT.NSEVERITY}	→ Trigger-based notifications → Problem update notifications	*Numeric value of the event severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported since 4.0.0.

{EVENT.RECOVERY.DATE}	→ 基于 Trigger 的通知和命令 * 恢复事件的发 → 内部通知 只能用 → 故障更新通知从 2.2.	时间。* 恢复消息。 开始支持。
{EVENT.RECOVERY.DATE}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Date of the recovery event.* Can be used in recovery messages only. Supported since 2.2.0.

{EVENT.RECOVERY.ID}	→ 基于 Trigger 的通知和命令 * 恢复事件的数 → 内部通知 只能用 → 故障更新通知	标识。* 恢复消息。从 2.2.0 开始支持。
{EVENT.RECOVERY.ID}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Numeric ID of the recovery event.* Can be used in recovery messages only. Supported since 2.2.0.

{EVENT.RECOVERY.STATUS}	→ 基于 Trigger 的通知和命令 * 恢复事件的状 → 内部通知 只能用 → 故障更新通知	。* 恢复消息。从 2.2.0 开始支持。
{EVENT.RECOVERY.STATUS}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Verbal value of the recovery event.* Can be used in recovery messages only. Supported since 2.2.0.

{EVENT.RECOVERY.TAGS}	→ 基于 Trigger 的通知和命令 * 逗号分隔的恢 → 故障更新通知从 3.2.	事件 tag 列表。* 如果不存在 tag，则为空字符串。 开始支持。
{EVENT.RECOVERY.TAGS}	→ Trigger-based notifications and commands → Problem update notifications	*A comma separated list of recovery event tags.* Expanded to an empty string if no tags exist. Supported since 3.2.0.

{EVENT.RECOVERY.TIME}	→ 基于 Trigger 的通知和命令 * 恢复事件的时 → 内部通知 只能用 → 故障更新通知	。* 恢复消息。从 2.2.0 开始支持。
{EVENT.RECOVERY.TIME}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Time of the recovery event.* Can be used in recovery messages only. Supported since 2.2.0.

{EVENT.RECOVERY.VALUE}	→ 基于 Trigger 的通知和命令 * 恢复事件的数 → 内部通知 只能用 → 故障更新通知	值。* 恢复消息。从 2.2.0 开始支持。

{EVENT.RECOVERY.VALUE}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Numeric value of the recovery event.* Can be used in recovery messages only. Supported since 2.2.0.

{EVENT.SEVERITY}	→ 基于 Trigger 的通知和命令 * 事件的级别。→ 故障更新通知从 4.0.	开始支持。
{EVENT.SEVERITY}	→ Trigger-based notifications → Problem update notifications	*Name of the event severity.* Supported since 4.0.0.

{EVENT.STATUS}	→ 基于 Trigger 的通知和命令 * 触发动作的事 → 发现通知 从 2.→ 自动注册通知 → 内部通知 → 故障更新通知	状态。* .0 开始支持。
{EVENT.STATUS}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Verbal value of the event that triggered an action.* Supported since 2.2.0.

{EVENT.TAGS}	→ 基于 Trigger 的通知和命令 * 用逗号分隔的 → 故障更新通知从 3.2.	件 tag 列表。* 如果不存在 tag，则为空字符串。 开始支持。
{EVENT.TAGS}	→ Trigger-based notifications and commands → Problem update notifications	*A comma separated list of event tags.* Expanded to an empty string if no tags exist. Supported since 3.2.0.

{EVENT.TIME}	→ 基于 Trigger 的通知和命令 * 触发动作的事 → 发现通知 → 自动注册通知 → 内部通知 → 故障更新通知 notifications	时间。*
{EVENT.TIME}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Time of the event that triggered an action.*

{EVENT.UPDATE.ACTION}	→ 故障更新通知 * 可读的操	名称。* 故障更新时执行。 解析为以下值: *acknowledged*, *commented*, *changed severity from (original severity) to (updated severity)* and *closed* (依赖于一次更新操作执行多少个动作). 从 4.0.0 开始支持。

{EVENT.UPDATE.ACTION}	→ Problem update notifications	*Human-readable name of the action(s)* performed during problem update. Resolves to the following values: *acknowledged*, *commented*, *changed severity from (original severity) to (updated severity)* and *closed* (depending on how many actions are performed in one update). Supported since 4.0.0.

{EVENT.UPDATE.DATE}	→ 故障更新通知 * 故障更新	间。(确认, 等)。* 取代以前的宏: {ACK.DATE}
{EVENT.UPDATE.DATE}	→ Problem update notifications	*Date of problem update (acknowledgement, etc).* Deprecated name: {ACK.DATE}

{EVENT.UPDATE.HISTORY}	→ 基于 Trigger 的通知和命令 * 记录故障更新 → 故障更新通知取代以前的	志。(确认, 等).* : {EVENT.ACK.HISTORY}
{EVENT.UPDATE.HISTORY}	→ Trigger-based notifications and commands → Problem update notifications	*Log of problem updates (acknowledgements, etc).* Deprecated name: {EVENT.ACK.HISTORY}

{EVENT.UPDATE.MESSAGE}	→ 故障更新通知 * 故障更新	息。* 取代以前的宏: {ACK.MESSAGE}
{EVENT.UPDATE.MESSAGE}	→ Problem update notifications	*Problem update message.* Deprecated name: {ACK.MESSAGE}

{EVENT.UPDATE.TIME}	→ 故障更新通知 * 故障更新	间。(确认, 等).* 取代以前的宏: {ACK.TIME}
{EVENT.UPDATE.TIME}	→ Problem update notifications	*Time of problem update (acknowledgement, etc).* Deprecated name: {ACK.TIME}

{EVENT.VALUE}	→ 基于 Trigger 的通知和命令 * 触发动作的事 → 发现通知 从 2.→ 自动注册通知 → 内部通知 → 故障更新通知	类型 (1 为故障, 0 为恢复)。* .0 开始支持。
{EVENT.VALUE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Numeric value of the event that triggered an action (1 for problem, 0 for recovering).* Supported since 2.2.0.

{HOST.CONN<1-9>}	→ 基于 Trigger 的通知和命令 * 设备 IP 或 D→ 内部通知 从 2.→ 故障更新通知 → 全局脚本 (包括确认文本) → 地图中的 Icon 标签[1] → Item key 值[2] → 设备接口 IP/DNS → 数据库监控附加字段[5] → SSH 和 Telnet 脚本[5] → JMX item endpoint 字段 → Web 监控[6] → Low-level 发现规则过滤正则表达式[8] → 动态 URL 仪表板小部件/屏幕元素的 URL 字段[8] → Trigger 名字和描述 → Trigger URLs[10] → 事件 tag 的名称和值 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	S 名称, 依赖于设备配置。*[3]. .0 开始支持。
{HOST.CONN<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Icon labels in maps[1] → Item key parameters[2] → Host interface IP/DNS → Database monitoring additional parameters[5] → SSH and Telnet scripts[5] → JMX item endpoint field → Web monitoring[6] → Low-level discovery rule filter regular expressions[8] → URL field of dynamic URL dashboard widget/screen element[8] → Trigger names and descriptions → Trigger URLs[10] → Event tags and values → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Host IP address or DNS name, depending on host settings*[3]. Supported in trigger names since 2.0.0.

{HOST.DESCRIPTION<1-9>}	→ 基于 Trigger 的通知和命令 设备描述。 → 内部通知 从 2.→ 故障更新通知 → 地图中的 Icon 标签[1]	<.0 开始支持。
{HOST.DESCRIPTION<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Icon labels in maps[1]	*Host description.* Supported since 2.4.0.

{HOST.DNS<1-9>}	→ 基于 Trigger 的通知和命令 * 设备 DNS 名 → 内部通知 tri→ 故障更新通知 → 全局脚本 (包括确认文本) → 地图中的 Icon 标签[1] → Item key 值[2] → 设备接口 IP/DNS → 数据库监控附加字段[5] → SSH 和 Telnet 脚本[5] → JMX item endpoint 字段 → Web 监控[6] → Low-level 发现规则过滤正则表达式[8] → 动态 URL 仪表板小部件/屏幕元素的 URL 字段[8] → Trigger 名字和描述 → Trigger URLs[10] → 事件 tag 的名称和值 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*[3]. ger 名字从 2.0.0 开始支持。
{HOST.DNS<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Icon labels in maps[1] → Item key parameters[2] → Host interface IP/DNS → Database monitoring additional parameters[5] → SSH and Telnet scripts[5] → JMX item endpoint field → Web monitoring[6] → Low-level discovery rule filter regular expressions[8] → URL field of dynamic URL dashboard widget/screen element[8] → Trigger names and descriptions → Trigger URLs[10] → Event tags and values → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Host DNS name*[3]. Supported in trigger names since 2.0.0.

{HOST.HOST<1-9>}	→ 基于 Trigger 的通知和命令 设备名称。 → 自动注册通知 {HOS→ 内部通知 → 故障更新通知 → 全局脚本 (包括确认文本) → Item key 值 → 地图中的 Icon 标签 ^[1](supported_by_location#footnotes)^ → 设备接口 IP/DNS → 数据库监控附加字段 ^[5](supported_by_location#footnotes)^ → SSH 和 Telnet 脚本 ^[5](supported_by_location#footnotes)^ → JMX item endpoint 字段 → Web 监控 ^[6](supported_by_location#footnotes)^ → Low-level 发现规则过滤正则表达式 ^[8](supported_by_location#footnotes)^ → 动态 URL 仪表板小部件/屏幕元素的 URL 字段 ^[8](supported_by_location#footnotes)^ → Trigger 名字和描述 → Trigger URLs ^[10](supported_by_location#footnotes)^ → 事件 tag 的名称和值 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file。|<NAME<1-9>} 已经不被支持。	
{HOST.HOST<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Item key parameters → Icon labels in maps[1] → Host interface IP/DNS → Database monitoring additional parameters[5] → SSH and Telnet scripts[5] → JMX item endpoint field → Web monitoring[6] → Low-level discovery rule filter regular expressions[8] → URL field of dynamic URL dashboard widget/screen element[8] → Trigger names and descriptions → Trigger URLs [10] → Event tags and values → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Host name.* `{HOSTNAME<1-9>}` is deprecated.

{HOST.ID<1-9>}	→ 地图中 URLs ＊设 → 动态 URL 仪表板小部件/屏幕元素的 URL 字段[8] → Trigger URLs[10] → 事件 tag 的名称和值	ID。＊
{HOST.ID<1-9>}	→ Map URLs → URL field of dynamic URL dashboard widget/screen element[8] → Trigger URLs[10] → Event tags and values	*Host ID.*

{HOST.IP<1-9>}	→ 基于 Trigger 的通知和命令 ＊设备 IP 地址 → 自动注册通知 从 2.0.→ 内部通知 → 故障更新通知 → 全局脚本 (包括确认文本) → 地图中的 Icon 标签[1] → Item key 值[2] → 设备接口 IP/DNS → Database monitoring additional parameters[5] → SSH 和 Telnet 脚本[5] → JMX item endpoint 字段 → Web 监控[6] → Low-level 发现规则过滤正则表达式[8] → 动态 URL 仪表板小部件/屏幕元素的 URL 字段[8] → Trigger 名字和描述 → Trigger URLs[10] → 事件 tag 的名称和值 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	＊[3]. 开始支持。宏 {IPADDRESS<1-9>} 已经不被支持。
{HOST.IP<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Icon labels in maps[1] → Item key parameters[2] → Host interface IP/DNS → Database monitoring additional parameters[5] → SSH and Telnet scripts[5] → JMX item endpoint field → Web monitoring[6] → Low-level discovery rule filter regular expressions[8] → URL field of dynamic URL dashboard widget/screen element[8] → Trigger names and descriptions → Trigger URLs[10] → Event tags and values → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Host IP address*[3]. Supported since 2.0.0. {IPADDRESS<1-9>} is deprecated.

{HOST.METADATA}	→ 自动注册通知 * 设备元数	。* 仅仅用于主动 agent 的自动注册。从 2.2.0 开始支持。
{HOST.METADATA}	→ Auto registration notifications	*Host metadata.* Used only for active agent auto-registration. Supported since 2.2.0.

{HOST.NAME<1-9>}	→ 基于 Trigger 的通知和命令 * 用于显示的设 → 自动注册通知 从 2.0.→ 故障更新通知 → 内部通知 → 全局脚本 (包括确认文本) → 地图中的 Icon 标签[1] → Item key 值 → 设备接口 IP/DNS → 数据库监控附加字段[5] → SSH 和 Telnet 脚本[5] → Web 监控[6] → Low-level 发现规则过滤正则表达式[8] → 动态 URL 仪表板小部件/屏幕元素的 URL 字段[8] → Trigger 名字和描述 → Trigger URLs[10] → 事件 tag 的名称和值 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	名称 * 开始支持。
{HOST.NAME<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Problem update notifications → Internal notifications → Global scripts (including confirmation text) → Icon labels in maps[1] → Item key parameters → Host interface IP/DNS → Database monitoring additional parameters[5] → SSH and Telnet scripts[5] → Web monitoring[6] → Low-level discovery rule filter regular expressions[8] → URL field of dynamic URL dashboard widget/screen element[8] → Trigger names and descriptions → Trigger URLs[10] → Event tags and values → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Visible host name.* Supported since 2.0.0.

{HOST.PORT<1-9>}	→ 基于 Trigger 的通知和命令 * 设备 (age→ 自动注册通知 从 2.0.→ 内部通知 从 2.→ 故障更新通知 → Trigger 名字和描述 → Trigger URLs[10] → JMX item endpoint 字段 → 事件 tag 的名称和值	t) 端口 *[3]. 开始支持自动注册通知。 .2 开始支持用于 trigger 名称和描述，内部通知，事件 tag 的名称和值。
{HOST.PORT<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Internal notifications → Problem update notifications → Trigger names and descriptions → Trigger URLs[10] → JMX item endpoint field → Event tags and values	*Host (agent) port*[3]. Supported in auto-registration since 2.0.0. Supported in trigger names, trigger descriptions, internal and trigger-based notifications since 2.2.2.

{HOSTGROUP.ID}	→ 地图 URLs *	备组标识。*
{HOSTGROUP.ID}	→ Map URLs	*Host group ID.*

{INVENTORY.ALIAS<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的别名字段。*
{INVENTORY.ALIAS<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Alias field in host inventory.*

{INVENTORY.ASSET.TAG<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的资产标签字段。*
{INVENTORY.ASSET.TAG<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Asset tag field in host inventory.*

{INVENTORY.CHASSIS<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的机箱字段。*
{INVENTORY.CHASSIS<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Chassis field in host inventory.*

{INVENTORY.CONTACT<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 宏 {→ 故障更新通知 → 事件 tag 的名称和值 | 中的联系人字段。* ROFILE.CONTACT<1-9>} 已经不被支持。	

{INVENTORY.CONTACT<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Contact field in host inventory.* {PROFILE.CONTACT<1-9>} is deprecated.

{INVENTORY.CONTRACT.NUMBER<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的联系号码字段。*
{INVENTORY.CONTRACT.NUMBER<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Contract number field in host inventory.*

{INVENTORY.DEPLOYMENT.STATUS<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的部署状态字段。*
{INVENTORY.DEPLOYMENT.STATUS<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Deployment status field in host inventory.*

{INVENTORY.HARDWARE<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 宏 {→ 故障更新通知 → 事件 tag 的名称和值 | 中的硬件信息字段。* ROFILE.HARDWARE<1-9>} 已经不被支持。	
{INVENTORY.HARDWARE<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Hardware field in host inventory.* {PROFILE.HARDWARE<1-9>} is deprecated.

{INVENTORY.HARDWARE.FULL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的硬件详细描述字段。*
{INVENTORY.HARDWARE.FULL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Hardware (Full details) field in host inventory.*

{INVENTORY.HOST.NETMASK<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的子网掩码字段。*
{INVENTORY.HOST.NETMASK<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Host subnet mask field in host inventory.*

{INVENTORY.HOST.NETWORKS<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的网络字段。*
{INVENTORY.HOST.NETWORKS<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Host networks field in host inventory.*

{INVENTORY.HOST.ROUTER<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的路由字段。*
{INVENTORY.HOST.ROUTER<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Host router field in host inventory.*

{INVENTORY.HW.ARCH<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的硬件架构字段。*
{INVENTORY.HW.ARCH<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Hardware architecture field in host inventory.*

{INVENTORY.HW.DATE.DECOMM<1-9>}	→ 基于 Trigger 的通知 * 主机清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的硬件下线日期字段。*
{INVENTORY.HW.DATE.DECOMM<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Date hardware decommissioned field in host inventory.*

{INVENTORY.HW.DATE.EXPIRY<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的保修期字段。*
{INVENTORY.HW.DATE.EXPIRY<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Date hardware maintenance expires field in host inventory.*

{INVENTORY.HW.DATE.INSTALL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的硬件上线日期字段。*
{INVENTORY.HW.DATE.INSTALL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Date hardware installed field in host inventory.*

{INVENTORY.HW.DATE.PURCHASE<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中硬件购买时间字段。*
{INVENTORY.HW.DATE.PURCHASE<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Date hardware purchased field in host inventory.*

{INVENTORY.INSTALLER.NAME<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的安装名称字段。*
{INVENTORY.INSTALLER.NAME<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Installer name field in host inventory.*

{INVENTORY.LOCATION<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 宏 {→ 故障更新通知 → 事件 tag 的名称和值 | 中的位置字段。* ROFILE.LOCATION<1-9>} 已经不被支持。	
{INVENTORY.LOCATION<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Location field in host inventory.* {PROFILE.LOCATION<1-9>} is deprecated.

{INVENTORY.LOCATION.LAT<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的位置纬度字段。*
{INVENTORY.LOCATION.LAT<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Location latitude field in host inventory.*

{INVENTORY.LOCATION.LON<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的位置经度字段。*
{INVENTORY.LOCATION.LON<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Location longitude field in host inventory.*

{INVENTORY.MACADDRESS.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 宏 {→ 故障更新通知 → 事件 tag 的名称和值 | 中的 MAC 地址字段。* ROFILE.MACADDRESS<1-9>} 已经不被支持。	

{INVENTORY.MACADDRESS.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*MAC address A field in host inventory.* {PROFILE.MACADDRESS<1-9>} is deprecated.

{INVENTORY.MACADDRESS.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 MAC 地址字段。*
{INVENTORY.MACADDRESS.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*MAC address B field in host inventory.*

{INVENTORY.MODEL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的模型字段。*
{INVENTORY.MODEL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Model field in host inventory.*

{INVENTORY.NAME<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的名称字段。* 宏 {PROFILE.NAME<1-9>} 已经不被支持。
{INVENTORY.NAME<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Name field in host inventory.* {PROFILE.NAME<1-9>} is deprecated.

{INVENTORY.NOTES<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的备注字段。* 宏 {PROFILE.NOTES<1-9>} 已经不被支持。
{INVENTORY.NOTES<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Notes field in host inventory.* {PROFILE.NOTES<1-9>} is deprecated.

{INVENTORY.OOB.IP<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 OOB IP 地址字段。*
{INVENTORY.OOB.IP<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*OOB IP address field in host inventory.*

{INVENTORY.OOB.NETMASK<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 OOB 子网掩码字段。*
{INVENTORY.OOB.NETMASK<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*OOB subnet mask field in host inventory.*

{INVENTORY.OOB.ROUTER<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 OOB 路由字段。*
{INVENTORY.OOB.ROUTER<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*OOB router field in host inventory.*

{INVENTORY.OS<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 宏 {→ 故障更新通知 → 事件 tag 的名称和值 | 中的操作系统字段。* ROFILE.OS<1-9>} 已经不被支持。	
{INVENTORY.OS<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*OS field in host inventory.* {PROFILE.OS<1-9>} is deprecated.

{INVENTORY.OS.FULL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的操作系统详细描述字段。*
{INVENTORY.OS.FULL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*OS (Full details) field in host inventory.*

{INVENTORY.OS.SHORT<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的操作系统缩写字段。*
{INVENTORY.OS.SHORT<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*OS (Short) field in host inventory.*

{INVENTORY.POC.PRIMARY.CELL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的主要 POC cell 字段。*
{INVENTORY.POC.PRIMARY.CELL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC cell field in host inventory.*

{INVENTORY.POC.PRIMARY.EMAIL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的主要 POC 邮件字段。*
{INVENTORY.POC.PRIMARY.EMAIL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC email field in host inventory.*

{INVENTORY.POC.PRIMARY.NAME<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 POC 名称字段。*
{INVENTORY.POC.PRIMARY.NAME<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC name field in host inventory.*

{INVENTORY.POC.PRIMARY.NOTES<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中 POC 备注字段。*
{INVENTORY.POC.PRIMARY.NOTES<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC notes field in host inventory.*

{INVENTORY.POC.PRIMARY.PHONE.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的主要 POC 联系电话字段。*
{INVENTORY.POC.PRIMARY.PHONE.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC phone A field in host inventory.*

{INVENTORY.POC.PRIMARY.PHONE.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的主要 POC 联系电话字段。*
{INVENTORY.POC.PRIMARY.PHONE.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC phone B field in host inventory.*

{INVENTORY.POC.PRIMARY.SCREEN<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的主要 POC screen 名称字段。*
{INVENTORY.POC.PRIMARY.SCREEN<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Primary POC screen name field in host inventory.*

{INVENTORY.POC.SECONDARY.CELL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的辅助 POC cell 字段。*
{INVENTORY.POC.SECONDARY.CELL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC cell field in host inventory.*

{INVENTORY.POC.SECONDARY.EMAIL<1-9>}	→ 基于 Trigger 的通知 * 主机清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的辅助 POC 电子邮件字段。*
{INVENTORY.POC.SECONDARY.EMAIL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC email field in host inventory.*

{INVENTORY.POC.SECONDARY.NAME<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的辅助 POC 名称字段。*
{INVENTORY.POC.SECONDARY.NAME<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC name field in host inventory.*

{INVENTORY.POC.SECONDARY.NOTES<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的辅助 POC 备注字段。*
{INVENTORY.POC.SECONDARY.NOTES<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC notes field in host inventory.*

{INVENTORY.POC.SECONDARY.PHONE.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中辅助 POC 电话号码字段。*
{INVENTORY.POC.SECONDARY.PHONE.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC phone A field in host inventory.*

{INVENTORY.POC.SECONDARY.PHONE.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中辅助 POC 电话号码字段。*
{INVENTORY.POC.SECONDARY.PHONE.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC phone B field in host inventory.*

{INVENTORY.POC.SECONDARY.SCREEN<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的辅助 POC screen 名称字段。*
{INVENTORY.POC.SECONDARY.SCREEN<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Secondary POC screen name field in host inventory.*

{INVENTORY.SERIALNO.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 宏 {→ 故障更新通知 → 事件 tag 的名称和值 | 中的序列号字段。* ROFILE.SERIALNO<1-9>} 已经不被支持。	
{INVENTORY.SERIALNO.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Serial number A field in host inventory.* {PROFILE.SERIALNO<1-9>} is deprecated.

{INVENTORY.SERIALNO.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的序列号字段。*
{INVENTORY.SERIALNO.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Serial number B field in host inventory.*

{INVENTORY.SITE.ADDRESS.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的站点地址字段。*
{INVENTORY.SITE.ADDRESS.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site address A field in host inventory.*

{INVENTORY.SITE.ADDRESS.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的站点地址字段。*
{INVENTORY.SITE.ADDRESS.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site address B field in host inventory.*

{INVENTORY.SITE.ADDRESS.C<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的站点地址字段。*

{INVENTORY.SITE.ADDRESS.C<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site address C field in host inventory.*

{INVENTORY.SITE.CITY<1-9>}	→ 基于 Trigger 的通知 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 中的站点城市字段。*
{INVENTORY.SITE.CITY<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site city field in host inventory.*

{INVENTORY.SITE.COUNTRY<1-9>}	→ 基于 Trigger 的通知 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 中站点所属国家字段。*
{INVENTORY.SITE.COUNTRY<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site country field in host inventory.*

{INVENTORY.SITE.NOTES<1-9>}	→ 基于 Trigger 的通知 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 中站点备注字段。*
{INVENTORY.SITE.NOTES<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site notes field in host inventory.*

{INVENTORY.SITE.RACK<1-9>}	→ 基于 Trigger 的通知 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 中的站点机架位置字段。*
{INVENTORY.SITE.RACK<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site rack location field in host inventory.*

{INVENTORY.SITE.STATE<1-9>}	→ 基于 Trigger 的通知 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 中站点所属州/省字段。*
{INVENTORY.SITE.STATE<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site state/province field in host inventory.*

{INVENTORY.SITE.ZIP<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的站点邮编字段。*
{INVENTORY.SITE.ZIP<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Site ZIP/postal field in host inventory.*

{INVENTORY.SOFTWARE<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的软件描述字段。* 宏 {PROFILE.SOFTWARE<1-9>} 已经不被支持。
{INVENTORY.SOFTWARE<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software field in host inventory.* {PROFILE.SOFTWARE<1-9>} is deprecated.

{INVENTORY.SOFTWARE.APP.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的应用软件字段。*
{INVENTORY.SOFTWARE.APP.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software application A field in host inventory.*

{INVENTORY.SOFTWARE.APP.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的应用软件字段。*
{INVENTORY.SOFTWARE.APP.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software application B field in host inventory.*

{INVENTORY.SOFTWARE.APP.C<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的应用软件字段。*
{INVENTORY.SOFTWARE.APP.C<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software application C field in host inventory.*

{INVENTORY.SOFTWARE.APP.D<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的应用软件字段。*
{INVENTORY.SOFTWARE.APP.D<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software application D field in host inventory.*

{INVENTORY.SOFTWARE.APP.E<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的应用软件字段。*
{INVENTORY.SOFTWARE.APP.E<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software application E field in host inventory.*

{INVENTORY.SOFTWARE.FULL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的软件详细描述字段。*
{INVENTORY.SOFTWARE.FULL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Software (Full details) field in host inventory.*

{INVENTORY.TAG<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 Tag 字段。* 宏 {PROFILE.TAG<1-9>} 已经不被支持。
{INVENTORY.TAG<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Tag field in host inventory.* {PROFILE.TAG<1-9>} is deprecated.

{INVENTORY.TYPE<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的类型字段。* 宏 {PROFILE.DEVICETYPE<1-9>} 已经不被支持。
{INVENTORY.TYPE<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Type field in host inventory.* {PROFILE.DEVICETYPE<1-9>} is deprecated.

{INVENTORY.TYPE.FULL<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的详细类型描述字段。*
{INVENTORY.TYPE.FULL<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Type (Full details) field in host inventory.*

{INVENTORY.URL.A<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 URL 字段。*
{INVENTORY.URL.A<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*URL A field in host inventory.*

{INVENTORY.URL.B<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 URL 字段。*
{INVENTORY.URL.B<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*URL B field in host inventory.*

{INVENTORY.URL.C<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的 URL 字段。*
{INVENTORY.URL.C<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*URL C field in host inventory.*

{INVENTORY.VENDOR<1-9>}	→ 基于 Trigger 的通知 * 设备清 → 内部通知 → 故障更新通知 → 事件 tag 的名称和值	中的供应商字段。*
{INVENTORY.VENDOR<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → Event tags and values	*Vendor field in host inventory.*

{ITEM.DESCRIPTION<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 内部通知 → 故障更新通知	达式中导致发送通知的第 N 个 item 的描述信息。* 从 2.0.0 开始支持。
{ITEM.DESCRIPTION<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Description of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.0.

{ITEM.ID<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 内部通知 → 故障更新通知 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	达式中导致发送通知的第 N 个 item 的数字标识。* 从 1.8.12 开始支持。
{ITEM.ID<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Numeric ID of the Nth item in the trigger expression that caused a notification.* Supported since 1.8.12.

{ITEM.KEY<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 内部通知 宏 {→ 故障更新通知 → HTTP agent 的 item 类型, item 原型和发现规则字段: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.| 达式中导致发送通知的第 N 个 item 的 key。* 从 2.0.0 开始支持。 RIGGER.KEY} 已经不被支持。	
{ITEM.KEY<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Key of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.0. `{TRIGGER.KEY}` is deprecated.

{ITEM.KEY.ORIG<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 内部通知 → 故障更新通知	达式中导致发送通知的第 N 个 item 的原始 key。* 从 2.0.6 开始支持。
{ITEM.KEY.ORIG<1-9>}	→ Trigger-based notifications → Internal notifications → Problem update notifications	*Original key (with macros not expanded) of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.6.

{ITEM.LASTVALUE<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 故障更新通知 如果最近一 → Trigger 名称和描述 从 1.4→ 事件 tag 的名称和值	达式中导致发送通知的第 N 个 item 的最近一个值。<*br*> 历史值采集时间已经超过参数 ZBX_HISTORY_PERIOD 定义的历史数据保存时间，那么在前端会显示值为 **UNKNOWN*。(参数 ZBX_HISTORY_PERIOD* 定义于defines.inc.php)。 3 开始支持。该宏等同于宏 `{{HOST.HOST}:{ITEM.KEY}.last()}` 从 Zabbix 3.2.0 开始支持自定义 宏值。
{ITEM.LASTVALUE<1-9>}	→ Trigger-based notifications → Problem update notifications → Trigger names and descriptions → Event tags and values	*The latest value of the Nth item in the trigger expression that caused a notification.* It will resolve to *UNKNOWN* in the frontend if the latest history value has been collected more than the *ZBX_HISTORY_PERIOD* time ago (defined in defines.inc.php). Supported since 1.4.3. It is alias to `{{HOST.HOST}:{ITEM.KEY}.last()}` Customizing the macro value is supported for this macro; starting with Zabbix 3.2.0.

{ITEM.LOG.AGE<1-9>}	→ 基于 Trigger 的通知 * 日志 i→ 故障更新通知	em 事件的持续时间。*
{ITEM.LOG.AGE<1-9>}	→ Trigger-based notifications → Problem update notifications	*Age of the log item event.*

{ITEM.LOG.DATE<1-9>} {ITEM.LOG.DATE<1-9>}	→ 基于 Trigger 的通知 * 日志 i→ 故障更新通知 → Trigger-based notifications → Problem update notifications	em 事件的发生时间。* *Date of the log item event.*

{ITEM.LOG.EVENTID<1-9>} {ITEM.LOG.EVENTID<1-9>}	→ 基于 Trigger 的通知 * 日志事 → 故障更新通知仅用于 Wi → Trigger-based notifications → Problem update notifications	的标识。* dows 事件日志监控。 *ID of the event in the event log.* For Windows event log monitoring only.

{ITEM.LOG.NSEVERITY<1-9>} {ITEM.LOG.NSEVERITY<1-9>}	→ 基于 Trigger 的通知 * 日志事 → 故障更新通知仅用于 Wi → Trigger-based notifications → Problem update notifications	的级别。* dows 事件日志监控。 *Numeric severity of the event in the event log.* For Windows event log monitoring only.

{ITEM.LOG.SEVERITY<1-9>} {ITEM.LOG.SEVERITY<1-9>}	→ 基于 Trigger 的通知 * 日志事 → 故障更新通知仅用于 Wi → Trigger-based notifications → Problem update notifications	的级别。* dows 事件日志监控。 *Verbal severity of the event in the event log.* For Windows event log monitoring only.

{ITEM.LOG.SOURCE<1-9>} {ITEM.LOG.SOURCE<1-9>}	→ 基于 Trigger 的通知 * 日志事 → 故障更新通知仅用于 Wi → Trigger-based notifications → Problem update notifications	的来源。* dows 事件日志监控。 *Source of the event in the event log.* For Windows event log monitoring only.

{ITEM.LOG.TIME<1-9>} {ITEM.LOG.TIME<1-9>}	→ 基于 Trigger 的通知 * 日志 i→ 故障更新通知 → Trigger-based notifications → Problem update notifications	em 事件的发生时间。* *Time of the log item event.*

{ITEM.NAME<1-9>} {ITEM.NAME<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 内部通知 → 故障更新通知 → Trigger-based notifications → Internal notifications → Problem update notifications	达式中导致发送通知的第 N 个 item 的名称。* *Name of the Nth item (with macros resolved) in the trigger expression that caused a notification.*

{ITEM.NAME.ORIG<1-9>} {ITEM.NAME.ORIG<1-9>}	→ 基于 Trigger 的通知 * 触发器 → 内部通知 从 2.→ 故障更新通知 → Trigger-based notifications → Internal notifications → Problem update notifications	达式中导致发送通知的第 N 个 item 的原始名称。* .6 开始支持。 *Original name (i.e. without macros resolved) of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.6.

{ITEM.STATE<1-9>}	→ 基于 Item 的内部通知 * 触发器表达	中导致发送通知的第 N 个 item 的状态。* 可能的值: **Not supported** 和 **Normal**. 从 2.2.0 开始支持。
{ITEM.STATE<1-9>}	→ Item-based internal notifications	*The latest state of the Nth item in the trigger expression that caused a notification.* Possible values: **Not supported** and **Normal**. Supported since 2.2.0.

{ITEM.VALUE<1-9>}	→ 基于 Trigger 的通知 可能的值 → 故障更新通知 1) 如果 → Trigger 名称和描述 2) 如 → 事件 tag 的名称和值在第一种情况	触发器状态更改的上下文中使用, 例如, 显示事件或发送通知。该值为触发器表达式中的第 N 个 item 的历史（at-the-time-of-event）值。 不在触发器状态更改的上下文中使用, 例如, 在弹出窗口中显示触发器列表时，该值为触发器表达式中的第 N 个 item 的最近一个值，类似于 {ITEM.LASTVALUE}。历史数据被删除或未入库时候值解析为 *UNKNOWN*。 在第二种情况如果最近一个历史值采集时间已经超过参数 *ZBX_HISTORY_PERIOD* 定义的历史数据保存时间，那么在前端会显示值为 *UNKNOWN*。(参数 *ZBX_HISTORY_PERIOD* 定义于defines.inc.php). 从 1.4.3 开始支持。 从 Zabbix 3.2.0 开始支持自定义宏值。
{ITEM.VALUE<1-9>}	→ Trigger-based notifications → Problem update notifications → Trigger names and descriptions → Event tags and values	Resolved to either: 1) the historical (at-the-time-of-event) value of the Nth item in the trigger expression, if used in the context of trigger status change, for example, when displaying events or sending notifications. 2) the latest value of the Nth item in the trigger expression, if used without the context of trigger status change, for example, when displaying a list of triggers in a pop-up selection window. In this case works the same as {ITEM.LASTVALUE} In the first case it will resolve to *UNKNOWN* if the history value has already been deleted or has never been stored. In the second case, and in the frontend only, it will resolve to *UNKNOWN* if the latest history value has been collected more than the *ZBX_HISTORY_PERIOD* time ago (defined in defines.inc.php). Supported since 1.4.3. Customizing the macro value is supported for this macro, starting with Zabbix 3.2.0.

{LLDRULE.DESCRIPTION}	→ LLD-rule based 内部通知 * 触发	知的 low-level 发现规则描述。* 从 2.2.0 开始支持。
{LLDRULE.DESCRIPTION}	→ LLD-rule based internal notifications	*Description of the low-level discovery rule 、which caused a notification.* Supported since 2.2.0.
{LLDRULE.ID}	→ LLD-rule based 内部通知 * 触发	知的 low-level 发现规则的数字标识。* 从 2.2.0 开始支持。
{LLDRULE.ID}	→ LLD-rule based internal notifications	*Numeric ID of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.KEY}	→ LLD-rule based 内部通知 * 触发	知的 low-level 发现规则的 key。* 从 2.2.0 开始支持。
{LLDRULE.KEY}	→ LLD-rule based internal notifications	*Key of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.KEY.ORIG}	→ LLD-rule based 内部通知 * 触发	知的 low-level 发现规则的原始 key（未扩展宏）。* 从 2.2.0 开始支持。
{LLDRULE.KEY.ORIG}	→ LLD-rule based internal notifications	*Original key (with macros not expanded) of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.NAME}	→ LLD-rule based 内部通知 * 触发	知的 low-level 发现规则的名称（未扩展宏）。* 从 2.2.0 开始支持。
{LLDRULE.NAME}	→ LLD-rule based internal notifications	*Name of the low-level discovery rule (with macros resolved) that caused a notification.* Supported since 2.2.0.
{LLDRULE.NAME.ORIG}	→ LLD-rule based 内部通知 * 触发	知的 low-level 发现规则的原始名称（未扩展宏）。* 从 2.2.0 开始支持。
{LLDRULE.NAME.ORIG}	→ LLD-rule based internal notifications	*Original name (i.e. without macros resolved) of the low-level discovery rule that caused a notification.* Supported since 2.2.0.
{LLDRULE.STATE}	→ LLD-rule based 内部通知 *lo	-level 发现规则的最新状态。* 可能的值: **Not supported** 和 **Normal**. 从 2.2.0 开始支持。
{LLDRULE.STATE}	→ LLD-rule based internal notifications	*The latest state of the low-level discovery rule.* Possible values: **Not supported** and **Normal**. Supported since 2.2.0.
{MAP.ID}	→ 地图 URLs *	络地图标识。*

{MAP.ID}	→ Map URLs	*Network map ID.*

{MAP.NAME}	→ 地图形状中的文字描述字段 网络地图名称。	< 从 3.4.0 开始支持。
{MAP.NAME}	→ Text field in map shapes	*Network map name.* Supported since 3.4.0.

{PROXY.DESCRIPTION<1-9>}	→ 基于 Trigger 的通知和命令 → 发现通知 → 自动注册通知 → 内部通知 → 故障更新通知
proxy 描述信息。可能的值: 1) 触发器表达式中第 N 个项的 proxy 的信息（基于 Trigger 的通知）。可以使用宏索引。 2) 执行发现的 proxy 信息 (发现通知)。可以使用宏 {PROXY.DESCRIPTION}，而不带宏索引。 3) 主动 agent 注册的 proxy 信息。(自动注册通知)。可以使用宏 {PROXY.DESCRIPTION}，而不带宏索引。 从 2.4.0 开始支持。	
{PROXY.DESCRIPTION<1-9>}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications
Description of the proxy. Resolves to either: 1) proxy of the Nth item in the trigger expression (in trigger-based notifications). You may use indexed macros here. 2) proxy, which executed discovery (in discovery notifications). Use {PROXY.DESCRIPTION} here, without indexing. 3) proxy to which an active agent registered (in auto-registration notifications). Use {PROXY.DESCRIPTION} here, without indexing. Supported since 2.4.0.	

{PROXY.NAME<1-9>}	→ 基于 Trigger 的通知和命令 *proxy 的 → 发现通知 1)→ 自动注册通知 2) 执行 → 内部通知 3)→ 故障更新通知从 1.8.	称。*。可能的值: 发器表达式中第 N 个项的 proxy 的名称（基于 Trigger 的通知）。可以使用宏索引。 现的 proxy 名称 (发现通知)。可以使用宏 {PROXY.NAME} ，而不带宏索引。 动 agent 注册的 proxy 名称。(自动注册通知)。可以使用宏 {PROXY.NAME}，而不带宏索引。 开始支持。

{PROXY.NAME<1-9>}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Name of the proxy*. Resolves to either: 1) proxy of the Nth item in the trigger expression (in trigger-based notifications). You may use indexed macros here. 2) proxy, which executed discovery (in discovery notifications). Use {PROXY.NAME} here, without indexing. 3) proxy to which an active agent registered (in auto-registration notifications). Use {PROXY.NAME} here, without indexing. Supported since 1.8.4.

{TIME}	→ 基于 Trigger 的通知和命令 * 时间格式为：→ 发现通知 → 自动注册通知 → 内部通知 → 故障更新通知	h:mm:ss.*
{TIME}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	*Current time in hh:mm:ss.*

{TRIGGER.DESCRIPTION}	→ 基于 Trigger 的通知 *Tri→ Trigger-based 内部通知 从 2.→ 故障更新通知宏 {TR|ger 描述信息。* 从 2.0.4 开始支持。 .0 开始如果在通知文本中使用“{TRIGGER.DESCRIPTION}”，trigger 描述将支持所有宏。 GGER.COMMENT} 已经不被支持。	
{TRIGGER.DESCRIPTION}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Trigger description.* Supported since 2.0.4. Starting with 2.2.0, all macros supported in a trigger description will be expanded if {TRIGGER.DESCRIPTION} is used in notification text. {TRIGGER.COMMENT} is deprecated.

{TRIGGER.EVENTS.ACK}	→ 基于 Trigger 的通知 * 地图中 → 故障更新通知 → 地图中的 Icon 标签[1]	素的已确认事件数，或者在通知中生成当前事件的触发器的已确认事件数。* 从 1.8.3 开始支持。
{TRIGGER.EVENTS.ACK}	→ Trigger-based notifications → Problem update notifications → Icon labels in maps[1]	*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}	→ 基于 Trigger 的通知 * 忽略状 → 故障更新通知 → 地图中的 Icon 标签[1]	的所有触发器的已确认故障事件数。* 从 1.8.3 开始支持。

{TRIGGER.EVENTS.PROBLEM.ACK}	→ Trigger-based notifications → Problem update notifications → Icon labels in maps[1]	*Number of acknowledged PROBLEM events for all triggers disregarding their state.* Supported since 1.8.3.

{TRIGGER.EVENTS.PROBLEM.UNACK}	→ 基于 Trigger 的通知 * 忽略状 → 故障更新通知 → 地图中的 Icon 标签[1]	的所有触发器的未确认故障事件数。* 从 1.8.3 开始使用。
{TRIGGER.EVENTS.PROBLEM.UNACK}	→ Trigger-based notifications → Problem update notifications → Icon labels in maps[1]	*Number of unacknowledged PROBLEM events for all triggers disregarding their state.* Supported since 1.8.3.

{TRIGGER.EVENTS.UNACK}	→ 基于 Trigger 的通知 * 地图中 → 故障更新通知 → 地图中的 Icon 标签[1]	素的未确认事件数，或者通知中生成当前事件的触发器的未确认事件数。* 从 1.8.3 开始支持地图元素标签。
{TRIGGER.EVENTS.UNACK}	→ Trigger-based notifications → Problem update notifications → Icon labels in maps[1]	*Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications.* Supported in map element labels since 1.8.3.

{TRIGGER.HOSTGROUP.NAME}	→ 基于 Trigger 的通知 * 基于 S→ 故障更新通知 → 基于 Trigger 内部通知	L 查询排序，逗号-空格分隔的 trigger 所属的设备组列表。* 从 2.0.6 开始支持。
{TRIGGER.HOSTGROUP.NAME}	→ Trigger-based notifications → Problem update notifications → Trigger-based internal notifications	*A sorted (by SQL query), comma-space separated list of host groups in which the trigger is defined.* Supported since 2.0.6.

{TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}	→地图 Icon 标签[1] * 触发	状态为问题的已确认问题事件数。* 从 1.8.3 开始支持。
{TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}	→Icon labels in maps[1]	*Number of acknowledged PROBLEM events for triggers in PROBLEM state.* Supported since 1.8.3.

{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}	→地图 Icon 标签[1] * 触发	状态为问题的未确认问题事件数。* 从 1.8.3 开始支持。
{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}	→Icon labels in maps[1]	*Number of unacknowledged PROBLEM events for triggers in PROBLEM state.* Supported since 1.8.3.

{TRIGGER.EXPRESSION}	→ 基于 Trigger 的通知 *Tri→ 基于 Trigger 内部通知 → 故障更新通知	ger 表达式。* 从 1.8.12 开始支持。
{TRIGGER.EXPRESSION}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Trigger expression.* Supported since 1.8.12.

{TRIGGER.EXPRESSION.RECOVERY}	→ 基于 Trigger 的通知 *Tri→ 基于 Trigger 内部通知 从 3.2.→ 故障更新通知	ger 恢复表达式。如果 恢复事件 * 在trigger 配置中设置为’Recovery expression’ 则返回表达式，否则返回空字符串。 开始支持。

{TRIGGER.EXPRESSION.RECOVERY}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Trigger recovery expression* if *OK event generation* in trigger configuration is set to 'Recovery expression'; otherwise an empty string is returned. Supported since 3.2.0.

{TRIGGER.ID}	→ 基于 Trigger 的通知 * 触发动 → Trigger-based 内部通知 从 1.→ 故障更新通知 → 图形 URLs → Trigger URLs	的 Trigger 数字标识。* .8 开始支持 trigger URLs。
{TRIGGER.ID}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications → Map URLs → Trigger URLs	*Numeric trigger ID which triggered this action.* Supported in trigger URLs since Zabbix 1.8.8.

{TRIGGER.NAME}	→ 基于 Trigger 的通知 *tri→ 基于 Trigger 内部通知 从 4.0→ 故障更新通知	ger 名称。* (支持宏解析)。 0 开始宏 {EVENT.NAME} 不能用于动作中去显示触发事件名称（支持宏解析）。
{TRIGGER.NAME}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Name of the trigger* (with macros resolved). Note that since 4.0.0 {EVENT.NAME} can be used in actions to display the triggered event/problem name with macros resolved.

{TRIGGER.NAME.ORIG}	→ 基于 Trigger 的通知 *tri→ 基于 Trigger 内部通知 从 2.0.→ 故障更新通知	ger 的原始名称 * (即没有宏解析). 开始支持。
{TRIGGER.NAME.ORIG}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Original name of the trigger* (i.e. without macros resolved). Supported since 2.0.6.

{TRIGGER.NSEVERITY}	→ 基于 Trigger 的通知 *tri→ 基于 Trigger 内部通知 从 Zabb→ 故障更新通知	ger 数字级别。* 可能的值: 0 - 未定义, 1 - 信息, 2 - 警告, 3 - 普通, 4 - 严重, 5 - 灾难. x 1.6.2 开始支持。
{TRIGGER.NSEVERITY}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Numerical trigger severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported starting from Zabbix 1.6.2.

{TRIGGER.SEVERITY}	→ 基于 Trigger 的通知 *Tri→ 基于 Trigger 内部通知 → 故障更新通知	ger 级别名称.* 可在管理 → 通用 → *Trigger* 级别功能中定义。
{TRIGGER.SEVERITY}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Trigger severity name.* Can be defined in *Administration → General → Trigger severities*.

{TRIGGER.STATE}	→ 基于 Trigger 内部通知 *trig	er 的最新状态。* 可能的值: **Unknown** and **Normal**. 从 2.2.0 开始支持。
{TRIGGER.STATE}	→ Trigger-based internal notifications	*The latest state of the trigger.* Possible values: **Unknown** and **Normal**. Supported since 2.2.0.

{TRIGGER.STATUS}	→ 基于 Trigger 的通知 * 当前 t→ 故障更新通知宏 {ST|igger 的值。* 可能是 PROBLEM 或 OK. TUS} 已经不被支持。	
{TRIGGER.STATUS}	→ Trigger-based notifications → Problem update notifications	*Current trigger value.* Can be either PROBLEM or OK. `{STATUS}` is deprecated.

{TRIGGER.TEMPLATE.NAME}	→ 基于 Trigger 的通知 * 排序（→ 基于 Trigger 内部通知 → 故障更新通知	过 SQL 查询），逗号-空格分隔的触发器所属模板列表，如果触发器应用于具体设备，则为 * UNKNOWN ** 从 2.0.6 开始支持。
{TRIGGER.TEMPLATE.NAME}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*A sorted (by SQL query), comma-space separated list of templates in which the trigger is defined, or *UNKNOWN* if the trigger is defined in a host.* Supported since 2.0.6.

{TRIGGER.URL}	→ 基于 Trigger 的通知 *Tri→ 基于 Trigger 内部通知 → 故障更新通知	ger URL.*
{TRIGGER.URL}	→ Trigger-based notifications → Trigger-based internal notifications → Problem update notifications	*Trigger URL.*

{TRIGGER.VALUE}	→ 基于 Trigger 的通知 * 触发器 → Trigger 表达式 → 故障更新通知	当前值。*: 0 - trigger 状态为 OK, 1 - trigger 状态为 PROBLEM。
{TRIGGER.VALUE}	→ Trigger-based notifications → Trigger expressions → Problem update notifications	*Current trigger numeric value*: 0 - trigger is in OK state, 1 - trigger is in PROBLEM state.

{TRIGGERS.UNACK}	→ 地图 Icon 标签[1] * 忽略	发器状态，地图元素的未确认触发器数。*\\如果至少有一个 PROBLEM 事件未被确认，则认为触发器未被确认。
{TRIGGERS.UNACK}	→ Icon labels in maps[1]	*Number of unacknowledged triggers for a map element, disregarding trigger state.* A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged.

{TRIGGERS.PROBLEM.UNACK}	→ 地图 Icon 标签[1] * 地图	素的未确认触发器（状态为 PROBLEM）数。* 如果至少有一个 PROBLEM 事件未被确认，则认为触发器未被确认。 从 1.8.3 开始支持。
{TRIGGERS.PROBLEM.UNACK}	→ Icon labels in maps[1]	*Number of unacknowledged PROBLEM triggers for a map element.* A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged. Supported since 1.8.3.

{TRIGGERS.ACK}	→ 地图 Icon 标签[1] * 忽略	发器状态，地图元素的确认触发器数，* 当所有 PROBLEM 事件都被确认后，trigger 才被认为已经确认。 从 1.8.3 开始支持。
{TRIGGERS.ACK}	→ Icon labels in maps[1]	*Number of acknowledged triggers for a map element, disregarding trigger state.* A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged. Supported since 1.8.3.

{TRIGGERS.PROBLEM.ACK}	→ 地图 Icon 标签[1] * 地图	素的确认触发器（状态为 PROBLEM）数。* 当所有 PROBLEM 事件都被确认后，trigger 才被认为已经确认。 从 1.8.3 开始支持。
{TRIGGERS.PROBLEM.ACK}	→ Icon labels in maps[1]	*Number of acknowledged PROBLEM triggers for a map element.* A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged. Supported since 1.8.3.

{USER.FULLNAME}	→ 故障更新通知 * 事件确认	作的用户全名。* 从 3.4.0 开始支持。
{USER.FULLNAME}	→ Problem update notifications	*Name and surname of the user* who added event acknowledgement. Supported since 3.4.0.

{host:key.func(param)}	→ 基于 Trigger 的通知 * 简单的 → 故障更新通知 → 地图 Icon/shape 标签[1] [4] 从 3.→ 地图 Link 标签[4] → 图形名称[7] → Trigger 表达式[9]	，用于构建触发器表达式*。 .2 开始支持 shape 标签。
{$MACRO}	→ 参考: 用户自定义宏使用场景 *[用户自定义](/m	nual/config/macros/user_macros) 宏。*
{#MACRO}	→ 参考: Low-level 发现宏 *Low	level 发现宏 * 从 2.0.0 开始支持。

{host:key.func(param)}	→ Trigger-based notifications → Problem update notifications → Icon/shape labels in maps[1] [4] → Link labels in maps[4] → Graph names[7] → Trigger expressions[9]	*Simple macros, as used in building trigger expressions*. Supported for shape labels in maps since 3.4.2.
{$MACRO}	→ See: User macros supported by location	*User-definable macros.*
{#MACRO}	→ See: Low-level discovery macros	*Low-level discovery macros.* Supported since 2.0.0.

脚注

Footnotes

[1] 从 1.8 开始地图标签支持宏。[1] Macros for map labels are supported since 1.8.

[2] 宏 {HOST.*} 用于 item key 参数将解析为所选 item 的接口。如果 item 无接口，将按优先顺序解析为设备的 Zabbix agent,SNMP,JMX，IPMI 接口。[2] The {HOST.*} macros supported in item key parameters will resolve to the interface that is selected for the item. When used in items without interfaces they will resolve to either the Zabbix agent, SNMP, JMX or IPMI interface of the host in this order of priority.

[3] 在 remote commands, global scripts, interface IP/DNS 字段和 web scenarios 宏将解析为主代理接口。如果不存在，则使用 SNMP 接口。如果 SNMP 接口也不存在，则使用 JMX 接口。如果 JMX 接口不存在则使用 IPMI 接口。[3] In remote commands, global scripts, interface IP/DNS fields and web scenarios the macro will resolve to the main agent interface, however, if it is not present, the main SNMP interface will be used. If SNMP is also not present, the main JMX interface will be used. If JMX is not present either, the main IPMI interface will be used.

[4] 地图标签中的宏仅仅支持 **avg**, **last**, **max** and **min** 函数, 以秒为单位。[4] Only the **avg**, **last**, **max** and **min** functions, with seconds as parameter are supported in this macro in map labels.

[5] 从 2.0.3 开始支持。

[6] 从 Zabbix 2.2.0 开始, 宏 {HOST.*} 可以用于 web scenario 中的 *Name*, *Variables*, *Headers*, *SSL certificate file* and *SSL key file* fields and in scenario step *Name*, *URL*, *Post*, *Headers* and *Required string* 字段。[6] Supported since Zabbix 2.2.0, {HOST.*} macros are supported in web scenario *Name*, *Variables*, *Headers*, *SSL certificate file* and *SSL key file* fields and in scenario step *Name*, *URL*, *Post*, *Headers* and *Required string* fields.

[7] 从 Zabbix 2.2.0 开始，地图标签中的宏仅支持 avg，last，max 和 min 函数，以秒为参数。宏 {HOST.HOST<1-9>} 可以用于引用某个设备。例如:

```
* {Cisco switch:ifAlias[{#SNMPINDEX}].last()}
* %%{{%%HOST.HOST}:ifAlias[{#SNMPINDEX}].last()}
```

[8] 从 2.4.0 开始支持。

[9] 虽然支持构建触发器表达式，但是不能在彼此内部使用简单的宏。[9] While supported to build trigger expressions, simple macros may not be used inside each other.

[10] 从 3.0.0 开始支持。[10] Supported since 3.0.0.

宏索引

Indexed macros

宏索引 {MACRO**<1-9>**} 语法仅限于触发器表达式的上下文。它能用于按顺序引用表达式中包含的设备。例如：在表达式中包含了设备 1，设备 2，设备 3，那么宏 {HOST.IP1}, {HOST.IP2}, {HOST.IP3} 将分别引用设备 1，设备 2，设备 3 的 IP 地址信息。The indexed macro syntax of {MACRO**<1-9>**} is limited to the context of **trigger expressions**. It can be used to reference hosts in the order in which they appear in the expression. Macros like {HOST.IP1}, {HOST.IP2}, {HOST.IP3} will resolve to the IP of the first, second and third host in the trigger expression (providing the trigger expression contains those hosts).

另外，可以在图形名称中使用宏 {host:key.func(param)}，同时再叠加使用宏 {HOST.HOST<1-9>}。示例, 图形名称中的宏 {{HOST.HOST2}:key.func()} 代表引用图形中的第二个设备。Additionally the {HOST.HOST<1-9>} macro is also supported within the {host:key.func(param)} macro in **graph names**. For example, {{HOST.HOST2}:key.func()} in the graph name will refer to the host of the second item in the graph.

Warning:
有些场景可以使用不带索引的宏。(例如：{HOST.HOST}, {HOST.IP}, 等)

Warning:
Use macros **without** index (i. e.{HOST.HOST}, {HOST.IP}, etc) in all other contexts.

2 用户自定义宏使用场景

2 User macros supported by location v

概述

Overview

用户自定义宏可以用于以下场景: User-definable macros are supported in the following locations:

- 设备
  - 接口 IP/DNS
  - 接口 port
- Hosts
  - Interface IP/DNS
  - Interface port
- 被动 proxy
  - 接口 port
- Items 和 item 原型
  - 名称
  - Key 参数
  - 更新间隔
  - 用户自定义间隔
  - 历史数据存储周期
  - 趋势数据存储周期
  - SNMPv3 context 名称
  - SNMPv3 security 名称
  - SNMPv3 auth 密码
  - SNMPv3 priv 密码
  - SNMPv1/v2 团体串
  - SNMP OID
  - SNMP port
  - SSH 用户名
  - SSH 公钥
  - SSH 私钥
  - SSH 密码
  - SSH 脚本
  - Telnet 用户名
  - Telnet 密码
  - Telnet 脚本
  - Calculated item 公式
  - Trapper item "Allowed hosts" 字段
  - 数据库监控附加字段
  - JMX item endpoint 字段
  - 从 Zabbix 4.0 开始支持以下字段:
    - item 值预处理步骤
    - HTTP agent URL 字段
    - HTTP agent HTTP query fields 字段
    - HTTP agent request body 字段
    - HTTP agent required status codes 字段
    - HTTP agent headers field key 和 value
    - HTTP agent HTTP 认证用户名字段
    - HTTP agent HTTP 认证密码字段
    - HTTP agent HTTP proxy 字段
    - HTTP agent SSL certificate 文件字段
    - HTTP agent SSL key 文件字段
    - HTTP agent SSL key 密码字段
    - HTTP agent HTTP 超时字段
    - HTTP agent HTTP 允许设备字段

- Items and item prototypes
  - Name
  - Key parameters
  - Update interval
  - Custom intervals
  - History storage period
  - Trend storage period
  - SNMPv3 context name
  - SNMPv3 security name
  - SNMPv3 auth pass
  - SNMPv3 priv pass
  - SNMPv1/v2 community
  - SNMP OID
  - SNMP port
  - SSH username
  - SSH public key
  - SSH private key
  - SSH password
  - SSH script
  - Telnet username
  - Telnet password
  - Telnet script
  - Calculated item formula
  - Trapper item "Allowed hosts" field
  - Database monitoring additional parameters
  - JMX item endpoint field
  - since Zabbix 4.0 also in:
    * item value preprocessing steps
    * HTTP agent URL field
    * HTTP agent HTTP query fields field
    * HTTP agent request body field
    * HTTP agent required status codes field
    * HTTP agent headers field key and value
    * HTTP agent HTTP authentication username field
    * HTTP agent HTTP authentication password field
    * HTTP agent HTTP proxy field
    * HTTP agent SSL certificate file field
    * HTTP agent SSL key file field
    * HTTP agent SSL key password field
    * HTTP agent HTTP timeout field
    * HTTP agent HTTP allowed hosts field
- 发现

  * 更新间隔
  * SNMPv3 context 字段
  * SNMPv3 security 字段
  * SNMPv3 auth 密码
  * SNMPv3 priv 密码
  * SNMPv1/v2 团体串
  * SNMP OID
* Discovery
  * Update interval
  * SNMPv3 context name
  * SNMPv3 security name
  * SNMPv3 auth pass
  * SNMPv3 priv pass
  * SNMPv1/v2 community
  * SNMP OID

- Low-level 发现规则
  - 名称
  - Key 参数
  - 更新间隔

  - 用户自定义间隔
  - SNMPv3 context 名称
  - SNMPv3 security 名称
  - SNMPv3 auth 密码
  - SNMPv3 priv 密码
  - SNMPv1/v2 团体串
  - SNMP OID
  - SNMP port
  - SSH 用户名
  - SSH 公钥
  - SSH 私钥
  - SSH 密码
  - SSH 脚本
  - Telnet 用户名
  - Telnet 密码
  - Telnet 脚本
  - Trapper item 允许设备字段
  - 数据库监控附加字段
  - JMX item endpoint 字段
  - 保持资源丢失时间
  - 过滤正则表达式
  - 从 Zabbix 4.0 开始支持以下字段:
    * HTTP agent URL 字段
    * HTTP agent HTTP query fields 字段
    * HTTP agent request body 字段
    * HTTP agent required status codes 字段
    * HTTP agent headers field key 和 value
    * HTTP agent HTTP 认证用户名字段
    * HTTP agent HTTP 认证密码字段
    * HTTP agent HTTP 超时字段
- Low-level discovery rule
  - Name
  - Key parameters
  - Update interval
  - Custom intervals
  - SNMPv3 context name
  - SNMPv3 security name
  - SNMPv3 auth pass
  - SNMPv3 priv pass
  - SNMPv1/v2 community
  - SNMP OID
  - SNMP port
  - SSH username
  - SSH public key
  - SSH private key
  - SSH password
  - SSH script
  - Telnet username
  - Telnet password
  - Telnet script
  - Trapper item ”Allowed hosts” field
  - Database monitoring additional parameters
  - JMX item endpoint field
  - Keep lost resources period
  - Filter regular expressions
  - since Zabbix 4.0 also in:
    * HTTP agent URL field
    * HTTP agent HTTP query fields field
    * HTTP agent request body field
    * HTTP agent required status codes field
    * HTTP agent headers field key and value
    * HTTP agent HTTP authentication username field
    * HTTP agent HTTP authentication password field

        * HTTP agent HTTP timeout field

    - Web scenario

      * 名称
      * 更新间隔
      * Agent
      * HTTP proxy
      * Variables
      * Headers
      * Step name
      * Step URL
      * Step post variables
      * Step headers
      * Step timeout
      * Required string
      * 必须的状态
      * 认证（用户名和密码）
      * SSL certificate 文件
      * SSL key 文件
      * SSL key 密码
* Web scenario
      * Name
      * Update interval
      * Agent
      * HTTP proxy
      * Variables
      * Headers
      * Step name
      * Step URL
      * Step post variables
      * Step headers
      * Step timeout
      * Required string
      * Required status codes
      * Authentication (user and password)
      * SSL certificate file
      * SSL key file
      * SSL key password

* Triggers
      * 名称
      * 表达式（仅在常量和函数参数中）
      * 描述信息
      * URLs

    - Triggers

      * Name
      * Expression (only in constants and function parameters)
      * Description
      * URLs

* 基于Trigger通知
* 基于Trigger内部通知
* 问题更新通知
* Trigger-based notifications
* Trigger-based internal notifications
* Problem update notifications

* 事件tags
      * Tag 名称
      * Tag 值
      * Tag 匹配

* Event tags
    * Tag name
    * Tag value
    * Tag for matching

- 动作操作

    * 默认步骤持续时间
    * 步骤持续时间
* Action operations
    * Default operation step duration
    * Step duration

* 动作条件
    * 时间周期条件
* Action conditions
    * Time period condition

- 全局脚本 (包括确认文本)
- Global scripts (including confirmation text)

- dynamic URL screen 元素的 URL 字段。
- URL field of dynamic URL screen element

- 管理 → 用户 → 媒介: 'When active' 字段。
- 管理 → 一般 → 工作时间: 'Working time' 字段。
- Administration → Users → Media: 'When active' field
- Administration → General → Working time: 'Working time' field

需要 Zabbix 所支持宏的完整列表, 请参考支持宏列表。For a complete list of all macros supported Zabbix, see macros supported by location.

8 单位符号说明

8 Unit symbols Overview 概述

Having to use some large numbers, for example '86400' to represent the number of seconds in one day, is both difficult and error-prone. This is why you can use some appropriate unit symbols (or suffixes) to simplify Zabbix trigger expressions and item keys. 若使用一些大数字时，例如'86400' 来表示一天中的秒数，既困难又容易出错。这就是您可以使用一些适当的单位符号（或后缀）来简化 Zabbix trigger 表达式和 item key 的原因。

Instead of '86400' for the number of seconds you can simply enter '1d'. Suffixes function as multipliers. 您可以直接输入'1d'，而不是'86400' 的秒数。后缀 d 用作乘数。

Time suffixes 时间后缀

For time you can use 您可使用如下:

- **s** - seconds (when used, works the same as the raw value) 秒（使用时，与原始值相同）
- **m** - minutes 分
- **h** - hours 时
- **d** - days 天
- **w** - weeks 周

Time suffixes are supported in: 以下支持时间后缀：

- trigger expression constants and function parameters 触发器expression 常量和函数参数
- item configuration ('Update interval', 'Custom intervals', 'History storage period' and 'Trend storage period' fields) 监控项配置（' 更新间隔'，' 自定义时间间隔'，' 历史数据保留时长' 和' 趋势存储时间' 字段）
- item prototype configuration ('Update interval', 'Custom intervals', 'History storage period' and 'Trend storage period' fields) 监控项原型配置（' 更新间隔'，' 自定义时间间隔'，' 历史数据保留时长' 和' 趋势存储时间' 字段）
- low-level discovery rule configuration ('Update interval', 'Custom intervals', 'Keep lost resources' fields) 低级别发现规则配置（' 更新间隔'，' 自定义时间间隔'，' 资源周期不足' 字段）
- network discovery configuration ('Update interval' field) 网络发现规则配置（' 更新间隔' 字段）
- web scenario configuration ('Update interval', 'Timeout' fields) web scenario 配置（' 更新间隔'，' 超时' 字段）
- action operation configuration ('Default operation step duration', 'Step duration' fields) 动作操作配置（' 默认操作步骤持续时间'，' 步骤持续时间' 字段）

- slide show configuration ('Default delay' field) 幻灯片展示配置（' 默认延迟' 字段）
- user profile settings ('Auto-logout', 'Refresh', 'Message timeout' fields) 用户基本资料配置（' 自动登录' ，' 刷新' ，' 消息超时' 字段）
- *Administration* → *General* → *Housekeeping* (storage period fields) 管理 → 一般 → 管家 (' 存储期' 字段)
- *Administration* → *General* → *Trigger displaying options* ('Display OK triggers for', 'On status change triggers blink for' fields) 管理 → 一般 → 触发器显示选项 (' 显示 OK 触发器于' ，' 于状态改变时, 触发器因此闪烁于' 字段)
- *Administration* → *General* → *Other* ('Refresh unsupported items' field) 管理 → 一般 → 其他 (' 刷新不支持的项目' 字段)
- parameters of the **zabbix[queue,<from>,<to>]** internal item 参数 **zabbix[queue,<from>,<to>]** internal item
- last parameter of aggregate checks aggregate checks最后一个参数

Memory suffixes 内存后缀

Memory size suffixes are supported in trigger expression constants and function parameters. 触发器expression 常量和函数参数支持内存大小后缀。

For memory size you can use 对于内存大小，用法如下:

- **K** - kilobyte 千字节
- **M** - megabyte 兆字节
- **G** - gigabyte 十亿字节
- **T** - terabyte 兆兆字节

Other uses 其他用法

Unit symbols are also used for a human-readable representation of data in the frontend. 单位符号还用于前端数据，人们可读的表示法。

In both Zabbix server and frontend these symbols are supported: 在 Zabbix 服务器和前端都支持这些符号：

- **K** - kilo
- **M** - mega
- **G** - giga
- **T** - tera

When item values in B, Bps are displayed in the frontend, base 2 is applied (1K = 1024). Otherwise a base of 10 is used (1K = 1000). 当 B，Bps 中的监控项值显示在前端时，应用基数 2（1K = 1024），否则使用 10 的基数（1K = 1000）。

Additionally the frontend also supports the display of: 此外，前端还支持以下显示：

- **P** - peta
- **E** - exa
- **Z** - zetta
- **Y** - yotta

Usage examples 用法示例

By using some appropriate suffixes you can write trigger expressions that are easier to understand and maintain, for example these expressions: 通过使用一些适当的后缀，您可以编写更易于理解和维护的触发器表达式，例如以下表达式：

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[].last()}<86400
{host:system.cpu.load.avg(600)}<10
{host:vm.memory.size[available].last()}<20971520
```

could be changed to: 可以改为：

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last()}<1d
{host:system.cpu.load.avg(10m)}<10
{host:vm.memory.size[available].last()}<20M
```

9 时间段配置

9 Setting time periods Overview 概述

To set a time period, the following format has to be used: 若要设定一个时间段，就要运用下面的格式：

```
d-d,hh:mm-hh:mm
```

where the symbols stand for the following: 符号用法：

Symbol 符号 D	scription 描述
d	Day of the week: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday 星期: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday
hh	Hours: 00-24 小时: 00-24
mm	Minutes: 00-59 分钟: 00-59

You can specify more than one time period using a semicolon (;) separator: 您可以使用分号（;）分隔符指定多个时间段：

`d-d,hh:mm-hh:mm;d-d,hh:mm-hh:mm...`

Leaving the time period empty equals 01-07,00:00-24:00, which is the default value. 如果时间段的参数为空，系统将默认为 01-07,00:00-24:00

Attention:
The upper limit of a time period is not included. Thus, if you specify 09:00-18:00 the last second included in the time period is 17:59:59. This is true starting from version 1.8.7, for everything, while Working time has always worked this way. 时间段的上限是开区间。因此，当您指定时间段为 09:00-18:00 时，该时间段包含的最后一秒钟将是 17:59:59。该规则从 1.8.7 版本之后适用于各类设定，而Working time 一直沿用该规则。

Examples 举例

Working hours. Monday - Friday from 9:00 till 18:00: 工作日。星期一到星期五的 9:00 到 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: 工作日加上周末。星期一到星期五的 9:00 到 18:00，以及周六周日的 10:00 到 16:00。

`1-5,09:00-18:00;6-7,10:00-16:00`

10 命令执行

10 Command execution Zabbix uses common functionality for external checks, user parameters, system.run items, custom alert scripts, remote commands and user scripts. Zabbix 用常规功外部检查、用户参数、system.run 监控项、自定义告警脚本、远程指令和用户指令。

Execution steps 执行步骤

The command/script is executed similarly on both Unix and Windows platforms: 在 Unix 和 Windows 系统平台上，指令/脚本的执行方式相近

1. Zabbix (the parent process) creates a pipe for communication Zabbix (父进程) 创建了一个交流通道。
2. Zabbix sets the pipe as the output for the to-be-created child process Zabbix 将通道设置为将要被创建的子进程的输出接口
3. Zabbix creates the child process (runs the command/script) Zabbix 创建子进程（运行指令/脚本）
4. A new process group (in Unix) or a job (in Windows) is created for the child process 为子进程创建一个新的进程组 (Unix 平台) 或一个作业 (Windows 平台)
5. Zabbix reads from the pipe until timeout occurs or no one is writing to the other end (ALL handles/file descriptors have been closed). Note that the child process can create more processes and exit before they exit or close the handle/file descriptor. Zabbix 从管道读取，直到发生超时或没有其他写入另一端（所有处理/文件描述符都已关闭）。请注意，子进程可以在退出或关闭处理/文件描述符之前创建更多进程并退出。
6. If the timeout has not been reached, Zabbix waits until the initial child process exits or timeout occurs 如果尚未达到超时，Zabbix 将等待，直到初始子进程退出或发生超时
7. If the initial child process exited and the timeout has not been reached, Zabbix checks exit code of the initial child process and compares it to 0 (non-zero value is considered as execution failure, only for custom alert scripts, remote commands and user scripts executed on Zabbix server and Zabbix proxy) 如果退出初始子进程并且尚未达到超时，Zabbix 将检查初始子进程的退出代码并将其与 0 进行比较（非零值被视为执行失败，仅适用于在 Zabbix server 和 Zabbix proxy 上执行自定义告警脚本，远程命令和用户脚本）
8. At this point it is assumed that everything is done and the whole process tree (i.e. the process group or the job) is terminated 此时，假设一切都已完成，整个过程 tree（即过程组或作业）终止

Attention:
Zabbix assumes that a command/script has done processing when the initial child process has exited AND no other process is still keeping the output handle/file descriptor open. When processing is done, ALL created processes are terminated. Zabbix 假定命令/脚本在初始子进程退出时已完成处理，并且没有其他进程仍保持输出处理/文件描述符处于打开状态。处理完成后，将终止所有创建的进程

All double quotes and backslashes in the command are escaped with backslashes and the command is enclosed in double quotes.
命令中的所有双引号和反斜杠都使用反斜杠进行转义，命令用双引号括起来。

Exit code checking 退出代码的检查

Exit code are checked with the following conditions: 使用以下条件检查退出代码：

```
*Only for custom alert scripts, remote commands and user scripts executed on Zabbix server and Zabbix prox
*Any exit code that is different from 0 is considered as execution failure.任何不同于0的退出代码都被视为执
*Contents of standard error and standard output for failed executions are collected and available in front
*Additional log entry is created for remote commands on Zabbix server to save script execution output and
```

Possible frontend messages and log entries for failed commands/scripts: 可能出现的失败指令/脚本的前端信息和日志条目：

- Contents of standard error and standard output for failed executions (if any). 执行失败的标准错误和标准输出的内容（如果有的话）
- "Process exited with code: N." (for empty output, and exit code not equal to 0). " 进程退出代码：N."（对于空输出，退出代码不等于 0）。
- "Process killed by signal: N." (for process terminated by a signal, on Linux only). " 进程被信号杀死：N."（对于由信号终止的进程，仅在 Linux 上）。
- "Process terminated unexpectedly." (for process terminated for unknown reasons). " 进程意外终止。"（由于未知原因终止进程）。

Read more about 了解更多:

- External checks
- User parameters
- system.run items
- Custom alert scripts
- Remote commands
- Global scripts

11 监控方案

11 Recipes for monitoring General 概括

Monitoring server availability 监控服务器可用性

At least three methods (or combination of all methods) may be used in order to monitor availability of a server. 可以使用至少三种方法（或所有方法的组合）来监视服务器的可用性。

- ICMP ping ("icmpping" key)
- "zabbix[host,agent,available]" item
- trigger function nodata() for monitoring the availability of hosts that use active checks only 触发函数 nodata() 以监控只进行主动性检查的主机的有效性

Sending alerts via WinPopUps 通过 WinPopUps 发送警告

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.□□□□□□□Windows□□□□□□□□Zabbix□□□□□WinPopUps□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□WinPopUps□□□□□□□□ http://www.zabbix.com/forum/showthread.php?t=2147.

Monitoring specific applications 监控特定的应用程序

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. 使用 SNMP 可以监视 IBM AS/400 平台，更多信息详见 http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg244504.html?Open.

MySQL

Several user parameters can be used for the monitoring of MySQL in the agent configuration file: /usr/local/etc/zabbix_agentd.conf 在 agent 配置文件夹/usr/local/etc/zabbix_agentd.conf 中，可以用若干用户参数来监控 MySQL

```
### Set of parameters for monitoring MySQL server (v3.23.42 and later)
### Change -u and add -p if required
#UserParameter=mysql.ping,mysqladmin -uroot  ping|grep alive|wc -l
#UserParameter=mysql.uptime,mysqladmin -uroot status|cut -f2 -d":"|cut -f2 -d" "
#UserParameter=mysql.threads,mysqladmin -uroot status|cut -f3 -d":"|cut -f2 -d" "
#UserParameter=mysql.questions,mysqladmin -uroot status|cut -f4 -d":"|cut -f2 -d" "
#UserParameter=mysql.slowqueries,mysqladmin -uroot status|cut -f5 -d":"|cut -f2 -d" "
#UserParameter=mysql.qps,mysqladmin -uroot status|cut -f9 -d":"|cut -f2 -d" "
#UserParameter=mysql.version,mysql -V
```

- *mysql.ping*

Check whether MySQL is alive. 检查 MySQL 是否运行正常.

```
Result: 0 - not started 1 - alive
```

- *mysql.uptime*

Number of seconds MySQL is running.MySQL 运行的秒数.

- *mysql.threads*

Number of MySQL threads.MySQL 的线程数量.

- *mysql.questions*

Number of processed queries.MySQL 的线程数量

- *mysql.slowqueries*

Number of slow queries. 处理查询数量

- *mysql.qps*

Queries per second. 慢查询数量

- *mysql.version*

Version of MySQL. For example: mysql Ver 14.14 Distrib 5.1.53, for pc-linux-gnu (i686) MySQL 的版本. 例如: mysql 14.14 版本 Distrib 5.1.53, for pc-linux-gnu (i686)

For additional information see also the userparameter_mysql.conf file in conf/zabbix_agentd directory. 获取更过信息，请访问 conf/zabbix_agentd 目录下的 userparameter_mysql.conf 文件

Mikrotik routers Mikrotik 路由器

Use SNMP agent provided by Mikrotik. See http://www.mikrotik.com for more information.

Windows

Use Zabbix Windows agent included (pre-compiled) into Zabbix distribution. 使用包含 (预编译) 到 Zabbix 发行中的 Zabbix Windows agent.

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. 在定义一个用户参数时，可以使用 Tuxedo Command Line 实用程序 tmadmin 和 qmadmin，以返回每个服务器/服务/队列性能计数器、Tuxedo 资源的可用性。

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. 用标准英孚美的 utility onstat，几乎可以监控 Informix 数据库的各个方面。而且，Zabbix 可以检索由 Informix SNMP agent 提供的信息。

HP OpenView

Zabbix can be configured to send messages to OpenView server. The following steps must be performed: 通过配置 Zabbix 来向 OpenView 服务器发送消息，必须遵循以下几个步骤：

Step 1 步骤 1

Define new media. 定义新 media.

The media will execute a script which will send required information to OpenView.media 将执行一个向 OpenView 发送所需信息的脚本。

Step 2 步骤 2

Define new user. 定义新用户

The user has to be linked with the media. 用户必须与 media 相连接.

Step 3 步骤 3

Configure actions. 配置操作。

Configure actions to send all (or selected) trigger status changes to the user. 配置操作会将所有 (或已选择的) 状态已修改的触发器发给用户。

Step 4 步骤 4

Write media script. 编写 media 脚本

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. 脚本将有如下逻辑操作：如果触发器为 ON, 那么执行 OpenView 命令 *opcmsg -id application=<application> msg_grp=<msg_grp> object=<object> msg_text=<text>*. 该指令将返回唯一的信息 ID，该 ID 必须存储在某处，并在 ZABBIX 数据库的新列表里处于优先位置。如果触发器为 OFF，那么 *opcmack <message id>* 要和从数据库中检索的信息 ID 一同执行。

Refer to OpenView official documentation for more details about opcmsg and opcmack. The media script is not given here. 更多关于 opcmsg 和 opcmack 的详情，请见 OpenView 官方文件。这里没有给出 media 脚本。

12 Performance tuning

Attention:
This is a work in progress.

Overview

It is very important to have Zabbix system properly tuned for maximum performance.

Hardware

General advice on hardware:

- Use fastest processor available
- SCSI or SAS is better than IDE (performance of IDE disks may be significantly improved by using utility hdparm) and SATA
- 15K RPM is better than 10K RPM which is better than 7200 RPM
- Use fast RAID storage
- Use fast Ethernet adapter
- Having more memory is always better

Operating system

- Use latest (stable!) version of OS
- Exclude unnecessary functionality from kernel
- Tune kernel parameters

Zabbix configuration parameters

Many parameters may be tuned to get optimal performance.

zabbix_server

StartPollers

General rule - keep value of this parameter as low as possible. Every additional instance of zabbix_server adds known overhead, in the same time, parallelism is increased. Optimal number of instances is achieved when queue, on average, contains minimum number of parameters (ideally, 0 at any given moment). This value can be monitored by using internal check zabbix[queue].

Note:
See the "See also" section at the bottom of this page to find out how to configure optimal count of zabbix processes.

DebugLevel

Optimal value is 3.

DBSocket

MySQL only. It is recommended to use DBSocket for connection to the database. That is the fastest and the most secure way.

Database engine

This is probably the most important part of Zabbix tuning. Zabbix heavily depends on the availability and performance of database engine.

- use fastest database engine, i.e. MySQL or PostgreSQL
- use stable release of a database engine
- rebuild MySQL or PostgreSQL from sources to get maximum performance
- follow performance tuning instructions taken from MySQL or PostgreSQL documentation
- for MySQL, use InnoDB table structure
- ZABBIX works at least 1.5 times faster (comparing to MyISAM) if InnoDB is used. This is because of increased parallelism. However, InnoDB requires more CPU power.
- tuning the database server for the best performance is highly recommended.
- keep database tables on different hard disks
- 'history', 'history_str, 'items' 'functions', triggers', and 'trends' are most heavily used tables.
- for large installations keeping MySQL temporary files in tmpfs is:
  - MySQL >= 5.5: not recommended (MySQL bug #58421)
  - MySQL < 5.5: recommended

GUI debugging

Problems related to the frontend performance may be diagnosed using the frontend debug mode.

General advice

- monitor required parameters only
- tune 'Update interval' for all items. Keeping a small update interval may be good for nice graphs, however, this may overload Zabbix
- tune parameters for default templates
- tune housekeeping parameters
- do not monitor parameters which return the same information.
- avoid the use of triggers with long period given as function argument. For example, max(3600) will be calculated significantly slower than max(60).

Viewing Zabbix process performance with "ps" and "top"

Since Zabbix 2.2 processes change their commandlines to display current activity and meaningful statistics, like:

```
UID        PID  PPID  C STIME TTY      TIME CMD
zabbix22  4584     1  0 14:55 ?    00:00:00 zabbix_server -c /home/zabbix22/zabbix_server.conf
zabbix22  4587  4584  0 14:55 ?    00:00:00 zabbix_server: configuration syncer [synced configuration in 0
zabbix22  4588  4584  0 14:55 ?    00:00:00 zabbix_server: db watchdog [synced alerts config in 0.018748 s
zabbix22  4608  4584  0 14:55 ?    00:00:00 zabbix_server: timer #1 [processed 3 triggers, 0 events in 0.0
zabbix22  4609  4584  0 14:55 ?    00:00:00 zabbix_server: timer #2 [processed 2 triggers, 0 events in 0.0
zabbix22  4637  4584  0 14:55 ?    00:00:01 zabbix_server: history syncer #4 [synced 35 items in 0.166198
zabbix22  4657  4584  0 14:55 ?    00:00:00 zabbix_server: vmware collector #1 [updated 0, removed 0 VMwar
zabbix22  4670     1  0 14:55 ?    00:00:00 zabbix_proxy -c /home/zabbix22/zabbix_proxy.conf
zabbix22  4673  4670  0 14:55 ?    00:00:00 zabbix_proxy: configuration syncer [synced config 15251 bytes
zabbix22  4674  4670  0 14:55 ?    00:00:00 zabbix_proxy: heartbeat sender [sending heartbeat message succ
zabbix22  4688  4670  0 14:55 ?    00:00:00 zabbix_proxy: icmp pinger #1 [got 1 values in 1.811128 sec, id
zabbix22  4690  4670  0 14:55 ?    00:00:00 zabbix_proxy: housekeeper [deleted 9870 records in 0.233491 se
zabbix22  4701  4670  0 14:55 ?    00:00:08 zabbix_proxy: http poller #2 [got 1 values in 0.024105 sec, id
zabbix22  4707  4670  0 14:55 ?    00:00:00 zabbix_proxy: history syncer #4 [synced 22 items in 0.008565 s
zabbix22  4738     1  0 14:55 ?    00:00:00 zabbix_agentd -c /home/zabbix22/zabbix_agentd.conf
zabbix22  4739  4738  0 14:55 ?    00:00:00 zabbix_agentd: collector [idle 1 sec]
zabbix22  4740  4738  0 14:55 ?    00:00:00 zabbix_agentd: listener #1 [waiting for connection]
zabbix22  4741  4738  0 14:55 ?    00:00:00 zabbix_agentd: listener #2 [processing request]
```

The main process is an exception. Instead of current activity the original commandline is shown. This helps to distinguish processes on systems with multiple Zabbix instances.

This feature is not implemented for Microsoft Windows.

If logging level is set to **DebugLevel=4** these activity and statistics messages are also written into log file.

Linux

On Linux systems `ps` command can be used together with `watch` command for observing how Zabbix is doing. For example, to run `ps` command 5 times per second to see process activities:

```
watch -n 0.2 ps -fu zabbix
```

To show only Zabbix proxy and agent processes:

```
watch -tn 0.2 'ps -f -C zabbix_proxy -C zabbix_agentd'
```

To show only history syncer processes:

```
watch -tn 0.2 'ps -fC zabbix_server | grep history'
```

The `ps` command produces a wide output (approximately 190 columns) as some activity messages are long. If your terminal has less than 190 columns of text you can try

```
watch -tn 0.2 'ps -o cmd -C zabbix_server -C zabbix_proxy -C zabbix_agentd'
```

to display only commandlines without UID, PID, start time etc.

`top` command also can be used for observing Zabbix performance. Pressing 'c' key in `top` shows processes with their command-lines. In our tests on Linux `top` and `atop` correctly displayed changing activities of Zabbix processes, but `htop` was not displaying changing activities.

BSD systems

If `watch` command is not installed, a similar effect can be achieved with

```
while [ 1 ]; do ps x; sleep 0.2; clear; done
```

AIX, HP-UX

If `watch` command is not available, one can try

```
while [ 1 ]; do ps -fu zabbix; sleep 1; clear; done
```

Solaris

By default the `ps` command does not show changing activities. One option is to use `/usr/ucb/ps` instead. If `watch` command is not installed, a periodically updated list of processes can be shown with

```
while [ 1 ]; do /usr/ucb/ps gxww; sleep 1; clear; done
```

On Solaris 11:

- `/usr/ucb/ps` is not installed by default. You may need to install *ucb* package, e.g. `pkg install compatibility/ucb`,
- if Zabbix daemon has been started by privileged user its activities are not shown to non-privileged user.
- the `sleep` command accepts not only whole seconds but also fractions of second (e.g. `sleep 0.2`).

See also

1. How to configure optimal count of zabbix processes

13 Version compatibility

Supported agents

Zabbix agents starting with version 1.4 are compatible with Zabbix 4.0. However, you may need to review the configuration of older agents as some parameters have changed, for example, parameters related to logging for versions before 3.0.

To take full advantage of new and improved items, improved performance and reduced memory usage, use the latest 4.0 agent.

Supported Zabbix proxies

Both Zabbix 4.0 proxies and Zabbix 4.0 server are supported to work only with Zabbix 4.0 server and Zabbix 4.0 proxies respectively.

Attention:
It is known to be possible to start an upgraded server and have older, yet unupgraded proxies report data to a newer server (the proxies can't refresh their configuration though). This approach, however, is not recommended and not supported by Zabbix and choosing it is entirely at your own risk. For more information, see the upgrade procedure.

Supported XML files

XML files, exported with 1.8, 2.0, 2.2, 2.4, 3.0, 3.2 and 3.4 are supported for import in Zabbix 4.0.

Attention:
In Zabbix 1.8 XML export format, trigger dependencies are stored by name only. If there are several triggers with the same name (for example, having different severities and expressions) that have a dependency defined between them, it is not possible to import them. Such dependencies must be manually removed from the XML file and re-added after import.

14 Database error handling

If Zabbix detects that the backend database is not accessible, it will send a notification message and continue the attempts to connect to the database. For some database engines, specific error codes are recognised.

MySQL

- CR_CONN_HOST_ERROR
- CR_SERVER_GONE_ERROR
- CR_CONNECTION_ERROR
- CR_SERVER_LOST
- CR_UNKNOWN_HOST
- ER_SERVER_SHUTDOWN
- ER_ACCESS_DENIED_ERROR
- ER_ILLEGAL_GRANT_FOR_TABLE
- ER_TABLEACCESS_DENIED_ERROR
- ER_UNKNOWN_ERROR

15 Zabbix sender dynamic link library for Windows

In a Windows environment applications can send data to Zabbix server/proxy directly by using the Zabbix sender dynamic link library (zabbix_sender.dll) instead of having to launch an external process (zabbix_sender.exe).

The dynamic link library with the development files is located in bin\winXX\dev folders. To use it, include the zabbix_sender.h header file and link with the zabbix_sender.lib library. An example file with Zabbix sender API usage can be found in build\win32\examples\zabbix_sender folder.

The following functionality is provided by the Zabbix sender dynamic link library:

```
int zabbix_sender_send_values(const char *address, unsigned short port,const char *source, const zabbix_
char **result);‘{.c}
```

The following data structures are used by the Zabbix sender dynamic link library:

```
typedef struct
{
    /* host name, must match the name of target host in Zabbix */
    char    *host;
    /* the item key */
    char    *key;
    /* the item value */
    char    *value;
}
zabbix_sender_value_t;
```

```
typedef struct
{
    /* number of total values processed */
    int total;
    /* number of failed values */
    int failed;
    /* time in seconds the server spent processing the sent values */
    double  time_spent;
}
zabbix_sender_info_t;
```

16 Issues with SELinux

Socket-based inter-process communication has been added since Zabbix 3.4. On systems where SELinux is enabled it may be required to add SELinux rules to allow Zabbix create/use UNIX domain sockets in the `SocketDir` directory. Currently socket files are used by server (alerter, preprocessing, IPMI) and proxy (IPMI). Socket files are persistent, meaning are present while the process is running.

1. 简介

请使用侧边栏导航来访问此章节中的内容。

1 手册结构

Structure

Zabbix 4.0 的手册内容分为几个章节和子章节，以便于您来访问感兴趣的特定主题。

当您导航到相应的章节时，请确保您展开该章节的被折叠页面，从而完整获取各个子章节和页面中的内容。

手册会将尽可能提供相关内容页面之间的交叉链接，以确保用户不会错过相关信息。

章节

简介 提供了关于 Zabbix 的常规信息。阅读本章节应该会为您选择 Zabbix 提供一些好的理由。

术语 解释了在 Zabbix 中使用到的术语，并提供了有关 Zabbix 组件的详细信息。

安装 和快速入门 章节可以帮助您开始使用 Zabbix。

Zabbix 应用 是一种可供选择的方案，通过此章节可以了解快速使用 Zabbix 的方法。

配置 是本手册中篇幅最多并且最为重要的章节之一。它包含了大量关于如何设置 Zabbix 去监控您的环境的基本建议，从设置主机到获取基本数据，再到查看数据，再到配置通知，以及问题拍错的相关命令。

IT services 章节详细说明了如何使用 Zabbix ，从更高层次的视角关注您的监控系统。

Web 监控 可以帮助您学会如何监控 Web 网站的可用性。

虚拟机监控 介绍了如何配置 VMware 环境的监控。

维护, 正则表达式, 事件确认 和配置的导入与导出 几个章节进一步展示了如何使用 Zabbix 的这些方面的功能。

自动发现 包含有关配置网络设备、Zabbix 客户端 (主动式)、文件系统、网络接口等的自动发现的说明。

分布式监控 使用 Zabbix 支撑更庞大和更复杂环境的相关内容。

加密 解释了如何对 Zabbix 组件之间的通讯进行加密。

Web 界面 包含了如何使用 Zabbix 的 Web 界面的内容。

API 介绍了使用 Zabbix API 的详细信息。

更为详细的技术细节，包含在附录 中。附录也包含常见问题的详细解答。

2 Zabbix 介绍

概述

Zabbix 是由 Alexei Vladishev 创建，目前是由 Zabbix SIA 在持续开发和提供支持。

Zabbix 是一种企业级的分布式开源监控解决方案。

Zabbix 是一款能够监控众多网络参数和服务器的健康度和完整性的软件。Zabbix 使用灵活的通知机制，允许用户为几乎任何事件配置基于邮件的警报。这样可以快速相应服务器问题。Zabbix 基于存储的数据提供出色的报告和数据可视化。这些功能使得 Zabbix 成为容量规划的理想选择。

Zabbix 支持轮询和被动捕获。所有的 Zabbix 报告、统计信息和配置参数都可以通过基于 Web 的前端页面进行访问。基于 Web 的前端页面确保您的网络状态和服务器健康状况可以从任何地方进行评估。在经过适当的配置后，Zabbix 可以在监控 IT 基础设施方面发挥重要作用。无论是对于拥有少量服务器的小型组织，还是拥有大量服务器的大型公司而言，同样适用。

Zabbix 是免费的。Zabbix 是根据 GPL 通用公共许可证的第二版编写和分发的。这意味着它的源代码是免费分发的，并且可供公共使用。

商业支持 由 Zabbix 公司提供。

了解更多Zabbix 功能。

Zabbix 的用户

世界上许多不同规模的组织都依赖 Zabbix 作为主要的监控平台。

3 Zabbix 功能

概述

Zabbix 是一种高度集成的网络监控解决方案，在单一的软件包中提供了多种功能。

数据采集

- 可用性和性能采集；
- 支持 SNMP（包括主动轮询和被动捕获）、IPMI、JMX、VMware 监控；
- 自定义检查；
- 按照自定义的时间间隔采集需要的数据；
- 通过 Server/Proxy 和 Agents 来执行数据采集。

灵活的阈值定义

- 您可以定义非常灵活的告警阈值，称之为触发器，触发器从后端数据库获得参考值。

高度可配置化的告警

- 可以根据递增计划、接收者、媒介类型自定义发送告警通知；
- 使用宏变量可以使告警通知变得更加高效有益；
- 自动动作包含远程命令。

实时图形

- 使用内置图形功能可实以将监控项绘制成图形。

Web 监控功能

- Zabbix 可以追踪模拟鼠标在 Web 网站上的点击操作，来检查 Web 网站的功能和响应时间。

丰富的可视化选项

- 能够创建可以将多个监控项组合到单个视图中的自定义图形；
- 网络拓扑图；
- 以仪表盘样式展示自定义聚合图形和幻灯片演示；
- 报表；
- 监控资源的高层次（业务）视图。

历史数据存储

- 存储在数据库中的数据；
- 可配置的历史数据；
- 内置数据管理机制（housekeeping）。

配置简单

- 将被监控设备添加为主机；
- 主机一旦添加到数据库中，就会采集主机数据用于监控；
- 将模板用于监控设备。

套用模板

- 在模板中分组检查；
- 模板可以关联其他模板，获得继承。

网络发现

- 自动发现网络设备；
- Zabbix Agent 发现设备后自动注册；
- 自动发现文件系统、网络接口和 SNMP OIDs 值。

快捷的 **Web** 界面

- 基于 PHP 的 Web 前端；
- 可以从任何地方访问；
- 您可以定制自己的操作方式；
- 审计日志。

Zabbix API

- Zabbix API 为 Zabbix 提供可编程接口，用于批量操作、第三方软件集成和其他用途。

权限管理系统

- 安全的用户身份验证；
- 将特定用户限制于访问特定的视图。

功能强大且易于扩展的 **Zabbix Agent**

- 部署于被监控对象上；
- 完美支持 Linux 和 Windows ；

二进制守护进程

- 为了更好的性能和更少的内存占用，采用 C 语言编写；
- 便于移植。

适应更复杂的环境

- 使用 Zabbix Proxy 代理，可以轻松实现分布式远程监控。

4 Zabbix 概述

架构

Zabbix 由几个主要的功能组件组成，其职责如下所示。

Server

Zabbix server 是 Zabbix agent 向其报告可用性、系统完整性信息和统计信息的核心组件。是存储所有配置信息、统计信息和操作信息的核心存储库。

数据库

所有配置信息以及 Zabbix 收集到的数据都被存储在数据库中。

Web 界面

为了从任何地方和任何平台轻松访问 Zabbix ，我们提供了基于 web 的界面。该界面是 Zabbix server 的一部分，通常（但不一定）和 Zabbix server 运行在同一台物理机器上。

Proxy

Zabbix proxy 可以替 Zabbix server 收集性能和可用性数据。Zabbix proxy 是 Zabbix 环境部署的可选部分；然而，它对于单个 Zabbix server 负载的分担是非常有益的。

Agent

Zabbix agents 部署在被监控目标上，用于主动监控本地资源和应用程序，并将收集的数据发送给 Zabbix server。

数据流

此外，重要的是，需要回过头来了解下 Zabbix 内部的整体数据流。首先，为了创建一个采集数据的监控项，您就必须先创建主机。其次，必须有一个监控项来创建触发器。最后，您必须有一个触发器来创建一个动作，这几个点构成了一个完整的数据流。因此，如果您想要收到 CPU load it too high on *Server X* 的告警，您必须首先为 //Server X // 创建一个主机条目，其次创建一个用于监视其 CPU 的监控项，最后创建一个触发器，用来触发 CPU is too high 这个动作，并将其发送到您的邮箱里。虽然这些步骤看起来很繁琐，但是使用模板的话，其实并不复杂。也正是由于这种设计，使得 Zabbix 的配置变得更加灵活易用。

2. 定义

概述　这部分统一解释，一些 Zabbix 常用术语的含义。

D 定义　主机（***host***）

- 你想要监控的联网设备，有 *IP/DNS*。

主机组（***host group)***

- 主机的逻辑组；可能包含主机和模板。一个主机组里的主机和模板之间并没有任何直接的关联。通常在给不同用户组的主机分配权限时候使用主机组。

监控项（***item***）

- 你想要接收的主机的特定数据，一个度量/指标数据。

值预处理（***value preprocessing***）

- 转化/预处理接收到的指标数据存入数据库之前。

触发器（***trigger***）

- 一个被用于定义问题阈值和 “评估” 监控项接收到的数据的逻辑表达式

当接收到的数据高于阈值时，触发器从”OK” 变成”Problem” 状态。当接收到的数据低于阈值时，触发器保留/返回”OK” 的状态。

事件（***event***）

- 一次发生的需要注意的事情，例如触发器状态改变、发现/监控代理自动注册

事件标签（***event tag***）

- 提前设置的事件标记可以被用于事件关联，权限细化设置等。

事件关联（***event correlation***）

- 自动灵活的、精确的关联问题和解决方案

比如说，你可以定义触发器 A 告警的异常可以由触发器 B 解决，触发器 B 可能采用完全不同的数据采集方式。

异常（***problems***） - 一个处在” 异常 ” 状态的触发器

异常更新（***problem update***）

- *Zabbix* 提供的问题管理选项，例如添加评论、确认异常、改变问题级别或者手动关闭等。

动作（***action***）

- 预先定义的应对事件的操作

一个动作由操作 (例如发出通知) 和条件 (什么时间进行操作) 组成

升级（***escalation***）

- 一个在动作内执行操作的自定义方式; 发送通知/执行远程命令的顺序安排。

媒介（***media***）

- 发送告警通知的方式；传送途径

通知（***notification***）

- 关于事件的信心，将通过选设定的媒介途径发送给用户。

远程命令（***remote command***）

- 一个预定义好的，满足特定条件的情况下，可以在被监控主机上自动执行的命令。

模版（***template***）

- 一组可以被应用到一个或多个主机上的实体（监控项，触发器，图形，聚合图形，应用，*LLD*，*Web* 场景）的集合

模版的应用使得主机上的监控任务部署快捷方便；也可以使监控任务的批量修改更加简单。模版是直接关联到每台单独的主机上。

应用（***application***）

- 一组监控项组成的逻辑分组

***Web* 场景（*web scenario*）**

- 检查网站可浏览性的一个或多个 *HTTP* 请求

前端（***frontend)***

- *Zabbix* 提供的 *web* 界面

Zabbix API

- *Zabbix API* 允许用户使用 *JSON RPC* 协议来创建、更新和获取 *Zabbix* 对象（如主机、监控项、图形和其他）信息或者执行任何其他的自定义的任务

Zabbix server

- *Zabbix* 监控的核心程序，主要功能是与 *Zabbix proxies* 和 *Agents* 进行交互、触发器计算、发送告警通知；并将数据集中保存等

Zabbix agent

- 部署在监控对象上的，能够主动监控本地资源和应用的程序

Zabbix proxy

- 一个帮助 *Zabbix Server* 收集数据，分担 *Zabbix Server* 的负载的程序

加密（***encryption***）

- 支持 *Zabbix* 组建之间的加密通讯 *(server, proxy, agent, zabbix_sender* 和 *zabbix_get* 程序*)* 使用 TLS（Transport Layer Security ）协议。

3. 进程

请使用侧边栏导航来访问此章节中的内容。

1 Server

概述

Zabbix server 是整个 Zabbix 软件的核心程序。

Zabbix Server 负责执行数据的主动轮询和被动获取，计算触发器条件，向用户发送通知。它是 Zabbix Agent 和 Proxy 报告系统可用性和完整性数据的核心组件。Server 自身可以通过简单服务远程检查网络服务（如 Web 服务器和邮件服务器）。

Zabbix Server 是所有配置、统计和操作数据的中央存储中心，也是 Zabbix 监控系统的告警中心。在监控的系统中出现任何异常，将被发出通知给管理员。

基本的 Zabbix Server 的功能分解成为三个不同的组件。他们是：Zabbix server、Web 前端和数据库。

Zabbix 的所有配置信息都存储在 Server 和 Web 前端进行交互的数据库中。例如，当你通过 Web 前端（或者 API）新增一个监控项时，它会被添加到数据库的监控项表里。然后，Zabbix server 以每分钟一次的频率查询监控项表中的有效项，接着将它存储在 Zabbix server 中的缓存里。这就是为什么 Zabbix 前端所做的任何更改需要花费两分钟左右才能显示在最新的数据段的原因。

服务进程

通过二进制包安装的组件

Zabbix server 进程以守护进程（Deamon）运行。Zabbix server 的启动可以通过执行以下命令来完成：

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

上述命令在大多数的 GNU/Linux 系统下都可以正常完成。如果是其他系统，你可能要尝试以下命令来运行：

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

类似的，停止、重启、查看状态，则需要执行以下命令：

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

手动启动

如果以上操作均无效，您可能需要手动启动，找到 Zabbix Server 二进制文件的路径并且执行：

```
shell> zabbix_server
```

您可以将以下命令行参数用于 Zabbix server：

```
-c --config <file>              配置文件路径（默认的是 /usr/local/etc/zabbix_server.conf）
-R --runtime-control <option>   执行管理功能
-h --help                       帮助
-V --version                    显示版本号
```

Note:
运行时控制不支持 OpenBSD 和 NetBSD 系统。

使用命令行参数运行 Zabbix server 的示例: :

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

运行时控制

运行时控制包含的选项：

选项描	目标
config_cache_reload	重新加载配置缓存。如果当前正在加载缓存，则忽略。
housekeeper_execute	启动管家程序。忽略当前正在进行中的管家程序。

选项描	目标	
log_level_increase[=<**target**>]	增加日志级别，如果未指定目标，将影响所有进程。**pid** - 进程标识符 (1 to	5535) **process type** - 指定进程的所有类型 (例如，poller) **process type,N** - 进程类型和编号 (例如，poller,3)
log_level_decrease[=<**target**>]	降低日志级别，如果未指定目标，则会影响所有进程。:::	

单一 Zabbix 进程的日志级别改变后，进程的 PIDs 的值也会改变，允许的范围为 1~65535。在具有大 PIDs <process type,N> 目标选项可更改单个进程的日志级别。

例如，使用 config_cache_reload 选项重新加载 server 的配置缓存：

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

例如，使用 housekeeper_execute 选项来触发管家服务执行：

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

例如，使用 log_level_increase 选项来改变日志级别：

增加所有进程的日志级别：

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

增加第二个 Poller 进程的日志级别：

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=poller,2
```

增加 PID 为 1234 进程的日志级别：

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=1234
```

降低 http poller 进程的日志级别：

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_decrease="http poller"
```

进程用户

Zabbix server 允许使用非 root 用户运行。它将以任何非 root 用户的身份运行。因此，使用非 root 用户运行 server 是没有任何问题的.

如果你试图以 “root” 身份运行它，它将会切换到一个已经 “写死” 的 “zabbix” 用户，您可以参考安装 章节。按此相应地修改 Zabbix server 配置文件中的 “AllowRoot” 参数，则可以只以 “root” 身份运行 Zabbix server。

如果 Zabbix server 和agent 均运行在同一台服务器上，建议您使用不同的用户运行 server 和 agent 。否则，，如果两者都以相同的用户运行，Agent 可以访问 Server 的配置文件, 任何 Zabbix 管理员级别的用户都可以很容易地检索到 Server 的信息。例如，数据库密码。

配置文件

有关配置 Zabbix server 的详细信息，请查阅配置文件 章节。

启动脚本

这些脚本用于在系统启动和关闭期间自动启动和停止 Zabbix 进程。此脚本位于 misc/init.d 目录下。

支持的平台

由于服务器操作的安全性要求和任务关键性，UNIX 是唯一能够始终如一地提供必要性能、容错和弹性的操作系统。Zabbix 以市场主流的操作系统版本运行。

经测试，Zabbix 可以运行在下列平台：

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

> **Note:**
> Zabbix 可以运行在其他类 Unix 操作系统上。

语言环境

值得注意的是，Zabbix server 需要 UTF-8 语言环境，以便可以正确解释某些文本项。大多数现代类 Unix 系统都默认使用 UTF-8 语言环境，但是，有些系统可能需要做特定的设置。

2 Agent

概述

Zabbix agent 部署在被监控目标上，以主动监控本地资源和应用程序（硬盘、内存、处理器统计信息等）。

Zabbix agent 收集本地的操作信息并将数据报告给 Zabbix server 用于进一步处理。一旦出现异常 (例如硬盘空间已满或者有崩溃的服务进程)，Zabbix server 会主动警告管理员指定机器上的异常。

Zabbix agents 的极高效率缘于它可以利用本地系统调用来完成统计数据的采集。

被动和主动检查

Zabbix agent 可以运行被动检查和主动检查。

在被动检查 模式中 agent 应答数据请求。Zabbix server（或 proxy）询求数据，例如 CPU load，然后 Zabbix agent 返还结果。

主动检查 处理过程将相对复杂。Agent 必须首先从 Zabbix sever 索取监控项列表以进行独立处理，然后会定期发送采集到的新值给 Zabbix server。

是否执行被动或主动检查是通过选择相应的监控项类型来配置的。Zabbix agent 处理 “Zabbix agent” 或 “Zabbix agent（active）” 类型的监控项。

支持的平台

Zabbix agent 支持以下平台：

- Linux

- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris: 9, 10, 11
- Windows：支持从 Windows XP 之后的桌面版和服务器版。

类 UNIX 系统上的 Agent

类 UNIX 系统上的 Zabbix agent 运行在被监控的主机上。

安装

有关通过二进制包安装 Zabbix agent 的详细信息，请查阅以二进制包安装 章节。

此外，如果您不想使用二进制包，请查阅以源码包安装的说明。

> **Attention:**
> 通常，32 位 Zabbix agent 可以在 64 位系统上运行，但在某些情况下可能会失败。

通过二进制包安装的组件

Zabbix agent 进程以守护进程（Deamon）运行。Zabbix agent 的启动可以通过执行以下命令来完成：

```
shell> service zabbix-agent start
```

上述命令在大多数的 GNU/Linux 系统下都可以正常完成。如果是其他系统，你可能要尝试以下命令来运行：

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

类似的，停止、重启、查看状态，则需要执行以下命令：

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

手动启动

如果以上操作均无效，您可能需要手动启动，找到 Zabbix agent 二进制文件的路径并且执行：

```
shell> zabbix_agentd
```

Windows 系统上的 Agent

Windows 系统上的 Zabbix agent 作为一个 Windows 服务运行。

准备

Zabbix agent 作为 zip 压缩文件分发。下载该文件后，您需要将其解压缩。选择任何文件夹来存储 Zabbix 代理和配置文件，例如：

```
C:\zabbix
```

复制二进制文件 \bin\zabbix_agentd.exe 和配置文件 \conf\zabbix_agentd.conf 到 c:\zabbix 下。

按需编辑 c:\zabbix\zabbix_agentd.conf 配置文件，确保指定了正确的”Hostname” 参数。

安装

完成此操作后，使用以下命令将 Zabbix agent 安装为 Windows 服务：

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

现在您可以像任何其他 Windows 服务一样配置 “Zabbix agent” 服务。

有关在 Windows 上安装和运行 Zabbix agent 的详细信息，请查阅于此。

其他 Agent 选项

您可以在主机上运行单个或多个 Agent 实例。单个实例可以使用默认配置文件或命令行中指定的配置文件。如果是多个实例，则每个 Agent 程序实例必须具有自己的配置文件（其中一个实例可以使用默认配置文件）。

以下命令参数可以在 Zabbix agent 中使用：

参数 *	描述 **
UNIX 和 Windows agent	
-c --config <config-file>	配置文件的绝对路径。 您可以使用此选项来制定配置文件，而不是使用默认文件。 在 UNIX 上，默认的配置文件是 /usr/local/etc/zabbix_agentd.conf 或由compile-time 中的 *--sysconfdir* 或 *--prefix* 变量来确定。 在 Windows 上, 默认的配置文件是 c:\zabbix_agentd.conf
-p --print	输出已知的监控项并退出。 注意：要返回用户自定义参数 的结果，您必须指定配置文件(如果它不在默认路径下)。
-t --test <item key>	测试指定的监控项并退出。 注意：要返回用户自定义参数 的结果，您必须指定配置文件(如果它不在默认路径下)。
-h --help	显示帮助信息
-V --version	显示版本号
仅 **UNIX agent**	
-R --runtime-control <option>	执行管理功能。请参阅运行时机制的控制。
** 仅 Windows agent **	
-m --multiple-agents	使用多 Agent 实例(使用 -i、-d、-s、-x)。为了区分实例的服务名称，每项服务名都会包涵来自配置文件里的 Hostname 值。
仅 **Windows agent** （功能）	
-i --install	以服务的形式安装 Zabbix Windows agent。
-d --uninstall	卸载 Zabbix indows agent 服务。
-s --start	启动 Zabbix Windows agent 服务。
-x --stop	停止 Zabbix Windows agent 服务。

使用命令行参数的具体示例：

- 打印输出所有内置监控项和它们的值。
- 使用指定的配置文件中的 “mysql.ping” 键值来测试用户自定义参数。
- 在 Windows 下使用默认路径下的配置文件 c:\zabbix_agentd.conf 安装 Zabbix agent 服务。

- 使用位于与 agent 可执行文件同一文件夹中的配置文件 zabbix_agentd.conf 为 Windows 安装 “Zabbix Agent [Hostname]” 服务，并通过从配置文件中的唯一 Hostname 值扩来为命名。

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

运行时控制

使用运行时控制选项，您可以更改代理进程的日志级别。

选项描	目标	
log_level_increase[=<target>]	增加日志级别。 目标可以被指如果未指定目标，将影响所有进程。pid - 进程标识符 (	为：to 65535) process type - 指定进程的所有类型 (例如，poller) process type,N - 进程类型和编号 (例如，poller,3)
log_level_decrease[=<target>]	降低日志级别。 ::: 如果未指定目标，将影响所有进程。	

值得注意的是，用于更改单个 Agent 进程的日志级别的 PIDs 的可用范围是 1 到 65535。在具有大 PIDs 的系统上，<process type,N> 目标可用于更改单个进程的日志级别。

例子：

- 给所有进程增加日志级别。
- 给第二个监听进程增加日志级别。
- 给 PID 号为 1234 的进程增加日志级别。
- 给所有主动检查进程降低日志级别。

```
shell> zabbix_agentd -R log_level_increase
shell> zabbix_agentd -R log_level_increase=listener,2
shell> zabbix_agentd -R log_level_increase=1234
shell> zabbix_agentd -R log_level_decrease="active checks"
```

Note:
运行时控制不支持 OpenBSD 和 NetBSD 和 Windows 系统。

进程用户

Zabbix agent 在 UNIX 上允许使用非 root 用户运行。它将以任何非 root 用户的身份运行。因此，使用非 root 用户运行 agent 是没有任何问题的.

如果你试图以 “root” 身份运行它，它将会切换到一个已经 “写死” 的 “zabbix” 用户，该用户必须存在于您的系统上。如果您只想以 “root” 用户运行 agent，您必须在 agent 配置文件里修改 ‘AllowRoot‘参数。

配置文件

有关配置 Zabbix agent 的详细信息，请查阅zabbix_agentd 或Windows agent 章节。

语言环境

值得注意的是，Zabbix agent 需要 UTF-8 语言环境，以便某些文本 Zabbix agent 监控项可以返回预期的内容。大多数现代类 Unix 系统都默认使用 UTF-8 语言环境，但是，有些系统可能需要特定的设置。

退出码

在 2.2 版之前，Zabbix agent 在成功退出时返回 0，在异常时返回 255。从版本 2.2 及更高版本开始，Zabbix agent 在成功退出时返回 0，在异常时返回 1。

3 Proxy

=== 概述 ===

Zabbix proxy 是一个可以从一个或多个受监控设备采集监控数据并将信息发送到 Zabbix server 的进程，主要是代表 Zabbix server 工作。所有收集的数据都在本地缓存，然后传输到 proxy 所属的 Zabbix server。

部署 Zabbix proxy 是可选的，但可能非常有利于分担单个 Zabbix server 的负载。如果只有代理采集数据，则 Zabbix server 上会减少 CPU 和磁盘 I/O 的开销。

Zabbix proxy 是无需本地管理员即可集中监控远程位置、分支机构和网络的理想解决方案。

Zabbix proxy 需要使用独立的数据库。

Attention:
值得注意的是，Zabbix proxy 支持 SQLite、MySQL 和 PostgreSQL 作为数据库。使用 Oracle 或 DB2 需要您承担一定的风险，例如，在自动发现规则中的遇到问题返回值 。

详见：在分布式环境中使用 Zabbix proxy。

Proxy 进程

通过二进制包安装的组件

Zabbix proxy 进程以守护进程（Deamon）运行。Zabbix proxy 的启动可以通过执行以下命令来完成：

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

上述命令在大多数的 GNU/Linux 系统下都可以正常完成。如果是其他系统，你可能要尝试以下命令来运行：

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

类似的，Zabbix proxy 的停止、重启、查看状态，则需要执行以下命令：

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

手动启动

如果以上操作均无效，您可能需要手动启动，找到 Zabbix proxy 二进制文件的路径并且执行：

```
shell> zabbix_proxy
```

您可以将以下命令行参数用于 Zabbix proxy：

```
-c --config <file>              配置文件路径
-R --runtime-control <option>   执行管理功能
-h --help                       帮助
-V --version                    显示版本号
```

Note:
运行时机制的控制不支持 OpenBSD 和 NetBSD 系统。

使用命令行参数运行 Zabbix proxy 的示例: :

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

运行时控制

运行时控制包含的选项 :

选项描	目标
config_cache_reload	重新加载配置缓存。如果当前正在加载缓存，则忽略。主动模式下的 Zabbix proxy 将连接到 Zabbix server 并请求配置数据。
housekeeper_execute	启动管家程序。忽略当前正在进行中的管家程序。

选项描	目标	
log_level_increase[=<**target**>]	增加日志级别，如果未指定目标，将影响所有进程。**pid** - 进程标识符 (1 to	5535) **process type** - 指定进程的所有类型 (例如，poller) **process type,N** - 进程类型和编号 (例如，poller,3)
log_level_decrease[=<**target**>]	降低日志级别，如果未指定目标，则会影响所有进程。:::	

单一 Zabbix 进程的日志级别改变后，进程的 PIDs 的值也会改变，允许的范围为 1~65535。在具有大 PIDs <process type,N> 目标选项可更改单个进程的日志级别。

例如，使用 config_cache_reload 选项重新加载 proxy 的配置缓存：

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

例如，使用 housekeeper_execute 选项来触发管家服务执行：

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

例如，使用 log_level_increase 选项来改变日志级别：

增加所有进程的日志级别：

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

增加第二个 Poller 进程的日志级别：

```
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=poller,2
```

增加 PID 为 1234 进程的日志级别：

```
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=1234
```

降低 http poller 进程的日志级别：

```
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_decrease="http poller"
```

进程用户

Zabbix proxy 允许使用非 root 用户运行。它将以任何非 root 用户的身份运行。因此，使用非 root 用户运行 proxy 是没有任何问题的.

如果你试图以 “root” 身份运行它，它将会切换到一个已经 “写死” 的 “zabbix” 用户，该用户必须存在于您的系统上。如果您只想以 “root” 用户运行 proxy，您必须在 proxy 配置文件里修改 ‘AllowRoot‘参数。

配置文件

有关配置 Zabbix proxy 的详细信息，请查阅配置文件 章节。

支持的平台

Zabbix proxy 在与 Zabbix server 相同的server# 受支持的平台 列表上运行。

语言环境

值得注意的是，Zabbix proxy 需要 UTF-8 语言环境，以便可以正确解释某些文本项。大多数现代类 Unix 系统都默认使用 UTF-8 语言环境，但是，有些系统可能需要专门设置。

4 Java gateway

概述

从 Zabbix 2.0 开始，以 Zabbix 守护进程方式原生支持监控 JMX 应用程序就存在了，称之为 “Zabbix Java gateway”。Zabbix Java gateway 的守护进程是用 Java 编写。为了在特定主机上找到 JMX 计数器的值，Zabbix server 向 Zabbix Java gateway 发送请求，后者使用 JMX 管理 API 来远程查询相关的应用。该应用不需要安装额外的软件。只需要在启动时，命令行添加 `-Dcom.sun.management.jmxremote` 选项即可。

Java gateway 接受来自 Zabbix server 或 Zabbix proxy 的传入连接，并且只能用作 “被动 proxy”。与 Zabbix proxy 相反，它也可以从 Zabbix proxy （Zabbix proxy 不能被链接）调用。在 Zabbix server 或 Zabbix proxy 配置文件中，可以直接配置每个 Java gateway 的访问，因此每个 Zabbix pserver 或 Zabbix proxy 只能配置一个 Java gateway。如果主机将有 **JMX agent** 或其他类型的监控项，则只将 **JMX agent** 监控项传递给 Java gateway 进行检索。

当必须通过 Java gateway 更新监控项时，Zabbix server 或 proxy 将连接到 Java gateway 并请求该值，Java gateway 将检索该值并将其传递回 Zabbix server 或 Zabbix proxy。因此，Java gateway 不会缓存任何值。

Zabbix server 或 Zabbix proxy 具有连接到 Java gateway 的特定类型的进程，由 **StartJavaPollers** 选项控制。在内部，Java gateway 启动多个线程，由 **START_POLLERS** 选项控制。在服务器端，如果连接超过 **Timeout** 选项配置的秒数，它将被终止，但 Java gateway 可能仍在忙于从 JMX 计数器检索值。为了解决这个问题，从 Zabbix 2.0.15、Zabbix 2.2.10 和 Zabbix 2.4.5 开始，Java gateway 中有 **TIMEOUT** 选项，允许为 JMX 网络操作设置超时。

Zabbix server 或 proxy 尝试尽可能地将请求汇集到单个 JMX 目标（受监控项取值间隔影响），并在单个连接中将它们发送到 Java Gateway 以获得更好的性能。

此外，建议让 **StartJavaPollers** 选项的值小于或等于 **START_POLLERS**，否则可能会出现 Java gateway 中没有可用线程来为传入请求提供服务的情况。

以下部分描述了如何获取和运行 Zabbix Java gateway，如何配置 Zabbix server（或 Zabbix proxy）来使用 Zabbix Java gateway 进行 JMX 监控，以及如何在 Zabbix GUI 中配置与特定 JMX 计数器对应的 Zabbix 监控项。

1 获取 Java gateway

获取 Java gateway 有两种方法。一种是从 Zabbix 网站下载 Java gateway 二进制包，另一种是从源代码编译 Java gateway。

1.1 从 Zabbix 网站下载

Zabbix Java gateway 二进制包（RHEL, Debian, Ubuntu）可以从 http://www.zabbix.com/download.php 下载。

1.2 从源码包中编译

为了编译 Java gateway，首先使用 `--enable-java` 选项运行 `./configure` 脚本。建议您使用 `--prefix` 选项来指定其他路径，而非默认的 /usr/local 路径，因为安装 Java gateway 将创建整个目录树，并非单个可执行文件。

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

要将 Java gateway 编译并打包到 JAR 文件中，请运行 `make`。值得注意的是，对于此步骤，会使用 `javac` 和 `jar` 可执行文件，因此需要确保它们处于正确的路径下。

```
$ make
```

现在您在 src/zabbix_java/bin 路径下有 zabbix-java-gateway-$VERSION.jar 文件。如果您熟悉在 src/zabbix_java 分发目录下运行 Java gateway，那么您可以继续执行配置和运行 Java gateway 的指令。否则，请确保您有拥有足够的权限来运行 `make install`。

```
$ make install
```

2 Java gateway 分发中的文件概述

无论您如何获得的 Java gateway，在 $PREFIX/sbin/zabbix_java 路径下您都会获得一系列的 shell 脚本、JAR 和配置文件。这些文件的作用的概述如下。

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

Java gateway JAR 文件。

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

Java gateway 依赖于：Logback、SLF4J 和 Android JSON 库。

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

用于 Logback 的配置文件：

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

启动和停止 Java gateway 的便捷脚本：

```
settings.sh
```

由上面启动和停止脚本提供的配置文件。

3 配置和运行 Java gateway

默认情况下，Java gateway 监听 10052 端口。如果您计划使用不同的端口来运行 Java gateway，则可以通过 setting.sh 脚本中指定端口。有关如何指定此选项和其他选项，详见Java gateway 配置文件。

Warning:
值得注意的是，端口 10052 并没有在 IANA 注册。

待熟悉设置后，您可以通过运行 startup 脚本来启动 Java gateway：

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

同样的，一旦您不需要 Java gateway，运行 shutdown 脚本即可关闭它。

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

请注意，与 Zabbix server 或 Zabbix proxy 不同，Java gateway 是轻量级的，并不需要数据库。

4 配置 server 以使用 Java gateway

现在 Java gateway 正在运行，您必须告诉 Zabbix server 从哪里找到 Zabbix Java gateway。因此需要在Zabbix server 配置文件 中指定 JavaGateway 和 JavaGatewayPort 参数。如果 Zabbix proxy 监控运行着 JMX 应用程序的主机，则在Zabbix proxy 配置文件 中指定连接参数。

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

默认情况下，Zabibx server 不会启动与 JMX 监控相关的任何进程。但是，如果要使用它，则必须指定 Java pollers 的 pre-forked 实例数。同样的，您也可以指定常规的 pollers 和 trappers。

```
StartJavaPollers=5
```

值得注意的是，在完成配置后，请不要忘记重新启动 Zabbix server 或 Zabbix proxy。

5 Java gateway 的调试

如果 Java gateway 出现任何问题或者您看到 Zabbix 前端中的监控项错误消息不充分时，您可以查看 Java gateway 的日志文件。

默认情况下，Java gateway 的日志会记录到 /tmp/zabbix_java.log 文件中，日志级别为 “info”。有时候这些信息是不够的，需要将日志级别修改为 “dubug”。为了提高日志记录级别，需要修改 lib/logback.xml 文件并将 <root> 标记的 level 属性更改为 “debug”：

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

值得注意的是，与 Zabbix server 或 Zabbix proxy 不同，更改 logback.xml 文件后无需重新启动 Zabbix Java gateway，将自动完成 logback.xml 中的更改。完成调试后，可以将日志记录级别修改回 “info”。

如果您希望记录到其他文件或完全不同的介质（如数据库），请调整 logback.xml 文件以满足您的需要。详见 Logback 手册。

有时为了方便调试，将 Java gateway 作为控制台应用程序而不是守护程序启动是更有用的。为此，请在 settings.sh 中注释掉 PID_FILE 变量。如果省略 PID_FILE ，则 startup.sh 脚本将 Java gateway 作为控制台应用程序启动，并让 Logback 使用 lib/logback-console.xml 文件，这不仅会记录到控制台，还会启用日志记录级别 “debug”。

最后，值得注意的，由于 Java gateway 使用 SLF4J 进行日志记录，因此可以适当地将 JAR 包放置在 lib 目录中，来将 Logback 替换为您所选的框架。详见 SLF4J 手册。

5 Sender

概述

Zabbix sender 是一个命令行应用程序，可用于将性能数据发送到 Zabbix server 进行处理。

该实用程序通常用于长时间运行的用户脚本，用于定期发送可用性和性能数据。

要将结果直接发送到 Zabbix server 或 proxy，必须配置trapper 监控项 类型。

运行 Zabbix sender

一个运行 Zabbix UNIX sender 的例子：

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

其中：

- z - Zabbix server 主机（也可以使用 IP 地址）
- s - 被监控主机的名称（在前端注册）
- k - 监控项键值
- o - 要发送的值

Attention:
包含空格的选项必须使用双引号引用。

Zabbix sender 可通过从输入文件发送多个值。详见Zabbix sender manpage。

Zabbix sender 接受 UTF-8 编码的字符串（对于类 UNIX 系统和 Windows ），且在文件中没有字节顺序标记（BOM）。

Zabbix sender 同样可以在 Windows 上运行：

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

从 Zabbix 1.8.4 开始，zabbix_sender 实时发送方案已得到改进，可以连续接收多个传递给它的值，并通过单个连接将它们发送到服务器。两个不超过 0.2 秒的值可以放在同一堆栈中，但最大 pooling 时间仍然是 1 秒。

Note:
Zabbix sender 如果指定的配置文件中存在无效（不遵循 *parameter=value* 注释）的参数条目，则 Zabbix sender 将终止。

6 Get

概述

Zabbix get 是一个命令行应用，它可以用于与 Zabbix agent 进行通信，并从 Zabbix agent 那里获取所需的信息。

该应用通常被用于 Zabbix agent 故障排错。

运行 Zabbix get

一个在 UNIX 下运行 Zabbix get 以从 Zabbix agent 获取 processor load 的值的例子。

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

另一个运行 Zabbix get 以从网站捕获一个字符串的例子：

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

请注意，此处的监控项键值包含空格，因此引号用于将监控项键值标记为 shell。引号不是监控项键值的一部分；它们将被 shell 修剪，不会被传递给 Zabbix agent。

Zabbix get 接受以下命令行参数：

```
  -s --host <host name or IP>      指定目标主机名或IP地址
  -p --port <port number>          指定主机上运行 Zabbix agent 的端口号。默认端口10050
  -I --source-address <IP address> 指定源 IP 地址
  -k --key <item key>              指定要从监控项键值检索的值
  -h --help                        获得帮助
  -V --version                     显示版本号
```

详见Zabbix get 手册。

Zabbix get 同样可以在 Windows 上运行：

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

4. 安装

请使用侧边栏导航来访问此章节中的内容。

1 获取 Zabbix

概述

获取 Zabbix 安装介质有四种方法：

- 从发行包 安装；
- 下载最新的归档源码包并编译它；
- 从容器 中安装；
- 下载Zabbix 应用。

请转到 Zabbix 下载页面 下载最新的源码包或应用，此页面提供最新版本的直接链接。如果要下载旧版本，请参阅以下稳定版本下载链接。

2 安装要求

硬件

内存和磁盘

Zabbix 运行需要物理内存和磁盘空间。如果刚接触 Zabbix，128 MB 的物理内存和 256 MB 的可用磁盘空间可能是一个很好的起点。然而，所需的内存和磁盘空间显然取决于被监控的主机数量和配置参数。如果您计划调整参数以保留较长的历史数据，那么您应该考虑至少有几 GB 磁盘空间，以便有足够的磁盘空间将历史数据存储在数据库中。

每个 Zabbix 守护程序进程都需要与数据库服务器建立多个连接。为连接分配的内存量取决于数据库引擎的配置。

> **Note:**
> 您拥有的物理内存越多，数据库（以及 Zabbix ）的工作速度就越快！

CPU

Zabbix，尤其是 Zabbix 数据库可能需要大量 CPU 资源，该具体取决于被监控参数的数量和所选的数据库引擎。

其他硬件

如果需要启用短信（SMS）通知功能，需要串行通讯口（serial communication port）和串行 GSM 调制解调器（serial GSM modem）。USB 转串行转接器也同样可以工作。

硬件资源配置参考

下表提供了几个硬件配置参考：

规模平	CPU	内存数据库	受监控的主机数量	
小型 C	ntOS V	rtual Appliance M	SQL InnoDB 1	0
中型 C	ntOS 2	CPU cores/2GB M	SQL InnoDB 5	0
大型 R	dHat En- ter- prise Linux 4	CPU cores/8GB R	ID10 MySQL InnoDB 或 PostgreSQL &g	;1000
极大型 Re	Hat En- ter- prise Linux 8	PU cores/16GB Fa	t RAID10 MySQL InnoDB 或 PostgreSQL &gt	10000

Note:
实际上，Zabbix 环境的配置非常依赖于监控项（主动）和更新间隔。如果是进行大规模部署，强烈建议将数据库独立部署。

受支持的平台

由于服务器操作的安全性要求和任务关键性，UNIX 是唯一能够始终如一地提供必要性能、容错和弹性的操作系统。Zabbix 以市场主流的操作系统版本运行。

经测试，Zabbix 可以运行在下列平台：

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris
- Windows：自 XP 以来的所有桌面和服务器版本（仅限 Zabbix agent）

Note:
Zabbix 可以在其他类 Unix 操作系统上运行。

Attention:
如果使用加密编译，Zabbix 将禁用核心转储（Core dumps），如果系统不允许禁用核心转储，则 Zabbix 不会启动。

软件

Zabbix 是基于先进 Apache Web 服务器、领先的数据库引擎和 PHP 脚本语言构建的。

数据库管理系统

数据库版本	备注	
MySQL	5.0.3 - 8.0.x	使用 MySQL 作为 Zabbix 后端数据库。需要 InnoDB 引擎。MariaDB 同样支持。
Oracle	10g or later	使用 Oracle 作为 Zabbix 后端数据库。

数据库版本	备注	
PostgreSQL	8.1 or later	使用 PostgreSQL 作为 Zabbix 后端数据库。建议使用 PostgreSQL 8.3 以上的版本，以提供更好的 VACUUM 性能。
IBM DB2	9.7 or later	使用 DB2 作为 Zabbix 后端数据库。

数据库版本	备注	
SQLite	3.3.5 or later	只有Zab-bix proxy支持SQLite，可以使用SQLite作为Zab-bix proxy数据库。

Attention:
值得注意的是，对于 IBM DB2 的支持是实验性的！

前端

Zabbix 前端需要使用下列软件:

软件版	备注
Apache	1.3.12 或以上
PHP	5.4.0 或以上
PHP 扩展库：	

软件版	备注	
gd	2.0 or later	PHP GD 扩展库必须支持 PNG 图像 (**--***with-png-dir*)、JPEG 图像 (**--***with-jpeg-dir*) 和 FreeType 2 (**--***with-freetype-dir*).
bcmath		php-bcmath (**--***enable-bcmath*)
ctype		php-ctype (**--***enable-ctype*)
libXML	2.6.15 或以上 ph	- xml or php5-dom，如果发布者提供独立的部署包。

软件版	备注
xmlreader	php-xmlreader，如果发布者提供独立的部署包。
xmlwriter	php-xmlwriter，如果发布者提供独立的部署包。
session	php-session，如果发布者提供独立的部署包。

软件版	备注
sockets	php-net-socket (**--***enable-sockets*) 。用户脚本支持所需要的组件。
mbstring	php-mbstring (**--***enable-mbstring*)
gettext	php-gettext (**--***with-gettext*)。用于多语言翻译支持。
ldap	php-ldap。只有在前端使用 LDAP 认证时才需要。

软件版	备注
ibm_db2	使用 IBM DB2 作为 Zabbix 后端数据库所需要的组件。
mysqli	使用 MySQL 作为 Zabbix 后端数据库所需要的组件。
oci8	使用 Oracle 作为 Zabbix 后端数据库所需要的组件。

软件版	备注
pgsql	使用 PostgreSQL 作为 Zabbix 后端数据库所需要的组件。

Zabbix 也许可以在以前的 Apache、MySQL、Oracle 和 PostgreSQL 版本上运行。

Attention:
值得注意的是，如果需要使用默认 DejaVu 以外的字体，可能会需要 PHP 的 imagerotate 函数。如果缺少，则在 Zabbix 前端查看图形时显示异常。该函数只有在使用捆绑的 GD 库编译 PHP 时才可用。在 Debian 和某些发行版本中，这个问题不存在。

客户端浏览器

浏览器必须启用 Cookies 和 Java Script 。

支持最新版本的 Google、Mozilla Firefox、Microsoft Internet Explorer 和 Opero。其他浏览器（Apple Safari、Konqueror）也许会支持。

Warning:
值得注意的，为了执行 IFrame 的 “同源政策”，意味着 Zabbix 不能放在不同域的 frames 中。

但是，如果放置在 frames 中的页面和 Zabbix 前端位于同一个域中，则置于 Zabbix frames 中的页面将可以访问 Zabbix 前端（通过 JavaScript）。像 `http://secure-zabbix.com/cms/page.html` 这样的页面，如果置于 `http://secure-zabbix.com/zabbix/` 的聚合图形或仪表盘上，将拥有对 Zabbix 的完整 JS 访问权限。

Server

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

Requirement	Status	Description
libpcre	Mandatory	PCRE library is required for Perl Compatible Regular Expression (PCRE) support. The naming may differ depending on the GNU/Linux distribution, for example ’libpcre3’ or ’libpcre1’. Note that you need exactly PCRE (v8.x); PCRE2 (v10.x) library is not used.
libevent		Required for bulk metric support and IPMI monitoring. Version 1.4 or higher. Note that for Zabbix proxy this requirement is optional; it is needed for IPMI monitoring support.
libpthread		Required for mutex and read-write lock support.
zlib		Required for compression support.
OpenIPMI	Optional	Required for IPMI support.
libssh2		Required for SSH support. Version 1.0 or higher.

Requirement	Status	Description
fping		Required for ICMP ping items.
libcurl		Required for web monitoring, VMware monitoring and SMTP authentication. For SMTP authentication, version 7.20.0 or higher is required. Also required for Elasticsearch.
libiksemel		Required for Jabber support.
libxml2		Required for VMware monitoring.
net-snmp		Required for SNMP support.

Java gateway

如果从源码存储库或归档中获取 Zabbix，则在源代码树中已包含必需的依赖关系。

如果从发行包中获取 Zabbix ，则封装系统里已提供了必要的依赖关系。

在上述两种情况下，即可准备部署软件了，而不需要下载额外的依赖包。

但是，如果您希望提供这些依赖关系的版本（例如，如果您正在为某些 Linux 发行版准备软件包），则下面是 Java gateway 已知可以使用的库的版本列表。Zabbix 也许可以与这些库的其他版本一起使用。

下表列出了原始代码中当前与 Java gateway 捆绑在一起的 JAR 文件：

库	可网站	备注	
logback-core-0.9.27.jar	EPL 1.0, LGPL 2.1	http://logback.qos.ch/	0.9.27、1.0.13 和 1.1.1 测试通过。
logback-classic-0.9.27.jar	EPL 1.0, LGPL 2.1	http://logback.qos.ch/	0.9.27、1.0.13 和 1.1.1 测试通过。
slf4j-api-1.6.1.jar	MIT License	http://www.slf4j.org/	1.6.1、1.6.6 和 1.7.6 测试通过

库	可网站	备注	
android-json-4.3_r3.1.jar	Apache License 2.0	https://android.googlesource.com/platform/libcore/+/master/json	2.3.3_r1.1 和 4.3_r3.1 测试通过。关于创建 JAR 文件，详见 src/zabbix_java/l 说明。

Java gateway 使用 Java 1.6 及更高版本编译和运行。如需要对 Java gateway 预编译版本进行编译，建议使用 Java 1.6 进行编译，直到最新版本。

数据库容量

Zabbix 配置文件数据需要固定数量的磁盘空间，且增长不大。

Zabbix 数据库大小主要取决于这些变量，这些变量决定了存储的历史数据量:

- 每秒处理值的数量

这是 Zabbix server 每秒接收的新值的平均数。例如，如果有 3000 个监控项用于监控，取值间隔为 60 秒，则这个值的数量计算为 3000/60 = ** 50 **。

这意味着每秒有 50 个新值被添加到 Zabbix 数据库中。

- 关于历史数据的管家设置

Zabbix 将接收到的值保存一段固定的时间，通常为几周或几个月。每个新值都需要一定量的磁盘空间用于数据和索引。

所以，如果我们每秒收到 50 个值，且希望保留 30 天的历史数据，值的总数将大约在 (**30***24*3600)* **50** = 129.600.000，即大约 130M 个值。

根据所使用的数据库引擎，接收值的类型（浮点数、整数、字符串、日志文件等），单个值的磁盘空间可能在 40 字节到数百字节之间变化。通常，数值类型的每个值大约为 90 个字节。

在上面的例子中，这意味着 130M 个值需要占用 130M * 90 bytes = **10.9GB** 磁盘空间。

> **Note:**
> 文本和日志类型的监控项值的大小是无法确定的，但可以以每个值大约 500 字节来计算。

- 趋势数据的管家设置

Zabbix 为表 **trends** 中的每个项目保留 1 小时的最大值 / 最小值 / 平均值 / 统计值。该数据用于趋势图形和历史数据图形。这一个小时的时间段是无法自定义。

Zabbix 数据库，根据数据库类型，每个值总共需要大约 90 个字节。

假设我们希望将趋势数据保持 5 年。3000 个监控项的值每年需要占用 3000*24*365* **90** = **2.2GB** 空间，或者 5 年需要占用 **11GB** 空间。

- 事件的管家设置

每个 Zabbix 事件需要大约 170 个字节的磁盘空间。很难估计 Zabbix 每天生成的事件数量。在最坏的情况下，假设 Zabbix 每秒生成一个事件。

这意味着如果想要保留 3 年的事件，这将需要占用 **3***365*24*3600* **170** = **15GB** 的空间。

下表包含可用于计算 Zabbix 系统所需磁盘空间的公式：

参数所	磁盘空间的计算公式（单位：字节）
Zabbix 配置文件固定大	。通常为 10MB 或更少。
History	days*(items/refresh rate)*24*3600*bytes items：监控项数量。 days：保留历史数据的天数。 refresh rate：监控项的更新间隔。 bytes：保留单个值所需要占用的字节数，依赖于数据库引擎，通常为 ~90 字节。
Trends	days*(items/3600)*24*3600*bytes items：监控项数量。 days：保留历史数据的天数。 bytes：保留单个趋势数据所需要占用的字节数，依赖于数据库引擎，通常为 ~90 字节。
Events	days*events*24*3600*bytes events：每秒产生的事件数量。假设最糟糕的情况下，每秒产生 1 个事件。 days：保留历史数据的天数。 bytes：保留单个趋势数据所需的字节数，取决于数据库引擎，通常为 ~170 字节。

Note:
根据使用 MySQL 后端数据库的实际统计数据中收集到的平均值，例如监控项为数值类型的值约 90 个字节，事件约 170 个字节。

因此，所需要的磁盘总空间按下列方法计算：

配置文件数据 + 历史数据 + 趋势数据 + 事件数据

在安装 Zabbix 后不会立即使用磁盘空间。数据库大小取决于管家设置，在某些时间点增长或停止增长。

时间同步

在运行 Zabbix 的服务器上拥有精确的系统日期非常重要。ntpd 是最受欢迎的守护进程，它将主机的时间与其他服务器的时间同步。对于所有运行 Zabbix 组件的系统，强烈建议这些系统的时间保持同步。

如果时间未同步，Zabbix 将在建立数据连接之后，根据得到的客户端和服务器的时间戳，并通过客户端和服务器的时间差对获得值的时间戳进行调整，将获得值的时间戳转化为 Zabbix server 的时间。为了尽可能简化并且避免可能的并发问题出现，网络延迟将会被忽略。因此，通过主动连接（active agent, active proxy, sender）获得的时间戳数据将包含网络延迟，通过被动连接（passive proxy）获得的数据已经减去了网络延迟。所有其他监控类型都在服务器时间里完成，并且不会调整其时间戳。

安全设置 **Zabbix** 的最佳实践

概述

本章节包含为了以安全的方式设置 Zabbix 应遵守的最佳实践。

Zabbix 的功能不依赖于此处的实践。但建议使用它们以提高系统的安全性。

Zabbix agent 的安全用户

在默认的配置中，Zabbix server 和 Zabbix agent 进程共享一个 “zabbix” 用户。如果您希望确保 Zabbix agent 无法访问 Zabbix server 配置中的敏感详细信息（例如，数据库登录信息），则应以不同的用户身份运行 Zabbix agent：

1. 创建一个安全用户；
1. 在 Zabbix agent 的配置文件 中指定此用户（修改’User’ parameter）；
1. 以拥有管理员权限的用户重启 Zabbix agent。之后，此权限将赋予给先前指定的用户。

为 Zabbix 前端设置 SSL

在 RHEL/Centos 操作系统上，安装 mod_ssl 包：

```
yum install mod_ssl
```

为 SSL keys 创建目录：

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

创建 SSL 证书：

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

下面提示内容适当填写。最重要的一行是请求 Common Name 的行。您需要输入要与服务器关联的域名。如果您没有域名，则可以输入公共 IP 地址。下面将使用 *example.com*。

```
Country Name (两个字母) [XX]:
State or Province Name (全名) []:
Locality Name (eg, city) [默认的城市]:
Organization Name (eg, company) [默认的公司名]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:example.com
Email Address []:
```

编辑 Apache SSL 配置：

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

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

重启 Apache 服务使以上修改的配置生效：

```
systemctl restart httpd.service
```

在 URL 的根目录上启用 Zabbix

将虚拟主机添加到 Apache 配置，并将文档根目录的永久重定向设置为 Zabbix SSL URL。不要忘记将 *example.com* 替换为服务器的实际名称。

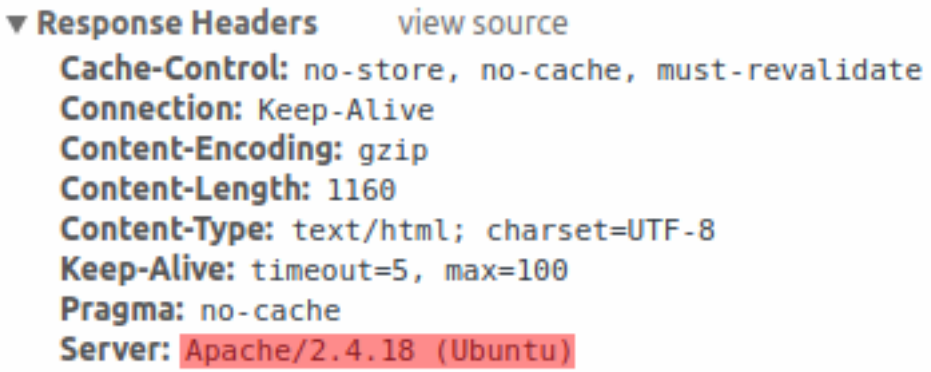
```
/etc/httpd/conf/httpd.conf

#Add lines

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

重启 Apache 服务使以上修改的配置生效：

```
systemctl restart httpd.service
```

禁用曝光的 Web 服务器信息

建议在 Web 服务器强化过程中禁用所有 Web 服务器签名。默认情况下，Web 服务器正在公开软件签名：



可以通过向 Apache（用作示例）配置文件添加两行来禁用签名：

```
ServerSignature Off
ServerTokens Prod
```

可以通过更改 php.ini 配置文件来禁用 PHP 签名（X-Powered-By HTTP header）（默认情况下禁用签名）：

```
expose_php = Off
```

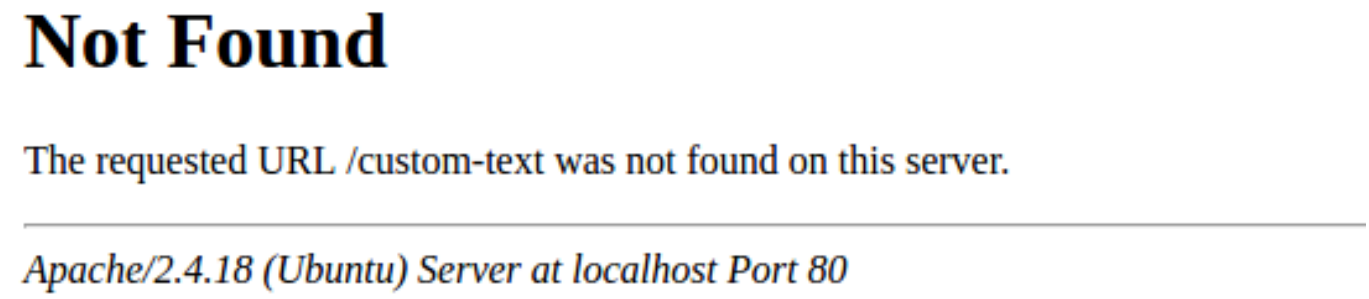
若要应用配置文件更改，需要重新启动 Web 服务器。

通过在 Apache 中使用 mod_security（libapache2-mod-security2）可以实现额外的安全级别。mod_security 允许删除服务器签名，而不是仅仅从服务器签名中删除版本。通过在安装 mod_security 之后将 “SecServerSignature” 更改为任何所需的值，可以将签名更改为任何值。

请参阅 Web 服务器的文档以获取有关如何删除/更改软件签名的帮助。

Disabling default web server error pages

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

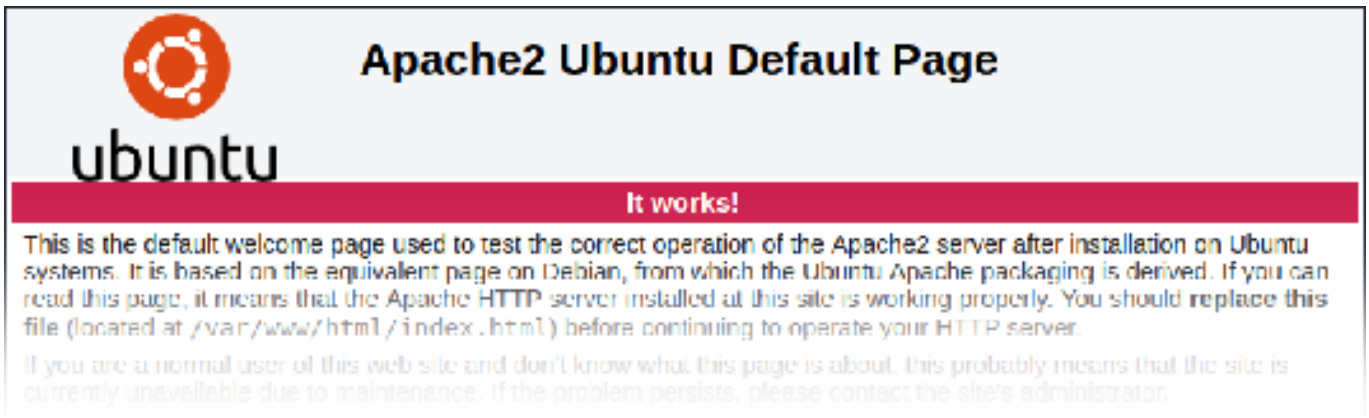


Default error pages should be replaced/removed as part of the web server hardening process. The ”ErrorDocument” directive can be used to define a custom error page/text for Apache web server (used as an example).

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

删除 Web 服务器的测试页面

建议删除 Web 服务器测试页以避免信息泄露。默认情况下，Web 服务器的 webroot 包含一个名为 index.html 的测试页（以 Ubuntu 上的 Apache2 为例）：



应删除测试页面，或者应将其作为 Web 服务器强化过程的一部分使用。

3 从源代码包安装

您可以通过从源代码编译来获取最新版本的 Zabbix。

这里提供了从源代码安装 Zabbix 的具体步骤。

1 安装 Zabbix 守护进程

1 下载源代码存档

转到 Zabbix download page 下载源代码存档。待下载完毕后，执行以下命令解压缩源代码存档：

```
$ tar -zxvf zabbix-5.2.0.tar.gz
```

Note:
请在命令中输入正确的 Zabbix 版本。它必须与下载的存档的名称匹配。

2 创建用户账户

对于所有 Zabbix 守护进程，需要一个非特权用户。如果从非特权用户帐户启动 Zabbix 守护程序，它将以该用户身份运行。

然而，如果一个守护进程以 “root” 启动，它会切换到 “zabbix” 用户，且这个用户必须存在。在 Linux 系统中，可以使用下面命令建立一个用户（该用户属于自己的用户组，“zabbix”）：

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

而对于 Zabbix 前端安装，并不需要单独的用户帐户。

如果 Zabbix server 和agent 运行在相同的机器上，建议使用不同的用户运行来 Zabbix server 和 agent。否则，如果两者都作为同一用户运行，则 Zabbix agent 可以访问 Zabbix server 配置文件，并且可以轻松检索到 Zabbix 中的任何管理员级别的用户，例如，数据库密码。

Attention:
以 `root` 、`bin` 或其他具有特殊权限的账户运行 Zabbix 是非常危险的。

3 创建 Zabbix 数据库

对于 Zabbix server 和proxy 守护进程以及 Zabbix 前端，必须需要一个数据库。但是 Zabbix agent 并不需要。

SQL 脚本 用于创建数据库 schema 和插入 dataset。Zabbix proxy 数据库只需要数据库 schema，而 Zabbix server 数据库在建立数据库 schema 后，还需要 dataset。

当创建数据库后，继续执行编译 Zabbix 的步骤。

4 配置源代码

当配置 Zabbix server 或者 proxy 的源代码时，需要指定所使用的数据库类型。一次只能使用 Zabbix server 或 Zabbix proxy 进程编译一种数据库类型。

如果要查看所有受支持的配置选项，请在解压缩的 Zabbix 源代码目录中运行：

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

如果要配置 Zabbix server 和 Zabbix proxy 的源代码，您可以运行以下内容：

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

Note:
从 Zabbix 3.0.0 起，STMP 认证需要需要 cURL 7.20.0 或更高版本的 --with-libcurl 配置选项。
从 Zabbix 2.2.0 起，虚拟机监控需要 --with-libcurl 和 --with-libxml2 配置选项。

Attention:
从 Zabbix 3.4 开始，Zabbix 将始终使用 PCRE 库进行编译；安装时它不是可选的。--with-libpcre=[DIR] 只允许指向特定的基础安装目录，而不是通过 libpcre 文件的多个公共位置搜索。

如果要配置 Zabbix server 的源代码（使用 PostgreSQL 等），您可以运行：

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

如果要配置 Zabbix proxy 的源代码（使用 SQLite 等），您可以运行：

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

如果要配置 Zabbix agent 的源代码，您可以运行：

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

您可以使用 --enable-static 标识静态连接类库。如果你打算在不同的服务器之间分发已编译的二进制文件，则必须使用此标识使这些二进制文件在没有必需库的情况下工作。请注意 --enable-static 在 Solaris 系统下无效 。

Attention:
不建议在搭建 Zabbix server 的时候使用 --enable-static 选项。

为了静态搭建 Zabbix server，您必须拥有每个所需的外部类库的静态版本。配置脚本中不提供这些类库的严格检查。

Note:
命令行工具 zabbix_get 和 zabbix_sender 只有在 --enable-agent 选项启用时才会被编译。

Note:
如果 MySQL 客户端类库不处在默认的位置，则需要添加可选的 MySQL 的配置文件 --with-mysql=/<path_to_the_file>/mysql_config ，以选择所需的路径。
这可以有效解决，一个系统上安装了多个版本的 MySQL 或者 MariaDB 的情况。

Note:
使用 --with-ibm-db2 以标识指定的 CLI API 位置。
使用 --with-oracle 以标识指定的 OCI API 位置。

关于使用加密，详见使用加密方式编译 Zabbix。

5 安装

Note:
如果从 git 安装，需要先运行以下命令：
`$ make dbschema`

```
make install
```

这步需要使用一个拥有足够权限的用户来运行 (如'root'，或者使用 sudo)。

运行 `make install` 将使用在 /usr/local/sbin 下的守护进程二进制文件（zabbix_server, zabbix_agentd, zabbix_proxy）和在 /usr/local/bin 下的客户端二进制文件进行默认安装。

Note:
如需要指定 /usr/local 以外的位置，可在之前的配置源代码的步骤中使用 --prefix，例如 --prefix=/home/zabbix。在这个案例中，守护进程的二进制文件会被安装在 <prefix>/sbin 下，工具会安装在 <prefix>/bin 下。帮助文件会安装在 <prefix>/share 下。

6 查看和编辑配置文件

- 在此编辑 Zabbix agent 的配置文件 **/usr/local/etc/zabbix_agentd.conf**

您需要为每台安装了 zabbix_agentd 的主机配置这个文件。

您必须在这个文件中指定 Zabbix server 的 **IP** 地址。若从其他主机发起的请求会被拒绝。

- 在此编辑 Zabbix server 的配置文件 **/usr/local/etc/zabbix_server.conf**

您必须指定数据库的名称、用户和密码（如果使用的话）。

如果您进行小型环境部署（最多十个受监控主机），其余参数的默认值将适合您的环境。如果要最大化 Zabbix server（或 proxy）的性能，则应更改默认参数。详见性能调整。

- 如果您安装了 Zabbix proxy，请在此编辑 proxy 的配置文件 **/usr/local/etc/zabbix_proxy.conf**

您必须指定 Zabbix server 的 IP 地址和 Zabbix proxy 主机名（必须被 Zabbix server 识别），同时也要指定数据库的名称、用户和密码 (如果使用的话)。

Note:
使用 SQLite 必须指定数据库文件的完整路径；数据库用户和密码不是必须的。

7 启动守护进程

在 Zabbix server 端运行 zabbix_server：

```
shell> zabbix_server
```

Note:
值得注意的是，确保您的系统允许分配 36MB（或更多）的共享内存，否则 Zabbix server 将无法启动，并会在 Zabbix server 日志文件中看到"Cannot allocate shared memory for <type of cache>." 这样的报错信息。这可能会发生在 FreeBSD 和 Solaris 8 上。
详见本页底部的" 另请参阅" 部分，了解如何配置共享内存。.

在受监控的主机上运行 zabbix_agentd：

```
shell> zabbix_agentd
```

Note:
值得注意的是，请确保您的系统允许分配 2MB 的共享内存，否则 Zabbix agent 可能会无法运行，并会在 Zabbix agent 日志文件中看到"Cannot allocate shared memory for collector." 这样的报错信息。这可能会发生在 Solaris 8 上。

如果您安装了 Zabbix proxy，请运行 zabbix_proxy：

```
shell> zabbix_proxy
```

2 安装 Zabbix web 界面

复制 PHP 文件

Zabbix 前端是 PHP 编写的，所以必须运行在支持 PHP 的 Web 服务器上。只需要简单的从 frontends/php 路径下复制 PHP 文件到 Web 服务器的 HTML 文档目录，即可完成安装。

Apache Web 服务器的 HTML 文档目录通常包括：

- /usr/local/apache2/htdocs （从源代码安装 Apache 的默认目录）
- /srv/www/htdocs (OpenSUSE, SLES)
- /var/www/html (Debian, Ubuntu, Fedora, RHEL, CentOS)

建议使用子目录替代 HTML 根目录。可以使用下列命令，以创建一个子目录并复制 Zabbix 的前端文件到这个目录下（注意替换为实际的目录）：

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

如果准备从 git 安装英语以外的语言，您必须生成翻译文件。可以运行下列命令：

```
locale/make_mo.sh
```

需要来自 gettext 安装包的 `msgfmt` 组件。

Note:
此外，使用英语以外的语言，需要在 Web 服务器上安装该语言对应的 locale 。详见 "用户文件" 页面中的" 另请参阅" 板块，以寻找如何安装它（如果需要的话）。

安装前端

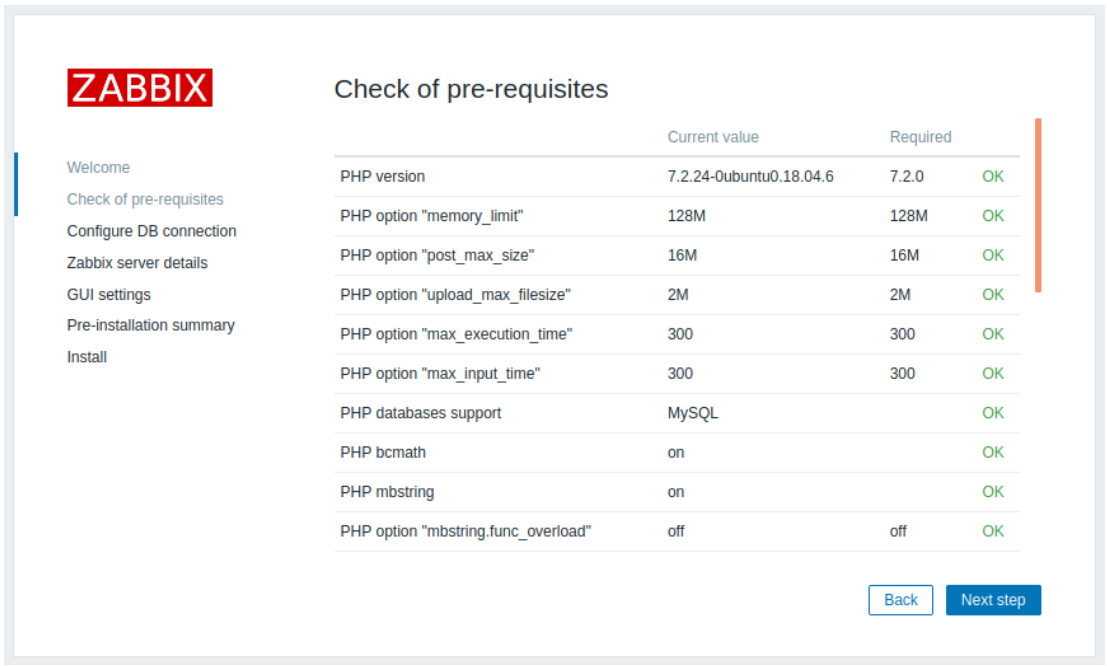
第一步

在您的浏览器打开 Zabbix 链接：http://<server_ip_or_name>/zabbix

您可以看到前端安装向导的第一个页面。

第二步

请确认满足所有的软件安装前置条件。



先决条件最低要	描述	
PHP 版本 5	4.0	
PHP memory_limit 选项 1	8MB 位	php.ini: memory_limit = 128M
PHP post_max_size 选项 1	MB 位	php.ini: post_max_size = 16M
PHP upload_max_filesize 选项 2	B 位	php.ini: upload_max_fil = 2M

先决条件最低要	描述	
PHP max_execution_time 选项 3	0 seconds (此值允许为 0 和 -1) 位于 php.	ni: max_execution = 300
PHP max_input_time 选项 3	0 seconds (此值允许为 0 和 -1) 位于 php.	ni: max_input_tim = 300
PHP session.auto_start 选项必	禁用此值 In php.	ni: session.auto_st = 0
数据库支持其中之一	MySQL,、Oracle、Post-greSQL、IBM DB2	mysql、oci8、pgsql、ibm_db2
bcmath	必须安装下列模块中的一种	php-bcmath
mbstring		php-mbstring
PHP mbstring.func_overload 选项必	禁用此值位于 php.	ni: mbstring.func_ = 0
PHP always_populate_raw_post_data option	必须禁用此值只适用于	HP 5.6.0 或更高的版本。位于 php.ini: always_populat = -1
sockets		php-net-socket 用于支持用户脚本。

先决条件最低要	描述
gd	2.0 或更高 ph - gd. PHP GD 扩展必须支持 PNG 图像 (- - *with-png-dir*),、 JPEG (- - *with-jpeg-dir*) 图像和 FreeType 2 (- - *with-freetype-dir*).
libxml	2.6.15 php-xml or php5-dom
xmlwriter	php-xmlwriter
xmlreader	php-xmlreader
ctype	php-ctype
session	php-session

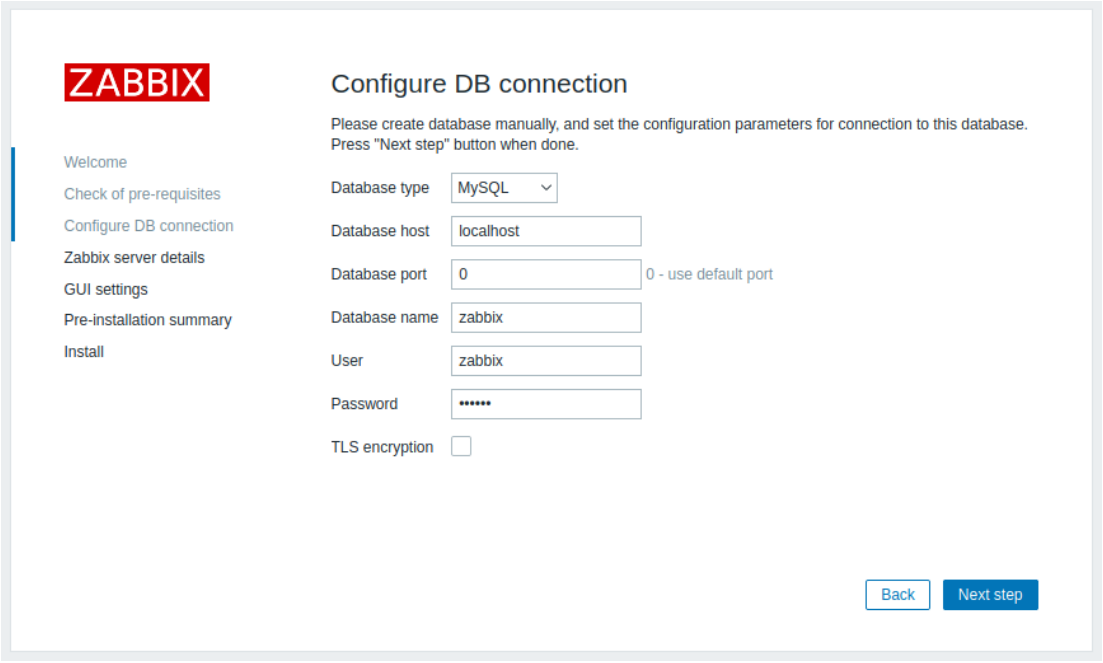
先决条件最低要	描述
gettext	php-gettext 从 Zabbix 2.2.1 起，PHP gettext 扩展不是安装 Zabbix 的强制性要求。如果 gettext 没有安装，前端也可以照常运行，但翻译将不可用。

列表中也可能包含可选的先决条件。不满足的可选先决条件以橙色显示，同时标识为 *Warning* 状态。即使存在不满足的可选先决条件，安装仍可以继续进行。

Attention:
如果需要更改 Apache 的用户或用户组，则必须验证会话文件夹的权限。否则 Zabbix 安装可能无法继续。

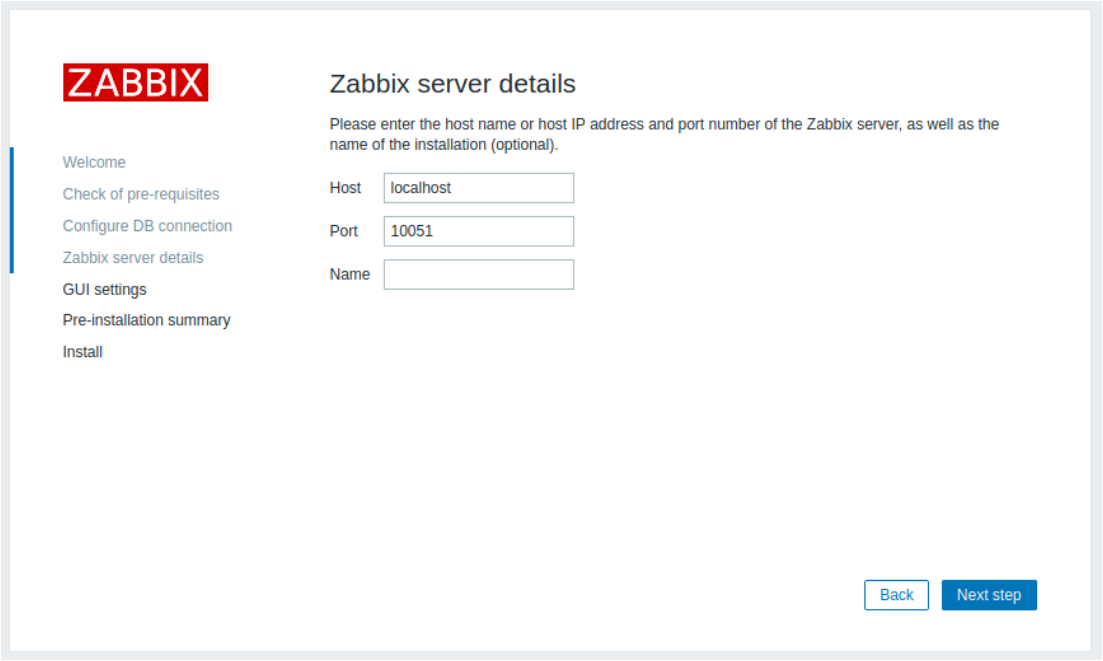
第三步

请输入连接到数据库的详细信息。Zabbix 数据库必须提前创建完成。



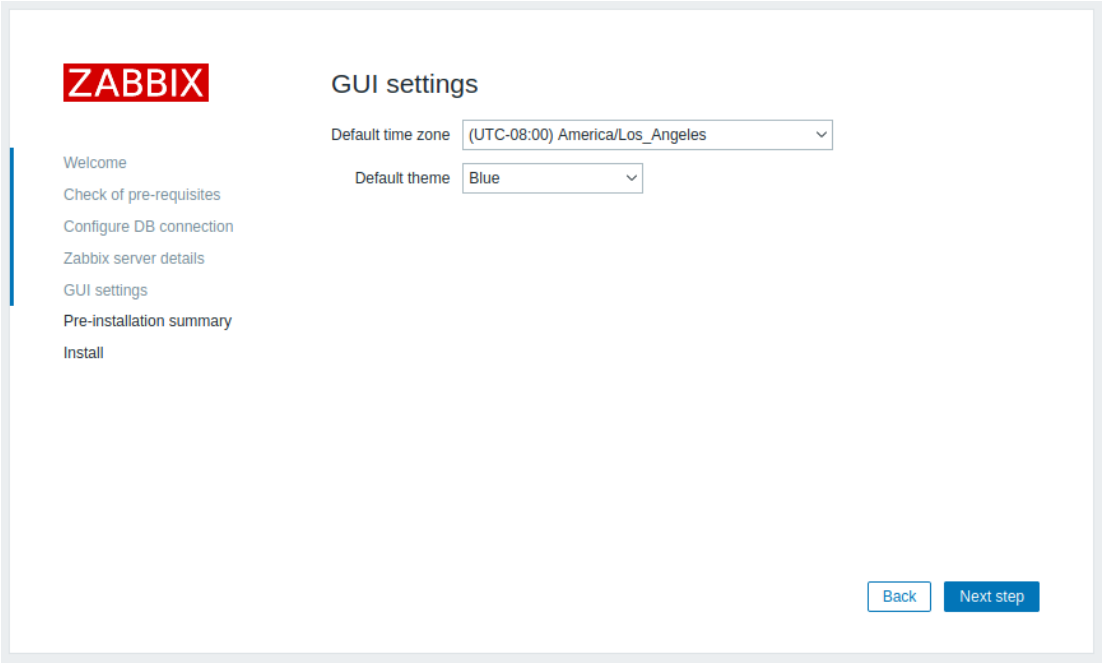
第四步

请输入 Zabbix server 的详细信息。



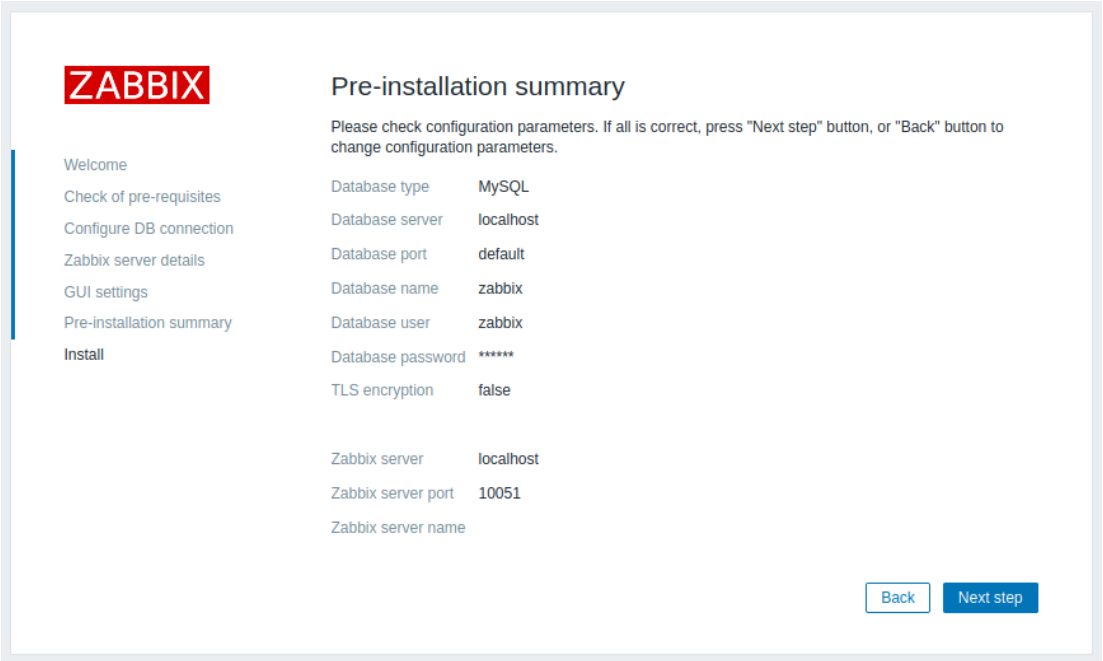
第五步

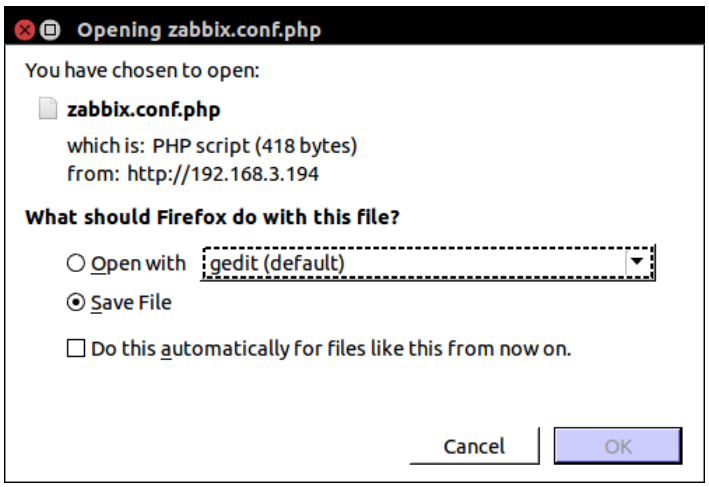
查看设置的摘要。



第六步

下载配置文件并将其放在 conf/ 路径下，即在您复制 Zabbix PHP 文件的 Web 服务器 HTML 文档子目录中。

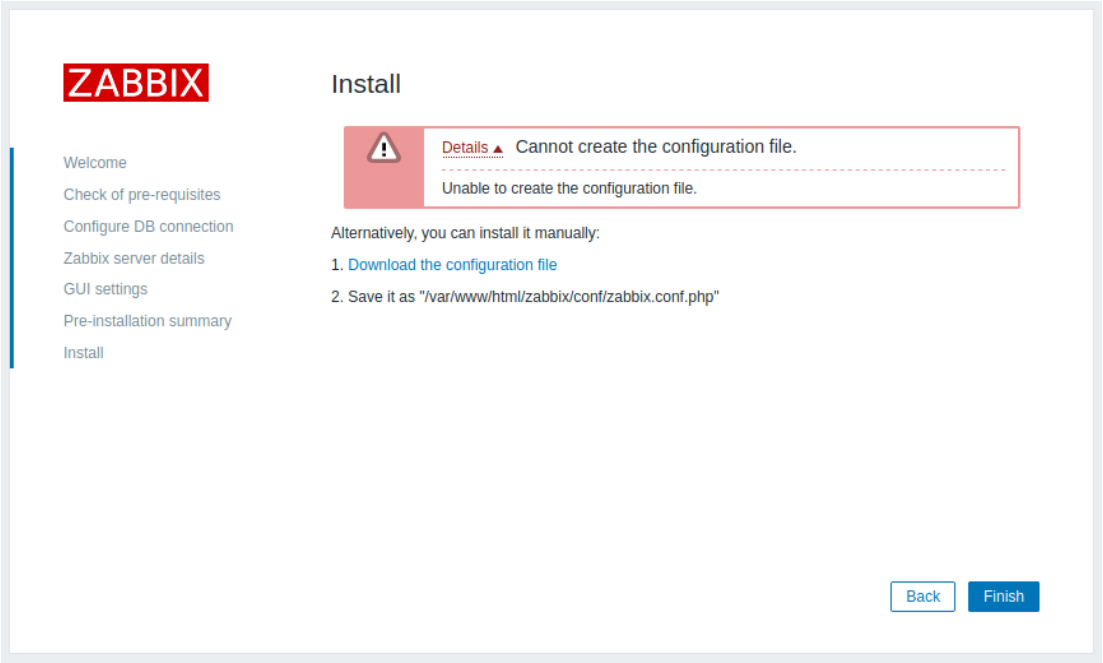




Note:
如果 Web 服务器用户具有对 conf/ 目录的写访问权，则会自动保存配置文件，并且可以直接继续执行下一步。

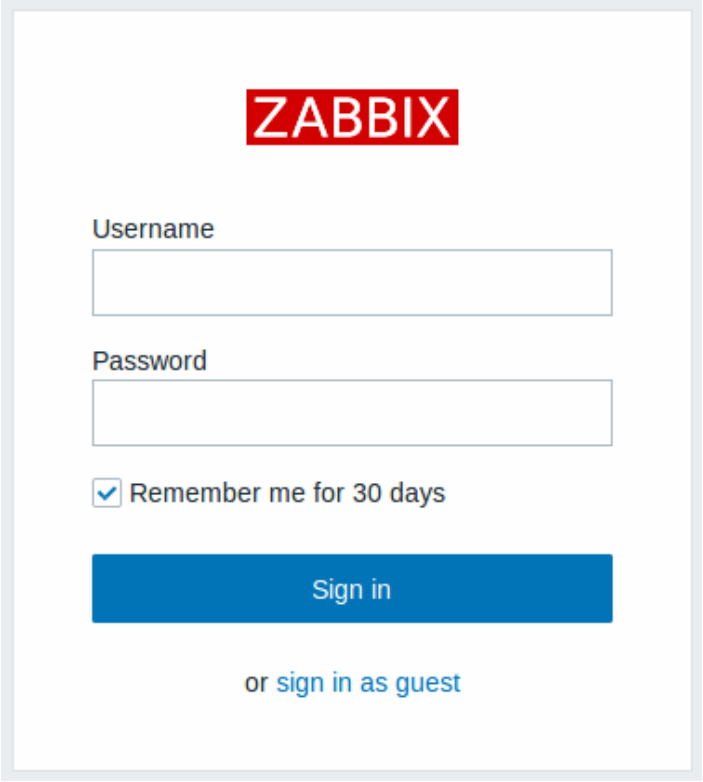
第七步

完成安装。



第八步

Zabbix 前端已经就绪！默认的用户名是 **Admin**，密码是 **zabbix**。



前往开始使用 Zabbix。

另请参阅

1. 如何为 Zabbix 守护进程配置共享内存？

4 从二进制包安装

从发行包安装

我们为大多数主流的操作系统提供了 Zabbix 的发行包。您可以使用这些发行包安装以 Zabbix。

> **Note:**
> 操作系统分发软件的仓库中可能会缺少 Zabbix 的最新版本。

从 Zabbix 官方的软件仓库安装

Zabbix SIA 提供了官方的 RPM 和 DEB 发行包：

- Red Hat Enterprise Linux/CentOS
- Debian/Ubuntu

发行包文件可在 repo.zabbix.com 下载。在服务器上也提供了 Yum 和 apt 软件仓库。这里的子页面提供了从发行包安装 Zabbix 的一步步教程。

1 Red Hat Enterprise Linux/CentOS

概述

官方的 Zabbix 发行包适用于 RHEL 7、CentOS 7 和 Oracle Linux 7。在本文中，将使用 RHEL 来指代这三个操作系统。

一些 Zabbix agent 和 Zabbix proxy 发行包也适用于 RHEL 6 和 RHEL 5。

添加 Zabbix 软件仓库

安装软件仓库配置包，这个包包含了 yum（软件包管理器）的配置文件。

RHEL 7:

```
# rpm -ivh http://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-release-4.0-1.el7.noarch.rpm
```

RHEL 6:

```
# rpm -ivh http://repo.zabbix.com/zabbix/4.0/rhel/6/x86_64/zabbix-release-4.0-1.el6.noarch.rpm
```

RHEL 5:

```
# rpm -ivh http://repo.zabbix.com/zabbix/4.0/rhel/5/x86_64/zabbix-release-4.0-1.noarch.rpm
```

前端安装的先决条件

Zabbix 前端需要额外的基础安装包。您需要在运行 Zabbix 前端的系统中启用可选 rpms 的软件仓库：

RHEL 7:

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

安装 Server/proxy/前端

安装 Zabbix server（适用于 RHEL7，在 RHEL 6 上弃用）并使用 MySQL 数据库：

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

安装 Zabbix proxy 并使用 MySQL 数据库：

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

安装 Zabbix 前端（适用于 RHEL 7，在 RHEL 6 上弃用）并使用 MySQL 数据库：

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

若使用 PostgreSQL，则将命令中的'mysql' 替换为'pgsql'；SQLite3 则替换为'sqlite3' （仅 Zabbix proxy）。

创建数据库

对于 Zabbix server 和proxy 守护进程而言，数据库是必须的。而运行 Zabbix agent 是不需要的。

> **Warning:**
> 如果 Zabbix server 和 Zabbix proxy 安装在相同的主机，它们必须创建不同名字的数据库！

使用MySQL 或PostgreSQL 提供的说明来创建数据库。

导入数据

使用 MySQL 来导入 Zabbix server 的初始数据库 schema 和数据，

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

系统将提示您输入新创建的数据库密码。

使用 PostgreSQL：

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

对于 Zabbix proxy，导入初始的数据库 schema：

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

对于使用 PostgreSQL（或 SQLite）的 Zabbix proxy：

```
# zcat /usr/share/doc/zabbix-proxy-pgsql*/schema.sql.gz | sudo -u <username> psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3*/schema.sql.gz | sqlite3 zabbix.db
```

为 Zabbix server/proxy 配置数据库

编辑 zabbix_server.conf 或 zabbix_proxy.conf 文件以使用已创建的数据库。例如：

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

在 DBPassword 参数中输入由 MySQL 或 PosgreSQL 创建的 Zabbix 数据库密码。

在 PostgreSQL 使用 DBHost=。您可能希望保留默认设置 DBHost=localhost（或 IP 地址），但这会使 PostgreSQL 使用网络套接字连接到 Zabbix。有关说明，详见下面的 **SELinux** 配置。

启动 Zabbix server 进程

运行以下命令以启动 Zabbix server 进程：

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

并在系统启动时让它自启：

RHEL 7 或更高版本：

```
# systemctl enable zabbix-server
```

RHEL 7 之前的版本：

```
# chkconfig --level 12345 zabbix-server on
```

使用'zabbix-proxy' 替换命令中的'zabbix-server' 以启动和自启 Zabbix proxy。

Zabbix 前端配置

对于 RHEL 7 和更高版本，Zabbix 前端的 Apache 配置文件位于 /etc/httpd/conf.d/zabbix.conf。

如果使用 RHEL 6，详见在 RHEL 6 上使用 Zabbix 前端 章节来了解如何配置前端。

虽然已经配置了一些 PHP 参数。但是有必要取消"date.timezone" 注释，并为其 设置正确的时区 。

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

以此前端安装步骤 来完成 Zabbix 前端的安装，并访问新安装的 Zabbix 前端页面。

> **Note:**
> Zabbix 官方软件仓库提供了 fping、iksemel、libssh2 包。这些包位于 *non-supported* 目录。

SELinux 配置

在 enforcing 模式下启用 SELinux 状态后，您需要执行以下命令以启用 Zabbix 前端和 Zabbix server 之间的通信：

RHEL 7 或更高版本：

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

如果数据库可以通过网络访问（在 PostgreSQL 情况下包括 'localhost'），您也需要允许 Zabbix 前端连接到数据库：

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

RHEL 7 之前的版本：

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

待前端和 SELinux 配置完成后，需要重新启动 Apache web 服务器:

```
# service httpd restart
```

安装 Agent

运行以下命令以安装 Zabbix agent ：

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

运行以下命令以启动 Zabbix agent：

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

在 RHEL 6 上使用 Zabbix 前端

由于 PHP 版本，不支持 RHEL 6 上的 Zabbix 前端。由于 Zabbix 3.0 的要求是 PHP 5.4.0 或更高版本，而 RHEL 6 最新版本是 5.3.3。

在大多数情况下，Zabbix server 和前端安装在同一台机器上。当从 2.2 升级到 3.0 时，Zabbix server 将执行数据库升级，前端会停止工作。无法回滚数据库更改，因此用户将被迫使用第三方软件包升级 PHP。这就是为什么 Zabbix server 在 RHEL 6 上也被弃用的原因。

如果您仍想在 RHEL 6 上使用 Zabbix 前端并使用第三方软件包升级您的 PHP，则需要首先启用 `zabbix-deprecated` 软件仓库：

- 打开 /etc/yum.repos.d/zabbix.repo 文件
- 找到 [zabbix-deprecated] 部分
- 将其启用 enabled=1
- 保存文件

至此，您将不得不进行更多手动配置。这是因为 Zabbix 无法识别 PHP 所需的 Apache 版本，因此无法为 Zabbix 前端提供正确的 Apache 配置。为此，在 zabbix-web 软件包中包含了 2 个 Apache 配置文件，一个用于 Apache 2.2，另一个用于 2.4，您需要手动与 Apache 配置集成：

- httpd22-example.conf
- httpd24-example.conf

要获取文件的完整路径，请执行：

```
$ rpm -ql zabbix-web | grep example.conf
```

2 Debian/Ubuntu

概述

官方 Zabbix 发行包适用于：

- Debian 9 (Stretch)
- Debian 8 (Jessie)
- Debian 7 (Wheezy)
- Ubuntu 18.04 (Bionic Beaver) LTS
- Ubuntu 16.04 (Xenial Xerus) LTS
- Ubuntu 14.04 (Trusty Tahr) LTS

添加 Zabbix 软件仓库

安装软件仓库配置包，这个包包含了 apt（软件包管理器）的配置文件。

对于 **Debian 9**，运行以下命令：

Note! 对于 Debian 8，将命令中的’stretch’ 替换为’jessie’ 。对于 Debian 7，将命令中’stretch’ 替换为’wheezy’ 。

```
# wget http://repo.zabbix.com/zabbix/3.5/debian/pool/main/z/zabbix-release/zabbix-release_3.5-1+stretch_al
# dpkg -i zabbix-release_3.5-1+stretch_all.deb
# apt update
```

对于 **Ubuntu 18.04 (bionic)**，运行以下命令：

```
# wget http://repo.zabbix.com/zabbix/3.5/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.5-1+bionic_all
# dpkg -i zabbix-release_3.5-1+bionic_all.deb
# apt update
```

- 对于 Ubuntu 16.04，将命令中的'bionic' 替换为'xenial'。
- 对于 Ubuntu 14.04，将命令中的'bionic' 替换为'trusty'。

安装 Server/proxy/前端

安装 Zabbix server 并使用 MySQL 数据库：

```
# apt install zabbix-server-mysql
```

安装 Zabbix proxy 并使用 MySQL 数据库：

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

安装 Zabbix 前端：

```
# apt install zabbix-frontend-php
```

将命令中的'mysql' 替换为'pgsql' 以使用 PostgreSQL，或替换为'sqlite3' 以使用 SQLite3（仅 Zabbix proxy）。

创建数据库

对于 Zabbix server 和 proxy 守护进程而言，数据库是必须的。而运行 Zabbix agent 是不需要的。

> **Warning:**
> 如果 Zabbix server 和 Zabbix proxy 安装在相同的主机，它们必须创建不同名字的数据库！

使用 MySQL 或 PostgreSQL 提供的说明来创建数据库。

导入数据

使用 MySQL 来导入 Zabbix server 的初始数据库 schema 和数据，

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

系统将提示您输入新创建的数据库密码。

使用 PostgreSQL：

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

对于 Zabbix proxy，导入初始的数据库 schema：

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

对于使用 PostgreSQL（或 SQLite）的 Zabbix proxy：

```
# zcat /usr/share/doc/zabbix-proxy-pgsql/schema.sql.gz | sudo -u <username> psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3/schema.sql.gz | sqlite3 zabbix.db
```

为 Zabbix server/proxy 配置数据库

编辑 zabbix_server.conf 或 zabbix_proxy.conf 文件以使用已创建的数据库。例如：

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

在 DBPassword 配置处输入由 MySQL 或 PosgreSQL 创建的 Zabbix 数据库密码。

在 PostgreSQL 使用 DBHost=。您可能希望保留默认设置 DBHost=localhost（或 IP 地址），但这会使 PostgreSQL 使用网络套接字连接到 Zabbix。有关 RHEL/CentOS 的说明，详见下面的SELINUX 配置。

启动 Zabbix server 进程

运行以下命令以启动 Zabbix server 进程，并使其开机自启：

```
# service zabbix-server start
# update-rc.d zabbix-server enable
```

使用'zabbix-proxy' 替换命令中的'zabbix-server' 以启动和自启 Zabibx proxy 进程。

RHEL/CentOS 系统请参考respective section

前端和 SELinux 配置完成后，需要重启 Apache 服务

```
# service apache2 restart
```

Zabbix 前端配置

Zabbix 前端的 Apache 配置文件位于 /etc/apache2/conf-enabled/zabbix.conf。虽然已经配置了一些 PHP 参数。但是有必要取消 "date.timezone" 注释，并为其设置为正确的时区。

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

以此前端安装步骤 来完成 Zabbix 前端的安装，并访问新安装的 Zabbix 前端页面。

安装 Agent

运行以下命令以安装 Zabbix agent ：

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

运行以下命令以启动 Zabbix agent :

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

5 从容器中安装

Docker Zabbix 为每个组件都提供了 Docker 镜像，作为弹性和自给自足的容器，促使加快部署和更新过程。

Zabbix provides Docker images for each Zabbix component as portable and self-sufficient containers to speed up deployment and update procedure.

Zabbix 组件支持 MySQL 和 PostgreSQL 数据库、Apache2 和 Nginx Web 服务器。这些镜像被分成多个不同的镜像。

Zabbix components come with MySQL and PostgreSQL database support, Apache2 and Nginx web server support. These images are separated into different images.

Docker 的基础镜像

Zabbix 组件提供了 Ubuntu、Alpine Linux 和 CentOS 的基础镜像：

Zabbix components are provided on Ubuntu, Alpine Linux and CentOS base images:

镜像版	
alpine	3.4
ubuntu	trusty
centos	latest

如果基础镜像升级了，所有的镜像被配置为重建成最新版本的镜像。

All images are configured to rebuild latest images if base images are updated.

> **Note:**
> Zabbix agent、Zabbix proxy、Zabbix server 和 Zabbix 应用镜像均基于 Alpine Linux 3.4 版，所有其他映像均基于 Alpine Linux 3.7。

Note:
Zabbix agent, Zabbix proxy, Zabbix server and Zabbix appliance images are based on Alpine Linux version 3.4, all other images are based on Alpine Linux 3.7.

Docker 源文件

每个人都可以在 github.com 上使用 Zabbix 官方镜像仓库，并关注其 Docker 文件变更情况。您可以根据官方 Docker 文件复制此项目或制作自己的镜像。

Everyone can follow Docker file changes using the Zabbix official repository on github.com. You can fork the project or make your own images based on official Docker files.

组件

所有 Zabbix 组件都可在以下 Docker 镜像仓库中使用：

All Zabbix components are available in the following Docker repositories:

- MySQL 数据库和 Nginx Web 服务器支持的 Zabbix 应用 - zabbix/zabbix-appliance
- Zabbix appliance with MySQL database support and Nginx web-server - zabbix/zabbix-appliance
- Zabbix agent - zabbix/zabbix-agent
- Zabbix server
  - MySQL 数据库支持的 Zabbix server - zabbix/zabbix-server-mysql
  - PostgreSQL 数据库支持的 Zabbix server - zabbix/zabbix-server-pgsql
- Zabbix server
  - Zabbix server with MySQL database support - zabbix/zabbix-server-mysql
  - Zabbix server with PostgreSQL database support - zabbix/zabbix-server-pgsql
- Zabbix web-interface
  - 基于 Apache2 Web 服务器以及支持 MySQL 数据库的 Zabbix web 接口 - zabbix/zabbix-web-apache-mysql
  - 基于 Apache2 Web 服务器以及支持 PostgreSQL 数据库的 Zabbix web 接口 - zabbix/zabbix-web-apache-pgsql
  - 基于 Nginx Web 服务器以及支持 MySQL 数据库的 Zabbix web 接口 - zabbix/zabbix-web-nginx-mysql
  - 基于 Nginx Web 服务器以及支持 PostgreSQL 数据库的 Zabbix web 接口 - - zabbix/zabbix-web-nginx-pgsql
- Zabbix web-interface
  - Zabbix web-interface based on Apache2 web server with MySQL database support - zabbix/zabbix-web-apache-mysql
  - Zabbix web-interface based on Apache2 web server with PostgreSQL database support - zabbix/zabbix-web-apache-pgsql
  - Zabbix web-interface based on Nginx web server with MySQL database support - zabbix/zabbix-web-nginx-mysql
  - Zabbix web-interface based on Nginx web server with PostgreSQL database support - zabbix/zabbix-web-nginx-pgsql
- Zabbix proxy
  - SQLite3 数据库支持的 Zabbix proxy - zabbix/zabbix-proxy-sqlite3
  - MySQL 数据库支持的 Zabbix proxy - zabbix/zabbix-proxy-mysql
- Zabbix proxy
  - Zabbix proxy with SQLite3 database support - zabbix/zabbix-proxy-sqlite3
  - Zabbix proxy with MySQL database support - zabbix/zabbix-proxy-mysql
- Zabbix Java Gateway - zabbix/zabbix-java-gateway

此外，对于 SNMP trap 的支持，它仅作为基于 Ubuntu Trusty 的额外镜像仓库 (zabbix/zabbix-snmptraps) 提供。它可以与 Zabbix server 和 Zabbix proxy 关联。

Additionally there is SNMP trap support. It is provided as additional repository (zabbix/zabbix-snmptraps) based on Ubuntu Trusty only. It could be linked with Zabbix server and Zabbix proxy.

版本

Zabbix 组件的每个镜像仓库都包含了下列标签：

- `latest` - 基于 Alpine Linux 镜像的最新稳定版的 Zabbix 组件；
- `alpine-latest` - 基于 Alpine Linux 镜像的最新稳定版的 Zabbix 组件；latest stable version of a Zabbix component based on Alpine Linux image
- `ubuntu-latest` - 基于 Ubuntu 镜像的最新稳定版的 Zabbix 组件；latest stable version of a Zabbix component based on Ubuntu image
- `alpine-4.0-latest` - 基于 Alpine Linux 镜像的最新次要版本的 Zabbix 4.0 组件；latest minor version of a Zabbix 4.0 component based on Alpine Linux image
- `ubuntu-4.0-latest` - 基于 Unbuntu 镜像的最新次要版本的 Zabbix 4.0 组件；latest minor version of a Zabbix 4.0 component based on Ubuntu image
- `alpine-4.0.*` - 基于 Alpine Linux 镜像的不同次要版本的 Zabbix 4.0 组件，其中 * 代表 Zabbix 组件的次要版本；
- `ubuntu-4.0.*` - 基于 Ubuntu 镜像的不同次要版本的 Zabbix 4.0 组件，其中 * 代表 Zabbix 组件的次要版本。

Each repository of Zabbix components contains the following tags:

- `latest` - latest stable version of a Zabbix component based on Alpine Linux image
- `alpine-latest` - latest stable version of a Zabbix component based on Alpine Linux image
- `ubuntu-latest` - latest stable version of a Zabbix component based on Ubuntu image
- `alpine-4.0-latest` - latest minor version of a Zabbix 4.0 component based on Alpine Linux image
- `ubuntu-4.0-latest` - latest minor version of a Zabbix 4.0 component based on Ubuntu image
- `alpine-4.0.*` - different minor versions of a Zabbix 4.0 component based on Alpine Linux image, where * is the minor version of Zabbix component
- `ubuntu-4.0.*` - different minor versions of a Zabbix 4.0 component based on Ubuntu image, where * is the minor version of Zabbix component

使用方法

环境变量

所有 Zabbix 组件镜像都提供环境变量来控制配置。这些环境变量在每个组件镜像仓库中列出。这些环境变量是 Zabbix 配置文件中的选项，但具有不同的命名方法。例如，ZBX_LOGSLOWQUERIES 等于来自 Zabbix server 和 Zabbix proxy 配置文件的 LogSlowQueries。

All Zabbix component images provide environment variables to control configuration. These environment variables are listed in each component repository. These environment variables are options from Zabbix configuration files, but with different naming method. For example, ZBX_LOGSLOWQUERIES is equal to LogSlowQueries from Zabbix server and Zabbix proxy configuration files.

Attention:
一些配置选项是不允许更改的。例如，PIDFile 和 LogType。

Attention:
Some of configuration options are not allowed to change. For example, PIDFile and LogType.

其中，一些组件有特定的环境变量，而这些环境变量在官方 Zabbix 配置文件并不存在：

变量 *	组件 ** ** 描	**
DB_SERVER_HOST	Server Proxy Web interface	这个变量指的是 MySQL 或 PostgreSQL 的 IP 或 DNS。 默认情况下，这个值根据 MySQL 和 PostgreSQL，分别为 mysql-server 或 postgres-server
DB_SERVER_PORT	Server Proxy Web interface	这个变量指的是 MySQL 或 PostgreSQL 的端口。 默认情况下，这个值根据 MySQL 和 PostgreSQL，分别为’3306’ 或’5432’ 。
MYSQL_USER	Server Proxy Web-interface	MySQL 数据库用户。 默认情况下，这个值为’zabbix’。
MYSQL_PASSWORD	Server Proxy Web interface	MySQL 数据库密码。 默认情况下，这个值为’zabbix’。
MYSQL_DATABASE	Server Proxy Web interface	Zabbix 数据库库名。 默认情况下，这个值根据 Zabbix server 和 Zabbix proxy，分别为’zabbix’ 和’zabbix_proxy’ 。
POSTGRES_USER	Server Web interface	PostgreSQL 数据库用户。 默认情况下，这个值为’zabbix’。
POSTGRES_PASSWORD	Server Web interface	PostgreSQL 数据库密码。 默认情况下，这个值为’zabbix’。
POSTGRES_DB	Server Web interface	Zabbix 数据库库名。 默认情况下，这个值根据 Zabbix server 和 Zabbix proxy，分别为’zabbix’ 和’zabbix_proxy’ 。
TZ	Web-interface	PHP 时区格式。所有支持的时区列表为 php.net. 默认情况下，这个值为’Europe/Riga’ 。

ZBX_SERVER_NAME	Web interface	Web 界面右上角显示的安装名称。 默认情况下，这个值为'Zabbix Docker'。
ZBX_JAVAGATEWAY_ENABLE	Server Proxy	是否启用 Zabbix Java gateway 以采集与 Java 相关的检查数据。 默认情况下，这个值为"false"。
ZBX_ENABLE_SNMP_TRAPS	Server Proxy	是否启用 SNMP trap feature 功能。这要求 **zabbix-snmptraps** 实例并共享 */var/lib/zabbix/snmptraps* 卷到 Zabbix server 或 proxy。

Some of components have specific environment variables, which do not exist in official Zabbix configuration files:

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

卷

镜像中允许使用一些挂载点。根据 Zabbix 组件类型，这些挂载点各不相同：

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

卷	* 描述 **
Zabbix agent	
/etc/zabbix/zabbix_agentd.d	该卷允许包含 *.conf 文件并使用 UserParameter 功能扩展 Zabbix agent。
/var/lib/zabbix/modules	该卷允许通过LoadModule 功能加载额外的模块以扩展 Zabbix agent。
/var/lib/zabbix/enc	该卷用于存放 TLS 相关的文件。这些文件名指定使用 ZBX_TLSCAFILE、ZBX_TLSCRLFILE、ZBX_TLSKEY_FILE 和 ZBX_TLSPSKFILE 环境变量。
Zabbix server	
/usr/lib/zabbix/alertscripts	该卷用于自定义告警脚本。即zabbix_server.conf 中的 AlertScriptsPath 参数。
/usr/lib/zabbix/externalscripts	该卷用于外部检查。即在zabbix_server.conf 中的 ExternalScripts 参数。
/var/lib/zabbix/modules	该卷允许通过LoadModule 功能加载额外的模块以扩展 Zabbix agent。
/var/lib/zabbix/enc	该卷用于存放 TLS 相关的文件。这些文件名指定使用 ZBX_TLSCAFILE、ZBX_TLSCRLFILE、ZBX_TLSKEY_FILE 和 ZBX_TLSPSKFILE 环境变量。
/var/lib/zabbix/ssl/certs	该卷用于用于存放客户端认证的 SSL 客户端认证文件。即在zabbix_server.conf 中的 SSLCertLocation 参数。
/var/lib/zabbix/ssl/keys	该卷用于存放客户端认证的 SSL 私钥文件。即在zabbix_server.conf 中的 SSLKeyLocation 参数。
/var/lib/zabbix/ssl/ssl_ca	该卷用于存放 SSL 服务器证书认证的证书颁发机构 (CA) 文件。即在zabbix_server.conf 中的 SSLCALocation 参数。
/var/lib/zabbix/snmptraps	该卷用于存放 snmptraps.log 文件。它可由 zabbix-snmptraps 容器共享，并在创建 Zabbix server 新实例时使用 Docker 的 volumes_from 选项继承。可以通过共享卷，并将 ZBX_ENABLE_SNMP_TRAPS 环境变量切换为'true' 以启用 SNMP trap 处理功能。
/var/lib/zabbix/mibs	该卷允许添加新的 MIB 文件。它不支持子目录，所有的 MIB 文件必须位于 /var/lib/zabbix/mibs 下。
Zabbix proxy	
/usr/lib/zabbix/externalscripts	该卷用于使用外部检查。即在zabbix_proxy.conf 中的 ExternalScripts 参数。
/var/lib/zabbix/modules	该卷允许通过LoadModule 功能加载额外的模块以扩展 Zabbix server。
/var/lib/zabbix/enc	该卷用于存放 TLS 相关的文件。这些文件名指定使用 ZBX_TLSCAFILE、ZBX_TLSCRLFILE、ZBX_TLSKEY_FILE 和 ZBX_TLSPSKFILE 环境变量。
/var/lib/zabbix/ssl/certs	该卷用于存放客户端认证的 SSL 客户端认证文件。即在zabbix_proxy.conf 中的 SSLCertLocation 参数。
/var/lib/zabbix/ssl/keys	该卷用于存放客户端认证的 SSL 私钥文件。即在zabbix_proxy.conf 中的 SSLKeyLocation 参数。
/var/lib/zabbix/ssl/ssl_ca	该卷用于存放 SSL 服务器证书认证的证书颁发机构 (CA) 文件。即在zabbix_proxy.conf 中的 SSLCALocation 参数。
/var/lib/zabbix/snmptraps	该卷用于存放 snmptraps.log 文件。它可由 zabbix-snmptraps 容器共享，并在创建 Zabbix server 新实例时使用 Docker 的 volumes_from 选项继承。可以通过共享卷，并将 ZBX_ENABLE_SNMP_TRAPS 环境变量切换为'true' 以启用 SNMP trap 处理功能。
/var/lib/zabbix/mibs	该卷允许添加新的 MIB 文件。它不支持子目录，所有的 MIB 文件必须位于 /var/lib/zabbix/mibs 下。
基于 **Apache2 Web** 服务器的 **Zabbix Web** 接口	
/etc/ssl/apache2	该卷允许为 Zabbix Web 接口启用 HTTPS。该卷必须包含为 Apache2 SSL 连接准备的 ssl.crt 和 ssl.key 两个文件。
基于 **Nginx Web** 服务器的 **Zabbix Web** 接口	
/etc/ssl/nginx	该卷允许为 Zabbix Web 接口启用 HTTPS。该卷必须包含为 Nginx SSL 连接装备的 ssl.crt 和 ssl.key 两个文件。
Zabbix snmptraps	

/var/lib/zabbix/snmptraps	该卷包含了以接收到的 SNMP traps 命名的 `snmptraps.log` 日志文件。
/var/lib/zabbix/mibs	该卷允许添加新的 MIB 文件。它不支持子目录，该 MIB 文件必须位于 `/var/lib/zabbix/mibs` 下。

关于更多的信息请在 Docker Hub 的 Zabbix 官方镜像仓库查看。

For additional information use Zabbix official repositories in Docker Hub.

使用方法实例

** 示例 1 **

该示例示范了如何使用内置 MySQL 数据库、Zabbix server、基于 Nginx Web 服务器的 Zabbix Web 界面和 Zabbix Java gateway 来运行 Zabbix 应用。

The example demonstrates how to run Zabbix appliance with built-in MySQL database, Zabbix server, Zabbix web interface based on the Nginx web server and Zabbix Java gateway.

```
# docker run --name zabbix-appliance -t \
      -p 10051:10051 \
      -p 80:80 \
      -d zabbix/zabbix-appliance:latest
```

Note:
Zabbix 应用实例将 10051/TCP 端口（Zabbix trapper）和 80/TCP 端口（HTTP）暴露给主机。

Note:
Zabbix appliance instance exposes 10051/TCP port (Zabbix trapper) and 80/TCP port (HTTP) to host machine.

** 示例 2 **

该示例示范了如何运行 MySQL 数据库支持、基于 Nginx Web 服务器的 Zabbix Web 界面和 Zabbix Java gateway。

The example demonstrates how to run Zabbix server with MySQL database support, Zabbix web interface based on the Nginx web server and Zabbix Java gateway.

1. 首先，启动空的 MySQL 服务器实例。

1. Start empty MySQL server instance

```
# docker run --name mysql-server -t \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -d mysql:5.7 \
      --character-set-server=utf8 --collation-server=utf8_bin
```

2. 其次，启动 Zabbix Java gateway 实例

2. Start Zabbix Java gateway instance

```
# docker run --name zabbix-java-gateway -t \
      -d zabbix/zabbix-java-gateway:latest
```

3. 然后，启动 Zabbix server 实例，并将其关联到已创建的 MySQL server 实例。

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

```
# docker run --name zabbix-server-mysql -t \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
      --link mysql-server:mysql \
      --link zabbix-java-gateway:zabbix-java-gateway \
```

```
      -p 10051:10051 \
      -d zabbix/zabbix-server-mysql:latest
```

> **Note:**
> Zabbix server 实例将 10051/TCP 端口（Zabbix trapper）暴露给主机。

> **Note:**
> Zabbix server instance exposes 10051/TCP port (Zabbix trapper) to host machine.

4. 最后，启动 Zabbix Web 界面，并将其关联到已创建的 MySQL server 和 Zabbix server 实例。

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

```
# docker run --name zabbix-web-nginx-mysql -t \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      --link mysql-server:mysql \
      --link zabbix-server-mysql:zabbix-server \
      -p 80:80 \
      -d zabbix/zabbix-web-nginx-mysql:latest
```

> **Note:**
> Zabbix web 界面实例将 80/TCP 端口（HTTP）暴露给主机。

> **Note:**
> Zabbix web interface instance exposes 80/TCP port (HTTP) to host machine.

** 示例 3 **

该示例示范了如何运行 PostgreSQL 数据库支持的 Zabbix server、基于 Nginx Web 服务器的 Zabbix Web 界面和 SNMP trap 功能。

The example demonstrates how to run Zabbix server with PostgreSQL database support, Zabbix web interface based on the Nginx web server and SNMP trap feature.

1. 首先，启动空的 PostgreSQL server 实例。

1. Start empty PostgreSQL server instance

```
# docker run --name postgres-server -t \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix" \
      -e POSTGRES_DB="zabbix_pwd" \
      -d postgres:latest
```

2. 其次，启动 Zabbix snmptraps 实例。

2. Start Zabbix snmptraps instance

```
# docker run --name zabbix-snmptraps -t \
      -v /zbx_instance/snmptraps:/var/lib/zabbix/snmptraps:rw \
      -v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
      -p 162:162/udp \
      -d zabbix/zabbix-snmptraps:latest
```

> **Note:**
> Zabbix snmptrap 实例将 162/UDP 端口（SNMP traps）暴露给主机。

> **Note:**
> Zabbix snmptrap instance exposes the 162/UDP port (SNMP traps) to host machine.

3. 然后，启动 Zabbix server 实例，并将其关联到已创建的 PostgreSQL server 实例。

3. Start Zabbix server instance and link the instance with created PostgreSQL server instance

```
# docker run --name zabbix-server-pgsql -t \
      -e DB_SERVER_HOST="postgres-server" \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix" \
      -e POSTGRES_DB="zabbix_pwd" \
      -e ZBX_ENABLE_SNMP_TRAPS="true" \
      --link postgres-server:postgres \
      -p 10051:10051 \
      --volumes-from zabbix-snmptraps \
      -d zabbix/zabbix-server-pgsql:latest
```

Note:
Zabbix server 实例将 10051/TCP 端口（Zabbix trapper）暴露给主机。

Note:
Zabbix server instance exposes the 10051/TCP port (Zabbix trapper) to host machine.

4. 最后，启动 Zabbix Web 界面，并将其关联到已创建的 PostgreSQL server 和 Zabbix server 实例。

4. Start Zabbix web interface and link the instance with created PostgreSQL server and Zabbix server instances

```
# docker run --name zabbix-web-nginx-pgsql -t \
      -e DB_SERVER_HOST="postgres-server" \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix" \
      -e POSTGRES_DB="zabbix_pwd" \
      --link postgres-server:postgres \
      --link zabbix-server-pgsql:zabbix-server \
      -p 443:443 \
      -v /etc/ssl/nginx:/etc/ssl/nginx:ro \
      -d zabbix/zabbix-web-nginx-pgsql:latest
```

Note:
Zabbix web 界面实例将 443/TCP 端口（HTTPS）暴露给主机。
/etc/ssl/nginx 目录必须包含具有所需名称的证书。.

Note:
Zabbix web interface instance exposes the 443/TCP port (HTTPS) to host machine.
Directory */etc/ssl/nginx* must contain certificate with required name.

Docker Compose Zabbix 为 Docker 提供了定义和运行复杂 Zabbix 组件的 compose 文件。这些 compose 文件可以在 github.com: https://github.com/zabbix/zabbix-docker 上的 Zabbix docker 官方镜像仓库中找到。这些 compose 文件作为示例添加，并支持广泛。例如，Zabbix proxy 支持 MySQL 和 SQLite3。

Zabbix provides compose files also for defining and running multi-container Zabbix components in Docker. These compose files are available in Zabbix docker official repository on github.com: https://github.com/zabbix/zabbix-docker. These compose files are added as examples, they are overloaded. For example, they contain proxies with MySQL and SQLite3 support.

以下为几个不同版本的 compose 文件：

There are a few different versions of compose files:

文件名 **	述 **
docker-compose_v3_alpine_mysql_latest.yaml	该 compose 文件运行基于 Alpine Linux 的 Zabbix 4.0 最新版本的组件，支持 MySQL 数据库。
docker-compose_v3_alpine_mysql_local.yaml	该 compose 文件本地构建和运行基于 Alpine Linux 的 Zabbix 4.0 最新版本的组件，支持 MySQL 数据库。
docker-compose_v3_alpine_pgsql_latest.yaml	该 compose 文件运行基于 Alpine Linux 的 Zabbix 4.0 最新版本的组件，支持 PostgreSQL 数据库。
docker-compose_v3_alpine_pgsql_local.yaml	该 compose 文件本地构建和运行基于 Apline Linux 的 Zabbix 4.0 最新版本的组件，支持 PostgreSQL 数据库。

docker-compose_v3_centos_mysql_latest.yaml	该 compose 文件运行基于 CentOS7 的 Zabbix 4.0 最新版本的组件，支持 MySQL 数据库。
docker-compose_v3_centos_mysql_local.yaml	该 compose 文件本地构建和运行基于 CentOS7 的 Zabbix 4.0 最新版本的组件，支持 MySQL 数据库。
docker-compose_v3_centos_pgsql_latest.yaml	该 compose 文件运行基于 CentOS7 的 Zabbix 4.0 最新版本的组件，支持 PostgreSQL 数据库。
docker-compose_v3_centos_pgsql_local.yaml	该 compose 文件本地构建和运行基于 CentOS7 的 Zabbix 4.0 最新版本的组件，支持 PostgreSQL 数据库。
docker-compose_v3_ubuntu_mysql_latest.yaml	该 compose 文件运行基于 Ubuntu 14.04 的 Zabbix 4.0 最新版本的组件，支持 MySQL 数据库。
docker-compose_v3_ubuntu_mysq l_local.yaml	该 compose 文件本地构建和运行基于 Ubuntu 14.04 的 Zabbix 4.0 最新版本的组件，支持 MySQL 数据库。
docker-compose_v3_ubuntu_pgsql_latest.yaml	该 compose 文件运行基于 Ubuntu 14.04 的 Zabbix 4.0 最新版本的组件，支持 PostgreSQL 数据库。
docker-compose_v3_ubuntu_pgsql_local.yaml	该 compose 文件本地构建和运行基于 Ubuntu 14.04 的 Zabbix 4.0 最新版本的组件，支持 PosegreSQL 数据库。

Attention:
Docker compose 文件支持 Docker Compose 2 版本和 Docker Compose 3 版本。

Attention:
Available Docker compose files support both versions 2 and 3 of Docker Compose.

存储

Compose 文件已经配置支持主机上的存储。当你使用 Compose 文件运行 Zabbix 组件时，Docker Compose 将在其所在文件夹中创建一个 zbx_env 目录，该目录将包含于Volumes 章节所述相同的结构，以用于数据库存储。

Compose files are configured to support local storage on a host machine. Docker Compose will create a zbx_env directory in the folder with the compose file when you run Zabbix components using the compose file. The directory will contain the same structure as described above in the Volumes section and directory for database storage.

此外，还有卷 /etc/localtime 和 /etc/timezone 下的文件为只读模式。

There are also volumes in read-only mode for /etc/localtime and /etc/timezone files.

环境变量文件

在 github.com 上与存放 compose 文件的同一目录中，您可以在 compose 文件中找到每个组件的默认环境变量文件，这些环境变量文件的命令与 .env_<type of component> 类似。

In the same directory with compose files on github.com you can find files with default environment variables for each component in compose file. These environment files are named like .env_<type of component>.

示例

** 示例 1 **

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

这个命令将会为每个 Zabbix 组件下载最新的 Zabbix 4.0 镜像，并以 detach 模式运行。

The command will download latest Zabbix 4.0 images for each Zabbix component and run them in detach mode.

Attention:
不要忘记从 github.com 的 Zabbix 官方镜像仓库下载 .env_<type of component> 文件和 compose 文件。

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

** 示例 2 **

```
# docker-compose -f ./docker-compose_v3_ubuntu_mysql_local.yaml up -d
```

这个命令将会下载基于 Ubuntu 14.04 的镜像，并在本地构建 Zabbix 4.0 组件，以 detach 模式运行。

The command will download base image Ubuntu 14.04, then build Zabbix 4.0 components locally and run them in detach mode.

6 升级步骤

概述

本章节提供了关于升级至 Zabbix **4.0** 的信息：

This section provides upgrade information for Zabbix **4.0**:

- 使用二进制包：:
  - 请参阅Red Hat Enterprise Linux/CentOS 的升级步骤
  - 请参阅Debian/Ubuntu 的升级步骤
- 使用源码包，请参阅sources 的升级步骤。

Zabbix **3.4**.x、**3.2**.x、**3.0**.x、**2.4**.x、**2.2**.x 和 **2.0**.x 可以直接升级到 Zabbix 4.0.x。如果要从更早的版本升级，请参阅 Zabbix 2.0 和更早期的文档。

Direct upgrade to Zabbix 4.0.x is possible from Zabbix **3.4**.x, **3.2**.x, **3.0**.x, **2.4**.x, **2.2**.x and **2.0**.x. For upgrading from earlier versions consult Zabbix documentation for 2.0 and earlier.

从二进制包升级

概述

本章节提供使用 Zabbix 的 RPM 和 DEB 二进制包成功升级至 Zabbix 4.0 所需的步骤：

This section provides the steps required for a successful upgrade using official RPM and DEB packages provided by Zabbix for:

- Red Hat Enterprise Linux/CentOS
- Debian/Ubuntu

1 Red Hat Enterprise Linux/CentOS

概述

本章节提供了在 Red Hat Enterprise Linux/CentOS 上使用 Zabbix 二进制包，从 Zabbix **3.4**.x 成功升级至 Zabbix **4.0** 所需的步骤。

This section provides the steps required for a successful upgrade from Zabbix **3.4**.x to Zabbix **4.0**.x using official Zabbix packages for Red Hat Enterprise Linux/CentOS.

虽然升级 Zabbix agent 不是强制性的，但建议将其升级，而 Zabbix server 和 Zabbix proxy 必须具有相同的主要版本。因此，在 Zabbix server 和 Zabbix proxy 的架构升级时，Zabbix server 和所有的 Zabbix proxy 必须停止并升级。

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded.

为了最大限度地减少升级期间的停机时间和数据丢失的几率，建议先停机并升级 Zabbix server，然后再逐个停机并升级 Zabbix proxy，待升级完毕后，先启动 Zabbix proxy，再启动 Zabbix server。在 Zabbix server 停机期间，运行中的 Zabbix Proxy 将继续采集和存储数据，并在 Zabbix server 启动并运行后，将数据传递给 Zabbix server。只有在升级的 Zabbix server 启动后，才会生成 Zabbix server 停机期间出现问题的任何通知。

To minimize downtime and data loss during the upgrade, it is recommended to stop and upgrade Zabbix server and then stop, upgrade and start Zabbix proxies one after another. When all proxies are upgraded, start Zabbix server. During the Zabbix server downtime, running proxies will keep collecting and storing data and will pass the data to Zabbix server when the server is up and running. Any notifications for problems during Zabbix server downtime will be generated only after the upgraded server is started.

请注意，对于 Zabbix proxy 上的 SQLite 数据库，升级前 Zabbix proxy 的历史数据将丢失，因为不支持 SQLite 数据库升级，而且必须手动删除 SQLite 数据库文件。当第一次启动 Zabbix proxy 并且缺少 SQLite 数据库文件时，Zabibx proxy 会自动创建它。

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

根据其数据库大小，数据库升级到 4.0 版本可能需要很长时间。

Depending on database size the database upgrade to version 4.0 may take a long time.

<note warning> 值得注意的是，在升级之前，请务必阅读相关的升级说明！ :::

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

请阅读下面的升级说明：

The following upgrade notes are available:

早期版本详细的	本升级说明版本之间升级的重要说明/	更
3.4.x	升级至 4.0 ’l	bpthread’和’zlib’库现在是必需的。; Support for plain text pro-to-col dropped and header is manda-tory; 不再支持 1.4 版本之前的 Zab-bix agent ; Zabibx proxy 被动模式配置的 Server 参数现在是必需的。

早期版本详细的	本升级说明版本之间升级的重要说明/	更
3.2.x	升级至 3.4 SQ	ite sup- port as back- end database dropped for Zab- bix server/frontend; Perl Com- pat- i- ble Reg- u- lar Ex- pres- sions (PCRE) sup- ported in- stead of POSIX ex- tended; ’libpcre’ and ’libevent’ li- braries manda- tory for Zab- bix server; Exit code checks added for user pa- ram- e- ters, re- mote com- mands and sys- tem.run[] items with- out the ’nowait’

早期版本详细的	本升级说明版本之间升级的重要说明/	更
3.0.x	升级至 3.2 Da	abase up-grade may be slow, de-pend-ing on the his-tory ta-ble size

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

此外，还需检查升级至 Zabbix 4.0 的需求。

You may also want to check the requirements for 4.0.

Note:
建议在升级期间运行两个 SSH 会话，其中一个会话执行升级步骤，另一个会话监控其升级期间的日志。例如，在第二个 SSH 会话上运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，以实时显示最新的常规日志输出或错误日志输出。这对生产环境至关重要。

Note:
It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

升级步骤

1 停止 Zabbix 进程

停止 Zabbix server 以确保没有新数据插入数据库。

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

```
# systemctl stop zabbix-server
```

如果需要升级 Zabbix proxy，那么同样停止 Zabbix proxy 进程。

If upgrading the proxy, stop proxy too.

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

Attention:
众所周知，仅升级 Zabbix server 而不升级 Zabbix proxy ，并且未升级的 Zabibx proxy 将数据发送至升级后的 Zabbix server 是可能的（Zabbix proxy 无法其刷新配置）。但是，不建议这样，Zabbix 不支持这样，选择这样做您需为其独自承担风险。

Attention:
It is known to be possible to upgrade the server only and have older, yet unupgraded proxies report data to a newer server (the proxies can't refresh their configuration though). This approach, however, is not recommended and not supported by Zabbix and choosing it is entirely at your own risk.

2 备份当前的数据库

这是非常重要的步骤。升级前请确保备份了数据库。如果升级失败（因磁盘空间不足、断电或其他意外导致的升级失败），备份的数据库将大有帮助。

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

3 备份配置文件、PHP 文件和 Zabbix 二进制文件

在升级前请确保备份了配置文件、PHP 文件和 Zabbix 二进制文件。

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

配置文件：

```
# mkdir /opt/zabbix-backup/
# cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
# cp /etc/httpd/conf.d/zabbix.conf  /opt/zabbix-backup/
```

PHP 文件和 Zabbix 二进制文件：

```
# cp -R /usr/share/zabbix/ /opt/zabbix-backup/
# cp -R /usr/share/doc/zabbix-* /opt/zabbix-backup/
```

4 升级 Zabbix 软件仓库配置包

在升级之前，必须更新当前的软件仓库包：

To proceed with the upgrade your current repository package has to be updated.

```
# rpm -Uvh http://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-release-4.0-1.el7.centos.noarch.rpm
```

5 升级 Zabbix 组件

运行以下命令以升级 Zabbix 组件：

To upgrade Zabbix components you may run something like:

```
# yum upgrade zabbix-server-mysql zabbix-web-mysql zabbix-agent
```

如果使用 PostgreSQL 数据库，请将命令中的 `mysql` 替换为 `pgsql`。如果升级 Zabbix proxy，请将命令中的 `server` 替换为 `proxy`。

If using PostgreSQL, substitute `mysql` with `pgsql` in the command. If upgrading the proxy, substitute `server` with `proxy` in the command.

6 检查 Zabbix 组件配置文件的参数

在新版本中，Zabbix 组件的配置文件发生了一些变化，关于这些强制变更，详见升级说明。

See the upgrade notes for details on mandatory changes.

7 启动 Zabbix 进程

启动升级后的 Zabbix 组件。

Start the updated Zabbix components.

```
# systemctl start zabbix-server
# systemctl start zabbix-proxy
# systemctl start zabbix-agent
```

8 清除浏览器的 Cookies 和缓存

待升级完毕后，可能需要清除浏览器的 Cookies 和缓存，以便 Zabbix 的 Web 界面能正常工作。

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Zabbix 次要版本之间的升级

如果要升级 Zabbix 的次要版本（例如，从 4.0.1 升级至 4.0.3），是非常容易的：

It is possible to upgrade between minor versions of 4.0.x (for example, from 4.0.1 to 4.0.3). Upgrading between minor versions is easy.

在升级 Zabbix 所有组件的次要版本时，只需运行以下命令：

To execute Zabbix minor version upgrade it is required to run:

```
$ sudo yum upgrade 'zabbix-*'
```

在升级 Zabbix server 的次要版本时，只需运行以下命令：

To execute Zabbix server minor version upgrade run:

```
$ sudo yum upgrade 'zabbix-server-*'
```

在升级 Zabbix agent 的次要版本时，只需运行以下命令：

To execute Zabbix agent minor version upgrade run:

```
$ sudo yum upgrade 'zabbix-agent-*'
```

请注意，您也可以在这些命令中使用'update' 而不是'upgrade'。虽然'upgrade' 会删除过时的包，但'update' 会保留它们。

Note that you may also use 'update' instead of 'upgrade' in these commands. While 'upgrade' will delete obsolete packages, 'update' will preserve them.

2 Debian/Ubuntu

概述

本章节提供了在 Debian/Ubuntu 上使用 Zabbix 二进制包，从 Zabbix 3.4.x 成功升级 至 Zabbix 4.0 所需的步骤。

This section provides the steps required for a successful upgrade from Zabbix **3.4**.x to Zabbix **4.0**.x using official Zabbix packages for Debian/Ubuntu.

虽然升级 Zabbix agent 不是强制性的，但建议将其升级，而 Zabbix server 和 Zabbix proxy 必须具有相同的主要版本。因此，在 Zabbix server 和 Zabbix proxy 的架构升级时，Zabbix server 和所有的 Zabbix proxy 必须停止并升级。

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded.

为了最大限度地减少升级期间的停机时间和数据丢失的几率，建议先停机并升级 Zabbix server，然后再逐个停机并升级 Zabbix proxy，待升级完毕后，先启动 Zabbix proxy，再启动 Zabbix server。在 Zabbix server 停机期间，运行中的 Zabbix Proxy 将继续采集和存储数据，并在 Zabbix server 启动并运行后，将数据传递给 Zabbix server。只有在升级的 Zabbix server 启动后，才会生成 Zabbix server 停机期间出现问题的任何通知。

To minimize downtime and data loss during the upgrade, it is recommended to stop and upgrade Zabbix server and then stop, upgrade and start Zabbix proxies one after another. When all proxies are upgraded, start Zabbix server. During the Zabbix server downtime, running proxies will keep collecting and storing data and will pass the data to Zabbix server when the server is up and running. Any notifications for problems during Zabbix server downtime will be generated only after the upgraded server is started.

请注意，对于 Zabbix proxy 上的 SQLite 数据库，升级前 Zabbix proxy 的历史数据将丢失，因为不支持 SQLite 数据库升级，而且必须手动删除 SQLite 数据库文件。当第一次启动 Zabbix proxy 并且缺少 SQLite 数据库文件时，Zabibx proxy 会自动创建它。

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

根据其数据库大小，数据库升级到 4.0 版本可能需要很长时间。

Depending on database size the database upgrade to version 4.0 may take a long time.

<note warning> 值得注意的是，在升级之前，请务必阅读相关的升级说明！:::

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

请阅读下面的升级说明：

The following upgrade notes are available:

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

此外，还需检查升级至 Zabbix 4.0 的需求。

You may also want to check the requirements for 4.0.

Note:
建议在升级期间运行两个 SSH 会话，其中一个会话执行升级步骤，另一个会话监控其升级期间的日志。例如，在第二个 SSH 会话上运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，以实时显示最新的常规日志输出或错误日志输出。这对生产环境至关重要。

Note:
It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

升级步骤

1 停止 Zabbix 进程

停止 Zabbix server 以确保没有新数据插入数据库。

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

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

如果需要升级 Zabbix proxy，那么同样停止 Zabbix proxy 进程。

If upgrading Zabbix proxy, stop proxy too.

```
# service zabbix-proxy stop
```

Attention:
众所周知，仅升级 Zabbix server 而不升级 Zabbix proxy ，并且未升级的 Zabibx proxy 将数据发送至升级后的 Zabbix server 是可能的（Zabbix proxy 无法其刷新配置）。但是，不建议这样，Zabbix 不支持这样，选择这样做您需为其独自承担风险。

Attention:
It is known to be possible to upgrade the server only and have older, yet unupgraded proxies report data to a newer server (the proxies can't refresh their configuration though). This approach, however, is not recommended and not supported by Zabbix and choosing it is entirely at your own risk.

2 备份当前的数据库

这是非常重要的步骤。升级前请确保备份了数据库。如果升级失败（因磁盘空间不足、断电或其他意外导致的升级失败），备份的数据库将大有帮助。

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

3 备份配置文件、PHP 文件和 Zabbix 二进制文件

在升级前请确保备份了配置文件、PHP 文件和 Zabbix 二进制文件。

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

配置文件：

Configuration files:

```
# mkdir /opt/zabbix-backup/
# cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
# cp /etc/apache2/conf-enabled/zabbix.conf /opt/zabbix-backup/
```

PHP 文件和 Zabbix 二进制文件：

PHP files and Zabbix binaries:

```
# cp -R /usr/share/zabbix/ /opt/zabbix-backup/
# cp -R /usr/share/doc/zabbix-* /opt/zabbix-backup/
```

4 升级 Zabbix 软件仓库配置包

在升级之前，必须卸载当前的软件仓库包：

To proceed with the update your current repository package has to be uninstalled.

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

然后再安装新的软件仓库包：

Then install the new repository configuration package.

在 **Debian 9** 上运行：

```
# wget http://repo.zabbix.com/zabbix/3.5/debian/pool/main/z/zabbix-release/zabbix-release_3.5-1+stretch_al
# dpkg -i zabbix-release_3.5-1+stretch_all.deb
```

在 **Debian 8** 上运行：

```
# wget http://repo.zabbix.com/zabbix/3.5/debian/pool/main/z/zabbix-release/zabbix-release_3.5-1+jessie_all
# dpkg -i zabbix-release_3.5-1+jessie_all.deb
```

在 **Debian 7** 上运行：

```
# wget http://repo.zabbix.com/zabbix/3.5/debian/pool/main/z/zabbix-release/zabbix-release_3.5-1+wheezy_all
# dpkg -i zabbix-release_3.5-1+wheezy_all.deb
```

在 **Ubuntu 18.04** 上运行：

```
# wget http://repo.zabbix.com/zabbix/3.5/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.5-1+bionic_all
# dpkg -i zabbix-release_3.5-1+bionic_all.deb
```

在 **Ubuntu 16.04** 上运行：

```
# wget http://repo.zabbix.com/zabbix/3.5/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.5-1+xenial_all
# dpkg -i zabbix-release_3.5-1+xenial_all.deb
```

在 **Ubuntu 14.04** 上运行：

```
# wget http://repo.zabbix.com/zabbix/3.5/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.5-1+trusty_all
# dpkg -i zabbix-release_3.5-1+trusty_all.deb
```

更新软件仓库信息。

Update the repository information.

```
# apt-get update
```

5 升级 Zabbix 组件

运行以下命令以升级 Zabbix 组件：

To upgrade Zabbix components you may run something like:

```
# apt-get install --only-upgrade zabbix-server-mysql zabbix-frontend-php zabbix-agent
```

如果使用 PostgreSQL 数据库，请将命令中的 `mysql` 替换为 `pgsql`。如果升级 Zabbix proxy，请将命令中的 `server` 替换为 `proxy`。

If using PostgreSQL, substitute `mysql` with `pgsql` in the command. If upgrading the proxy, substitute `server` with `proxy` in the command.

6 检查 Zabbix 组件配置文件的参数

在新版本中，Zabbix 组件的配置文件发生了一些变化，关于这些强制变更，详见升级说明。

See the upgrade notes for details on mandatory changes.

关于新的选项参数，详见Zabbix 4.0.0 新特性 章节。

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

7 启动 Zabbix 进程

启动升级后的 Zabbix 组件。

Start the updated Zabbix components.

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

8 清除浏览器的 Cookies 和缓存

待升级完毕后，可能需要清除浏览器的 Cookies 和缓存，以便 Zabbix 的 Web 界面能正常工作。

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Zabbix 次要版本之间的升级

如果要升级 Zabbix 的次要版本（例如，从 4.0.1 升级至 4.0.3），是非常容易的：

It is possible to upgrade minor versions of 4.0.x (for example, from 4.0.1 to 4.0.3). It is easy.

在升级 Zabbix 所有组件的次要版本时，只需运行以下命令：

To upgrade Zabbix minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix.*'
```

在升级 Zabbix server 的次要版本时，只需运行以下命令：

To upgrade Zabbix server minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix-server.*'
```

在升级 Zabbix agent 的次要版本时，只需运行以下命令：

To upgrade Zabbix agent minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix-agent.*'
```

从源代码包升级

概述

本章节提供了使用 Zabbix 官方源代码包，从 Zabbix 3.4.x 成功升级 至 Zabbix 4.0 所需的步骤。

This section provides the steps required for a successful upgrade from Zabbix **3.4**.x to Zabbix **4.0**.x using official Zabbix sources.

虽然升级 Zabbix agent 不是强制性的，但建议将其升级，而 Zabbix server 和 Zabbix proxy 必须具有相同的主要版本。因此，在 Zabbix server 和 Zabbix proxy 的架构升级时，Zabbix server 和所有的 Zabbix proxy 必须停止并升级。

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded.

为了最大限度地减少升级期间的停机时间和数据丢失的几率，建议先停机并升级 Zabbix server，然后再逐个停机并升级 Zabbix proxy，待升级完毕后，先启动 Zabbix proxy，再启动 Zabbix server。在 Zabbix server 停机期间，运行中的 Zabbix Proxy 将继续采集和存储数据，并在 Zabbix server 启动并运行后，将数据传递给 Zabbix server。只有在升级的 Zabbix server 启动后，才会生成 Zabbix server 停机期间出现问题的任何通知。

To minimize downtime and data loss during the upgrade, it is recommended to stop and upgrade Zabbix server and then stop, upgrade and start Zabbix proxies one after another. When all proxies are upgraded, start Zabbix server. During the Zabbix server downtime, running proxies will keep collecting and storing data and will pass the data to Zabbix server when the server is up and running. Any notifications for problems during Zabbix server downtime will be generated only after the upgraded server is started.

Attention:
众所周知，仅升级 Zabbix server 而不升级 Zabbix proxy ，并且未升级的 Zabibx proxy 将数据发送至升级后的 Zabbix server 是可能的（Zabbix proxy 无法其刷新配置）。但是，不建议这样，Zabbix 不支持这样，选择这样做您需为其独自承担风险。

Attention:
It is known to be possible to start the upgraded server and have older, yet unupgraded proxies report data to a newer server (the proxies can't refresh their configuration though). This approach, however, is not recommended and not supported by Zabbix and choosing it is entirely at your own risk.

请注意，对于 Zabbix proxy 上的 SQLite 数据库，升级前 Zabbix proxy 的历史数据将丢失，因为不支持 SQLite 数据库升级，而且必须手动删除 SQLite 数据库文件。当第一次启动 Zabbix proxy 并且缺少 SQLite 数据库文件时，Zabibx proxy 会自动创建它。

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

根据其数据库大小，数据库升级到 4.0 版本可能需要很长时间。

Depending on database size the database upgrade to version 4.0 may take a long time.

<note warning> 值得注意的是，在升级之前，请务必阅读相关的升级说明！ :::

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

请阅读下面的升级说明：

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Important notes/changes between versions
3.4.x	For 4.0	'libpthread' and 'zlib' libraries now mandatory; Support for plain text protocol dropped and header is mandatory; Pre-1.4 version Zabbix agents are no longer supported; The Server parameter in passive proxy configuration now mandatory
3.2.x	Also for 3.4	SQLite support as backend database dropped for Zabbix server/frontend; Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended; 'libpcre' and 'libevent' libraries mandatory for Zabbix server; Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts; Zabbix Java gateway has to be upgraded to support new functionality
3.0.x	Also for 3.2	Database upgrade may be slow, depending on the history table size
2.4.x	Also for 3.0	Minimum required PHP version upped from 5.3.0 to 5.4.0 LogFile agent parameter must be specified
2.2.x	Also for 2.4	Node-based distributed monitoring removed

Upgrade from	Read full upgrade notes	Important notes/changes between versions
2.0.x	Also for 2.2	Minimum required PHP version upped from 5.1.6 to 5.3.0; Case-sensitive MySQL database required for proper server work; 'mysqli' PHP extension required instead of 'mysql'

此外，还需检查升级至 Zabbix 4.0 的需求。

You may also want to check the requirements for 4.0.

Note:
建议在升级期间运行两个 SSH 会话，其中一个会话执行升级步骤，另一个会话监控其升级期间的日志。例如，在第二个 SSH 会话上运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，以实时显示最新的常规日志输出或错误日志输出。这对生产环境至关重要。

Note:
It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

Zabbix server 升级步骤

1 停止 Zabbix 进程

停止 Zabbix server 以确保没有新数据插入数据库。

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

2 备份当前的数据库

这是非常重要的步骤。升级前请确保备份了数据库。如果升级失败（因磁盘空间不足、断电或其他意外导致的升级失败），备份的数据库将大有帮助。

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

3 备份配置文件、PHP 文件和 Zabbix 二进制文件

在升级前请确保备份了配置文件、PHP 文件和 Zabbix 二进制文件。

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

4 从源代码包安装新的 Zabbix server

使用此说明 从源代码编译 Zabbixserver。

Use these instructions to compile Zabbix server from sources.

5 检查 Zabbix server 配置文件的参数

在新版本中，Zabbix server 的配置文件发生了一些变化，关于这些强制变更，详见升级说明。

See the upgrade notes for details on mandatory changes.

关于新的可选参数，详见Zabbix 4.0 新特征 章节。

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

6 启动新的 Zabbix 进程

启动新的 Zabbix 进程。检查日志文件以查看进程是否成功启动。

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

待 Zabbix server 的进程启动后，它将自动升级数据库。Zabbix server 将会报告当前（强制和可选）的和所需的数据库版本。如果当前的强制版本早于所需的版本，那么 Zabbix server 会自动执行所需数据库的升级修补程序。数据库升级的开始和进度（百分比）将会写入到 Zabbix server 的日志文件中。当升级完成后，会写入一条"database upgrade fully completed" 信息到日志文件中。如果升级失败，Zabbix server 将不会启动。如果当前的强制数据库版本比所需的数据库版本新时，则 Zabbix server 也将无法启动。只有当前强制数据库版本对应于所需的强制版本时，Zabbix server 才会启动。

Zabbix server will automatically upgrade the database. When starting up, Zabbix server reports the current (mandatory and optional) and required database versions. If the current mandatory version is older than the required version, Zabbix server automatically executes the required database upgrade patches. The start and progress level (percentage) of the database upgrade is written to the Zabbix server log file. When the upgrade is completed, a "database upgrade fully completed" message is written to the log file. If any of the upgrade patches fail, Zabbix server will not start. Zabbix server will also not start if the current mandatory database version is newer than the required one. Zabbix server will only start if the current mandatory database version corresponds to the required mandatory version.

```
8673:20161117:104750.259 current database version (mandatory/optional): 03040000/03040000
8673:20161117:104750.259 required mandatory version: 03040000
```

在启动 Zabbix server 之前：

Before you start the server:

- 请确保数据库用户拥有足够的权限（create table, drop table, create index, drop index）；
- Make sure the database user has enough permissions (create table, drop table, create index, drop index)
- 请确保磁盘有足够的空间。
- Make sure you have enough free disk space.

7 安装新的 Zabbix web 界面

其最小的需要为 PHP 5.4.0 版本。如果升级请按照安装说明 进行操作。

The minimum required PHP version is 5.4.0. Update if needed and follow installation instructions.

8 清除浏览器 Cookies 和缓存

待升级完毕后，可能需要清除浏览器的 Cookies 和缓存，以便 Zabbix 的 Web 界面能正常工作。

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Zabbix Proxy 升级步骤

1 停止 Zabbix proxy 进程

停止 Zabbix proxy 进程。

Stop Zabbix proxy.

2 备份配置文件和 Zabbix proxy 二进制文件

在升级前请确保备份了配置文件和 Zabbix proxy 二进制文件。

Make a backup copy of the Zabbix proxy binary and configuration file.

3 从源代码包安装新的 Zabbix proxy

使用此说明 从源代码编译 Zabbix proxy。

Use these instructions to compile Zabbix proxy from sources.

4 检查 Zabbix proxy 配置文件的参数

此版本中没有对 Zabbix proxy 的参数 进行强制的更改。有关新的可选参数，详见Zabbix 4.0.0 新特征章节。

There are no mandatory changes in this version to proxy parameters. For new optional parameters, see the What's new section.

5 启动新的 Zabbix proxy

启动新的 Zabbix proxy。检查日志文件以确定 Zabbix proxy 是否启动成功。

Start the new Zabbix proxy. Check log files to see if the proxy has started successfully.

Zabbix proxy 将自动升级数据库。数据库的升级和启动和Zabbix server 类似。

Zabbix proxy will automatically upgrade the database. Database upgrade takes place similarly as when starting Zabbix server.

Zabbix Agent 升级步骤

Attention:
升级 Zabbix agent 并不是强制性的。如果需要使用新功能时，则可以按需升级 Zabbix agent。

Attention:
Upgrading agents is not mandatory. You only need to upgrade agents if it is required to access the new functionality.

1 停止 Zabbix agent 进程

停止 Zabbix agent 进程。

Stop Zabbix agent.

2 备份配置文件和 Zabbix agent 二进制文件

在升级前请确保备份了配置文件和 Zabbix agent 二进制文件。

Make a backup copy of the Zabbix agent binary and configuration file.

3 从源代码包安装新的 Zabbix agent

使用此说明 从源代码编译 Zabbix agent。

Use these instructions to compile Zabbix agent from sources.

或者，从 Zabbix 下载页面 下载预编译的 Zabbix agent 包。

Alternatively, you may download pre-compiled Zabbix agents from the Zabbix download page.

4 检查 Zabbix agent 配置文件的参数

此版本中没有对 Zabbix agent 的参数进行强制的更改。

There are no mandatory changes in this version to agent parameters.

5 启动新的 Zabbix agent

启动新的 Zabbix agent。检查日志文件以确定 Zabbix agent 是否启动成功。

Start the new Zabbix agent. Check log files to see if the agent has started successfully.

Zabbix 次要版本之间的升级

在 Zabbix 4.0.x 的次要版本之间进行升级时（例如从 4.0.1 升级到 4.0.3 ），需要与为 server/proxy/agent 主要版本之间的升级期间执行相同的操作。唯一的区别是，在次要版本之间进行升级时，不会对数据库进行任何更改。

When upgrading between minor versions of 4.0.x (for example from 4.0.1 to 4.0.3) it is required to execute the same actions for server/proxy/agent as during the upgrade between major versions. The only difference is that when upgrading between minor versions no changes to the database are made.

7 已知问题

全局事件关联

如果第一次和第二次事件之间的时间间隔非常短，即半秒或更短，则事件可能无法正确关联。

Events may not get correlated correctly if the time interval between the first and second event is very small, i.e. half a second and less.

IPMI 检查

在 Debian 9（stretch）之前和 Ubuntu16.04（xenial）之前使用 OpenIPMI 库，IPMI 检查可能无法正常工作。若要解决此问题，需要重新编译 OpenIPMI 库并启用 OpenSSL，详见ZBX-6139。

IPMI checks will not work with the standard OpenIPMI library package on Debian prior to 9 (stretch) and Ubuntu prior to 16.04 (xenial). To fix that, recompile OpenIPMI library with OpenSSL enabled as discussed in ZBX-6139.

SSH 检查

一些 Linux 发行版本如 Debian、Ubuntu，如果使用了安装包安装了 libssh2 类库，则系统将不支持使用密码加密私钥，详见 ZBX-4850 获得更多信息。

Some Linux distributions like Debian, Ubuntu do not support encrypted private keys (with passphrase) if the libssh2 library is installed from packages. Please see ZBX-4850 for more details.

ODBC 检查

由于 upstream bug，如果 Zabbix server 或 proxy 使用 MySQL 作为其数据库，MySQL ODBC 库可能无法使用。有关更多信息和可用的解决办法，详见 ZBX-7665。

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

由于 Microsoft 的 问题。从 Microsoft SQL Server 查询的 XML 数据可能会被截断为 2033 个字符。

XML data queried from Microsoft SQL Server may get truncated to 2033 characters due to a Microsoft issue.

HTTPS 检查

在使用 https 协议的 Web 场景和 HTTP agent 监控项，如果目标服务器配置了禁止 TLS v1.0 或更低版本的协议，Zabbix agent 检查 `net.tcp.service[https...]` 和 `net.tcp.service.perf[https...]` 可能会失败。有关更多信息和可用的解决方法，详见ZBX-9879。

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

Web 监控和 HTTP agent

当"SSL verify peer" 在 Web 场景或 HTTP agent 启用时，由于upstream bug，Zabbix server 可能在 CentOS6、CentOS7 和其他相关 Linux 发行版本上发生内存泄露。有关更多信息和可用的解决方法，详见ZBX-10486。

Zabbix server leaks memory on CentOS 6, CentOS 7 and possibly other related Linux distributions due to an upstream bug when "SSL verify peer" is enabled in web scenarios or HTTP agent. Please see ZBX-10486 for more information and available workarounds.

简单检查

由于早于 v3.10 和 2.1.2 版本的 **fping** 存在一个 BUG，即它错误地处理重复的回放数据包。这可能会使监控项 `icmpping`、`icmppingloss`、`icmppingsec` 导致一些意外的结果。建议使用最新版本的 **fping**。详见 ZBX-11726 获得更多信息。

A bug exists in **fping** versions earlier than v3.10 release 2.1.2 that mishandles duplicate echo replay packets. This may cause unexpected results for `icmpping`, `icmppingloss`, `icmppingsec` items. It is recommended to use the latest version of **fping**. Please see ZBX-11726 for more details.

SNMP 检查

如果使用 OpenBSD 操作系统，并在 Zabbix server 的配置文件中设置了 SourceIP 参数，则在 5.7.3 版本的 Net-SNMP 库中的一个 Use-After-Free（UAF）漏洞可能导致 Zabbix server 崩溃。作为解决方法，请不要设置 SourceIP 参数。同样的问题也适用于 Linux，但它不会导致 Zabbix server 停止工作。应用与 OpenBSD 上 net-snmp 软件包的局部补丁，将会随 OpenBSD 6.3 版本一起发布。

If the OpenBSD operating system is used, a use-after-free bug in the Net-SNMP library up to the 5.7.3 version can cause a crash of Zabbix server if the SourceIP parameter is set in the Zabbix server configuration file. As a workaround, please do not set the SourceIP parameter. The same problem applies also for Linux, but it does not cause Zabbix server to stop working. A local patch for the net-snmp package on OpenBSD was applied and will be released with OpenBSD 6.3.

PHP 7.0 的兼容性问题

已经观察到，使用 PHP 7.0 导入具有 Web 监控触发器的模板，可能会因触发器表达式中的 Web 监控项的双引号错误而导入失败。但将 PHP 升级到 7.1 时，问题就随之消失了。

It has been observed that with PHP 7.0 importing a template with web monitoring triggers may fail due to incorrectly added double quotes to the web monitoring items in the trigger expressions. The issue goes away when upgrading PHP to 7.1.

图表

切换到夏令时（Daylight Saving Time，DST）会导致显示 X 轴标签错误（如日期重复，日期缺失等）。

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

日志文件监控

当文件系统空间为 100% 已满时，如果日志文件仍然在被追加，那么 `log[]` 和 `logrt[]` 监控想会反复从头重新读取日志文件。详见 ZBX-10884 获得更多信息。

`log[]` and `logrt[]` items repeatedly reread log file from the beginning if file system is 100% full and the log file is being appended (see ZBX-10884 for more information).

MySQL 的慢查询

如果监控项的值不存在，那么 Zabbix server 将会生成慢查询（关于 SELECT）。这是由于 MySQL 5.6/5.7 版本中一个已知的问题造成的。解决此问题的办法是在 MySQL 中禁用 index_condition_pushdown 优化器。详见ZBX-10652。

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

API

如果使用 `history.get` 方法，则 **output** 参数将无法正常工作。

The **output** parameter does not work properly with the `history.get` method.

API login

当使用带有 `user.login` 方法 的自定义脚本时，则可以创建大量开放式用户会话，而无需遵循 `user.logout`。

A large number of open user sessions can be created when using custom scripts with the `user.login` method without a following `user.logout`.

8 模板变更

此页面列出了 Zabbix 内置模板的所有变更。根据这些变更，建议对现有模板进行，可以通过导入最新版本或手动执行更改来完成。

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

5. 快速入门

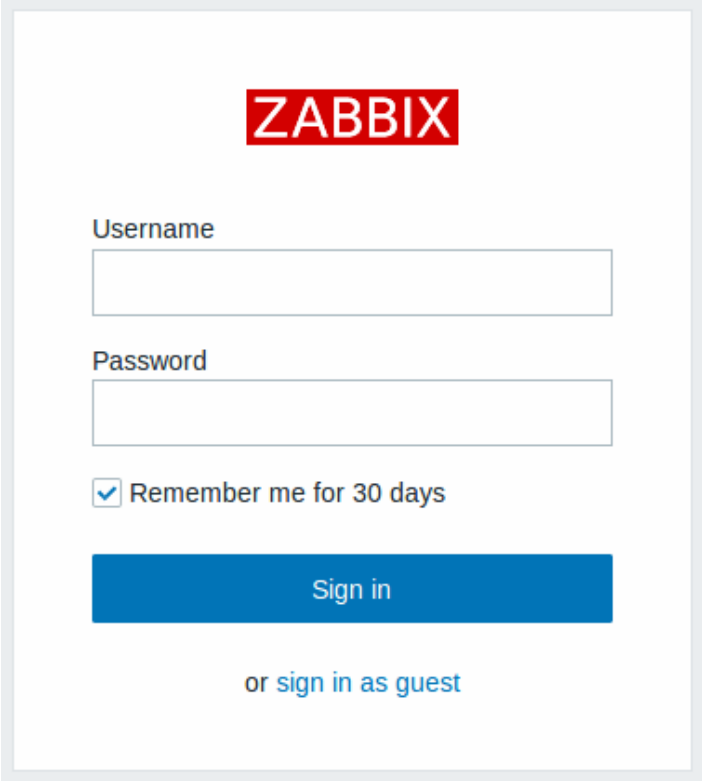
请使用侧边栏访问快速入门部分的内容。

1 登陆和配置用户

简介

本章你会学习如何登陆 Zabbix，以及在 Zabbix 内建立一个系统用户。

登陆



这是 Zabbix 的 “欢迎” 界面。输入用户名 **Admin** 以及密码 **zabbix** 以作为Zabbix 超级用户登陆。

登陆后，你将会在页面右下角看到 “以管理员连接（Connected as Admin）”。同时会获得访问 配置（*Configuration*） and 管理（*Administration*）菜单的权限。

暴力破解攻击的保护机制

为了防止暴力破解和词典攻击，如果发生连续五次尝试登陆失败，Zabbix 界面将暂停 30 秒。

在下次成功登陆后，将会在界面上尝试登录失败的 IP 地址

增加用户

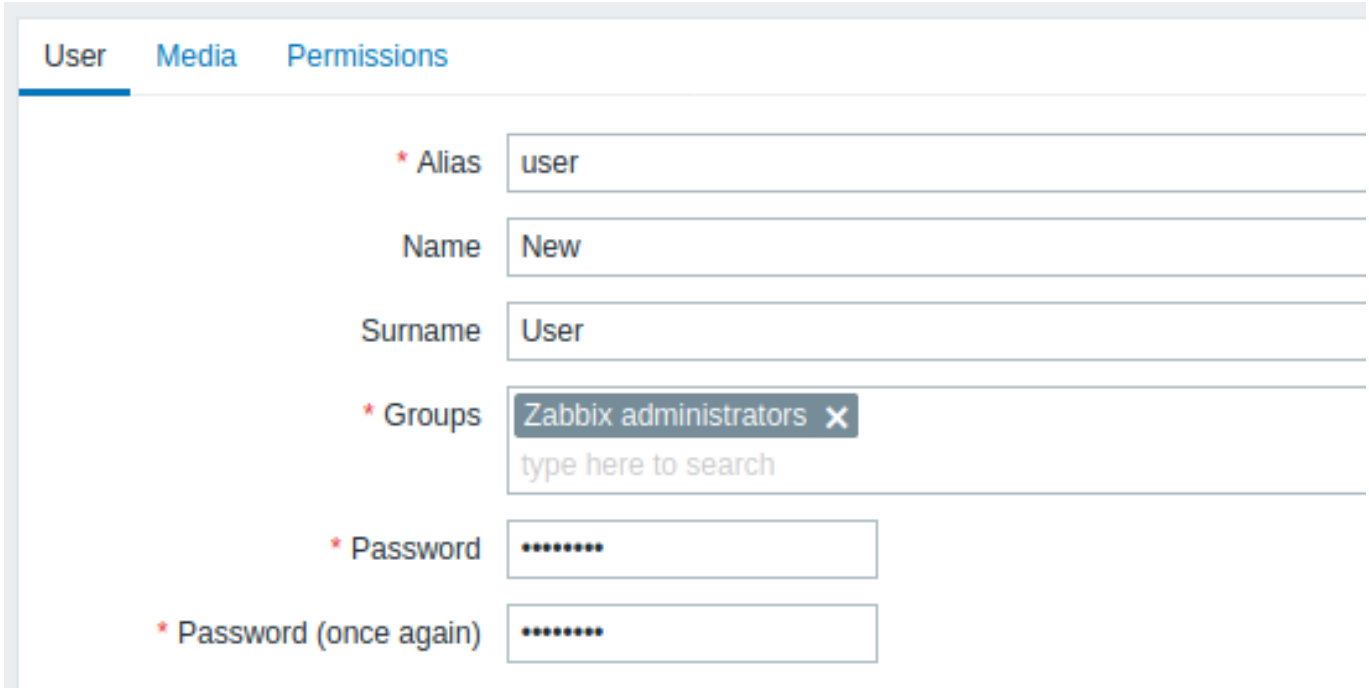
可以在管理（*Administration*）→ 用户（*Users*）下查看用户信息。

Zabbix 在安装后只定义了两个用户。

- 'Admin' 用户是 Zabbix 的一个超级管理员，拥有所有权限。
- 'Guest' 用户是一个特殊的默认用户。如果你没有登陆，你访问 Zabbix 的时候使用的其实是 "guest" 权限。默认情况下，"guest" 用户对 Zabbix 中的对象没有任何权限。

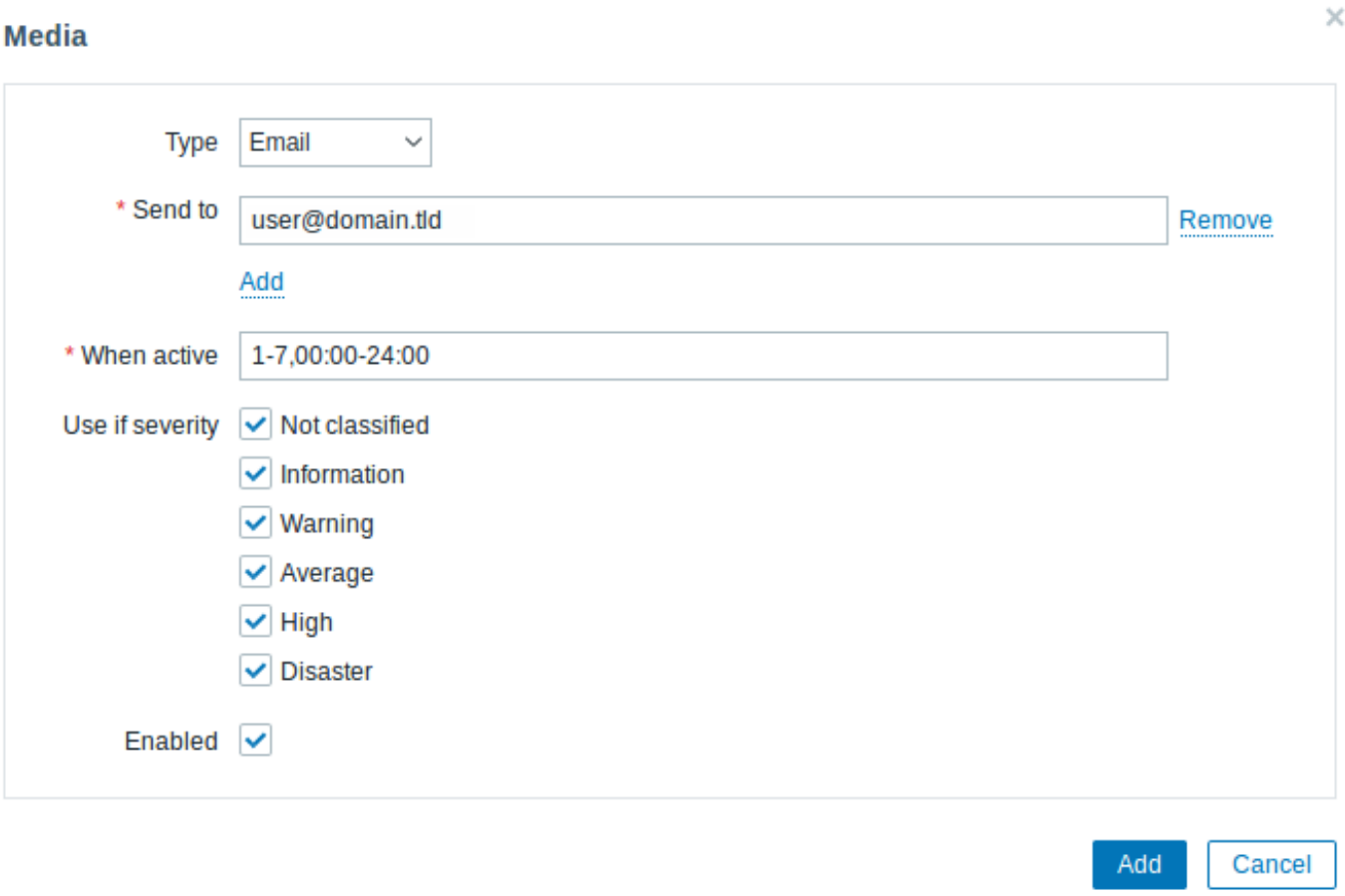
点击 创建用户（*Create user*）以增加用户。

在添加用户的表单中，确认将新增的用户添加到了一个已有的用户组，比如'Zabbix administrators'。



所有必填字端都以红色星标标记。

默认情况下，没有为新增的用户定义媒介（media，即通知发送方式）。如需要创建，可以到' 媒介（Media）' 标签下，然后点击增加（*Add*）。



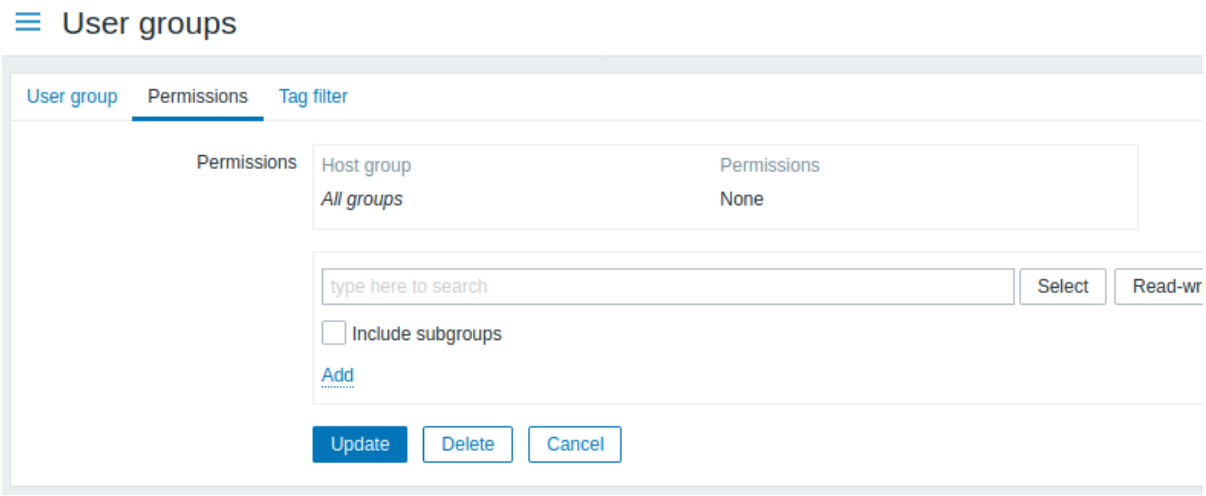
在这个对话框中，为用户输入一个 Email 地址。

你可以为媒介指定一个时间活动周期，(访问时间周期说明页面，查看该字段格式的描述)。默认情况下，媒介一直是活动的。你也可以通过自定义触发器严重等级来激活媒介，默认所有的等级都保持开启。

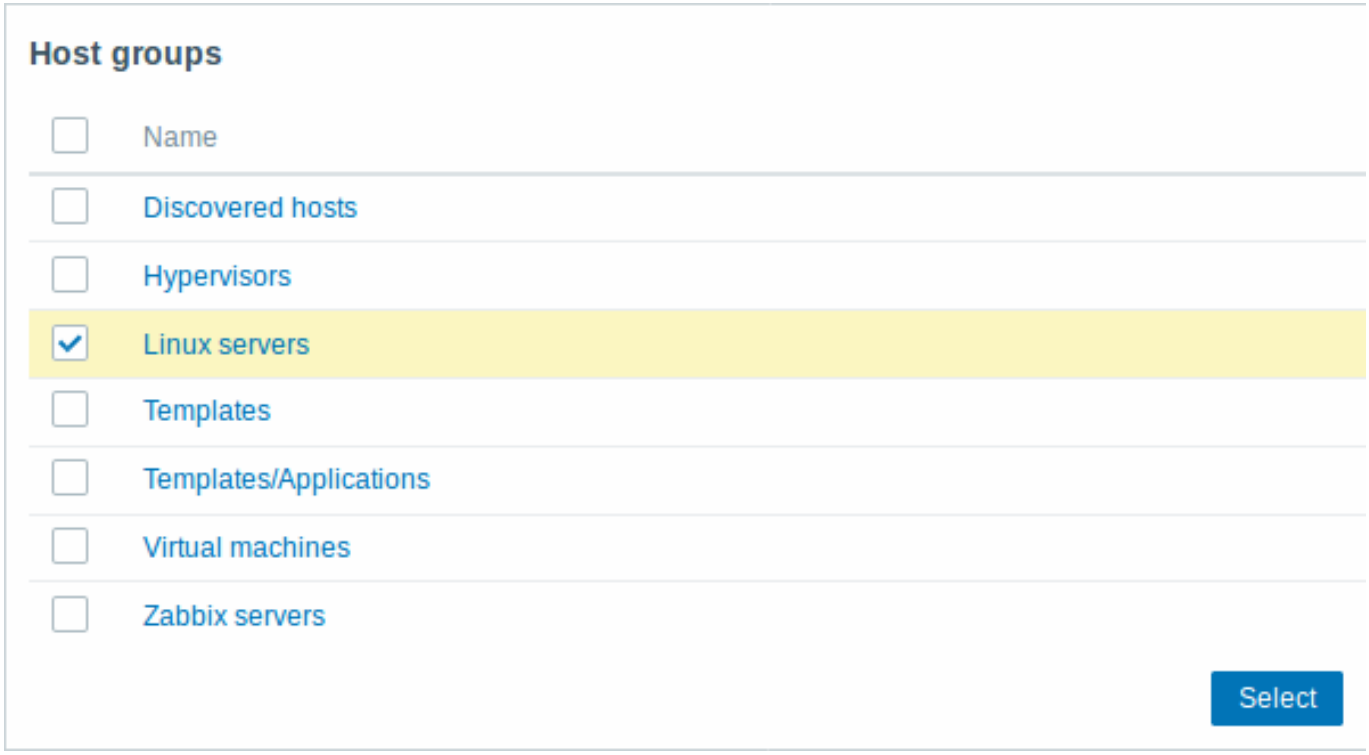
点击新增（*Add*），然后在用户属性表单中点击新增（*Add*）。新的用户将出现在用户清单中。

Adding permissions

By default, a new user has no permissions to access hosts. To grant the user rights, click on the group of the user in the *Groups* column (in this case - 'Zabbix administrators'). In the group properties form, go to the *Permissions* tab.



This user is to have read-only access to *Linux servers* group, so click on *Select* next to the user group selection field.



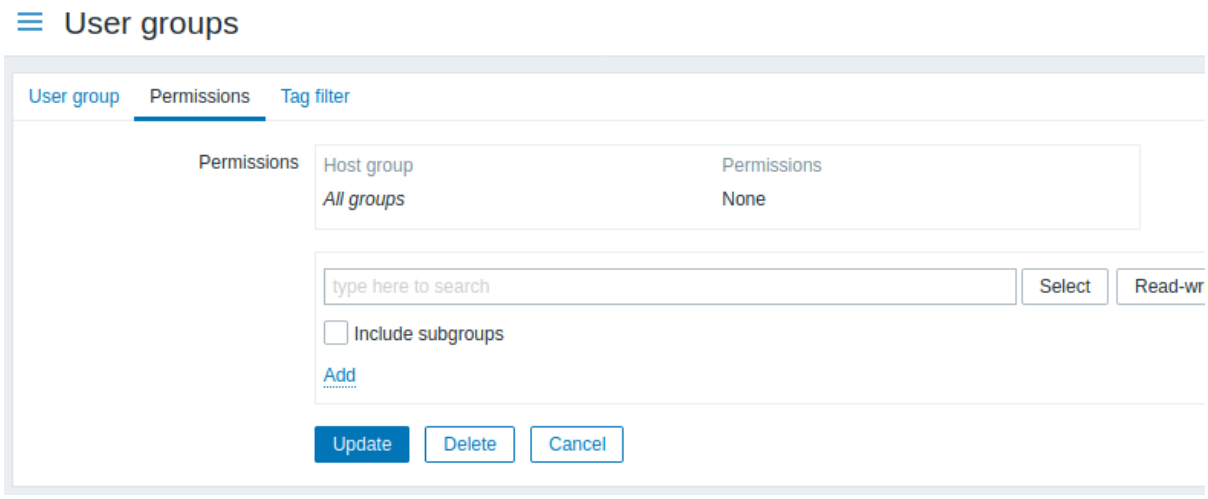
In this pop-up, mark the checkbox next to 'Linux servers', then click *Select*. *Linux servers* should be displayed in the selection field. Click the 'Read' button to set permission level and then *Add* to add the group to the list of permissions. In the user group properties form, click *Update*.

Attention:
In Zabbix, access rights to hosts are assigned to user groups, not individual users.

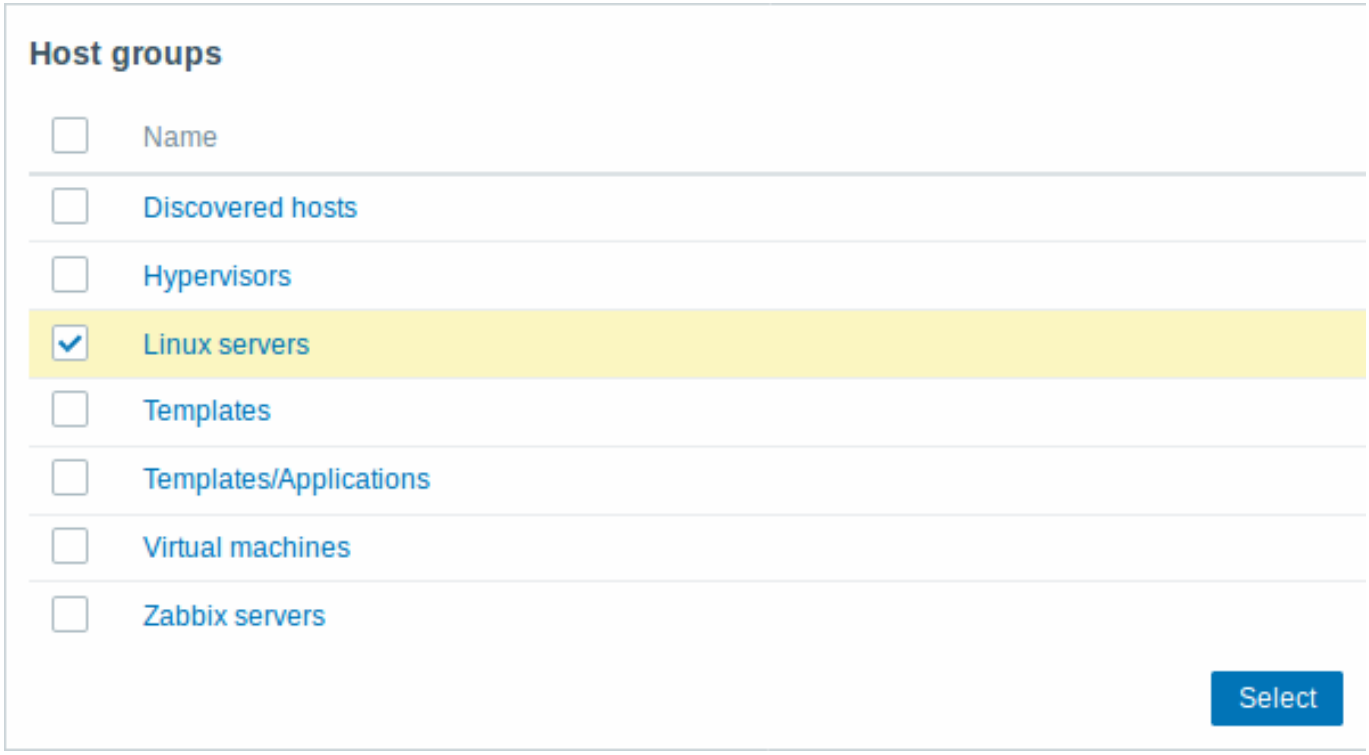
Done! You may try to log in using the credentials of the new user.

Adding permissions

By default, a new user has no permissions to access hosts. To grant the user rights, click on the group of the user in the *Groups* column (in this case - 'Zabbix administrators'). In the group properties form, go to the *Permissions* tab.



This user is to have read-only access to *Linux servers* group, so click on *Select* next to the user group selection field.



In this pop-up, mark the checkbox next to 'Linux servers', then click *Select*. *Linux servers* should be displayed in the selection field. Click the 'Read' button to set permission level and then *Add* to add the group to the list of permissions. In the user group properties form, click *Update*.

Attention:
In Zabbix, access rights to hosts are assigned to user groups, not individual users.

Done! You may try to log in using the credentials of the new user.

2 新建主机

简介

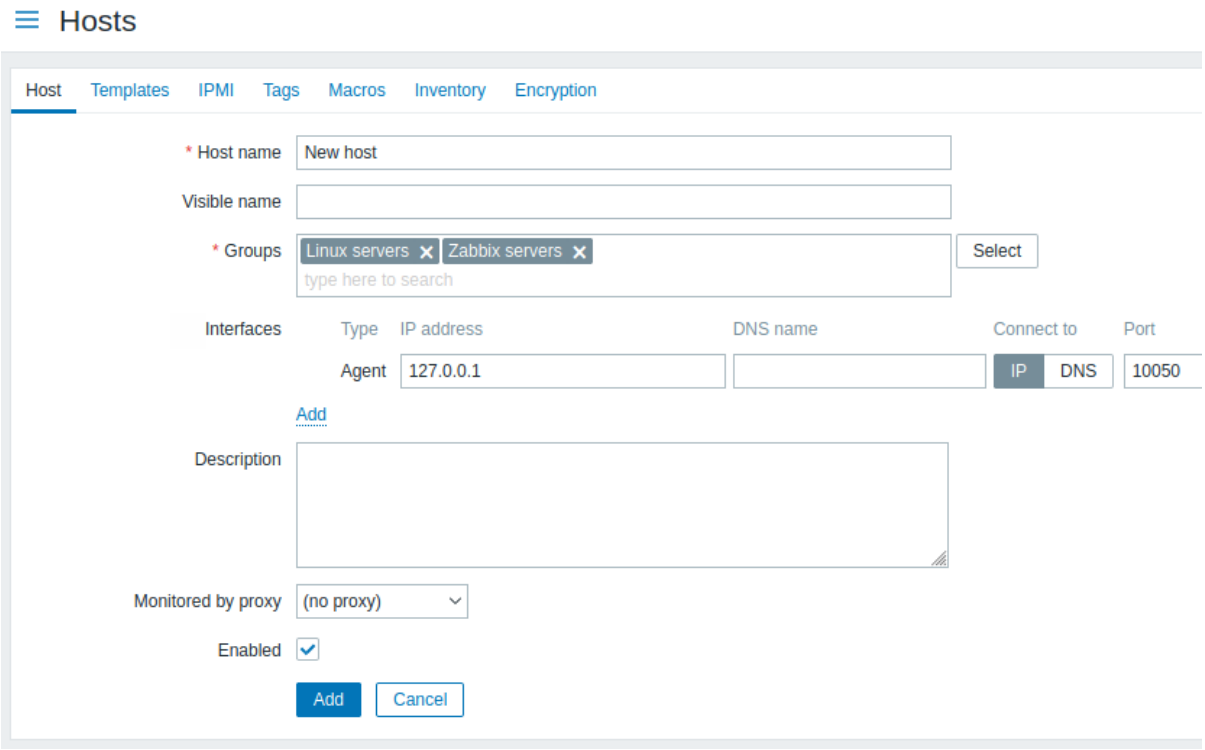
通过本节，你将会学习到如何建立一个新的主机。

Zabbix 中的主机（Host）是一个你想要监控的网络实体（物理的，或者虚拟的）。Zabbix 中，对于主机的定义非常灵活。它可以时一台物理服务器，一个网络交换机，一个虚拟机或者一些应用。

添加主机

Zabbix 中，可以通过配置（*Configuration*）→ 主机（*Hosts*）菜单，查看已配置的主机信息。默认已有一个名为'Zabbix server' 的预先定义好的主机。但我们需要学习如何添加另一个。

点击创建主机（*Create host*）以添加新的主机，这将向我们显示一张主机配置表格。



所有必填字端均以红色星标标示。

至少需要填写下列字段：

主机名称（***Host name***）

- 输入一个主机名称，可以使用字母数字、空格、点”.”、中划线”-”、下划线”_”。

组

- 从右边的选择框中，选择一个或者多个组，然后点击 « 移动它们到’ 所在组（In groups）’ 选择框。

Note:
所有访问权限都分配到主机组，而不是单独的主机。这也是主机需要属于至少一个组的原因。

IP 地址

- 输入主机的 IP 地址。注意如果这是 Zabbix server 的 IP 地址，它必须是 Zabbix agent 配置文件中 ‘Server’ 参数的值。

暂时保持其他选项的默认值。

当完成后，点击添加（*Add*）。你可以在主机列表中看到你新添加的主机。

<note tip> 如果可用性（*Availability*）列中的 *ZBX* 图标是红色的，通信可能存在一些问题。将你的鼠标移动到上面查看错误信息。如果这个图标是灰色的，说明目前状态还没更新。确认 Zabbix server 正在运行，同时尝试过会儿刷新这个页面。:::

3 新建监控项

简介

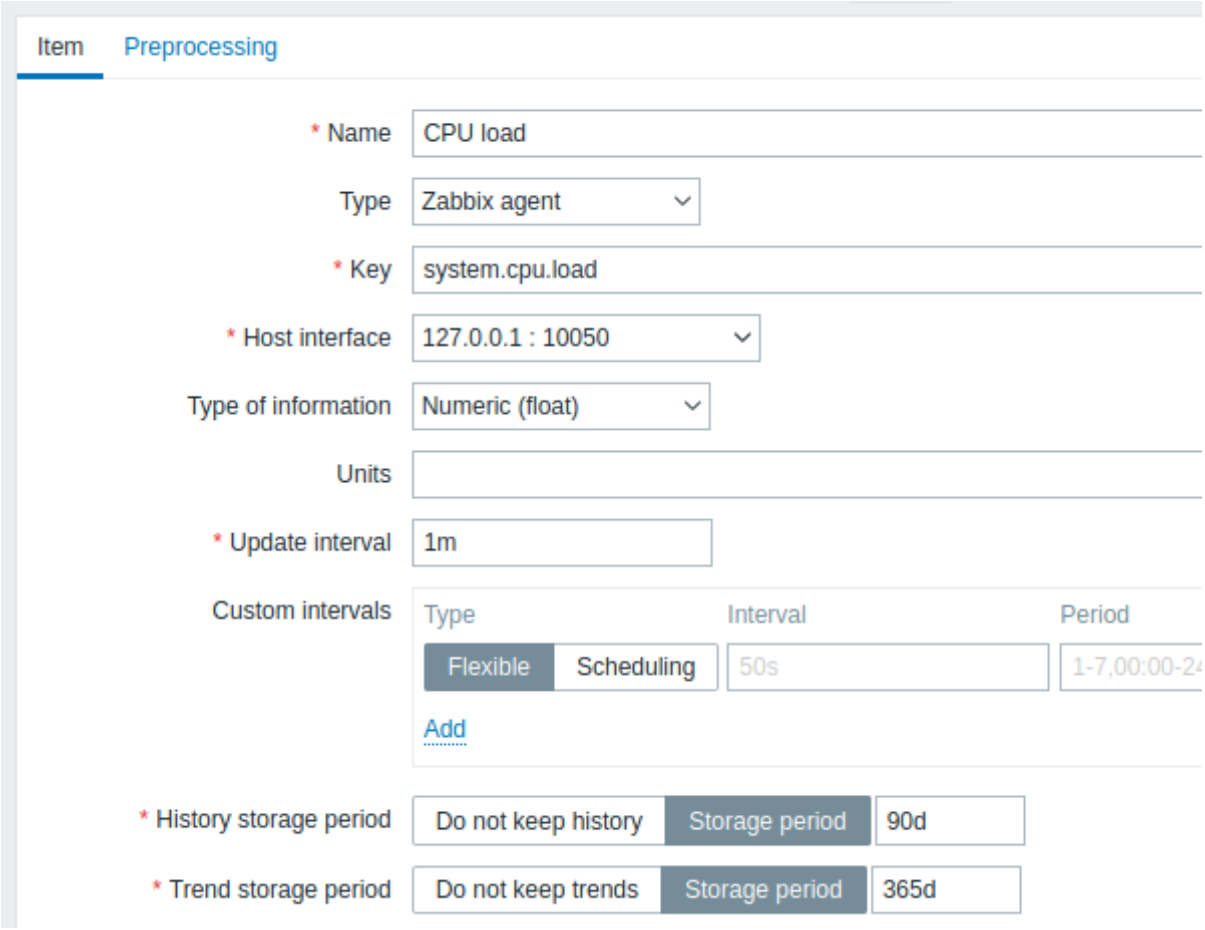
本节你会学习如何新建一个监控项（Item）。

监控项是 Zabbix 中获得数据的基础。没有监控项，就没有数据——因为一个主机中只有监控项定义了单一的指标或者需要获得的数据。

添加监控项

所有的监控项都是依赖于主机的。这就是当我们要配置一个监控项时，先要进入 配置 → 主机页面查找到新建的主机。

在’ 新主机（New host）’ 行中，监控项（*Items*）的链接旁的数量会显示为’0’。点击这个链接，然后点击创建监控项（*Create item*），将会显示一个监控项定义表格。



所有必填项均以红色星标标示。

对于监控项的示例，需要输入以下必要的信息：

名称（***Name***）

- 输入 *CPU Load* 作为值。在列表中和其他地方，都会显示这个值作为监控项名称。

值（***Key***）

- 手动输入 *system.cpu.load* 作为值。这是监控项的一个技术上的名称，用于识别获取信息的类型。这个特定值需要是 Zabbix Agent预定义值中的一种。

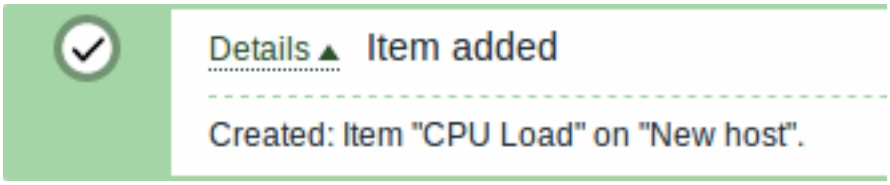
信息类型（***Type of information***）

- 在此处选择 *Numeric (float)*。这个属性定义了想获得数据的格式。

Note:
你也需要减少监控项历史保留的天数，7 或者 14 天。对于数据库而言，最佳实践是避免数据库保留过多的历史数据。

我们暂时保持其他选项的默认值。

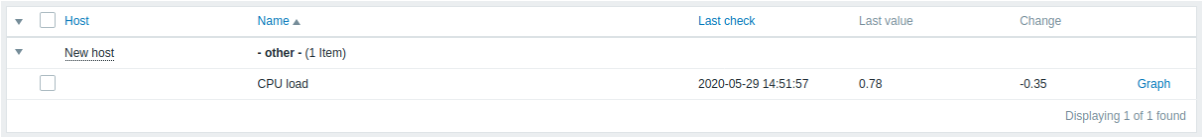
当完成后，点击添加（*Add*）。新的监控项将出现在监控项列表中。点击列表中的详细（*Details*）以查看具体细节。



查看数据

当一个监控项定义完成后，你可能好奇它具体获得了什么值。前往监控（*Monitoring*）→ 最新数据（*Latest data*），在过滤器中选择刚才新建的主机，然后点击应用（*Apply*）。

然后点击**- other -**前面的 **+** ，然后查看你之前定义的监控项和获得的值。



同时，第一次获得的监控项值最多需要 60 秒才能到达。默认情况下，这是服务器读取变化后的配置文件，获取并执行新的监控项的频率。

如果你在 '变化（Change）' 列中没有看到值，可能到目前为止只获得了一次值。等待 30 秒以获得新的监控项值。

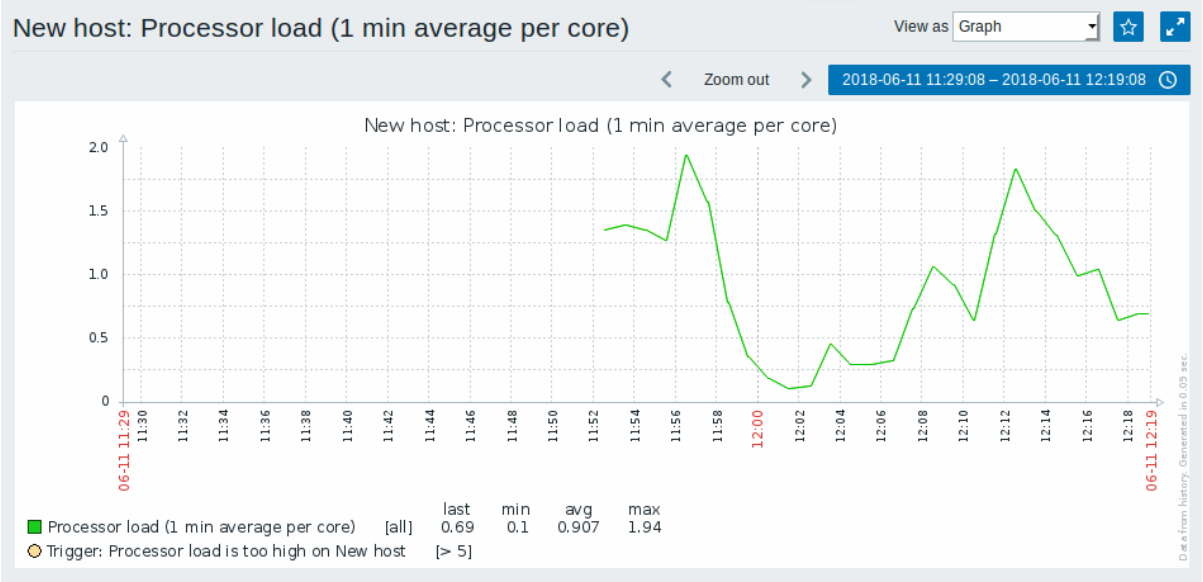
如果你在没有看到类似截图中的监控项信息，请确认：

- 你输入的监控项' 值（Key）' 和' 信息类型（Type of information）' 同截图中的一致
- agent 和 server 都在运行状态
- 主机状态为' 监控（Monitored）' 并且它的可用性图标是绿色的
- 在主机的下拉菜单中已经选择了对应主机，且监控项处于启用状态

图表

当监控项运行了一段时间后，可以查看可视化图表。简单图表 适用于任何被监控的数值型（numeric）监控项，且不需要额外的配置。这些图表会在运行时生成。

前往监控（*Monitoring*）→ 最新数据（*Latest data*），然后点击监控项后的' 图表（Graph）' 链接以查看图表。



4 新建触发器

概述

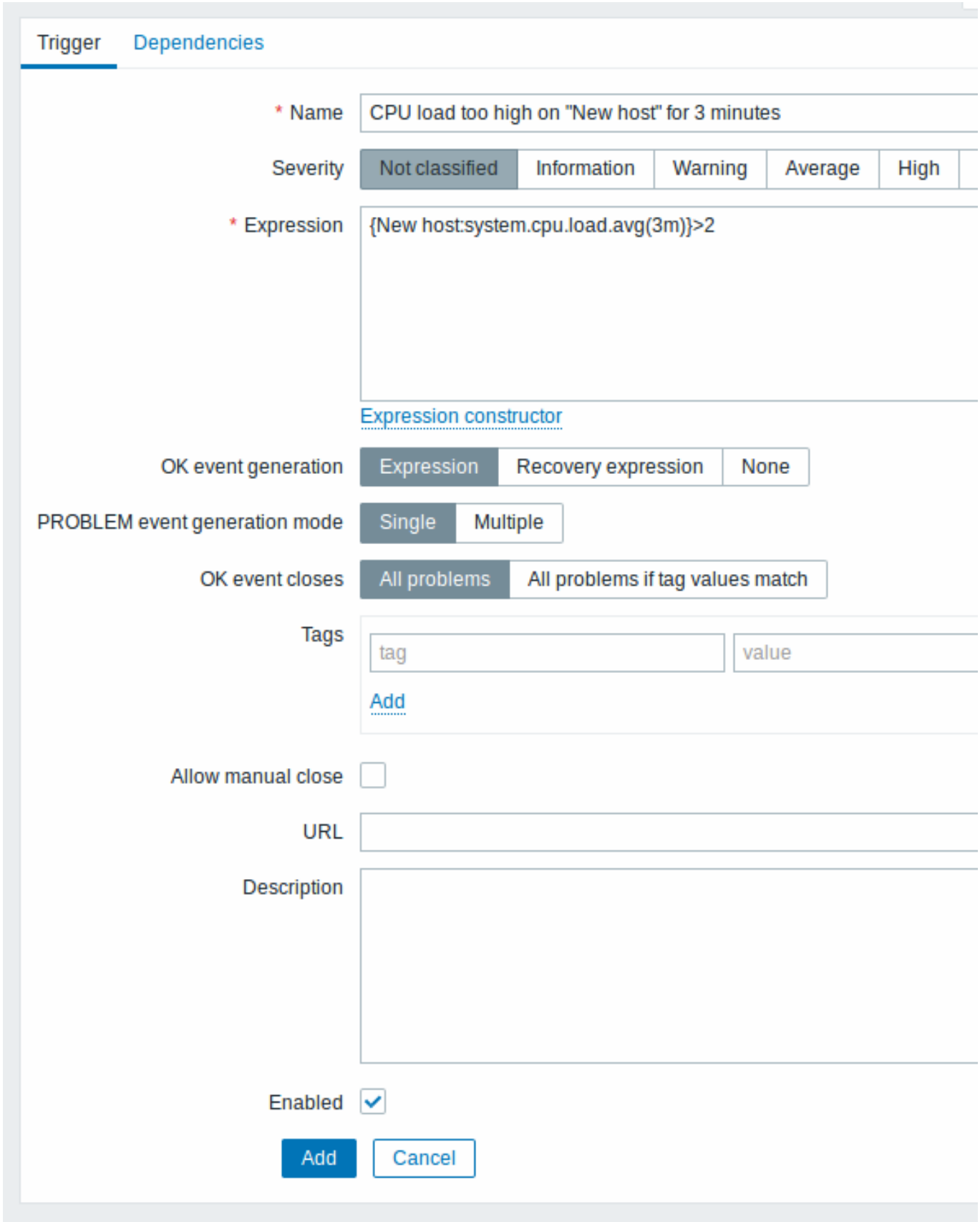
本节你会学习如何配置一个触发器（trigger）。

监控项只是用于收集数据。如果需要自动评估收到的数据，我们则需要定义触发器。触发器包含了一个表达式，这个表达式定义了数据的可接受的阈值级别。

如果收到的数据超过了这个定义好的级别，触发器将被 "触发"，或者进入 "异常（Problem）" 状态——从而引起我们的注意，让我们知道有问题发生。如果数据再次恢复到合理的范围，触发器将会到 "正常（Ok）" 状态。

添加触发器

为监控项配置触发器，前往配置（*Configuration*）→ 主机（*Hosts*），找到' 新增主机（New host）'，点击旁边的触发器（*Triggers*），然后点击创建触发器（*Create trigger*）。这将会向我们展现一个触发器定义表单。



所有必填字端均以红色星标标示。

对于触发器，有下列必填项：

名称（*Name*）

- 输入 *CPU load too high on 'New host' for 3 minutes* 作为值。这个值会作为触发器的名称被现实在列表和其他地方。

表达式（*Expression*）

- 输入：{New host:system.cpu.load.avg(3m)}>2

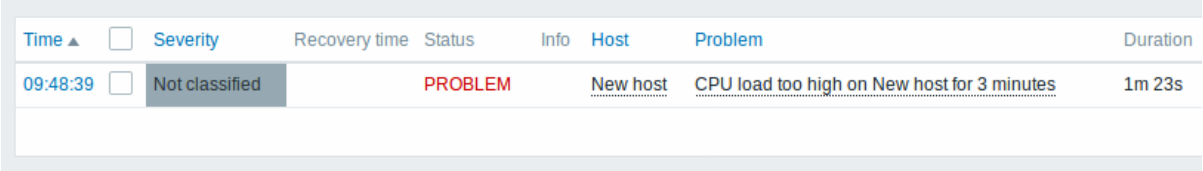
值时触发器的表达式。确认这个表达式输入正确，直到最后一个符号。此处，监控项值 (system.cpu.load) 用于指出具体的监控项。这个特定的表达式大致是说如果 3 分钟内，CPU 负载的平均值超过 2，那么就触发了问题的阈值。你可以查看更多的触发器表达式语法信息。

完成后，点击添加（*Add*）。新的触发器将会显示在触发器列表中。

显示触发器状态

当一个触发器定义完毕后，你可能想查看它的状态。

如果 CPU 负载超过了你在触发器中定义的阈值，这个问题将显示在监控（*Monitoring*）→ 问题（*Problems*）中。



闪烁意味着这个触发器状态最近 30 分钟内发生过变化。

5 获取问题通知

简介

在本节中，你会学习如何在 Zabbix 中以通知（notifications）的方式配置报警（alerting）。

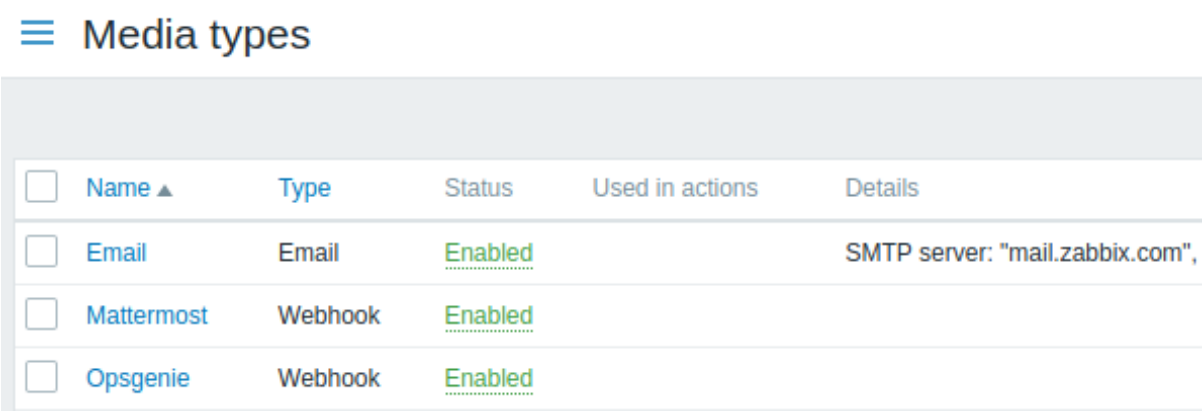
当监控项收集了数据后，触发器会根据异常状态触发报警。根据一些报警机制，它也会通知我们一些重要的事件，而不需要我们直接在 Zabbix 前端进行查看。

这就是通知（Notifications）的功能。E-mail 是最常用的异常通知发送方式。我们将会学习如何配置 e-mail 通知。

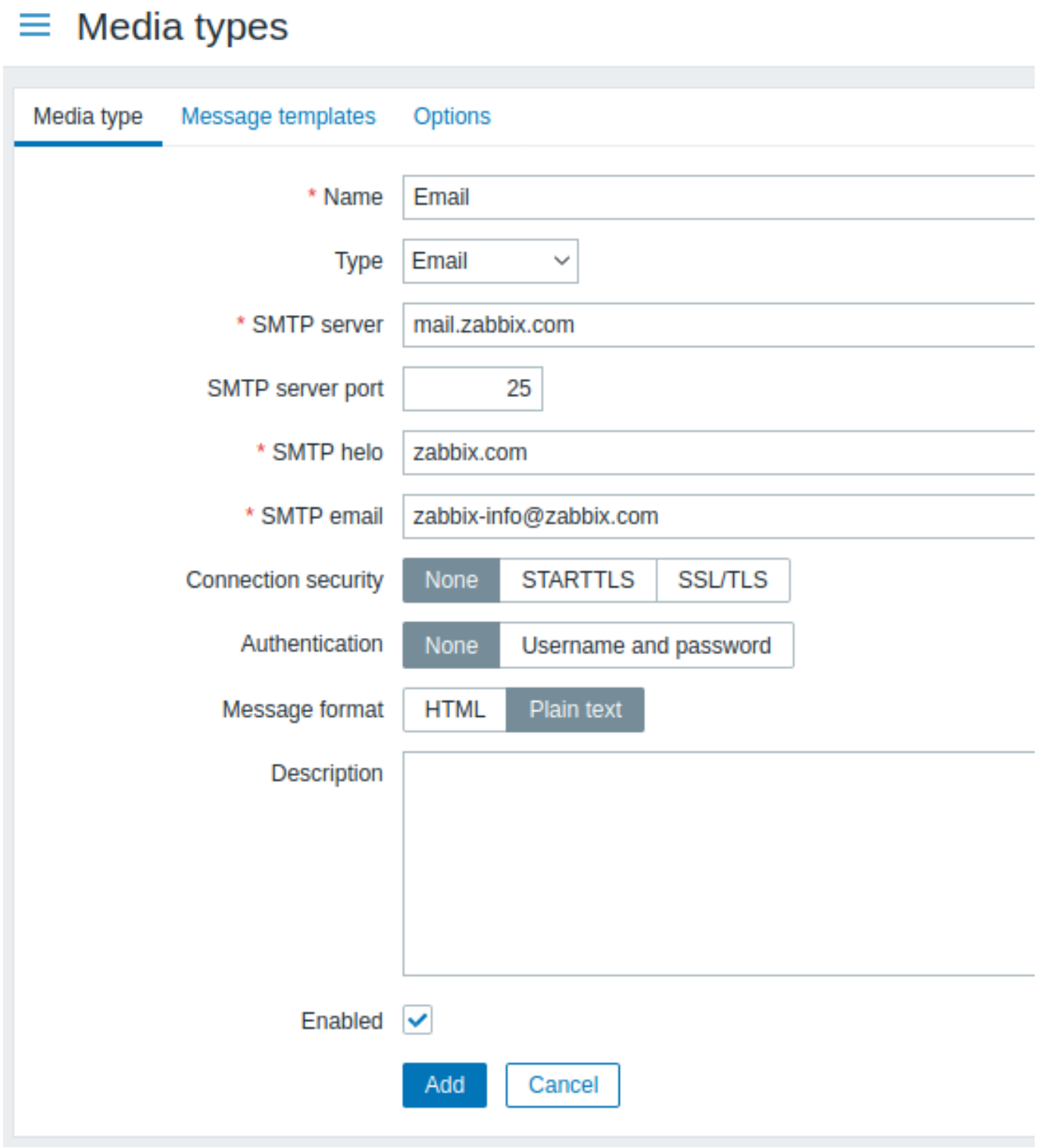
E-mail 设置

Zabbix 中最初内置了一些预定义的通知发送方式。E-mail 通知是其中的一种。

前往管理（*Administration*）→ 媒体类型（*Media types*），点击预定义媒体类型列表中的 *Email*，以配置 E-mail。



这将向我们展现 e-mail 设置定义表单。



所有必填字段均以红色星标标示。

根据你的环境，设置 SMTP 服务器，SMTP helo，SMTP e-mail 的值。

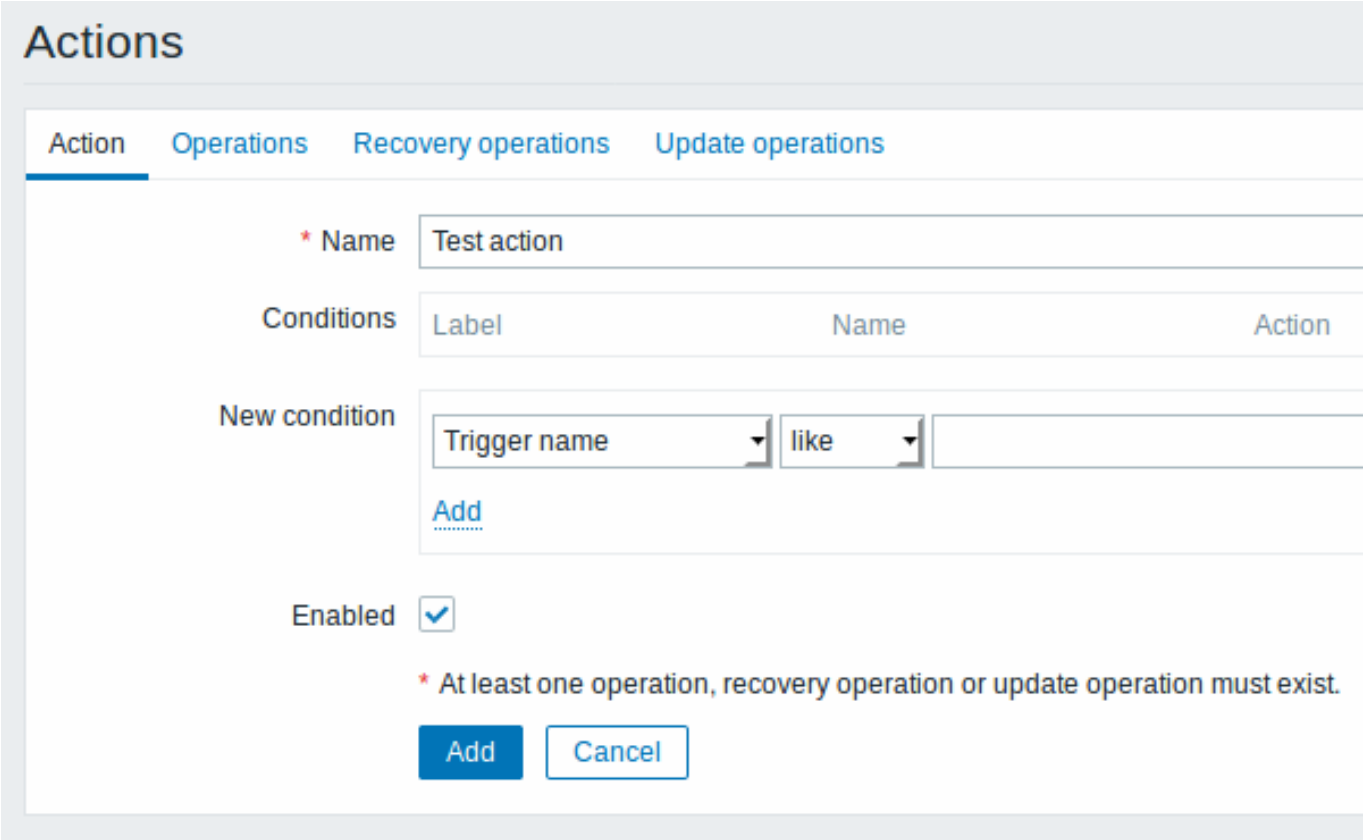
Note:
'SMTP email' 将作为 Zabbix 通知的' 发件人（From）' 地址。

一切就绪后，点击 更新（*Update*）。

现在你已经配置了'Email' 作为一种可用的媒体类型。一个媒体类型必须通过发送地址来关联用户 (如同我们在配置一个新用户中做的)，否则它将无法生效。

新建动作

发送通知是 Zabbix 中动作（actions）执行的操作之一。因此，为了建立一个通知，前往配置（*Configuration*）→ 动作（*Actions*），然后点击创建动作（*Create action*）。

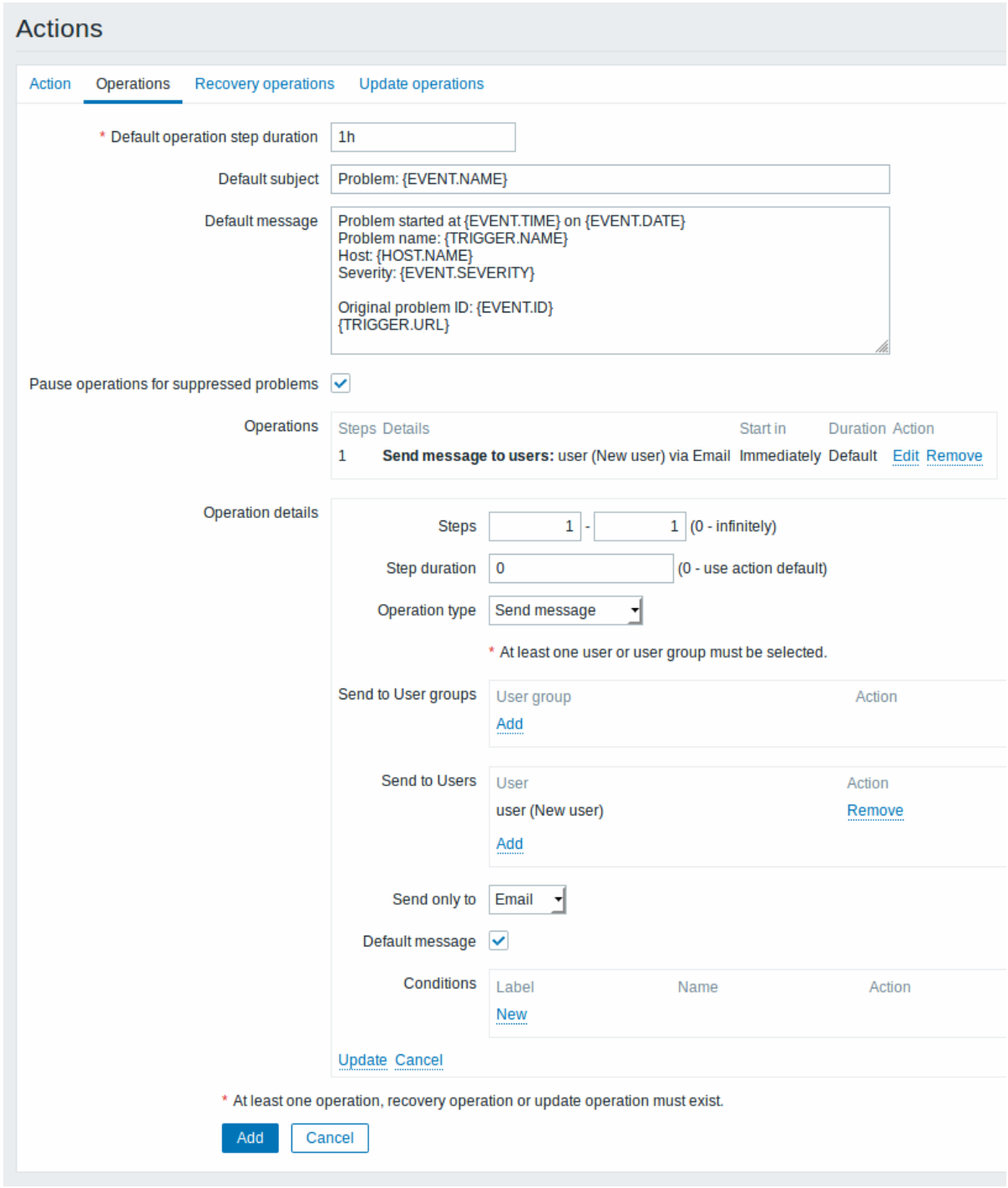


所有必填字段均以红色星标标示。

在这个表单中，输入这个动作的名称。

在大多数简单的例子中，如果我们不添加更多的指定条件，这个动作会在触发器从'Ok' 变为'Problem' 时发生。

我们还需要定义这个动作具体做了什么 —— 即在 操作（*Operations*）标签页中执行的操作。点击新建（*New*），将会打开一个操作表单。



所有必填字段均以红色星标标示。

这里，在发送给用户（*Send to Users*）块中点击添加（*Add*），然后选择我们之前定义的用户（'user'）。选择'Email' 作为 *Send only to* 的值。完成后，在操作明细区域中，点击添加（*Add*）。

在默认主题（*Default subject*）和 默认消息（*Default message*）字段可看到 {TRIGGER.STATUS} 和 {TRIGGER.NAME} 宏（或者变量），它们会被具体的触发器状态和触发器名称替换。

这是一个简单的动作配置步骤，即点击动作表单中的添加（*Add*）。

获得通知

现在，发送通知配置完成，我们看看它如何将通知发送给实际接收人。为了实现这个目的，我们需要你主机的负载，这样我们的触发器才会被触发，我们会收到问题通知。

打开主机的控制台，并运行：

```
cat /dev/urandom | md5sum
```

你需要运行一个或者多个这样的进程。

现在，前往监控（*Monitoring*）→ 最新数据（*Latest data*），查看'CPU Load' 的值是否已经增长。记住，为了使我们的触发器触发（*fire*），

'CPU Load' 的值需要在在 3 分钟运行的过程中超过 2。一旦满足这个条件：

- 在监控（*Monitoring*）→ 问题（*Problems*）中，你可以看到闪烁 'Problem' 状态的触发器。
- 你的 e-mail 中，会收到一个问题通知

Attention:
如果通知功能没有正常工作：

- 再次验证 e-mail 设置和动作设置已经被正确配置
- 确认你创建的用户对生成事件的主机至少拥有读（read）权限。正如添加用户步骤中提到的，'Zabbix administrators' 用户组中的用户必须对'Linux servers' 主机组（该主机所属组）至少拥有读（read）权限。
- 另外，你可以在报告（*Reports*）→ 动作日志（*Action log*）中检查动作日志。

6 新建模版

概述

在本节中，你将会学习如何配置一个模版。

我们在之前的章节中学会了如何配置监控项、触发器，以及如果从主机上获得问题的通知。

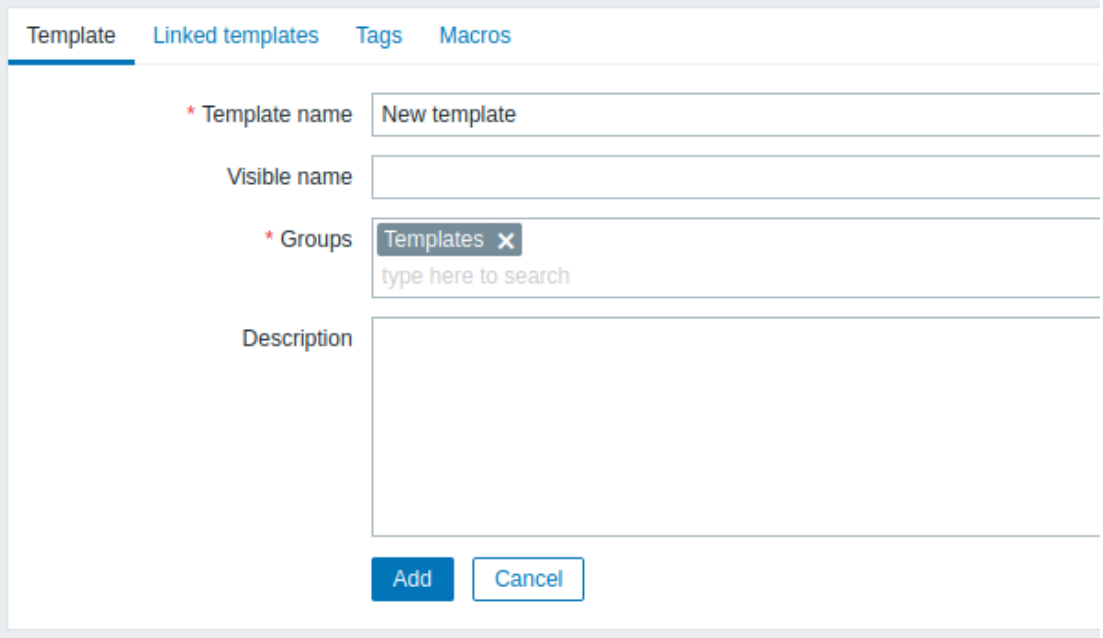
虽然这些步骤提供了很大的灵活性，但仍然需要很多步骤才能完成。如果我们需要配置上千台主机，一些自动化操作会带来更多便利性。

模版（templates）功能可以实现这一点。模版允许对有用的监控项、触发器和其他对象进行分组，只需要一步就可以对监控主机应用模版，以达到反复重用的目的。

当一个模版链接到一个主机后，主机会继承这个模版中的所有对象。简单而言，一组预先定义好的检查会被快速应用到主机上。

添加模版

开始使用模版，你必须先创建一个。在配置（*Configuration*）→ 模版（*Templates*）中，点击创建模版（*Create template*）。这将会像我们展现一个模版配置表格。



所有必填字段以红色星标标示。

需要输入以下必填字段：

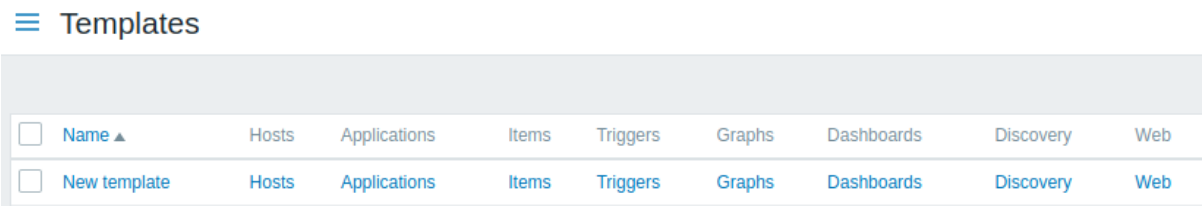
模版名称（***Template name***）

- 输入一个模版名称。可以使用数字、字母、空格及下划线。

组（***Groups***）

- 使用选择（*Select*）按钮选择一个或者多个组。模版必须属于一个组。

完成后，点击添加（*Add*）。你新建的模版可以在模版列表中查看。



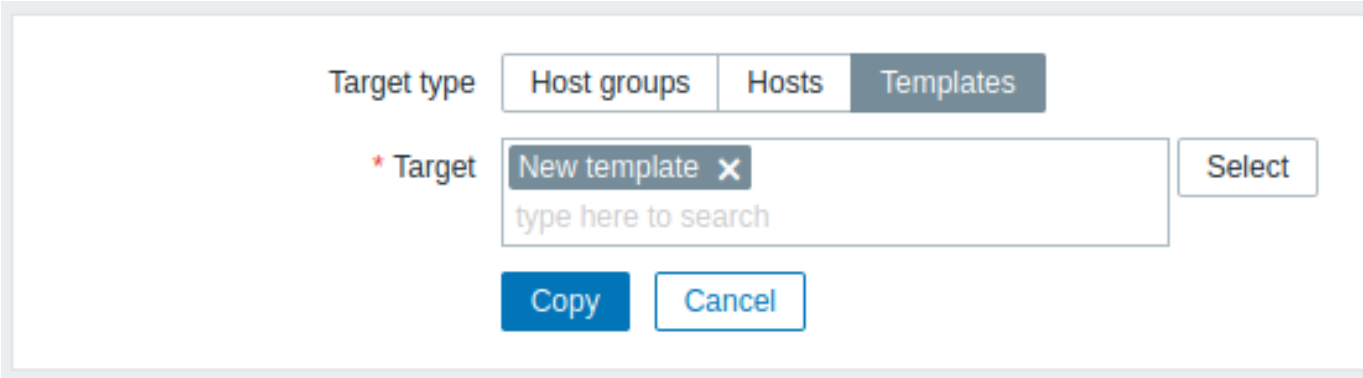
你可以在这看到模版信息。但这个模版中没有任何信息——没有监控项、触发器或者其他对象。

在模版中添加监控项

为了在模版中添加监控项，前往'New host' 的监控项列表。在配置（*Configuration*）→ 主机（*Hosts*），点击 'New host' 旁边的监控项（*Items*）。

然后：

- 选中列表中'CPU Load' 监控项的选择框
- 点击列表下方的复制（*Copy*）
- 选择想要复制这个监控项的目标模版



所有必填字段以红色星标标示。

- 点击复制（*Copy*）

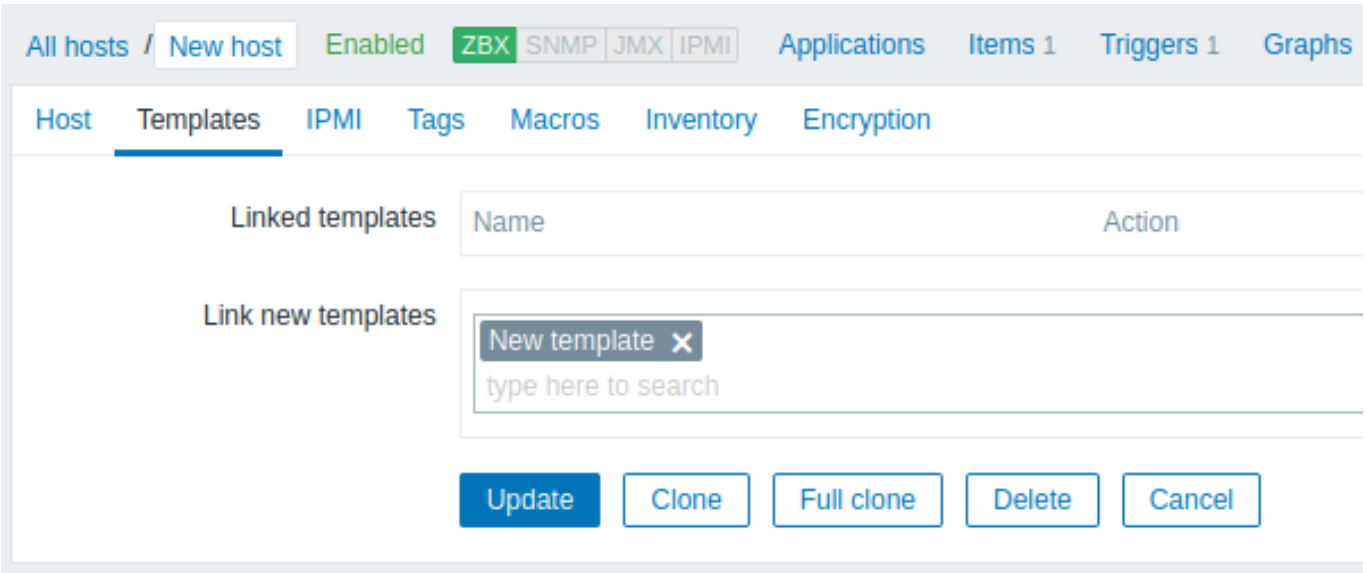
你现在可以前往配置（*Configuration*）→ 模版（*Templates*），' 新模版（New template）' 中会有一个新的监控项。

我们目前至创建了一个监控项，但你可以用同样的方法在模版中添加其他的监控项，触发器以及其他对象，直到完成满足特定需求（如监控 OS，监控单个应用）的完整的对象组合。

链接模版到主机

准备一个模版后，将它链接到一个主机。前往配置（*Configuration*）→ 主机（*Hosts*），点击' 新主机（New host）' 打开表单，前往模版（**Templates**）标签页。

点击链接新模版（*Link new templates*）旁边的选择（*Select*），在弹出的窗口中，点击我们创建模版的名称 ('New template')，它会出现在链接新模版（*Link new templates*）区域，点击添加（*Add*）。这个模版会出现在已链接模版（*Linked templates*）列表中。



点击更新（*Update*）保存配置。现在，新模版及其所有的对象被添加到了主机。

你可能会想到，我们可以使用同样的方法将模版应用到其他主机。任何在模版级别的监控项、触发器及其他对象的变更，也会传递给所有链接该模版的主机。

链接预定义模版到主机

你可能注意到，Zabbix 为各种操作系统、设备以及应用准备一些预定义的模版。为了快速部署监控，你可能会将它们中的一些与主机关联。但请注意，一些模版需要根据你的实际环境进行合适的调整。比如：一些检查项是不需要的，一些轮询周期过于频繁。

可参考该链接，查看更多关于模版的信息。

6. Zabbix 应用

简介　除了手动安装或者重新使用现有的服务器来运行 Zabbix 外，用户可通过download下载 Zabbix 应用或者包含 Zabbix 应用的光盘镜像。Zabbix server (MySQL), Zabbix server (PostgreSQL), Zabbix proxy (MySQL) 以及 Zabbix proxy (SQLite 3) 可使用 Zabbix 应用光盘，进行即时部署。

Zabbix 应用虚拟机已备好 MySQL 数据库以支持 Zabbix server，且是通过使用 Zabbix 应用安装光盘构建而成。

|<| |<| |-|

|<| |<| |-|

Zabbix 应用和安装 CD 版本均基于下列 Ubuntu 版本：

Zabbix 应用版本 Ubu	tu 版本
3.4.0	14.04.3

Zabbix 应用以下列格式提供:

- vmdk (VMware/Virtualbox)
- OVF (Open Virtualisation Format)
- KVM
- HDD/闪存镜像, USE 闪存盘
- Live CD/DVD
- Xen guest
- Microsoft VHD (Azure)
- Microsoft VHD (Hyper-V)

如需运行，启动应用，使用浏览器访问其通过访问动态主机配置协议获得的 IP: http://<host_ip>/zabbix

Zabbix server 在这个环境下已经配置完成，而且运行在 MySQL 数据库上。同时也提供了可用的前端。

这个应用使用了名为 Preseed 文件的标准 Ubuntu/Debian 特性进行构建。

1 Ubuntu 配置的更改　这里更改了一些 Ubuntu 的基础配置。

1.1 仓库

在*/etc/apt/sources.list* 中添加了官方 Zabbix 仓库：

```
## Zabbix repository
deb http://repo.zabbix.com/zabbix/3.4/ubuntu trusty main
deb-src http://repo.zabbix.com/zabbix/3.4/ubuntu trusty main
```

1.2 防火墙

此应用使用了预定义规则的 iptables 防火墙:

- 打开 SSH 端口 (22 TCP);
- 打开 Zabbix agent (10050 TCP) 和 Zabbix trapper (10051 TCP) 端口;
- 打开 HTTP (80 TCP) 和 HTTPS (443 TCP) 端口;
- 打开 SNMP trap 端口 (162 UDP);
- 打开 NTP 端口（123 UDP）的出向（outgoing）连接;
- ICMP pakets 限制为每秒 5 个数据包;
- 丢弃所有其他入向（incoming）连接。

1.3 额外的包

通常来说，Zabbix 已经结合的多种的基础工具使其工作和监控更容易：

- iptables-persistent
- mc
- htop
- snmptrapfmt
- snmp-mibs-downloader

Zabbix 使用了其中的一些包，另外一些用以帮助用户配置/管理应用设置。

1.4 使用静态 ip 地址

默认情况下，设备使用 DHCP 来获取 IP 地址。静态地址详细说明：

- 以 root 用户身份登陆；
- 在你最擅用的编辑器中打开/etc/network/interfaces 文件；
- *iface eth0 inet dhcp → iface eth0 inet static*
- 在 *iface eth0 inet static* 之后添加一下行:
  - *address* < 应用的 *IP*>
  - *netmask* < 网络掩码 >
  - *gateway* < 网关地址 >
- 执行命令 **sudo ifdown eth0 && sudo ifup eth0**.

Note:
有关其他可行选项的相关信息，参见官方 Ubuntu 文档.

要配置 DNS，在*/etc/resolv.conf* 中添加名称服务器条目，在每一行单独指定名称服务器：**nameserver 192.168.1.2**。

1.5 更改时区

应用默认使用 UTC 作为系统时钟。如需更改时区，那么从*/usr/share/zoneinfo* 中复制合适的文件到 */etc/localtime* 中，例如:

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

1.6 语言环境更改

这个应用包含了一些语言环境设置更改:

- 包含语言: *en_US.UTF-8*, *ru_RU.UTF-8*, *ja_JP.UTF-8*, *cs_CZ.UTF-8*, *ko_KR.UTF-8*, *it_IT.UTF-8*, *pt_BR.UTF-8*, *sk_SK.UTF-8*, *uk_UA.UTF-8*, *fr_FR.UTF-8*, *pl.UTF-8*;
- 默认语言环境是 *en_US.UTF-8*.

这些更改都须支持多语种 Zabbix WEB 界面。

1.7 其他更改

- 网络被配置为使用 DHCP 来获取 IP 地址;
- **fping** 工具被设置为拥有 4710 权限，且由 zabbix – suid 组所有，只允许由 Zabbix 组使用
- ntpd 配置为与公有 NTP 服务器进行同步: *ntp.ubuntu.com*;
- LVM 逻辑卷管理使用了 ext4 文件系统；
- *"UseDNS no"* 添加到 SSH 服务配置文件*/etc/ssh/sshd_config* 以避免长时间 SSH 连接等待；文件系统；
- 在*/etc/default/snmpd* 配置文件中禁用了 snmpd 守护进程。

2 Zabbix 配置　Zabbix 应用在安装过程中使用了下列密码和配置:

2.1 凭证 (登陆名: 密码)

系统:

- appliance:zabbix

数据库:

- root:<random>
- zabbix:<random>

Note:
数据库密码在安装过程中随机生成。
Root 密码存放在/root/.my.cnf 文件中，在"root" 账户下无需输入密码。

Zabbix 前端:

- Admin:zabbix

要更改数据库用户密码，需在以下位置更改:

- MySQL;
- /etc/zabbix/zabbix_server.conf;
- /etc/zabbix/web/zabbix.conf.php.

2.2 文件位置

- 配置文件存放在 in **/etc/zabbix**.
- Zabbix server, proxy 和 agent 日志文件存放在**/var/log/zabbix**.
- Zabbix 前端存放在**/usr/share/zabbix**.
- 用户 **zabbix** 的 Home 目录是**/var/lib/zabbix**.

2.3 Zabbix 配置更改

- Zabbix 前端的服务器名称被设置为"Zabbix Appliance";
- 前端时区设置为：Europe/Riga (可在**/etc/apache2/conf-available/zabbix.conf** 中修改);

2.4 配置保护

如果您正在运行该应用的 Live CD/DVD 版本，或者由于其他原因无法使用持久化存储，你可以备份整个数据库，包括所有配置和收集到的数据。

要创建备份，则运行：

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

现在你可以将文件 **dbdump.bz2** 传输到另一台机器。

要想从备份中进行恢复，将其传输到应用上并执行

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

> **Attention:**
> 在恢复时，确保 Zabbix server 处于停止状态。

3 前端访问　默认前端可以通过任何地方进行访问。

可通过 *http://<host>/zabbix* 访问前端。

可在**/etc/apache2/conf-available/zabbix.conf** 中定制，修改该文件后需要重启 web 服务器。为此，以 **root** 用户通过 SSH 登陆并执行：

```
service apache2 restart
```

4 防火墙　默认情况下，只有更改条目中所列出的端口是打开的。要打开额外端口，只需修改*"/etc/iptables/rules.v4"* or *"/etc/iptables/rules.v6"* 文件，并重新加载防火墙规则：

```
service iptables-persistent reload
```

5 监控能力　Zabbix 安装程序支持下列功能：

- SNMP
- IPMI
- Web 监控
- VMware 监控
- Jabber 通知
- EZ Texting 通知
- ODBC
- SSH2
- IPv6
- SNMP Traps
- Zabbix Java Gateway

6 SNMP traps　Zabbix 应用使用 *snmptrapfmt* 处理 SNMP trap。它被配置为接受来自于所有地方的任何 trap。

不要求进行身份认证。如果您希望启用认证，需要更改*/etc/snmp/snmptrapd.conf* 文件并指定所需的认证配置。

所有的 trap 存放在*/var/log/zabbix/snmptrapfmt.log* 文件中，它由 logrotate 按 2GB 的文件大小轮询存放。

7 升级　Zabbix 应用包可以进行升级。如要升级，则运行：

```
sudo apt install --only-upgrade 'zabbix.*'
```

8 命名**,** 初始化和其他脚本　已提供适当的初始脚本。要想控制 Zabbix server, 运行以下任一程序：

```
service zabbix-server status
```

如需要控制 Zabbix agent 守护进程，将 **server** 替换为 **agent**；如需要控制 Zabbix proxy 守护进程，则替换为 **proxy**。

8.1 增加可用磁盘空间

<note warning> 在尝试任何步骤之前，请创建所有数据备份。:::

设备中可用磁盘空间可能不够，这种情况下，可以扩展磁盘，为此，可首先在虚拟化环境中拓展块设备，然后执行以下步骤：

启动 *fdisk* 更改分区大小。以 *root* 用户运行:

```
fdisk /dev/sda
```

这将在磁盘 *sda* 上运行 *fdisk*。然后去使用一下命令切换分区：

```
u
```

> **Attention:**
> D 不要通过输入 **c** 来禁用 DOS 兼容模式，否则将破坏分区。

之后删除现有分区，并创建一个合乎需求的新分区。多数情况下，你会接受可用最大值，这会将文件系统扩展到你为虚拟磁盘提供的任一可用大小。为此，在 fdisk 提示中输入以下序列：

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

如果你想为额外分区（swap 等）留些空间，你可以在 *last sector* 中输入另一个值，完成后，发出以下指令以保存更改：

```
w
```

创建分区后（新增磁盘或者扩展现有的磁盘），创建物理卷：

```
pvcreate /dev/sdb1
```

<note warning> 示例中使用了名为/dev/sdb1 的分区，在你的环境下，磁盘名称和分区号可能不同。你可以使用 *fdisk -l /dev/sdb* 命令检查分区号。:::

检查新创建的物理卷:

```
pvdisplay /dev/sdb1
```

检查可用的物理卷。这里必须有两个卷：zabbix-vg 和新创建的：

```
pvs
```

用新创建的物理卷扩展现有的卷组:

```
vgextend zabbix-vg /dev/sdb1
```

检查 “zabbix-vg” 卷组:

```
vgdisplay
```

使用空闲的 PE 空间扩展你的逻辑卷:

```
lvextend -l +100%FREE /dev/mapper/zabbix--vg-root
```

重新分配 root 卷空间 (可在系统中实时完成):

```
resize2fs /dev/mapper/zabbix--vg-root
```

重新启动虚拟机（因为我们修改的分区目前正在使用）。如此，现在文件系统应扩展到分区大小。检查”/dev/mapper/zabbix--vg-root” 卷:

```
df -h
```

9 特定格式说明　9.1 Xen

为 **XenServer** 转换镜像

通过 Citrix Xenserver 使用 Xen 镜像，则需要转换磁盘镜像。因此，你需要:

- 创建一个虚拟磁盘，磁盘大小至少要和镜像一样大。
- 查找这块磁盘的 UUID

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

- 如果有许多磁盘，在创建磁盘时分配的那样，他们可以通过名称参数 *name-label* 进行过滤
- 导入镜像

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

上述操作步骤引用于 *Brian Radford* 的博客。

9.2 VMware

vmdk 格式的镜像可以直接被 VMware Player，Server 和 Workstation 等产品使用。如需要在 ESX，ESXi，vSphere 中使用，必须使用 VMware converter进行转换。

9.3 硬盘/闪存镜像 (raw)

```
dd if=./zabbix_appliance_3.4.0_x86_64.raw of=/dev/sdc bs=4k conv=fdatasync
```

使用闪存/硬盘设备的路径，替换*/dev/sdc*。

10 已知问题

7. 配置

请使用左侧导航栏来访问 “配置” 这一章节的内容。

7. Configuration

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

1 配置模板

概述

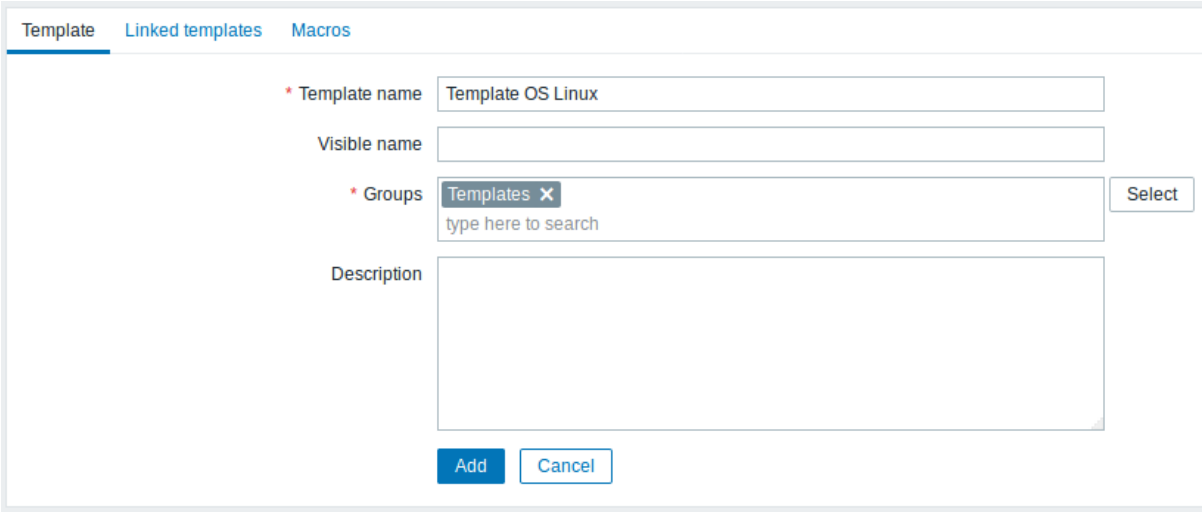
配置模板需要首先通过定义一些参数来创建模板，然后添加实体（项目，触发器，图形等）。

创建模板

要创建模板，请执行以下操作：

- 转到配置 → 模板
- 点击创建模板
- 编辑模板属性

模板选项卡包含常规模板属性。



模板属性：

参数描	
模板名称唯一的	板名称。
可见名称如果你	置了这个名字，那么它将是列表，地图等中可见的。
群组模	所属的主机/模板组。
新的群组可以创	一个新组来保存模板。\\如果为空忽略。
主机/模板应用模	的主机/模板列表。
描述输	模板说明。

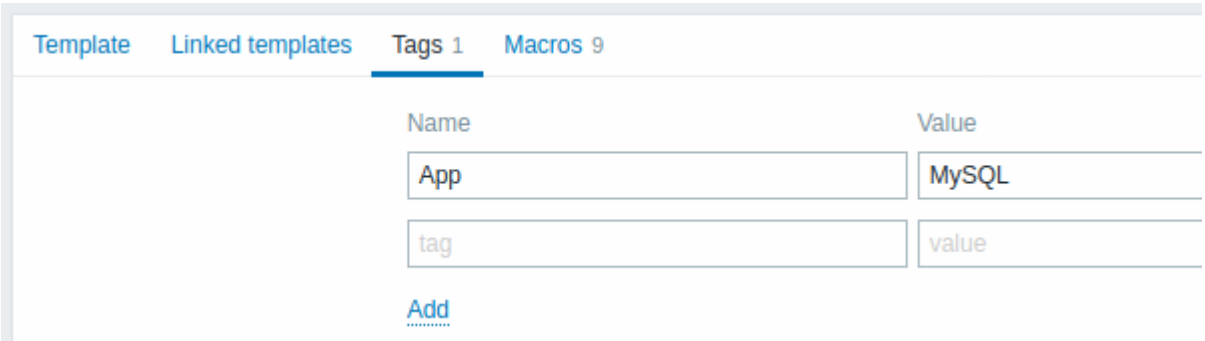
链接的模板选项卡允许您将一个或多个“嵌套”模板链接到此模板。所有实体（项目，触发器，图表等）将从链接的模板继承。

要链接新的模板，请开始输入链接指示器字段，直到出现与输入的字母对应的模板列表。向下滚动选择。当选择要链接的所有模板时，单击添加。

要取消链接模板，请使用链接的模板模块中的两个选项之一：

- 取消链接 - 取消链接模板，但保留其项目，触发器和图形
- 取消链接并清理 - 取消链接模板并删除其所有项目，触发器和图形

宏选项卡允许您定义模板级用户宏。如果选择了继承模板的宏选项，则还可以从链接的模板和全局宏中查看宏。在这里，模板的所有定义的用户宏都显示了它们所决定的值以及它们的起源。



为方便起见，提供了相应模板和全局宏配置的链接。也可以在模板级别上编辑嵌套模板/全局宏，有效地创建模板上宏的副本。

按钮:

Add	添加模板。添加的模板应该出现在列表中。
Update	更新现有模板的属性。
Clone	根据当前模板的属性创建另一个模板，包括从链接模板继承的实体（项目，触发器等）
Full clone	基于当前模板的属性创建另一个模板，包括从链接的模板继承并直接附加到当前模板的实体（项目，触发

Delete	删除模板; 模板（项目，触发器等）的实体与链接的主机保留。
Clear history and trends	从链接的主机中删除模板及其所有实体。
Cancel	取消编辑模板属性。

创建一个模板，开始添加一些实体。

<note important> 项目必须首先添加到模板中。如果没有相应的项目，则无法添加触发器和图形。:::

添加监控项，触发器，图形

要向模板添监控项，请执行以下操作：

- 转到配置 → 主机（或模板）
- 单击所需主机/模板行中的监控项
- 标记要添加到模板的项目的复选框
- 点击项目列表下面的复制
- 选择要复制的项目的模板（或模板组），然后单击复制

所有选定的监控项都应该被复制到模板中。

添加触发器和图形以类似的方式完成（分别从触发器和图形列表），请记住，只有在首先添加所需项目时，才能添加它们。

添加聚合图形

要在配置 → 模板中向屏幕添加聚合图形，请执行以下操作：

- 点击模板行中的聚合图形
- 按照通常的配置聚合图形的方法配置聚合图形

<note important> 可以包含在模板聚合图形中的元素有：简单图形，自定义图形，时钟，纯文本，URL。:::

<note tip> 有关访问从模板局和图形创建的主机聚合图形的详细信息，请参阅主机聚合图形部分。</ note>

配置自动发现规则

请参阅手册的自动发现部分。

添加 Web 场景

要将配置 → 模板中的 Web 场景添加到模板，请执行以下操作：

- 点击模板行中的 *Web*
- 按照通常的 Web 方案配置方式配置 Web 场景

2 链接/取消链接

概述

链接是将模板应用于主机的过程，而取消链接将从主机中删除与模板的关联。

<note important> 模板直接链接到各个主机，而不是主机组。只需将模板添加到主机组就不会链接到主机组。主机组仅用于主机和模板的逻辑分组。:::

链接模板

要将模板链接到主机，请执行以下操作：

- 转到配置 → 主机
- 单击所需的主机并切换到模板选项卡
- 点击链接指示器旁边的选择
- 在弹出窗口中选择一个或多个模板
- 单击主机属性窗体中的添加/更新

主机现在将拥有模板的所有实体（项目，触发器，图形等）。

<note important> 如果在那些模板中有相同监控项的项，如链接到相同的主机将失败。并且作为触发器和图形使用项目，如果使用相同的项目键，它们也不能从多个模板链接到单个主机。:::

当从模板添加实体（监控项，触发器，图表等）时：

- 主机上以前存在的相同实体被更新为模板的实体
- 添加模板中的实体
- 在模板连接之前，只存在于主机上的任何直接链接的实体保持不变

在列表中，模板中的所有实体都以模板名称为前缀，表示这些属于特定模板。模板名称本身（灰色文本）是允许访问模板级别上这些实体列表的链接。

如果某个实体（监控项，触发器，图表等）未被模板名称前缀，则表示该模板存在于主机之前，并未被模板添加。

实体唯一性标准

从模板中添加实体（监控项，触发器，图表等）时，重要的是要知道这些实体已经存在于主机上并需要更新，哪些实体有所不同。决定同一性/差异的唯一性标准是：

- 用于监控项 - 项目键
- 用于触发器 - 触发器名称和表达式
- 用于自定义图形 - 图形名称及其项目
- 用于应用集 - 应用集名称

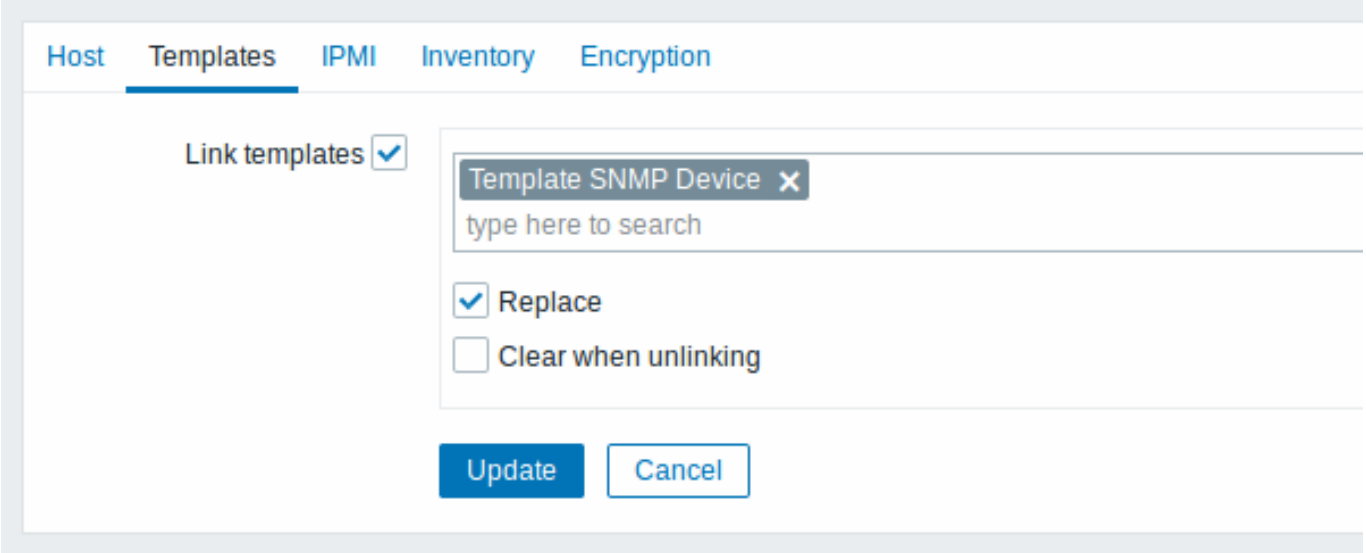
将模板链接到多个主机

有一些批量应用模板的方法（对许多主机一次搞定）：

- 要将模板链接到许多主机，请在配置 → 模板中单击模板，然后从其他主机框中的相应组中选择主机，然后单击 « 更新模板。

反之亦然，如果您在 “收件箱” 中选择链接的主机，请单击 » 并更新模板，从该模板中取消链接模板（主机仍将从模板继承监控项，触发器，图表等）。

- 要更新许多主机的模板链接，在配置 → 主机中通过标记其复选框来选择一些主机，然后单击列表下方的批量更新，然后在模板选项卡中选择链接其他模板：



选择链接模板，并在自动完成字段中开始输入模板名称，直到出现一个提供匹配模板的下拉列表。只需向下滚动即可选择要链接的模板。

在替换选项将允许同时取消关联之前被连接到主机的任何模板链接一个新的模板。在取消链接并清理选项将允许不仅取消链接任何以前链接的模板，但也从中移除（监控项，触发器等）继承了所有元素。

<note tip>Zabbix 提供了大量预定义的模板。您可以使用这些作为参考，但请注意在生产中不改变使用它们，因为它们可能包含太多监控项，并且太频繁地轮询数据。如果你喜欢它们，确保它们以适和你的需求。:::

编辑链接实体

如果你尝试编辑从模板链接的监控项或触发器，你可能会意识到许多关键选项被禁用以进行编辑。这是有道理的，因为模板的想法是在模板级别上以一触式方式编辑事物。但是，你仍然可以启用/禁用单个主机上的监控项，并设置更新间隔，历史长度和其他一些参数。

如果要完全编辑实体，则必须在模板级别进行编辑（模板级快捷方式以表单名称显示），请注意，这些更改将影响所有与此模板链接的主机。

取消链接模板

要从主机中取消链接模板，请执行以下操作：

- 转到配置 → 主机
- 单击所需的主机并切换到模板选项卡
- 单击取消链接或取消链接并清理模板旁边以取消链接

- 单击主机属性窗体中的更新

选择取消链接选项将简单地删除与模板的关联，同时将其所有实体（监控项，触发器，图形等）与主机保持一致。

选择取消链接并清理选项将删除与模板及其所有实体（监控项，触发器，图表等）的关联。

3 嵌套

概述

嵌套是一种包含一个或多个其他模板的模板的方式

因为将各个模板实体分离出来用于各种服务，应用程序等都是有意义的，所以您可能会得到相当多的模板，所有这些模板都可能需要链接到相当多的主机。为了简化图片，可以在一个“嵌套”模板中将一些模板链接在一起。

嵌套的好处在于，您仅需将一个模板链接到主机，并且主机将自动继承链接的模板的所有实体。

配置嵌套模板

如果要链接一些模板，首先可以使用现有模板或新模板，然后：

- 打开模板属性窗体
- 查找链接的模板选项卡
- 单击选择以在弹出窗口中选择模板
- 单击添加以列出所选模板
- 单击模板属性窗体中的添加/更新

现在，模板应该具有所链接的模板的所有实体（监控项，触发器，自定义图表等）

要取消链接任何链接的模板，以相同的形式使用取消链接或取消链接并清理按钮，然后单击更新。

选择取消链接选项将简单地删除与其他模板的关联，而不删除其所有实体（监控项，触发器，图形等）

选择取消链接并清理选项将删除与其他模板及其所有实体（监控项，触发器，图表等）的关联。

1 主机和主机组

什么是“主机”？

一般来讲，Zabbix 主机是指你希望监控的那些设备，例如服务器、工作站、交换机等等。

创建主机是使用 Zabbix 过程中的一个首要任务。例如，如果你想在一台服务器“X”上监控一些参数，你必须首先创建一个主机称之为“服务器 X”，然后就可以查看添加监控项到这台“服务器 X”上。

主机组是由主机组成的。

前往创建配置一台主机。

1 配置一台主机

概述

按照以下步骤在 Zabbix 前端创建一台主机：

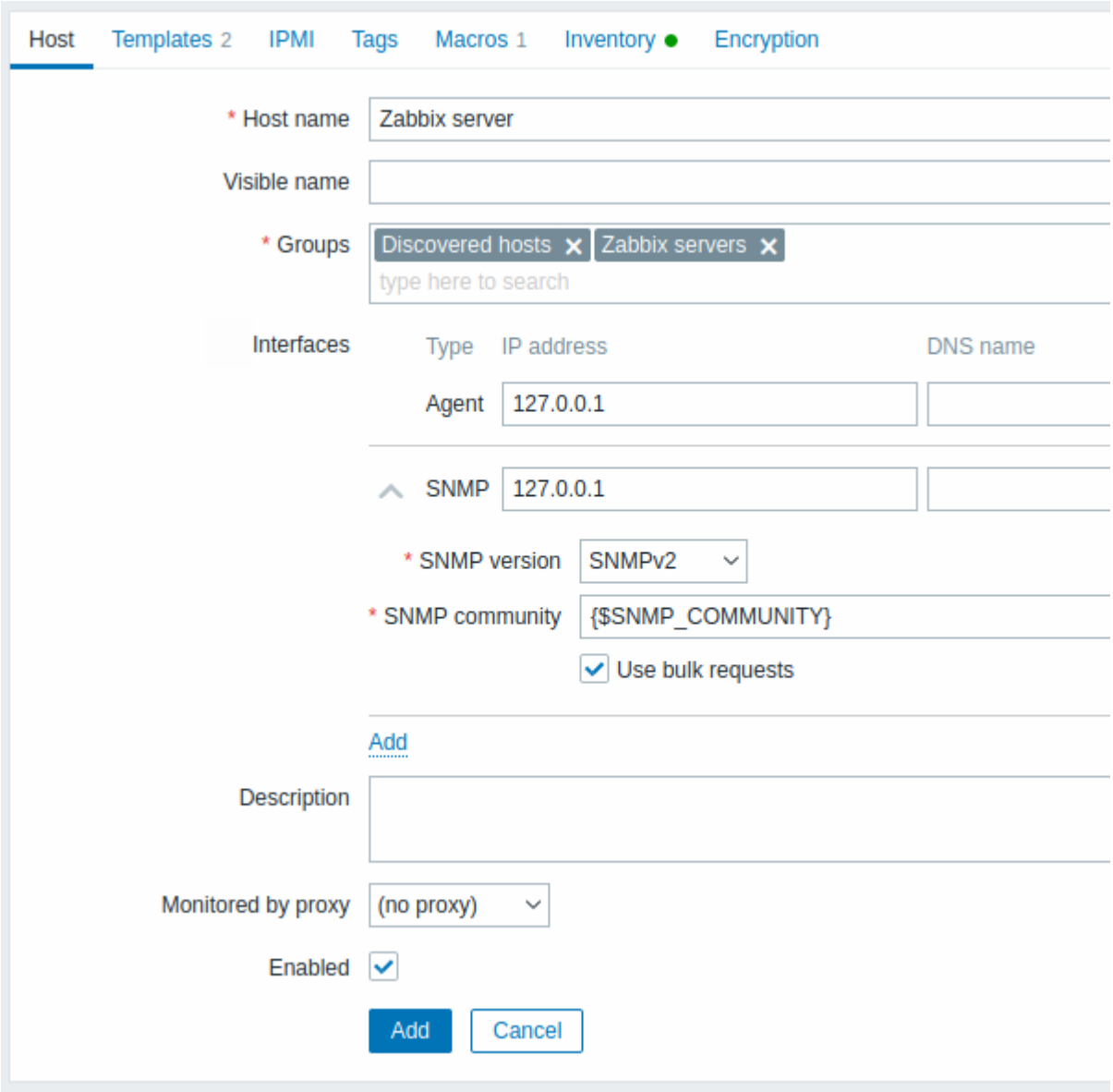
- 进入：配置 → 主机
- 在右侧点击 创建主机 (或者在主机名上编辑一台已有的主机)
- 在表单中输入主机的相关参数

你可以在已经存在的主机上使用 *Clone* 和 *Full clone* 按钮的形式创建一台新的主机，点击 *Clone* 将保留所有的主机参数和模板链接（保留这些模版中的所有实体），*Full clone* 将额外保留直接附加的实体（应用集、监控项、触发器、视图、底层自动发现规则和 Web 定制的场景）。

注意: 当主机被克隆时，它将保留最初在模板上的所有模板实体. 在现有主机级别上对这些实体所做的任何更改（例如更改的监控间隔、修改正则表达式或添加原型到底层发现规则）都不会克隆到新主机；相反，而是与最初模板一致。

配置

这个 **Host** 标签页包含了通用的主机属性：



所有必填输入字段都标有红色星号。

属性	描
Host name	输入一个唯一的主机名。允许有字母、空格、圆点、破折号和下划线。注意：由于Zabbix agent运行在你所配置的那台主机上，所以此agent配置文件的参数*Hostname*必须和这里输

属性	描
Visible name	显示名称。如果你设置了这个名称，它将会在列表、拓扑图等地方显示。此属性支持 UTF-8 。

属性	描
Groups	选择主机所属主机组。一个主机必须至少属于一个主机组。通过添加不存在的组名，可以创建新组并将其链接到主机组。

属性	描
Interfaces	主机支持多种主机接口类型: *Agent*, *SNMP*, *JMX* 和 *IPMI*. 要增加一个新接口，在 *Interfaces* 区域点击 *Add* 并输入 *IP/DNS*, *Connect to* 和 *Port* 信息。注意: 用在任何监控项的接口都不能被删除，并且

属性	描
IP address	主机的IP地址(可选)。
DNS name	主机的DNS名称(可选)。

属性	描
Connect to	点击对应的按钮告诉Zab-bix服务器采用哪种模式从代理端获取数据: **IP** - 连接到主机的IP地址(推荐) **DNS** - 连接到主机的DNS名称

属性	描
Port	TCP/UDP 端口. 默认端口：Zabbix agent 10050, SNMP agent 161, JMX 12345, IPMI 623.
Default	选择单选按钮设置默认接口.
Description	填写主机描述。

属性	描
Monitored by proxy	主机可以被Zabbix服务器或者Zabbix代理服务器监控: **(no proxy)** - 主机被Zabbix服务器监控 **Proxy name** - 主机被Zabbix代理服务器“代理服务器名称”监控

属性	描
Enabled	选中此项激活主机，准备接受监控。如果没选中，表示主机未激活，不能被监控。

Templates 选项卡允许你将templates 链接到主机。所有实体（监控项, 触发器, 图表和应用集）将从模板继承。

要链接一个新模板，请开始在 *Link new templates* 区域键入，直到匹配键入的模板列表出现。向下滚动选择你希望链接的模板。当所有的模板链接完成后，单击 *Add*.

要取消链接模板，请使用 *Linked templates* 区域的两个选项之一:

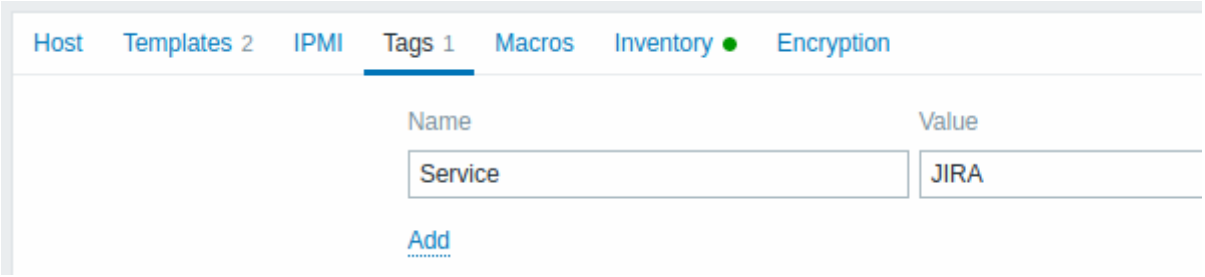
- *Unlink* - 取消链接模板，但保留它的监控项、触发器和图表
- *Unlink and clear* - 取消链接模板并删除所有它的监控项、触发器和图表

列出的模板名可以点击跳转到模板配置表单。

IPMI 选项卡包含 IPMI 管理属性。

参数	描
Authentication algorithm	选择认证算法。
Privilege level	选择权限级别。
Username	认证用户名。
Password	认证用户密码。

Macros 选项卡允许你定义主机级别的用户宏. 如果你选择了 *Inherited* 和 *host macros* 选项，你也可以在这里查看模板级的宏以及全局宏。那里是为主机定义全部用户宏的地方，用户宏显示解析的值以及来源。



为方便考虑，提供了相应模板和全局宏配置的链接。还可以在主机级别编辑一个模板/全局宏，有效地在主机上创建宏的副本。

Host inventory 选项卡允许你为主机手工输入资产 信息。你还可以选择启用 自动资产信息填充, 或者禁用此主机的资产信息填充。

Encryption 选项卡允许你要求加密 与主机建立连接。

参数描	
Connections to host	Zabbix 服务器或 Zabbix 代理服务器如何连接到主机上的 Zabbix Agent：无加密(默认)；使用 PSK(预共享密钥)或者证书。

参数	描
Connections from host	从主机选择允许的连接类型(例如 Zabbix agent 和 Zabbix Sender)。可以同时选择多种连接类型(对于测试及切换至其他连接类型时有帮助)。默认是“No encryption”。

参数	描
Issuer	允许颁发证书。证书首先会通过CA(认证机构)认证。如果是有效的，则由CA签名，然后可以使用*Issuer*字段来进一步限制允许的CA。如果你的Zabbix安装使用多个CA证

参数描	
Subject	允许的证书主题。证书首先通过CA验证。如果它是有效的，由CA签名，则*Subject*字段可以用于仅允许一个*Subject*字符串值。如果此字段为空，则接受由配置的CA签名的

参数	描
PSK identity	预共享密钥身份字符串.

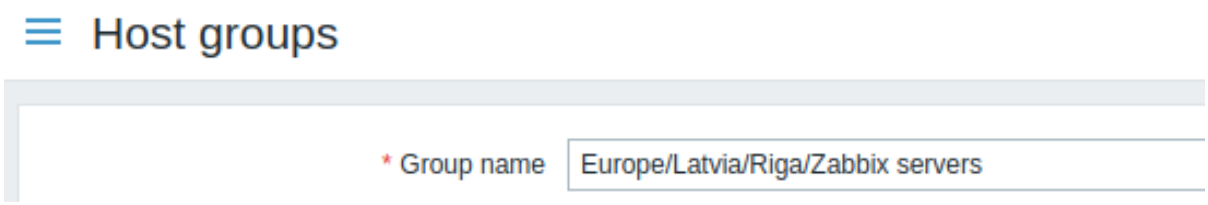
参数	描
PSK	预共享密钥（hex-string）。如果 Zabbix 使用 GnuTLS 或者 OpenSSL 库，最大长度：512 位十六进制数，如果 Zabbix 使用 mbed TLS（PolarSSL）库，则是 64 位十六进制（32 字节 PSK）。示例：1f87b595725ac5

创建一个主机组

要在 Zabbix 页面创建一个主机组，请执行以下步骤:

- 进入: *Configuration → Host groups*
- 单击页面右上角的 *Create Group*

- 在表单中输入组的参数



所有必填输入字段都标有红色星号。

参数描	
Group name	输入唯一的主机组名称。要创建一个嵌套的主机组，请使用'/'正斜杠分隔符，例如Europe/Latvi servers.即使不存在这3个父主机组(Europe/Latvi，你也可以创建该组。在这种情况下，创建父

参数描	
Apply permissions to all subgroups	复选框仅适用于Zabbix Super Admin用户，仅在编辑现有主机组时可用。\\选中此复选框并单击*Update*以对所有嵌套主机组应用相同级别的权限。对于可能已将权限分配给

参数描

嵌套主机组的权限

- 当将子主机组创建到现有的父主机组时，该子主机组用户组 的权限将从父组级继承，(例如，如果 `Riga` 已经存在，创建 `Riga/Zabbix servers`)
- 将父主机组创建到现有的子主机组时，不会设置父级的权限 (例如，如果 `Riga/Zabbix servers` 已经存在，创建 `Riga`)

2 资产管理

概述

你可以将联网设备的资产信息保存在 Zabbix 里。

Zabbix 管理页面有一个特殊的 *Inventory* 菜单。但你一开始不会看到任何数据，这里你也不能输入任何资产相关的信息。资产信息是在配置主机时人工录入建立的资产数据，或者通过使用某些自动填充选项完成的录入。

构建资产库

手动模式

当配置一台主机时，在 *Host inventory* 选项卡中，你可以输入设备类型、序列号、位置、负责人等详细信息 - 这些数据将填充资产信息。

如果主机资产信息中包含 URL，并以 “http” 或 “https” 开头，则会在 *Inventory* 中呈现为可点击的链接。

自动模式

主机资产也可以自动填充。为了使自动填充功能生效，配置主机时，*Host inventory* 选项卡中的清单模式必须设置为 *Automatic*。

然后，你可以通过配置主机监控项 以其值填充任何主机资产字段，指示监控项配置中具有相应属性（称为项目将填充主机资产的字段）的目标字段。

以下是对资产自动发现特别有用的监控项：

- system.hw.chassis[full|type|vendor|model|serial] - 默认是 [full], 需要 root 权限
- system.hw.cpu[all|cpunum,full|maxfreq|vendor|model|curfreq] - 默认是 [all,full]
- system.hw.devices[pci|usb] - 默认是 [pci]
- system.hw.macaddr[interface,short|full] - 默认是 [all,full], interface 支持正则表达式
- system.sw.arch
- system.sw.os[name|short|full] - 默认是 [name]
- system.sw.packages[package,manager,short|full] - 默认是 [all,all,full], package 支持正则表达式

资产模式选择

可以在主机配置过程中选择资产模式。

默认情况下，新主机的资产模式是根据 *Administration* → *General* → *Other* 中的默认主机资产模式设置选择的。

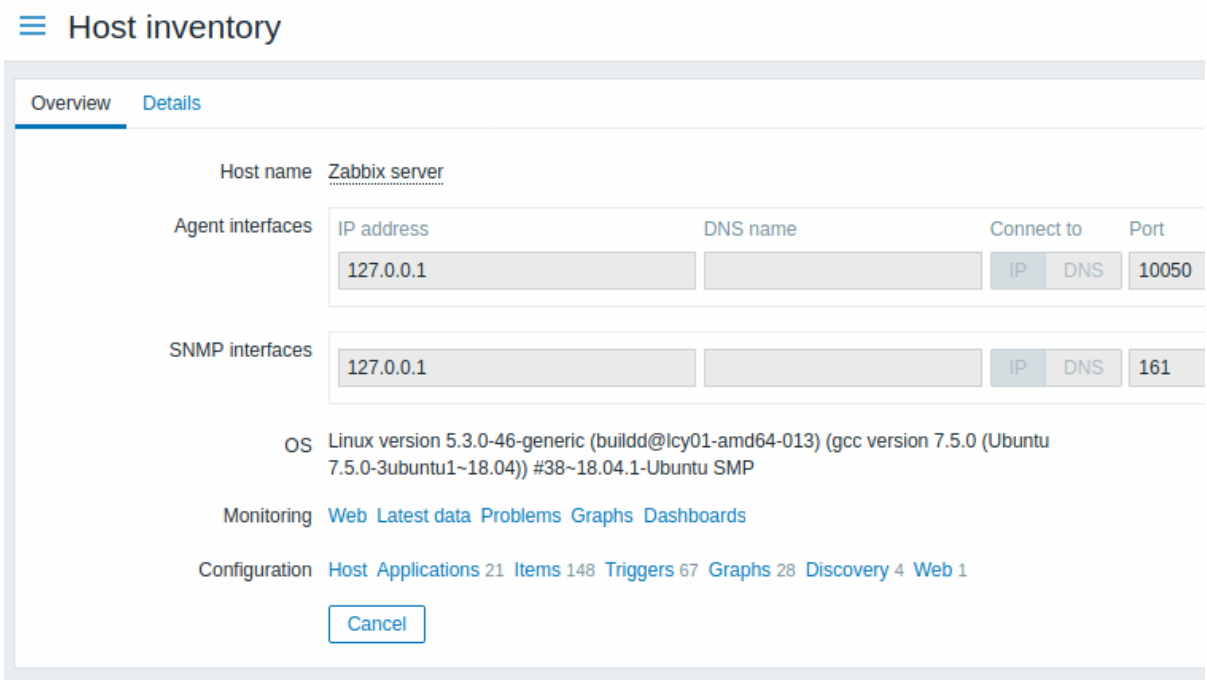
对于通过网络发现或自动注册操作添加的主机，可以定义 *Set host inventory mode* 操作，选择手动或自动模式。此操作将覆盖 *Default host inventory mode* 设置。

资产清单概述

Inventory 菜单中提供了所有现有资产数据的详细信息。

在 *Inventory* → *Overview* 你可以通过资产的各个字段获取主机数。

在 *Inventory* → *Hosts* 你可以看到所有具有资产信息的主机。单击主机名将以表单显示资产明细。



Overview 标签展示：

参数	描
Host name	主机名称。 单击名称将打开一个菜单，其中包含了为主机定义的脚本。 主机名显示为橙色图标，表示主机正在维护中。
Visible name	主机的显示名称（如已定义）
Host (Agent, SNMP, JMX, IPMI) interfaces	此区域提供了为主机配置的接口的详细信息。
OS	主机的操作系统资产清单字段（如已定义）。

参数	描
Hardware	主机硬件清单字段(如已定义)。
Software	主机软件清单字段(如已定义)。
Description	主机描述。
Monitoring	与监控部分的链接，其中包含该主机的这些数据: *Web*, *Latest data*, *Triggers*, *Problems*, *Graphs*, *Screens*.
Configuration	链接到此主机的这些配置部分： *Host*, *Applications*, *Items*, *Triggers*, *Graphs*, *Discovery*, *Web*. 配置的实体的数量在每个链接之后的括号中列出。

Details 选项卡显示填充的所有资产清单字段（不为空）。

资产清单宏

有可用于通知的主机资产清单宏 {INVENTORY。*}，例如：

“服务器在 {INVENTORY.LOCATION1} 有问题，负责人是 {INVENTORY.CONTACT1}，电话号码 {INVENTORY.POC.PRIMARY.PHONE.A1}。”

关于更多详细信息，请参阅Macros supported by location 页面。

3 批量更新

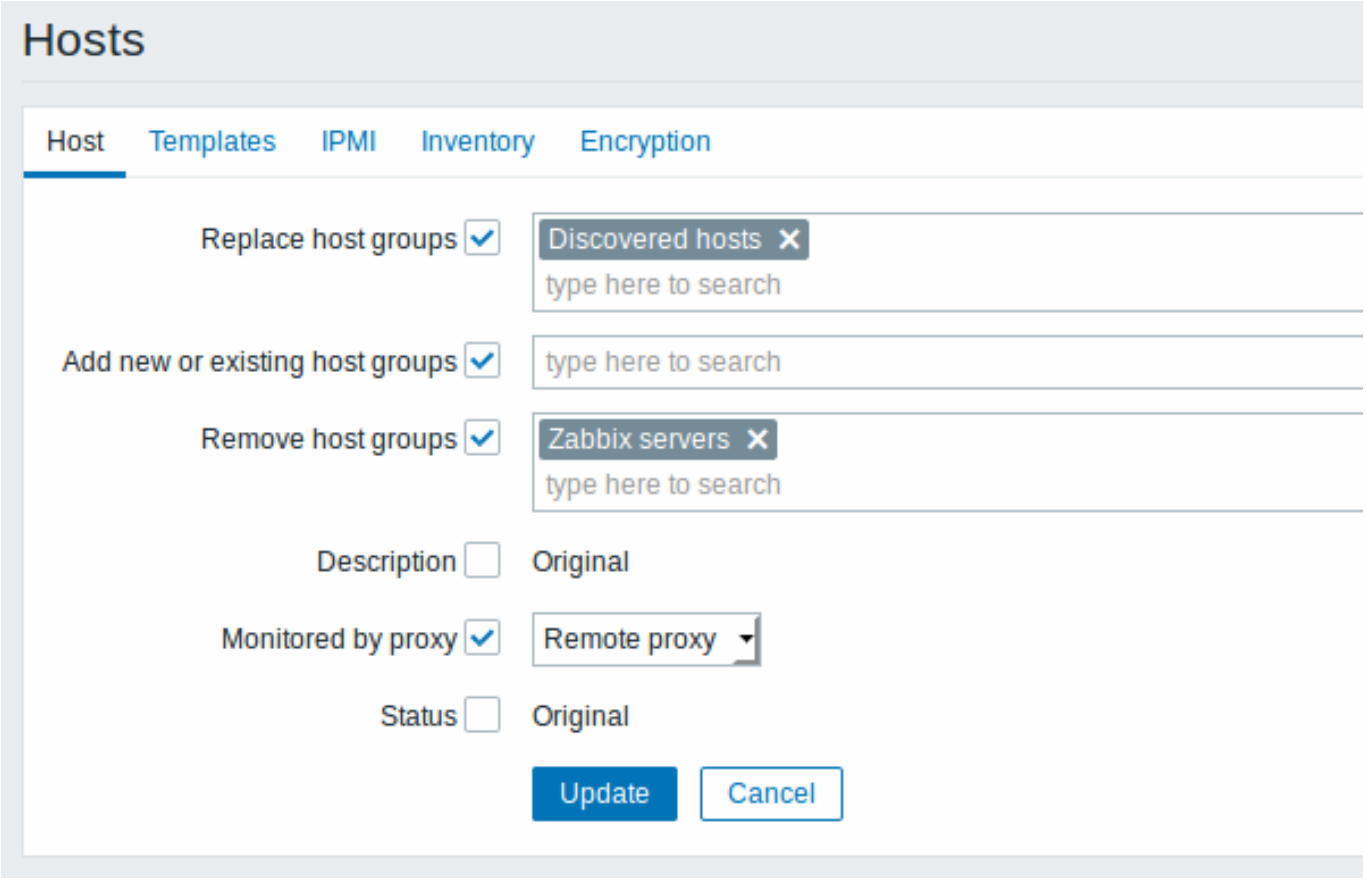
概述

有时你可能想要一次更改多个主机的某些属性，那么你可以使用批量更新功能来代替打开每个主机进行编辑。

使用批量更新

要批量更新某些主机，请执行以下操作:

- 主机列表中，在要更新的主机之前选中复选框
- 点击下方的 *Mass update* 按钮
- 跳转到属性对应所需的选项卡 (*Host*, *Templates*, *IPMI* 或者 *Inventory*)
- 标记要更新的任何属性的复选框，并为其输入新值

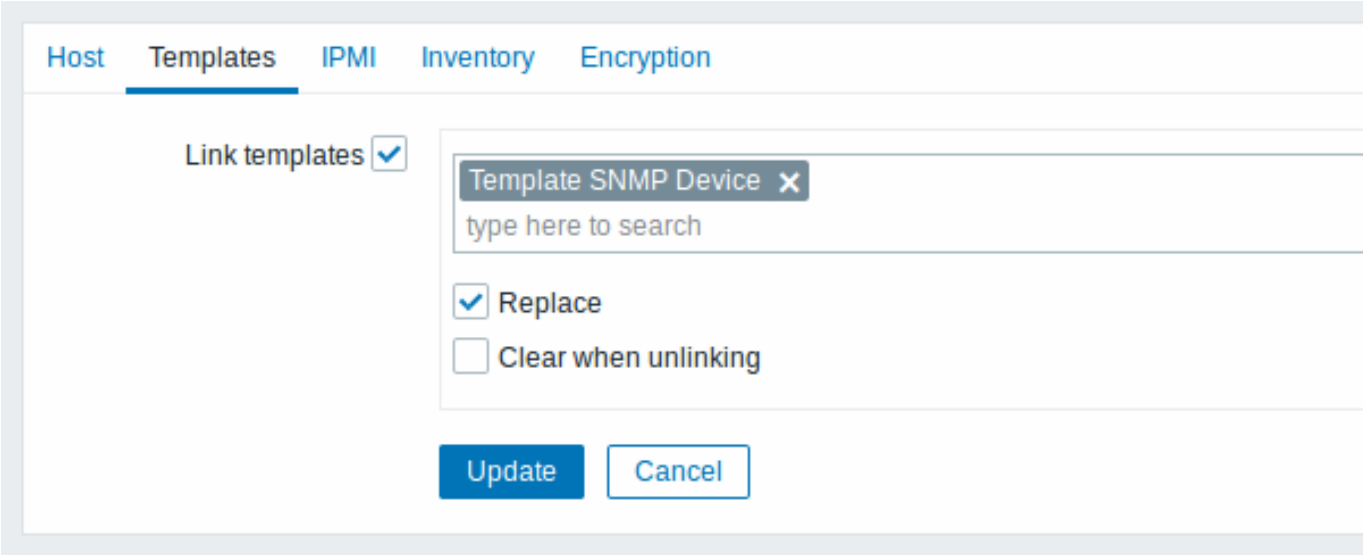


Replace host groups 将从任何现有主机组中删除主机，并将其替换为该字段中指定的主机。

Add new or existing host groups 允许从现有主机组指定其它主机组，或为主机输入全新的主机组。

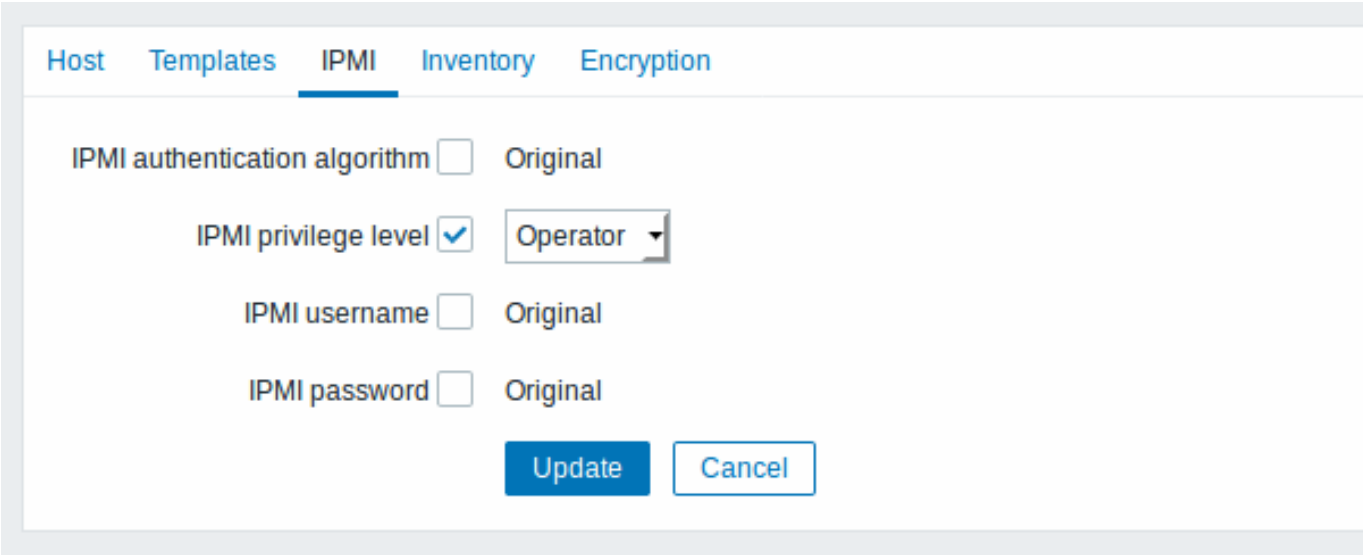
Remove host groups 将从主机中删除特定主机组。如果主机已在选定的组中，则将从这些组中删除主机。如果主机不在选定的组中，则不会添加或删除任何内容。如果同时替换和删除相同的主机组，则实际上主机没有组。

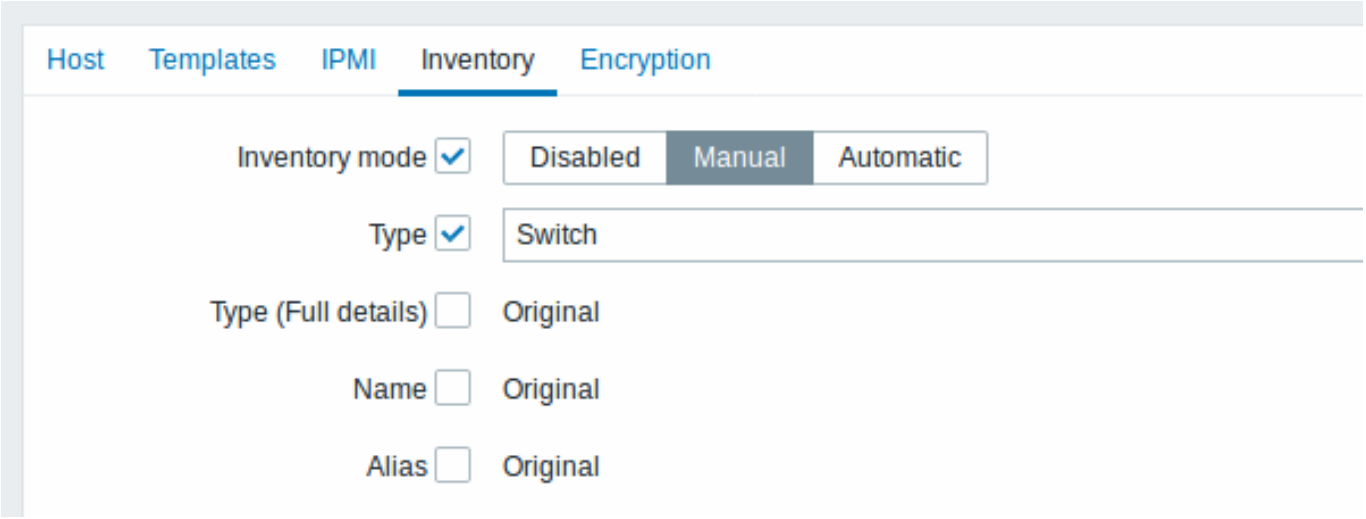
这些字段都是自动完成的 - 开始输入它们提供了匹配的主机组的下拉列表。如果主机组是新的，它也会出现在下拉列表中，并在字符串后用 *(new)* 表示。只需向下滚动即可选择。



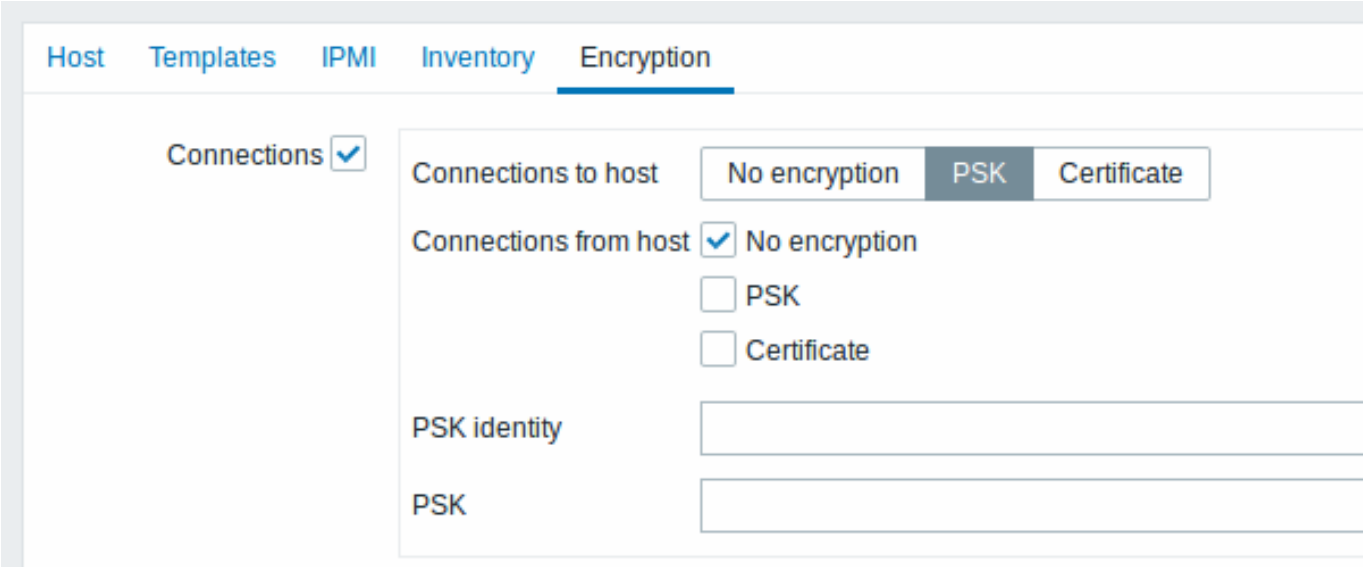
要在 **Templates** 选项卡中更新模板链接，请选择 *Link templates*，并在自动填充字段中开始输入模板名称，直到出现一个提供匹配模板的下拉列表。只需向下滚动即可选择要链接的模板。

Replace 选项将允许链接新模板，同时取消链接之前链接到主机的任何模板。*Clear when unlinking* 选项将不仅可以取消链接任何以前链接的模板，还可以删除所有继承自它们的元素（监控项、触发器等）。





为了能够批量更新资产字段，*Inventory mode* 应设置为 “手动” 或 “自动”。



完成所有必需的更改后，单击 *Update*，所有选定主机的属性将相应更新。

2 监控项

概述

Overview

监控项是从主机收集的数据信息。

配置主机后，你需要添加一些监控项以开始获取实际数据。

一个监控项是一个独立的指标。快速添加多个监控项的一种方法是将一个预定义的模板附加到主机。然而，为了优化系统性能，您可能需要对模板进行微调，使只有真正需要的监控项被频繁的监控到。

在单个监控项中，你可以指定从主机收集哪些数据。

为此，你可以使用监控项 key。从而，具有名称为 system.cpu.load 的监控项将收集处理器负载的数据，而名为 net.if.in 的监控项将收集传入的流量信息。

要用 key 指定更多的参数，请在 key 后添加方括号。例如，system.cpu.load**[avg5]** 将返回最近 5 分钟的处理器负载平均值，而 net.if.in**[eth0]** 将显示接口 eth0 中的流量。

Note:
对于所有支持的监控项类型和监控项的 Key，请参阅监控项类型的各个部分。

继续创建和配置监控项。

1 创建监控项

概述

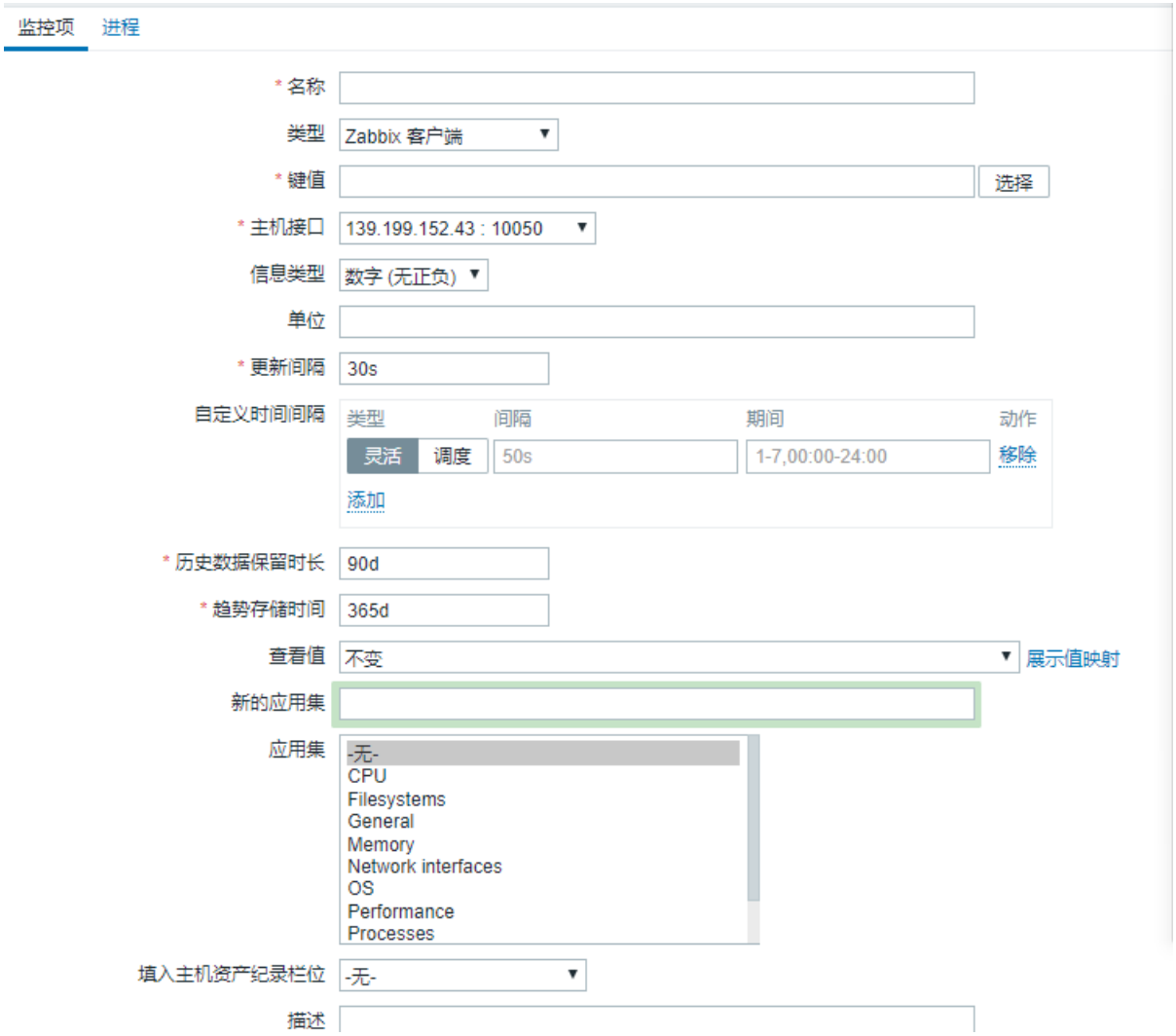
要在 Zabbix 管理页面创建一个监控项，请执行以下操作：

- 进入到: 配置 → 主机
- 在主机所在的行单击 监控项
- 点击屏幕右上角的创建监控项
- 输入表单中监控项的参数

你也可以通过打开一个监控项，按克隆按钮，然后以不同的名称保存。

配置

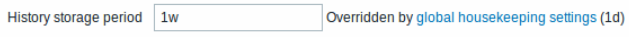
监控项选项卡包含了常规监控项属性：

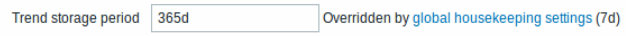


所有必填输入字段都标有红色星号。

参数描

名称这	命名监控项名称。 可以使用以下宏： **$1, $2...$9** - 指的是监控项的第 1、2...9 个参数 例如：$1 上的可用磁盘空间 如果监控项的 key 是”vfs.fs.size[/,free]”, 说明将自动更改为”Free disk space on /”
类型监	项类型。参考单个监控项类型 章节.
键值监	项键值. 可支持的监控项键值 能够在各个监控项类型中找到。 这个键值在单个主机中必须是唯一的。 如果键值的类型是’Zabbix 客户端’、’Zabbix 客户端 (主动式)’, ’简单检查’ 或者’Zabbix 整合’, 则此 key 必须被 Zabbix 客户端或者 Zabbix 服务端支持。 也可以查看: 正确的键值的格式.
主机接口选择主	接口。编辑主机级别的监控项时，此字段可用。

信息类型执行转	后存储在数据库中的数据类型（如果有）。 数字 **(**无正负**)** - 64 位无符号整数 数字 **(**浮点**)** - 浮点数 可以存储负值。 允许范围: -999999999999.9999 到 999999999999.9999. 从 Zabbix 2.2 开始，也支持科学计数值。例如。1e+7,1e-4。 字符 - 短文本数据 日志 - 具有可选日志相关属性的长文本数据 (timestamp, source, severity, logeventid) 文本 - 长文本数据。可参见文本数据限制。
单位如	设置了单位符号，Zabbix 将在收到数据后再加工处理，并使用设置单位后缀进行显示。 默认情况下，如果原始值超过 1000，则除以 1000，并相应显示。例如，如果设置 *bps* 并接收到值为 881764，则将显示为 881.76 Kbps。 特殊处理用于 B（字节），Bps（每秒字节数）单位，除以 1024. 因此，如果单位设置为 B 或 Bps，Zabbix 将显示： 1 为 1B/1Bps 1024 为 1KB/1KBps 1536 为 1.5KB/1.5KBps 如果使用以下与时间相关的单位，则使用特殊处理： **unixtime** - 转换成 “yyyy.mm.dd hh：mm：ss”。要正确转换，接收的值必须是数字（无符号）类型的信息。 **uptime** - 转换为”hh:mm:ss” 或者”N days, hh:mm:ss” 例如，如果你收到的值为 881764（秒），则显示为 “10 天，04:56:04” **s** - 转换成 “yyy mmm ddd hhh mmm sss ms”; 参数被视为秒数。 例如，如果您收到的值为 881764（秒），则显示为 “10d 4h 56m” 只显示 3 个主要单位，如 “1m 15d 5h” 或 “2h 4m 46s”。如果没有显示天数，则仅显示两个级别 - “1m 5h”（不显示分钟，秒或毫秒）。如果该值小于 0.001，将被转换成 “<1 ms”。 请参阅单位黑名单.
更新间隔每 N 秒	检索一次这个项目的新值。允许的最大更新间隔为 86400 秒（1 天）。 支持时间后缀，例如 30s，1m，2h，1d。 支持用户宏。 注意：如果设置为 “0”，则不会轮询该项。但是，如果自定义间隔（灵活/调度）也存在非零值，则会在自定义间隔持续时间期间轮询该项。 注意可以通过按立即检查按钮立即轮询现有被动监控项的值。
自定义时间间隔你可以创建用	检查监控项的自定义规则： **Flexible** - 创建更新间隔的异常（间隔不同的频率） **Scheduling** - 创建自定义轮询时间表。 详细信息请查看自定义间隔。 间隔字段支持时间后缀，例如 30s, 1m, 2h, 1d. 支持用户宏。 从 Zabix 3.0.0 开始支持时间表。 注意: 不适用于 Zabbix Agent 的活动监控项。
历史数据保留时长在数据库中保存	细历史的持续时间（1 小时至 25 年）。housekeeper 将删除较旧的数据。 按秒存储。支持时间后缀，例如 2h, 1d. 支持用户宏。 在 *Administration* → *General* → *Housekeeper* 中可以覆盖该值。 如果存在全局设置，将显示一条警告消息：  建议保留最小可能天数的记录值，以减少数据库中的历史记录的大小。你可以保留较长的趋势数据，而不是保存长期的历史数据。参见历史和趋势。请参考历史与趋势.

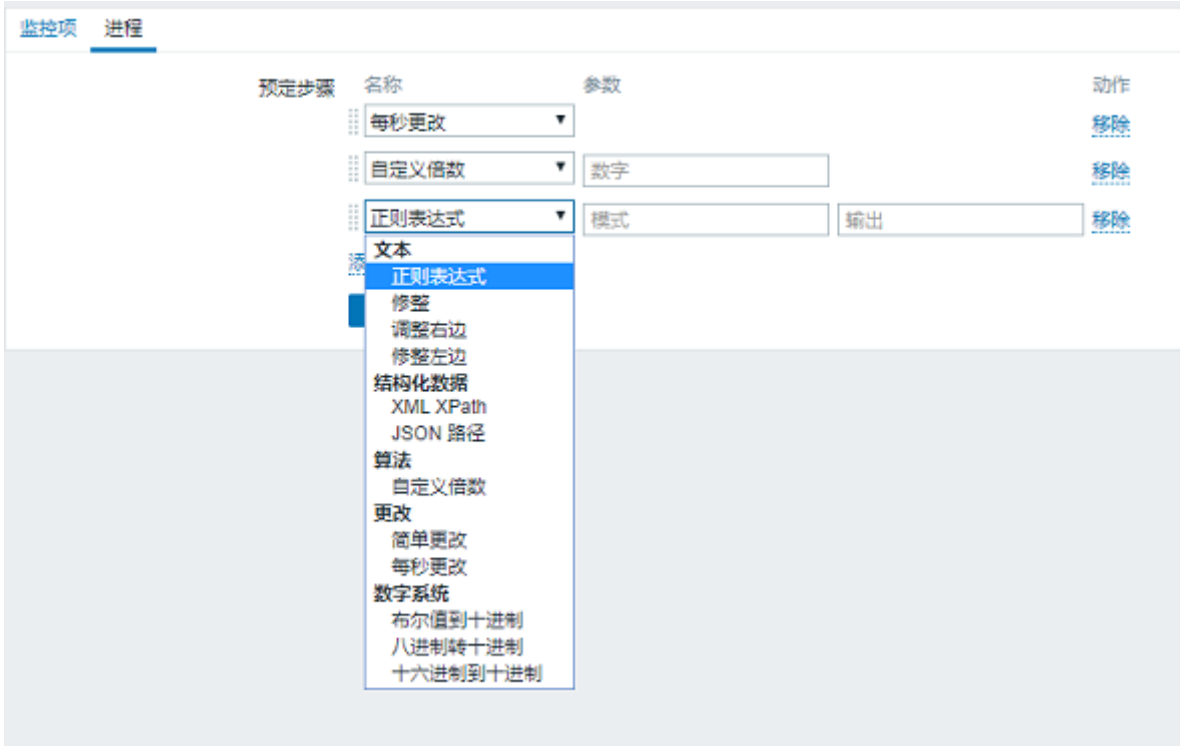
趋势存储时间在数据库中	持聚合（每小时最小值，最大值，平均值，计数）历史的持续时间（1 天到 25 年）。housekeeper 将删除较旧的数据。 按秒存储。支持时间后缀，例如 2h, 1d. 支持用户宏。 在 *Administration* → *General* → *Housekeeper*中可以覆盖该值。 如果存在全局设置，将显示一条警告消息：  注意*:* 保持趋势不适用于非数字数据 - 字符，日志和文本。 参考历史与趋势.
查看值将值	射应用于此监控项。值映射不会改变收到的值，仅用于显示数据。 它只适用于整数项。 例如, ”Windows service states”.
日志时间格式仅适用于 *	日志类型的监控项。支持的占位符**: *** y**: *Year (1970-2038)* *** M**: *Month (01-12)* *** d**: *Day (01-31)* *** h**: *Hour (00-23)* *** m**: *Minute (00-59)* *** s**: *Second (00-59)* 如果留空，则不会解析时间戳。 例如，从 Zabbix Agent 日志文件中考虑以下几行：“23480：20100328：154718.045 Zabbix 代理启动。Zabbix 1.8.2（修订 11211）。“ 它以 PID 的六个字符位置开始，后跟日期，时间和行的其余部分。 该行的日志时间格式为 “pppppp：yyyyMMdd：hhmmss”。请注意，“p” 和 “：” 字符只是占位符，只能是 “yMdhms”。
新的应用集输入监控	的新应用程序的名称。
应用集将监	项链接到一个或多个现有应用程序。
填入主机资产纪录栏位你可以选择项目的值	填充的主机资产字段，如果你为主机启用了自动发现模式资产管理 ，这将会起作用。
描述输	监控项描述。
已启用选中	复选框以启用该监控项，以便对其进行处理。

Note:
当编辑主机级别上的现有模板级别的监控项时，多个字段是只读的。你可以使用表单标题中的链接并转到模板级别并在其中进行编辑，但请记住，模板级别上的更改将更改模板链接到的所有主机的项目。

监控项值预处理

预处理选项卡允许为接收的值定义转换规则。在将值保存到数据库之前，可以进行一次或多次转换。转换按照定义的顺序执行。所有预处理都由 Zabbix 服务器完成。

可参见: 预定步骤详细信息



监控项值预处理参数支持用户宏和带有上下文的用户宏。

转换描

正则表达式将值与 &	t;pattern> 正则表达式匹配，并将值替换为 <output>。正则表达式支持使用\N 序列提取最多 10 个捕获的组。无法匹配输入值将使监控项不受支持。 参数: **pattern** - 正则表达式 **output** - 输出格式化模板。\N（其中 N = 1 ... 9）转义序列被替换为第 N 个匹配组。\ 0 转义序列将替换为匹配的文本。 版本 3.4.0. 之后支持 \\有关示例，请参阅正则表达式部分。
裁剪从	的起始和结尾删除指定的字符。
右裁剪从值	末尾删除指定的字符。
左裁剪从值	起始处删除指定的字符。

XML XPath	使用 XPath 功能从 XML 数据中提取值或段。
	要使此选项生效，必须使用 libxml 支持编译 Zabbix 服务器。
	示例：
	`number(/document/item/value)`
	`<document><item><value>10</value></item></document>`
	提取 10
	`number(/document/item/@attribute)` 将从
	`<document><item`
	`attribute="10"></item></document>` 提取 10
	`/document/item` 将从
	`<document><item><value>10</value></item></document>`
	提取 `<item><value>10</value></item>`
	请注意，不支持名称空间。
	版本 3.4.0. 之后支持

JSON 路径使	JSON 路径功能的简单子集从 JSON 数据中提取值或段。 示例： `$.document.item.value` 将从 `{"document":{"item":{"value": 10}}}` 提取 10 `$.document.item` 将从 `{"document":{"item":{"value": 10}}}` 提取 `{"value": 10}` `$['a document'].item.value` 将从 `{"a document":{"item":{"value": 10}}}` 提取 10 `$.document.items[1].value` 将从 `{"document":{"items":[{"value": 10}, {"value": 20}]}}` 提取 20 请注意，仅支持以点或括号表示法指向单个对象的直接路径。 在 JSON 路径点符号（`$.a.b.c`）中只能使用字母数字 + 下划线字符。如果 JSON 对象名称包含其他字符，则必须使用括号符号（`$['a']['b']['c']`）。两种符号可以混合（`$.a['b'].c`） 不支持提取多个值。 版本 3.4.0. 之后支持
自定义倍数将值乘以	定的整数或浮点值。\\使用此选项将以 KB，MBps 等接收的值转换为 B，Bps，否则 Zabbix 无法正确设置前缀（K，M，G 等）。 \\从 Zabbix 2.2 开始，也支持使用科学符号。例如。1e + 70。
简单更改计算当	值和上一个值之间的差值。\\评估为 **value**-**prev_value**，其中 *value* - 当前值; *prev_value* - 以前收到的值\\每个监控项只允许一个更改操作。
每秒更改计算每	速度的值变化（当前值和上一个值的差值）。 计算为 (**value**-**prev_value**)/(**time**-**prev_time**)，其中 *value* - 当前值; *prev_value* - 以前收到的值; *time* - 当前时间戳，*prev_time* - 以前值的时间戳。 这个设置是非常有用的，以获得每秒不断增长的速度值。如果当前值小于上一个值，Zabbix 将丢弃该差异（不存储）并等待另一个值。这有助于正常工作，例如，32 位 SNMP 计数器的包装（溢出）。 注意: 由于此计算可能产生浮点数，建议将’ 信息类型’ 设置为 数字 *(浮点)*，即使传入的原始值是整数。这对于小数部分尤其重要。如果浮点值很大并且可能超过’float’ 字段长度，在这种情况下，整个值可能会丢失，实际上建议使用数字（无符号），因此只会修整小数部分。 每个监控项只允许一个更改操作。
布尔值转十进制将值从布尔值	换为十进制。文本表示被转换为 0 或 1. 因此，“TRUE” 存储为 1，“FALSE” 存储为 0. 所有值都以不区分大小写的方式进行匹配。 当前被认为的布尔值值如下： *TRUE* - true, t, yes, y, on, up, running, enabled, available, ok, master *FALSE* - false, f, no, n, off, down, unused, disabled, unavailable, err, slave 此外，任何非零数值都被认为是 TRUE，0 被认为是 FALSE。 ok, master, err, slave 在 4.0.0 后支持
八进制转十进制将八进制格式	值转换为十进制。
十六进制转十进制将值从十六进制	式转换为十进制。 请注意，数字格式不支持包含空格的值。作为解决方法，您可以使用正则表达式预处理步骤删除十六进制到十进制步骤之前的空格。

<note tip> 如果使用自定义乘数或存储值每秒更改对于信息类型设置为数字（无正负）的项目，并且得到的计算值实际上是浮点数，则计算值仍被认为是正确的值，并通过修剪小数部分后将值存储为整数。:::

表单按钮

表单底部的按钮允许执行多种操作。

添加	添加监控项。此按钮仅适用于新监控项。
更新	更新监控项的属性。
克隆	根据当前监控项的属性创建另一个监控项。
Check now	立即执行新监控项值的检查。仅支持 **passive** 检查（参见更多详细信息）。 注意当立即检查值时，配置缓存不会更新，因此该值不会反映项目配置的最新更改。
清除历史和趋势	删除监控项历史记录和趋势。
删除	删除监控项。
取消	取消编辑监控项属性。

文本数据限制

文本数据限制取决于数据库后端。在将文本值存储到数据库之前，它们会被截断以匹配数据库值类型限制：

数据库信息	型		
	Character	Log	Text
Mysql	255 characters	65536 bytes	65536 bytes
Postgresql	255 characters	65536 characters	65536 characters
Oracle	255 characters	65536 characters	65536 characters
IBM DB2	255 bytes	2048 bytes	2048 bytes

Unit blacklisting

单位黑名单

默认情况下，为项目指定单位会导致添加该单位的乘数前缀 - 例如，单位为 “B” 的传入值 “2048” 将显示为 “2KB”。

但是，可以通过使用 ！ 前缀来阻止任何单位转换，例如 ！ 。为了更好地说明转换如何使用和不使用黑名单，请参阅以下值和单位示例：

```
1024 !B → 1024 B
1024 B → 1 KB
61 !s → 61 s
61 s → 1m 1s
0 !uptime → 0 uptime
0 uptime → 00:00:00
0 !! → 0 !
0 ! → 0
```

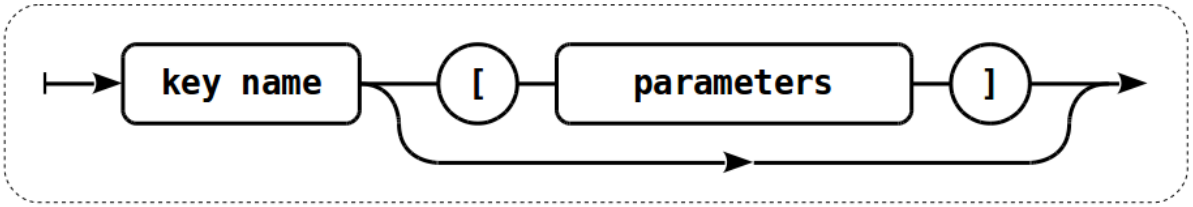
Note:
在 Zabbix 4.0 之前，有一个硬编码的单位黑名单包括 ms, rpm, RPM, %。这个黑名单已被弃用，因此将这些单位列入黑名单的正确方法是 !ms, !rpm, !RPM, !%。

不支持的监控项

如果由于某种原因无法检索其值，则该项可能不受支持。此类项目仍以固定间隔重新检查，可在管理页面. 中进行配置。

1 监控项键值的格式

监控项键值的格式（包括关键参数）必须遵循语法规则。以下插图描述了支持的语法。每个点的允许元素和字符可以通过跟随箭头来确定 - 如果可以通过线到达某个块，则允许，如果不能到达 - 则不允许。



要构建一个有效的监控项的键值，首先指定键值的名称，然后选择是否具有参数，如果都两个都满足则被执行。

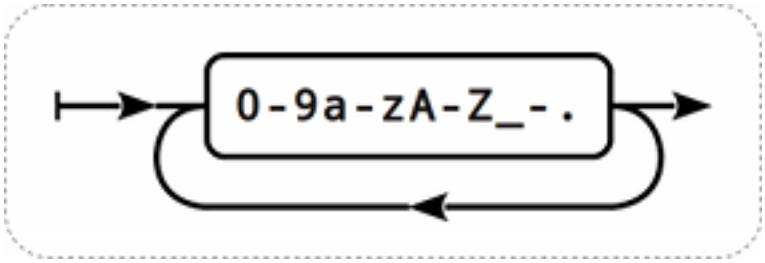
键值名称

Key 名本身具有有限的允许字符范围，允许的字符是：

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

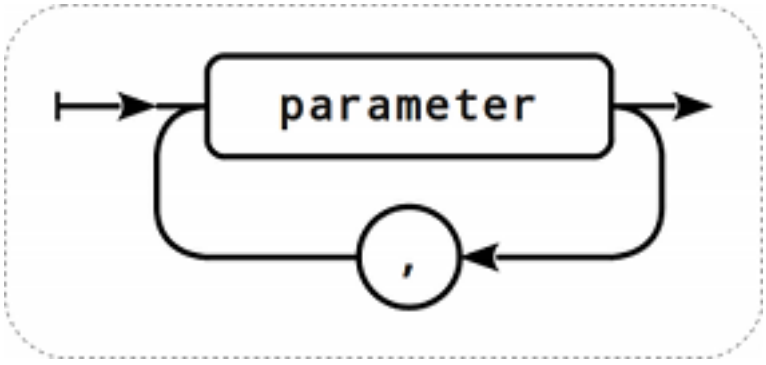
即:

- 所有的数字;
- 所有的小写字母;
- 所有大写字母;
- 下划线;
- 减号;
- 点.

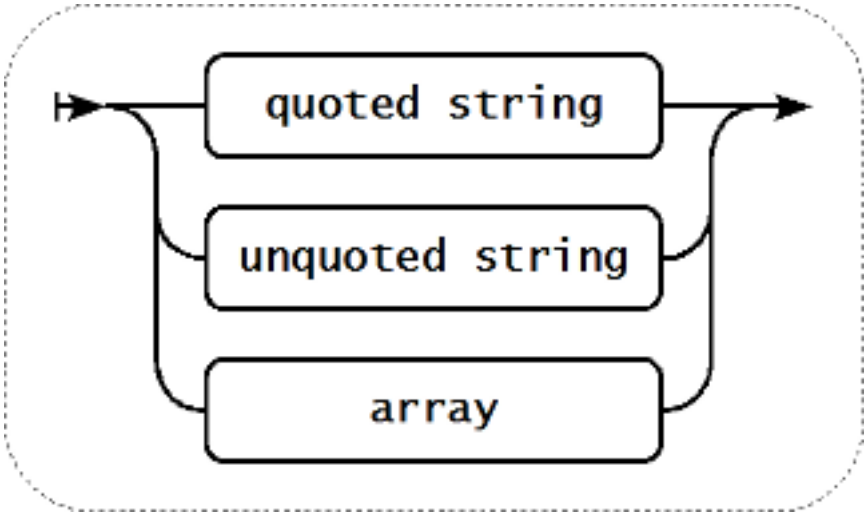


键值的参数

监控项的键值可以有多个逗号分隔的参数。



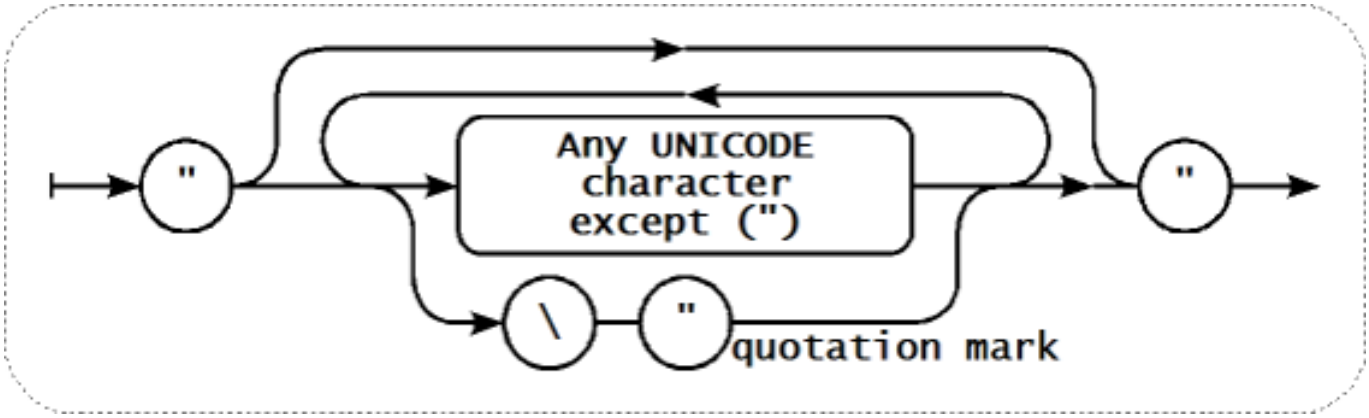
每个 key 参数可以是带引号、无引号的字符串或数组。



参数也可以为空，此时使用默认值。在这种情况下，如果指定了其它参数，则必须添加对应数量的逗号。例如，键值 **icmp-ping[„200„500]** 将指定每 ping 一次的时间间隔为 200 毫秒，超时时间为 500 毫秒，所有其它参数为默认值。

参数 - 带引号的字符串

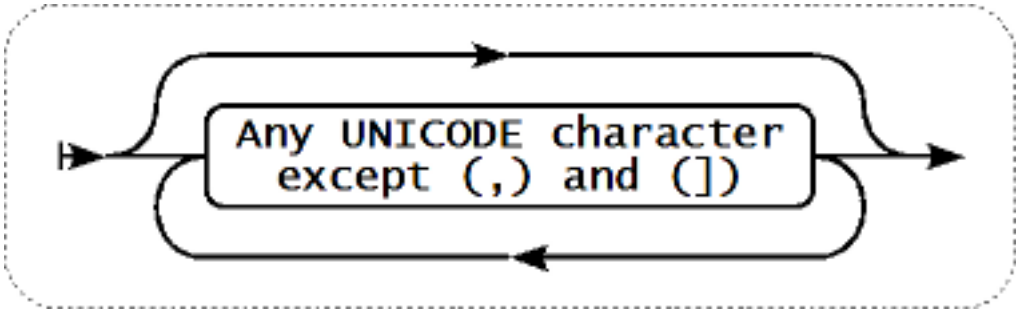
如果键值参数为带引号的字符串，则允许任何 Unicode 字符，如果包含双引号则需要被反斜杠转义。



<note warning> 要引用监控项键值参数，请仅使用双引号，不支持单引号。:::

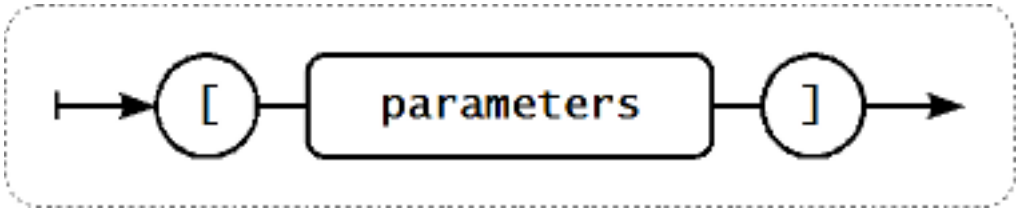
参数 - 不带引号的字符串

如果键值的参数是一个不带引号的字符串，除逗号和右方括号（]）之外，不带引号的参数不能以左方括号（[）开头。



参数 - 数组

如果 key 参数是一个数组，它需要包含在方括号中，其中各个参数需要符合多个参数的规则和语法。



Attention:
多级参数数组，例如 `[a,[b,[c,d]],e]`，是不支持的。

2 自定义间隔

概述

有两种方法可以创建检查项目的时间的自定义规则。灵活间隔，允许重新定义默认更新间隔，调度，可以在特定时间或时间序列执行项目检查。

Flexible intervals

灵活间隔

灵间隔允许重定义特定时间段的默认更新间隔。灵活的间隔被定义为间隔和期间，其中：

- 间隔 – 指定时间段的更新间隔
- 期间 – 灵活间隔有效的时间段（周期格式请参阅详细说明时间期间）

可以定义多达七种灵活的时间间隔。如果多个灵活间隔设置有冲突，则在冲突周期中使用最小的间隔值。请注意，如果灵活间隔的最小值为“0”，则不会进行轮询。在灵活间隔之外，使用默认更新间隔。

请注意，如果灵活间隔等于周期的长度，则该监控项将被精确检查一次。如果灵活间隔大于周期，则可能会检查该监控项一次，或者完全不检查该监控项（因此不建议这样配置）。如果灵活间隔小于周期，监控项将至少被检查一次。

如果灵活间隔设置为“0”，则在灵活间隔期间不轮询监控项，并在周期结束后根据默认更新间隔恢复轮询。示例：

间隔周	描述
10	1-5,09:00-18:00 监控项将在工作时间内每 10 秒检查一次。
0	1-7,00:00-7:00 监控项不会在夜间检查。
0	7-7,00:00-24:00 监控项在星期日不会被检查。
60	1-7,12:00-12:01 监控项将在每天 12:00 点检查。请注意，这种被用作计划检查的一般性方法从 Zabbix 3.0 开始建议使用调度间隔来实现。

调度间隔

调度间隔用于在特定时间检查监控项。虽然灵活间隔被设计为重新定义默认监控项的更新间隔，但是调度间隔用于指定独立执行的检查计划。

调度间隔定义为: `md<filter>wd<filter>h<filter>m<filter>s<filter>` 其中:

- **md** - month days
- **wd** - week days
- **h** - hours
- **m** - minutes
- **s** – seconds

`<filter>` 用于指定其前缀的值 (日, 时, 分, 秒) 并被定义为: `[<from>[-<to>]][/<step>][,<filter>]` 其中:

- `<from>` 和 `<to>` 定义匹配值的范围（包括）。如果忽略 `<to>` ，则过滤器匹配 `<from> - <from>` 范围。如果 `<from>` 也被省略，则过滤器匹配所有可能的值。
- `<step>` 通过该范围定义数字值的跳过。默认情况下，`<step>` 的值为 1，这意味着所有定义范围的值都匹配。

虽然过滤器定义是可选的，但必须至少使用一个过滤器。过滤器必须有一个范围或定义的 *<step>* 值。

如果没有定义低级过滤器，则一个空的 filter 既与 “0” 匹配，又匹配所有可能的值。例如，如果省略小时过滤器，仅当分钟和秒的过滤器也被省略则只有 “0” 小时将匹配，否则空的小时过滤器将匹配所有小时值。

它们各自的过滤器前缀的有效 `<from>` 和 `<to>` 值分别为：

前缀描	*&l	;from>* *&l	;to>*
md	Month days	1-31	1-31
wd	Week days	1-7	1-7
h	Hours	0-23	0-23
m	Minutes	0-59	0-59
s	Seconds	0-59	0-59

`<from>` 值必须小于或等于 `<to>` 值。`<step>` 值必须大于或等于 1 且小于或等于 `<to>` - `<from>`。

单个数字月份、小时、分钟和秒值可以前缀为 0. 例如 `md01-31` 和 `h/02` 是有效间隔，但 `md01-031` 和 `wd01-07` 无效。

在 Zabbix 管理 Web 端，多个调度间隔以单独的行输入。在 Zabbix API 中，它们连接成单个字符串，以分号 `;` 作为分隔符。

如果时间匹配了几个间隔，则只执行一次。例如，`wd1h9;h9` 将在星期一上午 9 点执行一次。

示例:

间隔描	
m0-59	每分钟执行一次
h9-17/2	从 9:00 开始每 2 小时执行一次（9:00，11:00 ...）
m0,30 or m/30	在每小时的 hh:00 和 hh:30 执行
m0,5,10,15,20,25,30,35,40,45,50,55 or m/5	每 5 分钟执行
wd1-5h9	每周一至周五 9:00
wd1-5h9-18	每个星期一到星期五在 9：00,10：00，...，18:00
h9,10,11 or h9-11	每天上午 9:00，10:00 和 11:00
md1h9m30	每个月的第一天在 9:30
md1wd1h9m30	如果是星期一，每个月的第一天在 9:30 执行
h9m/30	在 9:00，9:30 执行
h9m0-59/30	在 9:00，9:30 执行
h9,10m/30	在 9:00，9:30，10:00，10:30 执行
h9-10m30	在 9:30，10:30 执行
h9m10-40/30	在 9:10，9:40 执行
h9,10m10-40/30	在 9:10，9:40，10:10，10:40 执行
h9-10m10-40/30	在 9:10，9:40，10:10，10:40 执行
h9m10-40	在 9:10，9:11，9:12，... 9:40 执行
h9m10-40/1	在 9:10，9:11，9:12，... 9:40 执行
h9-12,15	在 9:00，10:00，11:00，12:00，15:00 执行
h9-12,15m0	在 9:00，10:00，11:00，12:00，15:00 执行
h9-12,15m0s30	在上午 9 时 30 分，上午 10 时 30 分，11 时 30 分，12 时 30 分，15 时 30 分执行
h9-12s30	在 9:00:30，9:01:30，9:02:30 ... 12:58:30，12:59:30 执行
h9m/30;h10	在 9:00，9:30，10:00 执行

2 监控项类型

概述

监控项类型包含从系统获取数据的多种方式。每个监控项类型都有一组自己支持的监控项 key 和所需的参数。

以下监控项类型由 Zabbix 提供：

- Zabbix 代理检查
- SNMP 代理检查
- SNMP traps
- IPMI 检查
- 简单检查
- VMware 监控
- 日志文件监控

- 计算监控项
- Zabbix 内部检查
- SSH 检查
- Telnet 检查
- 外部检查
- 汇总检查
- 捕捉器监控项
- JMX 监控
- ODBC 监控
- 相关项目
- HTTP 检查

所有监控项类型的详细描述都包含在本章的各个小节中。即使监控项类型提供了大量的数据收集的方式，你还可以通过用户参数 或可加载模块进一步扩展数据收集方式。

一些监控检查由 Zabbix 服务器执行（称作无代理监控），而其它监控检查则需要 Zabbix agent 或者 Zabbix Java 网关（使用 JMX 监视）执行。

<note important> 如果特定的项目类型需要特定的接口（如 IPMI 检查需要主机上的 IPMI 接口），该接口必须存在于主机定义中。:::

可以在主机定义中设置多个接口：Zabbix agent，SNMP agent，JMX 和 IPMI。如果一个监控项使用多个接口，它将搜索可用的主机接口（按照以下顺序：Agent→SNMP→JMX→IPMI）直到找到连接的第一个匹配的接口。

返回文本的所有监控项（字符，日志，文本信息类型）都可以返回空格（如适用）和值设置为空的字符串。（2.0 版本后支持）

1 Zabbix 客户端

概述

这些检查与 Zabbix 代理进行通信实现数据的采集。

有被动和主动 两种 agent 模式. 在配置监控项时，你可以选择所需的类型：

- *Zabbix* 客户端 - 被动模式，Zabbix Server 向 Agent 索要数据
- *Zabbix* 客户端 *(*主动式*)* - 主动模式，Agent 主动上报数据给 Zabbix Server

支持的监控项 key

下表提供了可用的 Zabbix 代理监控项键值的详细信息。

请参考:

- 不同平台支持的监控项
- 只用于 Windows 的监控项 Key

** 必填和可选参数 **

没有尖括号的参数是强制性的。标有尖括号 **< >** 的参数是可选的。

键值				
	描述 *	返回值 ** ** 参数	* 注释	
agent.hostname	客户端主机名. Strin		从配置文件	回客户端主机名的实际值。
agent.ping	客户端可用性检查 Nothing	- 不可用 1 - 可用	使用 **nodata	)** 触发器函数检查主机不可用性。
agent.version	Zabbix 客户端的版本字符串		例如返回值:	<1.8.2
kernel.maxfiles				

键值	
	系统支持的打开文件的最大数量整数
kernel.maxproc	
	系统支持的最大进程数整数
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]	

键值				
	日志文件监控。Log	**file	* - 日志文件完整路径和名称 监控项必须定义为主动检查(/z**regexp** - 描述所需模式的正则表达式 如果文件丢失或权限不允 **encoding** - 编码标识符 **maxlines** - Agent 将发送到 Zabbix 服务器或代理的每秒最大行数。此参数覆盖zabbix_agentd.conf中的 “MaxLines-PerSecond” 值 如果 output 为空 - 返回包含匹配文本的 **mode** - 可能的值: *all* (默认值), *skip* - 跳过处理历史的数据（仅影响新创建的监控项）。在客户端端使用 output 参数提取内容。 **output** - 可选项，输出格式模板。\0 转义序列替换为匹配的文本，而\N（其中 N = 1 ... 9）转义序列被替换为第 N 个匹配组（如果 N 超过捕获组的数量，则为空字符串）。 **maxdelay** - 最大延迟（秒）。类型：float。值：0-（默认）不忽略日志文件行; > 0.0-忽略旧行，以便在 “maxde-	/manual/appendix/items访问，则监控项不受支持。行。请注意，除 “Result 为 TRUE” 之外的所有全局正则表达式类型始终返回整个匹配行，并忽略 output 参数。 => log[/var/log/syslog] => log[/var/log/syslog,error => log[/home/zabbix/logs/lo 使用 *output* 参数从日志记录中提取数字的示例： log[/app1/app.log,”task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors”„\1]- will match a log record ”2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors” and send only number 6080 to server. Because a number is being sent, the ”Type of information” for this log item can be changed from ”Log” to ”Numeric (unsigned)” and the

键值				
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]	日志文件监控中匹配行的数量。整数	**file** - 日志文件	整的路径和名称 该监控项必须配置为主动检查(/zh/manual/regexp - 正则表达式 如果文件 **encoding** - 编码标识符 **maxproclines** - Agent 将分析每秒最大行数。默认值为 10*'MaxLinesPerSecond' 在zabbix_agent配置文件. 查看更多信息在日志文件监控(lo**mode** - 可选的值: *all* (默认), *skip* - 跳过处理老数据(仅影响新创建的监控项)。 从 Zabbix 3.2.0 开始支持 **maxdelay** - 最大延迟秒数。类型: float. 值: 0 - (默认) 从不忽略每行日志; > 0.0 - 忽略旧行，以便在 "maxdelay" 秒内获取最近分析的行。在使用前请阅读maxdelay参数 的注解！	ppendix/items/activepas 失或权限不允许访问，则监控项不受支持 **regexp** _items).
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]				

键值				
	支持监控轮询的日志文件。Log	**file_reg	xp** - 文件名以及正则表达式定义的文件名的绝对路径。监控项必须配置为主动检查(/zh/manual/ **regexp** - 描述匹配内容的正则表达式。日志轮询是基于文件的最后 **encoding** - 编码标识符 **maxlines** - Agent 发送到 Zabbix 服务器或者 Proxy 服务器的每秒最大生成行数。此参数将重写配置文件zabbix_agent配置文件的参数'MaxLinesPerSecond'的值 如果 `output` 为空 - 返回包含匹配文本的整行. 请注意，除 "Resu**mode** - 可选的值: *all* (默认), *skip* - 跳过处理旧数据(仅影响新创建的监控项)。 在 Agent 端使用输出参数提取内容。 **output** - 一个可选的输出格式模板。**\0 转义序列替换为匹配文本，而\N（其中 **N = 1 ... 9**）转义序列被替换为第 **N** 个匹配组（如果 **N** 超过捕获组的数量，则为空字符串）。 ** **maxdelay** - 最大延迟	/items/activepassive#ac 改时间。 t 为 TRUE" 之外的所有全局正则表达式类型始终返回整个匹配行，并忽略输出参数。数。 /home/zabbix/logs/^logf 9]{1,3}$",,,100] → 将返回一个文件类似"logfile1" (不会匹配".logfile1") *_[0-9]{1,3}$","pattern_to_m 8",100] →* 将从文件收集信息例如*"logfile_abc_1"* 或者*"logfile__001". <i* 使用 `output` 参数从日志记录中提取数字的示例：* logrt[/app1/^test.* run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,\1]- will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only number 6080 to server.* 由于正在发送一个数字，因此该日志项的"信息类型"可以从"日志"更改为"数字（无正负）"，并且该值可以用于图形，触发

键值
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]

键值					
	支持对循环日志文件监控中匹配的行数。整型	**file_regexp** -	件名以及正则表达式定义的文件名的绝对路径。监控项必须定义为主动检查(/zh/manual/appendix/items/**regexp** - 描述匹配内容的正则表达式。日志轮询是基于文件的最后 **encoding** - 编码标识符 **maxproclines** - Agent 将分析每秒最大新生成行数。默认值为 4*'Max-LinesPer-Second' 定义在zabbix_agent配置文件. 更多信息请参考日志文件监控(log_it**mode** - 可能的值: *all* (默认), *skip* - 跳过处理旧数据(仅影响新创建的监控项)。`options` 参数从 Zabbix 4**maxdelay** - 最大延迟(秒)。类型：float。值：0-（默认）不忽略日志文件行; > 0.0-忽略旧行，以便在 "maxde-lay" 秒内获取最近分析的行。使用前请阅读maxdelay参数注释！ **options** - 日志文件轮换的类型. 可能的值：从 Zabbix 3.2.0 轮换(默认), *copytrun-*	ctivepassive#active_che 改时间。ms)。0 开始支持。开始支持。

键值					
net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]	检查 DNS 服务是否开启。0 - DNS 宕	(服务器没有响应或 DNS 解析失败）**ip** - DNS 服务器的 IP 地址 (默 1 - DNS 正在运行 **t	DNS 服务器为空，在 Windows 上被忽略） 示例: **name** - 要查询的 DNS 名称 =>pe** - 要查询的记录类型 (默认为 *SOA*) **timeout** (在 windows 上忽略) - 请求的超时秒数 (默认为 1 秒） `type` 可选的值为: **count** (在 windows 上忽略) - 请求的尝试次数 (默认为 2）*ANY*, *A*, ***protocol*** - 用于执行 *DNS* 查询的协议：udp* (默认) 或者 *tcp*	<net.dns[8.8.8.8,zabbix. S, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS* (Windows 系统除外), *HINFO*, *MINFO*, *TXT*, *SRV* 不支持国际化域名，请改用 IDNA 编码名称。 Zabbix 3.0 支持 `protocol` 参数。 Zabbix Agent 从版本 1.8.6（Unix）和 2.0.0（Windows）开始支持 SRV 记录类型。 Zabbix 2.0 之前命名（仍然支持）：*net.tcp.dns*
net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]					

键值					
		执行一个 DNS 查询字符串与所	类型的信息 **ip** - DNS 服务器的	P 地址（默认 DNS 服务器为空，在 Windows 上被忽略）示例: **name** - 要查询的 DNS 名称 =>**type** - 要查询的记录类型 (默认为 *SOA*) **timeout** (在 windows 上忽略) - 请求的超时秒数 (默认为 1 秒） type 可选的值为: **count** (在 windows 上忽略) - 请求的尝试次数 (默认为 2）*ANY*, *A*, ***protocol*** - 用于执行 *DNS* 查询的协议：udp* (默认) 或者 *tcp*	<net.dns[8.8.8.8,zabbix. S, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS* (Windows 系统除外), *HINFO*, *MINFO*, *TXT*, *SRV* 不支持国际化域名，请改用 IDNA 编码名称。 Zabbix 3.0 支持 protocol 参数。 Zabbix Agent 从版本 1.8.6（Unix）和 2.0.0（Windows）开始支持 SRV 记录类型。 Zabbix 2.0 之前命名（仍然支持）: *net.tcp.dns.query*
net.if.collisions[if]					
		Number of out-of-window collisions.	整型 *	if** - 网卡名称	
net.if.discovery					

键值					
		网络接口列表用于低级发现。JSON 对象		Zabbix agent 从 2	0 之后开始支持。 Zabbix agent 在 FreeBSD, OpenBSD 和 NetBSD 系统从 2.2 开始支持。 某些 Windows 版本（例如 Server 2008）可能需要安装最新的更新以支持网卡名称中的非 ASCII 字符。
net.if.in[if,<mode>]		网卡流入量统计。整型	**if** - 网卡完整描述或 IPv4 地址（Windows）在 Windows 上，该选项从 64 位计数器获取值 **mode** - 可用的值: *bytes* - 字节数 (默认）从 Zab*packets* - 包数量 *errors* - 错误数量 示例:*dropped* - 丢包数量 =&g*overruns (fifo)* - FIFO 缓冲区错误的数量 => n*frame* - 包帧错误的数量 *compressed* - 设备驱动程序发送或接收的压缩包数 你可以使用 net.if.dis*multicasts* - 设备驱动程序接收的组播帧数	卡名 (Unix);	如果可用）。64 位接口统计计数器在 Windows Vista 和 Windows Server 2008 中引入。如果 64 位计数器不可用，代理使用 32 位计数器。ix Agent1.8.6 版本起，支持 Windows 上的多字节接口名称。; net.if.in[eth0,errors] t.if.in[eth0] overy 或 net.if.list 监控项在 Windows 上获取网卡说明。 你可以使用该键与 *Delta*（每秒速度）存储值，以获得每秒字节的统计信息。

键值				
net.if.out[if,<mode>]	网卡流出量统计。整型	**if** - 卡名称 (Unix); 网卡完整描述或 IPv4 地址（Windows）在 Windows 上，该选项从 64 位计数器获取值（**mode** - 可用的值: *bytes* - 字节数 (默认）从 Zab*packets* - 包数量 *errors* - 错误数量 示例:*dropped* - 丢包数量 =&g*overruns (fifo)* - FIFO 缓冲区错误的数量 => n*collisions (colls)* - 在接口上检测到的冲突数 *carrier* - 设备驱动程序检测到的载波丢失数 你可以使用 net.if.di*compressed* - 设备驱动程序发送或接收的压缩包数	果可用）。64 位接口统计计数器在 Windows Vista 和 Windows Server 2008 中引入。如果 64 位计数器不可用，代理使用 32 位计数器。ix Agent1.8.6 版本起，支持 Windows 上的多字节接口名称。; net.if.out[eth0,errors] t.if.out[eth0] covery 或 net.if.list 监控项在 Windows 上获取网卡说明。你可以使用该键与 *Delta*（每秒速度）存储值，以获得每秒字节的统计信息。
net.if.total[if,<mode>]				

键值					
	网卡的进出流量统计信息的总和。 整型	**if** - 网卡名称 (Un	x); 网卡完整描述或 IPv4 地址（Windows）在 Windows 上，该选项从 64 位计数器获取值（如果可用）。 6**mode** - 可用的值: *bytes* - 字节数 (默认）示例: *packets* - 包数量 =&*errors* - 错误数量 =&g*dropped* - 丢包数量 *overruns (fifo)* - FIFO 缓冲区错误的数量 You may*compressed* - 设备驱动程序发送或接收的压缩包数	位接口统计计数器在 Windows Vista 和 Windows Server 2008 中引入。如果 64 位计数器不可用，代理使用 32 位计数器。 t; net.if.total[eth0,errors] ; net.if.total[eth0] obtain network interface descrip- tions on Windows with net.if.discovery or net.if.list items. 你可以使用 net.if.discovery 或 net.if.list 监控项在 Windows 上获取网卡说明。 你可以使用该键与 *Delta*（每秒速度）存储值，以获得每秒字节的统计信息。 请注意，只有当 net.if.in 和 net.if.out 都用于平台上丢弃的数据包时，丢弃的数据包才被支持。
net.tcp.listen[port]					

键值				
	检查此 TCP 端口是否处于监听状态。0 - 未监听	**port** - TCP 端口 1 - 处于监听状态	示例:	<=> net.tcp.listen[80] 在 Zabbix 代理版本 1.8.4 之后支持 Linux。 从 Zabbix 3.0.0 之后，在 Linux 内核 2.6.14 及更高版本上从内核的 NETLINK 接口获取有关监听 TCP 套接字的信息。否则，将从/proc/net/tcp 和/proc/net/tcp6 文件中检索该信息。
net.tcp.port[<ip>,port]	检查是否可以将 TCP 连接到指定的端口。0 - 不能连接	**ip** - IP 地址 (默认是 1 - 可以连接	27.0.0.1) 示例: **port** - 端口 =	<gt; net.tcp.port[,80] → 可用于测试在端口 80 上运行的 Web 服务器的可用性。 对于简单的 TCP 性能测试，使用 net.tcp.service.perf[tcp, 请注意，这些检查可能会导致增加系统守护程序日志文件中的额外信息（通常会记录 SMTP 和 SSH 会话）。 旧的命名方式: *check_port[*]*
net.tcp.service[service,<ip>,<port>]				

键值				
net.tcp.service.perf[service,<ip>,<port>]	检查服务是否正在运行并接受 TCP 连接。0 - 服务停止运行	**service** - 如下任一服务: 1 - 服务正在运行 **ip*	示例: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *.imap*, *tcp*, *https*, *telnet* (查看详细信息) =>- IP 地址 (默认是 127.0.0.1) **port** - 端口号 (默认为标准服务端口号) 请注意，这些检测可能会导	<net.tcp.service[ftp„45] → 可用于检测 FTP 服务器上 TCP 端口 45 的可用性。 增加系统守护程序日志文件的信息 (通常会记录 SMTP 和 SSH 会话)。 目前不支持检测加密协议（如端口 993 上的 IMAP 或端口 995 上的 POP)。一个解决方案是使用 net.tcp.port 来检测这些。 目前不支持 Windows 客户端检测 LDAP 和 HTTPS。 请注意，telnet 检测查找登录提示符（’：’在结尾)。 请参考 HTTPS 服务检测的已知问题。 *https* 和 *telnet* 服务从 Zabbix 2.0 开始支持。 旧命名: *check_service[*]*

键值					
		检测 TCP 服务性能 0 - 服	停止运行。 **service** seconds - 连接到服务花费的时间（秒） **ip** - IP	如下任一服务：示例: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (参考详细描述) => 址 (默认为 127.0.0.1) **port** - 端口号 (默认为标准服务端口号) 目前不支持检测加密协议	<net.tcp.service.perf[ssh → 可以用来检测 SSH 服务器的初始响应速度。如 IMAP 上的端口 993 或者 POP 上的端口 995) 。一个解决方案是使用 net.tcp.service.perf[tcp, 来检测。 目前不支持 Windows 代理检查 LDAP 和 HTTPS。 请注意，telnet 检测查找登录提示符（': ' 在结尾）。 请参考检测 HTTPS 服务的已知问题 *https* 和 *telnet* 服务从 Zabbix 2.0 开始支持。 旧名称: *check_service_perf[*]*
net.udp.listen[port]		检测 UDP 端口是否处于监听状态。 0 - 未监听。	**port** - UDP 端口 1 - 处在监听状态。	示例:	<=> net.udp.listen[68] 在 Linux 平台从 Zabbix agent version 1.8.4 开始支持。
net.udp.service[service,<ip>,<port>]					

键值				
	检查服务是否正在运行并能响应 UDP 请求。0 - 服务停止运行。	**service** - *ntp* (参考 1 - 服务正在运行 **por	详细信息）示例: **ip** - IP 地址 (默认是 127.0.0.1) => - 端口号 (默认使用标准服务端口号)	<net.udp.service[ntp„45 → 可用于测试 UDP 端口 45 上 NTP 服务的可用性。此选项从 Zabbix 3.0.0 起支持，但 ntp 服务可用于以前版本中的 net.tcp.service [] 选项。
net.udp.service.perf[service,<ip>,<port>]				
	检测 UDP 服务的性能 0 - 服务	止运行 **service** seconds - 等待服务响应的秒数 **port**	*ntp* (参考详细信息）示例: **ip** - IP 地址 (默认为 127.0.0.1) => 端口 (默认使用标准服务端口号)	<net.udp.service.perf[nt → 可用于测试 NTP 服务的响应时间。此选项从 Zabbix 3.0.0 起支持，但 ntp 服务可用于以前版本中的 net.tcp.service [] 选项。
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]				

键值					
		进程 CPU 利用率百分比。浮点型	**name** - 程名 (默认为 *all processes*) 示例: **user** - 用户名 (默认为 *all users*) =>**type** - CPU 利用率类型: =&gt*total* (默认), *user*, *system* **cmdline** - 可按命令行过滤（支持正则表达式) 返回值基于单 CPU 核的利用率。 **mode** - 数据收集模式: *avg1* (默认), *avg5*, *avg15* **zone** - 目标区域: *current* (默认), *all*. 此参数仅在 Solaris 平台上受支持。从 Zabbix 3.0.3 开始，如果代理程序已在 Solaris 上编译且没有区域支持，而是在支持区域的较新 Solaris 上运行，并且 <zone> 参数为缺省值或当前值，则代理程序将返回 NOTSUPPORTED (该代理程序不能将结果限制为仅当前区)。但是，在这种情况下，支持 <zone> 参数值 *all*。进程 CPU 利用率数据由收集器收集，该收集器最多支持 1024 个唯	<proc.cpu.util[,root] → 在 “root” 用户下运行的所有进程的 CPU 利用率。 proc.cpu.util[zabbix_serv → 在 zabbix 用户下运行的所有 zabbix_server 进程的 CPU 利用率。 如，使用两个内核的进程的 CPU 利用率为 200％。 自 Zabbix 3.0.0 起支持此 Key，并可在多个平台上使用 (请查看平台支持的监控项).

键值				
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]	用户进程使用的内存。 整型	**name** - 程名 (默认是全部进程) 示例: **user** - 用户名 (默认是全部用户) => pro**mode** - 可能的值: =&g*avg*, *max*, *min*, *sum* (默认值) =&**cmdline** - 按命令行过滤（它是一个正则表达式） **memtype** - 进程使用的内存类型 注意: 当多进程使用共享内存时，进程使用的内存总和可能导致大到不切实际的值。 参考说明 关于选择进程 `name` 和 `cmdline` 参数 (指定为 Linux)。 `memtype` 参数从 Zabbix 3.0.0 开始在多个平台支持。	<.mem[,root] → “root” 用户运行的所有进程使用的内存 ; proc.mem[zabbix_server → zabbix 用户运行的所有 zabbix_server 进程使用的内存 t; proc.mem[,oracle,max,c → oracle 用户下，包含有 oracleZABBIX 命令行运行的所有内存最多的内存
proc.num[<name>,<user>,<state>,<cmdline>]				

键值					
	进程数量。整型	**name	* - 进程名称 (默认是 *all processes*) 示例: **user** - 用户名 (默认是 *all users*) =>**state** - 可选的值: =&g 所有状态 (默认), =>*disk* - 不间断休眠, *run* - 运行中, 参考 *sleep* - 间断休眠, *trace* - 停止的, 在 W*zomb* - 僵尸 **cmdline** - 按命令行过滤（它是一个正则表达式）自 Zabbix 3.4.0 起支持	<proc.num[,mysql] → 在 mysql 用户下运行的进程数 ; proc.num[apache2,www data] → 在 www-data 用户下运行的 apache2 进程数 proc.num[,oracle,sleep,c → 在 ora- cleZABBIX 命令行下的 oracle 用户运行的睡眠状态进程数。 说明 关于选择进程 `name` 和 `cmdline` 参数 (适用于 Linux). ndows 上，只支持 `name` 和 `user` 参数。 `state` 参数的磁盘和跟踪值。
sensor[device,sensor,<mode>]	硬件传感器读数。浮点型	**device**	- 设备名称 在 Linux 2.4 上读取 **sensor** - 传感器名 **mode** - 可能的值: 示例:*avg*, *max*, *min* (如果省略此参数，则会对设备和传感器进行逐字处理). => sen- sor[w83781d-	proc/sys/dev/sensors 2c-0- 2d,temp1] 在 Zabbix 1.8.4 之前，使用传感器 *[temp1]* 格式。 在 Linux 2.6 以后的版本上读取 /sys/class/hwmon 请参阅 Linux 上sensor项目的更详细说明。

键值				
				在 OpenBSD 上读取 *hw.sensors* MIB 文件 示例: => sensor[cpu0,temp0] → CPU 的温度 => sensor[”cpu[0-2]$”,temp,avg] → 前三个 CPU 温度的平均值 从 Zabbix 1.8.4 开始支持 OpenBSD。
system.boottime	系统启动时间整数 (U	ix 时间戳)		
system.cpu.discovery	检测到的 CPU/CPU 内核列表。用于低级发现。JSON 对象		所有平台从 2.4.0 开始支持	
system.cpu.intr	设备中断数整数			
system.cpu.load[<cpu>,<mode>]	CPU 负载. 浮	数 **cp	** - 可能的值: 示例: *all* (default), *percpu* (总负载除以在线 CPU 数) => s**mode** - 可能的值: *avg1* (一分钟平均值, 默认值), *avg5*, *avg15* *percpu*	<stem.cpu.load[,avg5] 从 Zabbix 2.0.0 开始支持 旧名称: *system.cpu.loadX*
system.cpu.num[<type>]	CPU 的数量整数	**ty	e** - 可能的值: 示例: *online* (默认), *max* =	<gt; system.cpu.num
system.cpu.switches	上下文交换的数量。整数		旧名称: *syst	m[switches]*
system.cpu.util[<cpu>,<type>,<mode>]				

键值				
	CPU 利用率。浮点型	**cpu*	- *<CPU 数量>* 或者 *all* (默认值) 示例: **type** - 可能的值: =&g*idle*, *nice*, *user* (默认值), *system* (Windows 系统默认值), *iowait*, *interrupt*, *softirq*, *steal*, *guest* (在 Linux kernels 2.6.24 以及以上支持), *guest_nice* (在 Linux kernels 2.6.33 以及以上支持) **mode** - 可能的值: 旧名称 *avg1* (1 分钟平均值, 默认值), *avg5*, *avg15*	<; system.cpu.util[0,user,avg5 *system.cpu.idleX*, *system.cpu.niceX*, *system.cpu.systemX*, *system.cpu.userX*
system.hostname[<type>]				

键值				
	系统主机名。 字符串型	**type**	仅 Windows 不得在其它系统上使用) - 可能的值: *netbios* (默认) 或者 *host* 该值由 Windows 上的 GetComputerName (	(对于 **netbios**) 或 gethostname () (用于 **host**) 函数以及其它系统上的 “hostname” 命令获取。 返回值示例: *Linux* 系统: => system.hostname → linux-w7x1 => system.hostname → www.zabbix.com *Windows* 系统: => system.hostname → WIN-SERV2008-I6 => system.hostname[host] → Win-Serv2008-I6LonG 参数 `type` Zabbix **1.8.6** 开始支持。 请参考更详细的描述.
system.hw.chassis[<info>]				

键值					
		机架信息。字符串	**info*	- 完整的 (默认)、型号、序列、类型或供应商之一示例: system.hw.chassis[ty	] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] 此 key 取决于SMBIOS表的可用性。将尝试从 sysfs 读取 DMI 表，如果 sysfs 访问失败，尝试直接从内存中读取。 需要 **Root** 权限，因为通过从 sysfs 或内存读取获取该值。 Zabbix agent 从 2.0 开始支持。
system.hw.cpu[<cpu>,<info>]		CPU 信息字	串或者整型 **cpu**	<*CPU* 数量> 或者 全部 (默认) 示例: **info** - 可能的值：=&gt*full* (默认), *curfreq*, *maxfreq*, *model* 或者 *vendor*	<system.hw.cpu[0,vendo → Authenti-cAMD 从 /proc/cpuinfo 和 /sys/devices/system/cpu 获取信息。 如果指定了 CPU 编号和 *curfreq* 或 *maxfreq*，则返回数值 (Hz)。 Zabbix agent 从版本 2.0 开始支持
system.hw.devices[<type>]					

键值				
	列出 PCI 或者 USB 设备文本型	**type**	- *pci* (默认) 或者 *usb* 示例:	<=> system.hw.devices[pci] → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [..] 返回 lspci 或 lsusb 实用程序的输出（没有任何参数） Zabbix agent 从版本 2.0 开始支持
system.hw.macaddr[<interface>,<format>]	列出 MAC 地址字符串	**inter	ace** - *all* (默认) 或者为一个正则表达式 列出与给定 `interface` 正 **format** - *full* (默认) 或者 *short*	表达式名称匹配的网卡的 MAC 地址（所有网卡的所有列表）。 示例: => system.hw.macaddr[”eth0$ → [eth0] 00:11:22:33:44:55 如果 `format` 被指定为 short，则不会列出接口名称和相同的 MAC 地址。 \\Zabbix agent 从版本 2.0 开始支持。
system.localtime[<type>]				

键值				
	系统时间整数	type 为 *utc* **type 字符串 - type 为 *local*	* - 可能的值：此监控项参数从 Zab*utc* - (默认值) 从纪元以来的时间（1970 年 1 月 1 日 00:00:00 UTC），以秒为单位。\ *local* - 'yyyy-mm-dd，hh：mm：ss.nnn，+hh：mm' 格式的时间示例:	ix agent 版本 2.0 开始支持。 => system.localtime[local] → 使用该 key 创建一个监控项，然后使用它在时钟screen element中显示主机时间。
system.run[command,<mode>]	在主机上运行指定的命令。命令执行的文本结果	**command** - 要执行的命令 1 - mode 为 nowait（不管命令结果如何）*wait* - 等	最多可以返回 512KB 的数据，包括截断的尾随空格。 **mode** - 可能的值：要被正执行结束（默认），*nowait* - 不等待示例	< 的处理，命令的输出必须是文本。 => system.run[ls -l /] → 根目录的详细文件列表。 注意: 要启用此功能，Zabbix agent 配置文件 必须包含 EnableRemoteCommands=1 选项。 监控项的返回值是标准输出以及由命令产生的标准错误输出。如果没有使用 nowait 标志，则会检查执行结果。 从 Zabbix 2.4.0 开始，空结果是允许的。 同时参考: 执行指令.
system.stat[resource,<type>]				

键值			
	系统信息。整型或者	点型 **ent** - 该	区有权接收的处理器单元数（float） **kthr,<type>** - 关于内核线程状态的信息: *r* - 平均可运行内核线程数（float） *b* - 虚拟内存管理器等待队列中的平均内核线程数（float） **memory,<type>** - 有关虚拟和真实内存使用情况的信息： *avm* - 活动虚拟页面（整数） *fre* - 自由列表的大小（整数） **page,<type>** - 关于页面错误和分页活动的信息： *fi* - 每秒文件页面输入（float） *fo* - 每秒文件页面输出（float） *pi* - 从调页空间（float）分页的页面 *po* - 页面分页到调页空间（float） *fr* - 页面被释放（页面替换）（浮点） *sr* - 通过页面替换算法扫描的页面（float） **faults,<type>** - trap 和中断率: *in* - 设备中断 (float) *sy* - 系统调用 (float) *cs* - 内核线程上下文切换 (float) **cpu,<type>** - 处理器时间使用百分比的细分: *us* - 用户时

键值				
system.sw.arch	软件架构信息。字符串型		示例:	<=> system.sw.arch → i686 信息从 uname () 函数中获取。 Zabbix agent 从版本 2.0 开始支持
system.sw.os[<info>]	操作系统信息字符串	**info**	- 可能的值: 示例: *full* (默认), *short* 或者 *name* =&g	<; system.sw.os[short]→ Ubuntu 2.6.35-28.50-generic 2.6.35.11 信息获取 (注意，并非所有发行版中都存在所有文件和选项): /proc/version (*full*) /proc/version_signature (*short*) /etc/os-release 中支持它的系统上的 PRETTY_NAME 参数， 或/etc/issue.net (name) Zabbix agent 从版本 2.0 开始支持。
system.sw.packages[<package>,<manager>,<format>]				

键值					
	列出已安装的软件包。 文本	**package**	- *all* (默认) 或者为正则表达式 列表（按字母顺序）安装的包名称与给定的包r**manager** - *all* (默认) 或者为包管理器 **format** - *full* (默认) 或者 *short* 示例:	gexp 匹配的包（全部列出它们全部）。 => system.sw.packages[mini,d → python-minimal, python2.6-minimal, ubuntu-minimal 支持包管理器 (执行命令): dpkg (dpkg --get-selections) pkgtool (ls /var/log/packages) rpm (rpm -qa) pacman (pacman -Q) 如果 `format` 被指定为 *full*，则软件包由包管理器分组（每个管理器在单独的行上以其方括号开头）。 如果 `format` 被指定为 *short*, 包管理器不分组，并列在一行里。 Zabbix agent 从版本 2.0 开始支持
system.swap.in[<device>,<type>]					

键值					
		交换（从设备到内存）统计。整型	**device** - 用	交换的设备（默认是 *all*）示例: **type** - 可能的值: =&gc*ount* (swapins 的数量), *sectors* (换入的区域), *pages* (换入的页). 有关默认的详细信息请参考支持的平台	<; sys-tem.swap.in[,pages] 这个信息的来源是: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.out[<device>,<type>]					
		交换（从内存到设备）统计。整型	**device** - 用	交换的设备（默认是 *all*）示例: **type** - 可能的值: =&gc*ount* (swapouts 的数量), *sectors* (换出的区域), *pages* (换出的页). 有关默认的详细信息请参考支持的平台	<; sys-tem.swap.out[,pages] 信息来源是: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.size[<device>,<type>]					

键值					
		交换空间大小（以字节为单位）或百分比（total）。 Integer - 字节	**device** - 用于交换的设备（ Float - 百分比 *f	认是 *all*）示例: **type** - 可能的值: =&gee*（可用的交换空间, 默认值), *pfree*（空闲交换空间，百分比), *pused*（使用交换空间，百分比), *total*（总交换空间), *used*（使用交换空间）	<; system.swap.size[,pfree] → 空闲 swap 空间百分比 如果没有指定 设备，Zabbix 代理只会考虑交换设备（文件），物理内存将被忽略。例如，在 Solaris 系统上，swap -s 命令包含一部分物理内存和交换设备（与 swap -l 不同）。 请注意，此 key 可能会报告虚拟化（VMware ESXi，VirtualBox）Windows 平台上的百分比不正确。在这种情况下，使用 perf_counter[\700（_Total）\702] 键来获取正确的交换使用数据。 旧名称: *system.swap.free*, *system.swap.total*
system.uname					

键值				
		系统相关信息字符串	返回值的示例 (U	ix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 返回值示例 (Windows): Windows ZABBIX-WIN 6.0.6001 Microsoft? Windows Server? 2008 Standard Service Pack 1 x86 从 Zabbix 2.2.0 开始在 Unix 上，该监控项的值是通过 uname() 系统调用获得的。以前它是通过调用 “uname -a” 获得的。此监控项的值可能与 “uname -a” 的输出不同，并且不包含基于其它来源输出的 “uname -a” 的信息。 从 Zabbix 3.0 开始的 Windows 系统上，该监控项的值是从 Win32_OperatingSystem 和 Win32_Processor WMI 类获取信息。以前它是从不稳定的 Windows API 和未记录的注册表项获得的。操作系统名称（包括版

键值			
system.uptime	系统正常运行时间（以秒为单位）整数	在监控项配置(/zh/man	al/config/items/item#co 中，使用 **s** 或者 **uptime** 单位来获取可读取的值。
system.users.num	已登录用户数整数	**who**	命令用于代理端获取该值。
vfs.dev.read[<device>,<type>,<mode>]			

键值				
	磁盘读取统计信息。整数 - type 为 *sectors*, *operations*, *bytes* **device** Float - type 为 *sps*, *ops*, *bps*	磁盘设备（默认为 *all*）不同操作系统的“类型”参数的默认值：**type** - 可能的值: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* AIX 须指定此参数，因为各种操作系统的默认值不同。FreeBSD - bps *sps*, *ops*, *bps* 表示: sectors, operations, bytes per second, respectively. L**mode** - 可能的值: *avg1* (1 分钟平均值, 默认), *avg5*, *avg15*. OpenBSD - 此参数仅支持的类型为: sps, ops, bps. Solaris -	<- operations nux - sps perations bytes 示例: => vfs.dev.read[,operations] 在支持的平台上的 *sps*, *ops* 和 *bps* 曾被限制为 8 个设备 (7 个独立的和 1 个 *all*). 从 Zabbix 2.0.1 开始，这个限制提高到 1024 个设备 (1023 个独立的和 1 个 *all*). 如果默认为全部用于第一个参数，那么该 key 将返回摘要统计信息，包括所有块设备，如 sda，sbd 及其分区（sda1，sda2，sdb3 ...）和基于这些块设备/分区的多个设备（MD raid）和基于这些设备/分区的逻辑卷（LVM）。在这种情况下，返回值只能作为相对值（动态时间）而不是绝对值。 LVM 的支持从 Zabbix 1.8.6 开始。 直到 Zabbix 1.8.6 才能使用相关的设备名称（例如，sda）。从那

键值
vfs.dev.write[<device>,<type>,<mode>]

键值			
	磁盘写入统计信息。整数 - type 为 *sectors*, *operations*, *bytes* **device** 浮点型 - type 为 *sps*, *ops*, *bps* 因为各	磁盘设备 (默认为 *all*) 不同操作系统的“类型” 参数的默认值： **type** - 可能的值: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* AIX 操作系统的默认值有所不同，所以这个参数必须被指定。 FreeBSD - bps *sps*, *ops*, *bps* 代表: sectors, operations, bytes per second, respectively. L**mode** - 可能的值: *avg1* (1 分钟平均值, 默认), *avg5*, *avg15*. OpenBSD -此参数仅支持这些类型: sps, ops, bps. Solaris -	<- operations nux - sps perations bytes 示例: => vfs.dev.write[,operations] *sps*, *ops* and *bps* 在支持的平台上的 *sps*, *ops* 和 *bps* 曾被限制为 8 个设备 (7 个独立的和 1 个 *all*). 从 Zabbix 2.0.1 开始，这个限制提高到 1024 个设备 (1023 个独立的和 1 个 *all*)。 如果默认为全部用于第一个参数，那么该 key 将返回摘要统计信息，包括所有块设备，如 sda，sbd 及其分区（sda1，sda2，sdb3 ...）和基于这些块设备/分区的多个设备（MD raid）和基于这些设备/分区的逻辑卷（LVM）。在这种情况下，返回值只能作为相对值（动态时间）而不是绝对值。 LVM 的支持从 Zabbix 1.8.6 开始。 直到 Zabbix 1.8.6 才能使用相关的设备名称

键值
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>

键值					
	目录条目计数. 整数	**dir**	- 目录的绝对路径 例如%APP_HOME%类的环境 - 正则表达式，描述包含的文件，目录和符号链接名称模式（包括所有文件，目录和符号链接，如果为空; 空字符串是默认值） **regex_excl** - 正则表达式描述文件，目录和符号链接名称模式以排除（不排除任何如果为空; 空字符串是默认值）隐藏目录”.” and ”..” 不会被计算。 **types_incl** - 要计算的一组目录条目类型，可能的值：*file* - 常规文件, *dir* - 子目录, *sym* - 符号链接, *sock* - 套接字, *bdev* - 块设备, *cdev* - 字符设备, *fifo*- FIFO, *dev*- ”bdev,cdev” 的同义词, *all*- 所有上述类型，即”file,dir,sym,sock,bdev,cdev,fifo”. 如果参数为空，all 为默认值。必须用逗号分隔多个类型，并用引号 “ ” 括起整个集合。 目录遍历不遵循符号链接。 **types_excl** - 要计数的一组目录条目类型，与 <types_incl> 相同的值和语法。如果	$HOME 和%TEMP% 子类的环境变量是不支持的。 对硬链接计数一次。(https://en.wikipedia.org (PCRE). `regex_incl` 和 `regex_excl` 在计算条目大小时都应用于文件和目录，但在选择遍历的子目录时会被忽略（如果 regex_incl 是 “(?i)^.+\.zip$” 并且未设置 max_depth，则将遍历所有子目录，但只会计算 zip 类型的文件）。如果文件名与 `regex_incl` 和 `regex_excl` 匹配，则不会计算此文件。 回任何数据，并且该项目将标记为 “不支持”。部分计数不会被退回。 按大小过滤时，只有常规文件具有有意义的大小。在 Linux 和 BSD 下，目录也具有非零大小（通常为几 Kb）。设备的大小为零，例如 **/dev/sda1** 的大小不反映相应的分区大小。因此，当使用 `<min_size>` 和 `<max_size>`

键值
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>]

键值					
		目录大小(以字节为单位)。整数	**dir** - 目录的绝	路径 仅计算具有 zabbix 用户读取权限的目录。 **regex_incl** - 正则表达式描述包含的文件名模式（如果为空则包括所有文件; 空字符串是默认值） **regex_excl** - 正则表达式描述用于排除的文件名模式（如果为空不排除任何文件; 空字符串是默认值） 在 Windows 上，将跳过任何符号链接，并且仅将硬链接考虑在内。 **mode** - 可能的值: *apparent* (默认) - 获得明确的文件大小，而不是磁盘利用率 (作为 `du -sb dir`), *disk* - 获取磁盘使用情况 (作为 `du -s -B1 dir`). 和 du 命令不同，vfs.dir.size 监控项在计算目录大小时会将隐藏的文件记录帐户 (作为 `du -sb .[^.]* *` 在 dir 内). **max_depth** - 要遍历的子目录的最大深度。**-1** (默认) - 无限, **0** - 不会遍历到子目录。 示例:	< 对于大型目录或慢速驱动器，由于客户端和服务端/代理 配置文件中的超时设置，此项可能会超时。根据需要增加超时值。 ? vfs.dir.size[/tmp,log] - 计算/tmp 中包含 “log” 的所有文件的大小 ? vfs.dir.size[/tmp,log,^.+ - 计算/tmp 中包含 “log” 的所有文件的大小，不包括包含’.old’ 的文件 文件大小限制取决于大文件支持。

键值				
vfs.file.cksum[file]	文件 checksum 校验，由 UNIX cksum 算法计算实现。整型	**file** - 文件全	径示例:	<=> vfs.file.cksum[/etc/passw 返回值示例: 1938292000 旧名字: *cksum* 文件大小限制取决于大文件支持。
vfs.file.contents[file,<encoding>]	检索文件的内容。文本	**file**	文件全路径 如果文件为空或仅包含 LF/C**encoding** - 编码页标识符	字符，则返回空字符串。 Example: => vfs.file.contents[/etc/pas 此选项对文件限制是不超过 64KB 的文件。 Zabbix agent 从版本 2.0 开始支持。
vfs.file.exists[file]	检测文件是否存在。0 - 不存在	**file** - 1 - 常规文件或到常规存在文件的 link（符号或硬）	件的全路径 示例:	<=> vfs.file.exists[/tmp/appli 返回值取决于 S_ISREG POSIX 宏返回的值。 文件大小限制取决于大文件支持。
vfs.file.md5sum[file]	文件的 MD5 checksum。字符串	文件的 MD5 哈希) **file** -	件的全路径 示例:	<=> vfs.file.md5sum[/usr/loc 返回值示例: b5052decb577e0fffd622 此项目的文件大小限制（64 MB）在 1.8.6 版中已删除。 文件大小限制取决于大文件支持。
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]				

键值					
		查找文件中的字符串。包含匹配字符串的行	或由可选输出参数指定的行。**file** - 文件完整路径	只返回第一个匹配行。\\如果没有行与表达式匹配，则返回空字符串。**regexp** - Perl Compatible Regular Expression（PCRE）或 POSIX 在 Zabbix 3.4 之前扩展了正则表达式 **encoding** - 编码页标识符 使用 o**start line** - 要查询的第一行的数量（默认为文件的第 1 行）。\ **end line** - 要查询的最后一行的数量（默认为文件的最后一行）。\start line,end line**output** - 一个可选的输出格式模板。\0 转义序列替换为匹配的文本，而\N（其中 N = 1 ... 9）转义序列被替换为第 N 个匹配组（如果 N 超过捕获组的数量，则为空字符串）。	<tput 参数的提取过程发生在代理端。和 output 参数从版本 2.2 开始支持。示例: => vfs.file.regexp\[/etc/pass => vfs.file.regexp\[/path/to/ 9\]+)$”„3,5,\\1\] => vfs.file.regexp\[/etc/pass 9\]+)„„\\1\] → 获取用户 *Zabbix* 的 ID| |vfs.file.regmatch\[file,re line>,<end line>\]|<|<|<|<| |<| 查询文件中的字符串。0 - 不匹配 |**file** - 文 1 - 匹配 *| 全路径\start line,en**regexp** - Perl Compatible Regular Expression（PCRE）或 POSIX 在 Zabbix 3.4 之前扩展了正则表达式 encoding** - 编码页标识符 示例: **start line** - 满足查询到的第一行的数量（默认为文件的第 1 行）。 => vfs.file.regmatc**end line** - 要查询的最后一行的数量（默认为文件的最后一行）。

Note:
一个特定于 Linux 的注意事项。Zabbix Agent 必须具有权限读取文件系统*/proc*。来自 www.grsecurity.org 的内核补丁限制非特权用户的访问权限。

可用的编码

`encoding` 参数用于指定处理相应监控项检查的编码，以便获取的数据不会被破坏。有关支持的编码（代码页标识符）的列表，请参阅相应的文档，例如libiconv（GNU Project）或 Microsoft Windows SDK 文档 “代码页标识符” 的文档。

如果传递空 `encoding`，则默认使用 UTF-8（用于较新的 Unix/Linux 发行版的默认语言环境，请参阅系统设置）或使用具有系统特定扩展名（Windows）的 ANSI。

关于监控项的一些疑难问题

1. 如果与 passive agent 一起使用，服务器配置中的 超时值可能需要高于代理配置文件中的 超时值。否则，该监控项可能无法获取任何值，因为服务器请求代理程序首先超时。

Windows 的监控项键值

监控项 Key

该表仅描述了 Zabbix Windows Agent 可用的监控项键值的详细信息。

键值			
	描述返	值参数	注释
eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]			

键值				
	事件日志监控。日志	**name**	- 事件日志的名称 该项目必须配置为主动检查(**regexp** - 所需模式的正则表达式 **severity** - 正则表达式描述的严重程度 示例: 此参数接受以下值: *"Information"*, *"Warning"*, *"Error"*, *"Critical"*, *"Verbose"* (从 Zabbix 2.2.0 开始支持，在 Windows Vista 或更高版本上运行) => event-log[Applic**source** - 源标识符的正则表达式 (从 Zabbix 2.2.0 开始支持正则表达式) => event-log[Secur**eventid** - 事件标识符的正则表达式 => even**maxlines** - Agent 将发送到 Zabbix Server 或 Proxy 的每秒最大新生成行数。此参数覆盖zabbix_agentd.win.conf中“MaxLines-PerSecond”的值 => event-log[System,**mode** - 可能的值: =&g*all* (默认), *skip* - 跳过处理旧数据（仅影响新创建的监控项)。	zh/manual/appendix/iter tion] ty„"Failure Au- dit"„^(529|680)$] log[System„"Warning\|Er „^1$] ; event- log[System„„@TWOSHO - 这里引用一个名为 TWOSHORT 的自定义正则表达式 (定义为 *Result is TRUE* 类型，表达式本身为 ^1$\|^70$)。 从 Zabbix 2.0.0 开始支持参数 mode 。 从 Zabbix 2.2.0 开始支持“Windows Eventing 6.0”。 请注意，为此选项选择非日志信息类型将导致本地时间戳记以及日志严重性和源信息的丢失。 另请参阅日志文件监控.

键值				
net.if.list	网卡列表(包括接口类型，状态，IPv4 地址，描述)。文本型		Zabbix agent 从版本 1.8.1 之后支持	从 Zabbix Agent 版本 1.8.6 起支持的多字节网卡名称。已禁用网卡不会列出。 请注意，启用/禁用某些组件可能会在 Windows 界面名称中更改其排序。 某些 Windows 版本（例如 Server 2008）可能需要安装最新的更新以支持网卡名称中的非 ASCII 字符。
perf_counter[counter,<interval>]	Windows 性能计数器的值。整数，浮点，	字符串或者文本 (取决于请求) **counter** - 计数器的路径	性能监视器可用于获取可用计数器列表。在版本 1.6 之前，此 **interval** - 最后 N 秒用于存储平均值。 The `interval` 必须在 1 到 900（包含）秒之间，默认值为 1。 请参考: [Windows 性能计	数仅为仅需要一个样本的计数器(如 \System\Threads）返回正确的值。对于需要更多样本的计数器(如 CPU 利用率)，它将无法正常工作。从 1.6 开始，可以使用 `interval` ，因此检查每次返回最后 “间隔” 秒的平均值。 器](/zh/manual/config/ite
proc_info[process,<attribute>,<type>]				

键值					
		关于具体进程的各种信息。浮点型	**process** - 程名 支持以下 属性: **attribute** - 所需的进程属性 *vm-siz**type*** - 表现类型（当具有相同名称的多个进程存在时有意义） wkset* - 进程工作集（进程使用的物理	<* (默认) - 进程虚拟内存的大小（以 KB 为单位） 存量）的大小（KB） *pf* - 页面错误数量 *ktime* - 进程内核时间（以毫秒为单位） *utime* - 以毫秒为单位进程用户时间 *io_read_b* - I/O 操作中进程读取的字节数 *io_read_op* - 由进程执行的读操作数 *io_write_b* - I/O 操作过程中进程写入的字节数 *io_write_op* - 由进程执行的写操作数 *io_other_b* - 在读操作和写操作之外的操作期间由进程传送的字节数 *io_other_op* - 通过进程执行的 I/O 操作数量，而不是读取和写入操作 *gdiobj* - 进程使用的 GDI 对象数 *userobj* - 进程使用的 USER 对象数 有效的 type 是: *avg* (default) - 名为 <process> 的所有进程的平均值 *min* - 名为 <process> 的所有进程的最小值 *max* - 名为 <process>

键值				
service.discovery	Windows 服务列表。用于低级别发现. JSON 对象		Zabbix agent 从	本 3.0 以后支持
service.info[service,<param>]	有关服务的信息。整型 - 使用	param 为 *state*, *startup* **service** - String - 使用的 param 是 *displayname*, *path*, *user* Text - 使用的 param 是 *description* 特定的状态: 0 - 运行, 1 - 暂停, 2 - 开始等待, 3 - 暂停等待, 4 - 继续等待, 5 - 停止等待, 6 - 停止, 7 - 未知, 255 - 没有这样的服务 专用于 启动: 0 - 自动的, 1 - 自动延迟, 2 - 手动, 3 - 禁用, 4 - 未知	一个真实的服务名称或显示名称，如 MMC 服务管理单元所示 示例: **param** - *state* (默认), *displayname*, *path*, *user*, *startup* 或者 *description* =&g=&g 请自 Za	<; service.info[SNMPTRAP] - SNMPTRAP 服务的状态; service.info[SNMPTrap] - 同一服务的状态，但指定了显示名称 => service.info[EventLog,startu - 事件日志服务的启动类型 service.info[service，state] 和 service.info[service] 将返回相同的信息。意，只有使用 param 作为 状态，此选项将返回不存在的服务（255）。bix 3.0.0 起支持此监控项。不建议使用旧的 service_state[service] 选项。
services[<type>,<state>,<exclude>]				

键值					
		服务列表 0 -	如果为空 **type* 文本 - 由换行符分隔的服务列表 **exclude**	- *all* (默认), *automatic*, *manual* 或者 *disabled* 示例: **state** - *all* (默认), *stopped*, *started*, *start_pending*, *stop_pending*, *running*, *con-* *tinue_pending*, *pause_pending* 或者 *paused* =&g 从结果中排除的服务。排除的服务应以双引号列出，用逗号分隔，不含空格。 => ser- vices[automatic, stopped] - 应该运行却已停	<; ser- vices[,started] - 启动的服务列表 服务的列表 => ser- vices[automatic, stopped, ”ser- vice1,service2,service3” - 应该运行却停止服务列表, 不包括的服务名称 service1, service2 和 service3 参数 exclude 从 Zabbix 1.8.1 开始被支持。
wmi.get[<namespace>,<query>]		执行 WMI 查询并返回第一个选定的对象。整型, 浮点, 字符串或者文本	取决于请求) **names- pace** - WMI 名字空间	示例: **query** - WMI 查询返回单个对象 => w	<i.get[root\cimv2, 选择类似名为’%PHYSICALDRIVE0%’ 的 Win32_DiskDrive 的状态] [’%PHYSI- CALDRIVE0%’] - 返回第一个物理磁盘的状态。 从 Zabbix 2.2.0 开始支持此 Key
vm.vmemory.size[<type>]					

键值				
	虚拟空间大小（以字节计）或百分比（总计）。整型 - 用于字节	**type** - 可能的值: 浮点 - 用于百分比	示例: *available* (虚拟内存的可用性), *pavailable* (可用的虚拟内存百分比), *pused* (使用虚拟内存，百分比), *total* (总虚拟内存，默认), *used* (已用的虚拟内存) => vm.vmemory.size[available] → 可用的虚	< 内存，百分比 虚拟内存统计信息基于: Zabbix 代理可以提交的最大内存量。 系统或 Zabbix 代理的当前提交内存限制，以较小者为准。 Zabbix 3.0.7 和 3.2.3 支持此 Key。

监控 Windows 服务

本教程提供了 Windows 服务监控配置说明。以下假设 Zabbix 服务器和代理已配置并可操作。

Step 1

获取服务名称。

你可以通过转到 MMC 服务管理单元并显示服务的属性来获取该名称。在 “常规” 选项卡中，将看到一个名为 “服务名称” 的字段。下面的值是设置监控项时使用的名称。

例如，如果要监控”workstation” 服务，那么你的服务可能是：** lanmanworkstation **。

Step 2

配置一个监控项 用于监控服务.

监控项 service.info[service,<param>] 检索有关特定服务的信息。根据你需要的信息，指定 *param* 选项接受以下值: *displayname*, *state*, *path*, *user*, *startup* 或者 *description*. 默认值是 *state* 如果 *param* 没有指定 (service.info[service]).

返回值的类型取决于选择的 *param*: 整数用于 *state* 和 *startup*; 字符串用于 *displayname*, *path* 和 *user*; 文本用于 *description*.

示例:

- 键值: service.info[lanmanworkstation]
- 信息类型: Numeric (unsigned)
- 查看值: 选择 *Windows service state* 值映射

两个值映射可用 *Windows service state* 和 *Windows service startup type* 将数值映射到前端中的文本表示。

Windows 服务的发现

低级别发现 提供了一种在计算机上为不同实体自动创建项目、触发器和图形的方法。Zabbix 可以自动开始监控机器上的 Windows 服务，无需知道服务的确切名称，也可以手动创建每个服务的项目。过滤器可用于仅为感兴趣的服务生成实际监控项、触发器和图形。

2 SNMP 代理

概述

你可能希望在启用 SNMP 的设备（如打印机、交换机、路由器或 UPS）上使用 SNMP 监控，因为在这些设备上尝试安装完整的操作系统和 Zabbix 代理是不可能的。

为了能够监控 SNMP 代理在这些设备上提供的数据，Zabbix 服务器初始化配置时必须具有 SNMP 支持。

仅通过 UDP 协议执行 SNMP 检查。

从 Zabbix 2.2.3 开始，Zabbix 服务器和代理守护进程在单个请求中查询多个值的 SNMP 设备。这会影响各种 SNMP 监控项（常规 SNMP 项目，具有动态索引的 SNMP 项目和 SNMP 低级别发现），它使 SNMP 处理更加高效。请参阅下面的技术细节部分，了解内部工作原理。从 Zabbix 2.4 开始，它还为每个接口提供了一个 “使用批量请求” 的设置，允许为无法正确处理它们的设备禁用批量请求。

从 Zabbix 2.2.7 和 Zabbix 2.4.2 开始，Zabbix 服务器和代理守护程序的日志在收到不正确的 SNMP 响应时会打印类似以下内容：`SNMP response from host "gateway" does not contain all of the requested variable bindings` 虽然它们没有涵盖所有有问题的情况，但它们对于识别应禁用批量请求的各个 SNMP 设备非常有用。

从 Zabbix 2.2 开始 Zabbix 服务器和代理守护程序在执行 SNMP 检查时使用对应的超时配置参数。另外，在单个不成功的 SNMP 请求（超时/错误凭据）之后，守护程序不执行重试。之前，实际使用了 SNMP 库默认超时和重试值（分别为 1 秒和 5 次重试）。

从 Zabbix 2.2.8 和 Zabbix 2.4.2 开始，Zabbix 服务器和代理守护程序将始终至少重试一次：通过 SNMP 库的重试机制或通过内部批量处理机制。

<note warning> 如果监控 SNMPv3 设备，请确保 msgAuthoritativeEngineID（也称为 snmpEngineID 或 “引擎 ID”）从不被两台设备共享。根据RFC 2571 （3.1.1.1 节），每个设备必须是唯一的。:::

配置 SNMP 监控

要通过 SNMP 开始监控设备，必须执行以下步骤：

步骤 1

使用 SNMP 接口为设备创建一个主机。

输入 IP 地址。你可以使用自动添加一套监控项提供的 SNMP 模板之一（*SNMP* 设备模板等）。但是，模板可能与主机不兼容。单击 *Add* 以保存主机。

> **Note:**
> SNMP 检查不使用 代理端口, 请忽略它。

步骤 2

找出要监控项目的 SNMP 字符串（或 OID）。

要获取 SNMP 字符串列表，请使用 **snmpwalk** 命令（net-snmp的部分软件应该在 Zabbix 安装时同时安装）或等效工具：

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

这里的’2c’ 代表 SNMP 版本，你也可以将其替换为’1’，以在设备上指定 SNMP 版本为 v1。

它会返回给你一个 SNMP 字符串及其最后一个值的列表。如果不是，那么 SNMP ’community’ 可能与标准的’public’ 不同，在这种情况下，请找出它是什么。

然后，你可以浏览列表，直到找到要监控的字符串，例如：如果要监视通过端口 3 进入交换机的字节，你将使用此行中的 `IF-MIB ::ifInOctets.3` 字符串：

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

你现在可以使用 **snmpget** 命令找出’IF-MIB :: ifInOctets.3’ 的数字 OID：

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

请注意，字符串中的最后一个数字是你要监控的端口号。请参考: 动态索引.

如下所示：

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

重复一遍，OID 中的最后一个号码是端口号。

> **Note:**
> 3COM 似乎是使用数百个端口号，例如端口 1= 端口 101，端口 3= 端口 103，但思科使用常规数字，例如。端口 3=3。

> **Note:**
> 一些最常用的 SNMP OID，Zabbix 将自动转换为数字表示。

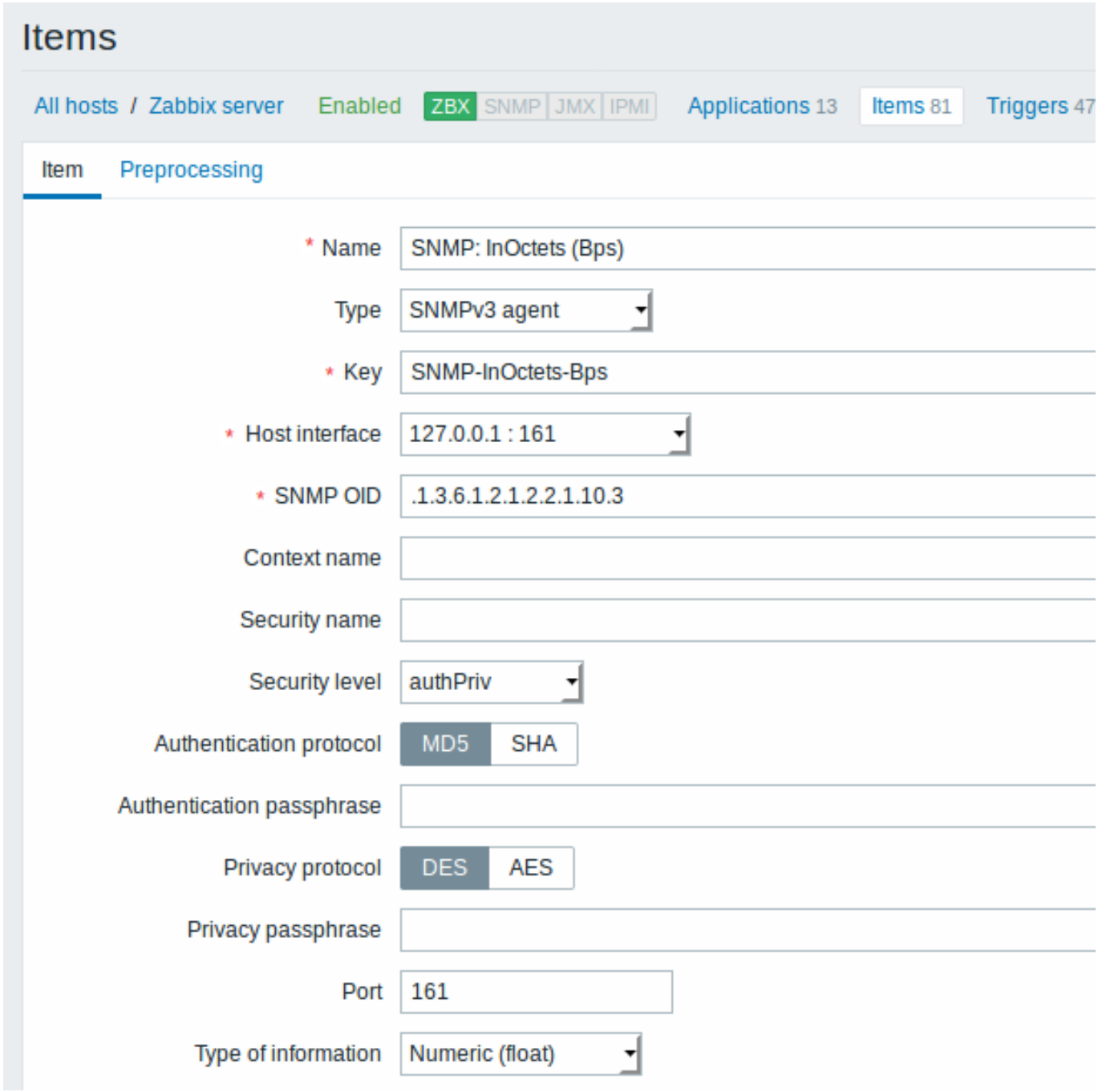
在上面的例子中，值类型是 “Counter32” 它在内部对应于 ASN_COUNTER 类型。完整的支持类型包括 ASN_COUNTER, ASN_COUNTER64, ASN_UINTEGER, ASN_UNSIGNED64, ASN_INTEGER, ASN_INTEGER64, ASN_FLOAT, ASN_DOUBLE, ASN_TIMETICKS, ASN_GAUGE, ASN_IPADDRESS, ASN_OCTET_STR 和 ASN_OBJECT_ID (从 2.2.8, 2.4.3 之后). 这些类型大致对应于 **snmpget** 输出的”Counter32”, ”Counter64”, ”UInteger32”, ”INTEGER”, ”Float”, ”Double”, ”Timeticks”, ”Gauge32”, ”IpAddress”, ”OCTET STRING”, ”OBJECT IDENTIFIER”, 但也有可能显示为”STRING”, ”Hex-STRING”, ”OID” 或者其它, 这取这取决于显示提示的表达方式。

步骤 3

创建一个监控项。

所以现在回到 Zabbix 并点击前面创建的 SNMP 主机的 监控项。如果你在创建主机时选择使用模板，你将拥有与主机相关联的 SNMP 监控项列表。我们假设你要使用 snmpwalk 和 snmpget 采集的信息创建监控项，单击 创建监控项。在新的监控项表单中，输入监控项 “名称”。确保 “主机接口” 字段中有你的交换机/路由器，并将” 类型” 字段更改为”SNMPv* 客户端”。输入 community （通常是 public），并将你之前检索到的文本或数字 OID 输入到’SNMP OID’ 字段中，例如：.1.3.6.1.2.1.2.2.1.10.3

输入 SNMP“端口” 为 161，“键值” 为有意义的内容，例如，SNMP-InOctets-Bps。将 “信息类型” 设置为 浮点数，并在进程预定步骤中添加 每秒更改的策略（重要！否则你将从 SNMP 设备获取累积值，而不是最新的变化）。如果你希望 “更新间隔” 和 “历史数据保留时长” 与默认值不同，请选择一个自定义乘数（如果需要），并输入。



所有必填输入字段都标有红色星号。

现在保存监控项，进入 监测中 → 最新数据来获取你的 SNMP 数据!

请注意 SNMPv3 监控的具体选项：

参数	描
上下文名称输入上下	名称以标识SNMP子网上的监控项。从Zab-bix 2.2开始SN-MPv3监控支持上下文名称。用户宏在此字段中解析。
安全名称输入安	名称用户宏在此字段中解析。
安全级别选择安	级别：**noAuthNoPriv** - 不使用身份验证或隐私协议 **AuthNoPriv** - 认证协议被使用，但不使用隐私协议 **AuthPriv** - 使用身份验证和隐私协议

参数描	
验证协议选择验	协议 - *MD5* 或者 *SHA*.
验证口令输入验	口令。用户宏在此字段中解析。
隐私协议选择隐	协议 - *DES* 或者 *AES*.
私钥输	私钥。用户宏在此字段中解析。

如果 SNMPv3 凭据（安全名称，验证协议/口令，隐私协议）错误，Zabbix 会从 net-snmp 收到错误，如果 私钥错误，在这种情况下，Zabbix 会从 net-snmp 收到 TIMEOUT 错误。

Warning:
验证协议, 验证口令, 隐私协议或 私钥修改后，需要重启服务器或代理来生效。

示例 1

一般范例：

参数描	
Community	public
OID	1.2.3.45.6.7.8.0 (or .1.2.3.45.6.7.8.0)
键值 &	t; 用作触发器引用的唯一字符串 > 例如, ”my_param”.

请注意，OID 可以以数字或字符串形式给出。但是，在某些情况下，字符串 OID 必须转换为数字表示。snmpget 可用于此目的：

在配置 Zabbix 源时指定了 --with-net-snmp 标志，可以监视 SNMP 参数。

示例 2

监控正常运行时间：

Parameter	Description
Community	public
Oid	MIB::sysUpTime.0
Key	router.uptime
Value type	Float
Units	uptime
Multiplier	0.01

参数描	
Community	public

参数描	
OID	MIB::sysUpTime.0
键值 r	uter.uptime
信息类型浮点数	
单位 u	time
乘数 0	01

批处理的内部工作

从 2.2.3 开始 Zabbix 服务器和代理查询 SNMP 设备在单个请求中的多个值。这会影响多种类型的 SNMP 监控项：

- 常规 SNMP 监控项;
- 具有动态索引的 SNMP 监控项;
- SNMP 低级发现规则.

具有相同参数的单个接口上的所有 SNMP 监控项都将同时进行查询。前两种类型的监控项由轮询器分批采集，最多 128 个监控项，而低级发现规则如前所述单独处理。

在较低级别上，执行查询值的操作有两种：获取多个指定对象和游历 OID 树。

对于”getting”，GetRequest-PDU 最多使用 128 个变量绑定。对于”walking”，GetNextRequest-PDU 用于 SNMPv1 和 GetBulkRequest，”max-repetitions” 字段最多 128 个用于 SNMPv2 和 SNMPv3。

因此，每个 SNMP 监控项类型的批量处理的优势如下：

* 常规SNMP项目受益于“getting” 的改进;
* 具有动态索引的SNMP监控项受益于“getting”和“walking”改进：“getting”用于索引验证，“walking”用于构建缓存;
* SNMP低级发现规则受益于“walking”的改进。

然而，有一个技术问题，并非所有设备都能够根据请求返回 128 个值。有些总是给出正确的回应，其它情况则会以 “tooBig（1）” 错误做出回应，或者一旦潜在的回应超过了一定的限度，则一律不回应。

为了找到最佳数量的对象来查询给定的设备，Zabbix 使用以下策略。它在请求中查询 “值 1” 时谨慎开始。如果成功，它会在请求中查询 “值 2”。如果再次成功，则查询请求中的 “值 3”，并通过将查询对象的数量乘以 1.5 来继续，导致以下请求大小的顺序：1,2,3,4,6,9,13,19，28，42，63，94，128。

然而，一旦设备拒绝给出适当的响应（例如，对于 42 个变量），Zabbix 会做两件事情。

首先，对于当前批量监控项，它将单个请求中的对象数减半，并查询 21 个变量。如果设备处于活动状态，那么查询应该在绝大多数情况下都有效，因为已知 28 个变量可以工作，21 个变量明显少于此。但是，如果仍然失败，那么 Zabbix 会逐渐回到查询值。如果此时仍然失败，那么设备肯定没有响应，请求大小也不是问题。

Zabbix 为后续批量监控项做的第二件事是它从最后成功的变量数量开始（在我们的示例中为 28），并继续将请求大小递增 1，直到达到限制。例如，假设最大响应大小为 32 个变量，后续请求的大小为 29,30,31,32 和 33. 最后一个请求将失败，Zabbix 将永远不再发出大小为 33 的请求。从那时起，Zabbix 将为该设备查询最多 32 个变量。

如果大型查询因此数量的变量而失败，则可能意味着两件事之一。设备用于限制响应大小的确切标准无法知晓，但我们尝试使用变量数来近似。因此，第一种可能性是，在一般情况下，此数量的变量大约是设备的实际响应大小限制：有时响应小于限制，有时它大于限制。第二种可能性是任何方向的 UDP 数据包都丢失了。由于这些原因，如果 Zabbix 查询失败，它会减少最大数量的变量以尝试深入到设备的舒适范围，但（从 2.2.8 开始）最多只能达到两次。

在上面的示例中，如果包含 32 个变量的查询失败，Zabbix 会将计数减少到 31. 如果发生这种情况也会失败，Zabbix 也会将计数减少到 30. 但是，Zabbix 不会将计数减少到 30 以下，因为它会假设进一步的失败是由于 UDP 数据包丢失，而不是设备的限制。

但是，如果设备由于其他原因无法正确处理批量请求，并且上述启发式方法不起作用，Zabbix 2.4 之后每个接口都有 “使用批量请求” 设置，允许禁用该设备的批量请求。

1 动态索引

概述

虽然你可能会在 SNMP OID 中找到所需的索引号（例如网络接口），但有时你不能完全依赖不变的索引号。

索引号可能是动态的 - 它们可能会随时间而改变，因此你的监控项可能会停止工作。

为了避免这种情况，可以定义一个考虑到索引号改变的可能性的 OID。

例如，如果需要检索索引值以匹配 Cisco 设备上的 **GigabitEthernet0/1** 接口的 **ifInOctets** ，请使用以下 OID：

```
ifInOctets["index","ifDescr","GigabitEthernet0/1"]
```

语法

使用 OID 的特殊语法：

<OID of data>[”index”,”<base OID of index>”,”<string to search for>”]

参数描	
OID of data	主 OID 用于监控项上的数据检索。
index	处理方法。目前支持一种方法： **index** – 搜索索引，并将其附加到数据 OID
base OID of index	该 OID 将被搜索以获取与该字符串对应的索引值。
string to search for	用于在进行查找时与值精确匹配的字符串。区分大小写。

示例

获取 *apache* 进程的内存使用率。

如果使用这种 OID 语法:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem["index","HOST-RESOURCES-MIB::hrSWRunPath", "/usr/sbin/apache2"]
```

索引号将在这里查找:

```
...
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"
...
```

现在我们有索引 5388. 索引将附加到此数据 OID，以便接收我们感兴趣的值：

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.5388 = INTEGER: 31468 KBytes
```

索引查找缓存

当请求动态索引项时，Zabbix 检索并缓存 base OID 下的整个 SNMP 表用于索引（即使早发现了匹配）。这是为了在另一个监控项稍后引用相同的 base OID - Zabbix 将在缓存中查找索引，而不是再次查询被监视的主机。请注意，每个轮询器进程使用单独的缓存。

在所有随后的值检索操作中，仅验证找到的索引。如果没有改变将请求结果值；如果已更改，则会重建高速缓存 - 遇到已更改索引的每个轮询器再次建立 SNMP 索引表。

2 特定 OID

一些最常用的 SNMP OID 自动转换为 Zabbix 的数字表示。例如，**ifIndex** 被翻译为 **1.3.6.1.2.1.2.2.1.1**，则将 **ifIndex.0** 转换为 **1.3.6.1.2.1.2.2.1.1.0**。

该表罗列了特定的 OID。

特定 OID 标	符描述	
ifIndex	1.3.6.1.2.1.2.2.1.1	每个接口的唯一值。

特定 OID 标	符描述	
ifDescr	1.3.6.1.2.1.2.2.1.2	包含有关接口信息的文本字符串。该字符串应包括制造商的名称、产品名称和硬件接口的版本。

特定 OID 标	符描述	
ifType	1.3.6.1.2.1.2.2.1.3	接口的类型，根据物理/链路协议，在协议栈的网络层“下面”进行快速区分。
ifMtu	1.3.6.1.2.1.2.2.1.4	可以在接口上发送/接收的最大数据报的大小，以八位字节指定。

特定 OID 标	符描述	
ifSpeed	1.3.6.1.2.1.2.2.1.5	接口当前带宽的估计，以位/秒为单位。
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	协议层的接口地址在协议栈的“网络层”之下。
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	接口的当前管理状态。
ifOperStatus	1.3.6.1.2.1.2.2.1.8	接口的当前操作状态。

特定 OID 标	符描述	
ifInOctets	1.3.6.1.2.1.2.2.1.10	接口上接收的八位字节总数，包括成帧字符。
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	传送到较高层协议的子网单播报文数量。

特定 OID 标	符描述	
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12	传送到较高层协议的非单播(即子网广播或子网多播)数据包的数量。

特定 OID 标	符描述	
ifInDiscards	1.3.6.1.2.1.2.2.1.13	即使没有检测到错误，也被选择丢弃的出栈数据包的数量，以防止它们被传输。丢弃这样的数据包的一个可能的原因是释放缓冲区空间。

特定 OID 标	符描述	
ifInErrors	1.3.6.1.2.1.2.2.1.14	包含错误的入栈数据包数量，阻止它们传递到较高层协议。
ifInUnknownProtos	1.3.6.1.2.1.2.2.1.15	通过接口接收到的数据包数量由于未知或不受支持的协议而被丢弃。

特定 OID 标	符描述	
ifOutOctets	1.3.6.1.2.1.2.2.1.16	从接口传出的八位字节总数，包括帧字符。

特定 OID 标	符描述	
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	要求发送更高级别协议的数据包的总数，并且没有寻址到此子层的多播或广播地址，包括丢弃或未发送的数据包。

特定 OID 标	符描述	
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18	要求发送更高级别协议的数据包的总数，并且被发送到该子层的多播或广播地址，包括丢弃或未发送的数据包。

特定 OID 标	符描述	
ifOutDiscards	1.3.6.1.2.1.2.2.1.19	即使没有检测到错误，也被选择丢弃的出栈数据包的数量，以防止它们被传输。丢弃这样的数据包的一个可能的原因是释放缓冲区空间。

特定 OID 标	符描述	
ifOutErrors	1.3.6.1.2.1.2.2.1.20	由于错误而无法传输的出栈数据包的数量。
ifOutQLen	1.3.6.1.2.1.2.2.1.21	输出包队列的长度(以包为单位)。

3 SNMP trap

概述

接收 SNMP trap 与查询启用 SNMP 的设备相反。

在这种情况下，信息发送自启用 SNMP 的设备并由 Zabbix 收集或"trapped"。

通常在某些条件更改时发送 trap，并且代理通过端口 162 连接到服务器（相反的，代理端的 161 端口是用于查询代理的）。使用 trap 可以检测在查询间隔期间发生的一些可能被查询数据遗漏的短期问题。

在 Zabbix 中接收 SNMP trap 旨在使用 **snmptrapd** 和内置机制之一来传递 trap 到 Zabbix - 一个 perl 脚本或 SNMPTT。

接收 trap 的工作流程：

1. **snmptrapd** 收到 trap
2. snmptrapd 将 trap 传递给 SNMPTT 或调用 Perl trap 接收器
3. SNMPTT 或 Perl trap 接收器解析，格式化并将 trap 写入文件
4. Zabbix SNMP trap 读取并解析 trap 文件
5. 对于每个 trap，Zabbix 发现主机接口与接收的 trap 地址匹配的所有 "SNMP trap" 监控项。请注意，在匹配期间只使用主机接口中选定的 "IP" 或 "DNS"。
6. 对于每个找到的监控项，将 trap 与 "snmptrap[regexp]" 中的 regexp 进行比较。trap 设置为 **all** 匹配项的值。如果没有找到匹配的监控项，并且有一个 "snmptrap.fallback" 监控项，则将 trap 设置为该监控项的值。
7. 如果 trap 未设置为任何监控项的值，Zabbix 默认记录未匹配的 trap。（通过管理 → 常规 → 其它中的 "记录未匹配的 SNMP trap（Log unmatched SNMP traps）" 进行配置。）

1 配置 SNMP trap

在前端页面中配置此监控项类型的以下字段：

* 你的主机必须具有SNMP接口

在 配置 → 主机中，在主机接口字段中设置具有正确 IP 或 DNS 地址的 SNMP 接口。将每个收到的 trap 的地址与所有 SNMP 接口的 IP 和 DNS 地址进行比较，以查找相应的主机。

* 配置监控项

在 **Key** 字段中使用一个 SNMP trap Key：

Key		
描述返	值注释	
snmptrap[regexp] 捕获与 regexp 中指定的正则表达式匹配的所有 SNMP trap。如果 regexp 未指定，则捕获任何 trap。SNMP trap 该监控项只能用于 SNMP 接口		< 此监控项从 Zabbix **2.0.0.** 开始支持 注意: 从 Zabbix 2.0.5 开始，该监控项的参数支持用户宏和全局正则表达式。
snmptrap.fallback 捕获未被该接口的任何 snmptrap[] 监控项捕获的所有 SNMP trap。SNMP trap 该监控项只能	于 SNMP 接口。	< 该监控项从 Zabbix **2.0.0.** 以后支持

Note:
目前不支持多行正则表达式匹配。

将要解析的时间戳的信息类型设置为'Log'。请注意，其它格式（如 "数字"）也是可以接受的，但可能需要自定义 trap 处理程序。

Note:
要使 SNMP trap 监控工作，必须首先正确设置。

2 设置 SNMP trap 监控

配置 Zabbix 服务器/代理服务器

要读取 trap，必须将 Zabbix 服务器或代理服务器配置为启动 SNMP trap 进程，并指向由 SNMPTT 或 perl trap 接收器写入的 trap 文件。为此，请编辑配置文件 (zabbix_server.conf 或者zabbix_proxy.conf)：

1. StartSNMPTrapper=1
2. SNMPTrapperFile=[TRAP FILE]

<note warning> 如果使用 systemd 参数**PrivateTmp**，则该文件不太可能在*/tmp* 下使用。:::

配置 SNMPTT

首先，snmptrapd 应该配置为使用 SNMPTT。

<note tip> 为了获得最佳性能，应将 SNMPTT 配置为使用 **snmptthandler-embedded** 的守护进程，并将 trap 传递给它。有关 SNMPTT 的配置，请查看其主页上的说明：
http://snmptt.sourceforge.net/docs/snmptt.shtml :::

当 SNMPTT 配置为接收 trap 时，配置 SNMPTT 记录 trap：

1. 将 trap 记录到 Zabbix 将读取的 trap 文件中：
   log_enable = 1
   log_file = [TRAP FILE]
2. 设置日期时间格式：
   date_time_format = %H:%M:%S %Y/%m/%d = [DATE TIME FORMAT]

现在格式化 Zabbix 的 trap 来识别它们（编辑 snmptt.conf）：

1. 每个 FORMAT 语句应以 "ZBXTRAP [address]" 开头，其中 [address] 将与 Zabbix 上 SNMP 接口的 IP 地址和 DNS 地址进行比较。例如:
   EVENT coldStart .1.3.6.1.6.3.1.1.5.1 "Status Events" Normal
   FORMAT ZBXTRAP $aA Device reinitialized (coldStart)
2. 请参阅下面的 SNMP trap 格式说明，了解更多信息。

Attention:
不要使用未知的 trap - Zabbix 将无法识别它们。未知 trap 可以通过在 snmptt.conf 中定义一个常规事件来处理：
EVENT general .* "General event" Normal

配置 Perl trap 接收器

要求：Perl，Net-SNMP 使用--enable-embedded-perl 编译（默认情况下从 Net-SNMP 5.4 支持）

Perl trap 接收器（查找 misc/snmptrap/zabbix_trap_receiver.pl）可以直接从 snmptrapd 将 trap 传递给 Zabbix 服务器。配置过程:

- 将 perl 脚本添加到 snmptrapd 配置文件（snmptrapd.conf）中，例如：
  perl do "[FULL PATH TO PERL RECEIVER SCRIPT]";
- 配置接收器, 例如:
  $SNMPTrapperFile = '[TRAP FILE]';
  $DateTimeFormat = '[DATE TIME FORMAT]';

如果没有引用脚本名称，snmptrapd 将拒绝启动消息，类似:

SNMP trap 格式

所有定制的 perl trap 接收器和 SNMPTT trap 配置必须按以下方式格式化 trap：**[timestamp] [the trap, part 1] ZBXTRAP [address] [the trap, part 2]**, 说明

- [timestamp] - 用于日志监控项的时间戳
- ZBXTRAP - 头表示新的 trap 从此行开始
- [address] - 用于查找此 trap 的主机的 IP 地址

注意，"ZBXTRAP" 和 "[address]" 将在处理过程中从消息中删除。如果 trap 格式化为其它方式，Zabbix 也许能意外的解析 trap。

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

3 系统要求

大文件支持

Zabbix 为 SNMP trap 文件提供了 "大文件支持"。Zabbix 可以读取的最大文件大小为 2^63（8 EiB）。请注意，文件系统可能会对文件大小施加下限。

日志轮换

Zabbix 不提供任何日志轮换系统（它应由用户处理）。日志轮换应该首先重命名旧文件，然后才能将其删除，以免丢失 trap：

1. Zabbix 在最后一个已知位置打开 trap 文件，并转到步骤 3
2. Zabbix 通过比较 inode 号和定义 trap 文件的 inode 号，检查当前打开的文件是否已经旋转。如果没有打开的文件，Zabbix 将重置最后一个位置并转到步骤 1。
3. Zabbix 从当前打开的文件中读取数据并设置新的位置。
4. 新数据被解析。如果这是旋转的文件，文件将关闭并返回到步骤 2。
5. 如果没有新的数据，Zabbix sleep 1 秒钟，然后回到步骤 2。

文件系统

由于 Trap 文件的执行，Zabbix 需要文件系统支持 inode 来区分文件（该信息由 stat() 调用获取）。

4 设置示例

本示例使用 snmptrapd + SNMPTT 将陷阱传递给 Zabbix 服务器。设置：

1. **zabbix_server.conf** - 配置 Zabbix 启动 SNMP trap 并设置 trap 文件：
   StartSNMPTrapper=1
   SNMPTrapperFile=/tmp/my_zabbix_traps.tmp
2. **snmptrapd.conf** - 添加 SNMPTT 作为 trap 处理程序：
   traphandle default snmptt
3. **snmptt.ini** - 配置输出文件和时间格式：
   log_file = /tmp/my_zabbix_traps.tmp
   date_time_format = %H:%M:%S %Y/%m/%d
4. **snmptt.conf** - 定义默认 trap 格式：
   EVENT general .* "General event" Normal
   FORMAT ZBXTRAP $aA $ar
5. 创建一个 SNMP 监控项测试：
   Host's SNMP interface IP: 127.0.0.1

Key: snmptrap[”General”]
Log time format: hh:mm:ss yyyy/MM/dd

结果如下：

1. 用于发送 trap 的命令：
   snmptrap -v 1 -c public 127.0.0.1 ’.1.3.6.1.6.3.1.1.5.3’ ’0.0.0.0’ 6 33 ’55’ .1.3.6.1.6.3.1.1.5.3 s ”teststring000”
2. 接收到的 trap:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal ”General event” localhost - ZBXTRAP 127.0.0.1 127.0.0.1
3. 测试监控项的值:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal ”General event” localhost - 127.0.0.1

<note tip> 这个简单的例子使用 SNMPTT 作为 **traphandle**。为了在生产系统上获得更好的性能，请使用嵌入式 Perl 将 trap 从 snmptrapd 传递到 SNMPTT 或直接传递到 Zabbix。

5 请参阅

- 来自 zabbix.org 的基于 CentOS 的 SNMP trap 教程

4 IPMI 检查

概述

你可以在 Zabbix 中监控智能平台管理接口（IPMI）设备的运行状况和可用性。要执行 IPMI 检查，Zabbix 服务器必须首先配置IPMI 支持。

IPMI 是计算机系统的远程” 关闭” 或 “带外” 管理的标准接口。它可以独立于操作系统直接从所谓的 “带外” 管理卡监视硬件状态。

Zabbix IPMI 监控仅适用于支持 IPMI 的设备 (HP iLO, DELL DRAC, IBM RSA, Sun SSP, 等等).

从 Zabbix 3.4 开始，添加了一个新的 IPMI 管理器进程来安排 IPMI 轮询器进行 IPMI 检查。现在，主机始终只由一个 IPMI 轮询器轮询，从而减少了与 BMC 控制器的打开连接数。通过这些更改，可以安全地增加 IPMI 轮询器的数量，而无需担心 BMC 控制器过载。启动至少一个 IPMI 轮询器时，将自动启动 IPMI 管理器进程。

也可以参考 IPMI 检查的已知问题。

配置

主机配置

主机必须配置为处理 IPMI 检查。必须添加 IPMI 接口，必须定义相应的 IP 和端口号，并且必须定义 IPMI 认证参数。

更多细节请查看主机定义。

服务器配置

默认情况下，Zabbix 服务器未配置为启动任何 IPMI 轮询，因此任何添加的 IPMI 监控项将无法正常工作。要更改此选项，请以 root 身份打开 Zabbix 服务器配置文件 (zabbix_server.conf)，并查找以下行：

```
# StartIPMIPollers=0
```

取消注释，并设置 poller 计数为 3，如下：

```
StartIPMIPollers=3
```

保存文件，然后重新启动 zabbix_server。

监控项配置

配置主机级别的监控项时：

- 对于 主机接口，选择 IPMI IP 和端口
- 选择’IPMI agent’ 作为 类型
- 指定 *IPMI* 传感器（例如在 Dell Poweredge 上的’FAN MOD 1A RPM’ ）。默认情况下，应指定传感器 ID。也可以在值之前使用前缀：

```
  * ''id:'' - 指定传感器ID；
  * ''name:'' - 指定传感器全名。 这在传感器只能通过指定全名来区分的情况下非常有用。
* 在主机中输入唯一的监控项[[zh:manual:config:items:item:key|key]]（例如，ipmi.fan.rpm）
* 在这种情况下，选择相应的信息类型 ('Numeric (float)'，对于离散传感器 - 'Numeric (unsigned)')，单位(最可能
```

超时和会话终止

IPMI 消息超时和重试计数在 OpenIPMI 库中定义。由于目前 OpenIPMI 的设计，无论在接口还是监控项级别都不能在 Zabbix 中使这些值进行配置。

LAN 的 IPMI 会话不活动超时时间为 60 +/- 3 秒。目前无法使用 OpenIPMI 定期发送激活会话命令。如果没有从 Zabbix 到特定 BMC 的 IPMI 项检查超过在 BMC 中配置的会话超时，则超时超时后的下一次 IPMI 检查将由于单个消息超时、重试或接收错误而超时。之后，打开

一个新的会话，并启动 BMC 的完全重新扫描。如果要避免 BMC 的不必要的 rescans，建议将 IPMI 监控项轮询间隔设置为低于 BMC 中配置的 IPMI 会话不活动超时。

关于 IPMI 离散传感器的注意事项

要在主机上找到传感器启动 Zabbix 服务器，启用 **DebugLevel=4**。等待几分钟，并在 Zabbix 服务器日志文件中查找传感器发现记录：

```
$ grep 'Added sensor' zabbix_server.log
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:7 id:'CATERR' reading_type:
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'CPU Therm Trip' read
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'System Event Log' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'PhysicalSecurity' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'IPMI Watchdog' readi
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'Power Unit Stat' rea
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Ctrl %' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Margin' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 2' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 3' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'P1 Mem Margin' readi
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'Front Panel Temp' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'Baseboard Temp' read
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +5.0V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +3.3V STBY' readi
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +3.3V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.5V P1 DDR3' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.1V P1 Vccp' re
8358:20130318:111122.174 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +1.05V PCH' readi
```

要解码 IPMI 传感器类型和状态，请在 http:*www.intel.com/content/www/us/en/servers/ipmi/ipmi-specifications.html (*在撰写本文时，最新的文件是 *http:*www.intel.com/content/dam/www/public/us/en/documents/product-briefs/second-gen-interface-spec-v2.pdf) 获取 IPMI 2.0 规范的副本

开始的第一个参数是 “reading_type”。从规范中使用 “表 42-1，事件/读取类型代码范围” 来解码 “reading_type” 代码。我们示例中的大多数传感器都有 “reading_type：0x1”，这意味着是 “threshold” 传感器。“表 42-3，传感器类型代码” 表示：“类型：0x1” 表示温度传感器；“类型：0x2” - 电压传感器；“类型：0x4” - 风扇等阈值传感器有时称为 “模拟” 传感器，因为它们测量连续参数，如温度，电压，每分钟转数。

另一个例子 - 一个带有 “read_type：0x3” 的传感器。“表 42-1，事件/读取类型代码范围” 表示读取类型代码 02h-0Ch 表示 “通用离散” 传感器。离散传感器具有多达 15 个可能的状态（换句话说-最多 15 个有意义的位）。例如，对于具有 “type：0x7” 的传感器 “CATERR”，“表 42-3，传感器类型代码” 表示此类型 “处理器”，各个位的含义是：00h（最低有效位）- IERR ；01h - 散热等。

在我们的示例中有几个传感器具有 “reading_type：0x6f”。对于这些传感器，“表 42-1，事件/读取类型代码范围” 建议使用 “表 42-3，传感器类型代码” 来解码位的含义。例如，传感器 “Power Unit Stat” 的类型为 “0x9”，表示 “Power Unit”。Offset 00h 表示 “PowerOff / Power Down”。换句话说，如果最低有效位为 1，则服务器断电。为了测试这个位，可以使用 **band** 与掩码 1 的功能。触发表达式可能就像

```
{www.zabbix.com:Power Unit Stat.band(#1,1)}=1
```

警告服务器关机。

关于 OpenIPMI-2.0.16,2.0.17,2.0.18 和 2.0.19 中离散传感器名称的注释

OpenIPMI-2.0.16,2.0.17 和 2.0.18 中的离散传感器的名称通常在附近附加一个额外的”0” （或其它数字或字母）。例如，当 `ipmitool` 和 OpenIPMI-2.0.19 将传感器名称显示为”`PhysicalSecurity`” 或”`CATERR`” 时，在 OpenIPMI-2.0.16,2.0.17 和 2.0.18 中，名称分别为”`PhysicalSecurity0`” 或”`CATERR0`”。

当使用 OpenIPMI-2.0.16,2.0.17 和 2.0.18 配置 IPMI 项目时，请在 IPMI 代理监控项的 IPMI 传感器字段中使用以 “0” 结尾的名称。当你的 Zabbix 服务器升级到使用 OpenIPMI-2.0.19（或更高版本）的新 Linux 发行版时，具有这些 IPMI 离散传感器的监控项将变为 “不支持”。你必须更改其 IPMI 传感器名称（最后删除 “0”），并等待一段时间才能再次转为”Enabled”。

关于阈值和离散传感器同时可用的注意事项

一些 IPMI 代理提供了相同名称的阈值传感器和离散传感器。在 2.2.8 和 2.4.3 之前的 Zabbix 版本中，选择了第一个提供的传感器。从 2.2.8 和 2.4.3 版本以后，偏向于阈值传感器。

连接终止注意事项

如果不执行 IPMI 检查（由于任何原因：所有主机 IPMI 监控项禁用/不支持、主机已禁用/已删除、主机维护等），IPMI 连接将从 Zabbix 服务器或代理服务器终止 3 到 4 小时，具体时间取决于 Zabbix 服务器/代理服务器何时启动。

5 简单检查

Overview

概览

简单检查通常用于检查远程未安装 Zabbix agent 的服务。

请注意，简单检查不需要 Zabbix agent，由 Zabbix server 和 Zabbix proxy 来负责处理（例如创建外部连接等）。

简单检查使用示例：

```
net.tcp.service[ftp,,155]
net.tcp.service[http]
net.tcp.service.perf[http,,8080]
net.udp.service.perf[ntp]
```

Note:
在简单检查项的配置中，用户名和 密码字段用于 Vmware 的监控项；非 VMware 监控项则可忽略。

支持的简单检查

Zabbix 支持的简单检查列表：

另请参考：

- VMware monitoring item keys
- VMware 监控项键值

键值				
	描述 *	返回值 ** ** 参数	* 注解	
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]	通过 ICMP ping，检测主机的可访问性 0 - ICMP pi	g 失败 **target** - 1 - ICMP ping 成功 *	机 IP 或者域名 示例: **packets** - 数据包数量 =&gtin-terval** - 连续数据包之间的时间间隔（以毫秒为单位） **size** - 数据包大小（以字节为单位） 另请参考: [默认值](**timeout** - 超时时间（以毫秒为单位）	<icmpping[,4] → 4 个包中只要一个有返回，那么该项则返回 1. imple_checks#icmp_pin 表
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]				

键值				
	丢失数据包的百分比数值（浮点数）	**target** - 主机	P 或者域名 另请参考: [默认值](simple_ch	cks#icmp_pings) 表.
			packets - 数据包数量 **interval** - 连续数据包之间的时间间隔（以毫秒为单位） **size** - 数据包大小（以字节为单位） **timeout** - 超时时间（以毫秒为单位）	
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]				
	ICMP ping 响应时间（以秒为单位）数值（浮点数）	**target** - 主机 IP	者域名 如果主机不可用 (达到超时), 则该监控项返回 0**packets** - 数据包数量 如果返回 **interval** - 连续数据包之间的时间间隔（以毫秒为单位） **size** - 数据包大小（以字节为单位） 另请参考: [默认值](**timeout** - 超时时间（以毫秒为单位） **mode** - 可能的值: *min*, *max*, *avg* (默认)	小于 0.0001 秒, 该值将被设置为 0.0001 秒. imple_checks#icmp_pin 表.
net.tcp.service[service,<ip>,<port>]				

键值					
		检测服务是否正在运行并且接受 TCP 连接. 0 - 服务停止	**service** - 可能的值: 1 - 服务正在运行 **por	*ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (另见详细说明) 示例: **ip** - IP 地址或者域名 (默认使用主机 IP/DNS) => net.t** - 端口号 (默认使用标准服务端口)	<p.service[ftp„45] → 可用于测试运行在 TCP 45 端口上 FTP 服务器的可用性. 请注意，使用 *tcp* 服务必须指定端口. 这些检查可能会在系统守护进程日志文件中产生额外的信息 (通常会记录 SMTP 和 SSH 会话). 目前不支持检测加密协议 (如端口 993 上的 IMAP 或端口 995 上的 POP). 作为一种解决方法, 请使用 net.tcp.service[tcp,<ip> 进行检测. 从 Zabbix 2.0 以后开始支持 *https* 和 *telnet* 服务。
net.tcp.service.perf[service,<ip>,<port>]					

键值				
	检测 TCP 服务性能. 浮点数.	**servic 0.000000** **-** 服务停止 p seconds - 连接到服务花费的时间 (秒)	可能的值: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (另见详细说明) 示例: **ip** - IP 地址或者域名 (默认使用主机 ip/DNS) => net.trt** - 端口号 (默认使用标准服务端口) 目前不支持检测加密协议 (	<p.service.perf[ssh] → 可以用来测试 SSH 服务器的初始响应速度. 请注意，使用 *tcp* 服务必须指定端口。 端口 993 上的 IMAP 或端口 995 上的 POP)。作为一种解决方法, 请使用 net.tcp.service.perf[tcp, 进行检测. 从 Zabbix 2.0 以后开始支持 *https* and *telnet* 服务。 在 Zabbix 2.0 之前，调用的是 tcp_perf。
net.udp.service[service,<ip>,<port>]				
	检测服务是否正在运行并响应 UDP 请求 0 - 服务停止	**service** - 可能的值 1 - 服务正在运行 **por	*ntp* (另见详细说明) 示例: **ip** - IP 地址或者域名 (默认使用主机 ip/DNS) => net.u** - 端口号 (默认使用标准服务端口).	<p.service[ntp,,45] → 可用于测试 UDP 端口 45 上 NTP 服务的可用性. 从 Zabbix 3.0 以后开始支持此监控项, 但在之前的版本中 *ntp* 服务可用于 net.tcp.service[] 监控项。
net.udp.service.perf[service,<ip>,<port>]				
	检测 UDP 服务的性能浮点数.	**service 0.000000** **-** 服务停止 p seconds - 等待服务响应的时间 (秒)	可能的值: *ntp* (另见详细说明) 示例: **ip** - IP 地址或者域名 (默认使用主机 IP/DNS) => net.urt** - 端口号 (默认使用标准服务端口).	<p.service.perf[ntp] → 可用于测试 NTP 服务的响应时间. 从 Zabbix 3.0 以后开始支持此监控项, 但在之前的版本中 *ntp* 服务可用于 net.tcp.service[] 监控项。

超时处理

如果简单检查时间超过了 zabbix server 或是 proxy 配置文件中设置的超时时间,zabbix 将不会做处理。

ICMP pings

Zabbix 使用外部程序 **fping** 来处理 ICMP ping

fping 不包含在 Zabbix 的发行版中，您需要另外安装。如果程序未安装、程序权限错误或者程序路径与配置文件中定义的不匹配（'FpingLocation' 参数），则不会处理 ICMP ping (**icmpping**, **icmppingloss**, **icmppingsec**)

另请参考：已知问题

fping fping 必须可被 Zabbix 守护进程以 root 身份执行, 需要设置 setuid 权限。为设置正确的权限，请以 root 身份执行这些命令：

```
shell> chown root:zabbix /usr/sbin/fping
shell> chmod 4710 /usr/sbin/fping
```

执行上述两条命令之后，检查 fping 可执行文件的所有权。在某些情况下，可以通过执行 chmod 命令来重置所有权。

还要检查一下，如果用户 zabbix 属于 zabbix 组，则运行：

```
shell> groups zabbix
```

如果没有添加上，通过如下命令解决：

```
shell> usermod -a -G zabbix zabbix
```

ICMP 检测参数的默认值、限制和以及数值的描述：

参数单	描述	Fping	标志 fping 默认设	Zab	ix 允许的	< 限制

Warning:
警告: 根据平台和版本的不同，fping 的默认值也会有所不同 - 如有疑问, 请参考 fping 文档。

Zabbix 将三个 *icmpping** 键值中任何一个 IP 地址写入一个临时文件中，然后传递给 **fping**。如果监控项有不同的键值参数，则只有具有相同键值参数的项才被写入单个文件。
所有写入到单个文件的 IP 地址将通过 fping 并行检查，因此 Zabbix icmp pinger 进程将花费固定的时间来忽略文件中的 IP 地址数量

1 VMware 监控项

监控项键值

该表提供了用于监控VMware 环境的简单检查的详细说明。

键值			
	描述返	值参数	注解
vmware.cluster.discovery[<url>]			
	发现 VMware 集群. JSO	对象 **url	* - VMware 服务的 URL 地址
vmware.cluster.status[<url>, <name>]			
	VMware 集群状态. 整型:	**url0 - 灰色; *1 - 绿色; 2 - 黄色; 3 - 红色	* - VMware 服务的 URL 地址 name** - VMware 集群名称
vmware.eventlog[<url>]			
	VMware 事件日志. Log	**u	l** - VMware 服务的 URL 地址
vmware.fullname[<url>]			

键值			
	VMware 服务全名. 字符串	**url*	- VMware 服务的 URL 地址
vmware.hv.cluster.name[<url>,<uuid>]			
	VMware 管理层集群名. 字符串	**url**	VMware 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.cpu.usage[<url>,<uuid>]			
	VMware 虚拟机管理程序处理器使用情况 (Hz). 整型	**url** - VMwar	服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.datacenter.name[<url>,<uuid>]			
	VMware 虚拟管理器数据中心名称. 字符串	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.datastore.discovery[<url>,<uuid>]			
	发现 VMware 虚拟管理程序数据存储. JSON 对象	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.datastore.read[<url>,<uuid>,<datastore>,<mode>]			
	从数据存储读取操作的平均时间 (毫秒). 整型 ^**[2](vmwar	_keys#footnotes)^ url** - VMware 服	的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名 **datastore** - 数据存储名称 **mode** - 延迟 (默认)
vmware.hv.datastore.size[<url>,<uuid>,<datastore>,<mode>]			

键值			
	VMware 数据存储空间（字节为单位）或占总数的百分比. 整型 - 字节数	**url** - VMware 服务的 URL 地址 浮点数 - 百分比 **uui	从 Zabbix 3.0.6, 3.2.2 以后可用 ** - VMware 虚拟机管理程序主机名 **datastore** - 数据存储名称 **mode** - 可能的值: total (默认), free, pfree (剩余百分比), uncom-mitted
vmware.hv.datastore.write[<url>,<uuid>,<datastore>,<mode>]			
	对数据存储区进行写操作的平均时间 (毫秒). 整型 ^**[2](vmware_	eys#footnote3)^ url** - VMware 服务的	RL 地址 **uuid** - VMware 虚拟机管理程序主机名 **datastore** - 数据存储名称 **mode** - 延迟 (默认)
vmware.hv.discovery[<url>]			
	发现 VMware 虚拟机管理程序. JSON 对象	**url** -	Mware 服务的 URL 地址
vmware.hv.fullname[<url>,<uuid>]			
	VMware 虚拟机管理程序名称. 字符串	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.cpu.freq[<url>,<uuid>]			
	VMware 虚拟机管理程序处理器频率 (Hz). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.cpu.model[<url>,<uuid>]			
	VMware 虚拟机管理程序处理器模式. 字符串	**url** - VMwa	e 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名

键值			
vmware.hv.hw.cpu.num[<url>,<uuid>]	VMware 虚拟机管理程序上处理器的内核数. 整型	**url** - VMware	务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.cpu.threads[<url>,<uuid>]	VMware 虚拟机管理程序上处理器的线程数. 整型	**url** - VMware	务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.memory[<url>,<uuid>]	VMware 虚拟机管理程序总内存 (字节). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.model[<url>,<uuid>]	VMware 虚拟机管理程序模式. 字符串	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.uuid[<url>,<uuid>]	VMware 虚拟机管理程序 BIOS UUID. 字符串	**url** -	VMware 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.hw.vendor[<url>,<uuid>]	VMware 虚拟机管理程序供应商名称. 字符串	**url** - VMwa	e 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.memory.size.ballooned[<url>,<uuid>]	VMware 虚拟机管理程序膨胀内存大小 (字节). 整型	**url** - VMware	务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名
vmware.hv.memory.used[<url>,<uuid>]			

键值				
	VMware 虚拟机管理程序内存使用大小 (字节). 整型	**url** - VMware	务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名	
vmware.hv.network.in[<url>,<uuid>,<mode>]				
	VMware 虚拟机管理程序网络输入数据统计 (每秒字节数). 整型	s#footnotes) 地址 url** - VMware 服务的 UR ^**[2](vmware_ke	**uuid** - VMware 虚拟机管理程序主机名 **mode** - bps (默认)	
vmware.hv.network.out[<url>,<uuid>,<mode>]				
	VMware 虚拟机管理程序网络输出数据统计 (每秒字节数). 整型	s#footnotes) 地址 url** - VMware 服务的 UR ^**[2](vmware_ke	**uuid** - VMware 虚拟机管理程序主机名 **mode** - bps (默认)	
vmware.hv.perfcounter[<url>,<uuid>,<path>,<instance>]				
	VMware 虚拟机管理程序性能计数器值. 整型 ^**[2](vm	are_keys#footnotes) 服务的 url** - VMwa	URL 地址 从 Zabbix 2.2.9, 2.4.**uuid** - VMware 虚拟机管理程序主机名 **path** - 性能计数器路径 [1] **instance** - 性能计数器实例. 对聚合值使用空实例 (默认)	以后开始支持
vmware.hv.sensor.health.state[<url>,<uuid>]				
	VMware 虚拟机管理程序健康状态汇总传感器. 整型:	**url** - VMware 服 0 - 灰色; *1 - 绿色; 2 - 黄色; 3 - 红色	的 URL 地址 从 Zabbix 2.2.16, 3.0.6,uuid** - VMware 虚拟机管理程序主机名	3.2.2 以后开始支持
vmware.hv.status[<url>,<uuid>]				

键值				
	VMware 虚拟机管理程序状态. 整型:	**url** -0 - 灰色; *1 - 绿色; 2 - 黄色; 3 - 红色	Mware 服务的 URL 地址 从 Zabbix 2.2.16,uuid** - VMware 虚拟机管理程序主机名	3.0.6, 3.2.2 开始支持使用主机系统整体状态属性
vmware.hv.uptime[<url>,<uuid>]				
	VMware 虚拟机管理程序运行时间 (秒). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名	
vmware.hv.version[<url>,<uuid>]				
	VMware 虚拟机管理程序版本. 字符串	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名	
vmware.hv.vm.num[<url>,<uuid>]				
	VMware 虚拟机管理程序上的虚拟主机数量. 整型	**url** - VMware	务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名	
vmware.version[<url>]				
	VMware 服务版本. 字符串	**url*	- VMware 服务的 URL 地址	
vmware.vm.cluster.name[<url>,<uuid>]				
	VMware 虚拟机名称. 字符串	**url**	- VMware 服务的 URL 地址 **uuid** - VMware 虚拟机主机名	
vmware.vm.cpu.num[<url>,<uuid>]				
	虚拟机上处理器的数量. 整型	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机主机名	
vmware.vm.cpu.ready[<url>,<uuid>]				

键值			
	虚拟机准备就绪，但不能在物理 CPU 上运行的时间 (以毫秒为单位). CPU 准备时间取决于主机上的虚拟机数量及其 CPU 负载 (%). 整型 [2]	**url** -VMware 服务的 URL 地址	从 Zabbix version 3.0.0 开始支持 **uuid** - VMware 虚拟机主机名
vmware.vm.cpu.usage[<url>,<uuid>]			
	VMware 虚拟机 cpu 的使用率 (Hz). 整型	**url**	VMware 服务的 URL 地址 **uuid** -VMware 虚拟机主机名
vmware.vm.datacenter.name[<url>,<uuid>]			
	VMware 虚拟机数据中心名称. 字符串	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.discovery[<url>]			
	自动发现 VMware 虚拟机. JSON 对象	**url**	VMware 服务的 URL 地址
vmware.vm.hv.name[<url>,<uuid>]			
	VMware 虚拟机管理程序名称. 字符串	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size[<url>,<uuid>]			
	VMware 虚拟机总内存大小 (字节). 整型	**url** - V	ware 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.ballooned[<url>,<uuid>]			
	VMware 虚拟机膨胀内存大小 (字节). 整型	**url** - VM	are 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.compressed[<url>,<uuid>]			

键值			
	VMware 虚拟机压缩内存大小 (字节). 整型	**url** - VM	are 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.private[<url>,<uuid>]			
	VMware 虚拟主机专用内存大小 (字节). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.shared[<url>,<uuid>]			
	VMware 虚拟机共享内存大小 (字节). 整型	**url** - VM	are 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.swapped[<url>,<uuid>]			
	VMware 虚拟机交换内存大小 (字节). 整型	**url** - VM	are 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.usage.guest[<url>,<uuid>]			
	VMware 虚拟机客户机内存使用量 (字节). 整型	**url** - VMwa	e 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.memory.size.usage.host[<url>,<uuid>]			
	VMware 虚拟机主机内存使用量 (字节). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机主机名
vmware.vm.net.if.discovery[<url>,<uuid>]			
	自动发现 VMware 虚拟机网络接口. JSON 对象	**url** - VM	are 服务的 URL 地址 **uuid** -VMware 虚拟机主机名
vmware.vm.net.if.in[<url>,<uuid>,<instance>,<mode>]			
	VMware 虚拟机网卡输入数据统计 (每秒字节/数据包). 整型 ^**[2](vmware_	eys#footnote2)1^ url** - VMware 服务的	地址 **uuid** - VMware 虚拟机主机名 **instance** - 网卡实例 **mode** - bps (默认)/pps - 每秒字节/数据包

键值				
vmware.vm.net.if.out[<url>,<uuid>,<instance>,<mode>]	VMware 虚拟机网卡输出数据统计 (每秒字节/数据包). 整型 ^**[2](vmware_	eys#footnotes)^ url** - VMware 服务的	RL 地址 **uuid** - VMware 虚拟机主机名 **instance** - 网卡实例 **mode** - bps (默认)/pps - 每秒字节/数据包	
vmware.vm.perfcounter[<url>,<uuid>,<path>,<instance>]	VMware 虚拟机性能计数器值. 整型 ^**[2	(vmware_keys#footnotes)^ url** -	VMware 服务的 URL 地址 Available since**uuid** -VMware 虚拟机主机名 **path** - 性能计数器路径 [1] **instance** - 性能计数器实例. 对聚合值使用空实例 (默认)	从 Zabbix 2.2.9, 2.4.4 开始支持
vmware.vm.powerstate[<url>,<uuid>]	VMware 虚拟机电源状态. 整型:	**url**0 - 关闭; *1 - 开机; 2 - 暂停	VMware 服务的 URL 地址 uuid** - VMware 虚拟机主机名	
vmware.vm.storage.committed[<url>,<uuid>]	VMware 虚拟机已提交存储空间 (字节). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** -VMware 虚拟机主机名	
vmware.vm.storage.uncommitted[<url>,<uuid>]	VMware 虚拟机未提交存储空间 (字节). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机主机名	
vmware.vm.storage.unshared[<url>,<uuid>]	VMware 虚拟机非共享存储空间 (字节). 整型	**url** - VMw	re 服务的 URL 地址 **uuid** - VMware 虚拟机主机名	

键值					
vmware.vm.uptime[<url>,<uuid>]					
	VMware 虚拟机运行时间 (秒). 整数	**url** -	VMware 服务的 URL 地址 **uuid** - VMware 虚拟机主机名		VMware 虚拟机主机名
vmware.vm.vfs.dev.discovery[<url>,<uuid>]					
	自动发现 VMware 虚拟机磁盘设备. JSON 对象	**url** - VM	are 服务的 URL 地址 **uuid** - VMware 虚拟机主机名		
vmware.vm.vfs.dev.read[<url>,<uuid>,<instance>,<mode>]					
	VMware 虚拟机磁盘设备读取统计数据 (每秒字节/操作数). 整型 ^**[2](vmware_ke	s#footnotes) 地址 url** - VMware 服务的 UR	**uuid** - VMware 虚拟机主机名 **instance** - 磁盘设备实例 **mode** - bps (默认)/ops - 每秒字节/操作数		
vmware.vm.vfs.dev.write[<url>,<uuid>,<instance>,<mode>]					
	VMware 虚拟机磁盘设备写入统计数据 (每秒字节/操作数). 整型 ^**[2](vmware_ke	s#footnotes) 地址 url** - VMware 服务的 UR	**uuid** - VMware 虚拟机主机名 **instance** - 磁盘设备实例 **mode** - bps (默认)/ops - 每秒字节/操作数		
vmware.vm.vfs.fs.discovery[<url>,<uuid>]					
	自动发现 VMware 虚拟机文件系统. JSON 对象	**url** - VM	are 服务的 URL 地址 VMware Tools 必须安装在 **uuid** - VMware 虚拟机主机名	户虚拟机上.	
vmware.vm.vfs.fs.size[<url>,<uuid>,<fsname>,<mode>]					

键值			
	VMware 虚拟机系统文件统计信息 (字节/百分比). 整型	**url** - VMware 服	的 URL 地 . 址 VMware Tools 必须 安装在客 户虚拟机 **uuid** - VMware 虚拟机主 机名 **fsname** - 文件系统 名 **mode** - to- tal/free/used/pfree/pused

脚注

[1] VMware 性能计数器路径具格式为 `group/counter[rollup]` ，其中:

- `group` - 性能计数器组, 例如 *cpu*
- `counter` - 性能计数器名称, 例如 *usagemhz*
- `rollup` - 性能计数器汇总类型, 例如 *average*

所以上述示例会给出如下计数器路径：`cpu/usagemhz[average]`

性能计数器组描述、计数器名称和汇总类型可以在VMware 文档中找到

[2] 这些项的值来自 VMware 性能计数器，VMwarePerfFrequency 参数 用于刷新 Zabbix VMware 缓存中的数据：

- vmware.hv.datastore.read
- vmware.hv.datastore.write
- vmware.hv.network.in
- vmware.hv.network.out
- vmware.hv.perfcounter
- vmware.vm.cpu.ready
- vmware.vm.net.if.in
- vmware.vm.net.if.out
- vmware.vm.perfcounter
- vmware.vm.vfs.dev.read
- vmware.vm.vfs.dev.write

更多信息

有关如何配置 Zabbix 以监控 VMware 环境的详细信息，请参阅虚拟机监控。

6 日志文件监控

概述

Zabbix 可以集中监控和分析支持/不支持日志轮询的日志文件。

当日志文件包含某些字符串或字符串模式时，可以使用通知来警告用户。

要监控日志文件，前提：

- 主机上已运行 Zabbix agent
- 设置日志监控项

> **Attention:**
> 被监控日志文件的大小限制取决于大文件支持。

配置

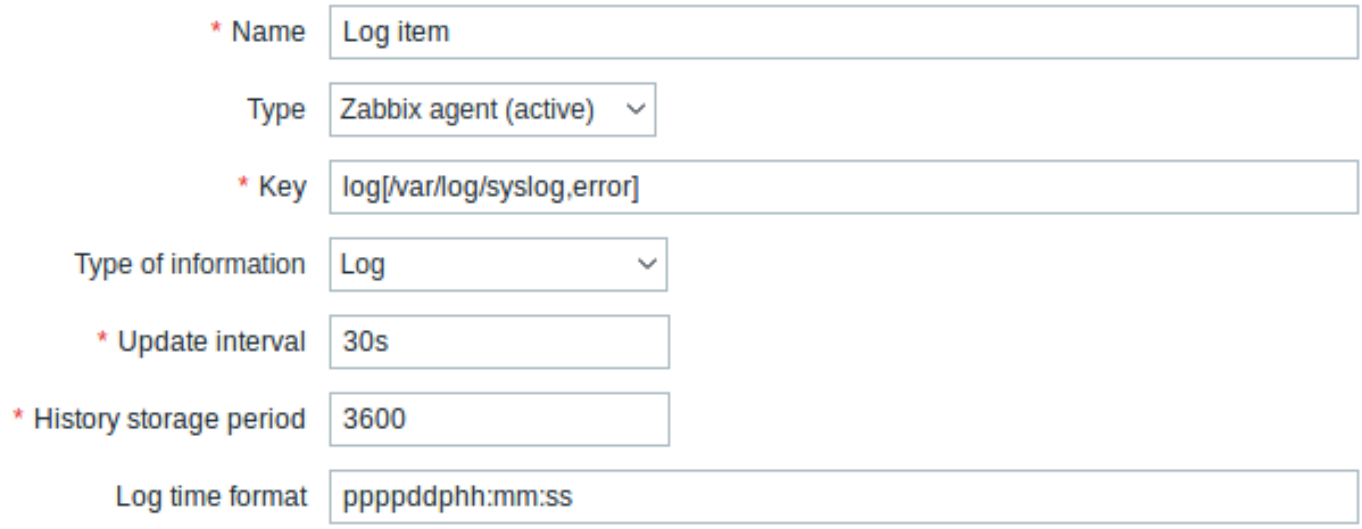
验证代理参数

确保在代理配置文件 中已设置：

- 'Hostname' 参数与前端的主机名一致
- 'ServerActive' 参数中的服务器被指定用于处理主动检查

监控项配置

配置一个日志监控项



所有标有红色星号的为必填字段。

具体日志监控项的输入：

Type	这里选择 **Zabbix agent (active)**
Key	设置: **log**[/path/to/file/file_name,<regexp>,<encoding>,<maxlines>,<mode>, or **logrt**[/path/to/file/regexp_describing_filename_pattern,<regexp>,<encod Zabbix agent 将通过内容正则表达式过（如果存在的话）滤日志文件的条目。 如果只需要匹配行的数量，设置: **log.count**[/path/to/file/file_name,<regexp>,<encoding>,<maxproclines> or **logrt.count**[/path/to/file/regexp_describing_filename_pattern,<regexp>, 确保 zabbix 用户具有文件的读写权限，否则监控项将被设置为“不支持”状态。 更多细节，请在Zabbix agent 监控项的键值章节中查看 **log**, **log.count**, **logrt** 和 **logrt.count** 的条目。
Type of information	在这里，log 和 logrt 选择 Log，log.count 和 logrt.count 选择 Numeric (unsigned). 如果可选使用 `output` 参数，则可以选择除“日志”之外的适当类型的信息. 注意，选择非日志类型的信息将导致本地时间戳的丢失.
Update interval (in sec)	该参数定义了 Zabbix 代理检查日志文件中任何更改的频率。将其设置为 1 秒将确保你能尽快的获得新记录。

Log time format	在此字段中，您可以选择指定解析日志行的时间戳的模式。 如果留空，则不会解析时间戳。 支持的占位符: * **y**: 年 *(0001-9999)* * **M**: 月 *(01-12)* * **d**: 日 *(01-31)* * **h**: 小时 *(00-23)* * **m**: 分 *(00-59)* * **s**: 秒 *(00-59)* 例如, 从 Zabbix agent 日志文件中查看以下行: " 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211)." 它以 PID 的六个字符位置开始，后面跟日期、时间和行的其余部分。 此行的日志时间格式为"pppppp:yyyyMMdd:hhmmss"。 注意，"p" 和 " : " 字符只是占位符，而且只能是 "yMdhms"

注意事项

* 服务器和代理将监视日志的大小和最后修改时间 (对于 logrt) 的跟踪保存在两个计数器中. 此外:

*代理还在内部使用inode编号（在UNIX/GNU/Linux上）、文件索引（在Microsoft Windows上）和前512个日志文件字节
* 在UNIX/GNU/Linux系统中，假定日志文件存储的文件系统是报告inode号，可以用来跟踪文件。
* 在Microsoft Windows系统上，Zabbix agent确定日志文件所在的文件系统类型，并使用:
* 在NTFS文件系统上64位文件索引。
* 在ReFS文件系统上（仅从Microsoft Windows Server 2012开始支持)128位文件ID。
* 在文件索引发生变化(例如FAT32、exFAT)的文件系统中，当日志文件旋转导致多个日志文件在相同的最后修改时
* inode号，文件索引和MD5总和由Zabbix代理在内部收集。 它们不传输到Zabbix服务器，并且在Zabbix代理停止时丢
* 不要使用"touch"程序修改日志文件的最后修改时间，不要在以后恢复原始名称的情况下复制日志文件（这将更改文
*如果logrt[]监控项有几个匹配的日志文件，并且Zabbix agent跟随其中最新的日志文件，并删除了最近的日志文件
* 代理从上次停止的点开始读取日志文件。
* 在代理刚刚启动或已收到以前被禁用或不支持的监控项的情况下，已经分析的字节数（大小计数器）和最后修改时间（
* 每当日志文件变得小于代理已知的日志大小计数器时，计数器将重置为零，代理从开始位置读取日志文件，将时间计数
* 如果目录中存在多个匹配文件，且最后修改时间相同，则代理会尝试以相同的修改时间对所有日志文件进行正确分析，
* Zabbix agent每 //更新间隔// 秒处理一次日志文件的新记录。
* Zabbix agent不会每秒发送超过日志文件的 **最大值**。 该限制可防止网络和CPU资源的过载，并会覆盖 [[:manual:
* 要找到所需的字符串，Zabbix将处理比MaxLinesPerSecond中设置的多10倍的新行。例如，如果"log[]"或"logrt[]"监控
* 此外，即使其中没有非日志值，日志和日志计数值始终限为代理发送缓冲区大小的50％。 因此，为了在一个连接（而不
* 在没有日志项的情况下，所有代理缓冲区大小都用于非日志值。当日志值出现时，它们会根据需要替换旧的非日志值，
* 对于大于256kB的日志文件记录，只有前256kB与正则表达式匹配，而其余部分将被忽略。但是，如果Zabbix agent在处
* 特别注意"\"路径分隔符: 如果file_format是"file\.log", 则不应该有"file"目录, 因为不可能明确地定义"."是转义
* 仅在文件名中支持"logrt"的正则表达式，不支持目录正则表达式匹配。
* 在UNIX平台上，如果要找的日志文件的目录不存在，则logrt[]监控项将变为NOTSUPPORTED。
* 在Microsoft Windows上，如果目录不存在，则监控项将不会变为NOTSUPPORTED（例如，目录在监控项键中拼写错误）。
* 没有用于logrt[]监控项的日志文件不会使其NOTSUPPORTED。读取logrt[]监控项的日志文件的错误将作为告警记录到Za
* Zabbix agent日志文件可以帮助你找出为什么log[]或logrt[]监控项会成为NOTSUPPORTED。Zabbix可以监视其代理日志

提取正则表达式的匹配部分

有时我们可能只想从目标文件中提取感兴趣的值，而不是在找到正则表达式匹配时返回整行。

自 Zabbix 2.2.0 以后，日志监控项能够从匹配的行中提取所需的值。这是在通过 "log" 和 "logrt" 监控项中附加 **output* 参数来实现的。

使用 "output" 参数可以指示我们可能感兴趣的匹配的子组。

例如

```
log[/path/to/the/file,"large result buffer allocation.*Entries: ([0-9]+)",,,,\1]
```

应该可以返回在以下内容中找到的条目数：

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result
buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC
ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

Zabbix 只返回数字的原因是因为这里的'output' 是由\1 定义的，指的是第一个也是唯一的想要的子组：**([0-9]+)**

而且，通过提取和返回数字的能力，该值可用于定义触发器。

使用 maxdelay 参数

日志监控项中的 “maxdelay” 参数允许忽略日志文件中的一些较旧的行，以便在 “maxdelay” 秒内获取最近分析的行。

> **Warning:**
> 指定’maxdelay’>0 可能导致 忽略重要的日志文件记录和错过的报警，只有在必要时才使用。

默认情况下，日志监控项将跟踪出现在日志文件中的所有新行。但是，有些应用程序在某些情况下开始在其日志文件中写入大量的消息。例如，如果数据库或 DNS 服务器不可用，则此类应用程序会向日志文件中注入数千条几乎相同错误消息，直到恢复正常为止。默认情况下，所有这些消息将被完全分析，并将匹配的行发送到配置为 “log” 和 “logrt” 监控项的服务器上。

内置防过载保护包括一个可配置的 “maxlines” 参数（保护服务器免受太多传入匹配的日志行）和 4*’maxlines 限制（保护主机 CPU 和 I/O 免受代理在一次检查中过载）。不过，内置保护有两个问题。首先，向服务器报告大量潜在的不太有用的消息，消耗数据库中的空间。第二，由于每秒分析的行数有限，代理可能会滞后于最新的日志记录数小时。你可能希望尽快了解日志文件中的当前情况，而不是检查数小时的历史记录

这两个问题的解决方案都是是使用了’maxdelay’ 参数。如果指定’maxdelay’> 0，在每次检查处理字节数时，将测量剩余字节数和处理时间。代理根据这些数字，计算估计的延迟 - 分析日志文件中所有剩余记录所需的秒数。

如果延迟不超过 “maxdelay”，那么代理将像往常一样继续分析日志文件。

如果延迟大于 “maxdelay”，那么代理将通过 “跳转” 到一个新的估计位置来忽略日志文件的一个块，以便在 “maxdelay” 秒内分析剩下的行。

请注意，代理甚至不会将忽略的行读入缓冲区，而是计算要在文件中跳转的大致位置。

跳过日志文件行的事实记录在代理日志文件中，如下所示：

```
14287:20160602:174344.206 item:"logrt["/home/zabbix32/test[0-9].log",ERROR,,1000,,,120.0]"
logfile:"/home/zabbix32/test1.log" skipping 679858 bytes
(from byte 75653115 to byte 76332973) to meet maxdelay
```

“to byte” 数字是近似的，因为在 “跳转” 之后，代理将文件中的位置调整到日志行开头，日志行可能在文件中更远或更早。

根据增长速度与分析日志文件的速度的不同，你可能会看到没有” 跳转”、少有或经常” 跳转”、大或小的” 跳转”，甚至每次检查中的” 跳转” 都很小。系统负载和网络延迟的波动也会影响延迟的计算，因此” 跳转” 可以跟上 “maxdelay” 参数。

不推荐设置’maxdelay’ <’update interval’（这可能会导致频繁的”jumps”）

处理 “copytruncatetable” 日志文件旋转的注意事项

带有 “copytruncatetable” 选项的 “logrt” 假定不同的日志文件有不同的记录 (至少它们的时间戳不同)，因此初始块的 MD5(最多 512 字节) 将不同。两个具有相同的 MD5 初始块和的文件意味着其中一个是原始块，另一个是副本。

使用 “copytruncatetable” 选项的 “logrt” 将努力正确处理日志文件副本，而不报告副本。但是，与 logrt[] 监控项更新间隔相比，生成具有相同时间戳的多个日志文件副本、日志文件旋转频率更高、不建议频繁重新启动代理。代理试图合理地处理所有这些情况，但是在所有情况下都不能保证良好的结果。

代理和服务器之间的通信失败时的操作

来自 log[] 和 logrt[] 监控项的每个匹配行以及每个 log.count[] 和 logrt.count[] 监控项检查的结果都需要代理发送缓冲区中指定的 50％区域中的空闲时隙。缓冲区元素定期发送到服务器（或代理服务器），缓冲区可以再次释放。

虽然代理发送缓冲区中的指定日志区域中有空闲时隙，并且代理和服务器（或代理服务器）之间的通信失败，但是日志监控结果在发送缓冲区中累积。这有助于缓解短暂的通信故障。

在较长的通信失败期间，所有日志槽都被占用，并采取以下操作：

- “log[]” 和 “logrt[]” 监控项检查已停止。当通信恢复并且缓冲器中的空闲插槽可用时，从先前的位置恢复检查。若没有匹配的行丢失，稍后再报告.
- 如果 maxdelay=0（默认），则 log.count[] 和 logrt.count[] 监控被停止。这种行为类似于上述的 log[] 和 logrt[] 监控项。请注意，这可能会影响 log.count[] 和 logrt.count[] 结果：例如，一次检查计算出日志文件中有 100 个匹配行，但是由于缓冲区中没有空闲插槽，因此停止检查。当通信恢复时，代理将计数相同的 100 条匹配行，还有 70 条新的匹配行。代理会发送 count=170，就像它们在一次检查中发现的一样。
- log.count[] 和 logrt.count[] 检查与 maxdelay>0：如果在检查期间没有 “跳转”，则行为类似于上述。如果在日志文件行上发生 “跳转”，则保留 “跳转” 之后的位置，同时计算结果被丢弃。因此，即使在通信失败的情况下，代理也试图跟上日志文件的增长速度。

7 可计算监控项

概述

你可以基于其它监控项来创建可计算监控项。

因此，可计算监控项是创建虚拟数据源的一种方式，这些值将根据算术表达式定期计算。所有计算都由 Zabbix 服务器完成，与 Zabbix agent 或 proxy 执行的计算无关。

生成的数据将存储在 Zabbix 数据库中，与其他监控项一样 -这就意味着要存储历史和趋势值，以便快速生成图表。可计算监控项可用于触发器表达式中，由宏或其它实体引用，与任何其它监控项类型相同。

要使用可计算监控项，请选择监控项类型为 **Calculated**。

可配置字段

对于每一台主机，**key** 是唯一的监控项标识符。您可以使用支持的符号创建任何键名。

计算定义应在 公式字段中输入。公式和键值之间实际上没有联系，键值参数在公式中不会以任何方式使用。

一个简单公式的正确语法是：

```
func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)
```

参数定	
func	触发器表达式支持的函数: last, min, max, avg, count 等

参数	定
key	另一监控项的键值，该键值的数据是你想要使用的。它可以被定义为 **key** 或者 **host-name:key**。注意*:* 强烈建议将整个键放在双引号 (“…”) 中，以避免由于键内的空格或逗号而导致错误的解析。\\如果键中也有引用的参数，那么必须使用反斜杠 (\) 来转义这些双引号。请参考下文的示例 **5** 。

参数定	
parameter(s)	功能参数（如果需要）

<note tip> 从可计算监控项公式引用的所有监控项都必须存在并且正在收集数据 (功能和不支持的监控项除外)。此外，如果更改引用项的项键，则必须手动更新正在使用这个键值的公式。:::

Attention:
如果用于引用函数参数或常数，公式中的用户宏 将被扩展。如果引用函数、主机名、监控项键值、键值参数或运算符，用户宏将不会被扩展。

更为复杂的公式可以使用函数、运算符和括号的组合。你可以使用触发器表达式支持的所有功能和运算符。请注意，语法略有不同，但是逻辑和运算符的优先级完全相同。

与触发器表达式不同，Zabbix 根据监控项的更新间隔来处理可计算监控项，而不是在接收到新值时处理。

Note:
如果计算结果是一个浮点值，且如果可计算监控项信息类型是 *Numeric (unsigned)* ，则该值将被修剪为一个整数。

在几种情况下，可计算监控项可能不受支持:

1. 引用的监控项
   - 没有找到
   - 被禁用了
   - 属于一个被禁止的主机
   - 不支持 (查阅例外情况功能和不支持的监控项, 具有不支持的监控项和未知值的表达式 and 运算符)
2. 没有数据来计算一个函数
3. 被零除
4. 使用不正确的语法

在 Zabbix 1.8.1 中引入了对可计算监控项的支持。
从 Zabbix 3.2 开始，可计算监控项在某些情况下可能涉及不支持的监控项，如这些所述功能和不支持的监控项，具有不支持的监控项和未知值的表达式 和运算符 。

用法示例

示例 1

计算根分区上可用磁盘空间的百分比

使用 **last** 功能：

```
100*last("vfs.fs.size[/,free]")/last("vfs.fs.size[/,total]")
```

Zabbix 将获取最新的空闲和总磁盘的空间值，并根据给定的公式计算百分比。

示例 2

计算 Zabbix 处理的数值的 10 分钟的平均值

使用 **avg** 功能：

```
avg("Zabbix Server:zabbix[wcache,values]",600)
```

请注意，长时间使用可计算监控项可能会影响 Zabbix server 的性能。

示例 3

计算 eth0 的总带宽

两个功能综合：

```
last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]")
```

示例 4

计算入站流量的百分比

更为复杂的表达式：

```
100*last("net.if.in[eth0,bytes]")/(last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]"))
```

示例 5

在可计算监控项中正确使用聚合

注意双引号是如何在引号内转义的：

```
last("grpsum[\"video\",\"net.if.out[eth0,bytes]\",\"last\"]") / last("grpsum[\"video\",\"nginx_stat.sh[act
```

8 内部检查

概述

内部检查可以监控 Zabbix 的内部进程。换句话说，你可以监控 Zabbix server 或 Zabbix proxy 的运行情况。

内部检查是：

- 在 Zabbix server 上 - 主机是否被服务器监控
- 在 Zabbix proxy 上 - 主机是否被代理服务器监控

内部检查由服务器或代理服务器执行，无论主机维护状态如何（从 Zabbix 2.4.0 起）

要使用此监控项，请选择 **Zabbix internal** 监控项类型。

> **Note:**
> 内部检查由 Zabbix 轮询器处理。

支持的检查

- 没有尖括号的参数是常量 - 例如, `zabbix[host,<type>,available]` 中的'host' and 'available'. 在监控项键值中使用它们。
- 仅当主机被服务器监控时，才能收集 "代理服务器不支持" 的监控项和监控项参数的值。反之亦然，"服务器不支持" 的值只能在代理监视主机时收集。

键值				
▲	描述	返	值注释	
zabbix[boottime]	Zabbix server 或 Zabbix proxy 进程启动时间（秒）	整数		
zabbix[history]	存储在 HISTORY 表中的数量值.	整数.	如果使用 MySQL	nnoDB, Oracle or Post-greSQL，请勿使用! *(代理服务器不支持)*
zabbix[history_log]	存储在 HIS-TORY_LOG 表中的数量值	整数.	如果使用 MySQL	nnoDB, Oracle or Post-greSQL，请勿使用! 从 Zabbix **1.8.3** 开始支持此监控项 *(代理服务器不支持)*
zabbix[history_str]				

键值				
	存储在 HIS-TORY_STR 表中的数量值	整数.	如果使用 MySQL	nnoDB, Oracle or Post-greSQL，请勿使用! *(代理服务器不支持)*
zabbix[history_text]				
	存储在 HIS-TORY_TEXT 表中的数量值	整数	如果使用 MySQL	nnoDB, Oracle or Post-greSQL，请勿使用! 从 Zabbix **1.8.3** 开始支持此监控项 *(代理服务器不支持)*
zabbix[history_uint]				
	存储在 HIS-TORY_UINT 表中的值数	整数	如果使用 MySQL	InnoDB, Oracle or Post-greSQL，请勿使用! 从 Zabbix **1.8.3** 开始支持此监控项 *(代理服务器不支持)*
zabbix[host„items]				
	主机上启用的监控项的数量 (受支持和不受支持) .	整数	从 Zabbix **3.0.0.** 开始支持	监控项
zabbix[host„items_unsupported]				
	主机上启用的不受支持的监控项数量	整数	从 Zabbix **3.0.0.*	开始支持此监控项
zabbix[host„maintenance]				
	当前主机的维护状态	0 - 主机处于	常状态，此监控项始终由 Zabbix 服务器 1 - 主机处于维护状态但采集数据，第二个参数必须为空，并保 2 - 主机处于维护状态不采集数据. 此监控项从 Zabbix*	理，无论主机位置如何（在服务器或代理服务器上）。代理将不会使用配置数据接收该监控项。供将来使用。 2.4.0.** 开始支持

键值				
zabbix[host,discovery,interfaces]	Zabbix frontend 中主机所有配置接口的详细信息	JSON 对象	此监控项可以在 [低级发现](	manual/discovery/low_l 中使用 此监控项从 Zab-bix**3.4.0.** 开始支持 *(代理服务器不支持)*
zabbix[host,<type>,available]	主机上特殊类型的检查。该监控项的值对应于主机列表中的可用性图标	0 - 不可用, 1 - 可用, 2 - 未知.	有效的类型是: **agent**, **snmp**, **ipmi**	**jmx**. 监控项的值根据有关主机不可达/不可用 的配置参数计算. 此监控项从 Zab-bix**2.0.0.** 开始支持
zabbix[hosts]	已监控主机数量.	整数	此监控项从 Zab	ix**2.2.0** 开始支持.
zabbix[items]	已启用监控项的数量（受支持和不受支持的）	整数		
zabbix[items_unsupported]	不支持的监控项数量	整数		
zabbix[java„<param>]				

键值				
	有关 Zabbix Java 网关的信息	如果 <	aram> 为 **ping**, 则返回 “1”. 可以使用 nodata () 触发功能来检查 Java 网关的可用性。 <param> 的有效值是: *ping*, *ver 如果 <param> 是 **version**, 则返回 Java 网关的版本。例如: ”2.0.0”. 第二个参数必须为空，并保留	ion* 将来使用。 此监控项从 Zab-bix**2.0.0.** 开始支持
zabbix[preprocessing_queue]				
	预处理队列中队列数量	整数	此监控项可用于监控预处	队列长度 \\此监控项从 Zab-bix**3.4.0.** 开始支持
zabbix[process,<type>,<mode>,<state>]				

键值					
	时间是一个特定的 Zabbix 进程或一组进程（由 <type> 和 <mode> 标识），以百分比形式在 <state> 中使用。仅在最后一分钟计算。\\如果 <mode> 是没有运行的 Zabbix 进程号 (例如，运行 <mode> 的 5 个轮询器被指定为 6), 则此监控项将变为不受支持的状态。最小和最大值是指单个进程的使用百分比。因此，如果在一组 3 个轮询器中，每个进程的使用百分比为 2,18 和 66，则 min 将返回 2，max 将返回 66。\\进程报告它们在共享内存中所做的事情，而自我监视进程每秒都会对这些数据进行汇总。状态改变 (忙/空闲) 在更改时被注册 - 因此一个进程变得繁忙，并且直到状态变为空	< 报警任务管理器	时间百分比 浮点数	目前支持以下进程类型: **alerter** - 发送通知的进程 *(代理服务器不支持)* **configuration**	<yncer** - 用于管理配置数据的内存中缓存的进程 **data sender** - 代理服务器数据发送者 *(不支持 Zabbix server)* **discoverer** - 设备发现进程 **escalator** - action 升级进程 *(代理服务器不支持)* **heartbeat sender** - 代理服务器心跳发送方 *(不支持 Zabbix server)* **history syncer** - 历史数据库写入者 **housekeeper** - 删除旧历史数据的进程 **http poller** - web 轮询检查器 **icmp pinger** - icmpping 轮询检查器 **ipmi manager** - IPMI 轮询管理 **ipmi poller** - IPMI 轮询检查器 **java poller** - Java 检查轮询器 **poller** - 被动检查的通用轮询器 **preprocessing manager** - 预处理任务管理

键值				
zabbix[proxy,<name>,<param>]	有关 Zabbix proxy 的信息.	整数	<na	e> - 代理服务器名 支持的参数列表 (<param>): lastaccess - 从代理服务器上收到的最后心跳消息的时间戳 示例: => zabbix[proxy,”Germany”,la **fuzzytime()** 触发器函数 可用于检查代理的可用性。 此监控项从 Zabbix 2.4.0 开始支持，该监控项始终由 Zabbix 服务器处理，无论主机位置如何 (在服务器或代理服务器上)。
zabbix[proxy_history]	代理服务器历史表中等待发送到服务器的值的数量。	整数	此监控项从 Zabbix **2.2.0** 开始支	。 *(不支持 Zabbix server)*
zabbix[queue,<from>,<to>]	队列中被监视的监控项数量至少延迟了从 <from> 秒，但小于 <to> 秒.	整数	<from> - 默认: 6 秒	«to> - 默认: 无限 Time-unit symbols (s,m,h,d,w) 被这些参数支持 参数 `from` 和 `to` 从 Zabbix **1.8.3.** 开始支持
zabbix[rcache,<cache>,<mode>]				

键值				
	Zabbix 配置缓存的可用性统计信息	整数（大小）；浮点数（	分比）缓存: **buffer**	<Mode: **total** - 缓冲区的总大小 **free** - 可用缓冲区大小 **pfree** - 可用缓存区百分比 **used** - 已用的缓存区大小
zabbix[requiredperformance]				
	Zabbix server 或 Zabbix proxy 所需的性能，以每秒新增的值计算.	浮点数	与 *Reports → [系统信息]	/manual/web_interface/ 中的 “所需服务器性能，每秒新值” 大致相关。 此监控项从 Zabbix **1.6.2.** 开始支持
zabbix[trends]				
	存储在 TRENDS 表中的数量值	整数	如果使用 MySQL	nnoDB、Oracle 或 Post-greSQL，请勿使用！ *(代理服务器不支持)*
zabbix[trends_uint]				
	存储在 TRENDS_UINT 表中的数量值	整数	如果使用 MySQL	nnoDB、Oracle 或 Post-greSQL，请勿使用！ 此监控项从 Zabbix **1.8.3.** 开始支持 *(代理服务器不支持)*
zabbix[triggers]				
	Zabbix 数据库中启用的触发器数量，在启用的主机上启用所有的监控项	整数	*(代理服务器不支持)*	
zabbix[uptime]				
	Zabbix server 或 zabbix proxy 正常运行时间（秒）.	整数		
zabbix[vcache,buffer,<mode>]				

键值				
	Zabbix 值缓存的可用性统计信息	整数（大小）；浮点数	百分比） 模式:	**total** - 缓冲区的总大小 **free** - 可用缓冲区大小 **pfree** - 可用缓冲区百分比 **used** - 已用的缓冲区大小 **pused** - 已用的缓冲区百分比 此监控项从 Zabbix **2.2.0** 开始支持 *(代理服务器不支持)*
zabbix[vcache,cache,<parameter>]				
	Zabbix 值缓存的有效性统计	整数	参数: 使用模式参数: **hit0 -** 正常模式**,** m1 - 低内存模式 **mo	**requests** - 总请求数量 ** - 缓存命中数（从缓存中取出的历史值） sses** - 高速缓存未命中数（从数据库获取的历史值） e** - 值缓存操作模式 此监控项从 Zabbix **2.2.0** 开始支持，模式参数从 Zabbix **3.0.0** 开始支持 *(代理服务器不支持)* 您可以使用这个键来进行 每秒更改预处理步骤，以便获得每秒统计值。
zabbix[vmware,buffer,<mode>]				

键值					
	Zabbix vmware 缓存的可用性统计信息		整数（大小）；浮点	（百分比）模式:	**total** - 缓冲区的总大小 **free** - 可用缓冲区大小 **pfree** - 可用缓冲区百分比 **used** - 已用的缓冲区大小 **pused** - 已用的缓冲区百分比 此监控项从 Zabbix **2.2.0** 开始支持
zabbix[wcache,<cache>,<mode>]	Zabbix 写缓存的统计和可用性			必须指定 <c	che>
	缓存 *	模式 **			
	values	all (默认)	由 Zabbix server 或 Zabbix proxy 处理的值的总数（不支持的监控项除外）整数	计数器	您 < 以使用这个键来进行 每秒更改预处理步骤，以便获得每秒统计值。
		float	处理的浮点值的数量. 整数	计数器	
		uint	处理的无符号整数值的数量. 整数	计数器	
		str	处理的字符/字符串值的数量 整数	计数器	
		log	处理日志值的数量 整数	计数器	
		text	已处理文本值的数量整数	计数器	
		not supported	项目处理导致监控项不受支持或保持该状态的次数. 整数	计数器	*Not supported* 模式从 Zabbix **1.8.6.** 开始支持

键值					
	history	pfree (默认)	可用历史缓冲区的百分比. 浮点数	历史缓存用于存储监控项值。	比较低表示数据库端会有性能问题。
		free	可用历史缓冲区大小整数		
		total	历史缓冲区总大小整数		
		used	已用的历史缓冲区大小整数		
	index	pfree (默认)	可用的历史索引缓冲区的百分比浮点数	历史索引缓存用于索引存储在历史缓 *	中的值。引 * 缓存从 Zabbix **3.0.0** 开始支持
		free	可用历史索引缓冲区的大小整数		
		total	历史记录索引缓冲区的总大小整数		
		used	已用的历史索引缓冲区的大小整数		
	trend	pfree (默认)	可用趋势缓存的百分比浮点数	趋势缓存存储接收数据的所 *	监控项的当前小时的聚合。代理服务器不支持)*
		free	可用趋势缓存大小整数	*(代理服务器不支	)*
		total	趋势缓存总大小整数	*(代理服务器不	持)*
		used	已用的趋势缓存大小整数	*(代理服务器不支持	*

9 SSH 检查

Overview

概述

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

SSH 检查不依赖于 Zabbix agent，可对无 agent 代理的设备进行监控。

To perform SSH checks Zabbix server must be initially configured with SSH2 support.

Attention:
The minimum supported libssh2 library version is 1.0.0.

要执行 SSH 检查操作，Zabbix server 必须支持 SSH2。<note important>libssh2 库的最低版本是 1.0.0. :::

Configuration

配置

Passphrase authentication

密码验证

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

SSH 检查提供了两种身份验证方式，一种是用户/密码对，另一种是基于密钥文件的验证方式。

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

如果你不打算使用密钥，除了将 libssh2 连接到 Zabbix，就不需要额外的配置了（如果是源码安装）。

Key file authentication

密钥文件认证

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

要对 SSH 监控项使用基于密钥的身份验证，需要对服务器配置进行某些更改。

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

以 root 身份打开 Zabbix server 的配置文件，查找以下行

```
# SSHKeyLocation=
```


```
# SSHKeyLocation=
```


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

取消注释，配置公钥和私钥所在文件夹的完整路径：

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


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


Save the file and restart zabbix_server afterwards.

保存文件并重启 zabbix_server 服务

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

/home/zabbix 在这里是 *zabbix* 用户的主目录；*.ssh* 是一个目录，由 ssh-keygen 这个命令产生的公钥和密钥将默认放到这个目录中。

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

不同发行版操作系统的 zabbix-server 安装程序，会在不太明显的地方（与系统账户一样）创建一个带有主目录的 zabbix 用户账户。例如，对于 CentOS 系统，在 */var/lib/zabbix* 位置，而 Debian 系统则是在 */var/run/zabbix* 。

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

在生成密钥之前，可以考虑将主目录重新分配到更熟悉的地方（更为直观），与上述提到的 Zabbix server 配置中 *SSHKeyLocation* 的参数对应。

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

如果根据安装章节 手动添加了 zabbix 账户，则这些步骤可以省略，因为在这种情况下，主目录很可能已经是位于 */home/zabbix* 。

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

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

要更改 *zabbix* 账户的设置，必须停止所有正在使用它的进程：

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

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

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

要更改主目录的位置，以尝试移动它（如果存在），要执行一条命令：

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

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

```
# test -d /home/zabbix || mkdir /home/zabbix
```

在旧的地方不存在主目录是完全可能的，因此需要在新的地方创建。一个安全的做法是：

```
# test -d /home/zabbix || mkdir /home/zabbix
```

To be sure that all is secure, additional commands could be executed to set permissions to the home directory:

```
# chown zabbix:zabbix /home/zabbix
# chmod 700 /home/zabbix
```

为确保一切都是安全的，可以执行其他命令来设置主目录的权限：

```
# chown zabbix:zabbix /home/zabbix
# chmod 700 /home/zabbix
```

Previously stopped processes now can be started again:

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

之前被停止的进程现在可以重新启动了：

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

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

现在，可以通过如下命令来生成公钥和私钥：

```
# sudo -u zabbix ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zabbix/.ssh/id_rsa):
Created directory '/home/zabbix/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zabbix/.ssh/id_rsa.
Your public key has been saved in /home/zabbix/.ssh/id_rsa.pub.
The key fingerprint is:
90:af:e4:c7:e3:f0:2e:5a:8d:ab:48:a2:0c:92:30:b9 zabbix@it0
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|       .         |
|      o          |
| .     o         |
|+     . S        |
|.+   o =         |
|E .   * =        |
|=o . ..* .       |
|... oo.o+        |
+-----------------+
```

Note: public and private keys (*id_rsa.pub* and *id_rsa* respectively) have been generated by default in the */home/zabbix/.ssh* directory which corresponds to the Zabbix server *SSHKeyLocation* configuration parameter.

请注意：在默认情况下，公钥和私钥 (分别为 *id_rsa.pub* 和 *id_rsa*) 生成在 */home/zabbix/.ssh* 目录，这与 Zabbix server 配置中 *SSHKeyLocation* 的参数是对应的。

> **Attention:**
> Key types other than "rsa" may be supported by the ssh-keygen tool and SSH servers but they may not be supported by libssh2, used by Zabbix.

<note important>ssh-keygen 工具和 SSH 服务器除了 “rsa” 之外，也可支持其他密钥类型，但 Zabbix 使用的 libssh2 可能不支持它们。
:::

Shell configuration form

Shell 配置方式

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

对于每台被 SSH 检测的主机，此步骤只需要执行一次。

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

通过使用以下命令，公钥会安装到远程主机 *10.10.10.10* 上，以便可以使用 *root* 账户执行 SSH 检查：

```
# sudo -u zabbix ssh-copy-id root@10.10.10.10
The authenticity of host '10.10.10.10 (10.10.10.10)' can't be established.
RSA key fingerprint is 38:ba:f2:a4:b5:d9:8f:52:00:09:f7:1f:75:cc:0b:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.10' (RSA) to the list of known hosts.
root@10.10.10.10's password:
Now try logging into the machine, with "ssh 'root@10.10.10.10'", and check in:
  .ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

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

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

现在可以使用 *zabbix* 用户的默认私钥 (*/home/zabbix/.ssh/id_rsa*) 检查 SSH 登陆了：

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

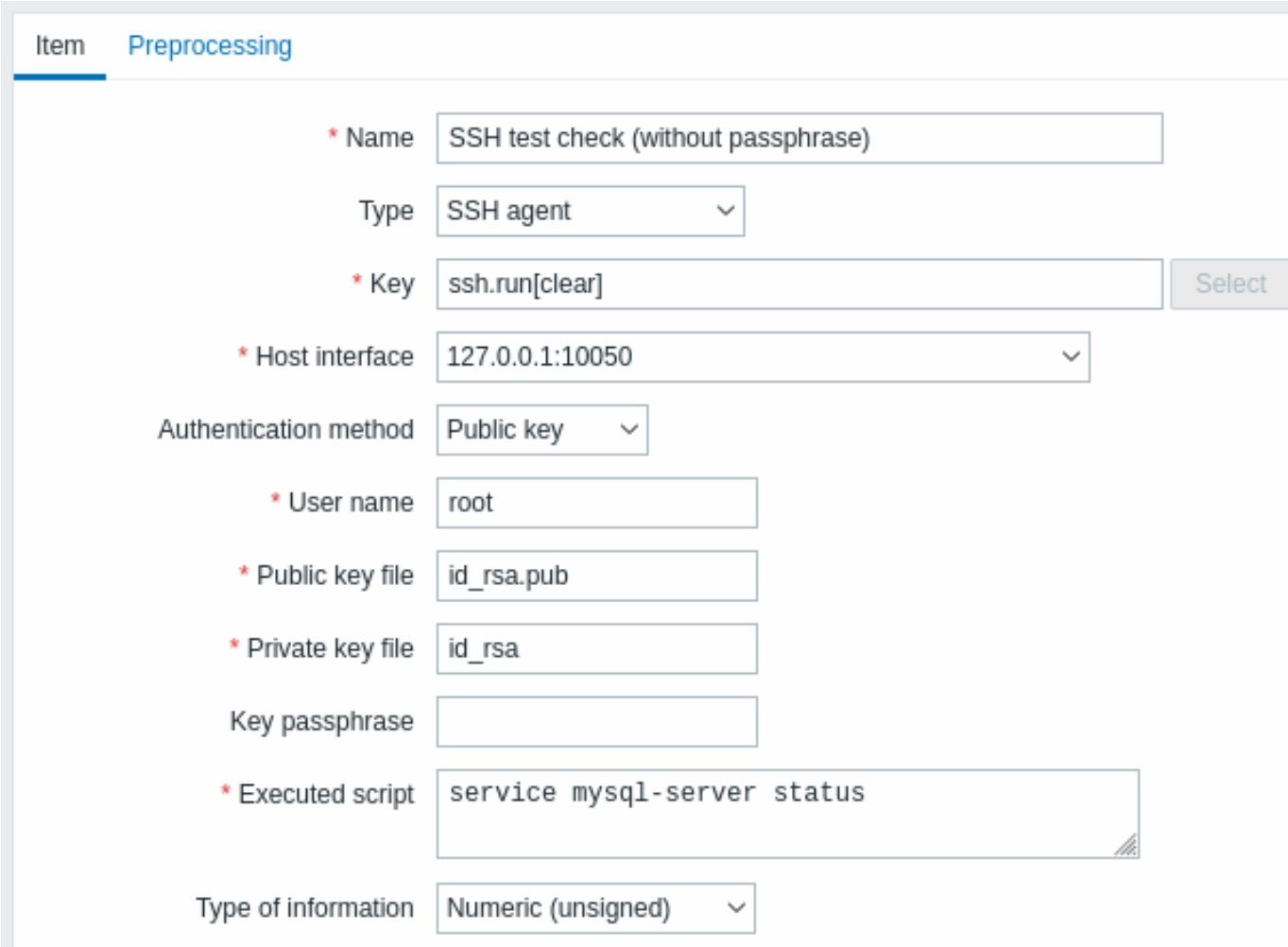
If the login is successful, then the configuration part in the shell is finished and remote SSH session can be closed.

如果登陆成功，那么 Shell 中的配置部分就完成了，并可以关闭远程 SSH 会话。

Item configuration

监控项配置

Actual command(s) to be executed must be placed in the **Executed script** field in the item configuration.
Multiple commands can be executed one after another by placing them on a new line. In this case returned values also will be formatted as multi lined.
要执行的实际命令必须放在监控项配置的 执行脚本中。
如要执行多条命令，在执行脚本字段中一行写一条，命令将会逐条执行。这种情况下，返回值也将为多行显示。



All mandatory input fields are marked with a red asterisk.

所有标有红色星号的为必填项。

The fields that require specific information for SSH items are:

需要为 SSH 监控项提供特定信息的字段是：

Parameter	Description	Comments
Type	Select **SSH agent** here.	
Key	Unique (per host) item key in format **ssh.run[<unique short description>,<ip>,<port>,<encoding>]**	<unique short description> is required and should be unique for all SSH items per host Default port is 22, not the port specified in the interface to which this item is assigned
Authentication method	One of the ”Password” or ”Public key”	
User name	User name to authenticate on remote host. Required	
Public key file	File name of public key if *Authentication method* is ”Public key”. Required	Example: *id_rsa.pub* - default public key file name generated by a command ssh-keygen
Private key file	File name of private key if *Authentication method* is ”Public key”. Required	Example: *id_rsa* - default private key file name
Password or *Key passphrase*	Password to authenticate or Passphrase **if** it was used for the private key	Leave the *Key passphrase* field empty if passphrase was not used See also known issues regarding passphrase usage

Parameter	Description	Comments
Executed script	Executed shell command(s) using SSH remote session	Examples: *date +%s* *service mysql-server status* *ps auxww | grep httpd | wc -l*

参数描	注释
Type	在这里选择 **SSH agent**

参数描	注释	
Key	格式为 **ssh.run[<unique short description>,<ip>,<port>,<encoding>]** 每台主机唯一的监控项键值 <unique sho	t de- scrip- tion 参数是必须的，对于每台主机的所有 SSH 监控项都应该是唯一的默认端口为 22，而不是分配给该监控项的接口中指定的端口
Authentication method	”密码”认证或者”公钥”认证，两者选其一	
User name	在远程主机上进行身份验证的用户名 必填项	

参数描	注释	
Public key file	如果 身份验证方式为 “公钥”，此处则为公钥的文件名。必填项示例: *id_rsa.pub* - 由 [ss	- keygen](http://er keygen) 命令生成的默认公钥文件名
Private key file	如果 身份验证方式为 “公钥”，此处则为私钥的文件名. 必填项示例: *id_rsa* - 默认私钥文件名	
Password or *Key passphrase*	如果密码用于私钥，则验证密码或密码短语如果没有使用密码短语，则将 * 密码短	* 字段留空关于密码短语的使用，另请参阅已知问题
Executed script	使用 SSH 远程会话执行 shell 命令示例:	*date +%s service mysql-server sta-tus ps auxww | grep httpd | wc -l*

Attention:
libssh2 library may truncate executable scripts to ~32kB.

<note important>libssh2 库可能会将可执行脚本截断到 ~32kB :::

10 Telnet 检查

概述

Telnet 检查不需要安装 Zabbix agent，可对未安装代理的主机进行监控。

可配置字段

要执行的实际命令必须放在监控项配置中的 执行脚本字段中。
如要执行多条命令，一行写一条，命令将逐条执行。这种情况下，返回值也将为多行显示。

支持的 shell 提示符可以是：

- $
- #
-
- %

Note:
以这些字符之一结尾的 telnet 提示行将从返回值中删除，但只用于命令列表中的第一个命令中，即仅在 telnet 会话开始处。

键值描	注释
telnet.run[<unique short description>,<ip>,<port>,<encoding>]	使用 telnet 连接在远程设备上运行命令

<note important> 若 telnet 检查返回的是非 ASCII 字符的值，又是非 UTF8 编码，那么应该正确指定键值中 *encoding* 参数。详细信息请参阅返回值的编码 。:::

11 外部检查

概述

外部检查是由 Zabbix server 通过运行 shell 脚本 或是二进制文件执行的检查。然而当主机是通过 Zabbix proxy 监控时，外部检查则由 Zabbix proxy 执行。

外部检查不需要在被监控的主机上运行任何代理。

监控项键值的语法：

```
script[<parameter1>,<parameter2>,...]
```

Where:

参数定	
script	shell 脚本或二进制文件的名称
parameter(s)	可选的命令行参数

如果你不想将任何参数传递给脚本，可以使用：

```
script[] or
script
```

Zabbix server 将查找外部脚本位置的目录 (Zabbix server 配置文件 中'ExternalScripts' 的参数)，然后执行该命令。该命令将以 Zabbix 用户执行，因此任何访问权限或环境变量都应该在包装器脚本中处理，并且该命令的权限应允许该用户执行它。只有指定目录中的命令才可执行。

Warning:
不要过度使用外部检查! 由于每个脚本都需要 Zabbix server 启动一个 fork 进程, 运行太多的脚本会降低 Zabbix 的性能。

使用示例

使用第一个参数 "-h" 执行 **check_oracle.sh** 脚本，第二个参数将被 IP 地址或 DNS 名称替换，这取决于主机属性中的选择。

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

假设主机配置为使用 IP 地址，Zabbix 将执行：

```
check_oracle.sh '-h' '192.168.1.4'
```

外部检查结果

检查的返回值与标准错误一起通过标准输出（从 zabbix 2.0 开始，返回完整输出，并去掉了末尾的空格）

<note important> 在标准错误输出的情况下，文本（字符、日志或文本信息类型）的监控项将被支持. :::

如果没有找到所请求的脚本，或者 Zabbix server 没有执行该脚本的权限，则不支持该监控项，并将设置相应的错误消息。在超时的情况下，监控项也将被标记为不受支持，并显示相应的错误消息，脚本的分支进程将被杀死。

12 聚合检查

概述

在聚合检查中，Zabbix 通过直接从数据库中查询监控信息，然后进行信息聚合。

聚合检查不需要在被监控主机上运行任何代理。

语法

聚合监控项键值的语法是：

```
groupfunc["host group","item key",itemfunc,timeperiod]
```

支持的组函数：

组函数	描述
grpavg	平均值
grpmax	最大值
grpmin	最小值
grpsum	值的总和

可以通过插入以逗号分隔的数组来包含多个主机组。指定父主机组将包括父组和所有包含监控项的嵌套主机组。

从聚合监控项键值引用的所有监控项必须存在并且正在收集数据。只有主机和监控项都被启用才能进行聚合计算。

<note important 如果引用监控项的键值被更改，则必须手动更新聚合监控项的键值. :::

支持的监控项函数：

监控项函数	描述
avg	平均值
count	数值
last	最后一次的值
max	最大值
min	最小值
sum	值的总和

timeperiod 参数指定最近收集的值的时间周期。为方便起见，可以在此参数中使用支持的单位符号。例如，使用'5m' (分钟) 来代替'300' (秒)，或者是用'1d' (天) 来代替'86400' (秒)。

Warning:
在该段时间内，不支持多个数值（前缀为 **#** ）.

如果第三个参数（监控项函数）是 *last* ，服务器将忽略 Timeperiod，因此可以省略：

`groupfunc["host group","item key",last]`

> **Note:**
> 如果聚合产生的是一个浮点数，同时聚合的监控项信息类型为 *Numeric (unsigned)* ，则该值将会被修剪为整数。

如果出现以下情况，聚合监控项可能会变成不支持状态：

- 没有找到引用的监控项 (监控项键值不正确、监控项不存在或是所有包含的组都不正确时，可能会发生此情况)
- 没有数据用来计算一个函数

用法示例

用户聚合检查的键值示例：

示例 1

'MySQL Servers' 主机组的磁盘总空间

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

示例 2

'MySQL Servers' 主机组处理器的平均负载

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

示例 3

'MySQL Servers' 主机组 5 分钟内平均每秒查询数量

`grpavg["MySQL Servers",mysql.qps,avg,5m]`

示例 4

多个主机组中所有主机 CPU 负载的平均值

`grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last]`

13 捕捉器监控项

概述

捕捉器监控项接收传入的数据，而不是查询它。

对于任何你想要推送到 Zabbix 的数据都是使用的。

要使用捕捉器监控项，你需要：

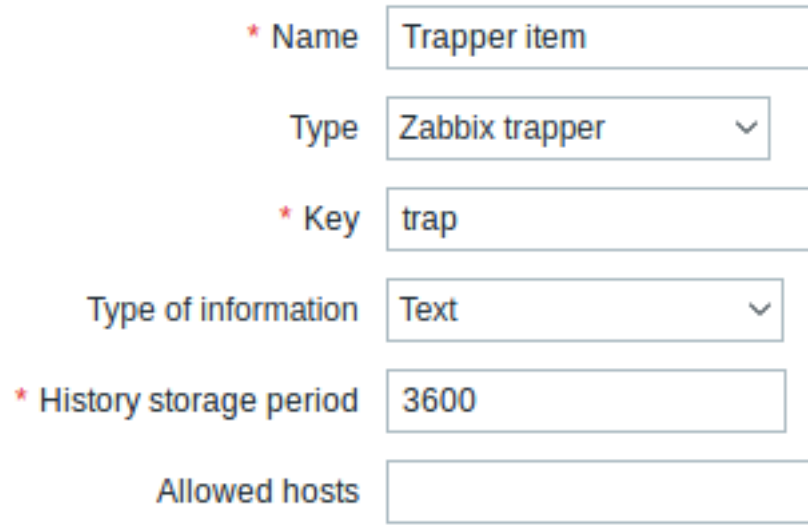
- 在 Zabbix 中建立一个捕捉器监控项
- 将数据发送到 Zabbix

配置

监控项配置

配置捕捉器监控项：

- 进入: *Configuration* → *Hosts*
- 在主机的那一行，点击 *Items*
- 点击 *Create item*
- 输入表单中监控项的参数



标有红色星号的为必填字段

需要捕捉器监控项的特定信息的字段是：

Type	这里选择 **Zabbix trapper**
Key	输入一个用于在发送数据时识别该监控项的键.
Type of information	选择与将要发送的数据格式相对应的信息类型
Allowed hosts	以逗号分隔的 IP 地址列表或主机名，可选择以 CIDR 表示法. 如果指定, 那么只有从这些指定的主机传入的连接才会被接受. 如果启用了 IPv6，'127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' 是一样的，'::/0' 将允许任何 IPv4 或 IPv6 地址。 '0.0.0.0/0' 可用于允许任何 IPv4 地址. 注意, "IPv4 兼容的 IPv6 地址"（0000::/96 前缀）能够被支持，但 RFC4291不推荐使用。 示例: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain 从 Zabbix 2.2.0 开始，允许使用空格和user macros 。

Note:
在保存监控项之后，您可能需要等待最多 60 秒的时间，直到服务器从配置缓存更新中获取更改，然后才能发送值。

数据发送

在最简单的情况下，我们可以使用zabbix_sender 程序来发送一些 "测试值"：

```
zabbix_sender -z <server IP address> -p 10051 -s "New host" -k trap -o "test value"
```

我们使用下列这些键来发送值

-z - 指定 Zabbix server 的 IP 地址

-p - 指定 Zabbix server 的端口（默认为 10051）

-s -指定主机（请确保在此使用 "技术含义" 的主机名 ，而不是 "可见" 名称）

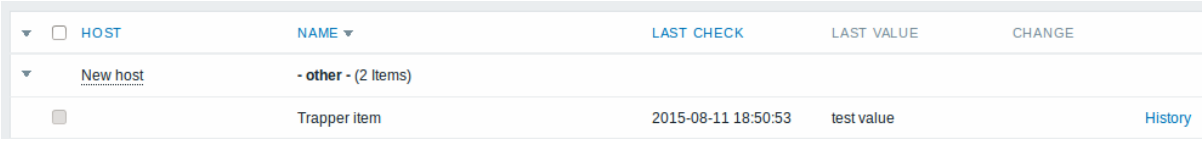
-k - 指定我们之前定义的监控项的键值

-o - 指定要发送的实际值

Attention:
Zabbix trapper 进程不会扩展监控项键值中使用的宏，以检查目标主机对应的监控项键值是否存在。

展示

这是 *Monitoring* → *Latest data* 的结果



时间戳

如果使用 “zabbix_sender” 从具有时间戳的文件发送值，那么将会调整这些时间戳以匹配服务器的时间。例如，如果一个监控项的时间戳是 “10:30:50”，zabbix_sender 机器上的当前时间是 “10:40:03”，Zabbix server 主机的时间是”10:40:05”，那么该监控项的值将以 “10:30:52” 的时间戳存储在数据库中。

类似地，如果一个值首先发送到 Zabbix proxy，然后再将其发送到 Zabbix server，那么时间戳将首先会被调整以匹配 Zabbix proxy 的时间，然后再被调整以匹配 Zabbix server 的时间。

14 JMX 监控

概述

JMX 监控可用于监控 Java 应用程序的 JMX 计数器。

从 zabbix 2.0 开始，JMX 监视器以 Zabbix 守护进程的形式运行，称为 “Zabbix Java gateway”。

要检索某台主机特定 JMX 计数器的值，Zabbix server 查询 Zabbix **Java** 网关，进而使用 JMX management API 来远程查询相关应用。

有关更多细节和设置，请参考Zabbix Java 网关 这一章节。

<note warning>Java 网关和 JMX 应用程序之间的通信不应被防火墙阻止. :::

为 Java 应用程序启用远程 JMX 监控

Java 应用程序不需要安装任何附加的软件，但需要使用以下指定的命令行，设置启动，以支持远程 JMX 监控。

最小化的情况下，如果你只希望通过在本地主机上监控一个简单的 Java 应用程序，不考虑其安全性，那么可以使用以下设置进行启动：

```
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

这使得 Java 可以侦听来自本地主机 12345 端口上传入的 JMX 连接，并告知不需要身份验证或 SSL。

如果要允许其它接口上的连接，请将-Djava.rmi.server.hostname 参数设置为该接口的 IP。

如果您对安全性有更严格的要求，可以使用许多其他的 Java 设置。例如，下一个示例以一组更通用的设置启动应用程序，适用于更广泛的网络，而不仅仅是本地主机。

```
java \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-openjdk/management/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

这些设置的大部分（或许全部）可以在/etc/java-6-openjdk/management/management.properties 文件中指定（或者此文件在系统的其他存放）。

请注意，如果您希望使用 SSL，则必须通过向 Java 网关添加 -Djavax.net.ssl.* 选项来修改 startup.sh 脚本，以便知道在哪里可以找到密钥和信任存储。

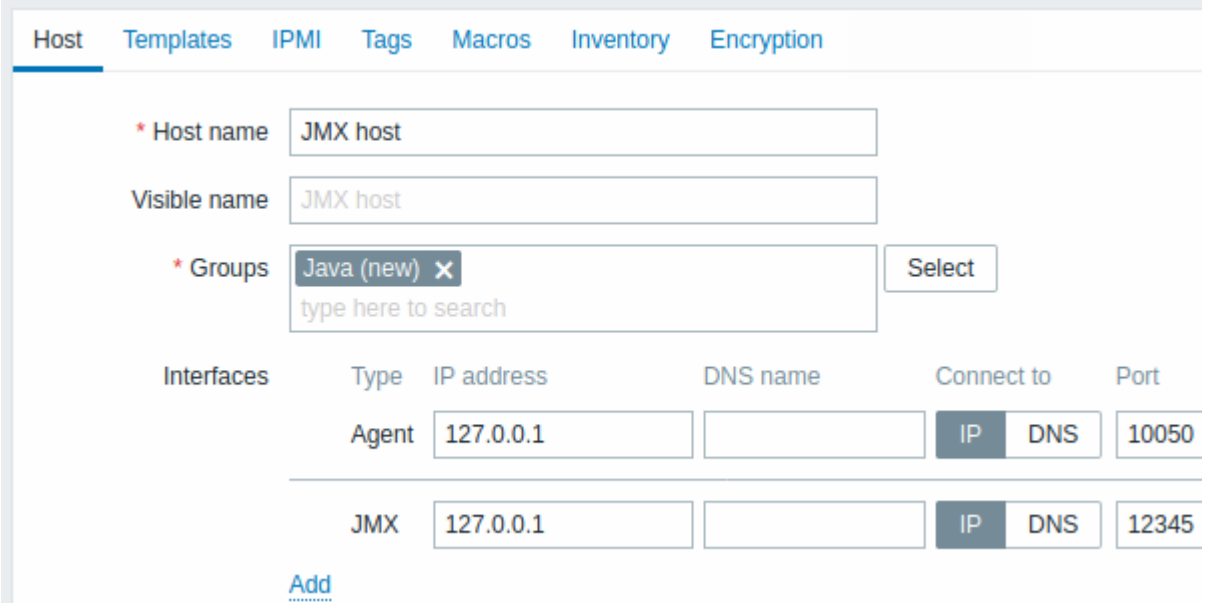
详细说明请参考 使用 JMX 监控和管理 。

在 Zabbix web 管理页面上配置 JMX 接口和监控项

Java 网关在运行时，服务器知道在哪里找到它，并且 Java 应用程序开始了远程 JMX 监视，现在可以在 Zabbix GUI 中配置接口和监控项了。

配置 JMX 接口

首先在相关主机上创建一个 JMX 类型的接口。

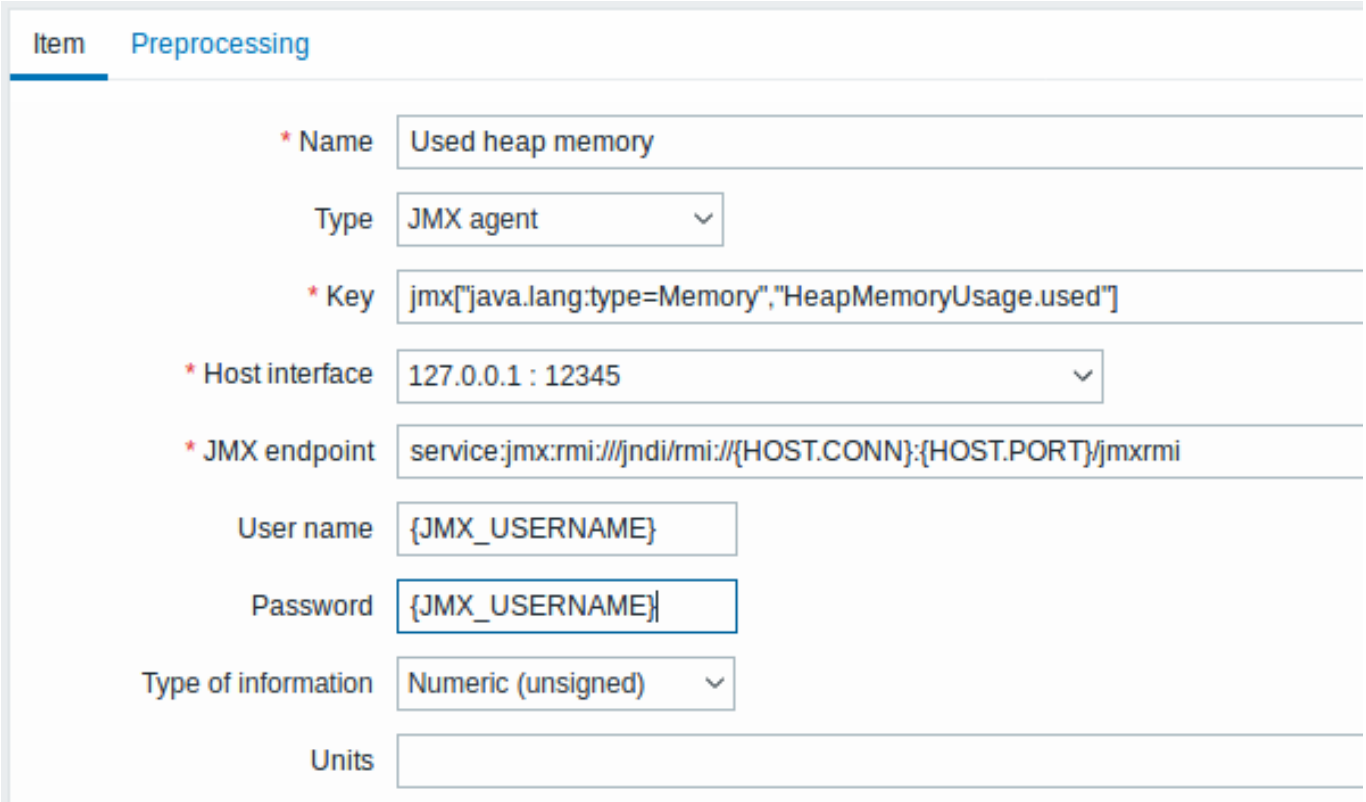


标有红色星号的为必填项。

添加 JMX 代理监控项

对于你感兴趣的每个 JMX 计数器，都可以在接口上添加一个 **JMX** 代理类型的监控项。

下面截图中的键值参数是这样配置的 `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`。



标有红色星号的为必填项。

需要 JMX 监控项特定信息的字段，如下：

Type	这里设置为 **JMX agent**

Key	jmx[] 监控项键值包含两个参数: **object name** - MBean 的对象名; **attribute name** - 一个 MBean 属性名，可选的复合数据字段名由点分隔 有关 JMX 监控项键值的更多细节，请参见下面内容. 从 Zabbix 3.4 开始, 你可以使用 jmx.discovery[] 低级发现监控项来自动发现 MBeans 和 MBean 属性.
JMX endpoint	您可以指定一个自定义的 JMX 端点，确保 JMX 端点连接参数与 JMX 接口匹配。这可以通过在默认 JMX 端点中使用 {HOST.*} 宏来实现。 This field is supported 从 Zabbix 3.4.0 开始支持此字段。支持 {HOST.*} 宏 和用户宏。
User name	如果在 Java 应用程序上配置了身份验证，请指定用户名. 支持用户宏
Password	如果在 Java 应用程序上配置了身份验证，请指定密码. 支持用户宏

如果要监控一个 “true” 或 “false” 的布尔值计数器，那么你需要将信息类型指定为”Numeric (unsigned)”，在预处理选项卡中选择”Boolean to decimal” 预处理步骤，服务器将分别将布尔值存储为 1 或 0。

JMX 监控项详细信息

简单属性

MBean 对象名只不过是 Java 应用程序中定义的字符串。另一方面，属性名可能更为复杂。如果一个属性返回原始数据类型，这并没有什么可担心的。这个键值会是这样的：

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

在这个示例中，对象名是 “com.example:Type=Hello”，属性名是 “weight”，返回值的类型可能是 “Numeric (float)”。

属性返回复合数据

当属性返回复合数据时将变得更加复杂。例如：属性名是 “apple”，它返回一个表示其参数的哈希，如”weight”, ”color” 等。键值可能如下所示：

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

这就是使用点符号分隔属性名和哈希键的方法。同理，如果属性返回嵌套的复合数据，则各部分之间用点分隔：

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

Problem with dots

关于点的问题

到目前为止都还好。但是，如果属性名或散列键包含点符号呢? 下面就是个例子：

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

如何告诉 Zabbix 属性名是”all.fruits”，而不只是 “all” 呢？如何区分作为属性名称一部分的点与分隔属性名和散列键的点呢？这是一个问题。

在 **2.0.4** 版本之前，Zabbix Java 网关是无法处理此类情况的，在监控项里，用户只能留下 UNSUPPORTED 项了。从 2.0.4 开始解决了此问题，你所需要做的就是用反斜杠来转义名字的一部分点：

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

同样，如果哈希键包含一个点，你也可以转义它：

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

其他问题

属性名中的反斜杠字符特应该被转义：

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

有关处理 JMX 监控项键值中的其他特殊字符，请参见this section。这就是全部了，祝 JMX 监控快乐！

15 ODBC 监控

概述

ODBC 监控对应于 Zabbix 前端中的 数据库监视器监控项类型。

ODBC 是 C 语言编写的中间件 API，用于访问数据库管理系统 (DBMS)。ODBC 是由 Microsoft 开发的，后来被移植到了其它平台。

Zabbix 可以查询任何支持 ODBC 的数据库。为此，Zabbix 不直接连接数据库，而是使用 ODBC 接口和在 ODBC 中设置的驱动程序。该功能允许出于多种目的，更加有效地监视不同的数据库。例如，检测特定的数据库队列、使用统计信息等。Zabbix 支持 unixODBC，是最常用的开源 ODBC API 实现之一。

安装 unixODBC

安装 unixODBC 建议的方式是使用 Linux 操作系统默认的软件包仓库。在最流行的 Linux 发行版中，unixODBC 默认是包含在软件包仓库中的。如果没有，可以在 unixODBC 主页获取：http://www.unixodbc.org/download.html

使用 *yum* 软件包管理器在基于 RedHat/Fedora 的系统上安装 unixODBC：

```
shell> yum -y install unixODBC unixODBC-devel
```

使用 *zypper* 软件包管理器，在基于 SUSE 的系统上安装 unixODBC：

```
# zypper in unixODBC-devel
```

Note:
编译 Zabbix 以支持 unixODBC 功能时，需要使用到 unixODBC-devel 这个包。

安装 unixODBC 驱动

应该为将要被监控的数据库安装 unixODBC 数据库驱动。unixODBC 有一个支持的数据库和驱动程序列表:http://www.unixodbc.org/drivers.html□□□□Linux□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ *yum* 软件包管理器，在基于 RedHat/Fedora 的系统上安装 MySQL 数据库驱动：

```
shell> yum install mysql-connector-odbc
```

使用 zypper 软件包管理器在基于 SUSE 的系统上安装 MySQL 数据库驱动程序：

```
zypper in MyODBC-unixODBC
```

配置 unixODBC

通过编辑 **odbcinst.ini** 和 **odbc.ini** 文件来完成 ODBC 配置。要确认配置文件位置，请键入：

```
shell> odbcinst -j
```

odbcinst.ini 用于列出已安装的 ODBC 数据库驱动程序：

```
[mysql]
Description = ODBC for MySQL
Driver      = /usr/lib/libmyodbc5.so
```

参数详细信息：

属性描	
mysql	数据库驱动名称
Description	数据库驱动描述
Driver	数据库驱动程序库位置

odbc.ini 用来定义数据源

```
[test]
Description = MySQL test database
Driver      = mysql
Server      = 127.0.0.1
User        = root
Password    =
Port        = 3306
Database    = zabbix
```

参数详细信息：

属性描	
test	数据源名称 (DSN)
Description	数据源描述.
Driver	数据库驱动名称 - 在 odbcinst.ini 文件中指定
Server	数据库服务器的 IP/DNS
User	用于数据库连接的用户名
Password	数据库用户的密码
Port	数据库连接端口
Database	数据库名称

要验证 ODBC 连接是否正常运行，应测试到数据库的连接。可以使用 **isql** 程序（包含在 unixODBC 软件包中）：

```
shell> isql test
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

编译支持 ODBC 的 Zabbix

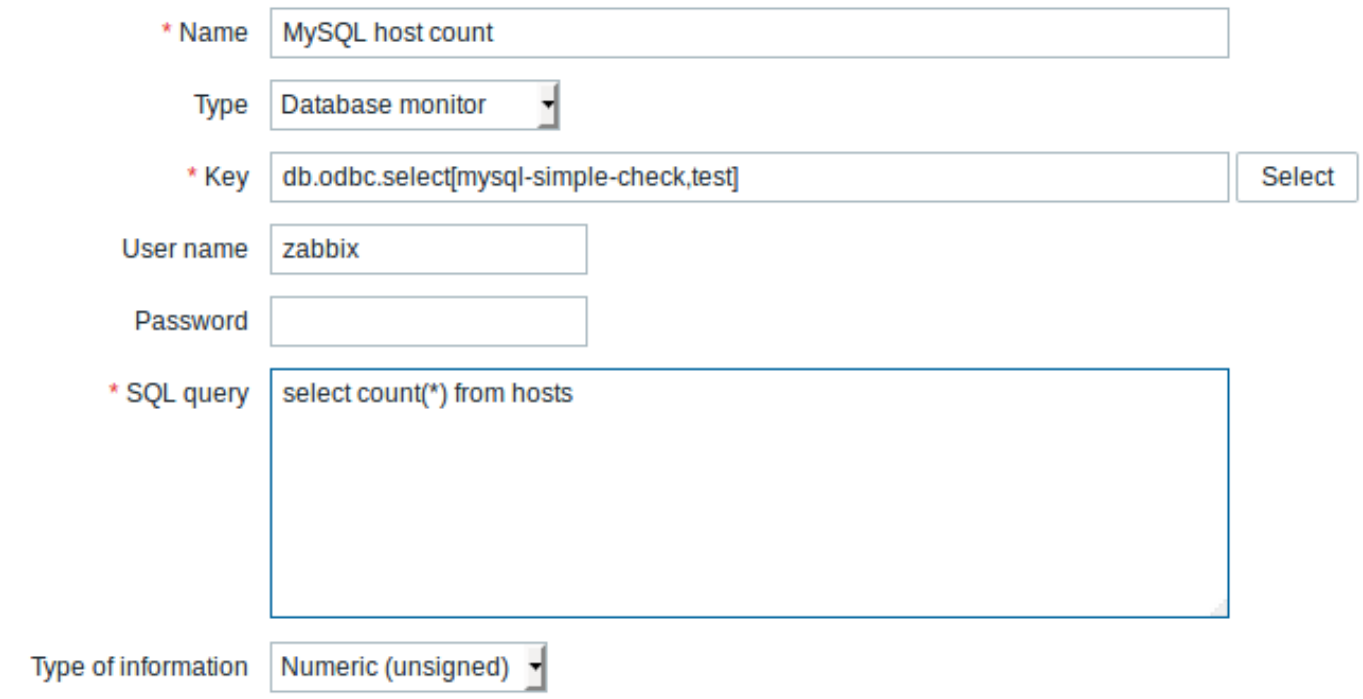
要启用 ODBC 支持，Zabbix 应该使用以下标志进行编译：

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

Note:
更多关于 Zabbix 安装信息请参考源代码。

在 Zabbix 前端配置监控项

配置数据的监控项



所有标有红色星号的为必填字段。

对数据库监控，必须输入的监控项：

Type	这里选择 数据库监控器
Key	输入 **db.odbc.select**[unique_description,data_source_name] 这里唯一的描述将用于识别触发器中的监控项等 数据源名称 (DSN) 必须按照 odbc.ini 中指定的方式设置。
User name	输入数据库用户名 (如果用户在 odbc.ini 中已指定，此项可选填)
Password	输入数据用户密码 (如果用户在 odbc.ini 中已指定，此项可选填)
SQL query	输入 SQL 查询
Type of information	了解查询返回的信息类型很重要, 以便在此处选择正确的类型。 若使用不正确的 信息类型监控项将不受支持。

注意事项

- Zabbix 不限制查询执行时间。用户可以选择在合理时间内执行的查询。
- Zabbix server 的Timeout 参数值也用作于 ODBC 登陆超时时间 (请注意，根据 ODBC 驱动，登录超时设置可能会被忽略)。
- 查询只能返回一个值。
- 如果查询返回多个列，则只读取第一列。
- 如果查询返回多行，则只读取第一行。
- SQL 命令必须以 `select` 开头。
- SQL 命令不能包含任何换行符。
- 另请参阅 ODBC 检查的已知问题

Error messages

错误信息

ODBC 错误消息被构造成字段，以提供详细信息。例如：

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

                                                  Native error code              Native error message
                                              SQLState
              Zabbix message     ODBC return code
```

请注意，错误消息长度限制为 2048 字节，因此信息可以被截断。如果有多个 ODBC 诊断记录，只要长度限制允许，Zabbix 将尝试把它们连接起来（用 “|” 分隔)。

1 MySQL 推荐的 UnixODBC 设置

安装

```
*** Red Hat Enterprise Linux/CentOS**:

# yum install mysql-connector-odbc

***Debian/Ubuntu**:
```

请参考 MySQL 文档 来下载相应平台必要的数据库驱动。

如需其他相关信息，请参阅安装 unixODBC 。

配置

通过编辑 **odbcinst.ini** 和 **odbc.ini** 文件来完成 ODBC 的配置。这些配置文件可以在*/etc* 文件夹中找到。**odbcinst.ini** 文件可能不存在，这时我们需要手动来创建它。

odbcinst.ini

```
[mysql]
Description = General ODBC for MySQL
Driver      = /usr/lib64/libmyodbc5.so
Setup       = /usr/lib64/libodbcmyS.so
FileUsage   = 1
```

请考虑以下 **odbc.ini** 配置参数的示例。

- 通过 IP 连接的示例：

```
[TEST_MYSQL]
Description = MySQL database 1
Driver  = mysql
Port = 3306
Server = 127.0.0.1
```

- 通过 IP 连接并使用凭据的示例，默认使用 zabbix 数据库：

```
[TEST_MYSQL_FILLED_CRED]
Description = MySQL database 2
Driver  = mysql
User = root
Port = 3306
Password = zabbix
Database = zabbix
Server = 127.0.0.1
```

- 通过套接字连接并使用凭据的示例，默认使用 zabbix 数据库：

```
[TEST_MYSQL_FILLED_CRED_SOCK]
Description = MySQL database 3
Driver  = mysql
User = root
Password = zabbix
Socket = /var/run/mysqld/mysqld.sock
Database = zabbix
```

所有其他可能的配置参数选项都可以在网站上找到：MySQL official documentation

2 PostgreSQL 数据库推荐的 UnixODBC 设置

安装

```
*** Red Hat Enterprise Linux/CentOS**:

# yum install postgresql-odbc

***Debian/Ubuntu**:
```

请参考 PostgreSQL 文档 来下载相应平台必要的数据库驱动。

如需其他相关信息，请参阅安装 unixODBC 。

配置

通过编辑 odbcinst.ini 和 odbc.ini 文件来完成 ODBC 的配置。这些配置文件可以在/etc 文件夹中找到。odbcinst.ini 文件可能不存在，这时我们需要手动来创建它。

请考虑以下 **odbc.ini** 配置参数的示例。

odbcinst.ini

```
[postgresql]
Description = General ODBC for PostgreSQL
Driver      = /usr/lib64/libodbcpsql.so
Setup       = /usr/lib64/libodbcpsqlS.so
FileUsage   = 1
# Since 1.6 if the driver manager was built with thread support you may add another entry to each driver e
# This entry alters the default thread serialization level.
Threading   = 2
```

odbc.ini

```
[TEST_PSQL]
Description = PostgreSQL database 1
Driver  = postgresql
#CommLog = /tmp/sql.log
Username = zbx_test
Password = zabbix
```

```
# Name of Server. IP or DNS
Servername = 127.0.0.1
# Database name
Database = zabbix
# Postmaster listening port
Port = 5432
# Database is read only
# Whether the datasource will allow updates.
ReadOnly = No
# PostgreSQL backend protocol
# Note that when using SSL connections this setting is ignored.
# 7.4+: Use the 7.4(V3) protocol. This is only compatible with 7.4 and higher backends.
Protocol = 7.4+
# Includes the OID in SQLColumns
ShowOidColumn = No
# Fakes a unique index on OID
FakeOidIndex  = No
# Row Versioning
# Allows applications to detect whether data has been modified by other users
# while you are attempting to update a row.
# It also speeds the update process since every single column does not need to be specified in the where c
RowVersioning = No
# Show SystemTables
# The driver will treat system tables as regular tables in SQLTables. This is good for Access so you can s
ShowSystemTables = No
# If true, the driver automatically uses declare cursor/fetch to handle SELECT statements and keeps 100 ro
Fetch = Yes
# Bools as Char
# Bools are mapped to SQL_CHAR, otherwise to SQL_BIT.
BoolsAsChar = Yes
# SSL mode
SSLmode = Yes
# Send tobackend on connection
ConnSettings =
```

3 Oracle 数据库推荐的 UnixODBC 设置

安装

请参阅 Oracle documentation 来获取详细说明。

如需其他相关信息，请参阅安装 unixODBC 。

4 MSSQL 数据库设推荐的 UnixODBC 设置

安装

*** Red Hat Enterprise Linux/CentOS**:

```
# yum -y install freetds unixODBC
```

***Debian/Ubuntu**:

请参考 FreeTDS 用户向导 来下载相应平台必要的数据库驱动。

如需其他相关信息，请参阅安装 unixODBC 。

配置

通过编辑 **odbcinst.ini** 和 **odbc.ini** 文件来完成 ODBC 的配置。这些配置文件可以在*/etc* 文件夹中找到。**odbcinst.ini** 文件可能不存在，这时我们需要手动来创建它。

请考虑以下 **odbc.ini** 配置参数的示例。

odbcinst.ini

```
$ vi /etc/odbcinst.ini
[FreeTDS]
Driver = /usr/lib64/libtdsodbc.so.0
```

odbc.ini

```
$ vi /etc/odbc.ini
[sql1]
Driver = FreeTDS
Server = <SQL server 1 IP>
PORT = 1433
TDS_Version = 8.0
```

16 从属监控项

概述

有时一个监控项一次会收集多个度量，或者同时收集相关度量显得更有意义，例如：

- 单个内核的 CPU 利用率
- 输入/输出的总网络流量

为了允许在几个相关监控项中进行批量度量收集和同时使用，Zabbix 支持从属监控项。从属监控项使用主项在一个查询中同时收集它们的数据。主监控项的新值自动填充依赖监控项的值。

Zabbix 预处理选项可用于从主监控项数据中提取依赖监控项所需的部分。

预处理是由一个 “预处理管理器” 进程管理的，它已经被添加到了 Zabbix 3.4 版本中，与 worker 进程一起执行预处理步骤。来自不同收集器的值（不管是否有预处理），在添加到历史缓存之前，都要经过预处理管理器。基于套字节的 IPC 连接用于数据收集器 (pollers, trappers 等) 和预处理进程之间。

只有 Zabbix server 执行预处理步骤，并处理从属监控项。

任何类型的监控项，甚至是从属监控项，都可以设置为主监控项。附加的从属监控项级别可用于从现有的从属监控项的值中提取较小的部分。

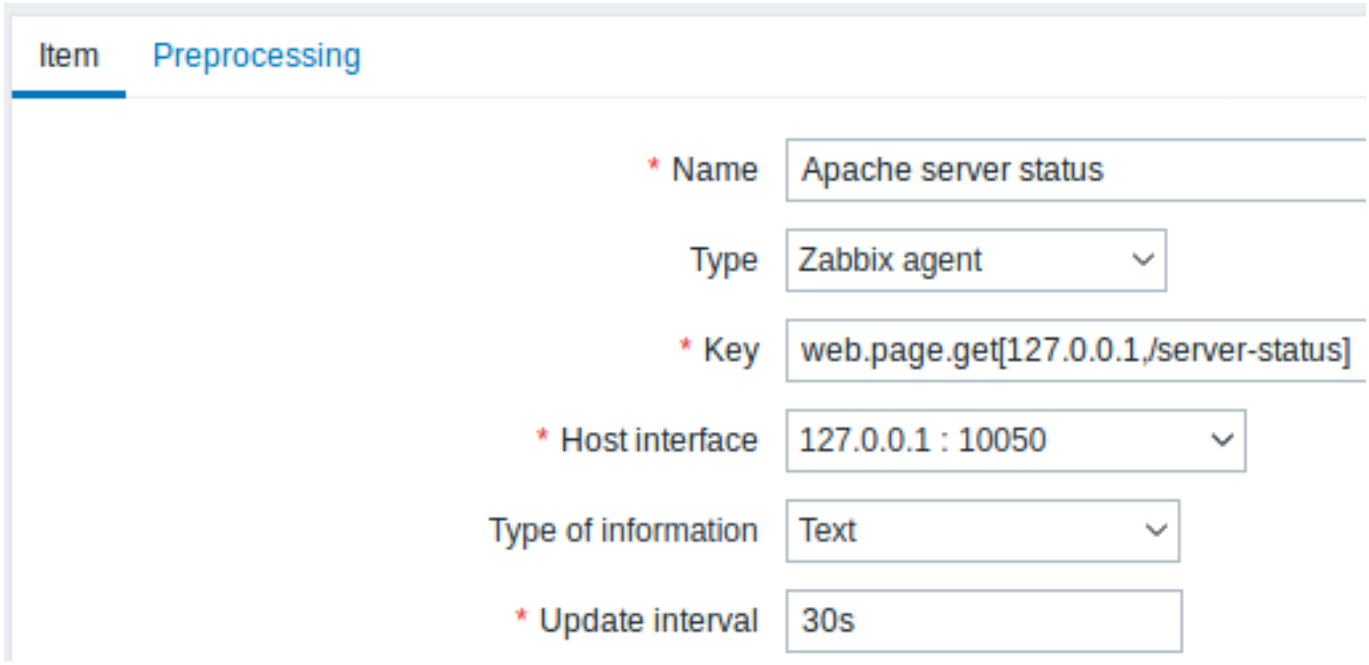
局限性

- 只允许相同的主机（模板）从属项
- 主监控项的从属项最大计数被限制为 999
- 最大允许 3 个从属级别
- 带有主项的从属监控项不能导出到 XML

监控项配置

从属监控项依赖于它主项的数据，这就是为什么必须首先配置 主监控项 (或着已经存在了)

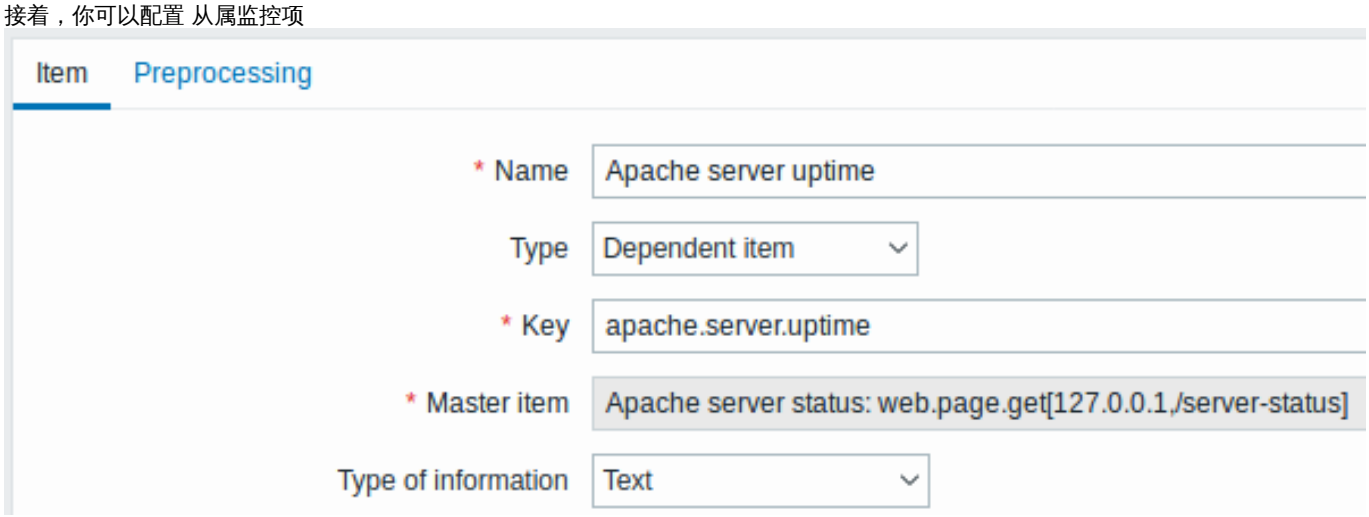
- 进入: *Configuration* → *Hosts*
- 在主机那一行点击 *Items*
- 点击 *Create item*
- 下表中输入监控项的参数



所有标有红色星号的为必填字段。

点击 *Add* 保存主监控项。

接着，你可以配置 从属监控项

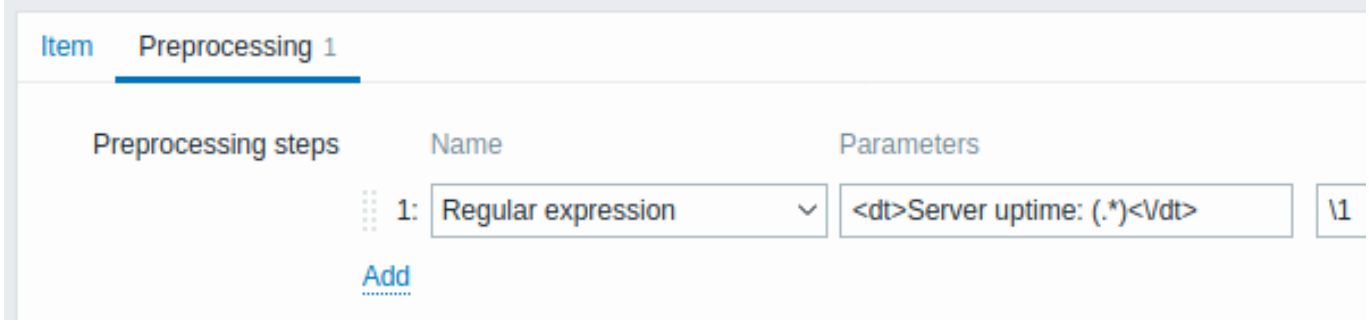


所有标有红色星号的为必填字段。

需要从属监控项的特定信息的字段是：

Type	这里选择 **Dependent item**
Key	输入一个用于识别监控项的键
Master item	选择主监控项。主监控项的值将用于填充从属监控项的值。
Type of information	选择与将要存储的数据格式相对应的信息类型

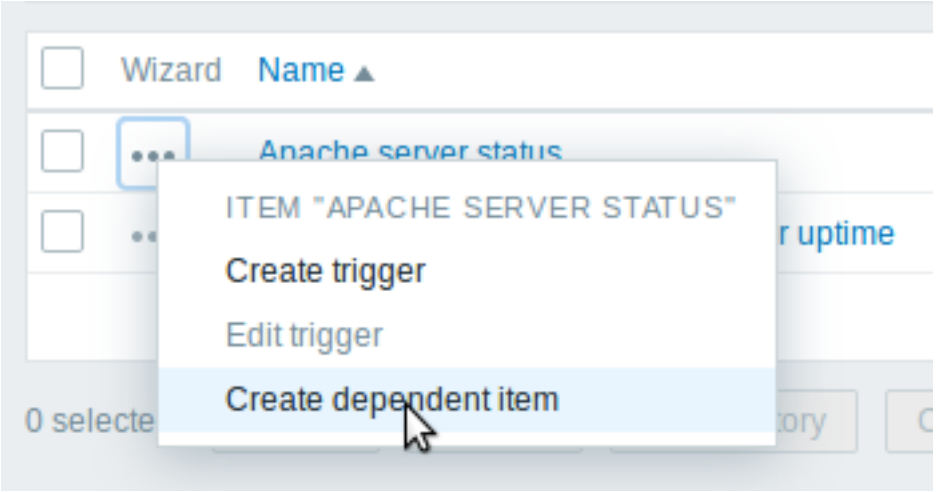
你可以使用监控项的值预处理 来提取主监控项值的所需部分。



如果不进行预处理，从属监控项的值将与主监控的值完全相同。

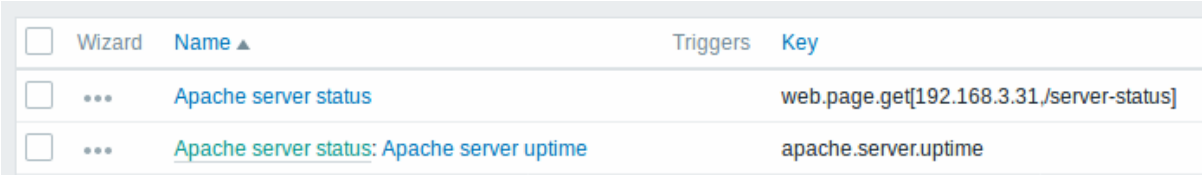
点击 *ADD* 保存从属监控项。

创建从属监控项的快捷方式是使用在监控项列表中的向导



展示

在监控项列表中，从属监控项以其主监控项的名称作为前缀显示。



如果主监控项被删除，那么它的所有从属监控项也将会被删除。

17 HTTP 代理

概述

此监控项类型允许使用 HTTP/HTTPS 协议进行数据轮询。使用 Zabbix sender 或 Zabbix sender 协议也可以进行捕获。

HTTP 代理同时支持 HTTP 和 HTTPS。Zabbix 可以选择跟随重定向（参考下文 *Follow redirects* 的选项）

了解何时使用 HTTPS 协议，另请参阅已知问题

> **Attention:**
> Zabbix server/proxy 必须首先配置 cURL(libcurl) 支持。

配置

配置 HTTP 监控项：

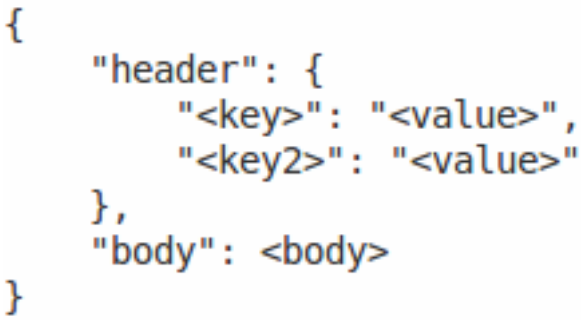
- 进入: *Configuration* → *Hosts*
- 在主机的那行点击 *Items*
- 点击 *Create item*
- 在表格中输入监控项的参数

所有标有红色星号的为必填字段。

需要的 HTTP 监控项特定信息的字段是：

Type	在这里选择 **HTTP agent**
Key	输入一个唯一的监控项键值

URL	连接和检索数据的 URL. 例如: https://www.google.com http://www.zabbix.com/download\\可以用 Unicode 字符指定域名。在执行 web 场景步骤时，它们将自动转换为 ASCII。 *Parse* 可以使用 Parse 按钮将可选查询字段 (比如?name=Admin&password=mypassword) 与 URL 分离，将属性和值移动到查询字段中，以便自动 URL 编码. 限制在 2048 个字符。 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级发现宏 这是设置CURLOPT_URL cURL 选项.
Query fields	URL 的变量 (参见上文). 指定为属性和值对。 值是自动的 URL 编码。从宏中解析值，然后自动编码 url 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级自动发现宏. 设置 cURL 选项 CURLOPT_URL.
Request type	选择请求方法类型: *GET*, *POST*, *PUT* or *HEAD*
Timeout	Zabbix 不会花超过设定的时间来处理 URL (最大 1 分钟)。实际上，这个参数定义了连接 URL 的最大时间和执行 HTTP 请求的最大时间。因此，Zabbix 不会在一次检查中花费超过 2 倍的超时时间。 支持时间后缀, 例如 30s, 1m. 支持的宏: 用户宏, 低级发现宏。 设置 cURL 选项 CURLOPT_TIMEOUT
Request body type	选择请求体类型: **Raw data** - 自定义 HTTP 请求体，替换宏，但不执行编码。 **JSON data** - HTTP 请求体是 JSON 格式的，宏可以用作字符串、数字、真和假; 用作字符串的宏必须包含在双引号中。从宏中解析值，然后自动转义。如果没有指定 header，那么服务器将把默认的 header 值设置为”Content-Type: application/json” **XML data** - HTTP 请求体的 XML 格式。宏可以用作文本节点、属性或 CDATA 部分。从宏中解析值，然后在文本节点和属性中自动转义。如果没有指定 header，那么服务器将把默认的 header 值设置为”Content-Type: application/xml” 注意选择 *XML data*，需要 libxml2 的支持。
Request body	输入请求体 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级自动发现宏。
Headers	执行请求时将发送的自定义 HTTP 头。 指定为属性和值对。 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级自动发现宏。 设置 CURLOPT_HTTPHEADER cURL option.
Required status codes	期望的 HTTP 状态码的列表。如果 Zabbix 得到不在列表中的代码，那么这个项目将不受支持。如果为空，则不执行检查. 例如: 200,201,210-299 列表里支持的宏: 用户宏, 低级自动发现宏。 这个使用了 CURLINFO_RESPONSE_CODE cURL option.
Follow redirects	标记复选框以跟随 HTTP 重定向。 设置 CURLOPT_FOLLOWLOCATION cURL option.
Retrieve mode	选择必须检索的响应部分: **Body** - 仅主体 **Headers** - 仅头部 **Body and headers** - 主体和头部

Convert to JSON	头文件作为属性和值对保存在”header” 键下. 如果遇到’Content-Type: application/json’ 主体被保存为对象，否则它被存储为 string, 例如: 
HTTP proxy	可以使用格式 `http://[username[:password]@]proxy.mycompany.com[:port]` 指定要使用的 HTTP 代理。 默认将使用 1080 端口. 如果指定，代理将覆盖与代理相关的环境变量，如 http_proxy、HTTPS_PROXY. 如果没有指定，代理将不会覆盖与代理相关的环境变量。输入的值将被传递”as is”, 没有进行健全检查。 您还可以输入 SOCKS 代理地址。如果您指定了错误的协议，那么连接将失败，监控项将不受支持。由于没有指定协议，代理将被视为 HTTP 代理。 注意 HTTP 代理只支持简单的身份验证。 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级自动发现宏。 设置 CURLOPT_PROXY cURL option.
HTTP authentication	验证类型: **None** - 不使用身份验证. **Basic authentication** - 使用脚本身份验证. **NTLM authentication** - 使用 NTLM (Windows NT LAN Manager) 验证。 选择身份验证方法将为输入用户名和密码提供两个额外的字段，其中支持用户宏和低级发现宏。 设置 CURLOPT_HTTPAUTH cURL option.
SSL verify peer	标记复选框以验证 web 服务器的 SSL 证书。服务器证书将自动从系统范围的证书颁发机构 (CA) 位置获取。可以使用 Zabbix 服务器或代理配置参数 SSLCALocation 重写 CA 文件的位置。 设置 CURLOPT_SSL_VERIFYPEER cURL option.
SSL verify host	标记复选框以验证 web 服务器证书的通用名称字段或主题备用名称字段是否匹配. 设置 CURLOPT_SSL_VERIFYHOST cURL option.
SSL certificate file	用于客户端身份验证的 SSL 证书文件的名称。证书文件必须是 PEM[1] 格式. 如果证书文件也包含私钥，则将 SSL 密钥文件字段保留为空。如果密钥已加密，请在 SSL 密钥密码字段中指定密码。包含此文件的目录由 Zabbix server 或 zabbix proxy 配置参数 SSLCertLocation 指定。 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级自动发现宏。 设置 CURLOPT_SSLCERT cURL option.
SSL key file	用于客户端身份验证的 SSL 私钥文件的名称。私钥文件必须是 PEM[1] 格式。包含此文件的目录由 Zabbix server 或 zabbix proxy 配置参数 SSLKeyLocation 指定。 支持的宏: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, 用户宏, 低级自动发现宏。 设置 CURLOPT_SSLKEY cURL option.
SSL key password	SSL 私钥文件密码。 支持的宏: 用户宏, 低级自动发现宏 设置 CURLOPT_KEYPASSWD cURL option.

Enable trapping	选中此复选框后，该项目也将作为trapper 监控项项发挥作用，并将接受 Zabbix sender 或使用 Zabbix sender 协议发送给该监控项的数据. 只有勾选了 *Enable trapping* 复选框才可见。
Allowed hosts	由逗号分隔的 IP 地址列表，可选地使用 CIDR 符号或主机名。 \\如果指定，传入连接将仅从这里列出的主机接受。 如果启用了 IPv6，'127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' 这些是一样的，'::/0' 将允许任何 IPv4 或 IPv6 地址。 '0.0.0.0/0' 可用于允许任何 IPv4 地址。 注意, IPv4 兼容的 IPv6 地址 (0000::/96 prefix) 能够被支持，但RFC4291不推荐使用。 示例: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain 在这个字段，空格和用户宏是被允许的。

<note tip> 如果 *HTTP* 代理字段为空，则使用 HTTP 代理的另一种方法是设置与代理相关的环境变量.

对于 HTTP - 为 Zabbix server 用户设置 “http_proxy” 环境变量。例如:
//http_proxy=http:%%//%%proxy_ip:proxy_port//.

对于 HTTPS - 设置 “HTTPS_PROXY” 环境变量. 例如:
//HTTPS_PROXY=http:%%//%%proxy_ip:proxy_port//。可以通过运行 shell 命令获得更多细节: *# man curl*. :::

Attention:
[1] Zabbix 只支持 PEM 格式的证书和私有密匙文件。如果您的证书和私钥数据是 PKCS #12 格式文件 (通常扩展名为 *.p12 or *.pfx) 您可以使用以下命令从它生成 PEM 文件:

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

示例

示例 1

发送简单的 GET 请求来从诸如 Elasticsearch 这样的服务中检索数据:

*使用URL创建一个GET项： ''localhost:9200/?pretty''
*注意其响应

```
{
  "name" : "YQ2VAY-",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "kH4CYqh5QfqgeTsjh2F9zg",
  "version" : {
    "number" : "6.1.3",
    "build_hash" : "af51318",
    "build_date" : "2018-01-26T18:22:55.523Z",
    "build_snapshot" : false,
    "lucene_version" : "7.1.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You know, for search"
}
```

- 现在使用 JSONPath 预处理步骤提取版本号：`$.version.number`

Example 2

示例 2

发送简单的 POST 请求来检索来自 Elasticsearch 等服务的数据：

- 使用 URL 创建一个 POST 项: `http://localhost:9200/str/values/_search?scroll=10s`
- 配置以下 POST 主体以获取处理器负载 (每核 1 分钟的平均值)

```
{
```

```
    "query": {
        "bool": {
            "must": [{
                "match": {
                    "itemid": 28275
                }
            }],
            "filter": [{
                "range": {
                    "clock": {
                        "gt": 1517565836,
                        "lte": 1517566137
                    }
                }
            }]
        }
    }
}
```

- Received :
- 接收：

```
{
    "_scroll_id": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAJRZZUTJWQVktVFN
    "took": 18,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 1.0,
        "hits": [{
            "_index": "dbl",
            "_type": "values",
            "_id": "dqX9VWEBV6sEKSMyk6sw",
            "_score": 1.0,
            "_source": {
                "itemid": 28275,
                "value": "0.138750",
                "clock": 1517566136,
                "ns": 25388713,
                "ttl": 604800
            }
        }]
    }
}
```

- 现在使用 JSONPath 预处理步骤获取项值：`$.hits.hits[0]._source.value`

3 历史数据与趋势数据

概述

历史数据（history）和趋势数据（trends）是 Zabbix 中存储收集到的数据的两种方式。

历史数据：每一个收集到的监控数据
趋势数据：按小时统计计算的平均值数据

历史数据的留存

通过设置历史数据保留时长，可以指定历史数据留存的时长。
在以下位置，你可以找到相关的输入框：

- 监控项配置页-历史数据保留时长
- 在批量更新监控项配置页-历史数据保留时长
- 管家配置页-历史记录-数据存储期

任何过旧的历史数据会被管家从数据库中删除。

一般来讲，强烈建议将历史数据保留时长设置得尽可能的小。这么做可以让数据库不会因存储了大量的历史数据，导致超负荷运行。

可以选择长时间的保留趋势数据，来替代长期需要的历史数据。例如：设置成保留 14 天历史数据和 5 年的趋势数据。

参考数据库空间大小页，来了解历史数据和趋势数据各自需要的数据库空间。

当设置了较短的历史数据保留时间，图形会使用趋势数据值显示旧数据，因此依旧可以通过图形查看旧数据。

<note important> 如果历史数据保留时长被设置为 “0”，那么该监控项将仅可用于更新资产记录。:::

<note tip> 作为保存历史数据的替代方法，考虑使用可加载模块 “导出历史数据” 功能

:::

趋势数据的留存

趋势数据是一种内建的历史数据压缩机制，可以用来存储数字类型监控项的每小时的最小值、最大值、平均值和记录数量。

通过设置趋势存储时间，可以指定趋势数据留存的时长。
在以下位置，你可以找到相关的输入框：

- 监控项配置页-趋势存储时间
- 在批量更新监控项配置页-趋势存储时间
- 管家配置页-趋势-数据存储期

通常趋势数据设置的的留存时间应当比历史数据留存时间设置的长。任何过旧的趋势数据会被管家从数据库删除。

Attention:
如果趋势存储时间被设置为 “0”，Zabbix server 将不再计算或存储该监控项的趋势数据

Note:
趋势数据的计算和存储将会使用与原值相同的数据类型。
无符号数字（unsigned Numeric）数据类型的值，平均值计算的结果小数点后会被舍去，所以记录值之间的间隔越小，计算结果结果将会精确度越低。举个例子：如果监控项的得到了得到了两个值，分别是 “0” 和 “1”，那么平均值的计算结果将会是 “0”，而不是 “0.5”。

此外，重启服务器可能会导致当前小时无符号数字类型的数据，平均值计算的精度损失。

4 用户自定义参数

概述

用户定义参数可以用来帮助用户实现通过 Zabbix agent 执行非 Zabbix 原生的 agent check。

你可以编写一个命令来检索所需的数据，并将其包含在用户自定义参数agent 配置文件中 (’UserParameter’ 参数配置)。

一条用户自定义参数配置应当使用以下语法：

```
UserParameter=<key>,<command>
```

如你所见，一条用户自定义参数除了命令部分，还包括一个 key。这个 key 将在配置监控项时使用。输入你选择的易于引用的 key（key 在一台主机中必须是唯一的）。重启 agent。

接下来，在配置配置监控项时，输入要执行的来自用户自定义参数中的，引用命令的 key。

用户自定义参数是由 Zabbix agent 来执行命令的。在监控项预处理步骤前，最多可以返回 512KB 的数据。但是，请注意，最终可以存储在数据库中的文本值，在 MySQL 上的限制为 64KB（其他数据库的信息请参阅数据表）。

/bin/sh 在 UNIX 操作系统中，作为命令行解释器使用。用户自定义参数参照 agent check 超时；如果超时时间到了，那么执行用户自定义参数的子进程将会被中止。

参见：

- 分布教程 配置用户自定参数 parameters
- 命令执行

用户自定义参数用例

一个简单的命令：

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

agent 将始终为使用 “ping” 为 key 的监控项返回 “1”。

一个复杂一些的例子：

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

如果 Mysql 服务器是活动状态，agent 将返回 “1”，否则会返回 “0”。

灵活的用户自定义参数

灵活的用户自定义参数可以从 key 中接受参数。这是一种使用一个用户自定义参数创建多个监控项的方式。

灵活的用户自定义参数有以下语法：

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

Parameter 参数 D	scription 描述
Key	唯一的监控项 key。[*] 用于定义该 key 接受括号内的参数。 参数需在配置监控项时给出
Command	命令在执行时，引用 key 中指定的值 只对灵活的用户参数有效: 你可以在命令中使用位置引用 $1 … $9 来引用监控项 Key 中的相应参数。 Zabbix 解析监控项 Key 的 [] 中包含的参数，并相应地替换 $1，…，$9。 $0 会替换为完整的原始命令（在对 $0，…，$9 执行替换之前的命令）运行。 不管位置参数（$0,…,$9) 是用双引号 (”) 还是单引号 (’) 括起来，都会解析位置引用。 要使用位置引用解析，请指定双美元符号（$）- 例如，” awk ’{print $$2}’ ” 。在这种情况下，执行命令时，$$2 实际上会变成 $2 。

Attention:
仅对灵活的用户自定义参数进行搜索具有” $ ” 符号的位置引用并由 Zabbix agent 解析替换。对于简单的用户自定义参数，跳过此类参考处理，因此不需要任何 $ 符号引用。

Attention:
默认情况下，不允许用户在用户自定义参数中使用某些特殊符号。详情请移步UnsafeUserParameters ，查询相关的符号列表

示例一

先来一个简单的：

```
UserParameter=ping[*],echo $1
```

我们可以定义无数个监控项来监控所有形如 ping[something] 格式的东西。

- ping[0] - 将总是返回 ‘ 0 ’
- ping[aaa] - 将总是返回 ‘aaa’

示例二

让我们更进一步！

```
UserParameter=mysql.ping[*],mysqladmin -u$1 -p$2 ping | grep -c alive
```

这个用户自定义参数可以用来监控 MySQL 数据库的状态。可以想下面的样式传入用户名和密码：

```
mysql.ping[zabbix,our_password]
```

示例三

一个文件中有多少行匹配正则表达式？

```
UserParameter=wc[*],grep -c "$2" $1
```

这个用户自定义参数能用来计算一个文件中有多少行匹配相应的表达式。就像下面一样：

```
wc[/etc/passwd,root]
wc[/etc/services,zabbix]
```

命令结果

命令的返回值是标准输出和标准错误。

<note important> 标准错误情况下，不支持文本（字符、日志或是文本类型的信息）的监控项:::

返回文本的用户自定义参数（字符，日志，文本信息类型）可以返回空格。如果结果不可用，那么这个监控项会变为不支持状态。

1 扩展 **Zabbix Agents**

本教程提供了有关如何使用用户自定义参数扩展 Zabbix 代理功能的分步说明。

第一步

写一个脚本或命令行以检测所需的参数。

举个例子，我们编辑了下面的命令以获取 MySQL Server 执行的查询总数：

```
mysqladmin -uroot status | cut -f4 -d":" | cut -f1 -d"S"
```

当这个命令被执行，将恢复返回 SQL 查询的总数。

第二步

添加命令到 zabbix_agentd.conf:

```
UserParameter=mysql.questions,mysqladmin -uroot status | cut -f4 -d":" | cut -f1 -d"S"
```

mysql.questions 作为 key 需要是唯一标识符。可以是任何有效的字符，比如 *queries*。

通过使用带有 ‘-t’ 标识的 zabbix_agentd 命令测试此用户自定义参数的执行。(如果是以 root 用户运行，请注意 agent 守护进程的执行者的权限)：

```
zabbix_agentd -t mysql.questions
```

第三步

重启 Zabbix Agent。

Agent 会重载配置文件。

使用zabbix_get 实用程序测试该用户自定义参数。

第四步

在被监控主机中添加使用 key 值为 'mysql.questions' 的新监控项。监控项类型必须使用 Zabbix Agent 或 Zabbix Agent（Active）。

注意在 Zabbix Server 上。必须设置正确的返回值类型，否则 Zabbix 将不会接受它们。

5 可加载模块

概览

可加载模块提供了一个侧重性能的选项，来扩展 Zabbix 的功能。

目前已经有以下的功能来扩展 Zabbix 功能:

- 用户自定义变量 (Agent 指标)
- 扩展检查 (无 Agent 监控)
- `system.run[]` Zabbix Agent 监控项.

这些功能工作的十分优秀，但是存在一个重要的缺陷，名字叫 fork()。在每次处理用户指标的时候都必须创建一个新的子进程，这样不会有优秀的性能表现。通常这并不是个大问题，然而这会在监控嵌入式系统、拥有大量监控参数或运行具有逻辑繁多或启动时间长的脚本的情况下成为一个严重的问题。

可加载模块提供了在不额外消耗性能的情况下，扩展 Zabbix Agent、Server 和 Proxy。

一个可加载模块是基于一个在 Zabbix 守护进程启动时加载的共享库。这个库包含了一些功能，以便 Zabbix 可以检测到该文件确实事一个可以被加载和使用的模块。

可加载模块具有许多有点。出众的性能和实现任何逻辑的能力非常重要，但最重要的能力是开发、使用和分享的 Zabbi 模块。可加载模块有助于实现无故障维护，有助于更轻松的提供新功能并且不依赖于 Zabbix 核心代码库。

二进制形式的模块的授权和分发应在 GPL 许可证的许可下管理（模块运行时连接到 Zabbix 并且使用 Zabbix 的头文件；目前 ZABBIX 的代码根据 GPL 许可证进行授权）。ZABBIX 不保证二进制兼容性。

在一个 ZABBIX LTS(长期支持)版本支持周期内保证 API 模块的稳定性。ZABBIX API 的稳定性无法保证（从技术上讲，可以从模块调用 ZABBIX 内部函数，但不能保证这些模块可以工作）。

模块 API

为了将共享库视作 ZABBIX 模块，它应该实现并导出一些函数。目前，ZABBIX 模块 API 中由六个函数，其中一个是强制性的，另外五个是可选的。

强制接口

唯一的强制函数是 **zbx_module_api_version()**:

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

此函数应该返回实现这个模块以来的 API 版本，并且为了模块能被加载，这个版本必须与 ZABBIX 支持的模块 API 版本匹配。Zabbux 支持的模块 API 的版本为 ZBX_MODULE_API_VERSION。座椅这个函数应该返回这个常量。用于此目的的旧常量 ZBX_MODULE_API_VERSION_ONE，现在被定义为等于 ZBX_MODULE_API_VERSION 以保持源兼容性，但不建议使用它。

1 可选接口

可选的函数是 **zbx_module_init()**, **zbx_module_item_list()**, **zbx_module_item_timeout()**, **zbx_module_history_write_cbs()** and **zbx_module_uninit()**:

```
int zbx_module_init(void);
```

这个函数应该对模块的执行进行必要的初始化（如果有的话）。如果成功，则返回 ZBX_MODULE_OK。否则它应该返回 ZBX_MODULE_FAIL。若为后一种情况，ZABBIX 将无法启动。

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

此函数应当返回一个支持的监控项的列表。每个监控项目被定义为 ZBX_METRIC 的结构下，详细信息请见后文。这个列表应以 “key” 字段为 NULL 作为 ZBX_METRIC 结构的终止。

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

如果模块输出 **zbx_module_item_list()**，那么基于这个模块的监控项会遵守这个函数，而不是遵照 ZABBIX 配置文件中的超时设置。这边，“timeout” 参数以秒为单位。

```
ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void);
```

这个函数应当返回 ZABBIX 服务器将用于导出不同数据类型历史记录的回调函数。回调函数应以 ZBX_HISTORY_WRITE_CBS 结构的字段提供，如果模块对于某种类型的历史纪录不感兴趣，则字段可以为 NULL 。

```
int zbx_module_uninit(void);
```

这个函数应当执行必要的反初始化（如果有的话），如释放分配的资源、关闭文件描述符等。

所有的函数会在 ZABBIX 启动的时候加载模块时，除了 zbx_module_uninit() 都将被调用一次。在卸载磨块时，zbx_module_uninit() 会被 ZABBIX 调用一次。

定义监控项

每个监控项都应当被定义在 ZBX_METRIC 结构中：

```
typedef struct
{
    char        *key;
    unsigned    flags;
    int     (*function)();
    char        *test_param;
}
ZBX_METRIC;
```

这里的 **key** 指的时监控项的 key（例如：“ dummy.random ”），**flags** 可以是 CF_HAVEPARAMS 或 0 (取决于监控项是否接受参数)，**function** 是实现该监控项的 C 函数（例如：”zbx_module_dummy_random”），最后 **test_param** 是使用” -P ” 标志启动 ZABBIX Agent 时使用的参数里列表（例如：”1,1000”, 可以是 NULL）。下面是一个具体示例：

```
static ZBX_METRIC keys[] =
{
    { "dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random, "1,1000" },
    { NULL }
}
```

每个实现一个监控项的函数应该接受俩哥哥指针参数函数，第一个是一种 AGENT_REQUEST 类型，第二个是一种 AGENT_RESULT 类型：

```
int zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    ...

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}
```

如果这个监控项的值被成功获取，这些函数应当返回 SYSINFO_RET_OK 。否则，应当返回 SYSINFO_RET_FAIL。关于如何从 AGENT_REQUEST 获取信息以及如何设定 AGENT_RESULT 的详情，请参阅示例 “dummy” 模块。

提供历史记录输出的回调

从 ZABBIX 4.0.0 开始，不再支持通过 ZABBIX Proxy 经模块输出历史记录:::

模块可以按来行指定输出历史数据的函数：数字（浮点）、数字（无符号）、字符串、文本和日志：

```
typedef struct
{
    void    (*history_float_cb)(const ZBX_HISTORY_FLOAT *history, int history_num);
    void    (*history_integer_cb)(const ZBX_HISTORY_INTEGER *history, int history_num);
    void    (*history_string_cb)(const ZBX_HISTORY_STRING *history, int history_num);
    void    (*history_text_cb)(const ZBX_HISTORY_TEXT *history, int history_num);
    void    (*history_log_cb)(const ZBX_HISTORY_LOG *history, int history_num);
}
ZBX_HISTORY_WRITE_CBS;
```

每个输出历史纪录的函数都应当把"history_num" 元素作为 "history" 数组的参数。依据需要输出的历史记录类型，"history" 分别是以下结构的数组：

```
typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    double      value;
}
ZBX_HISTORY_FLOAT;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    zbx_uint64_t    value;
}
ZBX_HISTORY_INTEGER;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
}
ZBX_HISTORY_STRING;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
}
ZBX_HISTORY_TEXT;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
    const char  *source;
```

```
    int     timestamp;
    int     logeventid;
    int     severity;
}
ZBX_HISTORY_LOG;
```

回调会在 ZABBIX server 的历史记录同步进程完成历史记录同步操作，数据被写入 ZABBIX 数据库并将值保存在值缓存中后执行。

构建模块

目前，模块应当再 ZABBIX 源代码树中构建，因为模块 API 依赖于一些 ZABBIX 头文件中定义的一些数据结构。

对可加载模块来说，最重要的头是 **include/module.h**，它定义了这些住居结构。另一个很有用的头文件 **include/sysinc.h** ，它的执行会包含必要的系统头文件，这有助于 include/module.h 的正常工作。

为了 include/module.h 和 include/sysinc.h 被导入，应在 ZABBIX 源代码树的根目录下执行**./configure** 命令。这将创建 **include/config.h** 文件，其中包含了 include/sysinc.h 依赖。(如果你获得的 ZABBIX 源代码来自子版本存储库，则./configure 脚本尚不存在，应首先运行 **./bootstrap.sh** 脚本来生成它。)

记住这些信息，一切都准备好了去构建模块。该模块应包含 **sysinc.h** 和 **module.h** ，构建脚本应确保这两个文件包含于路径中。有关详细信息，参见下文 “dummy” 模块。

其它有用的头文件 **include/log.h**，它定义了 **zabbix_log()** 函数，可用于记录和调试目的。

配置参数

ZABBIX Agent, Server 和 Proxy 支持两个参数来处理模块：

- LoadModulePath – 可加载模块所在的完整路径
- LoadModule – 启动时加载的模块。这些模块必须位于 LoadModulePath 制定的目录中。允许包含多个 LoadModule 参数

举个例子：要扩展 ZABBIX Agent 我们可以添加以下参数：

```
LoadModulePath=/usr/local/lib/zabbix/agent/
LoadModule=mariadb.so
LoadModule=apache.so
LoadModule=kernel.so
LoadModule=dummy.so
```

在启动 Agent 时，它将从/usr/local/lib/zabbix/agent/目录加载 mariadb.so, apache.so, kernel.so and dummy.so 模块。如果发生缺少模块、权限错误或该共享库文件不是 ZABBIX 模块，那么 Agent 的启动将失败。

前端配置

ZABBIX Agent、Server 和 Proxy 支持可加载模块。因此 ZABBIX 前端中的监控项类型依据模块在哪里被加载。如果模块在 Agent 端被加载那么监控项类型应当设置为 “Agent 检查” 或 “Agent 检查 (主动) ”。如果在 Server 端或 Proxy 端被加载，那么响应的类型应当为 “简单检查”。

通过 ZABBIX 模块历史记录输出不需要进行前端配置。如果模块成功加载并提供 **zbx_module_history_write_cbs()** 函数且该函数应至少返回一个非 NULL 回调方法，则将自动启动历史记录输出。

Dummy 模块

ZABBIX 包含一个用 C 语言编写的示例模块。该模块位于” src/modules/dummy”：

```
alex@alex:~trunk/src/modules/dummy$ ls -l
-rw-rw-r-- 1 alex alex 9019 Apr 24 17:54 dummy.c
-rw-rw-r-- 1 alex alex   67 Apr 24 17:54 Makefile
-rw-rw-r-- 1 alex alex  245 Apr 24 17:54 README
```

这个模块由详细的文档，可以作为您编写自己的模块的模板。

如上所述，在 ZABBIX 源代码根目录下运行./configure 命令后，至于要运行 **make** 即可构建 **dummy.so**.

```
/*
** Zabbix
** Copyright (C) 2001-2016 Zabbix SIA
```

```
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
**/

####include "sysinc.h"
####include "module.h"

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

/* module SHOULD define internal functions as static and use a naming pattern different from Zabbix intern
/* symbols (zbx_*) and loadable module API functions (zbx_module_*) to avoid conflicts
static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);

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

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_api_version                                           *
 *                                                                            *
 * Purpose: returns version number of the module interface                    *
 *                                                                            *
 * Return value: ZBX_MODULE_API_VERSION - version of module.h module is       *
 *               compiled with, in order to load module successfully Zabbix   *
 *               MUST be compiled with the same version of this header file   *
 *                                                                            *
 ******************************************************************************/
int zbx_module_api_version(void)
{
    return ZBX_MODULE_API_VERSION;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_item_timeout                                          *
 *                                                                            *
 * Purpose: set timeout value for processing of items                         *
 *                                                                            *
 * Parameters: timeout - timeout in seconds, 0 - no timeout set               *
 *                                                                            *
 ******************************************************************************/
void    zbx_module_item_timeout(int timeout)
```

```
{
    item_timeout = timeout;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_item_list                                             *
 *                                                                            *
 * Purpose: returns list of item keys supported by the module                 *
 *                                                                            *
 * Return value: list of item keys                                            *
 *                                                                            *
 ******************************************************************************/
ZBX_METRIC  *zbx_module_item_list(void)
{
    return keys;
}

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

    return SYSINFO_RET_OK;
}

static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char    *param;

    if (1 != request->nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param = get_rparam(request, 0);

    SET_STR_RESULT(result, strdup(param));

    return SYSINFO_RET_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: dummy_random                                                     *
 *                                                                            *
 * Purpose: a main entry point for processing of an item                      *
 *                                                                            *
 * Parameters: request - structure that contains item key and parameters      *
 *              request->key - item key without parameters                    *
 *              request->nparam - number of parameters                        *
 *              request->timeout - processing should not take longer than     *
 *                                 this number of seconds                     *
 *              request->params[N-1] - pointers to item key parameters        *
 *                                                                            *
 *             result - structure that will contain result                    *
 *                                                                            *
 * Return value: SYSINFO_RET_FAIL - function failed, item will be marked      *
 *                                  as not supported by zabbix                *
 *               SYSINFO_RET_OK - success                                     *
 *                                                                            *
```

```
 * Comment: get_rparam(request, N-1) can be used to get a pointer to the Nth  *
 *          parameter starting from 0 (first parameter). Make sure it exists  *
 *          by checking value of request->nparam.                             *
 *                                                                            *
 ******************************************************************************/
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char    *param1, *param2;
    int from, to;

    if (2 != request->nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param1 = get_rparam(request, 0);
    param2 = get_rparam(request, 1);

    /* there is no strict validation of parameters for simplicity sake */
    from = atoi(param1);
    to = atoi(param2);

    if (from > to)
    {
        SET_MSG_RESULT(result, strdup("Invalid range specified."));
        return SYSINFO_RET_FAIL;
    }

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_init                                                  *
 *                                                                            *
 * Purpose: the function is called on agent startup                           *
 *          It should be used to call any initialization routines             *
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - module initialization failed               *
 *                                                                            *
 * Comment: the module won't be loaded in case of ZBX_MODULE_FAIL             *
 *                                                                            *
 ******************************************************************************/
int zbx_module_init(void)
{
    /* initialization for dummy.random */
    srand(time(NULL));

    return ZBX_MODULE_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_uninit                                                *
 *                                                                            *
 * Purpose: the function is called on agent shutdown                          *
 *          It should be used to cleanup used resources if there are any      *
```

```
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - function failed                            *
 *                                                                            *
 ******************************************************************************/
int zbx_module_uninit(void)
{
    return ZBX_MODULE_OK;
}

/******************************************************************************
 *                                                                            *
 * Functions: dummy_history_float_cb                                          *
 *            dummy_history_integer_cb                                        *
 *            dummy_history_string_cb                                         *
 *            dummy_history_text_cb                                           *
 *            dummy_history_log_cb                                            *
 *                                                                            *
 * Purpose: callback functions for storing historical data of types float,    *
 *          integer, string, text and log respectively in external storage    *
 *                                                                            *
 * Parameters: history     - array of historical data                         *
 *             history_num - number of elements in history array              *
 *                                                                            *
 ******************************************************************************/
static void dummy_history_float_cb(const ZBX_HISTORY_FLOAT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_integer_cb(const ZBX_HISTORY_INTEGER *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_string_cb(const ZBX_HISTORY_STRING *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_text_cb(const ZBX_HISTORY_TEXT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
```

```
}

static void dummy_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_history_write_cbs                                     *
 *                                                                            *
 * Purpose: returns a set of module functions Zabbix will call to export      *
 *          different types of historical data                                *
 *                                                                            *
 * Return value: structure with callback function pointers (can be NULL if    *
 *               module is not interested in data of certain types)           *
 *                                                                            *
 ******************************************************************************/
ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void)
{
    static ZBX_HISTORY_WRITE_CBS    dummy_callbacks =
    {
        dummy_history_float_cb,
        dummy_history_integer_cb,
        dummy_history_string_cb,
        dummy_history_text_cb,
        dummy_history_log_cb,
    };

    return dummy_callbacks;
}
```

这个模块导出三个新的监控项类型：

- dummy.ping - 总是返回'1'
- dummy.echo[param1] - 总是返回第一个参数，例如 dummy.echo[ABC] 将返回" ABC "
- dummy.random[param1, param2] - 返回 param1 与 param2 范围内的随机数，例如, dummy.random[1,1000000]

限制

仅对类 Unix 平台实现了可加载模块的支持。这意味着它不适用于 Windows 平台的 Agent。

某些情况下，模块可能要从 *zabbix_agentd.conf* 读取与模块相关的配置参数。目前不支持这么操作。如果您需要模块使用某些配置参数，则应该实现特定与模块的配置文件解析。

6 Windows 性能计数器

概览

你可以使用 *perf_counter[]* 这个 key 有效的监控 Windows 性能计数器。

例如：

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

或

```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

有关使用此 key 的更多信息请参阅Windows 专用监控项。

为了获取可用于监控的新能计数器完整列表，你可以运行：

```
typeperf -qx
```

数字表示

由于性能计数器的命名在不同的 Windows 服务器上可能不同，这取决于服务器的地区设置。因此，在创建用于监控具有不同地区设置的多台 Windows 设备的模板时，会引发一定的问题。

同时，每个新能计数器也可以通过其数字形式来引用，无论如何，数字形式都是唯一的，因此你可以使用数字表示而不是字符串。

为了找到同义的数字，需要运行 **regedit** ，然后找到 *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009* 这个注册表。

注册表中包含形如下面所示的信息：

```
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
....
```

这样你就可以找到性能计数器每个字符串对应的数字，例如：

```
System → 2
% Processor Time → 6
```

然后你就可以使用这些数字来表示性能计数器路径：

```
\2\6
```

性能计数器参数

你可以部署一些 PerfCounter 参数，来完成通过 Windows 性能计数器监控。

例如，你可以将下面的内容添加到 ZABBIX 代理配置文件中：

```
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30
or
PerfCounter=UserPerfCounter2,"\4\24",30
```

配置了这些参数后，你就可以简单的使用 *UserPerfCounter1* 或 *UserPerfCounter2* 作为 key 来创建相应的监控项。

当然，别忘了在更改了配置文件后重新启动 ZABBIX Agent。

故障处理

有时 ZABBIX Agent 不能再基于 Windows 2000 的系统中检索性能计数器的值，因为 pdh.dll 文件已过时。这个错误会在 ZABBIX Agent 和 Server 的日志文件中会有失败信息。在这种情况下，phd.dll 应当被更新到更新的 5.0.2195.2668 版本。

7 批量更新

概览

有时你可能想要一次更改多个监控想的某些属性。你可以使用批量更新功能，而不是打开每个独立的监控项进行编辑。

使用批量更新

要批量更新某些监控项，请按如下步骤操作：

- 在监控项列表，标记想要更新的监控项的复选框
- 点击列表下方的 批量更新按钮
- 标记想要更新的属性的复选框
- 键入新的值，然后单击更新按钮

Item | Preprocessing

Type	☐	Original
Host interface	☐	Original
JMX endpoint	☐	Original
URL	☐	Original
Request body type	☐	Original
Timeout	☐	Original
Request body	☐	Original
Headers	☐	Original
Type of information	☐	Original
Units	☐	Original
Authentication method	☐	Original
User name	☐	Original
Public key file	☐	Original
Private key file	☐	Original
Password	☐	Original
Update interval	☐	Original
History storage period	☑	Do not keep history / Storage period / 7d
Trend storage period	☐	Original
Status	☐	Original
Log time format	☐	Original
Show value	☐	Original
Enable trapping	☐	Original
Allowed hosts	☐	Original
Applications	☑	Add / Replace / Remove; type here to search
Master item	☐	Original
Description	☐	Original

替换应用程序将从监控项中删除任何现有应用，并将其替换为此字段中指定的项目。

添加新的或者已经存在的应用程序允许为监控项从现有应用中指定其它应用或输入全新的应用。

这两个字段自动补全 - 在开始输入时即提供了匹配应用的下拉列表。如果应用程序是新的，它也会出现在下拉列表中，并在该字符串后面有 (new) 表示。只需向下滚动即可选择。

8 值映射

概览

为了接收到的值能更 “人性化” 的表示，你可以使用包含数值和字符串表示之间映射的值映射。

值映射也能在 ZABBIX 的前端和通过电子邮件/SMS/jabber 等发送的告警中被使用。

举个例子，一个监控项有值 ‘0’ 和 ‘1’ 能通过值映射，以认可读的形式表示值：

- ’0’ => ’ 不可用’
- ’1’ => ’ 可用’

或者, 一组备份关系的值映射可以是:

- ’F’ → ’ 全量备份’
- ’D’ → ’ 差异备份’
- ’I’ → ’ 增量备份’

在配置监控项时，你可以使用一组值映射来 “人性化” 的方式显示监控项的值。为此，定在查看值下拉菜单中选择事先定义的值映射方案的名称。

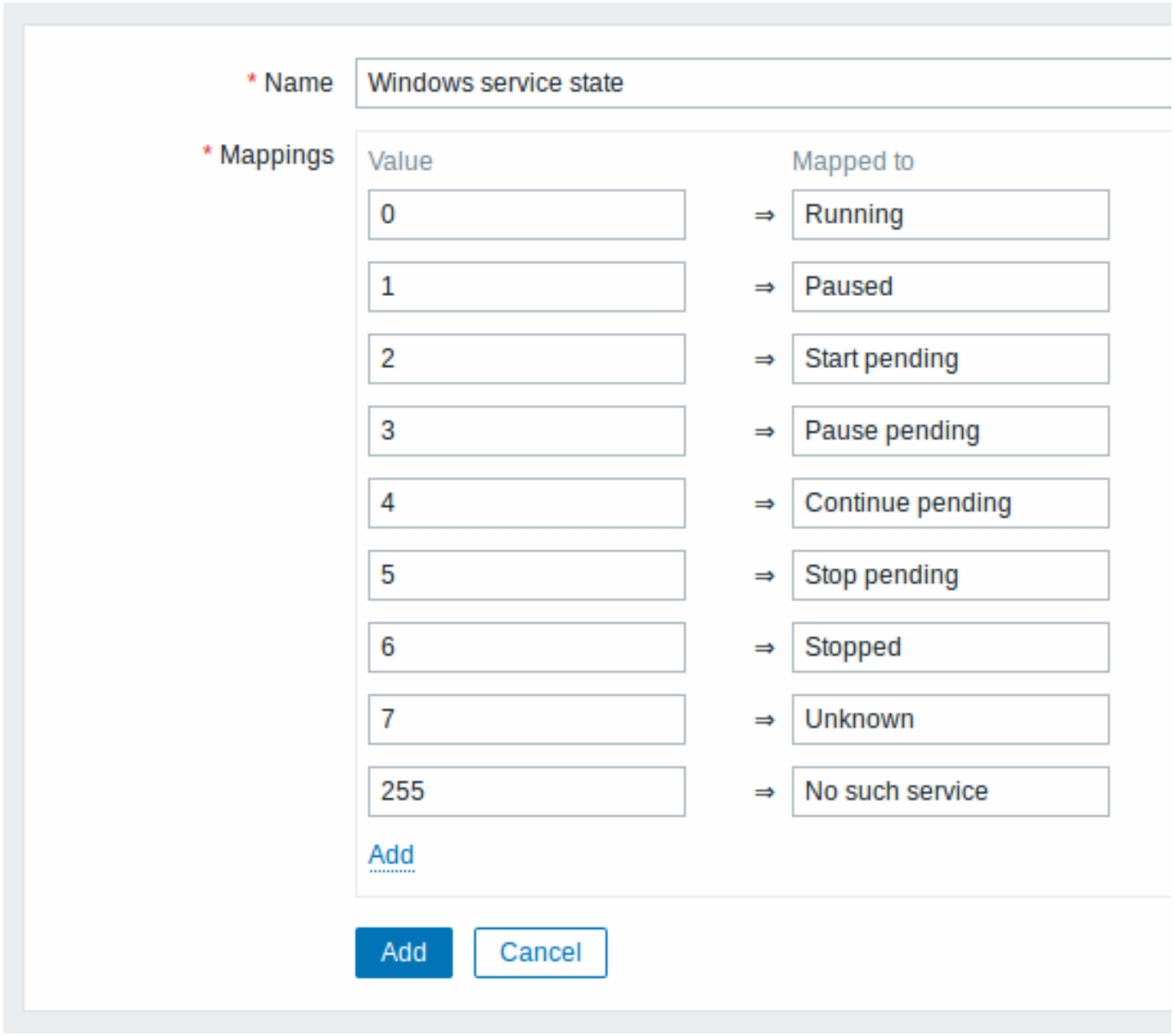
> **Note:**
> 值映射能被用来替换 数字（无符号），数字（浮点）和 字符类型的监控项信息

值映射在 ZABBIX3.0 版本起，可以被独立导出/导入，也可以与相应的模板或主机一同导出/导入。

Configuration 配置

要定义值映射：

- 前往: 管理 → 一般
- 从下拉列表中选择 值映射
- 点击创建值映射 (或点击一个现有值映射的名称上)



值映射的参数：

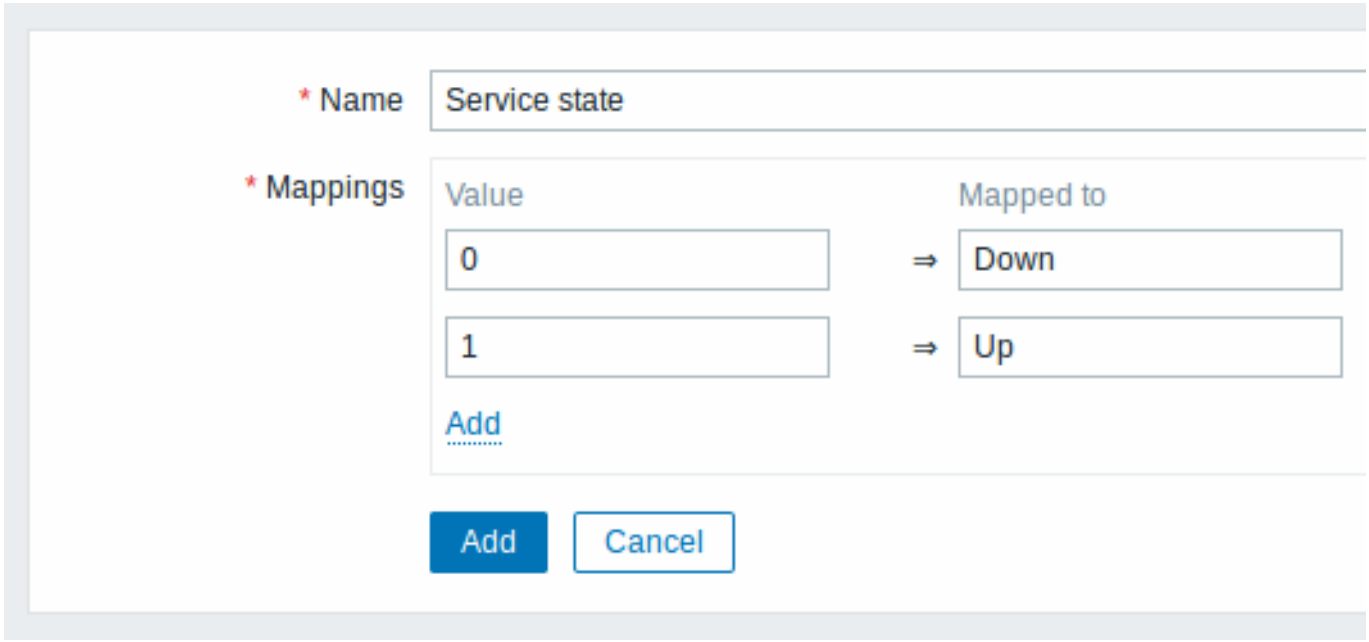
参数	描
名称	值映射的名称，应当时唯一的
映射单	映射 - 一对值与字符串表示.

所有标星号的字段都需要填入。

要添加一个新的映射对，请按添加。

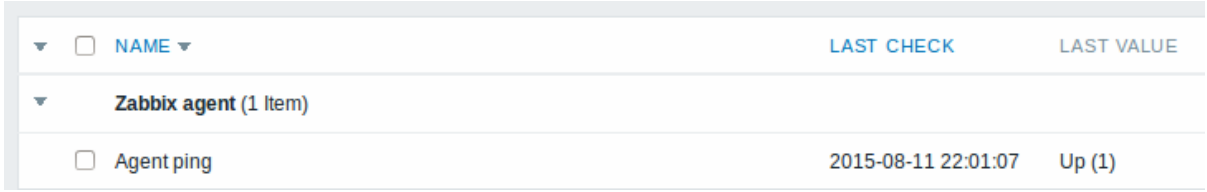
值映射如何工作的

举个例子，有一个预定义的 Agent 监控项‘Ping to the server (TCP)’ 使用了一个已经存在的值映射名字叫‘Service state’ ，来显示其值。



在监控项的配置页面，你可以从显示值字段看到对此值映射的引用。

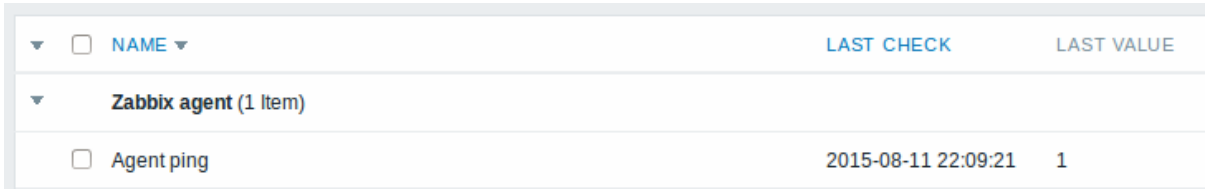
这样配置以后，在监控中 → 最新数据会以映射的值 “Up” 显示（括号中显示的时原始值）。



在最新数据部分中，显示的值会算短为 20 个符号，如果使用值映射，则此缩短规则不会应用于映射值，而是仅应用于原始值（显示在括号中）。

<note tip> 当接受通知时，以人类可读的形式显示值，也更容易理解。

如果没有预定义的值映射，你只能看到：



这样的情况下，要么猜测 “1” 是什么意思，要么去搜索文档以找到答案。

9 应用

概览

应用，是一种用于把监控项分组的逻辑组。

举个例子：这里有一个叫 *MySQL* 服务器应用，它关联了 MySQL 服务器相关的所有监控项：MySQL 的可用性、磁盘空间、处理器负载、每秒事务数、慢查询数等。

应用也用于给 Web 场景分组。

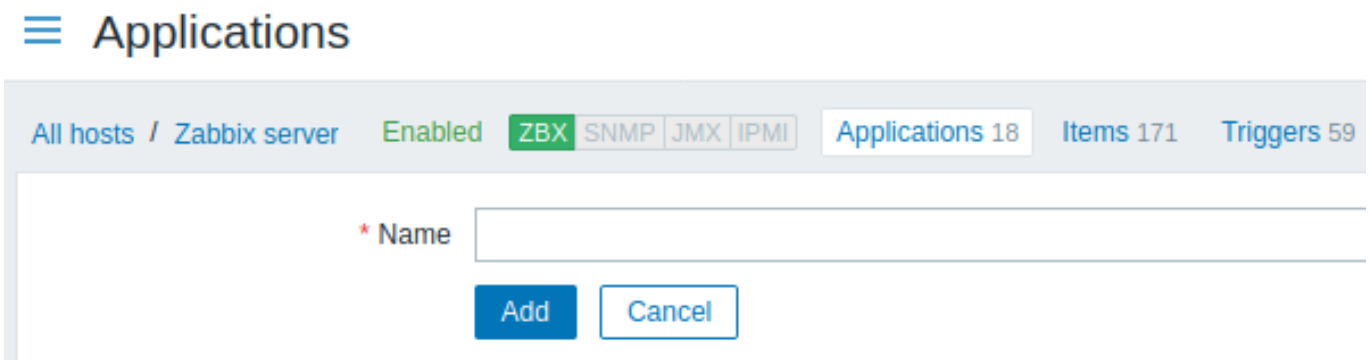
如果你正在使用应用，那么在监控中-最新数据中，你将看到按各应用分组下的监控项和 Web 场景。

配置

要使用应用，你必须先创建它们，然后将监控项或 Web 场景链接到它们。

要创建应用，请执行下述操作：

- 前往配置 → 主机或 模板
- 点击应用集然后通过右上角的菜单跳转到指定主机或模板
- 点击 创建应用集
- 键入应用的名称，然后单击 添加以将修改保存。



你也可以直接创建一个新应用，在监控项属性表单页面中。

监控项被监控项属性表单页面中的应用。可以为监控项选择属于一个或多个应用。

Web 场景连接到 Web 场景定义表单页面中选择的应用，在这个页面可以选择 web 场景所属的应用。

10 队列

概览

对列显示正在等待刷新的监控项。队列只是数据的一种逻辑上表现。ZABBIX 中并没偶 IPC 队列或者其它任何队列的机制。

由 Proxy 们监控的监控项也会被包含在列中 - 这些监控项将按 Proxy 历史数据更新周期被计数为队列

只有具有刷新时间计划的监控项才会记录在队列中。这表示，队列中将不包含以下的监控项类型：

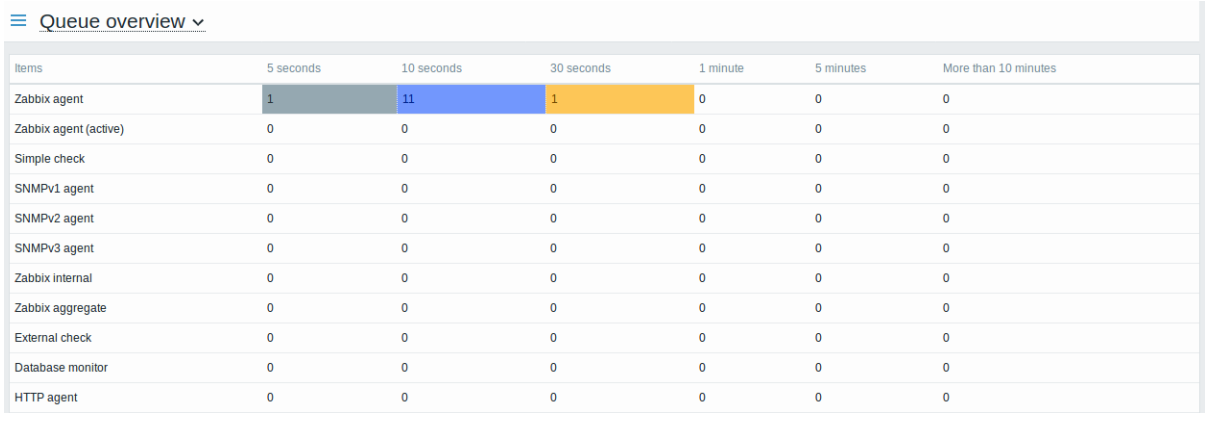
- log, logrt and eventlog 相关的 ZABBIX Agent（主动）监控项
- SNMP trap 类型监控项
- trapper 类型监控项
- web 场景监控的监控项

队列显示的统计信息是 ZABBIX Server 是否健康的指标。

使用 JSON 协议直接从 ZABBIX Server 检索队列。这个页面的信息只在 ZABBIX Server 运行时可用。

阅读队列

要查看队列，请跳转管理 → 队列。在右侧的下达菜单中选择概览。



Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

如图所示，大片的绿色，这样我们可以假设服务器运行时正常的。

队列里有一个监控项等待 5 秒，有 5 个等待 30 秒。知道这些项目具体是什么会更好。

马上帮你实现，在右上角的下拉菜单中选择细节。现在既可以看到这些延迟监控项的列表。

Queue details

Scheduled check	Delayed by	Host	Name	Proxy
2019-09-02 11:46:40	58s	My host	CPU idle time	Remote proxy
2019-09-02 11:46:41	57s	My host	CPU interrupt time	Remote proxy
2019-09-02 11:46:42	56s	My host	CPU iowait time	Remote proxy
2019-09-02 11:46:43	55s	My host	CPU nice time	Remote proxy
2019-09-02 11:46:44	54s	My host	CPU softirq time	Remote proxy
2019-09-02 11:46:45	53s	My host	CPU steal time	Remote proxy
2019-09-02 11:46:46	52s	My host	CPU system time	Remote proxy

通过这些细节信息，可以找出这些监控项发生延迟的原因。

有一两个延迟监控项，不要慌张。它们有可能在一秒内被更新。但是如果你看到了一大堆延迟很久的监控项，这可能导致严重的问题。

ITEMS	5 SECONDS	10 SECONDS	30 SECONDS	1 MINUTE	5 MINUTES	MORE THAN 10 MINUTES
Zabbix agent	0	13	7	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	5	1	9	0	0	0

是不是 Agent 进程下线了？

队列项目

有一个特别的内部监控项 **zabbix[queue,<from>,<to>]** 可以用于监控 ZABBIX 中队列的健康状态。他会返回指定时间区间的监控项数目。有关更多信息请参阅内部监控项。

11 值缓存

Overview 概览

为了更快地计算触发器表达式、计算或聚合类型监控项和一些宏。自 ZABBIX 2.2 起，ZABBIX Server 支持值缓存选项。

这个存放在内存中的缓存，可以用于访问历史数据，而不需要对数据库直接执行 SQL 调用。如果缓存中不存在请求得历史值，则会从数据库请求缺失的数据，并相应地更新缓存。

要启用值缓存功能，Zabbix 服务器配置文件支持可选的 **ValueCacheSize** 参数。

有两个内部的监控项来监控值缓存：**zabbix [vcache，buffer，<mode>]** 和 **zabbix [vcache，cache，<parameter>]**。查看更多细节，请参阅 [[zh:manual:config:items:itemtypes:internal| 内部监控项]。

12 立刻检查

概览

在 ZABBIX 中检查一个新监控项的值会基于已配置的更行间隔循环过程。虽然对于大多数监控项来说间隔非常短，但是还有些其它监控项(包括低级自动发现规则)，更新间隔会很长。因此在显示情况下，可能需要更快的检查新的值。-例如，立刻获取可发现资源的变化。为了满足这种必要性，可以重新安排被动检查并立刻检索新的值。

这个功能仅支持被动检查。支持以下监控项的类型：

- Zabbix agent (被动)
- SNMPv1/v2/v3 agent
- IPMI agent
- 简单检查
- Zabbix 内部
- Zabbix 聚合
- 外部检查
- 数据库监控
- JMX 代理
- SSH 代理
- Telnet
- 计算
- HTTP 代理

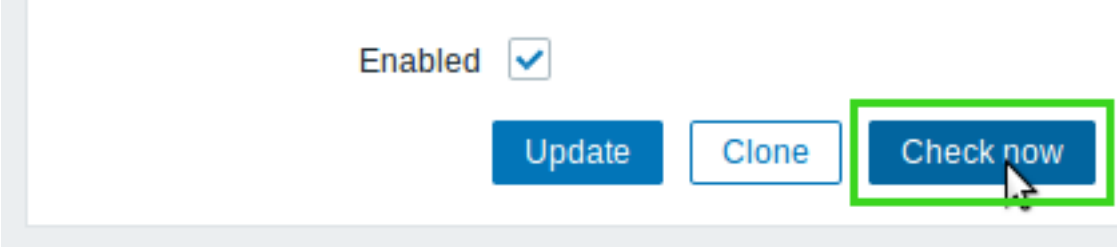
<note important> 检查必须存在于配置缓存中才能执行。有关详细信息请参阅缓存更新频率。
在执行检查前，若配置缓存没有更新，那么将不会检索最近更改配置的监控项/自动发现规则。同样也无法检查刚刚创建的监控项/规则的最新值。

:::

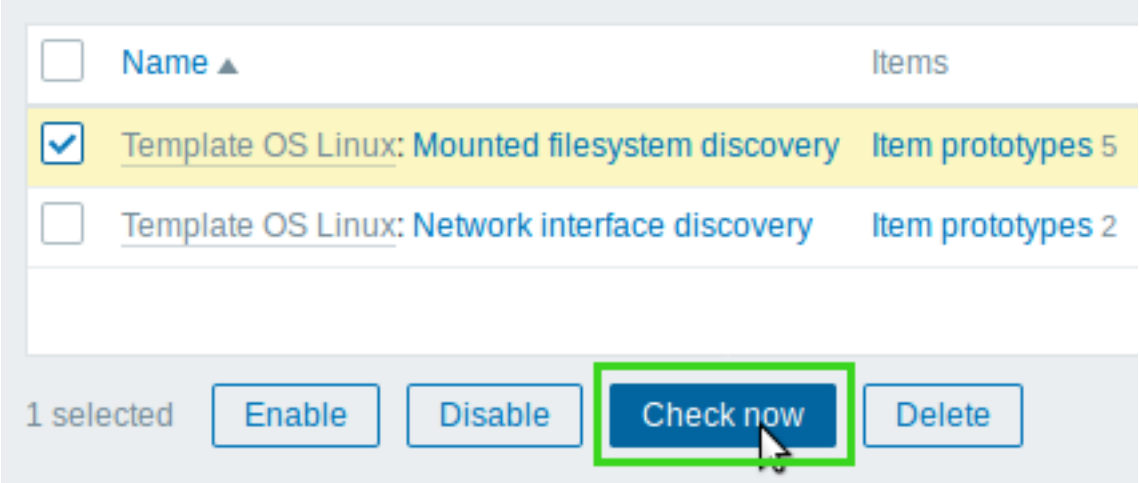
配置

要立刻执行被动检查：

- 在已存在的监控项（或自动发现规则）配置表单中点击 *Check now*:



- 在监控项（或发现规则列表）中，选定监控项/规则后，单击 *Check now*:



在第二种情况下，可以选择几个项目/规则，一次性对它们进行 “立刻检查”。

3 触发器

概述

触发器是 “评估” 由监控项采集的数据并表示当前系统状况的逻辑表达式。

当监控项用于采集系统的数据时，始终遵循这些数据是非常不切合实际的，因为这些数据始终在等待一个令人担忧或者值得关注的状态。然而这个 “评估” 数据的工作可以留给触发器表达式。

触发器表达式允许定义一个什么状况的数据是“可接受”的阈值。因此，如果接收的数据超过了可接受的状态，则触发器会被触发 - 或将状态更改为异常。

一个触发器可以拥有下面几种状态：

值	述
OK	这是一个正常的触发器状态。在旧版本的 Zabbix 中称为 FALSE。
PROBLEM	通常意味着发生了某些事情。例如，处理器的负载较高。在旧版本的 Zabbix 中称为 TRUE。

每当 Zabbix server 接收到作为表达式一部分的新值时，都会重新计算触发器状态（表达式）。

如果在表达式中使用基于时间的函数 (nodata(), date(), dayofmonth(), dayofweek(), time(), now())，触发器就会由 Zabbix history syncer 进程每 30 秒重新计算一次。如果在表达式中同时使用基于时间和非基于时间的函数，当接收到一个新值和每隔 30 秒都会重新计算触发器的状态。

你可以构建触不同复杂程度的触发器表达式

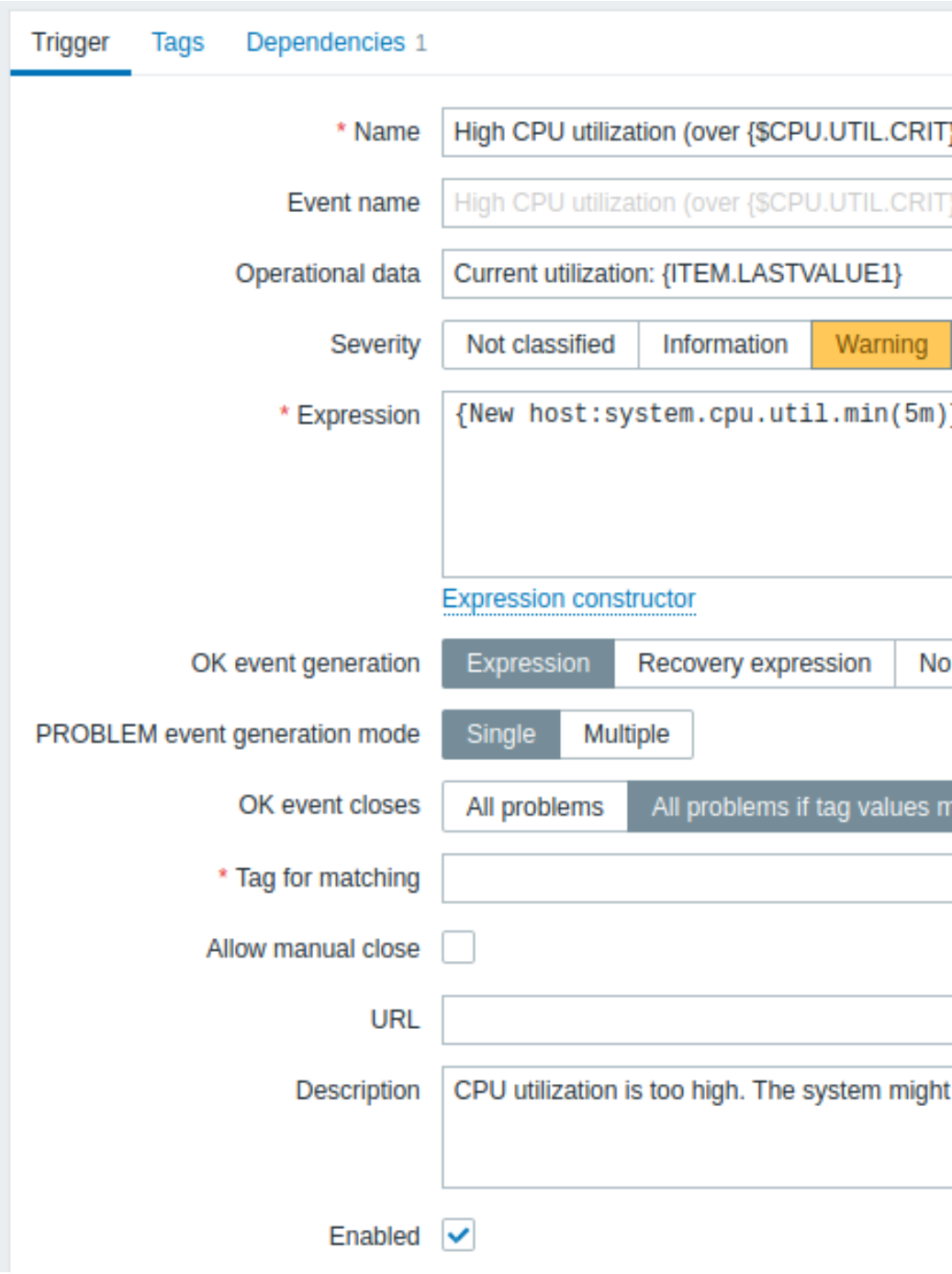
1 配置一个触发器

概述

配置一个触发器，进行下面步骤：

- 进入：配置 → 主机
- 点击主机一行的 触发器
- 点击右上角的 创建触发器（或者点击触发器名称去修改一个已存在的触发器）
- 在窗口中输入触发器的参数

配置



触发器标签页包含了所有必要的触发器属性。

所有强制输入字段都用红色星号标记。

参数	描
名称	触器名称. 名称中可以包含支持的宏: {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {ITEM.VALUE}, {ITEM.LASTVAL 和 {$MACRO}。 **$1, $2...$9** 宏可以用来指第一, 第二... 第九表达式的常量。 备注: 如果引用了相对简单的常量或明确的表达式, $1-$9

参数	描
严重性通过	击对应的按钮来设置所需的触发器严重性。
异常表达式用于定义	常条件的逻辑表达式。

参数	描
事件成功迭代事件成功迭	选项: 表达式 - OK 事件基于与问题事件相同的表达式生成; 恢复表达式 - 如果问题表达式计算为 false，恢复表达式计算为 true，则生成 OK 事件; **None** - 在这种情况下，触发器将

参数	描
恢复表达式逻辑 [表	式](expression) 用于定义问题解决的条件。只有在表达式表达式计算为 FALSE 之后才对恢复表达式进行评估。如果问题条件仍然存在，则不可能通过恢复表达式来解决问题。此

参数	描
异常事件生成模式	生成异常事件的式：- 单个：当触发器第一次进入‘异常‘状态时，生成一条单个事件。；- 多重：每一个触发器“异常”评估都将生产一条事件。

参数	描
事件成功关闭如果选择事	成功关闭: 所有问题 - 此触发器的所有问题 所有问题如果标签值匹配 - 只有那些匹配事件标签值引发的问题。 从 Zabbix 3.2.0 开始支持。

参数	描
匹配标记输入事	标记名称以用于事件关联。如果在事件成功关闭中选择了‘所有问题如果标签值匹配’，在这种情况下是强制性的。从 Zabbix 3.2.0 开始支持。

参数	描
标记设	自定义标签来标记触发器事件。事件标记可用于事件关联，在动作条件下，也将在监视→问题中被看到。标记是一对标记名和值。您可以仅使用名称或值对。\\在事

参数	描
允许手动关闭	检查是否允许手动关闭由该触发器生成的问题事件。在确认问题事件时，手动关闭是可能的。从 Zabbix 3.2.0 开始支持。

参数	描
URL	如果不为空，在监测→问题和问题仪表板中点击触发器名称，这里输入的URL可以作为链接。(*URL*选项在触发器上下文菜单)可以在触发器URL字段中使用宏 - {TRIGGER.ID}，多

参数描	
描述文	字段用于提供有关此触发器的更多信息。可能包含解决具体问题的指令、负责人员的联系细节等。从*Zabbix 2.2*开始，描述可以包含与触发器名称相同的宏集。

参数描	
已启用如果	要，不选中该框将禁用触发器。

依赖关系标签里包含所有触发器依赖关系。

点击 添加添加新的依赖关系。

Note:
你也可以打开一个已存在的触发器，点击 克隆按钮，然后用一个不同的名字保存。

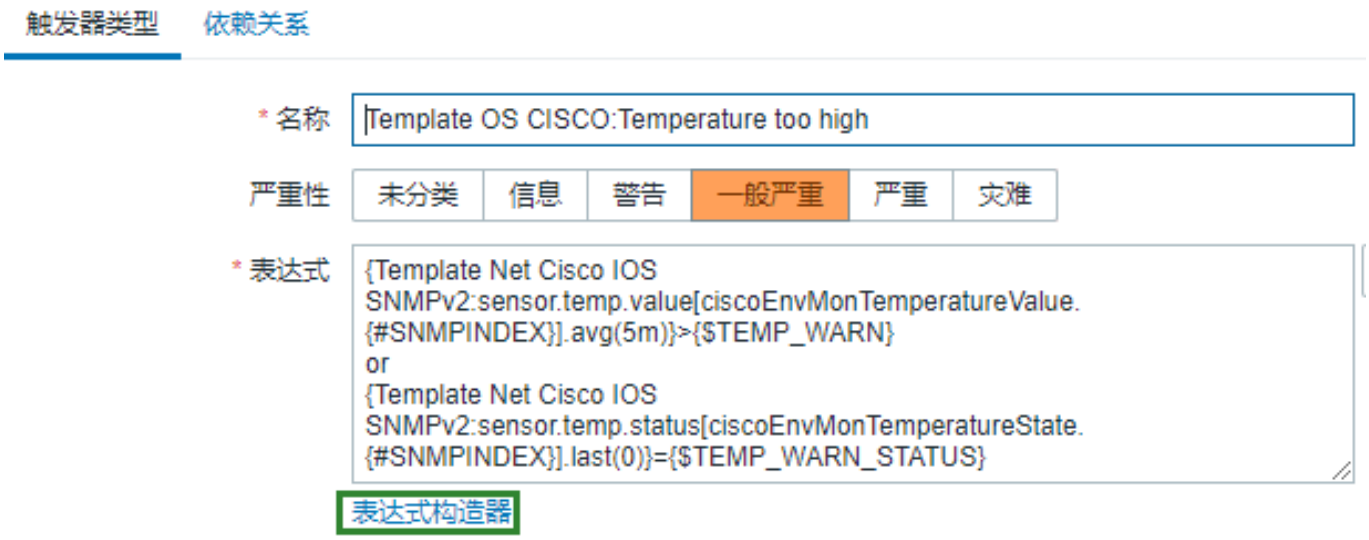
测试表达式

可以根据接收的值测试配置的触发器表达式，以确定表达式结果。

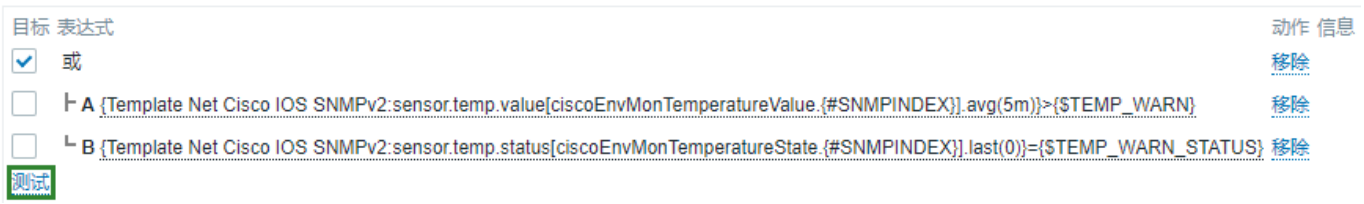
以官方模板的表达为例：

```
{Template Net Cisco IOS SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}].avg(5m)}>{$TEMP
or
{Template Net Cisco IOS SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}].last(0)}={$TEM
```

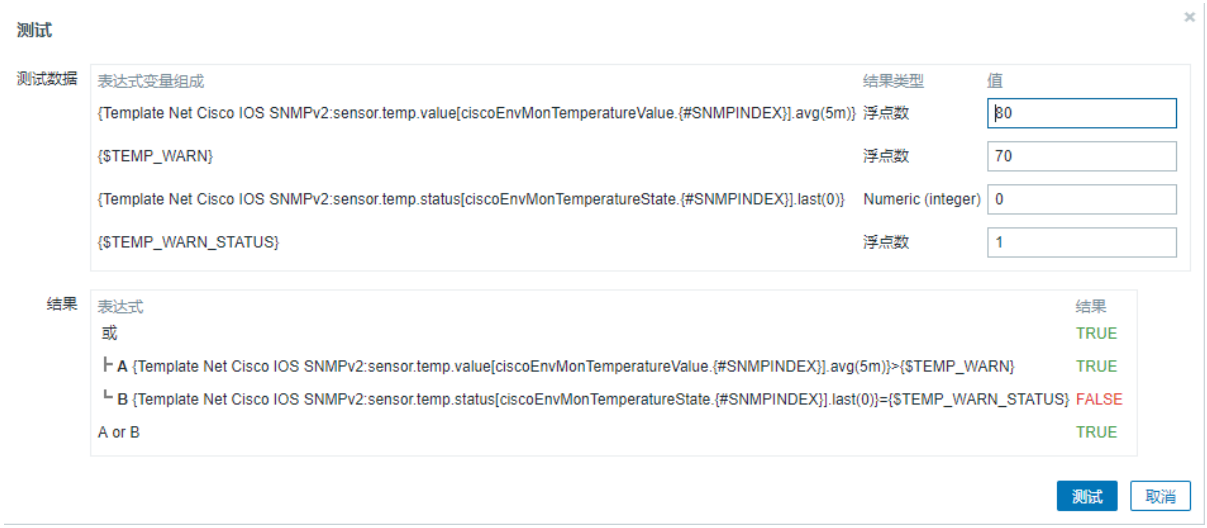
若要测试表达式，请点击表达式字段下的 表达式构造器。



在表达式构造器列出了所有单个表达式。打开测试窗口，点击在表达式列表下方测试。



在测试窗口中，您可以输入示例值（在这个示例中为 “80, 70, 0，1”），然后点击 测试按钮查看表达式结果。



可以看到每个表达的结果以及整个表达的结果。

“TRUE” 结果意味着指定的表达式是正确的。在这个特定的情况 A 下，“80” 大于 {$TEMP_WARN} 指定值 “70”，出现 “TRUE” 结果。

“FALSE” 结果表示指定的表达式不正确。在这个特定的情况 B 下，在这个例子中 {$TEMP_WARN_STATUS} 是 “1”，需要与指定的 “0” 值相等，这是错误的。出现 “FALSE” 结果。

选择的表达式类型是 “OR”/“true”。如果指定条件中的至少一个（在这种情况下为 A 或 B）是真的，那么最终结果也是 TRUE。意味着，当前值超过了警告值，出现了异常。

2 触发器表达式

概述

触发器中使用的表达式是非常灵活的。你可以使用它们去创建关于监控统计的复杂逻辑测试。

一个简单有效的表达式看起来像：:

```
{<server>:<key>.<function>(<parameter>)}<operator><constant>
```

函数

触发器函数允许引用采集的值，当前时间和其他因素。

可以使用的支持函数完整列表。

函数参数

大多数数字型的函数接受秒数来作为参数。

你可以使用前缀 **#** 来指定参数具有不同的含义：

函数调用含义	
sum(600)	600 秒内所有值的总和
sum(#5)	最后 5 个值的总和

函数 last 当以 # 作为前缀使用时具有不同的含义 - 它可以选择第 N 次前的值，返回值 3, 7, 2, 6, 5 (最近五次)，**last(#2)** 将返回值为 7 ，**last(#5)** 将返回值为 5。

一些函数支持额外的第二个参数时间偏移量。这个参数允许从过去一段时间内引用数据。例如，**avg(1h,1d)** 将会返回一天前 1 小时的平均值。

你可以在触发器表达式中使用支持的单位符号，例如 ‘5m’（分钟）代替 ‘300’ 秒，‘1d’（天）代替 ‘86400’ 秒。‘1k’ 代表 ‘1024’bytes。

运算符

触发器支持下列运算符（在执行中优先级递减）

优先级运算	定义	**[未知值]	/manual/config/triggers/expression#expressions_with_unsupported_items_and_unknown_values) 注释
1	**-**	负	*-**Unknown → Unknown

优先级运算	定义	**[未知值]	/manual/config/triggers/expression#expressions_with_unsupported_items_and_unknown_values) 注释
2	**not**	逻辑非 **	ot** Unknown → Unknown
3	*	乘	* Unknown → Unknown (yes, Unknown, not 0 - to not lose Unknown in arithmetic operations) 1.2 * Unknown → Unknown
	/	除	nknown / 0 → error Unknown / 1.2 → Unknown 0.0 / Unknown → Unknown
4	+	加	.2 + Unknown → Unknown
	-	减	.2 - Unknown → Unknown
5	<	小于。该运算符定义： 1.2 **&lt A<B ⇔ (A<B-0.000001)	** Unknown → Unknown
	<=	小于等于。该运算符定义： Unknown **& A<=B ⇔ (A≤B+0.000001)	t;=** Unknown → Unknown
	>	大于. 该运算符定义： A>B ⇔ (A>B+0.000001)	
	>=	大于等于。该运算符定义： A>=B ⇔ (A≥B-0.000001)	
6	=	相等。该运算符定义： A=B ⇔ (A≥B-0.000001) and (A≤B+0.000001)	
	<>	不等于。该运算符定义： A<>B ⇔ (A<B-0.000001) or (A>B+0.000001)	
7	**and**	逻辑与 0	*and **Unknown → 0 1** and **Unknown → Unknown Unknown** and** Unknown → Unknown
8	**or**	逻辑或 1	*or **Unknown → 1 0** or **Unknown → Unknown Unknown** or** Unknown → Unknown

not, **and** and **or** 运算符区分大小写，而且必须为小写。它们也必须被空格或括号包围。

所有运算符中, 除了 **-** 和 **not** , 都有左到右的关联性。**-** 和 **not** 是非结合的 (意味着**-(-1)** 和 **not (not 1)** 应该用**--1** and **not not 1** 代替).

计算结果：

- **<**, **<=**, **>**, **>=**, **=**, **<>** 如果指定的关系为真，运算符将会在触发器表达式中产生 '1'。如果指定的关系为假，则返回 '0'。如果至少有一个运算数未知，则结果未知；
- **and** 对于已知的运算对象，如果两个运算对象的比较不等于 "0"，则运算符将会在触发器表达式中产生 "1"，否则，它产生 "0"；对于未知的运算对象，如果两个运算对象的比较等于 "0"，则会产生 "0"，否则，则会产生 "Unknown"；
- **or** 对于已知的运算对象，如果其中任意一个运算对象的比较不等于 "0"，则运算符会在触发器表达式中产生 "1"，否则，它产生 "0"；对于未知的运算对象进行 "or" 运算，则只有当一个运算对象的比较不等于 "0"，才会产生 "1"，否则，它会产生 "Unknown"；
- 如果操作数的值不等于 "0"，则已知操作数的逻辑否定运算符 **not** 的结果是 "0"；如果操作数的值等于 "0"，则为 "1"。对于未知的操作数 **not** 产生 "Unknown"。

缓存值

触发器评估所需的值由 Zabbix server 缓存。由于此触发器评估在服务器重新启动后一段时间导致较高的数据库负载。当监控项历史数据被移除（手动或 housekeeper）时，缓存值不会被清除，因此服务器将使用缓存的值，直到它们比触发器函数中定义的时间段或服务器重启的时间长。

触发器示例

示例 1

www.zabbix.com 的处理器负载过高

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5
```

'www.zabbix.com:system.cpu.load[all,avg1]' 给出了被监控参数的简短名称。它指定了服务器是 "www.zabbix.com"，监控项的键值是 "system.cpu.load[all,avg1]"。通过使用函数 "last()" 获取最新的值。最后，">5" 意味着当 www.zabbix.com 最新获取的处理器负载值大于 5 时触发器就会处于异常状态。

示例 2

www.zabbix.com is overloaded

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5 or {www.zabbix.com:system.cpu.load[all,avg1].min(10m)}
```

当前处理器负载大于 5 或者最近 10 分钟内最小值大于 2，表达式为 true。

示例 3

/etc/passwd 文件被修改

使用函数 diff：

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff()}=1
```

当文件/etc/passwd 的 checksum 值与最近的值不同时，表达式为 true。

类似的，表达式可以用于监控重要文件的修改, 如/etc/passwd, /etc/inetd.conf, /kernel 等

示例 4

有人正在从互联网上下载一个大文件

使用 min 函数:

```
{www.zabbix.com:net.if.in[eth0,bytes].min(5m)}>100K
```

在过去 5 分钟内，eth0 上接收字节数大于 100kb 时，表达式为 true。

示例 5

SMTP 服务群集的两个节点都停止。注意在一个表达式中使用两个不同的主机:

```
{smtp1.zabbix.com:net.tcp.service[smtp].last()}=0 and {smtp2.zabbix.com:net.tcp.service[smtp].last()}=0
```

当 SMTP 服务器 smtp1.zabbix.com 和 smtp2.zabbix.com 都停止，表达式为 true

示例 6

Zabbix agent 需要升级

使用 str() 函数:

```
{zabbix.zabbix.com:agent.version.str("beta8")}=1
```

如果 Zabbix agent 版本是 beta8（可能是 1.0beta8），则表达式为真。

示例 7

服务器无法访问

```
{zabbix.zabbix.com:icmpping.count(30m,0)}>5
```

当主机 “zabbix.zabbix.com” 在 30 分钟内超过 5 次不可达，则表达式为真。

示例 8

3 分钟内没有心跳检查

使用 nodata() 函数:

```
{zabbix.zabbix.com:tick.nodata(3m)}=1
```

要使用这个触发器，’tick’ 必须定义成一个 Zabbix[:manual/config/items/itemtypes/trapper|trapper]] 监控项。主机应该使用 zabbix_sender 定期发送这个监控项的数据。

如果在 180 秒内没有接收到数据，则触发值变为异常状态。

注释 ‘nodata’ 可以在任何类型的监控项中使用。

示例 9

夜间的 CPU 负载

使用 time() 函数:

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2 and {zabbix:system.cpu.load[all,avg1].time()}>000000 and {zab
```

仅在夜间 (00:00-06:00)，触发器状态变可以变为真。

示例 10

检查客户端本地时间是否与 Zabbix 服务器时间同步

使用 fuzzytime() 函数:

```
{MySQL_DB:system.localtime.fuzzytime(10)}=0
```

当 MySQL_DB 服务器的本地时间与 Zabbix server 之间的时间相差超过 10 秒，触发器将变为异常状态。

示例 11

比较今天的平均负载和昨天同一时间的平均负载（使用第二个 “时间偏移” 参数）。

```
{server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,1d)}>2
```

如果最近一小时平均负载超过昨天相同小时负载的 2 倍，触发器将触发。

示例 12

使用了另一个监控项的值来获得触发器的阈值：

```
{Template PfSense:hrStorageFree[{#SNMPVALUE}].last()}<{Template PfSense:hrStorageSize[{#SNMPVALUE}].last()
```

如果剩余存储量下降到 10% 以下，触发器将触发。

示例 13

使用评估结果获取超过阈值的触发器数量:

```
({server1:system.cpu.load[all,avg1].last()}>5) + ({server2:system.cpu.load[all,avg1].last()}>5) + ({server
```

如果表达式中至少有两个触发器大于 5，触发器将触发。

滞后

有时我们需要一个 OK 和问题状态之间的区间，而不是一个简单的阈值。例如，我们希望定义一个触发器，当机房温度超过 20C 时，触发器会出现异常，我们希望它保持在那种状态，直到温度下降到 15C 以下。

为了做到这一点，我们首先定义问题事件的触发器表达式。然后在事件成功迭代中选择 ‘恢复表达式’，并为 OK 事件输入恢复表达式。

请注意，只有首先解决问题事件才会评估恢复表达式。如果问题条件仍然存在，则不能通过恢复表达式来解决问题。

示例 1

机房温度过高。

问题表达式:

`{server:temp.last()}>20`

恢复表达式:

`{server:temp.last()}<=15`

示例 2

磁盘剩余空间过低。

问题表达式: it is less than 10GB for last 5 minutes

`{server:vfs.fs.size[/,free].max(5m)}<10G`

恢复表达式: it is more than 40GB for last 10 minutes

`{server:vfs.fs.size[/,free].min(10m)}>40G`

不支持项的表达式和未知的值

Zabbix3.2 之前的版本对触发器表达式中不支持的监控项非常严格。表达式中的任何不支持的监控项都会立即将触发器值呈现为 “未知”。

从 Zabbix3.2 开始通过将未知值引入到表达式评估中，对不受支持的项有更灵活的方法：

- 对于某些函数，它们的值不受监控项是否支持的影响。这样的函数即使它们引用不支持的项，也会对它们进行评估。请参阅函数和不支持的监控项清单。
- Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
  - ”1 or 不支持的监控项函数 1 or 不支持的监控项函数 2 or ...” 可以被评估为’1’ (True)，
  - ”0 and 不支持的监控项函数 1 and 不支持的监控项函数 2 and ...” 可以被评估为’0’ (False)，
    Zabbix 试图评估不支持的项目作为 `Unknown` 值的逻辑表达式。在上述两种情况下，将产生一个已知值；在其他情况下，触发值将是 `Unknown`。
- 如果对受支持的监控项的一个函数评估结果为错误，那么这个函数的值为 Unknown ，并且它将参与进一步的表达式评估。

如上所述，未知值可以在逻辑表达式中 “消失”。在算数表达式中未知值总会导致结果为 “Unknown”（除以 0 除外）。

如果具有多个不支持的监控项的触发器表达式评估为 “Unknown”，前端的错误消息是指最后一个不支持的监控项。

3 触发器依赖关系

概述

有时候一台主机的可用性依赖于另一台主机。如果一台路由器宕机，则路由器后端的服务器将变得不可用。如果这两者都设置了触发器，你可能会收到关于两个主机宕机的通知，然而只有路由器是真正故障的。

这就是主机之间某些依赖关系可能有用的地方，设置依赖关系的通知可能会被抑制，而只发送根本问题的通知。

虽然 Zabbix 不支持主机之间的直接依赖关系，但是它们可以定义另外一种更加灵活的方式 - 触发器依赖关系。一个触发器可以有一个或多个依赖的触发器。

因此在我们简单示例中，我们打开服务器触发器配置的窗口，并设置它依赖于路由器的相应触发器。有了这样的依赖性，只要它所依赖的触发器处于 “异常” 状态，服务器触发器就不会改变状态，因此不会执行依赖的动作，也不会发送通知。

如果服务器和路由器都宕机且有依赖关系，Zabbix 将不执行依赖触发器的动作。

依赖触发器上的动作不会被执行，如果触发器依赖于:

- 状态从’PROBLEM’ 修改为’UNKNOWN’
- 通过关联或者基于时间功能的手工关闭
- 被非依赖触发器的监控项值恢复
- 已禁用，已禁用监控项或禁用项目主机

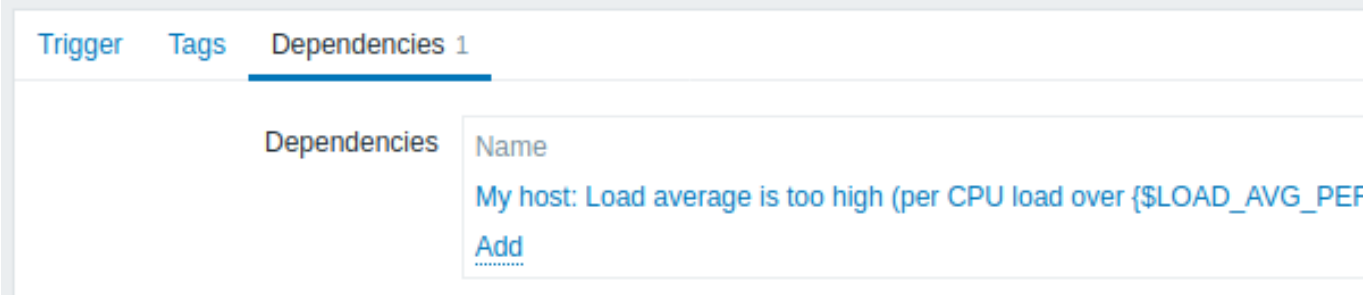
请注意，上述情况下的 “次要”（依赖）触发器不会立即更新。

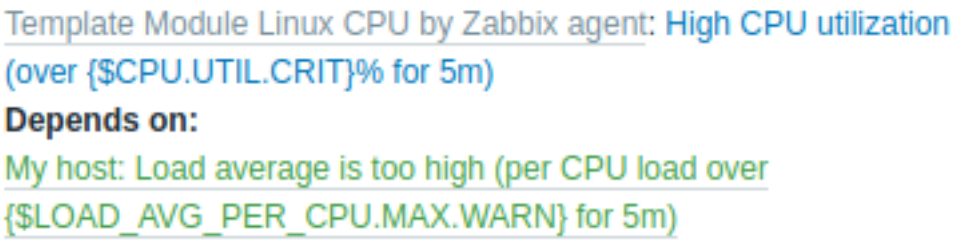
另外：

- 触发器依赖可以从任何主机触发器添加到任何其他主机触发器，只要它不会导致循环依赖。
- 触发器依赖可以从一个模板添加到另一个模板，如果模板 A 的触发器依赖于模板 B 的触发器，模板 A 只能与模板 B 一起链接到主机（或其他模板），但是模板 B 可以单独链接到主机（或其他模板）。
- 触发器依赖可以从模板触发器添加到主机触发器。在这种情况下，例如, 有一个触发器依赖于路由器（主机）触发器的模板。链接到这个模板的所有主机都将依赖于特定的路由器。
- 可以不添加从主机触发器到模板触发器的触发器依赖性。
- 触发器依赖可以从一个触发器原型添加到另一个触发器原型 (在同一个 Low-level discovery 规则中) 或真实触发器中。触发器原型可以不依赖来自不同 LLD 规则的触发器原型或者触发器原型中创建的一个触发器。主机触发器原型不能依赖于模板中的触发器。

配置

若要定义依赖关系，在触发器配置表格打开依赖关系标签。单击 “依赖关系” 块中的 添加，并选择触发器将依赖的一个或多个触发器。





点击 更新，现在列表中触发器有了依赖性标示。

几个依赖关系的示例

例如，主机位于路由器 2 后面，路由器 2 在路由器 1 后面。

```
Zabbix - 路由器1 - 路由器2 - 主机
```

如果路由器 1 宕机，显然主机和路由器 2 也不可达，然而我们不想收到主机、路由器 1 和路由器 2 都宕机的 3 条通知。

因此，在这种情况下我们定义了两个依赖关系：

```
'主机宕机' 触发器依赖于 '路由器2宕机' 触发器
'路由器2宕机' 触发器依赖于 '路由器1宕机' 触发器
```

在改变 “主机宕机” 触发器的状态之前，Zabbix 将会检查相应触发器的依赖关系，如果找到，并且一个触发器处于 “异常” 状态，则触发器状态不会发生改变，因此不会执行动作，也不会发送通知。

Zabbix 递归执行此检查，如果路由器 1 或路由器 2 是不可达的状态，那么主机触发器则不会更新。

4 触发器严重性

触发器严重性定义了触发器的重要程度，Zabbix 支持下列触发器的严重程度：

严重性定义	颜色
未分类未知	重性灰色
信息提	浅蓝色
警告警	黄色
一般严重一般问	橙色
严重发	重要的事情浅红色
灾难灾	，财务损失等红色

触发器严重性用于:

- 触发器的直观表示，不同的颜色代表不同的严重程度。
- 全局报警音频。不同的音频代表不同的严重程度。
- 用户媒介，不同的用户媒介（通知渠道）代表不同的严重程度。例如，SMS - 高严重性，email - 其他。
- 通过触发器严重程度的条件来限制动作。

可以自定义触发器严重性的名称和颜色。

5 自定义触发器严重性

可以在管理 → 一般 → 触发器严重性中配置触发器严重性名称和严重性颜色相关的 GUI 主题。颜色在所有 GUI 主题之间共享。

翻译自定义严重性的名称

Attention:
如果使用 Zabbix 前端翻译, 自定义严重性名称将会覆盖默认翻译名称。

默认触发器严重性名称适用于所有语言环境的翻译。如果更改了严重性名称，则会在所有的语言环境中使用自定义名称，因此需要额外的手动翻译。

自定义严重性名称的翻译步骤：

- 设置自定义严重性名称, 例如 '重要'
- 编辑 <frontend_dir>/locale/<required_locale>/LC_MESSAGES/frontend.po
- 添加如下 2 行:

```
msgid "重要"
msgstr "<翻译字符串>"
```

保存文件。

- 在 <frontend_dir>/locale/README 创建.mo 文件作为描述

这里 **msgid** 应该匹配新的自定义严重性名称，**msgstr** 应该是用特定语言翻译的。

此过程应在每个严重性名称更改之后执行。

7 批量更新

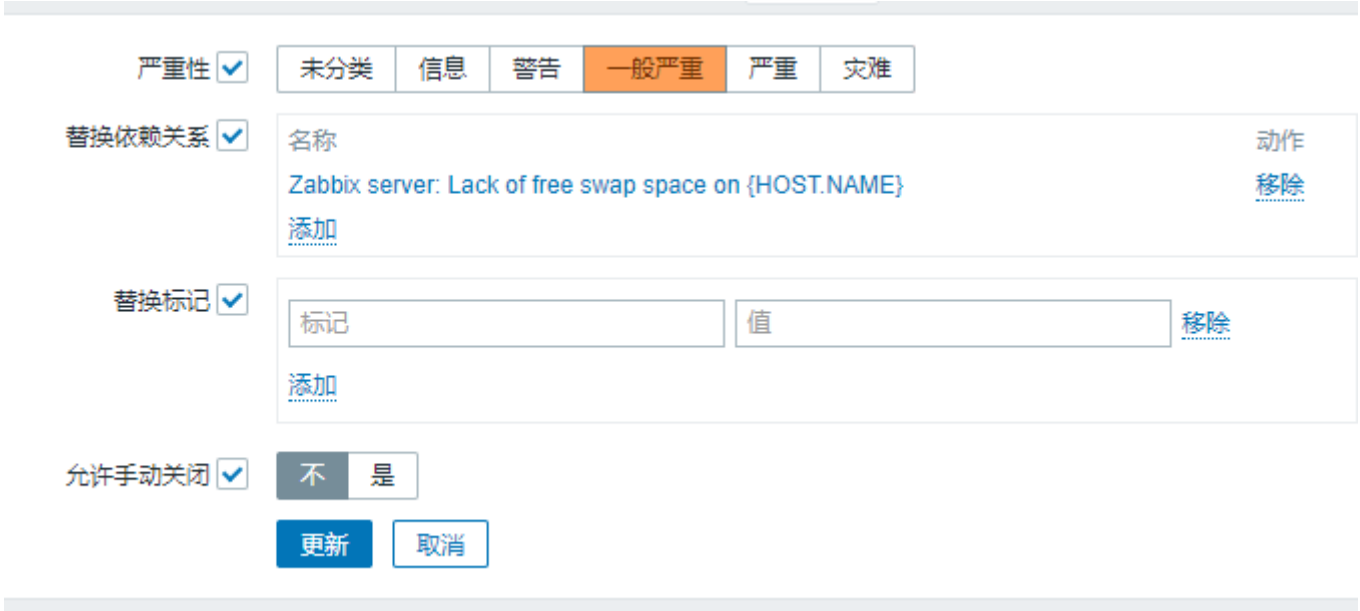
概述

使用批量更新，你可以同时更改一些触发器的属性，从而节省了打开每个触发器进行编辑的需要。

使用批量更新

要批量更新某些触发器，请执行以下步骤：

- 在清单中选中需要更新的触发器复选框
- 点击清单下的批量更新按钮
- 标记要更新的属性的复选框
- 为属性指定新值，并点击的更新



在一次指定的批量更新中替换依赖关系和替换标记将替换现有的触发器依赖关系/标签（如果有的话）。

8 预测触发功能

概述

有时候有即将到来问题的迹象。可以发现这些迹象，以便提前采取行动，以防止或至少最小化问题的影响。

Zabbix 具有基于历史数据预测受监视系统的未来行为的工具。这些工具通过预测触发功能实现。

1 功能

需要知道的两件事是如何定义问题状态以及需要多少时间来采取行动。有两种方法可以设置一个关于潜在的不必要的情况的触发信号。第一：触发器必须在系统发生 “时间作用” 之后才会发生故障状态。第二：当系统在不到 “时间行为” 的时候达到问题状态时，触发器必须触发。使用相应的触发器功能是 **forecast** 和 **timeleft **。请注意，两个功能的基本统计分析基本相同。您可以设置触发器，以您喜欢的方式，以类似的结果。

2 参数

这两个功能使用几乎相同的参数集。列表请参见supported functions 支持的功能。

2.1 时间间隔

首先，你应该指定 Zabbix 应该分析的历史时期来进行预测。你可以通过 “秒” 或 “#num” 参数和可选的 “time_shift” 以熟悉的方式进行操作，就像使用 **avg**，**count**，**delta**，**max**，**min** 和 **sum** 功能。

2.2 预测范围

(**forecast** only)
参数 time 指定了将来 Zabbix 应该在多大程度上推断其在历史数据中找到的依赖关系。无论是否使用 “time_shift”，“时间” 始终从当前时刻算起。

2.3 阈值

(**timeleft** only)
参数 “阈值” 指定分析的项目必须达到的值，如果从上或下都没有差异。一旦我们确定了 f(t)（见下文），我们就要解方程 f(t) = “阈值”，如果没有这样的根，返回更靠近现在和向右的根或 999999999999.9999。

Note:
当监控项值接近阈值并超过它时, **timeleft** 假定交叉点已经过去，因此切换到下一个 “阈值” 级别的交叉点（如果有的话）。最佳实践应该是使用预测作为普通问题诊断的补充，而不是替代。[a]

[a] 例如，一个简单的触发器 `{host:item.timeleft(1h,,X)} < 1h` 当监控项值接近 X 时可能进入异常状态，然后一旦达到值 X 就突然恢复。如果异常是监控项值低于 X，请使用：`{host:item.last()} < X or {host:item.timeleft(1h,,X)} < 1h` 如果异常是项目值高于 X，请使用：`{host:item.last()} > X or {host:item.timeleft(1h,,X)} < 1h`

2.4 Fit 函数

默认 `fit` 是线性函数。但是如果你的监控系统更复杂，你有可以有更多的选择。

`fit`	x = f(t)
线性 x	= a + b*t
多项式[3] x	$a_0 + a_1*t + a_2*t^2 + ... + a_n*t^n$
指数 x	= a*exp(b*t)
对数 x	= a + b*log(t)
幂	$= a*t^b$

2.5 模式

(**forecast** only)
每次触发功能被评估时，它都会从指定的历史时段获得数据并将指定的函数拟合到数据。因此，如果数据略有不同，拟合函数将略有不同。对于某些 “fit” 选项（如多项式），未来的简单值可能会产生误导。

模式 *	预测 ** 结果
值	(now + time)
最大 m	$x_{now <= t <= now + time}$ f(t)
最小 m	$n_{now <= t <= now + time}$ f(t)
增量 *	ax* - *min*
平均 f	t) 的平均值 (now <= t <= now + time) 参考定义

3 细节

为了避免大量的计算，我们考虑在指定周期内的第一个值的时间戳加上 1ns 作为一个新的零时间。(当前时间时期是 10^9, 时期平方是 10^{18}, 双精度约为 10^{-16})。添加 1 ns 以提供对数和幂拟合的所有正时间值，其涉及计算 log（t)。时间偏移不影响线性、多项式、指数（除了更容易和更精确的计算），但改变对数和幂函数的状态。

[3] 多项式度可以是从 1 到 6, 多项式 *1* 等于线性。然而，谨慎使用更高阶多项式with caution。如果评估周期包含比确定多项式系数所需的更少的点数，则多项式度将降低。(例如请求多项式 *5*，但只有 4 点，因此多项式 *3* 更合适)。

4 潜在错误

函数如下情况下返回- 1：

- 指定的评估期不包含数据；
- 数学运算结果未定义[4]；
- 数值问题 (不幸的是，对于一些输入数据范围和双精度浮点格式的精度变得不足)[5].

<note tip> 如果选择合适不好描述提供的数据或只有太少的数据用于精确预测，就不会有警告或错误被标记。:::

5 示例和错误处理

要在主机上的可用磁盘空间用完时收到警告，可以使用如下触发器表达式：

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h
```

然而，错误代码-1 可能会发挥作用，并将您的触发器置于异常状态。一般来说，这是很好的，因为你收到一个警告，你的预测不能正常工作，你应该更深入地了解它们，找出原因。但有时它是坏的，因为-1 可以简单地意味着没有关于最后一小时内获得的主机可用磁盘空间的数据。如果您收到太多错误警报，则应考虑使用更复杂的触发器表达式[6]：

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1
```

形势比预测有点困难。首先，-1 可能会也可能不会将触发器置于问题状态，具体取决于您是否具有表达式：

或者像

```
{host:item.forecast(...)}>...
```

此外，如果项目值为负值，-1 可能是有效的预测。但这种情况实际发生的可能性很小，(参见运算符 =如何 工作)。因此添加

```
... and {host:item.forecast(...)}<>-1
```

如果你想或不想把 1 作为一个问题来对待。

参阅

1. Predictive trigger functions (pdf) on zabbix.org

4 事件

概述

在 Zabbix 中可以生成以下几种类型的事件：

- trigger events - 触发器事件，当触发器改变他的状态时（*OK→PROBLEM→OK*）；
- discovery events - 发现事件，当主机或服务被检测到；
- auto registration events - 自动注册事件，当主动的 agents 被自动注册到 server 时；
- internal events - 内部事件，当监控项 item/低级别自动发现规则 low-level discovery rule 变得不受支持或触发器进入了一个未知状态。

> **Note:**
> 从 Zabbix2.2 版本开始支持内部事件。

事件是以时间戳的，并可以作为发送电子邮件等动作的基础。

要查看前端事件的详细信息，点击 *Monitoring → Problems*。那里你可以点击事件的日期和时间来查看事件的详细信息。

关于更多的可供参考信息，请查看：

- trigger events
- other event sources

[4] 比如将指数或者幂函数计入 log() 监控项值. 如果数据包含零或负数，您将收到错误，因为 log() 仅限于正值。

[5] 对于线性，指数，对数和幂适合所有必要的计算都可以明确地写出来。对于多项式，只有在没有任何附加步骤的情况下才能计算出值。计算 *avg* 涉及计算多项式反导数（解析）。计算最大，最小和增量涉及计算多项式导数（解析），并找到其根源（数字）。求解 f(t) = 0 涉及求多项式根（数值）。

[6] 但是在这种情况下，1 可能导致触发器从问题状态恢复。充分保护使用: `{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and ({TRIGGER.VALUE}=0 and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1 or {TRIGGER.VALUE}=1)`

1 触发器事件生成

概述

触发器状态的变化是事件最常见和最重要的来源。每次触发器的状态改变时，都会生成一个事件。该事件包含了触发器状态变更的详细信息、发生时间以及触发器的新状态。

触发器会创建两种类型的事件：问题（Problem）和正常（OK）。

问题事件

在以下情况下，一个问题事件（Problem event）将被创建：

- 当触发器状态为正常（OK）时，触发器表达式的计算结果为 TRUE。
- 如果为触发器启用了多重问题事件生成，那么每次触发器表达式计算结果为 TRUE。

正常事件

一个正常事件（OK event）关闭关联的问题事件（Problem event），可由以下三个部分创建：

- 触发器 - 基于 “正常事件迭代（OK event generation）” 和 “正常事件关闭（OK event closes）” 的设置；
- 关联项事件；
- 任务管理器 - 当事件被manual/config/events/手动关闭。

触发器

触发器有 “事件成功迭代（OK event generation）” 的设置，用来控制如何生成正常事件（OK event）：

- 表达式 - 当触表达式的计算结果为 FALSE 的时候，触发器在问题（Problem）状态中生成一个正常事件（OK event）。这是一个最简单的设置，为默认启动。
- 恢复表达式 - 当表达式的计算结果为 FALSE，并且恢复表达式的计算结果为 TURE 的时候，会为问题（Problem）状态的触发器生成一个正常事件（OK event）。如果触发器的恢复条件和问题标准不同，则可以使用此设置。
- 无 - 正常事件从来不生成。这个可以和多重问题事件生成一起结合使用，以便在某事件发生时可以更简单的发送通知。

此外，触发器有 “事件成功关闭（OK event closes）” 的设置，用来控制哪些问题事件（Problem events）被关闭：

- 所有问题 - 正常事件（OK event）将关闭触发器创建的所有打开的问题；
- 所有问题如果标记的值匹配- 正常事件（OK event）将关闭触发器创建的打开的问题，并且至少有一个匹配的标记值。标记由 “匹配” 触发器设置标记定义。如果没有问题事件（Problem event）关闭，那么正常事件（OK event）将不会生成。这通常被称为触发级事件关联。

事件关联

事件关联（也被称为全局事件关联）是一种设置自定义事件关闭（导致正常事件生成）的规则。

这个规则定义了新的问题事件如何于现有的问题事件配对，并通过生成相应的正常事件来关闭新的事件或匹配事件。

但是，必须仔细地配置事件关联，因为它可能会对事件处理性能造成负面影响，或者如果配置不当，则会关闭比预期更多的事件（在最坏的情况下可能会关闭所有的问题事件）。以下是几个关于配置的小提示：

1. 通过为控制事件（与旧事件配对的事件）设置唯一的标签来减小事件关联的范围，并使用 “新的事件标记（new event tag）” 来关联条件；
2. 不要忘记在使用 “过去的事件标记” 操作时添加基于过去事件的条件，否则可能会关闭所有现有的问题；
3. 避免在使用不同关联配置时使用通用的标记名称。

任务管理器

如果允许在触发器中启用 “允许手动关闭”，那么可以手动关闭触发器生成的问题事件。这在manual/acknowledges# 更新问题的界面中完成。这个事件并不是直接关闭，而是创建一个 “关闭事件” 的任务，任务管理器很快会处理它。任务管理器将会生成一个相应的正常事件，并且问题事件将会关闭。

2 手动关闭问题事件

概述

当触发器的状态从 “问题（Problem）” 变成 “正常（OK）” 时，问题事件通常会自动解决，但是有一些情况很难判断一个问题是否是通过触发器表达式的方式解决的。在这种情况下，就需要手动解决问题。

例如，*syslog* 可能会报告一些内核参数需要调整以获得最佳性能。在这种情况下，问题报告给 Linux 管理员，它们会修复它，然后手动关闭此问题。

只有在触发器选项中启用允许手动关闭选项，问题事件才可以被手动关闭。

当一个问题事件是 “手动关闭” 时，Zabbix 会为 Zabbix Server 生成了一个新的内部任务，然后任务管理器进程执行这个任务，并生成正常事件，以关闭问题事件。

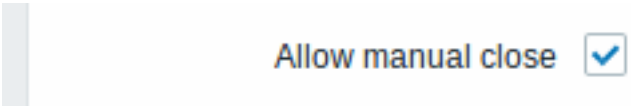
手动关闭问题事件并不意味着底层的触发器将永远不会再次进入 “问题” 状态。当触发器表达式中包含的任何监控项有新数据达到时，将重新计算整个表达式，并可能会再次生成问题。

配置

需要两步来手动关闭问题事件。

触发器配置

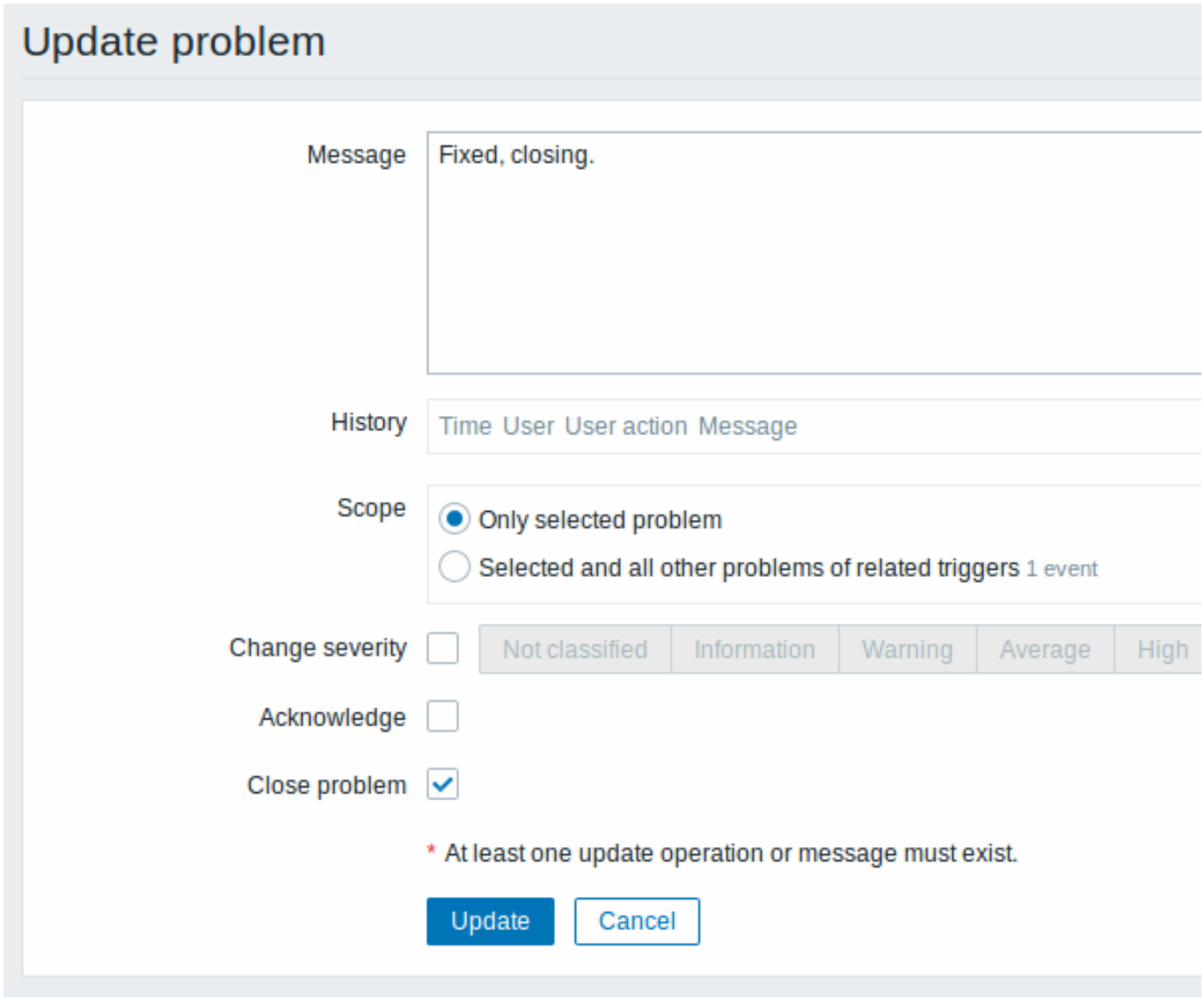
在触发器的配置页面上，启用 允许手动关闭选项。



问题更新页面

如果已启用允许手动关闭的触发器出现问题，你可以进入该触发器的 “确认事件” 页面，并手动关闭该问题。

要关闭这个问题，可以在确认事件页面查看关闭问题选项，并点击更新。



所有必须输入的区域都用红色星号进行了标记。

请求通过 Zabbix server 处理。常需要几秒才能关闭问题。在此期间，该问题在前端页面的监测中 → 问题显示的状态为关闭中。

验证

下面的方式可以验证该问题是否被手动关闭：

- 通过监测中 → 问题页面查看事件的详细信息；
- 通过在提供此信息的通知消息中使用宏 {EVENT.UPDATE.HISTORY} 来验证。

3 其他事件来源

发现事件

Zabbix 定期扫描网络发现规则中定义的 IP 范围。可以为每个规则单独配置检查频率。一旦发现主机或服务，就会生成一个发现事件（或多个事件）。

Zabbix 可以生成以下事件：

事件描	
Service Up	每当 Zabbix 检测到活跃的服务。
Service Down	每当 Zabbix 无法检测到服务。
Host Up	如果一个 IP 至少有一个活跃的服务。
Host Down	如果所有的服务都没有响应。
Service Discovered	如果服务在维护时间之后恢复或者第一次被发现。
Service Lost	如果服务在运行后丢失。
Host Discovered	如果主机在维护时间滞后恢复或者第一次被发现。
Host Lost	如果主机在运行后丢失。

主动式客户端自动发现事件

主动式客户端自动注册会在 Zabbix 创建事件。

如果配置了自动注册，当以前未知的主动式客户端向服务器发起检测请求或者主机的元数据被改变，服务器会生成主动注册事件。服务器使用主动式客户端请求的 IP 地址和端口，添加一个新的自动注册主机。

关于自动注册更多的信息，请查阅active agent auto-registration 页面。

内部事件

在下面的情况下，会发生内部事件：

- 监控项的状态从 “正常” 变为 “不支持的”；
- 监控项的状态从 “不支持的” 变为 “正常”；
- 低级别自动发现规则的状态从 “正常” 变为 “不支持的”；
- 低级别自动发现规则的状态从 “不支持的” 变为 “正常”；
- 触发器的状态从 “正常” 变为 “未知的”；
- 触发器的状态从 “未知的” 变为 “正常”。

从 Zabbix2.2 开始支持内部事件。引入内部事件的目的是允许在发生任何内部事件时通知用户，例如，一个监控项的状态变为不支持的，并停止采集数据。

5 事件关联

概述

事件关联允许以一种非常精确和灵活的方式关联问题事件和他们的解决方法。

事件关联可以定义为：

- 触发器级别 - 一个触发器可能被用于关联不同的问题和他们的解决方法
- 全局的 - 问题可以使用全局关联规则通过不同触发器和轮询方法与他们的解决方法进行关联。

1 基于触发器的时间关联

概述

基于触发器的事件关联允许关联一个触发器产生的不同问题。

通常，在 Zabbix 中正常事件会关闭一个触发器生成的所有问题事件，但在某些情况下需要更加细致的方法。例如，当监控日志文件时，在日志文件中想要发现某些问题，并将它们单独关闭，而不是一起关闭。

当触发器配置页面的多重问题时间生成选项为启用的情况下，通常适用于日志监控、被动采集（trap）处理等。

换言之，相同的触发器可以创建由事件标签标识的的不同事件。因此，可以一个一个单独地标识问题事件，并基于事件标签地标识单独关闭。

工作原理

在日志监控中，可能会遇到下面类似地输出：

```
Line1: 应用1停止

Line2: 应用2停止

Line3: 应用1重启

Line4: 应用2重启
```

事件关联地想法是将从 “Line1” 的问题事件到 “Line3” 的恢复事件，从 “Line2” 的问题事件到 “Line4” 的恢复事件相匹配，并能逐个关闭这些问题：

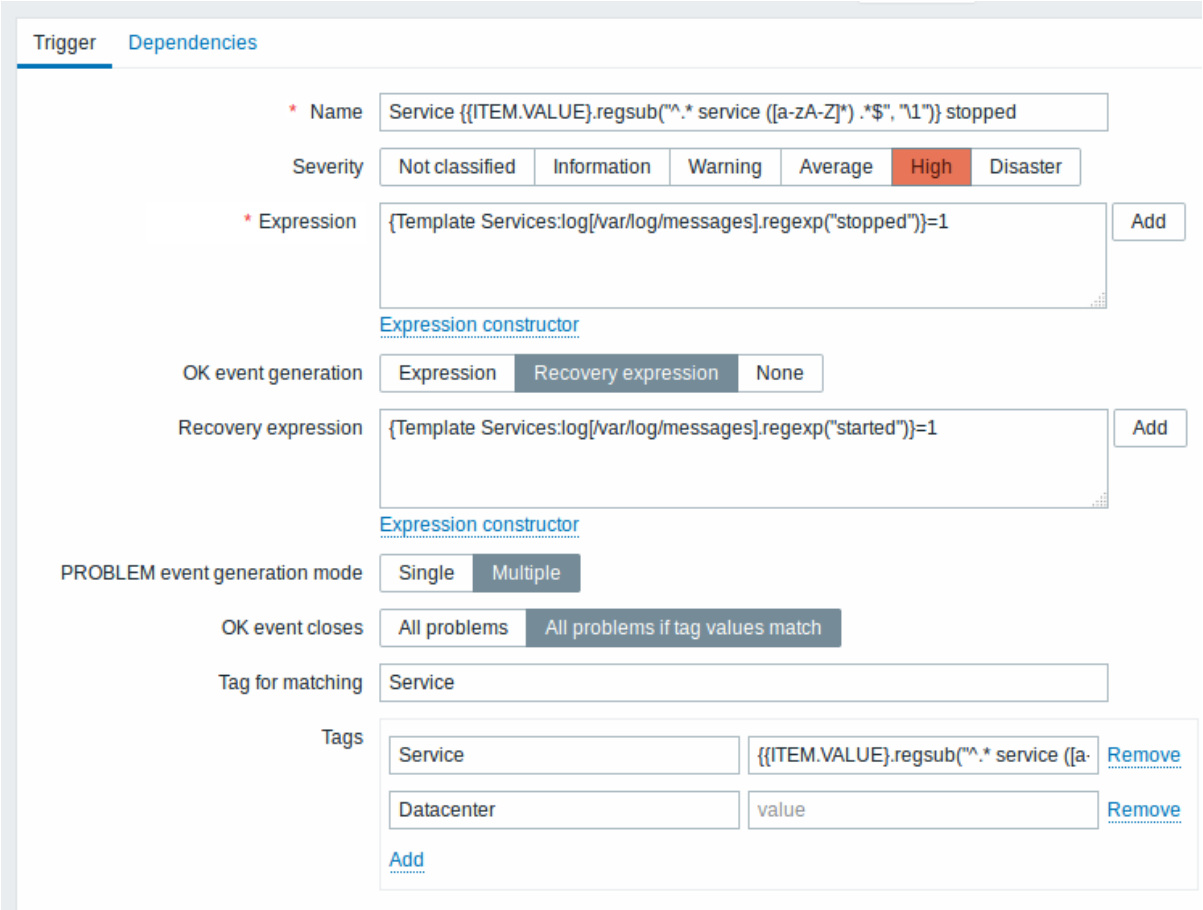
```
Line1: 应用1停止

Line3: 应用1重启#问题来自于Line1关闭

Line2: 应用2停止

Line4: 应用2重启#问题来自于Line2关闭
```

为此，需要通过标签将这些事件相关联，例如，可以标识为 “Application 1” 和 “Application 2”。这个过程也可以将正则表达式应用于日志中来提取标签的值。然后，当事件创建时，他们分别给标识为 “Application 1” 和 “Application 2”，并且问题可以与解决方法相匹配

配置

在触发器的配置界面配置事件关联：

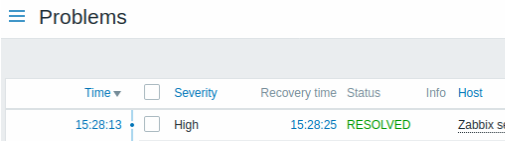
- 转到触发器的配置项；



所有必须输入的区域都通过红色星号进行标记。

- 选择 “问题事件生成模式” 的多重选项；
- 选择 “正常事件关闭” 的如果标签匹配的所有问题；

- 输入事件匹配的标签名称；
- 从日志中提取标签的值以配置标签



如果配置成功，能够看到标记 “application ” 的问题事件，并与监测中 → 问题页面看到结果相匹配

<note warning> 因为有可能出现错误配置，当为不相关的问题创建相似的事件标签时，请查阅下面标记出来的情况：:::

- 当由两个 applications 向相同的日志文件写入故障和恢复信息，用户通过在标签中使用单独的正则表达式来提取标签的名称。例如 “application A” 和来自宏 {ITEM.VALUE} 的 “application B”（当消息格式不同时），然而，如果和正则表达式不匹配的话，可能会无法按照计划工作。不匹配的正则表达式将生成空的标签值，并且在问题和正常事件中的单个空标签值足以关联它们。因此，来自 “application A” 的恢复消息可能会意外地关闭来自 “application B” 地错误消息。
- 实际上标签和标签的值只有在触发器触发时才会显示。如果所使用的正则表达式无效的话，则会使用默认的字段 “UNKNOWN” 进行替换。如果错过了标签值 “UNKNOWN” 的初始问题事件，那么可能会出现与标签值 “UNKNOWN” 的后续正常事件，并有可能导致关闭不应该关闭的问题事件。
- 如果用户使用没有宏功能的宏 {ITEM.VALUE} 作为标签值，则会有 255 个字符串的限制。当日志消息很长，并且前面 255 个字符串是不明确的话，就有可能导致类似的事件标签用于不相关的问题上。

2 全局事件关联

概述

全局事件关联允许覆盖 Zabbix 监控的所有指标并创建关联性。

可以关联由完全不同的触发器创建的事件，并对它们应用相同的操作。通过创建智能关联规则，实际上可以避免数以千计的重复通知，并专注于问题的根本原因！

全局事件关联是一种强大的机制，它可以让您从基于单个触发的问题和解决逻辑中解开自己。到目前为止，单个问题事件是由一个触发器创建的，我们依赖于相同的问题解决触发器。我们无法用另一个触发器解决一个触发器创建的问题。但是基于事件标记的事件关联，我们可以。

例如，日志触发器可以报告应用程序问题，而轮询触发器可以报告应用程序启动并运行。利用事件标记，您可以将日志触发器标记为状态：*Down*，而将轮询触发器标记为状态：*Up*。然后，在全局关联规则中，您可以关联这些触发器并为此关联分配适当的操作，例如关闭旧事件。

在另一种用途中，全局关联可以识别类似的触发器并对它们应用相同的操作。如果我们每个网络端口问题只能获得一个问题报告怎么办？无需全部报告。通过全局事件关联也是能够实现的。

全局事件关联在关联规则中配置。关联规则定义新问题事件如何与现有问题事件配对以及在匹配情况下要执行的操作（关闭新事件，通过生成相应的 OK 事件来关闭匹配的旧事件）。如果问题被全局关联关闭，则会在监控 - > 问题的消息列中报告。

配置全局关联规则仅适用于 Zabbix 超级管理员级别用户。

<note important> 必须非常仔细地配置事件关联，因为它会对事件处理性能产生负面影响，或者如果配置错误，会关闭比预期更多的事件（在最坏的情况下，甚至可以关闭所有问题事件）。</ note>

要安全地配置全局关联，请遵循以下重要提示：

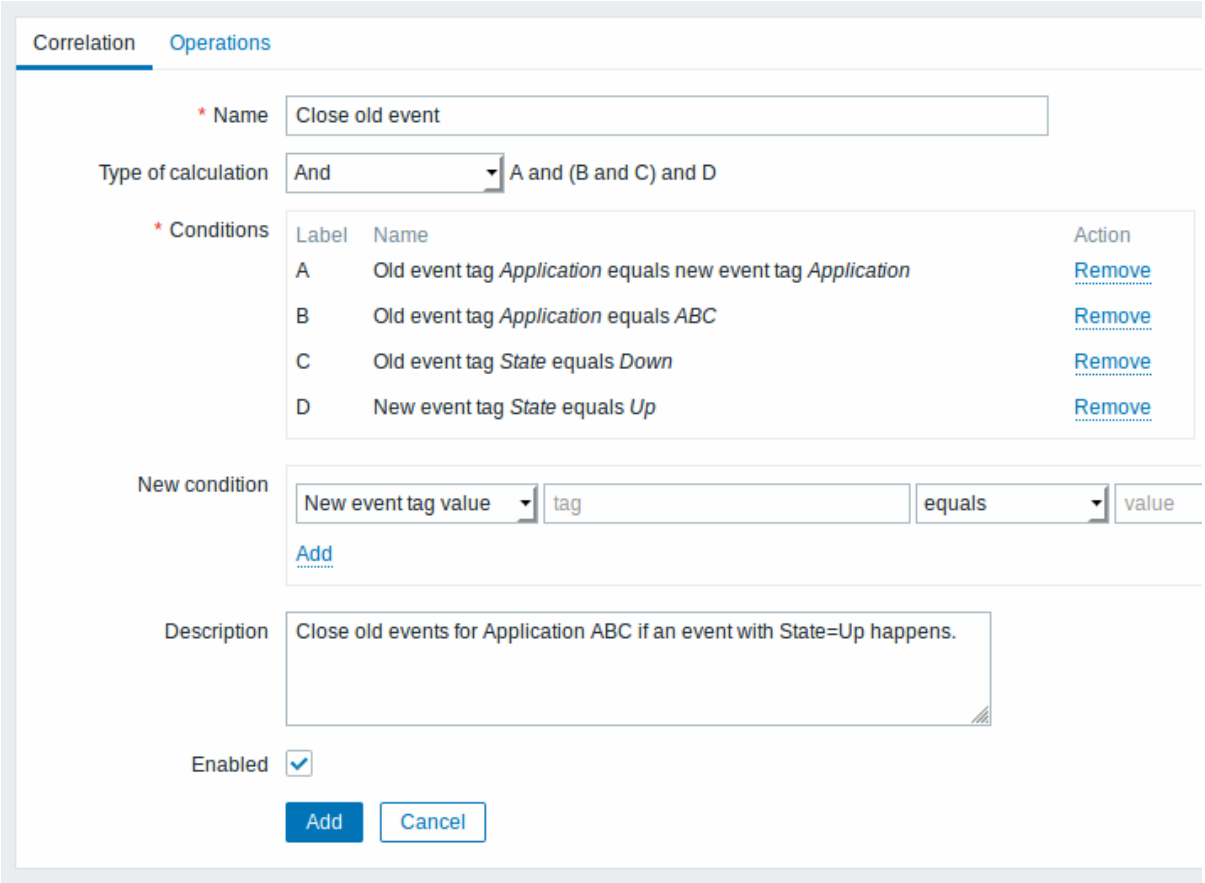
- 减少相关范围。始终为与旧事件配对的新事件设置唯一标记，并使用新事件标记关联条件;
- 使用关闭旧事件操作时，根据旧事件添加条件（或者可以关闭所有现有问题）；
- 避免使用可能最终被不同关联配置使用的常见标记名称;
- 保持关联规则的数量仅限于您真正需要的数量。

可参考: known issues.

配置

要全局配置事件关联规则：

- 去 *Configuration* → *Event correlation*
- 单击右侧创建相关性（或在相关名称上编辑现有规则）
- 在表单中输入关联规则的参数



所有必填输入字段都标有红色星号。

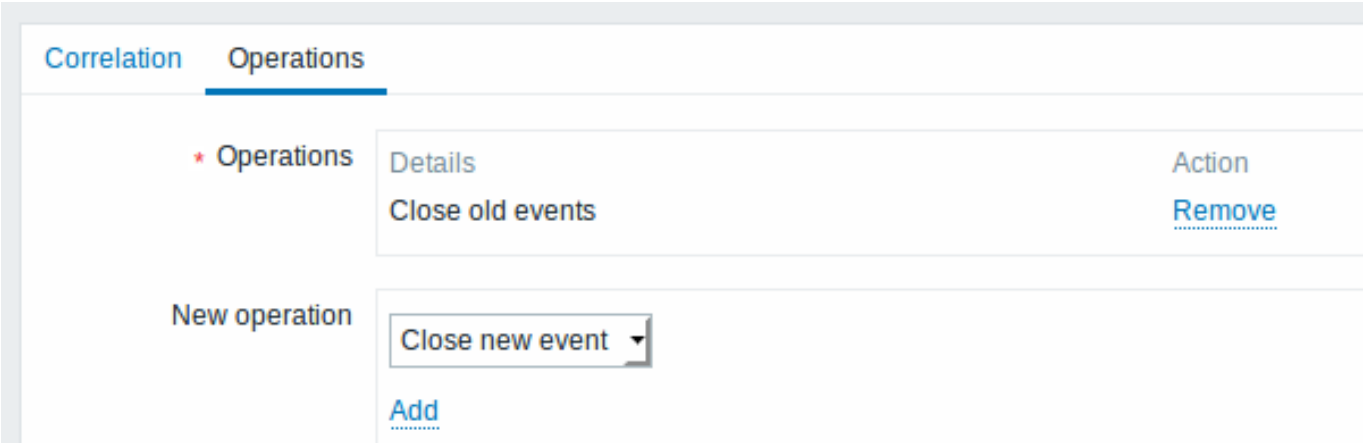
参数描	
名称唯	的关联规则名。

参数	描	
计算类型可以使		以下计算条件选项：和 - 必须满足所有条件 或 - 如果满足一个条件就足够了 和 /或 - 具有不同条件类型的 AND 和具有相同条件类型的 OR 自定义表达式 - 用于评估操作条件的用户定义计算公式。它必须包括所有条件 (表示为大写字母 A，B，C，...)，可能包括空格，制表符，括号 ()，和 (区分大小写)，或（区分大小写)，不区分大小写)。
条件条		列表，从新条件字段中选择。

参数描	
新条件选择	事件相关的条件，然后单击添加。可以使用以下条件：示例 1]]) |旧事件标记 - 将新事件与具有相应旧事件标记的旧事件匹配 |旧事件标记值 - 将新事件与旧事件匹配：新事件标记 - 匹配具有相应事件的新事件事件标记到旧事件 // = // - 具有相应的旧事件标记值 新事件主机组 - 将属于相应主机组的新事件与旧事件匹配 // %% <> %% // -事件标记对 - 将新事件与旧事件，如果两个事件中指定标签的值匹配。标签名称无需 < 有相应的旧事件标记值 *not like* - 在相应的旧事件标记中没有此字符串 value 新事件标记值 - 如果新事件：// = // - 具有相应的新事件标记值 // %% <> %, 则将新事件与旧事件匹配 % // - 没有相应的新事件标记值 // like // - 在新事件标记值中具有相应的字符串 *not like* - 在相应的新事件中没有此字符串标签值

参数描	
说明关 已启用如果	规则说明。 中此复选框，则将启用关联规则。

- 在表单中选择关联规则的操作



参数描	
操作从	新操作 * 字段中选择的操作列表

参数描	
新操作选择	事件关联时执行的操作，然后单击添加。可以使用以下操作：关闭旧事件 - 在发生新事件时关闭旧事件。使用关闭旧事件操作时，始终根据旧事件添加条件，或者可以关闭所有现有问题。关闭新事件 - 当事件发生时关闭新事件

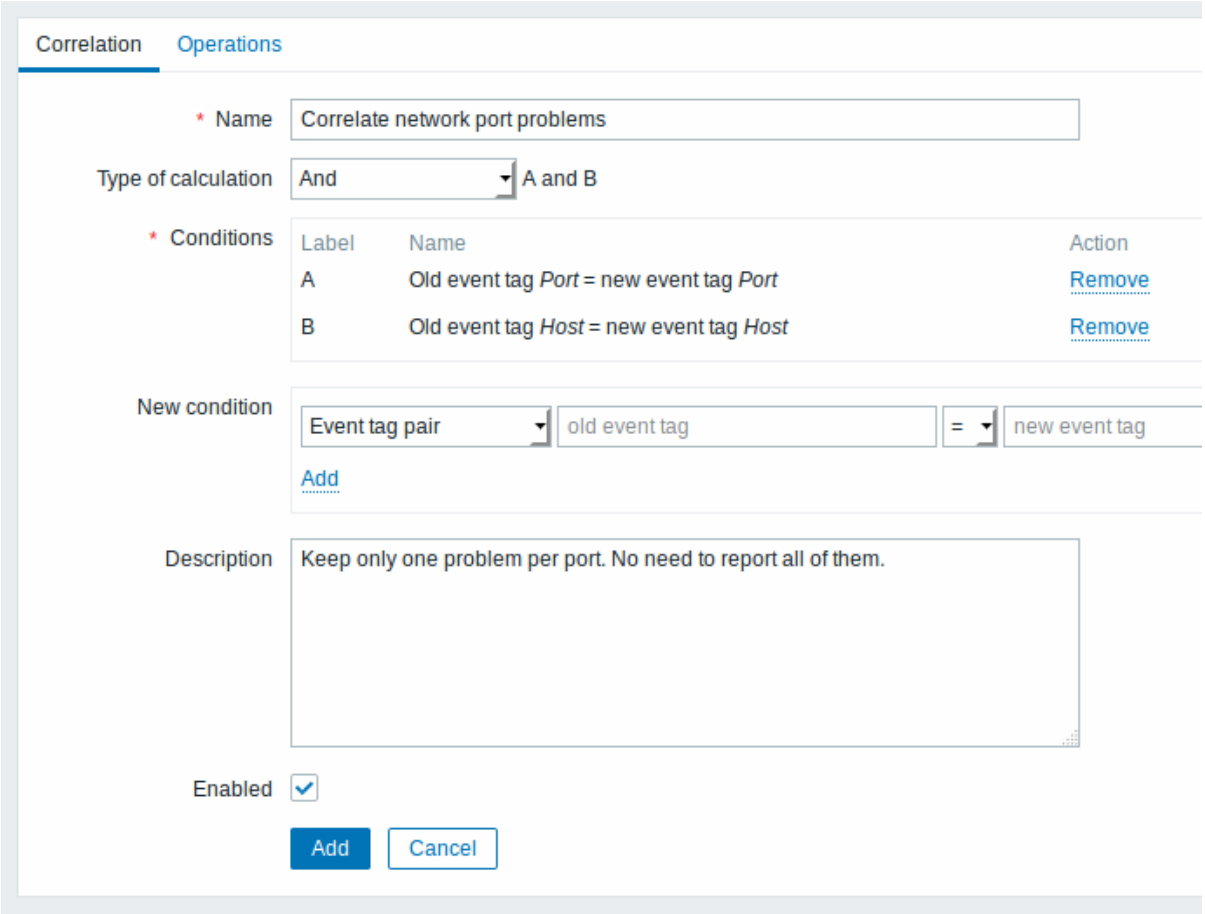
<note warning> 由于配置错误，可能会为无关问题创建类似的事件标记，请查看下面列出的案例！</ note>

- 实际标记和标记值仅在触发器触发时可见。如果使用的正则表达式无效，则使用 * UNKNOWN * 字符串静默替换它。如果错过了具有 * UNKNOWN * 标记值的初始问题事件，则可能会出现具有相同 * UNKNOWN * 标记值的后续 OK 事件，这些事件可能会关闭它们不应关闭的问题事件。
- 如果用户使用不带宏函数的 {ITEM.VALUE} 宏作为标记值，则有 255 个字符的限制。当日志消息很长并且前 255 个字符是非特定的时，这也可能导致类似的事件标记用于不相关的问题。

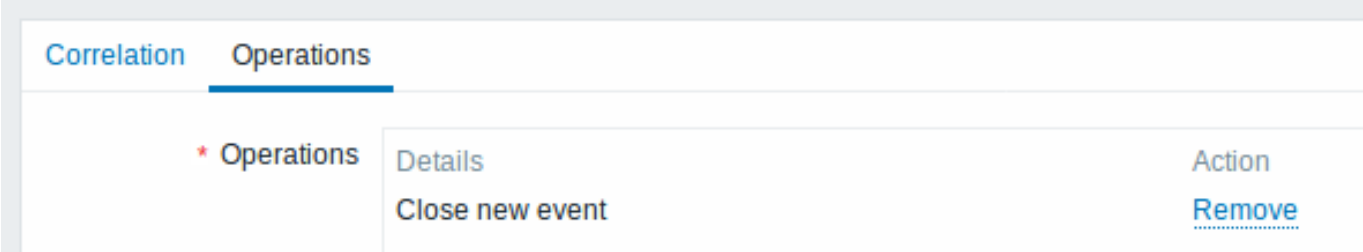
示例

示例 1

停止来自同一网络端口的重复问题事件。



如果触发器上存在// Host 和 Port //标记值，并且它们在原始事件和新事件中相同，则此全局关联规则将关联问题。



此操作将关闭同一网络端口上的新问题事件，仅保持原始问题打开。

7 模板

概述

模板是可以方便地应用于多个主机的一组实体。实体可以是：

- 监控项
- 触发器
- 图形
- 应用
- 聚合图形（自 *Zabbix 2.0* 起）
- 自动发现规则 (自 *Zabbix 2.0* 起)
- web 场景 (自 *Zabbix 2.0* 起)

由于现实生活中的许多主机是相同或类似的，所以，您为一个主机创建的一组实体（项目，触发器，图形，...）可能对许多人有用。当然，您可以将它们复制到每个新的主机上，但需要费很大功夫。相反，使用模板，您可以将它们复制到一个模板，然后根据需要将模板应用于尽可能多的主机。

当模板链接到主机时，模板的所有实体（项目，触发器，图形，...）都将添加到主机。模板直接分配给每个单独的主机（而不是主机组）。

模板通常用于为特定服务或应用程序（如 Apache，MySQL，PostgreSQL，Postfix ...）分组实体，然后应用于运行这些服务的主机。

使用模板的另一个好处是当所有主机都需要更改时。只需要在模板上更改某些内容将会将更改应用到所有链接的主机。

因此，使用模板是减少工作量并简化 Zabbix 配置的好方法。

在创建和配置模板中继续。

8 模板开箱即用

概述

Zabbix 致力于提供越来越多有用的开箱即用templates列表。开箱即用的模板已预先配置，因此是加速监视作业部署的有用方法。

1 网络设备的标准化模板

概述

为了对交换机和路由器等网络设备进行监控，我们创建了两个所谓的模型：网络设备本身（基本上是机箱）和网络接口。

由于 Zabbix 3.4 提供了许多网络设备系列模板。所有模板都覆盖（尽可能从设备中获取这些项目）：

- 机箱故障监控（电源，风扇和温度，总体状态）
- 机箱性能监控（CPU 和内存项）
- 机箱资产收集（序列号，型号名称，固件版本）
- 使用 IF-MIB 和 EtherLike-MIB 进行网络接口监控（接口状态，接口流量负载，以太网的双工状态）

这些模板可用：

- 在配置 - > 模板的新安装中;
- 在official Zabbix 模板库中。如果您已从 3.4 之前的 Zabbix 版本升级，则可以从 XML 导入这些模板。

如果要导入新的开箱即用模板，您可能还需要将 “@Network 自动发现接口” 全局正则表达式更新为：

```
Result is FALSE: ^Software Loopback Interface
Result is FALSE: ^(In)?[lL]oop[bB]ack[0-9._]*$
Result is FALSE: ^NULL[0-9.]*$
Result is FALSE: ^[lL]o[0-9.]*$
Result is FALSE: ^[sS]ystem$
Result is FALSE: ^Nu[0-9.]*$
```

在大多数系统上过滤掉环回和空接口。

设备

可用模板的设备系列列表：

模板名称提供商	设备系列	已知模型	操作系统已用 MIB 库		**标签(/manual/config/templates_out_of_the_box/network_d	
Template Net Alcatel Timetra TiMOS SNMPv2	Alcatel	Alcatel Time-tra	ALCATEL SR 7750	TiMOS	TIMETRA-SYSTEM-MIB,TIMETRA-CHASSIS-MIB	Certified
Template Net Brocade FC SNMPv2	Brocade	Brocade FC switches	Brocade 300 SAN Switch-	-	SW-MIB,ENTITY-MIB	Performance, Fault
Template Net Brocade_Foundry Stackable SNMPv2	Brocade	Brocade ICX	Brocade ICX6610, Brocade ICX7250-48, Brocade ICX7450-48F		FOUNDRY-SN-AGENT-MIB, FOUNDRY-SN-STACKING-MIB	Certified

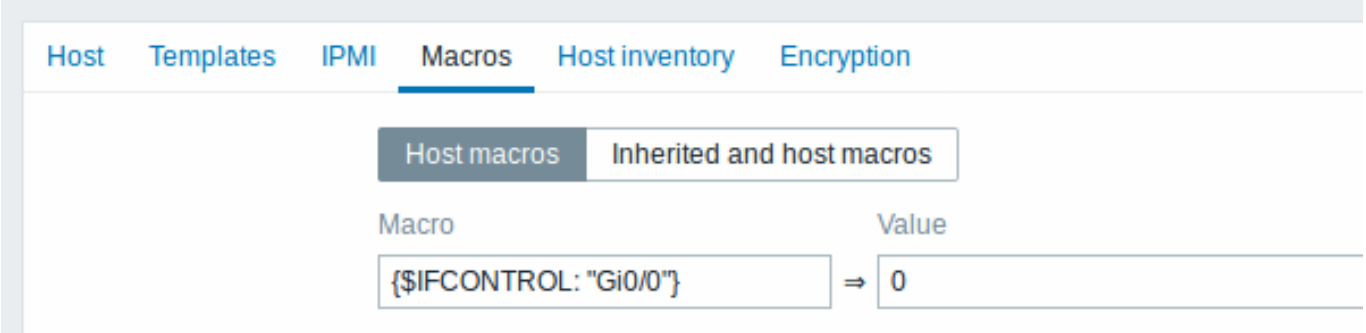
模板名称提供商	设备系列	已知模型	操作系统已用 MIB 库	**标签(/manual/config/templates_out_of_the_box/network_d		
Template Net Brocade_Foundry Nonstackable SNMPv2	Brocade Foundry	Brocade MLX, Foundry	Brocade MLXe, Foundry FLS648, Foundry FWSX424		FOUNDRY-SN-AGENT-MIB	Performance, Fault
Template Net Cisco IOS SNMPv2	Cisco	Cisco IOS ver > 12.2 3.5	Cisco C2950	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	Certified
Template Net Cisco releases later than 12.0_3_T and prior to 12.2_3.5_ SNMPv2	Cisco	Cisco IOS > 12.0 3 T and 12.2 3.5	-	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	Certified
Template Net Cisco releases prior to 12.0_3_T SNMPv2	Cisco	Cisco IOS 12.0 3 T	-	IOS	OLD-CISCO-CPU-MIB,CISCO-MEMORY-POOL-MIB	Certified
Template Net D-Link DES_DGS Switch SNMPv2	D-Link	DES/DGX switches	D-Link DES-xxxx/DGS-xxxx,DLINK DGS-3420-26SC	-	DLINK-AGENT-MIB,EQUIPMENT-MIB,ENTITY-MIB	Certified
Template Net D-Link DES 7200 SNMPv2	D-Link	DES-7xxx	D-Link DES 7206	-	ENTITY-MIB,MY-SYSTEM-MIB,MY-PROCESS-MIB,MY-MEMORY-MIB	Performance Fault Interfaces
Template Net Dell Force S-Series SNMPv2	Dell	Dell Force S-Series	S4810		F10-S-SERIES-CHASSIS-MIB	Certified
Template Net Extreme Exos SNMPv2	Extreme	Extreme EXOS	X670V-48x	EXOS	EXTREME-SYSTEM-MIB,EXTREME-SOFTWARE-MONITOR-MIB	Certified
Template Net Huawei VRP SNMPv2	Huawei	Huawei VRP	S2352P-EI	-	ENTITY-MIB,HUAWEI-ENTITY-EXTENT-MIB	Certified
Template Net Intel_Qlogic Infiniband SNMPv2	Intel/QLogic	Intel/QLogic In-fini-band de-vices	Infiniband 12300		ICS-CHASSIS-MIB	Fault Inventory

模板名称提供商	设备系列	已知模型	操作系统已用 MIB 库	**标签(/manual/config/templates_out_of_the_box/network_d	
Template Net Juniper SNMPv2	Juniper	MX,SRX,EX mod-els	Juniper MX240, Juniper EX4200-24F	JunOS	JUNIPER-MIB	Certified
Template Net Mellanox SNMPv2	Mellanox	Mellanox In-fini-band de-vices	SX1036	MLNX-OS	HOST-RESOURCES-MIB,ENTITY-MIB,ENTITY-SENSOR-MIB,MELLANOX-MIB	Certified
Template Net Mikrotik SNMPv2	Mikrotik	Mikrotik RouterOS de-vices	Mikrotik CCR1016-12G, Mikrotik RB2011UAS-2HnD, Mikrotik 912UAG-5HPnD, Mikrotik 941-2nD, Mikrotik 951G-2HnD, Mikrotik 1100AHx2	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
Template Net QTech QSW SNMPv2	QTech	Qtech de-vices	Qtech QSW-2800-28T	-	QTECH-MIB,ENTITY-MIB	Performance Inventory
Template Net Ubiquiti AirOS SNMPv1	Ubiquiti	Ubiquiti AirOS wire-less de-vices	NanoBridge,NanoStation	AirOS	HOST-RESOURCES-MIB,IEEE802dot11-MIB	Performance
Template Net HP Comware HH3C SNMPv2	HP	HP (H3C) Comware	HP A5500-24G-4SFP HI Switch	-	HH3C-ENTITY-EXT-MIB,ENTITY-MIB	Certified
Template Net HP Enterprise Switch SNMPv2	HP	HP En-ter-prise Switch	HP ProCurve J4900B Switch 2626, HP J9728A 2920-48G Switch	-	STATISTICS-MIB,NETSWITCH-MIB,HP-ICF-CHASSIS,ENTITY-MIB,SEMI-MIB	Certified
Template Net TP-LINK SNMPv2	TP-LINK	TP-LINK	T2600G-28TS v2.0	-	TPLINK-SYSMONITOR-MIB,TPLINK-SYSINFO-MIB	Performance Inventory
Template Net Netgear Fastpath SNMPv2	Netgear	Netgear Fast-path	M5300-28G	-	FASTPATH-SWITCHING-MIB,FASTPATH-BOXSERVICES-PRIVATE-MIB	Fault Inventory

模板设计

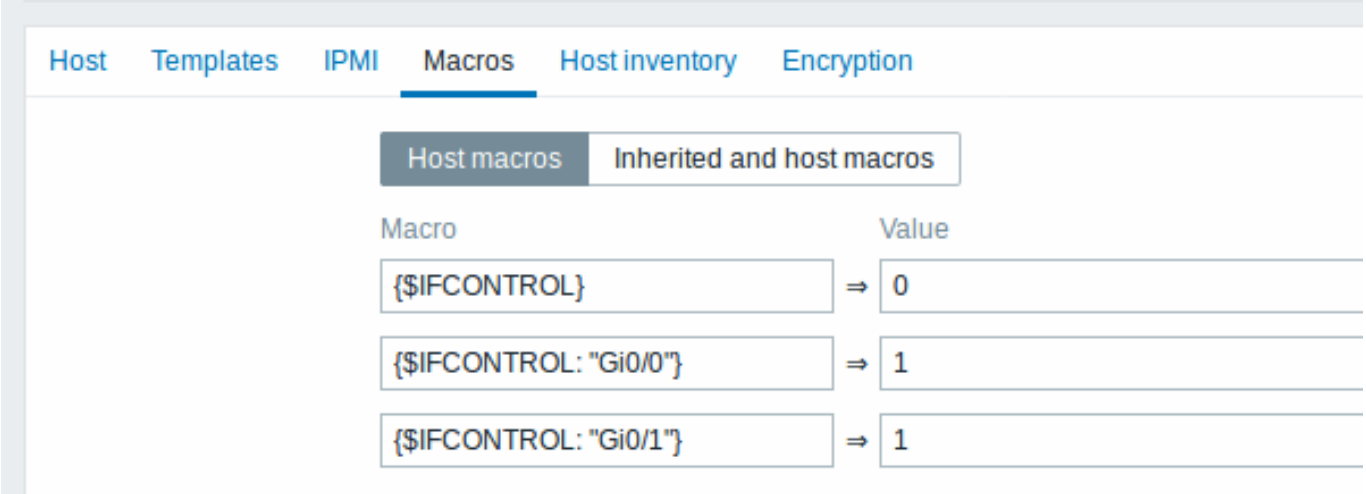
模板的设计考虑到以下因素：

- 尽可能使用用户宏，因此用户可以调整触发器
- 尽可能使用低级别发现来最小化不支持的项目数
- 为 SNMPv2 提供了模板。如果已知大多数设备不支持 SNMPv2，则使用 SNMPv1。
- 所有模板都依赖于 Template ICMP Ping，因此 ICMP 也会检查所有设备
- 项目不使用任何 MIB - SNMP OID 用于项目和低级别发现。因此，没有必要将任何 MIB 加载到 Zabbix 中以使模板工作。
- 发现接口以及 ifAdminStatus = down（2）的接口时，环回网络接口被过滤
- 尽可能使用 IF-MIB :: ifXTable 中的 64 位计数器。如果不支持，则使用默认的 32 位计数器。
- 所有发现的网络接口都有一个控制其运行状态（链接）的触发器。
- 如果您不想监视特定接口的此条件，请创建具有值为 0 的上下文的用户宏。例如：



其中 Gi0 / 0 是 {#IFNAME}。这样，触发器不再用于此特定接口。

* 您还可以更改所有触发器不会触发的默认行为，并仅将此触发器激活到有限数量的接口（如上行链路）



标签

- 性能 - 设备系列 MIB 提供了一种监控 CPU 和内存项的方法;
- 故障 - 设备系列 MIB 提供监控至少一个温度传感器的方法;
- 资产 - 设备系列 MIB 提供了至少收集设备序列号和型号名称的方法;
- 认证 - 涵盖上述所有三个主要类别。

9 事件通知（**ZX**）

9 Notifications upon events 概述

Overview

假设我们已经配置了一些监控项和触发器，并且由于触发器状态的改变，导致发生一些事件，之后要考虑的就是动作 (action)。

Assuming that we have configured some items and triggers and now are getting some events happening as a result of triggers changing state, it is time to consider some actions.

首先，我们不想一直盯着触发器或事件列表。最好是在发生比较严重的事情（如异常）时，接收到通知。另外，当发生问题时，我们希望看到所有相关的人都能收到通知。

To begin with, we would not want to stare at the triggers or events list all the time. It would be much better to receive notification if something significant (such as a problem) has happened. Also, when problems occur, we would like to see that all the people

concerned are informed.

这就是为什么发送通知是 Zabbix 提供的主要动作之一，可以在特定事件中通知到谁以及什么时间通知。

That is why sending notifications is one of the primary actions offered by Zabbix. Who and when should be notified upon a certain event can be defined.

为了能够发送和接收 Zabbix 的通知，您必须：

- 定义一些媒介 (media)
- 配置 action 向已定义的 media 发送消息

To be able to send and receive notifications from Zabbix you have to:

- define some media
- configure an action that sends a message to one of the defined media

Actions 由 conditions 和 operations 组成。总的说来，当条件满足时，执行相应的操作。两个主要操作是发送消息（通知）和执行远程命令。

Actions consist of *conditions* and *operations*. Basically, when conditions are met, operations are carried out. The two principal operations are sending a message (notification) and executing a remote command.

对于发现和自动注册创建的事件，可以使用一些其它操作。包括添加或删除主机，链接模板等。

For discovery and auto-registration created events, some additional operations are available. Those include adding or removing a host, linking a template etc.

1 媒介类型

1 Media types 概述

Overview

媒介是用于在 Zabbix 中发送通知和警报的传送通道。

Media are the delivery channels used for sending notifications and alerts in Zabbix.

您可以配置多种媒介类型：

- E-mail
- SMS
- Jabber
- Ez Texting
- 自定义警报脚本

You can configure several media types:

- E-mail
- SMS
- Jabber
- Ez Texting
- Custom alertscripts

1 E-mail

1 E-mail

概述

Overview

要将电子邮件配置为邮件的传递通道，您需要将电子邮件配置为媒介类型，并为用户分配具体的邮件地址。

To configure e-mail as the delivery channel for messages, you need to configure e-mail as the media type and assign specific addresses to users.

配置

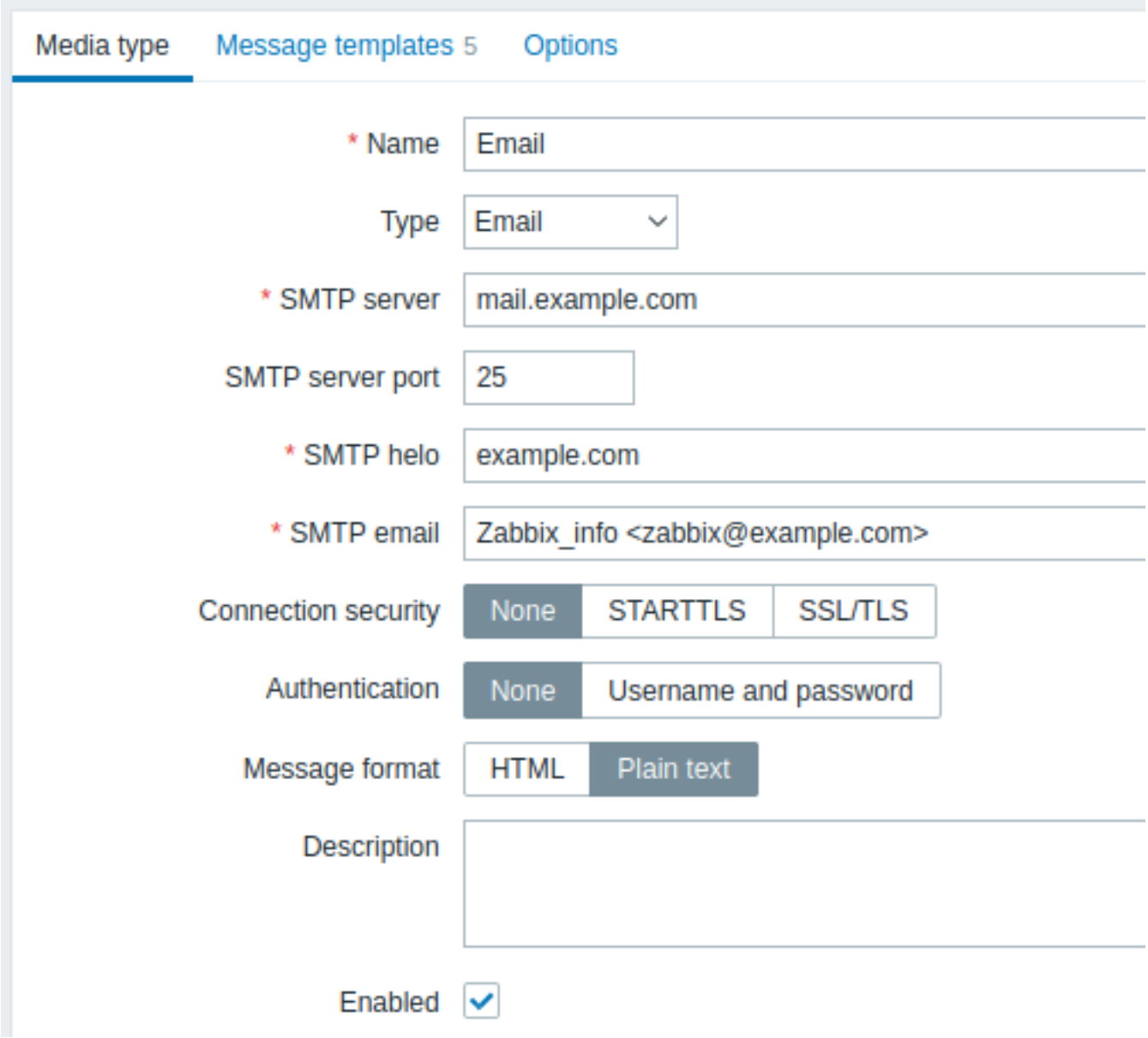
Configuration

配置电子邮件为媒介类型：

To configure e-mail as the media type:

- 在 管理 - > 媒体类型中设置
- 点击创建媒介类型 (或者点击预定义媒介类型的列表中的 *E-mail*)

- Go to *Administration → Media types*
- Click on *Create media type* (or click on *E-mail* in the list of pre-defined media types).



媒介类型属性：

The **Media type** tab contains general media type attributes:

All mandatory input fields are marked with a red asterisk.

参数说	
Name	媒介类型的名称.
Type	选择 *Email*.

参数	说
SMTP server	设置SMTP服务器来处理传出的消息.
SMTP server port	设置SMTP服务器端口来处理传出的消息. *Zabbix 3.0*版本之后支持此选项.
SMTP helo	设置正确的SMTP helo值，通常是域名.

参数	说
SMTP email	此处输入的地址将被用作发送消息的**From地址。选中该复选框以验证**SMTP**服务器的在***Zabbix-HQ <zabbix@company.com>***中添加发件人显示名称（如***“Zabbix-HQ”***）**Zabbix 2.2**版本之后，支持实际的电子 SL证书. URLOPT_CAPATH](http://curl.haxx.se/l 中以进行证书验证 FYPEER](http://curl.haxx.se/libcurl/c/C x 3.0*之后支持此选项.

参数说	
SSL verify host	标记该复选框以验证SMTP服务器证书的公用名称字段或主题备用名称字段是否匹配 选择认证级别：设置cURL选项CURLOPT_SSL_VERIFYHOST. **N//Zabbix 3.0**之后支持此选项**.** **|** **|Authentication//** Norm 认证中使用的用户名. ne** - 没有设置cURL选项 设置[CURLOl password** - CURLOPT_LOGIN_OPTIONS在”AUTH=PLAIN 中设置 // Zabbix// Zabbix 3.0之后. 支持此选项 | |Username// 认证中使用的密码. T_USERNAME](http://curl.haxx. 的值. 设置[CURLOPT_.0 之后. 支持此选项 | |Password// Zabbix 3.0之后 *. 标记该复选框以启用媒体类型.ASSWORD](http://cur 的值. 支持此选项 | |*Enable//

Parameter	Description
Name	Name of the media type.
Type	Select *Email* as the type.
SMTP server	Set an SMTP server to handle outgoing messages.
SMTP server port	Set the SMTP server port to handle outgoing messages. This option is supported *starting with Zabbix 3.0*.
SMTP helo	Set a correct SMTP helo value, normally a domain name.
SMTP email	The address entered here will be used as the **From** address for the messages sent. Adding a sender display name (like ”Zabbix-HQ” in *Zabbix-HQ <zabbix@company.com>* in the screenshot above) with the actual e-mail address is supported since Zabbix 2.2 version. There are some restrictions on display names in Zabbix emails in comparison to what is allowed by RFC 5322, as illustrated by examples: Valid examples: *zabbix@company.com* (only email address, no need to use angle brackets) *Zabbix HQ <zabbix@company.com>* (display name and email address in angle brackets) *∑Ω-monitoring <zabbix@company.com>* (UTF-8 characters in display name) Invalid examples: *Zabbix HQ zabbix@company.com* (display name present but no angle brackets around email address) *”Zabbix\@\<H(comment)Q\>” <zabbix@company.com>* (although valid by RFC 5322, quoted pairs and comments are not supported in Zabbix emails)
Connection security	Select the level of connection security: **None** - do not use the CURLOPT_USE_SSL option **STARTTLS** - use the CURLOPT_USE_SSL option with CURLUSESSL_ALL value **SSL/TLS** - use of CURLOPT_USE_SSL is optional This option is supported *starting with Zabbix 3.0*.
SSL verify peer	Mark the checkbox to verify the SSL certificate of the SMTP server. The value of ”SSLCALocation” server configuration directive should be put into CURLOPT_CAPATH for certificate validation. This sets cURL option CURLOPT_SSL_VERIFYPEER. This option is supported *starting with Zabbix 3.0*.
SSL verify host	Mark the checkbox to verify that the *Common Name* field or the *Subject Alternate Name* field of the SMTP server certificate matches. This sets cURL option CURLOPT_SSL_VERIFYHOST. This option is supported *starting with Zabbix 3.0*.
Authentication	Select the level of authentication: **None** - no cURL options are set (since 3.4.2) **Username and password** - implies ”AUTH=*” leaving the choice of authentication mechanism to cURL (until 3.4.2) **Normal password** - CURLOPT_LOGIN_OPTIONS is set to ”AUTH=PLAIN” This option is supported *starting with Zabbix 3.0*.
Username	User name to use in authentication. This sets the value of CURLOPT_USERNAME. This option is supported *starting with Zabbix 3.0*.
Password	Password to use in authentication. This sets the value of CURLOPT_PASSWORD. This option is supported *starting with Zabbix 3.0*.
Enabled	Mark the checkbox to enable the media type.

Attention:
要使 SMTP 验证选项可用，Zabbix 服务器应使用 cURL 7.20.0 或更高版本的编译选项 - with-libcurl 进行编译。

Attention:
To make SMTP authentication options available, Zabbix server should be compiled with the --with-libcurl compilation option with cURL 7.20.0 or higher.

Options 标签页

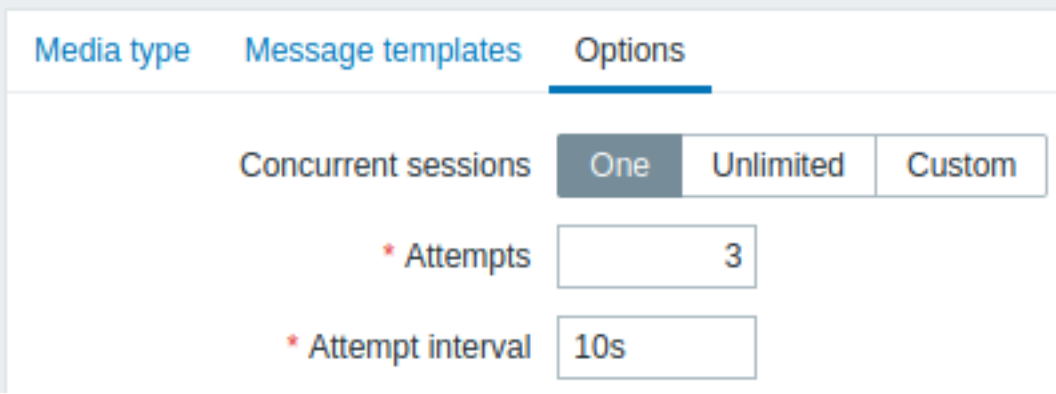
Options

在 e-mail 媒介类型的 **Options** 标签页配置 包含了告警进程设置。同样，其他媒介也有同样的可配置的选项。

The **Options** tab in the e-mail media type configuration contains alert processing settings. The same set of options are configurable for other media types, too.

所有的媒介类型是并行处理的。每个媒介类型的当前最大会话数量是可配置的，且服务端告警进程数量通过参数StartAlerters 限制。一个触发器产生的警报是顺序执行的。

All media types are processed in parallel. The maximum number of concurrent sessions is configurable per media type, but the total number of alerter processes on server can only be limited by the StartAlerters parameter. Alerts generated by one trigger are processed sequentially.



参数说

并行会话选择并	告警会话媒介类型的数量: **One** - 单会话 **Unlimited** - 不限制会话数量 **Custom** - 自定义会话数量 Unlimited/high 取值意味在发送通知时会产生更多并行会话且会话数量不断增加. Unlimited/high 取值应该应用在同时发送大量通知的场景的下使用.
尝试次数发送通	(异常或失败时) 尝试的次数, 该值最大可设置为 10, 默认情况下为 3. 如果设置为 1, 那么通知只会发送一次, 即使失败也不会重发.
重试次数在发送	败的情况下重试发送的频度, 默认单位为秒。如果设置为 0, 那么发送失败时会立即重发. 设置也支持自定义时间单位, 例如,5s,1m.

Parameter	Description
Concurrent sessions	Select the number of parallel alerter sessions for the media type: **One** - one session **Unlimited** - unlimited number of sessions **Custom** - select a custom number of sessions Unlimited/high values mean more parallel sessions and increased capacity for sending notifications. Unlimited/high values should be used in large environments where lots of notifications may need to be sent simultaneously.

Parameter	Description
Attempts	Number of attempts for trying to send a notification. Up to 10 attempts can be specified; default value is '3'. If '1' is specified Zabbix will send the notification only once and will not retry if the sending fails.
Retry interval	Frequency of trying to resend a notification in case the sending failed, in seconds (0-60). If '0' is specified, Zabbix will retry immediately. Time suffixes are supported, e.g. 5s, 1m.

用户媒介

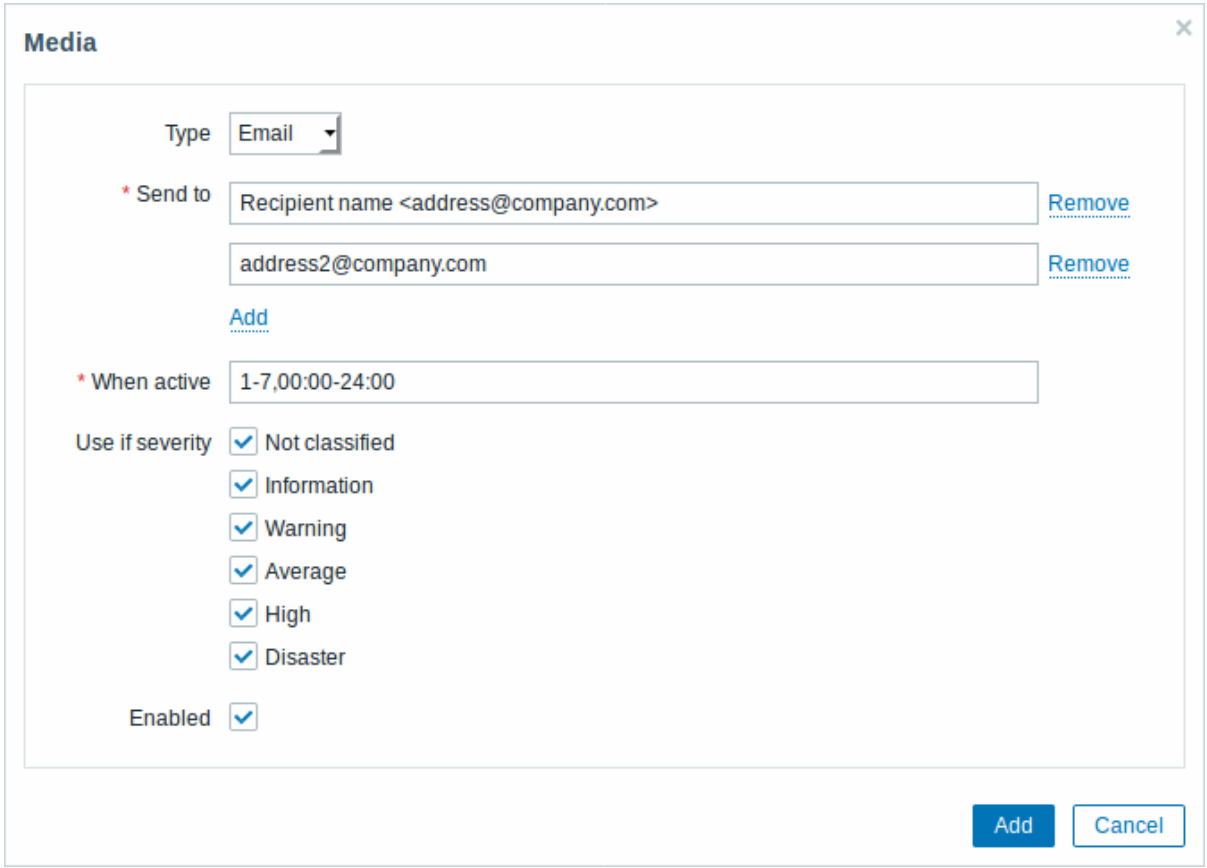
User media

要为用户分配一个邮件地址：

To assign a specific address to the user:

- 在 管理 - > 用户中进行设置
- 打开用户属性窗体
- 在媒介选项卡中，单击 *Add*

- Go to *Administration → Users*
- Open the user properties form
- In Media tab, click on *Add*



用户媒介属性：

User media attributes:

参数说	
Type	选择 *Email* 为媒介类型.

参数	说
Send to	指定发送消息的电子邮件地址. 添加收件人显示名称 (*Some User <user@domain.tld>* 如上截图中的 “Some User”) 以及 Zab-bix 2.2 版本之后支持的实际电子邮件地址. 请参阅媒体类型属性SMTP电子邮件描述中显示名称和电子邮件地址的示例和限制.

参数	说
When active	您可以限制邮件发送的时间，例如仅限工作日 (1-5,09：00-18：00）. 格式的描述请参见时间段格式页面.
Use if severity	标记您要接收通知的触发严重性的复选框. *Note* 对于非触发事件，使用默认严重性（’未分类’），因此如果要接收非触发事件的通知，请将其保留.

参数说	
Status	用户媒介的状态 **Enabled** - 正在使用. **Disabled** - 没有被使用.

Parameter	Description
Type	Select *Email* as the type.
Send to	Specify e-mail addresses to send the messages to. To add more than one address click on *Add* below the address field. If multiple e-mail addresses are specified, one e-mail will be sent to all the specified recipients. You may add the recipient display name (like ”Recipient name” in *Recipient name <address1@company.com>* in the screenshot above) with the actual e-mail address. See examples and restrictions on display name and email address in media type attribute SMTP email description.
When active	You can limit the time when messages are sent, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format. User macros are supported.
Use if severity	Mark the checkboxes of trigger severities that you want to receive notifications for. *Note* that the default severity (’Not classified’) **must be** checked if you want to receive notifications for non-trigger events. After saving, the selected trigger severities will be displayed in the corresponding severity colours while unselected ones will be greyed out.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

2 SMS

2 SMS

概述

Overview

Zabbix 支持使用连接到 Zabbix 服务器的串行端口的串行 GSM 调制解调器发送 SMS 消息。

Zabbix supports the sending of SMS messages using a serial GSM modem connected to Zabbix server’s serial port.

需要确保如下：

Make sure that:

- 串行设备的速度（在 Linux 下通常为/dev/ttyS0）与 GSM 调制解调器的速度相匹配。Zabbix 没有设置串行链路的速度。它使用默认设置。
- ’zabbix’ 用户对串行设备有读/写访问权。运行命令 ls -l /dev/ttyS0 来查看串口设备的当前权限。
- GSM 调制解调器输入 PIN 码，并在电源复位后保留 PIN 码。或者，您可以在 SIM 卡上禁用 PIN。可以通过在终端软件（如 Unix minicom 或 Windows 超级终端）中发出命令 AT + CPIN =“NNNN”（NNNN 是您的 PIN 号，引号必须存在）来输入 PIN。
- The speed of the serial device (normally /dev/ttyS0 under Linux) matches that of the GSM modem. Zabbix does not set the speed of the serial link. It uses default settings.

- The 'zabbix' user has read/write access to the serial device. Run the command ls –l /dev/ttyS0 to see current permissions of the serial device.
- The GSM modem has PIN entered and it preserves it after power reset. Alternatively you may disable PIN on the SIM card. PIN can be entered by issuing command AT+CPIN="NNNN" (NNNN is your PIN number, the quotes must be present) in a terminal software, such as Unix minicom or Windows HyperTerminal.

Zabbix 已经使用这些 GSM 调制解调器进行了测试：

Zabbix has been tested with these GSM modems:

- Siemens MC35
- Teltonika ModemCOM/G10

- Siemens MC35
- Teltonika ModemCOM/G10

要将 SMS 配置为邮件的传送通道，需将 SMS 配置为媒体类型，并输入相应的电话号码。

To configure SMS as the delivery channel for messages, you also need to configure SMS as the media type and enter the respective phone numbers for the users.

配置

Configuration

要将 SMS 配置为媒介类型：

To configure SMS as the media type:

- 进入管理 - > 媒介类型
- 点击 创建媒介类型 (或者点击预定义的媒介类型列表中的 *SMS*).

- Go to *Administration* → *Media types*
- Click on *Create media type* (or click on *SMS* in the list of pre-defined media types).

媒介类型属性：

The **Media type** tab contains general media type attributes:

参数说	
Description	媒介类型的名称.
Type	选择 *SMS* 为媒介类型.
GSM modem	设置 GSM 调制解调器的串行设备名称.

Parameter	Description
Description	Name of the media type.
Type	Select *SMS* as the type.
GSM modem	Set the serial device name of the GSM modem.

Options 标签页包含了与其它媒介常见的告警设置 。请注意，并行发送 SMS 通知是不可能的。

The **Options** tab contains alert processing settings that are common for all media types. Note that parallel processing of sending SMS notifications is not possible.

用户媒介

User media

为用户分配电话号码：

To assign a phone number to the user:

- 进入 管理 - > 用户
- 打开用户属性窗体
- 在媒介选项卡中，单击 *Add*

- Go to *Administration* → *Users*
- Open the user properties form
- In Media tab, click on *Add*

用户媒介属性：

User media attributes:

参数说	
Type	选择 *SMS* 作为媒介类型.
Send to	指定要发送消息的电话号码.
When active	您可以限制邮件发送的时间，例如仅限工作日（1-5,09：00-18：00）. 格式描述请参见时间段规则 页面.
Use if severity	标记您要接收通知的触发严重性的复选框.
Status	媒介的状态 **Enabled** - 使用中. **Disabled** - 禁用.

Parameter	Description
Type	Select *SMS* as the type.
Send to	Specify the phone number to send messages to.
When active	You can limit the time when messages are sent, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format.
Use if severity	Mark the checkboxes of trigger severities that you want to receive notifications for. *Note* that the default severity ('Not classified') **must be** checked if you want to receive notifications for non-trigger events. After saving, the selected trigger severities will be displayed in the corresponding severity colours while unselected ones will be greyed out.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

5 自定义警报提示

5 Custom alertscripts

概述

Overview

如果您需更多告警媒介类型，则可以使用其它方式来执行此操作。您可以创建一个事件通知自定义的脚本。

If you are not satisfied with existing media types for sending alerts there is an alternative way to do that. You can create a script that will handle the notification your way.

告警脚本在 Zabbix 服务器上执行。这些脚本位于服务器配置文件中定义的目录中 **AlertScriptsPath**.

Alert scripts are executed on Zabbix server. These scripts are located in the directory defined in the server configuration file **AlertScriptsPath** variable.

这是一个示例警报脚本：

Here is an example alert script:

```
#####!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF
```

```
#####!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF
```

<note important> 从版本 3.4 开始，Zabbix 检查执行的命令和脚本的退出代码。任何与 ** 0 ** 不同的退出代码都被视为命令执行错误. 在这种情况下，Zabbix 会尝试重复执行失败。:::

Attention:
Starting from version 3.4 Zabbix checks for the exit code of the executed commands and scripts. Any exit code which is different from **0** is considered as a command execution error. In such case Zabbix will try to repeat failed execution.

环境变量不会为脚本保留或创建，因此它们应该被明确处理。

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

配置

Configuration

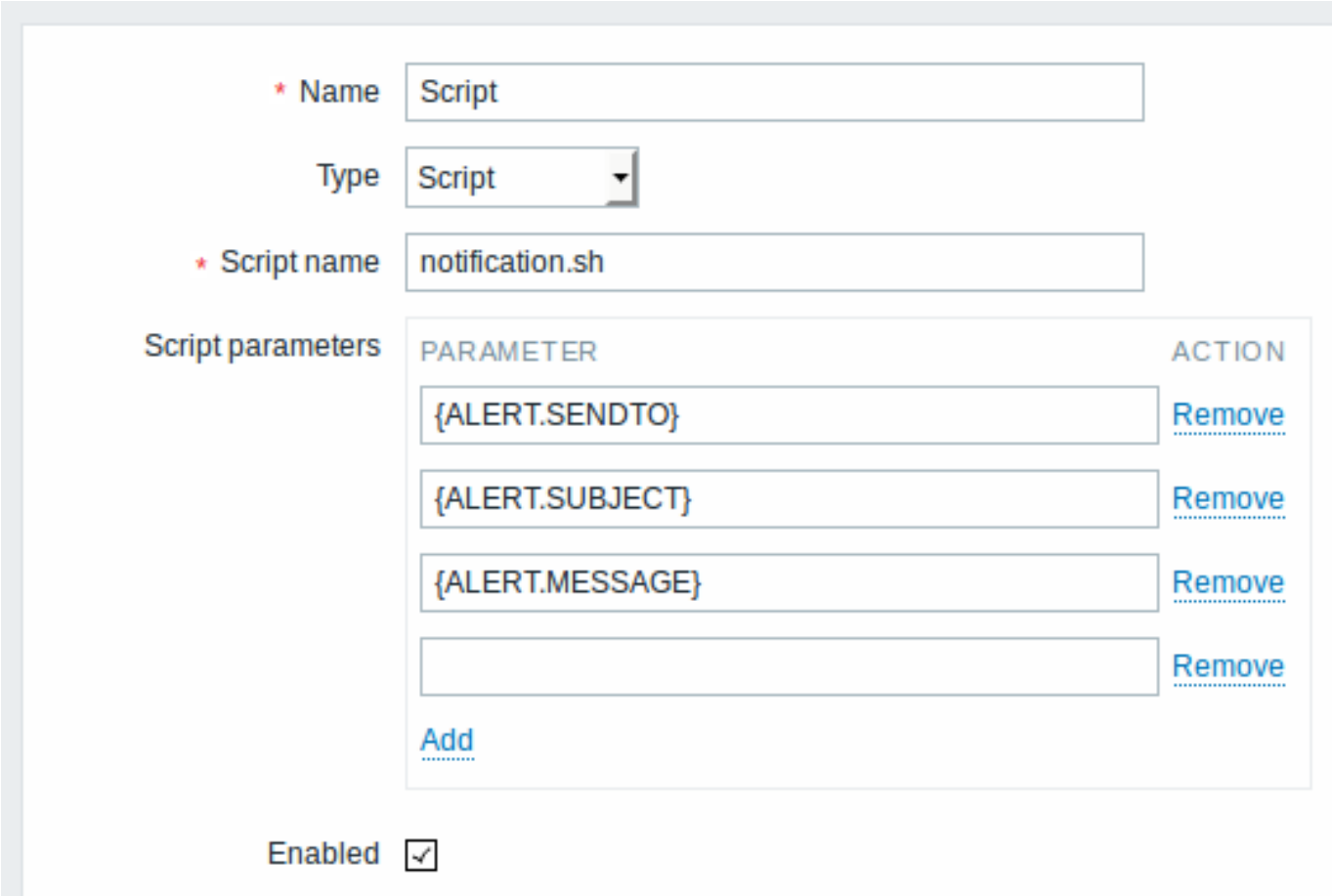
将自定义告警文本配置为媒介类型：

To configure custom alertscripts as the media type:

- 进入 管理 - > 媒介类型
- 点击创建媒介类型

- Go to *Administration → Media types*
- Click on *Create media type*

The **Media type** tab contains general media type attributes:



媒介类型属性：

All mandatory input fields are marked with a red asterisk.

参数说	
Name	输入媒介类型的名称.
Type	选择 *Script* 作为媒介类型.
Script name	输入脚本的名称.

参数说	
Script parameters	向脚本添加命令行参数. {ALERT.SENDTO} {ALERT.SUBJECT} and {ALERT.MESSAGE 宏 在脚本参数中是支持的 Zabbix 3.0 支持自定义脚本参数.

Parameter	Description
Name	Enter name of the media type.
Type	Select *Script* as the type.
Script name	Enter the name of the script.
Script parameters	Add command-line parameters to the script. {ALERT.SENDTO}, {ALERT.SUBJECT} and {ALERT.MESSAGE} macros are supported in script parameters. Customizing script parameters is supported since Zabbix 3.0.

Options 标签页包含了所以媒介类型常见的告警设置 。

The **Options** tab contains alert processing settings that are common for all media types.

Attention:
从 Zabbix 3.4.0 版本开始实现了多个告警媒介并行处理, 所以需要注意的是，当配置了多个告警脚本时，这些脚本是可以被告警进程并行处理的. 告警进程的进程数可以通过配置项 StartAlertersparameter进行限制。

Attention:
As parallel processing of media types is implemented since Zabbix 3.4.0, it is important to note that with more than one script media type configured, these scripts may be processed in parallel by alerter processes. The total number of alerter processes is limited by the StartAlerters parameter.

用户媒介

User media

为用户分配自定义告警提示符：

To assign custom alertscripts to the user:

- 进入 管理 - > 用户
- 打开用户属性窗体
- 在媒介选项卡中，单击 *Add*

- Go to *Administration → Users*
- Open the user properties form
- In Media tab, click on *Add*

用户媒介属性：

User media attributes:

参数说	
Type	选择自定义的alertcripts媒介类型.
Send to	指定收件人接收警报.
When active	您可以限制执行警示标记的时间，例如，仅限工作日 (1-5,09：00-18：00）.\\格式说明参见时间段规格.页面.

参数说	
Use if severity	勾选复选框 (Not classified/Information/W 标识您要接收通知的触发严重性.
Status	用户媒介的状态 **Enabled** - 使用中. **Disabled** -禁用.

Parameter	Description
Type	Select the custom alertscripts media type.
Send to	Specify the recipient to receive the alerts.
When active	You can limit the time when alertscripts are executed, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format.
Use if severity	Mark the checkboxes of trigger severities that you want to activate the alertscript for. *Note* that the default severity ('Not classified') **must be** checked if you want to receive notifications for non-trigger events. After saving, the selected trigger severities will be displayed in the corresponding severity colours while unselected ones will be greyed out.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

2 动作

2 Actions 概述

Overview

如果您希望由于事件而发生某些操作（例如发送通知），则需要配置动作 (actions)。

If you want some operations taking place as a result of events (for example, notifications sent), you need to configure actions.

可以根据所有支持的类型的事件来定义操作：

Actions can be defined in response to events of all supported types:

- 触发事件 - 当 trigger 的状态从 *OK* 转到 *PROBLEM* 或者从 *PROBLEM* 转到 *OK*
- 发现事件 - 发生网络发现时
- 自动注册事件 -当新的活动代理自动注册
- 内部事件 - 当项目不受支持或触发器进入未知状态

- Trigger events - when trigger status changes from *OK* to *PROBLEM* and back
- Discovery events - when network discovery takes place
- Auto registration events - when new active agents auto-register (or host metadata changes for registered ones)
- Internal events - when items become unsupported or triggers go into an unknown state

配置动作

Configuring an action

配置动作

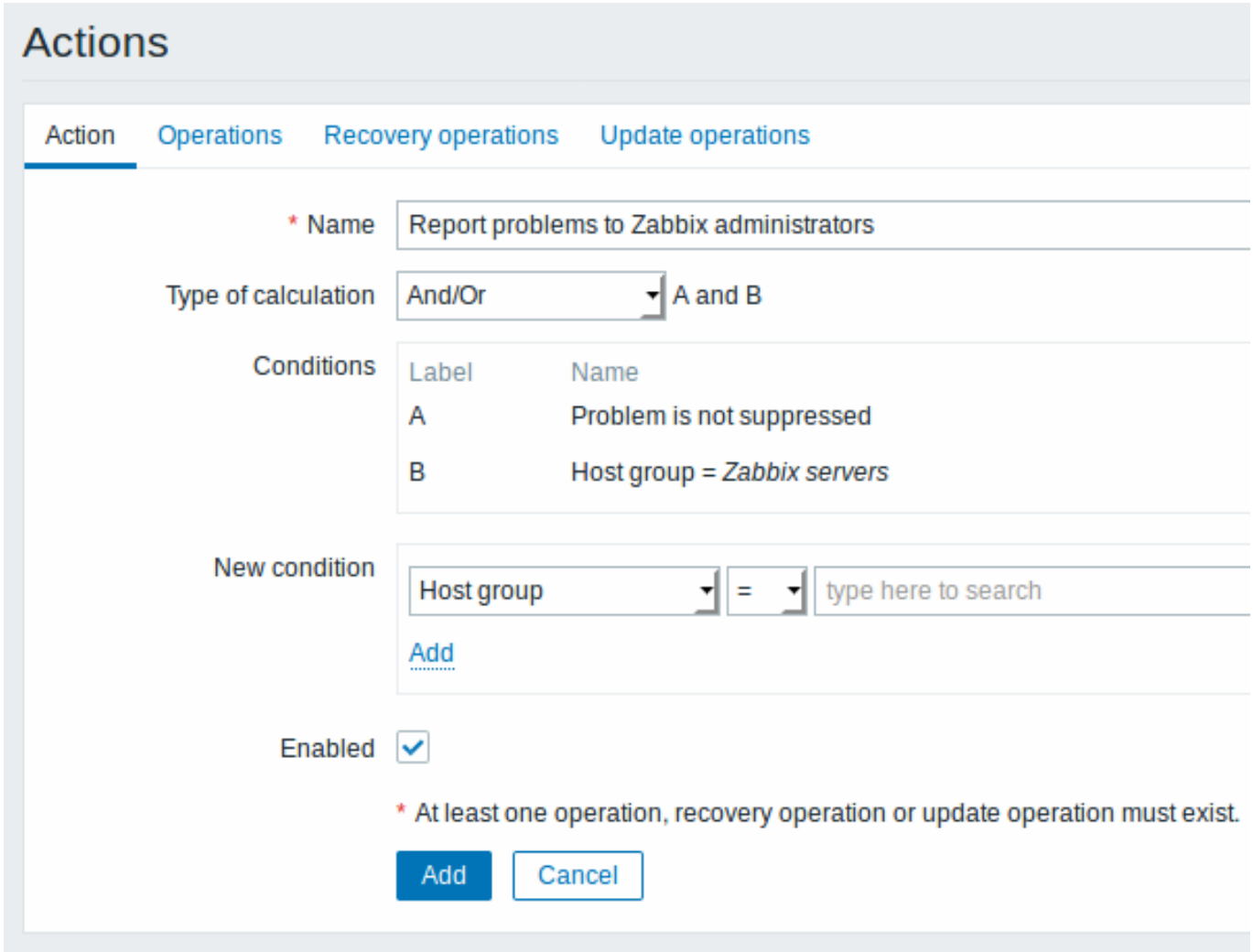
To configure an action, do the following:

- 进入 配置 **-** > 操作
- 从 *Event source* 下拉单中选择所需的来源
- 点击 创建 *action*
- 命名 action
- 选择进行操作的条件
- 选择操作来执行
- 选择恢复操作来执行

- Go to *Configuration* → *Actions*
- From the *Event source* dropdown select the required source
- Click on *Create action*
- Name the action
- Choose conditions upon which operations are carried out
- Choose the operations to carry out
- Choose the recovery operations to carry out

常见动作属性：

General action attributes:



All mandatory input fields are marked with a red asterisk.

参数说	
Name	action 名称.

参数说	
Type of calculation	选择评估manual/config/选项作为行动条件(有多个条件): **And** - 必须满足所有条件 **Or** - 如果满足一个条件就足够了 **And/Or** - 两者的组合：AND与不同的条件类型和OR具有相同的条件类型

参数说	
Conditions	行动条件清单。
New condition	选择一个新的动作条件并且点击 *Add*.
Enabled	选中该复选框以启用该操作. 否则将被禁用.

Parameter	Description
Name	Unique action name.
Type of calculation	Select the evaluation option for action conditions (with more than one condition): **And** - all conditions must be met **Or** - enough if one condition is met **And/Or** - combination of the two: AND with different condition types and OR with the same condition type **Custom expression** - a user-defined calculation formula for evaluating action conditions.
Conditions	List of action conditions.
New condition	Select a new action condition and click on *Add*.
Enabled	Mark the checkbox to enable the action. Otherwise it will be disabled.

1 条件

1 Conditions

概述

Overview

只有在事件与定义的条件匹配的情况下才执行操作。配置动作时设置条件。

An action is executed only in case an event matches a defined set of conditions. Conditions are set when configuring an 动作.

可以为基于触发的动作设置以下条件：

The following conditions can be set for trigger-based actions:

条件类型支持的	作说明
Application	= like not like

指定要排除的应用程序或应用程序.
= -事件属于与指定应用程序链接的项目的触发器.
like -事件属于与包含字符串的应用程序链接的项目的触发器.
not like -事

条件类型支持的	作说明	
Host group	= <>	指定要排除的主机组或主机组. **=** - 事件属于此主机组. **<>** - 事件不属于此主机组.指定父主机组隐含地选择所有嵌套的主机组.要仅指定父组，必须使用 ** %%

条件类型支持的	作说明	
Host	= <>	指定要排除的主机或主机 **=** - 属于这个主机的事件. **<>** - 不属于这个主机的事件.

条件类型支持的	作说明	
Tag	= <> like not like	指定事件标记或要排除的事件标记. **=** - 含有该标记的事件 **<>** - 不含该标记的事件 **like** -标签中包含此字符串的事件 **not like** - 标签中不包含此字符串的事件

条件类型支持的	作说明	
Tag value	= <> like not like	指定事件标签和值组合或要排除的标签和值组合 **=** - 包含该值和标签的事件 **<>** -不包含该值和标签的事件 **like** - 值和标签中包含该字符串的事件 **not like** - 值

条件类型支持的	作说明	
Trigger	= <>	指定触发器或要排除的触发器. **=** - 由该触发器产生的事件 **<>** - 除了这一个，由任何其他触发器生成的事件.

条件类型支持的	作说明	
Trigger name	like not like	在触发器名称中指定一个字符串或要排除的字符串. **like** - 事件由触发器生成，在名称中包含此字符串. 区分大小写. **not like** - 触发器名称中不包含该字符串. 区分

条件类型支持的	作说明	
Trigger severity	= <> >= <=	指定触发严重性. **=** - 等于触发严重性 **<>** - 不等于触发严重性 **>=** - 大于或等于触发严重性 **<=** - 小于或等于触发严重性.

条件类型支持的	作说明	
Time period	in not in	指定时间段或要排除的时间段. **in** - 事件时间在该时间段内. **not in** - 事件时间不在该时间段内. 格式描述参见Time period specification页面.

条件类型支持的	作说明	
Maintenance status	in not in	指定主机进行维护或不进行维护. **in** - 主机处于维护模式. **not in** - 主机不在维护模式. *Note*: 如果触发表达式中涉及到多个主机，则至少有一个主机不在维护模式

条件类型支持的		作说明

Condition type	Supported operators	Description
Application	= like not like	Specify an application or an application to exclude. **=** - event belongs to a trigger of the item that is linked to the specified application. **like** - event belongs to a trigger of the item that is linked to an application containing the string. **not like** - event belongs to a trigger of the item that is linked to an application not containing the string.
Host group	= <>	Specify host groups or host groups to exclude. **=** - event belongs to this host group. **<>** - event does not belong to this host group. Specifying a parent host group implicitly selects all nested host groups. To specify the parent group only, all nested groups have to be additionally set with the **<>** operator.
Template	= <>	Specify templates or templates to exclude. **=** - event belongs to a trigger inherited from this template. **<>** - event does not belong to a trigger inherited from this template.
Host	= <>	Specify hosts or hosts to exclude. **=** - event belongs to this host. **<>** - event does not belong to this host.
Tag	= <> like not like	Specify event tag or event tag to exclude. **=** - event has this tag **<>** - event does not have this tag **like** - event has a tag containing this string **not like** - event does not have a tag containing this string

Condition type	Supported operators	Description
Tag value	= <> like not like	Specify event tag and value combination or tag and value combination to exclude. **=** - event has this tag and value **<>** - event does not have this tag and value **like** - event has a tag and value containing these strings **not like** - event does not have a tag and value containing these strings
Trigger	= <>	Specify triggers or triggers to exclude. **=** - event is generated by this trigger. **<>** - event is generated by any other trigger, except this one.
Trigger name	like not like	Specify a string in the trigger name or a string to exclude. **like** - event is generated by a trigger, containing this string in the name. Case sensitive. **not like** - this string cannot be found in the trigger name. Case sensitive. *Note*: Entered value will be compared to trigger name with all macros expanded.
Trigger severity	= <> >= <=	Specify trigger severity. **=** - equal to trigger severity **<>** - not equal to trigger severity **>=** - more or equal to trigger severity **<=** - less or equal to trigger severity
Time period	in not in	Specify a time period or a time period to exclude. **in** - event time is within the time period. **not in** - event time is not within the time period. See the time period specification page for description of the format. User macros are supported, since Zabbix 3.4.0.

Condition type	Supported operators	Description
Maintenance status	in not in	Specify a host in maintenance or not in maintenance. **in** - host is in maintenance mode. **not in** - host is not in maintenance mode. *Note*: If several hosts are involved in the trigger expression, the condition matches if at least one of the hosts is/is not in maintenance mode.

可以为基于发现的事件设置以下条件：

The following conditions can be set for discovery-based events:

条件类型支持的	作说明	
Host IP	= <>	指定要发现的主机的IP地址范围或要排除的范围. **=** -主机IP在该范围内. **<>** -主机IP不在该范围内. 它可能有以下格式： 单IP：192.168.1.33 IP地址范围：192.168.1-10.1-254 IP mask: 192.168.4.0/24 List: 192.168.1.1-

条件类型支持的	作说明	
Service type	= <>	指定已发现服务的服务类型或者要排除的服务类型. **=** - 匹配发现的服务. **<>** -与发现的服务不匹配. 可用服务类型：SSH, LDAP, SMTP, FTP, HTTP, HTTPS *(available since Zabbix 2.2 version)*, POP, NNTP, IMAP, TCP,

条件类型支持的	作说明
Service port	= <> 指定发现的服务或的TCP端口范围或者要排除的TCP端口范围. **=** - 服务端口在该范围内. **<>** - 服务端口不在该范围内.

条件类型支持的	作说明	
Discovery rule	= <>	指定发现规则或要排除的发现规则. **=** - 使用这个发现规则. **<>** - 使用除此之外的任何其他发现规则.

条件类型支持的	作说明	
Discovery check	= <>	指定 dis-cov-ery check 或要排除的 dis-cov-ery check **=** - 使用这个 dis-cov-ery check. **<>** -使用除此之外的其他任何 dis-cov-ery check.

条件类型支持的	作说明	
Discovery object	=	指定发现的对象. **=** - 等于发现的对象(设备或服务). **Up** -匹配’Host Up’和’Service Up’事件 **Down** - 匹配’Host Down’和’Service Down’事件 **Discovered** - 匹配’Host Dis-cov-ered’和’Service Dis-cov-ered’事件 **Lost** - 匹配’Host Lost’和’Service Lost’事件.
Discovery status	=	

条件类型支持的	作说明	
Uptime/Downtime	>= <=	Uptime for 'Host Up' and 'Service Up' events. Downtime for 'Host Down' and 'Service Down' events. **>=** - 大于或者等于. 参数以秒为单位给出。 **<=** - 小于或等于. 参数以秒为单位给出.

条件类型支持的	作说明	
Received value	= <> >= <= like not like	指定从代理接收的值(Zabbix，SNMP)。区分大小写字符串比较.如果为规则配置了多个Zabbix代理或SNMP检查，则检查所有的Zabbix代理或SNMP检查(每个检查生成与所有条

条件类型支持的	作说明	
Proxy	= <>	指定代理或要排除的代理. **=** - 使用这个代理. **<>** -使用除此之外的任何其他代理.

Condition type	Supported operators	Description
Host IP	= <>	Specify an IP address range or a range to exclude for a discovered host. **=** - host IP is in the range. **<>** - host IP is not in the range. It may have the following formats: Single IP: 192.168.1.33 Range of IP addresses: 192.168.1-10.1-254 IP mask: 192.168.4.0/24 List: 192.168.1.1-254, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 Support for spaces in the list format is provided since Zabbix 3.0.0.

Condition type	Supported operators	Description
Service type	= <>	Specify a service type of a discovered service or a service type to exclude. **=** - matches the discovered service. **<>** - does not match the discovered service. Available service types: SSH, LDAP, SMTP, FTP, HTTP, HTTPS *(available since Zabbix 2.2 version)*, POP, NNTP, IMAP, TCP, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping, telnet *(available since Zabbix 2.2 version)*.
Service port	= <>	Specify a TCP port range of a discovered service or a range to exclude. **=** - service port is in the range. **<>** - service port is not in the range.
Discovery rule	= <>	Specify a discovery rule or a discovery rule to exclude. **=** - using this discovery rule. **<>** - using any other discovery rule, except this one.
Discovery check	= <>	Specify a discovery check or a discovery check to exclude. **=** - using this discovery check. **<>** - using any other discovery check, except this one.
Discovery object	=	Specify the discovered object. **=** - equal to discovered object (a device or a service).
Discovery status	=	**Up** - matches 'Host Up' and 'Service Up' events **Down** - matches 'Host Down' and 'Service Down' events **Discovered** - matches 'Host Discovered' and 'Service Discovered' events **Lost** - matches 'Host Lost' and 'Service Lost' events
Uptime/Downtime	>= <=	Uptime for 'Host Up' and 'Service Up' events. Downtime for 'Host Down' and 'Service Down' events. **>=** - is more or equal to. Parameter is given in seconds. **<=** - is less or equal to. Parameter is given in seconds.

Condition type	Supported operators	Description
Received value	= <> >= <= like not like	Specify the value received from an agent (Zabbix, SNMP) check in a discovery rule. Case sensitive string comparison. If several Zabbix agent or SNMP checks are configured for a rule, received values for each of them are checked (each check generates a new event which is matched against all conditions). **=** - equal to the value. **<>** - not equal to the value. **>=** - more or equal to the value. **<=** - less or equal to the value. **like** - contains the substring. Parameter is given as a string. **not like** - does not contain the substring. Parameter is given as a string.
Proxy	= <>	Specify a proxy or a proxy to exclude. **=** - using this proxy. **<>** - using any other proxy except this one.

Note:
在发现规则里的服务检测，并不是同时发生的。因此, 如果动作里 **multiple** 值配置了 `Service type`, `Service port` or `Received value` ，他们被视为一次一个发现事件，而不是多个事件。这样，有多个值的同类检测的动作，可能错误地执行。

Note:
Service checks in a discovery rule, which result in discovery events, do not take place simultaneously. Therefore, if **multiple** values are configured for `Service type`, `Service port` or `Received value` conditions in the action, they will be compared to one discovery event at a time, but **not** to several events simultaneously. As a result, actions with multiple values for the same check types may not be executed correctly.

基于活动代理自动注册的动作可以设置以下条件：

The following conditions can be set for actions based on active agent auto-registration:

条件类型支持的	作说明	
Host metadata	like not like	指定主机元数据或要排除的主机元数据. **like** - 主机元数据包含字符串. **not like** - 主机元数据不包含字符串. 可以在代理配置文件中指定主机元数据.

条件类型支持的	作说明	
Host name	like not like	指定主机名或要排除的主机名. **like** - 包含字符串的主机名. **not like** - 不包含字符串的主机名.

条件类型支持的	作说明	
Proxy	= <>	指定代理或要排除的代理. **=** - 使用这个代理. **<>** - 使用除此之外的任何其他代理.

Condition type	Supported operators	Description
Host metadata	like not like	Specify host metadata or host metadata to exclude. **like** - host metadata contains the string. **not like** - host metadata does not contain the string. Host metadata can be specified in an agent configuration file.
Host name	like not like	Specify a host name or a host name to exclude. **like** - host name contains the string. **not like** - host name does not contain the string.
Proxy	= <>	Specify a proxy or a proxy to exclude. **=** - using this proxy. **<>** - using any other proxy except this one.

可以根据内部事件为动作设置以下条件：

The following conditions can be set for actions based on internal events:

Application	= like not like	指定应用程序或要排除的应用程序. **=** - 属于与指定应用程序链接的项的事件. **like** - 属于与包含字符串的应用程序链接的项的事件. **not like** - 属于不包含字符串的应用程序链接的项的事件.
Event type	=	**Item in "not supported" state** - 匹配监控项从 "正常" 到 "不支持" 状态的事件 **Low-level discovery rule in "not supported" state** - 匹配低级发现规则从 "正常" 到 "不支持" 状态的事件 **Trigger in "unknown" state** - 匹配触发从 "正常" 到 "未知" 状态的事件
Host group	= <>	指定主机组或要排除的主机组. **=** - 属于此主机组的事件. **<>** - 不属于此主机组的事件.
Template	= <>	指定模板或要排除的模板. **=** -属于从此模板继承的监控项/触发器/低级发现规则的事件. **<>** - 不属于从此模板继承的监控项/触发器/低级发现规则的事件.
Host	= <>	指定主机或要排除的主机. **=** - 属于这个主机的事件. **<>** - 不属于这个主机的事件.

Condition type	Supported operators	Description
Application	= like not like	Specify an application or an application to exclude. **=** - event belongs to an item that is linked to the specified application. **like** - event belongs to an item that is linked to an application containing the string. **not like** - event belongs to an item that is linked to an application not containing the string.
Event type	=	**Item in "not supported" state** - matches events where an item goes from a 'normal' to 'not supported' state **Low-level discovery rule in "not supported" state** - matches events where a low-level discovery rule goes from a 'normal' to 'not supported' state **Trigger in "unknown" state** - matches events where a trigger goes from a 'normal' to 'unknown' state

Condition type	Supported operators	Description
Host group	= <>	Specify host groups or host groups to exclude. **=** - event belongs to this host group. **<>** - event does not belong to this host group.
Template	= <>	Specify templates or templates to exclude. **=** - event belongs to an item/trigger/low-level discovery rule inherited from this template. **<>** - event does not belong to an item/trigger/low-level discovery rule inherited from this template.
Host	= <>	Specify hosts or hosts to exclude. **=** - event belongs to this host. **<>** - event does not belong to this host.

运算类型

Type of calculation

计算条件的以下选项可用：

The following options of calculating conditions are available:

- **And** - 必须满足所有条件
- **And** - all conditions must be met

请注意，当它们被选为 “Trigger =” “条件时，在几个触发器之间不允许使用”And“计算。操作只能根据一个触发事件执行。

Note that using ”And” calculation is disallowed between several triggers when they are selected as a `Trigger=` condition. Actions can only be executed based on the event of one trigger.

- **Or** - 如果满足一个条件就足够了
- **And/Or** - 两者的组合：AND 与不同的条件类型和 OR 具有相同的条件类型，例如：

- **Or** - enough if one condition is met
- **And/Or** - combination of the two: AND with different condition types and OR with the same condition type, for example:

Host group = Oracle servers
Host group = MySQL servers
Trigger name like ’Database is down’
Trigger name like ’Database is unavailable’

Host group = Oracle servers
Host group = MySQL servers
Trigger name like ’Database is down’
Trigger name like ’Database is unavailable’

计算为

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’**)**

- 自定义表达式 - 用户自己定义的动作条件表达式. 必须包含所有的条件 (以大写字母 A, B, C, ...) 以及可能包含空格, 制表符, 括号, 和大小写敏感字符。
- **Custom expression** - a user-defined calculation formula for evaluating action conditions. It must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), **and** (case sensitive), **or** (case sensitive).

当前例包含 And/Or，例如 (A or B) and (C or D), 还可以有等价的其它表达写法:

While the previous example with And/Or would be represented as (A or B) and (C or D), in a custom expression you may as well have multiple other ways of calculation:

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

由于删除对象，被禁用的 actions

Actions disabled due to deleted objects

如果某个操作条件/操作中使用的某个对象（主机，模板，触发器等）被删除，则会删除条件/操作，禁用该操作以避免操作的错误执行。该操作可以由用户重新启用。

If a certain object (host, template, trigger, etc) used in an action condition/operation is deleted, the condition/operation is removed and the action is disabled to avoid incorrect execution of the action. The action can be re-enabled by the user.

当删除以下项目是会发生这种情况：

This behavior takes place when deleting:

- 主机组（“主机组” 条件，特定主机组上的 “远程命令” 操作)）；
- 主机（“主机” 条件，特定主机上的 “远程命令” 操作）；
- 模板（“模板” 条件，“链接到模板” 和 “与模板的链接” 操作）；
- 触发器（“触发” 条件）；
- 发现规则（使用 “发现规则” 和 “发现检查” 条件时）；
- 代理（“代理” 条件)。

- host groups (”host group” condition, ”remote command” operation on a specific host group);
- hosts (”host” condition, ”remote command” operation on a specific host);
- templates (”template” condition, ”link to template” and ”unlink from template” operations);
- triggers (”trigger” condition);
- discovery rules (when using ”discovery rule” and ”discovery check” conditions);
- proxies (”proxy” condition).

Note: 如果远程命令有许多目标主机，并且我们删除其中的一个，则只有该主机将从目标列表中删除，操作本身将保留。但是，如果它是唯一的主机，操作也将被删除。“链接到模板” 和 “取消与模板的链接” 操作也是一样。

Note: If a remote command has many target hosts, and we delete one of them, only this host will be removed from the target list, the operation itself will remain. But, if it’s the only host, the operation will be removed, too. The same goes for ”link to template” and ”unlink from template” operations.

删除 “发送消息” 操作中使用的用户或用户组时，操作不会被禁用。

Actions are not disabled when deleting a user or user group used in a ”send message” operation.

2 操作

2 Operations

概述

Overview

您可以为所有事件定义以下操作：

You can define the following operations for all events:

- 发送信息
- 执行远程命令 (包括 IPMI)

- send a message
- execute a remote command (including IPMI)

<note important> 如果用户被明确地设置了主机动作 (action) 和操作 (opertion) 权限为”denied” 或用户根本没有该主机的访问权限, 那么 Zabbix server 并不产生告警。:::

Attention:
Zabbix server does not create alerts if access to the host is explicitly ”denied” for the user defined as action operation recipient or if the user has no rights defined to the host at all.

对于发现事件，还有其他操作可用：

For discovery and auto-registration events, there are additional operations available:

- 添加主机
- 删除主机
- 启用主机
- 禁用主机
- 添加到主机群组
- 从主机群组中删除
- 链接到模板
- 取消与模板的链接
- 设置主机资产清单

- add host
- remove host
- enable host
- disable host
- add to host group
- remove from host group
- link to template
- unlink from template
- set host inventory mode

配置操作

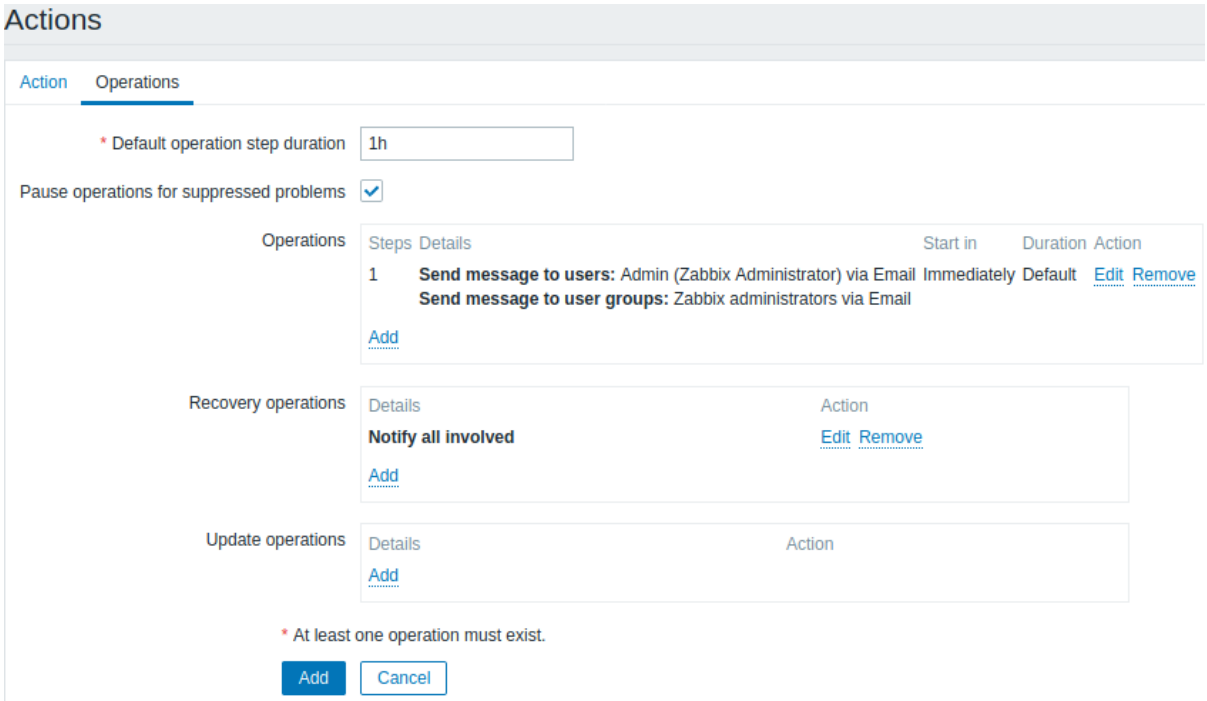
Configuring an operation

要配置操作，进入 action 配置 中的 操作选项卡，然后单击操作块中的 *New*。编辑操作步骤，然后单击 *Add* 添加到 *Operations* 列表中。

To configure an operation, go to the *Operations* tab in action configuration and click on *New* in the Operations block. Edit the operation step and click on *Add* to add to the list of *Operations*.

操作属性：

Operation attributes:



All mandatory input fields are marked with a red asterisk.

参数	说
Default operation step duration	一个操作步骤默认持续时间(最少60秒）. 例如，一小时的持续时间意味着如果执行操作，则在下一步之前将经过一小时.
Default subject	默认消息主题为通知. 主题可能包含宏.
Default message	通知的默认消息. 消息可能包含宏.
Pause operations while in maintenance	标记此复选框以延长维护期间的操作.\\如果取消选中此复选框，即使在维护期间，操作也将立即执行. Zabbix 3.2.0之后支持此选项.

参数	说
Operations	显示操作操作，具体如下： **Steps** - 分配操作的升级步骤 **Details** - 操作类型及其收件人/目标. 自 Zabbix 2.2 以来，操作列表还显示了发送消息中使用的媒体类型（电子邮件，SMS，Jabber 等）以及通知收件人的名称和姓氏（在别名之后的括号中） **Start in** -执行操作后的事件多长时间 **Duration (sec)** - 显示步长。如果步骤使用默认持续时间，则显示 *Default* 如果使用自定义持续时间，则显示时间。 **Action** - 显示用于编辑和删除操作的链接。要配置新操作，

参数		说
Operation details		此块用于配置操作的详细信息.
	Steps	在升级计划表中选择步骤分配操作: **From** - 从这一步开始执行 **To** - 执行到此步骤 (0=infinity, 执行不会受到限制)
	Step duration	这些步骤的自定义持续时间 (0 = 使用默认步骤持续时间). 几个操作可以分配到同一步骤. 如果这些操作具有不同的步长定义，则考虑最短的步骤并将其应用于该步骤.

参数		说
	Operation type	所有事件都有两种操作类型：**Send message** - 发送消息给用户 **Remote command** - 执行远程命令 更多的操作可用于发现和基于自动注册的事件(见上文）.
	操作类型：发信息	
	Send to user groups	点击 *Add* 选择要发送消息的用户组. 用户组必须至少具有“读取”权限以获得主机的通知.
	Send to users	点击 *Add* 选择要发送消息的用户. 用户组必须至少具有“读取”权限 以获得主机的通知.
	Send only to	发送消息到所有定义的媒体类型或只选一个.

参数	说	
	Default mes-sage	如果选择, 将使用默认消息(见上文）.
	Subject	自定义消息的主题. 主题可能包含宏.
	Message	自定义消息. 消息可能包含宏。
	操作类型：远程命令	

参数		说
	Target list	选择要执行命令的目标： **Current host** - 命令在导致问题事件的触发器的主机上执行. 如果触发器中有多个主机, 则此选项将无法正常工作. **Host** - 选择主机以执行命令. **Host group** -选择主机组以执行命令. 指定父主机组隐含地选择所有嵌套的主机组. 因此, 远程命令也将在嵌套组的主机上执行. 主机上的命令只能执行一次, 即使主机与多次匹配(例如来自多个主机组, 单独地和从主机组匹配）. 如果在Zabbix服务器上执行命令, 目标列表是无意义的, 在这种情

参数		说
	Type	选择命令类型：**IPMI** - 执行IPMI命令 **Custom script** - 执行一组自定义的命令 **SSH** - 执行SSH 命令 **Telnet** - 执行Telnet命令 **Global script** - 执行管理 **->** 脚本中定义的全局脚本之一.
	Execute on	在Zabbix服务器或Zabbix代理上执行自定义脚本. 要在代理上执行脚本，它必须是配置为允许来自服务器的远程命令. 如果选择“自定义脚本”作为类型，则该字段可用.

参数	说	
	Commands	输入命令. 支持的宏将根据导致事件的触发表达式进行解析. 例如，主机宏将解析为主机的触发器表达式（而不是目标列表）.
	Conditions	执行操作的条件： **Not ack** - 只有当事件未被确认时 **Ack** - 只有事件被确认时.

Parameter	Description
Default operation step duration	Duration of one operation step by default (60 seconds to 1 week). For example, an hour-long step duration means that if an operation is carried out, an hour will pass before the next step. Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0.
Default subject	Default message subject for notifications. The subject may contain macros. It is limited to 255 characters.
Default message	Default message for notifications. The message may contain macros. It is limited to certain amount of characters depending on the type of database (see Sending message for more information).
Pause operations while in maintenance	Mark this checkbox to delay the start of operations for the duration of a maintenance period. When operations are started, after the maintenance, all operations are performed including those for the events during the maintenance. If you unmark this checkbox, operations will be executed without delay even during a maintenance period. This option is supported since Zabbix 3.2.0.

Parameter			Description
Operations			Action operations are displayed, with these details: **Steps** - escalation step(s) to which the operation is assigned **Details** - type of operation and its recipient/target. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS, Jabber, etc) used in sending a message as well as the name and surname (in parentheses after the alias) of a notification recipient. **Start in** - how long after an event the operation is performed **Duration (sec)** - step duration is displayed. *Default* is displayed if the step uses default duration, and a time is displayed if custom duration is used. **Action** - links for editing and removing an operation are displayed. To configure a new operation, click on *New*.
Operation details			This block is used to configure the details of an operation.
	Steps		Select the step(s) to assign the operation to in an escalation schedule: **From** - execute starting with this step **To** - execute until this step (0=infinity, execution will not be limited)
	Step duration		Custom duration for these steps (0=use default step duration). Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. Several operations can be assigned to the same step. If these operations have different step duration defined, the shortest one is taken into account and applied to the step.
	Operation type		Two operation types are available for all events: **Send message** - send message to user **Remote command** - execute a remote command More operations are available for discovery and auto-registration based events (see above).
		Operation type: send message	
		Send to user groups	Click on *Add* to select user groups to send the message to. The user group must have at least ”read” permissions to the host in order to be notified.
		Send to users	Click on *Add* to select users to send the message to. The user must have at least ”read” permissions to the host in order to be notified.
		Send only to	Send message to all defined media types or a selected one only.
		Default message	If selected, the default message will be used (see above).

Parameter		Description
	Subject	Subject of the custom message. The subject may contain macros. It is limited to 255 characters.
	Message	The custom message. The message may contain macros. It is limited to certain amount of characters depending on the type of database (see Sending message for more information).
	Operation type: remote command	
	Target list	Select targets to execute the command on: **Current host** - command is executed on the host of the trigger that caused the problem event. This option will not work if there are multiple hosts in the trigger. **Host** - select host(s) to execute the command on. **Host group** - select host group(s) to execute the command on. Specifying a parent host group implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups. A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). The target list is meaningless if a custom script is executed on Zabbix server. Selecting more targets in this case only results in the script being executed on the server more times. Note that for global scripts, the target selection also depends on the *Host group* setting in global script configuration.
	Type	Select the command type: **IPMI** - execute an IPMI command **Custom script** - execute a custom set of commands **SSH** - execute an SSH command **Telnet** - execute a Telnet command **Global script** - execute one of the global scripts defined in *Administration→Scripts*.
	Execute on	Execute a custom script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. To execute scripts on proxy, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as *Type*.

Parameter		Description
	Commands	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
	Conditions	Condition for performing the operation: **Not ack** - only when the event is unacknowledged **Ack** - only when the event is acknowledged.

1 发送消息

1 Sending message

概述

Overview

发送消息是通知人们遇到问题的最佳方式之一。这就是为什么它是 Zabbix 提供的主要动作之一。

Sending a message is one of the best ways of notifying people about a problem. That is why it is one of the primary actions offered by Zabbix.

配置

Configuration

为了能够发送和接收 Zabbix 的通知，您必须：

- 定义 media发送消息

To be able to send and receive notifications from Zabbix you have to:

- define the media to send a message to

<note warning> 如果您想要接收如发现、代理自动注册等非触发类的事件通知, 那么在用户媒介中必须检查下默认的触发器级别 ('Not classified') configuration. :::

Warning:
The default trigger severity ('Not classified') **must be** checked in user media configuration if you want to receive notifications for non-trigger events such as discovery, active agent auto-registration or internal evens.

- 配置动作操作 向一个定义的媒体发送消息
- configure an action operation that sends a message to one of the defined media

<note important>Zabbix 仅向至少有主机 "读" 权限的用户发送事件通知。至少，该用户须可访问配置了触发器表达式的主机。:::

Attention:
Zabbix sends notifications only to those users that have at least 'read' permissions to the host that generated the event. At least one host of a trigger expression must be accessible.

您可以配置使用通知升级发送消息的自定义场景。

You can configure custom scenarios for sending messages using escalations.

要成功接收和阅读 Zabbix 的电子邮件，电子邮件服务器/客户端必须支持标准的 "SMTP/MIME 电子邮件" 格式，因为 Zabbix 发送 UTF-8 数据（如果主题仅包含 ASCII 字符，则不是 UTF-8 编码）。消息的主题和主体是 base64 编码，遵循 "SMTP/MIME 电子邮件" 格式标准。

To successfully receive and read e-mails from Zabbix, e-mail servers/clients must support standard 'SMTP/MIME e-mail' format since Zabbix sends UTF-8 data (If the subject contains ASCII characters only, it is not UTF-8 encoded.). The subject and the body of the message are base64-encoded to follow 'SMTP/MIME e-mail' format standard.

Message limit after all macros expansion is the same as message limit for Remote commands.

跟踪消息

Tracking messages

您可以在监控 - > 问题中查看发送的消息的状态。

You can view the status of messages sent in *Monitoring* → *Problems*.

在 *Actions* column 您可以看到有关所采取 actions 的汇总信息。在那里绿色的数字表示发送的消息，红色的-失败的消息。进行中表示启动了一个动作。失败通知没有成功执行任何操作。

In the *Actions* column you can see summarized information about actions taken. In there green numbers represent messages sent, red ones - failed messages. *In progress* indicates that an action is initiated. *Failed* informs that no action has executed successfully.

如果您点击活动时间查看活动详细信息，您还将看到消息动作块包含由于事件发送（或未发送）的消息的详细信息。

If you click on the event time to view event details, you will also see the *Message actions* block containing details of messages sent (or not sent) due to the event.

在报表 → 动作日志您将看到为配置操作的那些事件所采取的所有操作的详细信息。

In *Reports* → *Action log* you will see details of all actions taken for those events that have an action configured.

2 远程命令

2 Remote commands

概述

Overview

使用远程命令，您可以定义在某些情况下，监视的主机上会自动执行某个预定义的命令。

With remote commands you can define that a certain pre-defined command is automatically executed on the monitored host upon some condition.

因此，远程命令是智能主动监控的强大机制。

Thus remote commands are a powerful mechanism for smart pro-active monitoring.

在功能最明显的用途中，您可以尝试：

In the most obvious uses of the feature you can try to:

- 如果没有响应，则自动重新启动某些应用程序（Web 服务器，中间件，CRM）
- 如果不响应请求，请使用 IPMI“reboot” 命令重新启动一些远程服务器
- 如果磁盘空间不足，可自动释放磁盘空间（删除较旧的文件，清理/tmp）
- 根据 CPU 负载，将 VM 从一个物理机移植到另一个物理机
- 在 CPU（磁盘，内存，任何资源）不足的情况下，将新节点添加到云环境中

- Automatically restart some application (web server, middleware, CRM) if it does not respond
- Use IPMI ’reboot’ command to reboot some remote server if it does not answer requests
- Automatically free disk space (removing older files, cleaning /tmp) if running out of disk space
- Migrate a VM from one physical box to another depending on the CPU load
- Add new nodes to a cloud environment upon insufficient CPU (disk, memory, whatever) resources

配置远程命令的操作类似于发送消息的操作，唯一的区别是 Zabbix 将执行命令而不是发送消息。

Configuring an action for remote commands is similar to that for sending a message, the only difference being that Zabbix will execute a command instead of sending a message.

远程命令可以通过 Zabbix server, proxy 或 agent 执行。其在 Zabbix agent 上可以直接通过 Zabbix server 或 Zabbix proxy 执行。同时，在 Zabbix agent 和 Zabbix proxy 上的远程命令默认是不开启的, 通过将参数 EnableRemoteCommands 设置为 1 开启。

Remote commands can be executed by Zabbix server, proxy or agent. Remote commands on Zabbix agent can be executed directly by Zabbix server or through Zabbix proxy. Both on Zabbix agent and Zabbix proxy remote commands are disabled by default. They can be enabled by setting the EnableRemoteCommands parameter to ’1’.

远程命令限制为 255 个字符。可以通过将多个命令放置在新行上来执行多个命令。远程命令可能包含宏。

Remote command limit after all macros expansion depends on the type of database and character set (non- ASCII characters require more than one byte to be stored):

- 如果没有响应，则自动重新启动某些应用程序（Web 服务器，中间件，CRM）
- 如果不响应请求，请使用 IPMI“reboot” 命令重新启动一些远程服务器
- 如果磁盘空间不足，可自动释放磁盘空间（删除较旧的文件，清理/ tmp）

- 根据 CPU 负载，将 VM 从一个物理盒移植到另一个物理盒
- 在 CPU（磁盘，内存，任何资源）不足的情况下，将新节点添加到云环境中

Database	//Limit in characters //	//Limit in bytes //
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited
IBM DB2	2048	2048
SQLite (only Zabbix proxy)	65535	not limited

配置远程命令的操作类似于发送消息的操作，唯一的区别是 Zabbix 将执行命令而不是发送消息。

Remote commands executed by Zabbix server are run as described in Command execution including exit code checking.

即使目标主机处于维护状态，也会执行远程命令。

Remote commands are executed even if the target host is in maintenance.

以下教程提供了有关如何设置远程命令的分步说明。

The following tutorial provides step-by-step instructions on how to set up remote commands.

配置

Configuration

在 Zabbix 代理（自定义脚本）上执行的那些远程命令必须首先在相应的命令中启用zabbix_agentd.conf.

Those remote commands that are executed on Zabbix agent (custom scripts) must be first enabled in the respective zabbix_agentd.conf.

确保 ** EnableRemoteCommands 参数设置为 1 ** 并取消注释。如果更改此参数，请重新启动代理守护程序。

Make sure that the **EnableRemoteCommands** parameter is set to **1** and uncommented. Restart agent daemon if changing this parameter.

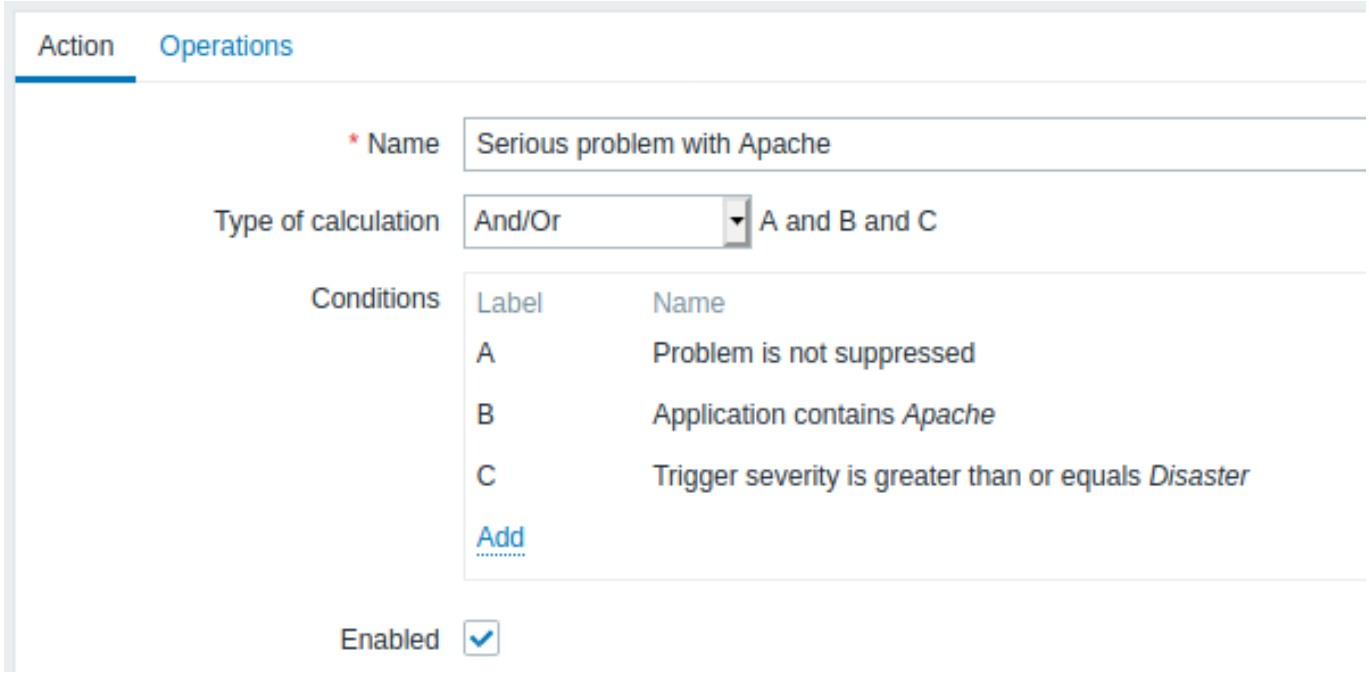
Attention:
远程命令不适用于主动模式 Zabbix 代理。

Attention:
Remote commands do not work with active Zabbix agents.

然后，在配置新的动作时进入配置 - > 操作:

Then, when configuring a new action in *Configuration* → *Actions*:

- 定义适当的条件。在此示例中，设置在 Apache 应用程序之一的任何灾难问题时激活该操作：
- Define the appropriate conditions. In this example, set that the action is activated upon any disaster problems with one of Apache applications:



All mandatory input fields are marked with a red asterisk.

- 在操作选项卡中，选择远程命令操作类型
- 选择远程命令类型（IPMI，自定义脚本，SSH，Telnet，全局脚本）
- 输入远程命令

- In the *Operations* tab, select the **Remote command** operation type
- Select the remote command type (IPMI, Custom script, SSH, Telnet, Global script)
- If *Custom script* type is selected choose the way how custom script will be executed (by Zabbix agent, Zabbix server (proxy) or Zabbix server only)
- Enter the remote command

例如：

```
sudo /etc/init.d/apache restart
```

在这种情况下，Zabbix 将尝试重新启动 Apache 进程。使用此命令，确保该命令在 Zabbix 代理上执行 (点击// Zabbix 代理按钮执行//)。

For example:

```
sudo /etc/init.d/apache restart
```

In this case, Zabbix will try to restart an Apache process. With this command, make sure that the command is executed on Zabbix agent (click the *Zabbix agent* button against *Execute on*).

Attention:
Note the use of **sudo** - 默认情况下，Zabbix 用户没有权限重新启动系统服务。有关如何配置 ** sudo ** 的提示，请参见下文。

Attention:
Note the use of **sudo** - Zabbix user does not have permissions to restart system services by default. See below for hints on how to configure **sudo**.

Note:
Zabbix 代理应在远程主机上运行并接受传入连接。Zabbix 代理在后台执行命令。

Note:
Zabbix agent should run on the remote host and accept incoming connections. Zabbix agent executes commands in background.

Zabbix 代理程序上的远程命令由系统无延迟执行。运行 [，nowait] 键，不检查执行结果。在 Zabbix 服务器上，远程命令是在 zabbix_server 的 TrapperTimeout 参数中设置的超时执行的。conf 文件被检查 以执行结果。

Remote commands on Zabbix agent are executed without timeout by the system.run[,nowait] key and are not checked for execution results. On Zabbix server remote commands are executed with timeout as set in the TrapperTimeout parameter of zabbix_server.conf file and are checked for execution results.

访问权限

确保'zabbix' 用户具有已配置命令的执行权限。可能有兴趣使用 ** sudo ** 来访问特权命令。要配置访问，请以 root 身份执行：

```
# visudo
```

可以在 *sudoers* 文件中使用的行:

```
# 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
```

Access permissions

Make sure that the 'zabbix' user has execute permissions for configured commands. One may be interested in using **sudo** to give access to privileged commands. To configure access, execute as root:

```
# visudo
```

Example lines that could be used in *sudoers* file:

```
# allows 'zabbix' user to run all commands without password.
zabbix ALL=NOPASSWD: ALL

# allows 'zabbix' user to restart apache without password.
zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```

<note tip> 在某些系统上 *sudoers* 文件将阻止非本地用户执行命令。若要修改，在*/etc/sudoers* 中添加 **requiretty** 注释。:::

> **Note:**
> On some systems *sudoers* file will prevent non-local users from executing commands. To change this, comment out **requiretty** option in */etc/sudoers*.

具有多个接口的远程命令

Remote commands with multiple interfaces

如果目标系统具有所选类型的多个接口（Zabbix 代理或 IPMI），则将在默认接口上执行远程命令。

If the target system has multiple interfaces of the selected type (Zabbix agent or IPMI), remote commands will be executed on the default interface.

可以通过 SSH 和 Telnet 使用除 Zabbix 代理之外的其他界面执行远程命令。可用的使用界面按以下顺序选择：

It is possible to execute remote commands via SSH and Telnet using another interface than the Zabbix agent one. The available interface to use is selected in the following order:

```
  * Zabbix agent default interface
  * SNMP default interface
  * JMX default interface
  * IPMI default interface

  * Zabbix agent default interface
  * SNMP default interface
  * JMX default interface
  * IPMI default interface
```

对于 IPMI 远程命令，应使用以下语法：

```
<command> [<value>]
```

where

- <command> - one of IPMI commands without spaces
- <value> - 'on', 'off' or any unsigned integer. <value> is an optional parameter.

IPMI remote commands

For IPMI remote commands the following syntax should be used:

`<command> [<value>]`

where

- <command> - one of IPMI commands without spaces
- <value> - 'on', 'off' or any unsigned integer. <value> is an optional parameter.

示例

Examples

示例 1

Example 1

在一定条件下重新启动 Windows。

Restart of Windows on certain condition.

为了在 Zabbix 检测到问题时自动重新启动 Windows，请定义以下操作：

In order to automatically restart Windows upon a problem detected by Zabbix, define the following actions:

参数说	
Operation type	'Remote command'
Type	'Custom script'
Command	c:\windows\system32\shutdown.exe -r -f

PARAMETER	Description
Operation type	'Remote command'
Type	'Custom script'
Command	c:\windows\system32\shutdown.exe -r -f

示例 2

Example 2

使用 IPMI 控制重新启动主机。

Restart the host by using IPMI control.

参数说	
Operation type	'Remote command'
Type	'IPMI'
Command	reset

PARAMETER	Description
Operation type	'Remote command'
Type	'IPMI'
Command	reset

示例 3

Example 3

使用 IPMI 控制关闭主机电源。

Power off the host by using IPMI control.

参数说	
Operation type	’Remote command’
Type	’IPMI’
Command	power off

PARAMETER	Description
Operation type	’Remote command’
Type	’IPMI’
Command	power off

3 附加操作

3 Additional operations

概述

Overview

在本章节，你可以从additional operations中找到发现/自动注册事件一些详细信息。

In this section you may find some details of additional operations for discovery/auto-registration events.

添加主机

Adding host

主机在发现过程中被添加，而不是发现过程结束。

Hosts are added during the discovery process, as soon as a host is discovered, rather than at the end of the discovery process.

<note tip> 当初期配置了较多的主机/服务时，网络发现可能需要一些时间，建议您耐心等待或配置调整合理 IP 地址范围。:::

Note:
As network discovery can take some time due to many unavailable hosts/services having patience and using reasonable IP ranges is advisable.

添加主机时，其名称由标准 ** gethostbyname ** 函数决定。如果可以解析主机，则使用解析名称。如果没有，则使用 IP 地址。此外，如果 IPv6 地址必须用于主机名，则所有 “：”（冒号）将被替换为 “_”（下划线），因为主机名中不允许冒号。

When adding a host, its name is decided by the standard **gethostbyname** function. If the host can be resolved, resolved name is used. If not, the IP address is used. Besides, if IPv6 address must be used for a host name, then all ”:” (colons) are replaced by ”_” (underscores), since colons are not allowed in host names.

<note important> 如果执行代理发现，当前主机名查找仍然发生在 Zabbix 服务器上。:::

Attention:
If performing discovery by a proxy, currently hostname lookup still takes place on Zabbix server.

<note important> 如果一个主机已经存在于与新发现的 Zabbix 配置中相同名称的主机上，1.8 之前的 Zabbix 版本将添加具有相同名称的另一个主机。Zabbix 1.8.1 和更高版本将 ** _ N 添加到主机名，其中 N ** 是增加数字，从 2 开始。:::

Attention:
If a host already exists in Zabbix configuration with the same name as a newly discovered one, versions of Zabbix prior to 1.8 would add another host with the same name. Zabbix 1.8.1 and later adds **_N** to the hostname, where **N** is increasing number, starting with 2.

4 在信息中使用宏

4 Using macros in messages

概述

Overview

在消息主题和消息文本中，您可以使用宏来更有效的问题报告。

In message subjects and message text you can use macros for more efficient problem reporting.

由 Zabbix 提供支持完整的宏列表 。

A full list of macros supported by Zabbix is available.

示例

Examples

这里的例子说明了如何在消息中使用宏。

Examples here illustrate how you can use macros in messages.

示例 1

Message subject:

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

Example 1

Message subject:

```
Problem: {TRIGGER.NAME}
```

收到消息后，消息主题将被替换为：

When you receive the message, the message subject will be replaced by something like:

```
zabbix.zabbix.com服务器上的处理器负载太高: PROBLEM
```

```
Problem: Processor load is too high on Zabbix server
```

示例 2

Message:

```
Processor load is: {zabbix.zabbix.com:system.cpu.load[,avg1].last()}
```

收到消息后，消息主题将被替换为：

```
Processor load is: 1.45
```

Example 2

Message:

```
Processor load is: {zabbix.zabbix.com:system.cpu.load[,avg1].last()}
```

When you receive the message, the message will be replaced by something like:

```
Processor load is: 1.45
```

示例 E 3

Message:

```
Latest value: {{HOST.HOST}:{ITEM.KEY}.last()}
MAX for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.max(900)}
MIN for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.min(900)}
```

Example 3

Message:

```
Latest value: {{HOST.HOST}:{ITEM.KEY}.last()}
MAX for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.max(900)}
MIN for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.min(900)}
```

收到消息时，消息将被替换为：

When you receive the message, the message will be replaced by something like:

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

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

示例 4

Message:

```
http://<server_ip_or_name>/zabbix/events.php?triggerid={TRIGGER.ID}&filter_set=1
```

Example 4

Message:

```
http://<server_ip_or_name>/zabbix/events.php?triggerid={TRIGGER.ID}&filter_set=1
```

收到消息时，它将包含一个有所有问题触发的事情的链接。

When you receive the message, it will contain a link to all events of the problem trigger.

示例 5

Informing about values from several hosts in a trigger expression.

Message:

```
Trigger: {TRIGGER.NAME}
Trigger expression: {TRIGGER.EXPRESSION}

1. Item value on {HOST.NAME1}: {ITEM.VALUE1} ({ITEM.NAME1})
2. Item value on {HOST.NAME2}: {ITEM.VALUE2} ({ITEM.NAME2})
```

Example 5

Informing about values from several hosts in a trigger expression.

Message:

```
Problem name: {TRIGGER.NAME}
Trigger expression: {TRIGGER.EXPRESSION}

1. Item value on {HOST.NAME1}: {ITEM.VALUE1} ({ITEM.NAME1})
2. Item value on {HOST.NAME2}: {ITEM.VALUE2} ({ITEM.NAME2})
```

收到消息时，消息将被替换为：

When you receive the message, the message will be replaced by something like:

```
Problem name: Processor load is too high on a local host
Trigger expression: {Myhost:system.cpu.load[percpu,avg1].last()}>5 | {Myotherhost:system.cpu.load[percpu,a

1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))

Problem name: Processor load is too high on a local host
Trigger expression: {Myhost:system.cpu.load[percpu,avg1].last()}>5 | {Myotherhost:system.cpu.load[percpu,a

1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))
```

示例 6

Receiving details of both the problem event and recovery event in a recovery message:

Message:

```
Problem:

Event ID: {EVENT.ID}
Event value: {EVENT.VALUE}
```

```
Event status: {EVENT.STATUS}
Event time: {EVENT.TIME}
Event date: {EVENT.DATE}
Event age: {EVENT.AGE}
Event acknowledgement: {EVENT.ACK.STATUS}
Event acknowledgement history: {EVENT.ACK.HISTORY}

Recovery:

Event ID: {EVENT.RECOVERY.ID}
Event value: {EVENT.RECOVERY.VALUE}
Event status: {EVENT.RECOVERY.STATUS}
Event time: {EVENT.RECOVERY.TIME}
Event date: {EVENT.RECOVERY.DATE}
```

Example 6

Receiving details of both the problem event and recovery event in a recovery message:

Message:

```
Problem:

Event ID: {EVENT.ID}
Event value: {EVENT.VALUE}
Event status: {EVENT.STATUS}
Event time: {EVENT.TIME}
Event date: {EVENT.DATE}
Event age: {EVENT.AGE}
Event acknowledgement: {EVENT.ACK.STATUS}
Event update history: {EVENT.UPDATE.HISTORY}

Recovery:

Event ID: {EVENT.RECOVERY.ID}
Event value: {EVENT.RECOVERY.VALUE}
Event status: {EVENT.RECOVERY.STATUS}
Event time: {EVENT.RECOVERY.TIME}
Event date: {EVENT.RECOVERY.DATE}
```

收到消息时，消息将被替换为：

When you receive the message, the macros will be replaced by something like:

```
Problem:

Event ID: 21874
Event value: 1
Event status: PROBLEM
Event time: 13:04:30
Event date: 2018.01.02
Event age: 5m
Event acknowledgement: Yes
Event update history: 2018.01.02 13:05:51 "John Smith (Admin)"
Actions: acknowledged.

Recovery:

Event ID: 21896
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2018.01.02

Problem:
```

```
Event ID: 21874
Event value: 1
Event status: PROBLEM
Event time: 13:04:30
Event date: 2018.01.02
Event age: 5m
Event acknowledgement: Yes
Event update history: 2018.01.02 13:05:51 "John Smith (Admin)"
Actions: acknowledged.

Recovery:

Event ID: 21896
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2018.01.02
```

Attention:
Zabbix 2.2.0 之后支持把原始问题事件和恢复事件使用的通知宏分离开。

Attention:
Separate notification macros for the original problem event and recovery event are supported since Zabbix 2.2.0.

3 恢复操作

3 Recovery operations

概述

Overview

恢复操作允许在问题解决时通知您。

Recovery operations allow you to be notified when problems are resolved.

恢复操作支持消息和远程命令。恢复操作不支持通知升级 - 因为所有操作都分配到一个单独的步骤。

Both messages and remote commands are supported in recovery operations. Recovery operations do not support escalating - all operations are assigned to a single step.

使用场景

Use cases

恢复操作的一些用例如下：

Some use cases for recovery operations are as follows:

1. 通知所有通知有关问题的用户

```
   * 选择“发送恢复消息”作为操作类型
- 恢复时有多个操作：发送通知并执行远程命令
   * 添加发送消息和执行命令的操作类型
- 在外部帮助台/票务系统中打开机票，并在问题解决时将其关闭
   * 创建一个与帮助台系统通信的外部脚本
   * 创建一个操作，该操作具有执行此脚本的操作，从而打开一张票据
   * 恢复操作，使用其他参数执行此脚本并关闭故障单
   * 使用{EVENT.ID}宏来引用原始问题
```

1. Notify all users that were notified on the problem

```
   * Select 'Send recovery message' as operation type
- Have multiple operations upon recovery: send a notification and execute a remote command
   * Add operation types for sending a message and executing a command
- Open a ticket in external helpdesk/ticketing system and close it when the problem is resolved
   * Create an external script that communicates with the helpdesk system
   * Create an action having operation that executes this script and thus opens a ticket
```

```
    * Have a recovery operation that executes this script with other parameters and closes the ticket
    * Use the {EVENT.ID} macro to reference the original problem
```

配置恢复操作

Configuring a recovery operation

配置恢复操作：

To configure a recovery operation:

- 进入 action配置中的恢复操作标签
- 点击操作块中的 *New*
- 编辑操作详情并且点击 *Add*

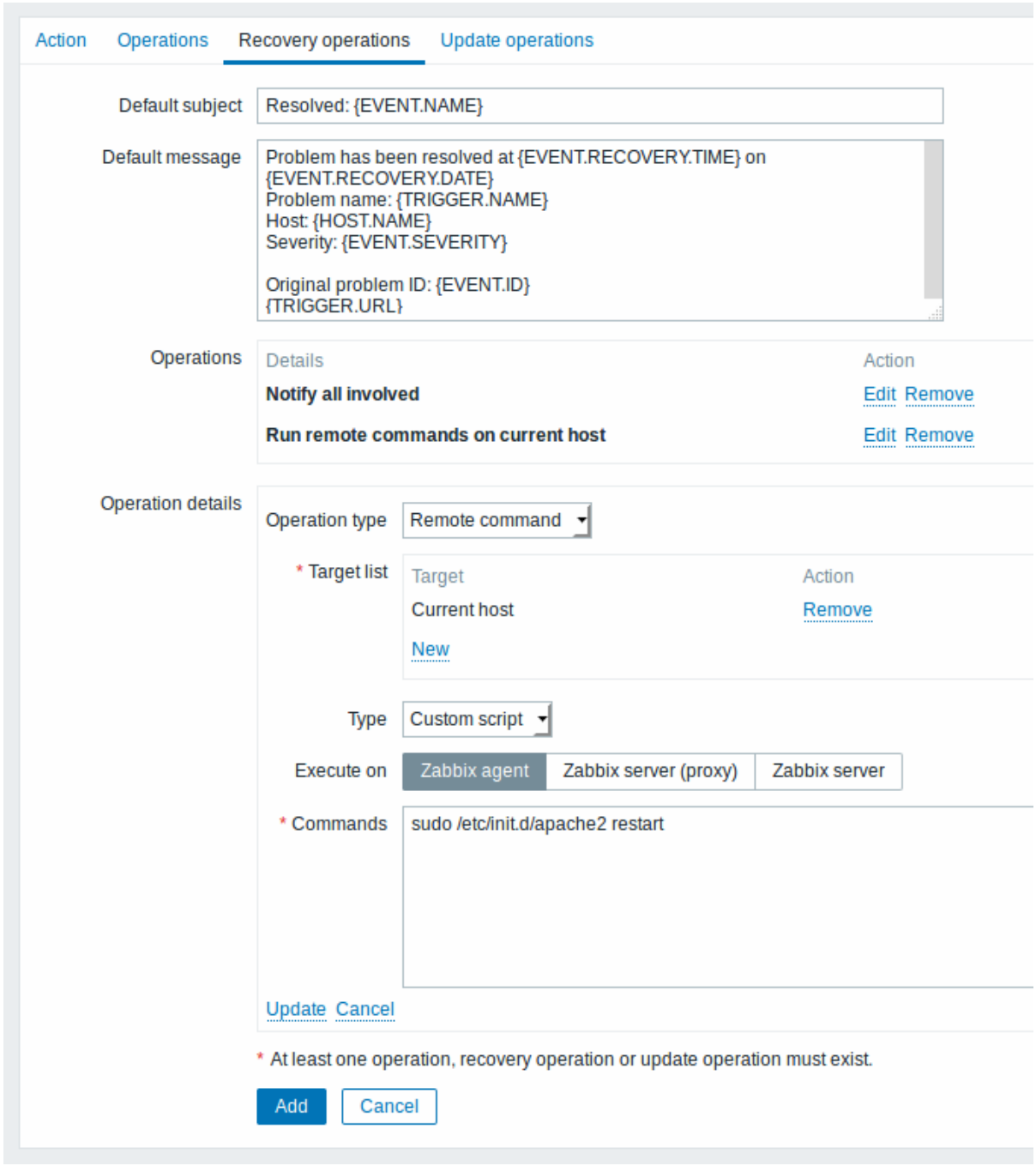
- Go to the *Recovery operations* tab in action configuration
- Click on *New* in the Operations block
- Edit the operation details and click on *Add*

以添加几个操作。

Several operations can be added.

恢复操作属性：

Recovery operation attributes:



All mandatory input fields are marked with a red asterisk.

参数	说
Default subject	恢复通知的默认消息主题. 主题可能包含宏.
Default message	恢复通知的默认消息. 消息可能包含宏.

参数	说		
Operations			恢复操作详细信息显示. 要配置新的恢复操作，请单击 *New*.
Operation details			此块用于配置恢复操作的详细信息.
	Operation type		有三种操作类型可用于恢复事件： **Send recovery message** - 所有在问题事件通知的用户发送恢复消息 **Send message** - 发送恢复信息给指定的用户 **Remote command** - 执行远程命令
		操作类型：发送恢复信息	
		Default message	如果选择，将使用默认消息(见上文）.
		Subject	自定义消息的主题. 主题可能包含宏.

参数			说	
	Message			自定义消息. 消息可能包含宏.
	操作类型：发信息			
	发送到用户组	点击	*A	d* 选择要发送恢复消息的用户组. 用户组必须至少具有“读取”权限向主机通知.
	Send to users	点击		*Add* 选择要发送恢复消息的用户组. 用户组必须至少具有“读取”” 权限 向主机通知.
	Send only to			将恢复消息发送到所有定义的媒体类型或仅选定的媒体类型.
	Default mes-sage			如果选择, 将使用默认消息（见上文）.
	Subject			自定义消息的主题. 主题可能包含宏.
	Message			自定义消息. 消息可能包含宏.

参数	说	
	Operation type: 远程命令	
	Target list	选择当前主机，其他主机或主机组作为目标执行命令.
	Type	选择命令类型：**IPMI** - 执行 IPMI 命令 **Custom script** - 执行一组自定义的命令. 您可以选择在 Zabbix 代理或 Zabbix 服务器上执行该命令。 **SSH** - 执行 SSH 命令 **Telnet** - 执行 Telnet 命令 **Global script** -执行管理 **-** > 脚本其中定义的全局脚本之一.
	Execute on	在 Zabbix 代理或 Zabbix 服务器上执行命令.
	Commands	输入命令.

Parameter			Description
Default subject			Default message subject for recovery notifications. The subject may contain macros.
Default message			Default message for recovery notifications. The message may contain macros.
Operations			Recovery operation details are displayed. To configure a new recovery operation, click on *New*.
Operation details			This block is used to configure the details of a recovery operation.
	Operation type		Three operation types are available for recovery events: **Send message** - send recovery message to specified user **Remote command** - execute a remote command **Notify all involved** - send recovery message to all users who were notified on the problem event Note that if the same recipient with unchanged default subject/message is defined in several operation types, duplicate notifications are not sent.
		Operation type: send message	
		Send to user groups	Click on *Add* to select user groups to send the recovery message to. The user group must have at least "read" permissions to the host in order to be notified.
		Send to users	Click on *Add* to select users to send the recovery message to. The user must have at least "read" permissions to the host in order to be notified.
		Send only to	Send recovery message to all defined media types or a selected one only.
		Default message	If selected, the default message will be used (see above).
		Subject	Subject of the custom message. The subject may contain macros.
		Message	The custom message. The message may contain macros.
		Operation type: remote command	

Parameter		Description
	Target list	Select targets to execute the command on: **Current host** - command is executed on the host of the trigger that caused the problem event. This option will not work if there are multiple hosts in the trigger. **Host** - select host(s) to execute the command on. **Host group** - select host group(s) to execute the command on. Specifying a parent host group implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups. A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). The target list is meaningless if the command is executed on Zabbix server. Selecting more targets in this case only results in the command being executed on the server more times. Note that for global scripts, the target selection also depends on the *Host group* setting in global script configuration.
	Type	Select the command type: **IPMI** - execute an IPMI command **Custom script** - execute a custom set of commands **SSH** - execute an SSH command **Telnet** - execute a Telnet command **Global script** - execute one of the global scripts defined in *Administration→Scripts*.
	Execute on	Execute a custom script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as *Type*.
	Commands	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
	Operation type: notify all involved	
	Default message	If selected, the default message will be used (see above).
	Subject	Subject of the custom message. The subject may contain macros.

Parameter		Description
	Message	The custom message. The message may contain macros.

5 通知升级

5 Escalations

概述

Overview

通过 Escalations，您可以创建发送通知或执行远程命令的自定义场景。

With escalations you can create custom scenarios for sending notifications or executing remote commands.

实际应用中，这意味着：

In practical terms it means that:

- 用户可以立即收到新问题通知
- 通知可以重复，直到问题解决
- 发送通知可以延时
- 通知可以升级到另一个 “较高” 的用户组
- 可以立即执行远程命令，或者长时间不解决问题

- Users can be informed about new problems immediately
- Notifications can be repeated until the problem is resolved
- Sending a notification can be delayed
- Notifications can be escalated to another ”higher” user group
- Remote commands can be executed immediately or when a problem is not resolved for a lengthy period

操作会根据升级步骤进行通知升级。每一步都有一段时间。

Actions are escalated based on the **escalation step**. Each step has a duration in time.

您可以定义默认持续时间和单个步骤的自定义持续时间。一个升级步骤的最短持续时间为 60 秒。

You can define both the default duration and a custom duration of an individual step. The minimum duration of one escalation step is 60 seconds.

您可以从任何步骤开始执行操作，例如发送通知或执行命令。第一步是立即采取行动。如果要延迟操作，可以将其分配给稍后的步骤。对于每个步骤，可以定义几个操作。

You can start actions, such as sending notifications or executing commands, from any step. Step one is for immediate actions. If you want to delay an action, you can assign it to a later step. For each step, several actions can be defined.

通知升级步骤的数量不受限制。

The number of escalation steps is not limited.

配置操作是即可定义 Escalations. Escalations 仅对问题操作支持，而不是恢复。

Escalations are defined when configuring an operation. Escalations are supported for problem operations only, not recovery.

Escalations 的其他方面

Miscellaneous aspects of escalation behaviour

让我们考虑如果一个操作包含几个升级步骤，在不同的情况下会发生什么。

Let’s consider what happens in different circumstances if an action contains several escalation steps.

情况运

在发送初始问题通知后，所涉及的主机进入维护状态取决于 action配置(/manual/onfig/notificati中设置的在维护期间暂停操作,所有剩余的升级步骤都由维护期间或延迟引起的延迟执行.维护期不能取消操作.

情况运
在时间段操作条件中定义的时间段在发送初始通知后结束执行所有剩余的升级步骤. 时间段条件不能停操作;它对于何时启动/未启动操作而不是操作具有效果。
维护过程中出现问题，维护结束后继续（未解决）取决于 action配置(/manual config/notificat中在维护期间暂停操作的设置，所有升级步骤都可以从维护结束或立即执行.
在无数据维护期间会出现问题，并在维护结束后继续（未解决）在执行所有升级步骤之前，必须等待触发器触发.

情况运
不同的升级紧随其后并重叠每个新的升级的执行取代

前的升级，但是至少一个升级步骤总是在以前的升级中执行.在针对触发器的每个事件评估创建的事件的操作中，此行为都是相关的.

情况运	
在升级过程中（如正在发送的消息），基于任何类型的事件： 发送正在发送的消息，然后再发送一条关于升级的消息. 后- 该操作被禁用 <*br*>- 该事件被删除 <*br*> 基于触发事件：<*br*>- 触发器被禁用或删除 <*br*>- 主机或项目被禁用 <*br*> 基于关于触发器的内部事件：<*br*>- 触发器被禁用或删除 <*br*> 基于关于项目/低级发现规则的内部事件：<*br*>- 该项目被禁用或删除 <*br*>- 主机被禁用	消息将在邮件正文的开头有取消文本(注意：取消邮件已取消)命名原因(例如，注意：取消升级：动作'<动作名称>'禁用)。通过这种方式，收件人被通知升级被取消，不

情况运
在升级过程中（如发送消息），删除该操作不再发送消息. 信息被记录到服务器日 文件 (从Debug Level 3=Warn-ing) 开始, 例如: `escalation cancelled: action id:334 deleted`

Situation	Behaviour
The host in question goes into maintenance after the initial problem notification is sent	Depending on the *Pause operations while in maintenance* setting in action configuration, all remaining escalation steps are executed either with a delay caused by the maintenance period or without delay. A maintenance period does not cancel operations.
*The time period defined in the **Time period** action condition ends after the initial notification is sent*	All remaining escalation steps are executed. The *Time period* condition cannot stop operations; it has effect with regard to when actions are started/not started, not operations.
A problem starts during maintenance and continues (is not resolved) after maintenance ends	Depending on the *Pause operations while in maintenance* setting in action configuration, all escalation steps are executed either from the moment maintenance ends or immediately.
A problem starts during a no-data maintenance and continues (is not resolved) after maintenance ends	It must wait for the trigger to fire, before all escalation steps are executed.
Different escalations follow in close succession and overlap	The execution of each new escalation supersedes the previous escalation, but for at least one escalation step that is always executed on the previous escalation. This behavior is relevant in actions upon events that are created with EVERY problem evaluation of the trigger.
During an escalation in progress (like a message being sent), based on any type of event: - the action is disabled - the event is deleted Based on trigger event: - the trigger is disabled or deleted - the host or item is disabled Based on internal event about triggers: - the trigger is disabled or deleted Based on internal event about items/low-level discovery rules: - the item is disabled or deleted - the host is disabled	The message in progress is sent and then one more message on the escalation is sent. The follow-up message will have the cancellation text at the beginning of the message body (*NOTE: Escalation cancelled*) naming the reason (for example, *NOTE: Escalation cancelled: action '<Action name>' disabled*). This way the recipient is informed that the escalation is cancelled and no more steps will be executed. This message is sent to all who received the notifications before. The reason of cancellation is also logged to the server log file (starting from Debug Level 3=Warning).
During an escalation in progress (like a message being sent) the action is deleted	No more messages are sent. The information is logged to the server log file (starting from Debug Level 3=Warning), for example: `escalation cancelled: action id:334 deleted`

Escalation 示例

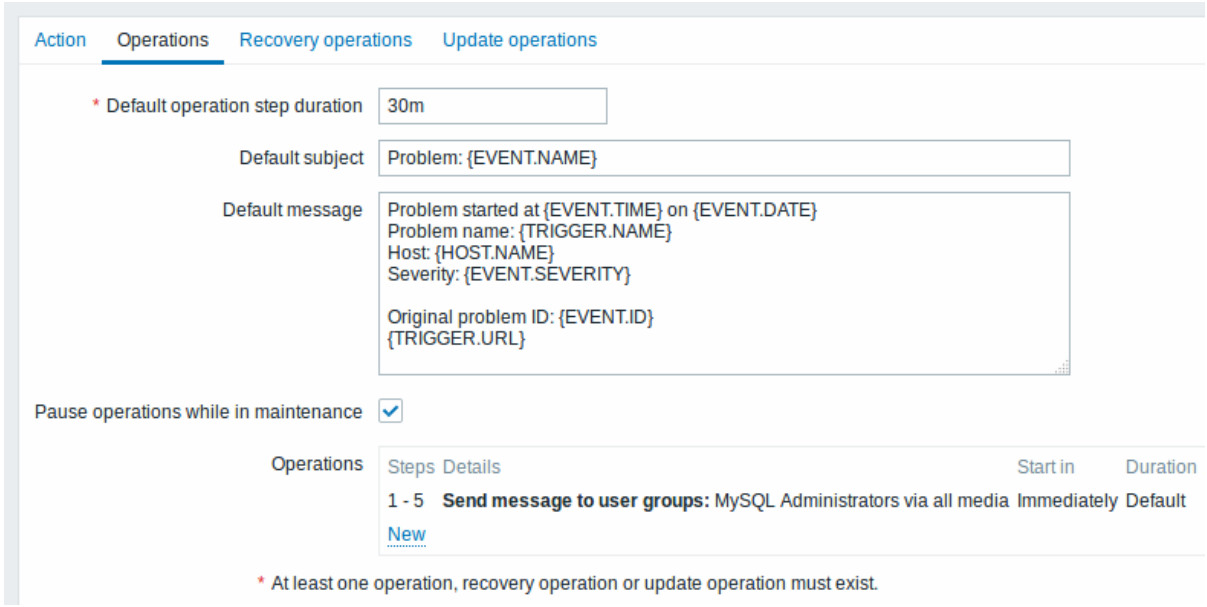
Escalation examples

示例 e 1

Example 1

Sending a repeated notification once every 30 minutes (5 times in total) to a 'MySQL Administrators' group. To configure:

- in Operations tab, set the *Default operation step duration* to '1800' seconds (30 minutes)
- Set the escalation steps to be *From* '1' *To* '5'
- Select the 'MySQL Administrators' group as recipients of the message



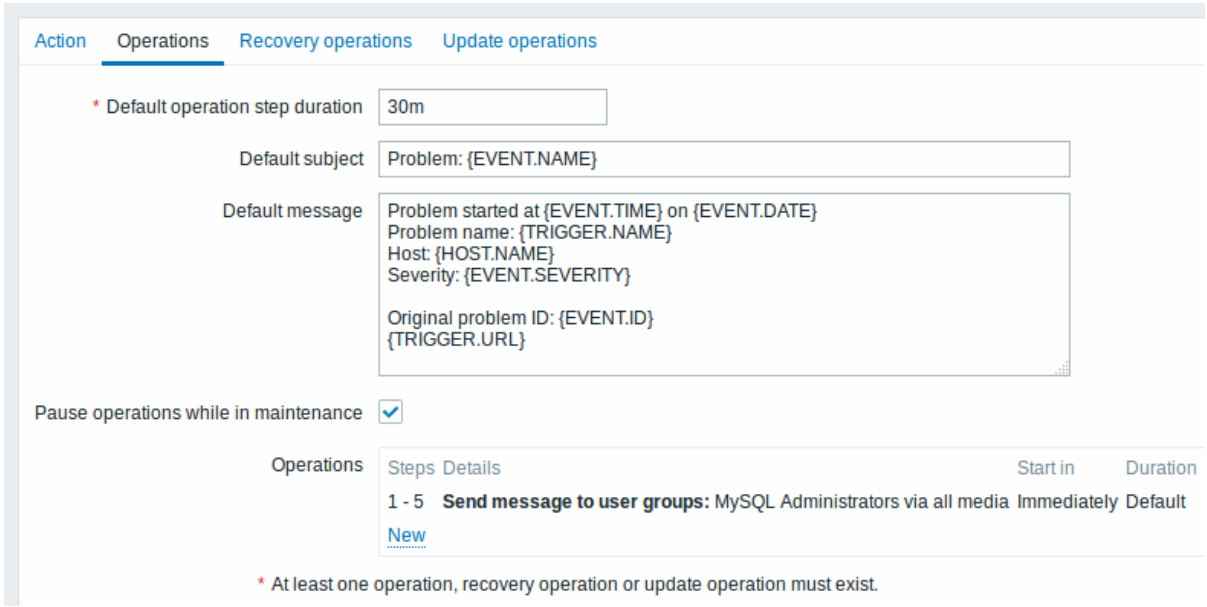
Notifications will be sent at 0:00, 0:30, 1:00, 1:30, 2:00 hours after the problem starts (unless, of course, the problem is resolved sooner).

If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

Note:
If the trigger that generated an active escalation is disabled, Zabbix sends an informative message about it to all those that have already received notifications.

Sending a repeated notification once every 30 minutes (5 times in total) to a 'MySQL Administrators' group. To configure:

- in Operations tab, set the *Default operation step duration* to '30m' (30 minutes)
- Set the escalation steps to be *From* '1' *To* '5'
- Select the 'MySQL Administrators' group as recipients of the message



All mandatory input fields are marked with a red asterisk.

Notifications will be sent at 0:00, 0:30, 1:00, 1:30, 2:00 hours after the problem starts (unless, of course, the problem is resolved sooner).

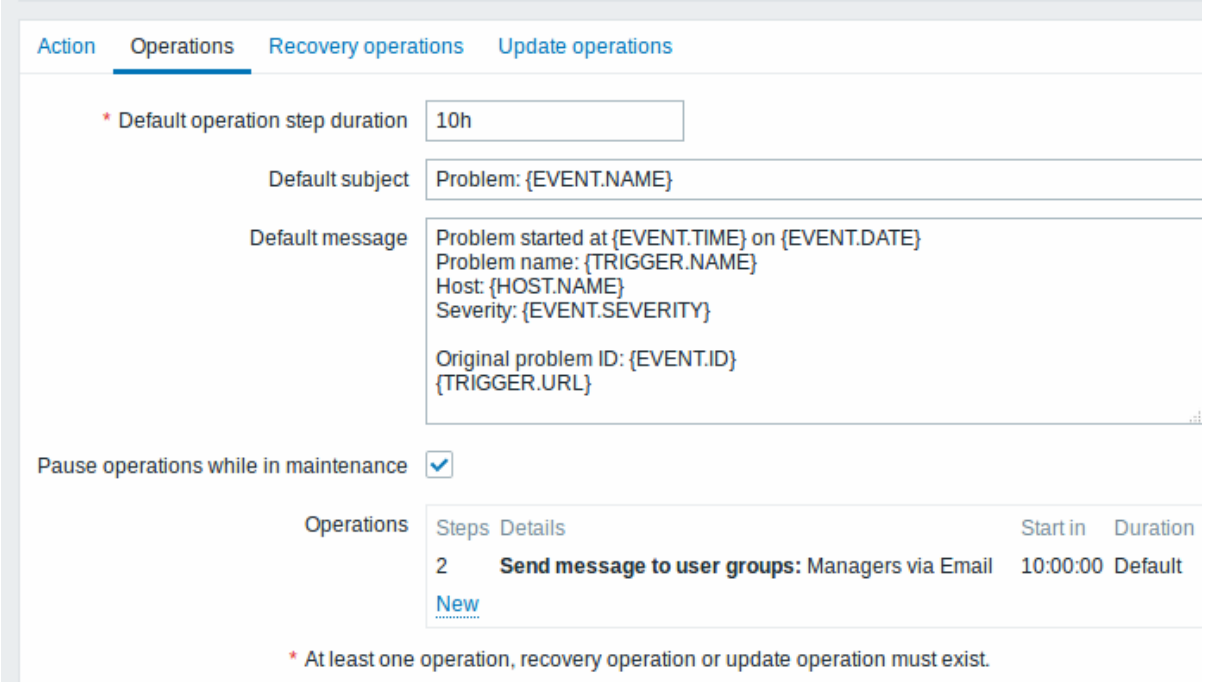
If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

Note:
If the trigger that generated an active escalation is disabled, Zabbix sends an informative message about it to all those that have already received notifications.

示例 2

Sending a delayed notification about a long-standing problem. To configure:

- In Operations tab, set the *Default operation step duration* to ’36000’ seconds (10 hours)
- Set the escalation steps to be *From* ’2’ *To* ’2’



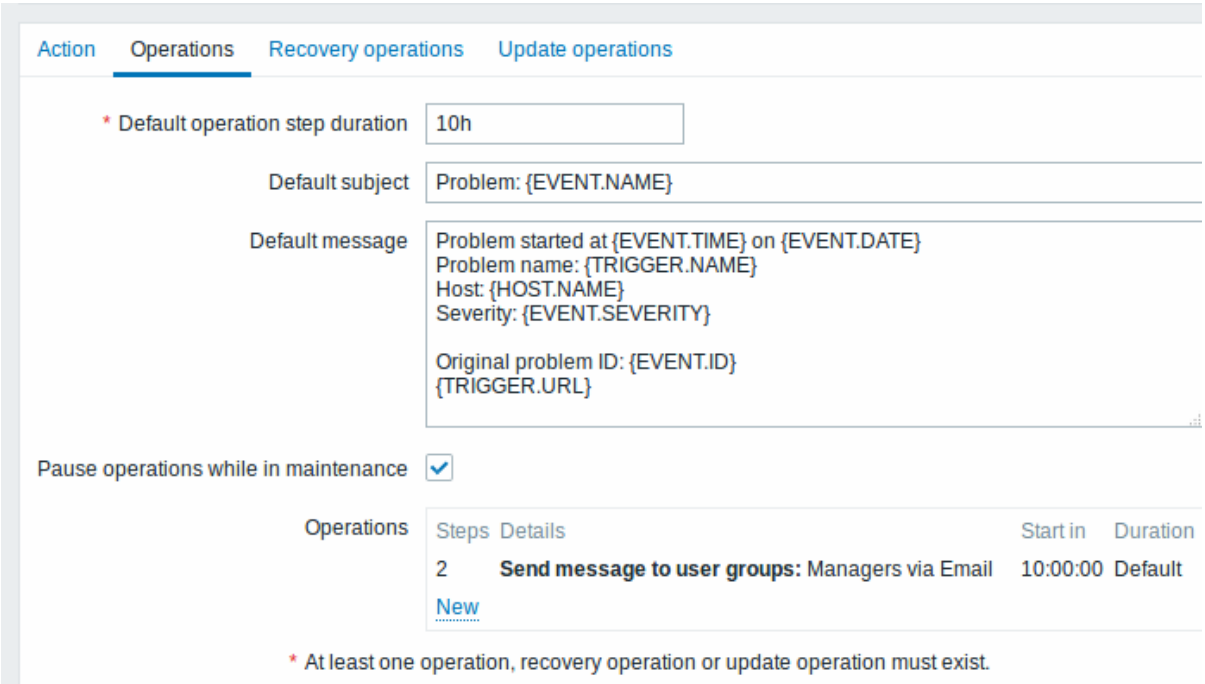
A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like ’The problem is more than 10 hours old’.

Example 2

Sending a delayed notification about a long-standing problem. To configure:

- In Operations tab, set the *Default operation step duration* to ’10h’ seconds (10 hours)
- Set the escalation steps to be *From* ’2’ *To* ’2’



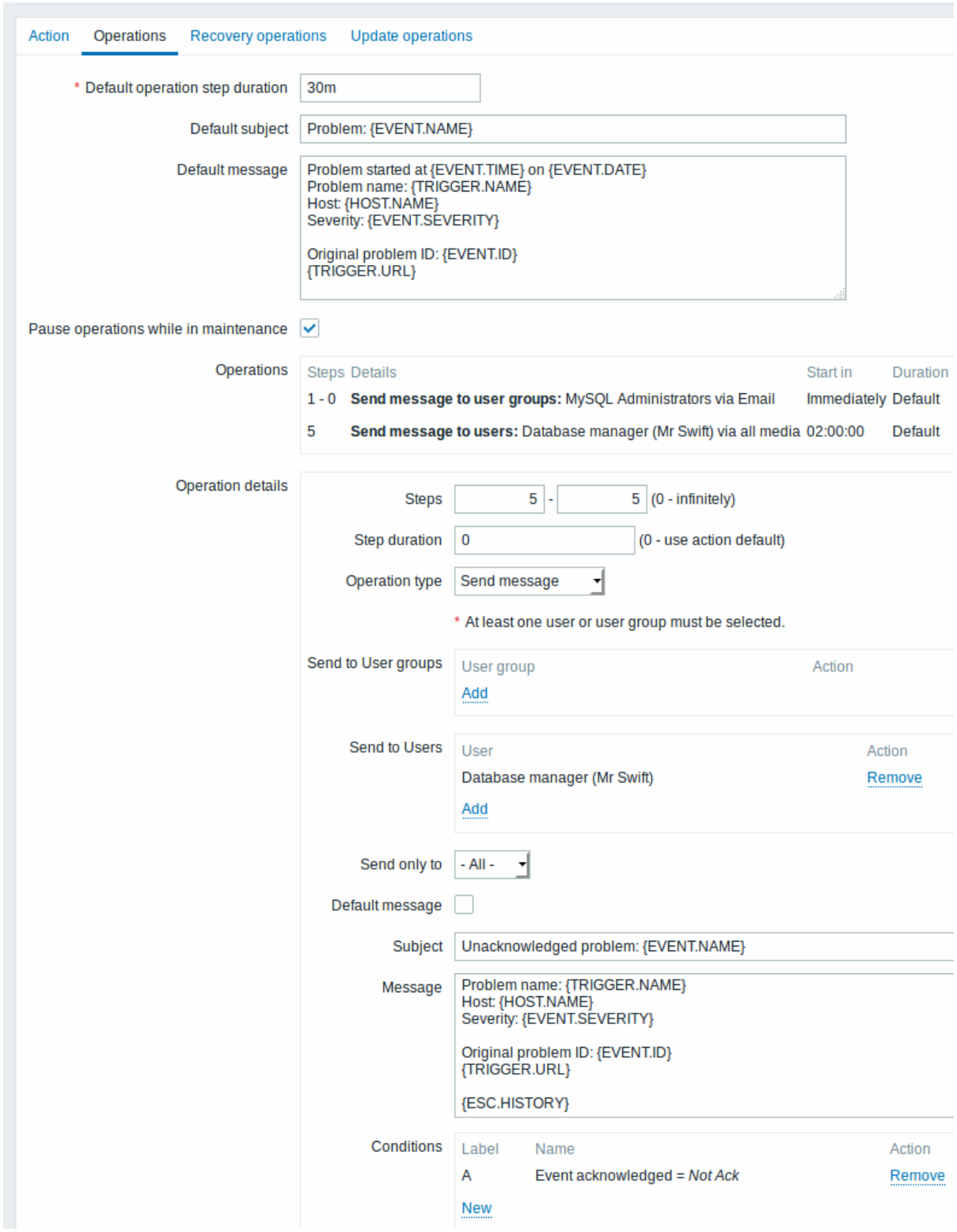
A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like 'The problem is more than 10 hours old'.

示例 3

Escalating the problem to the Boss.

In the first example above we configured periodical sending of messages to MySQL administrators. In this case, the administrators will get four messages before the problem will be escalated to the Database manager. Note that the manager will get a message only in case the problem is not acknowledged yet, supposedly no one is working on it.

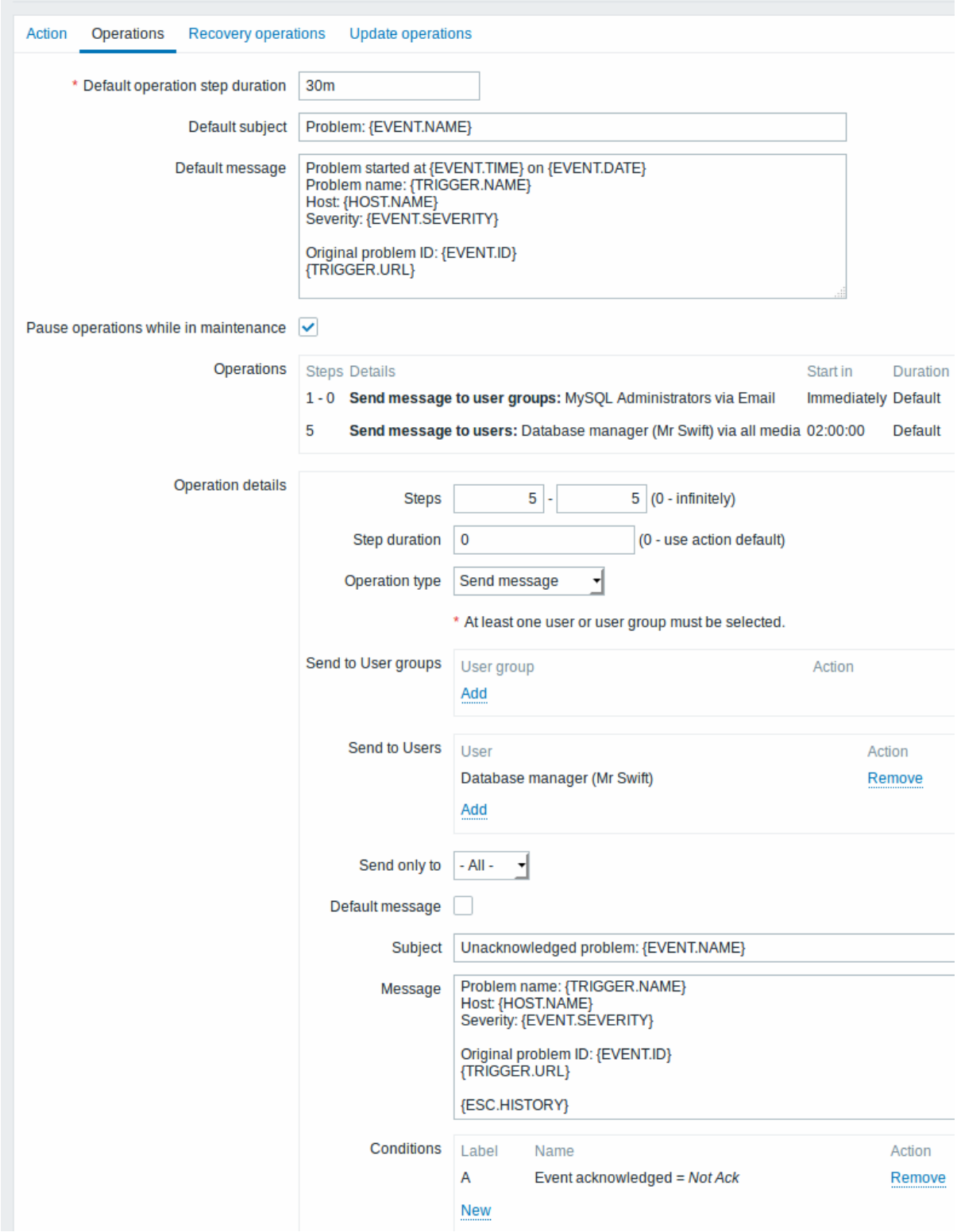


Note the use of {ESC.HISTORY} macro in the message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

Example 3

Escalating the problem to the Boss.

In the first example above we configured periodical sending of messages to MySQL administrators. In this case, the administrators will get four messages before the problem will be escalated to the Database manager. Note that the manager will get a message only in case the problem is not acknowledged yet, supposedly no one is working on it.



Note the use of {ESC.HISTORY} macro in the message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

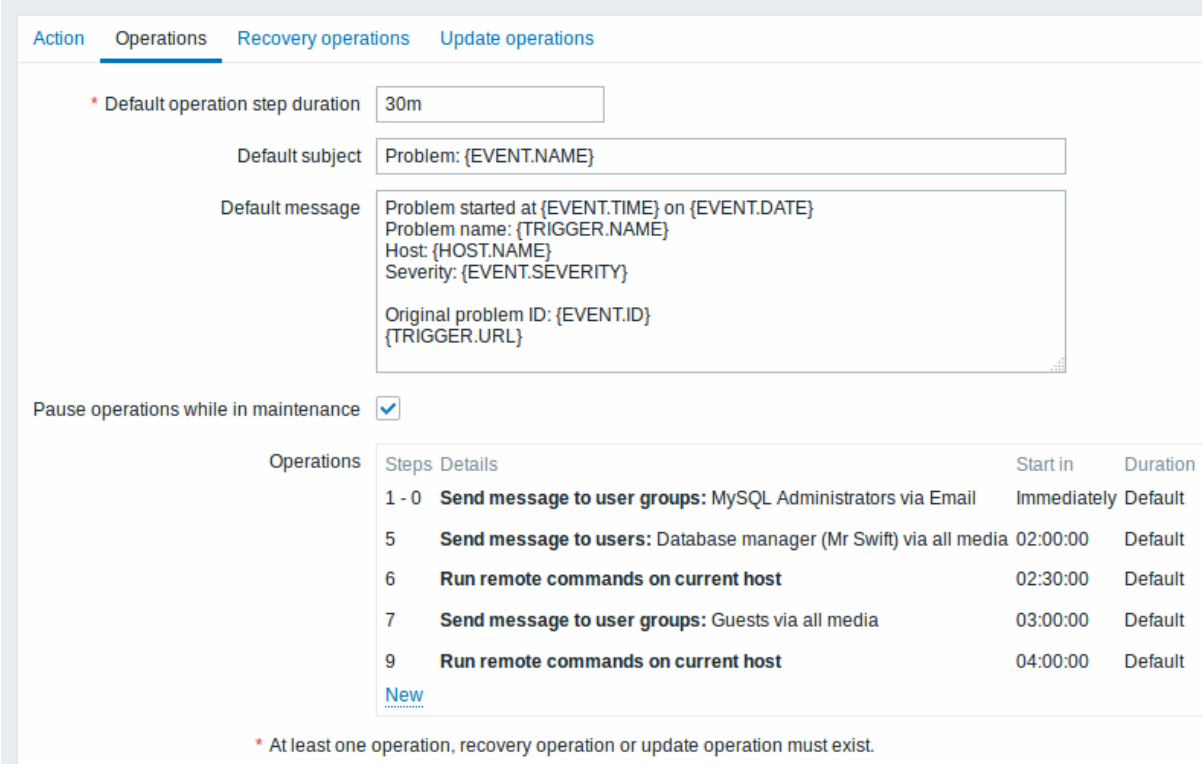
示例 4

A more complex scenario. After multiple messages to MySQL administrators and escalation to the manager, Zabbix will try to restart the MySQL database. It will happen if the problem exists for 2:30 hours and it hasn't been acknowledged.

If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

If this does not help, after another hour Zabbix will reboot server with the MySQL database (second remote command) using IPMI

commands.

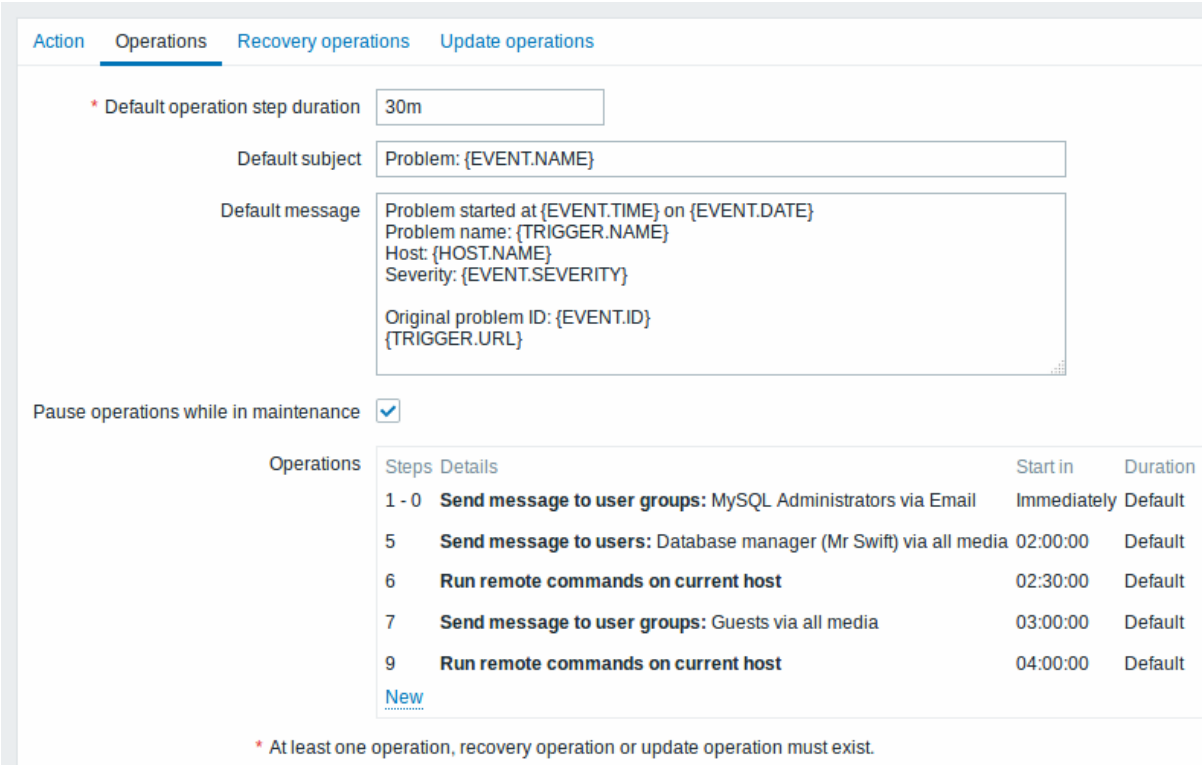


Example 4

A more complex scenario. After multiple messages to MySQL administrators and escalation to the manager, Zabbix will try to restart the MySQL database. It will happen if the problem exists for 2:30 hours and it hasn't been acknowledged.

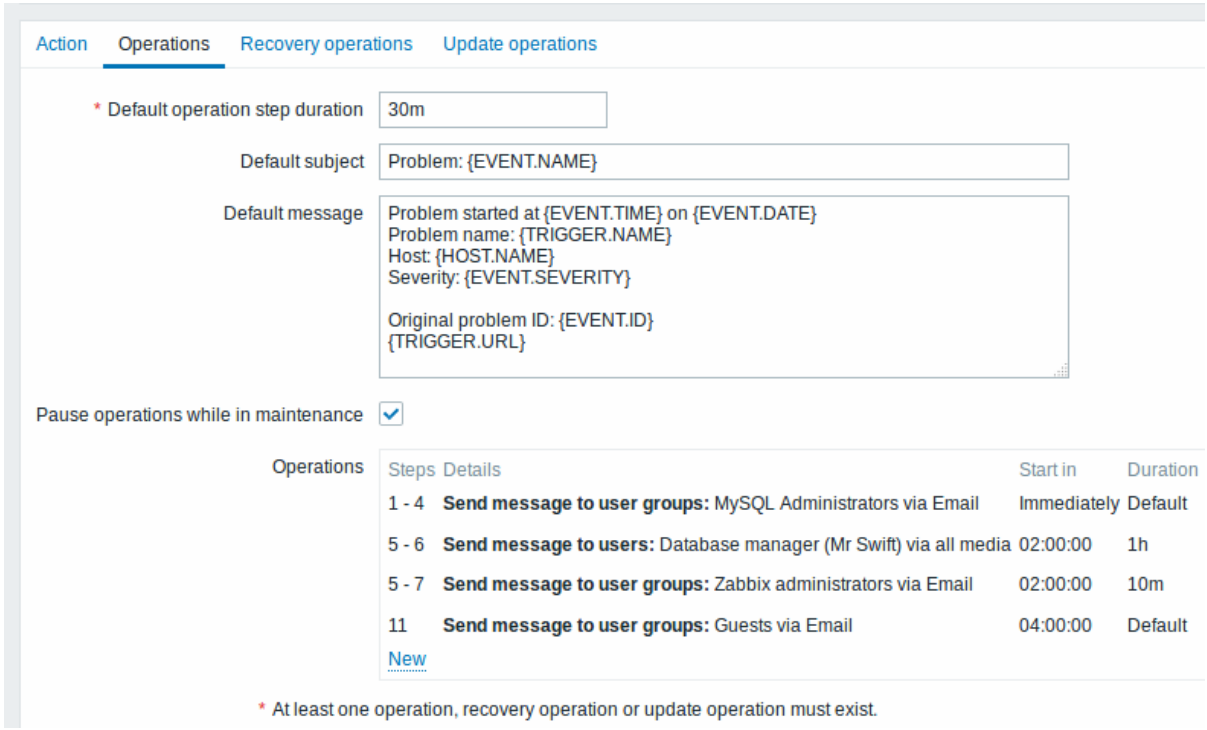
If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

If this does not help, after another hour Zabbix will reboot server with the MySQL database (second remote command) using IPMI commands.



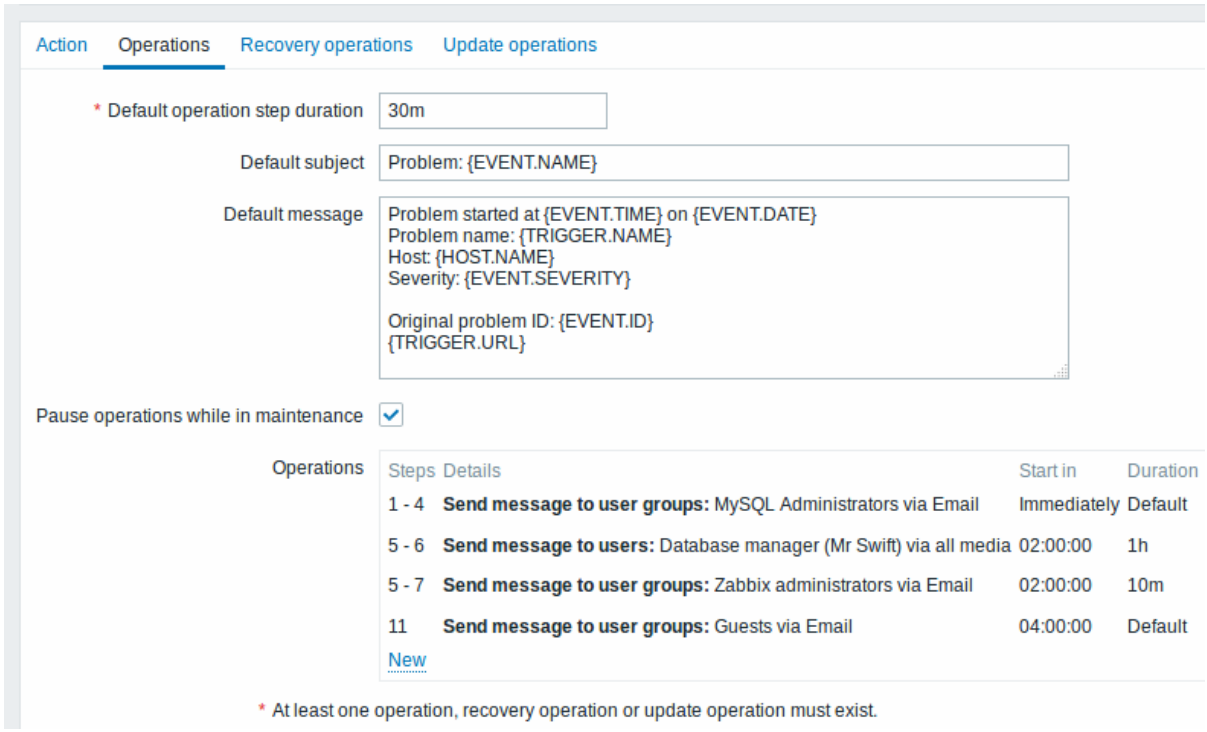
示例 5

An escalation with several operations assigned to one step and custom intervals used. The default operation step duration is 30 minutes.



Example 5

An escalation with several operations assigned to one step and custom intervals used. The default operation step duration is 30 minutes.



通知将发送如下：

Notifications will be sent as follows:

- to MySQL administrators at 0:00, 0:30, 1:00, 1:30 after the problem starts
- to Database manager at 2:00 and 2:10 (and not at 3:00; seeing that steps 5 and 6 overlap with the next operation, the shorter custom step duration of 600 seconds in the next operation overrides the longer step duration of 3600 seconds tried to set here)
- to Zabbix administrators at 2:00, 2:10, 2:20 after the problem starts (the custom step duration of 600 seconds working)
- to guest users at 4:00 hours after the problem start (the default step duration of 30 minutes returning between steps 8 and 11)

- to MySQL administrators at 0:00, 0:30, 1:00, 1:30 after the problem starts

- to Database manager at 2:00 and 2:10 (and not at 3:00; seeing that steps 5 and 6 overlap with the next operation, the shorter custom step duration of 10 minutes in the next operation overrides the longer step duration of 1 hour tried to set here)
- to Zabbix administrators at 2:00, 2:10, 2:20 after the problem starts (the custom step duration of 10 minutes working)
- to guest users at 4:00 hours after the problem start (the default step duration of 30 minutes returning between steps 8 and 11)

3 接收不受支持的项目的通知

3 Receiving notification on unsupported items 概述

Overview

Zabbix 2.2 之后支持接收不支持的项目的通知。

Receiving notifications on unsupported items is supported since Zabbix 2.2.

它是 Zabbix 内部事件概念的一部分，允许用户在这些场合获得通知。内部事件反映了状态的变化：

It is part of the concept of internal events in Zabbix, allowing users to be notified on these occasions. Internal events reflect a change of state:

- 当监控项从 “正常” 变成 “不支持”（或反之，即从 “不支持” 变成 “正常”）
- 当触发器从 “正常” 改为 “未知”（或反之，即从 “未知” 改为 “正常”）
- 当低级发现规则从 “正常” 到 “不支持”（或反之，即从 “不支持” 到 “正常”）

- when items go from ’normal’ to ’unsupported’ (and back)
- when triggers go from ’normal’ to ’unknown’ (and back)
- when low-level discovery rules go from ’normal’ to ’unsupported’ (and back)

本节介绍了当项目不受支持时接收通知的操作方法。

This section presents a how-to for **receiving notification** when an item turns unsupported.

配置

Configuration

配置一些媒介, 例如电子邮件，短信或 Jabber，用于通知。请参阅手册的相应章节执行此任务。

Overall, the process of setting up the notification should feel familiar to those who have set up alerts in Zabbix before.

步骤 1

Step 1

配置一些媒介, 例如电子邮件，短信或 Jabber，用于通知。请参阅手册的相应章节执行此任务。

Configure some media, such as e-mail, SMS or Jabber, to use for the notifications. Refer to the corresponding sections of the manual to perform this task.

<note important> 为了通知内部事件，使用默认严重性（’ 未分类’），因此如果要接收内部事件的通知，请在配置用户媒介时选中。

:::

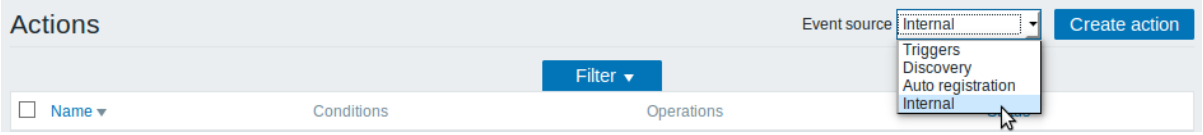
Attention:
For notifying on internal events the default severity (’Not classified’) is used, so leave it checked when configuring user media if you want to receive notifications for internal events.

步骤 2

Step 2

进入配置 - > 操作选择 内部作为事件来源. 点击右上角创建 *action* 开一个动作配置表单。

Go to *Configuration→Actions* and select *Internal* as the event source. Click on *Create action* on the upper right to open an action configuration form.

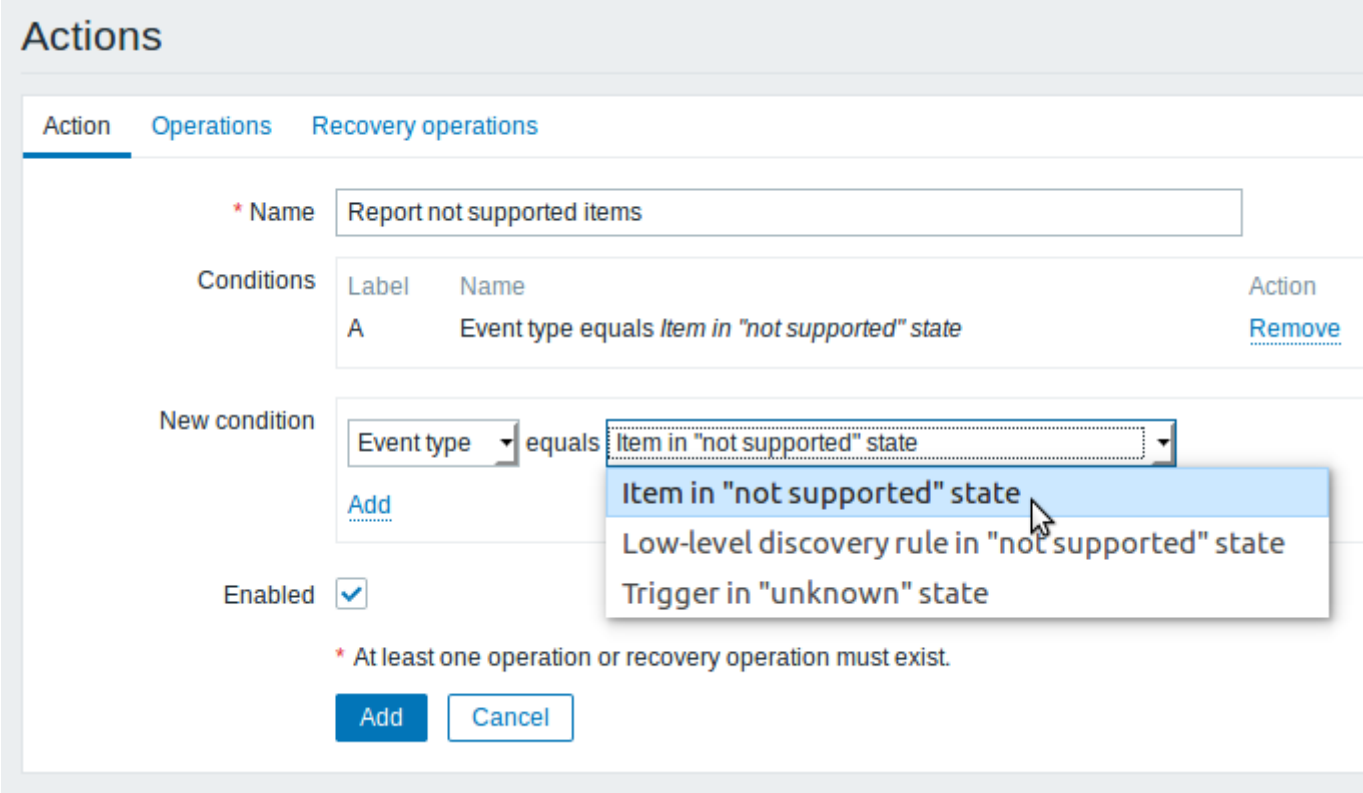


步骤 3

Step 3

在动作选项卡中输入操作的名称。然后在新条件块中选择 事件类型，选择项目处于 “不支持” 状态作为值.

In the **Action** tab enter a name for the action. Then select *Event type* in the New condition block and select *Item in "not supported" state* as the value.



不要忘记点击 *Add* 来实际列出条件块中的条件。

Don’t forget to click on *Add* to actually list the condition in the *Conditions* block.

步骤 4

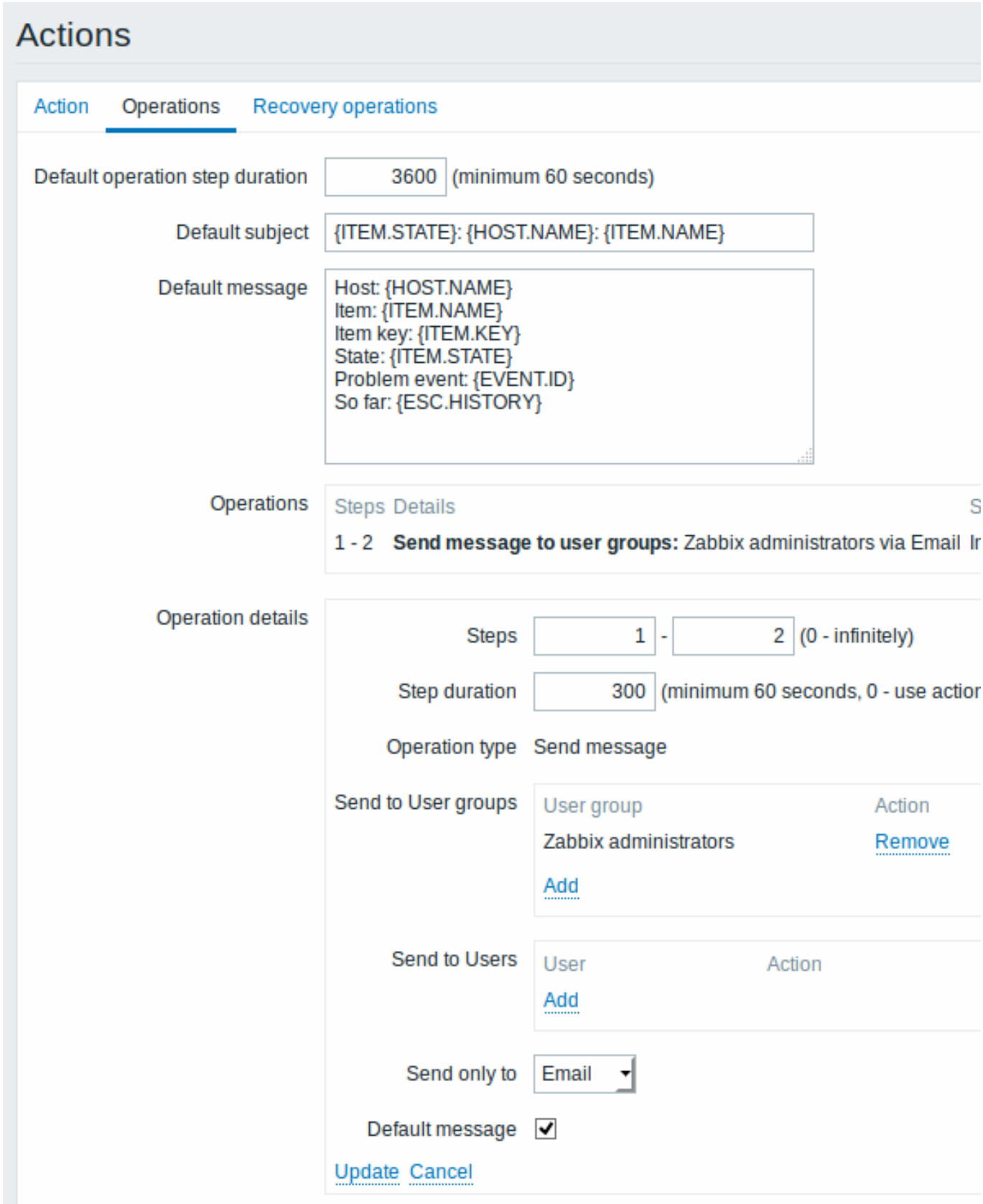
Step 4

在操作选项卡中，输入问题消息的主题/内容。

In the **Operations** tab, enter the subject/content of the problem message.

点击 操作模块中的 *New*，并且选择消息的一些接收者（用户组/用户）和用于传送的媒体类型（或 “全部”）。

Click on *New* in the *Operations* block and select some recipients of the message (user groups/users) and the media types (or ’All’) to use for delivery.



点击操作细节块中的 *Add*，添加 操作模块中实际所包含的操作。

Click on *Add* in the *Operation details* block to actually list the operation in the *Operations* block.

如果您希望收到多个通知，请设置操作步骤持续时间（发送消息之间的间隔）并添加其他操作。

If you wish to receive more than one notification, set the operation step duration (interval between messages sent) and add another operation.

步骤 5

Step 5

恢复操作选项卡允许在项目恢复到正常状态时配置恢复通知。

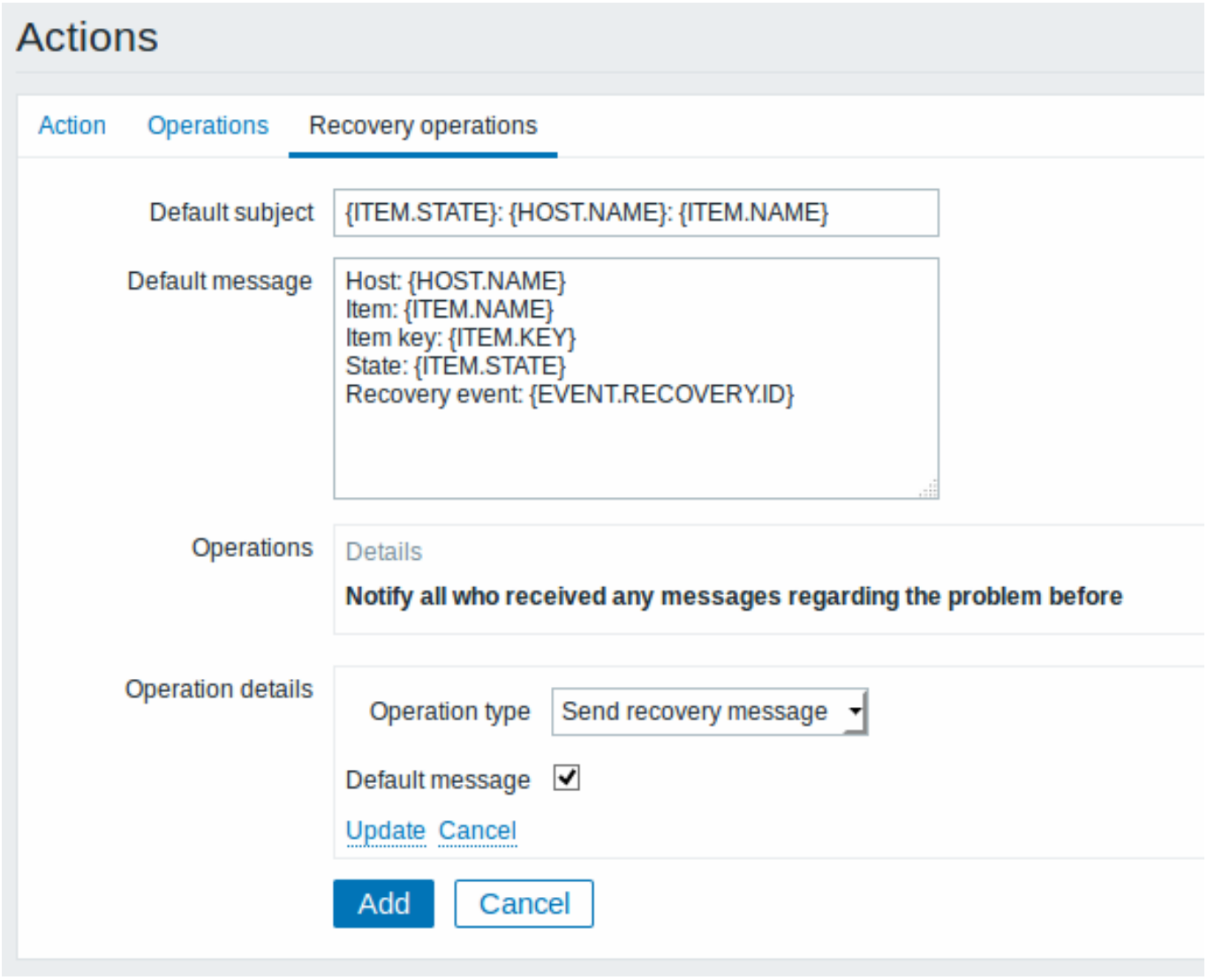
The **Recovery operations** tab allows to configure a recovery notification when an item goes back to the normal state.

输入恢复信息的主题/内容。

Enter the subject/content of the recovery message.

点击 操作模块中的 *New*，并且选择消息的一些接收者（用户组/用户）和用于传送的媒体类型（或 “全部”）。

Click on *New* in the *Operations* block and select some recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.



点击操作细节块中的 *Add*，添加 操作模块中实际所包含的操作。

Click on *Add* in the *Operation details* block to actually list the operation in the *Operations* block.

步骤 6

Step 6

完成后，单击表单下方的 “添加 **” 按钮。

When finished, click on the **Add** button underneath the form.

就这样，你完成了！现在，如果某些项目不受支持，您可以期待收到 Zabbix 的第一个通知。

And that's it, you're done! Now you can look forward to receiving your first notification from Zabbix if some item turns unsupported.

10 宏

10 Macros 概述

Overview

Zabbix 支持许多在多种情况下使用的宏。宏是一个变量，由如下特殊语法标识：

Zabbix supports a number of macros which may be used in various situations. Macros are variables, identified by a specific syntax:

```
{MACRO}
```

{MACRO}

根据在上下文中，宏解析为一个特殊的值。

Macros resolve to a specific value depending on the context.

有效地使用宏可以节省时间，并使 Zabbix 变地更加高效。

Effective use of macros allows to save time and make Zabbix configuration more transparent.

在一个的典型用途中，宏可以用于模板中。因此，模板的触发器可能命名为 “Processor load is too high on {HOST.NAME}”。当这个模板应用与主机（如 Zabbix Server ）时，并且当触发器展示在监控页面上时，触发器的名称讲解析为 “Processor load is too high on Zabbix server”。

In one of typical uses, a macro may be used in a template. Thus a trigger on a template may be named ”Processor load is too high on {HOST.NAME}”. When the template is applied to the host, such as Zabbix server, the name will resolve to ”Processor load is too high on Zabbix server” when the trigger is displayed in the Monitoring section.

宏可以在监控项键值参数中使用。宏只能用在监控项键值参数的一部分中，例如 `item.key[server_{HOST.HOST}_local]` 。双引号参数不是必须的，因为 Zabbix 将处理任何模糊不清的特殊参数（如果这些参数存在于已解析的宏中）。

Macros may be used in item key parameters. A macro may be used for only a part of the parameter, for example `item.key[server_{HOST.HOST}_local]`. Double-quoting the parameter is not necessary as Zabbix will take care of any ambiguous special symbols, if present in the resolved macro.

详细请查阅：

See also:

- 受支持的宏的完整列表；
- 宏函数；
- 如何配置用户宏。

- full list of supported macros
- macro functions
- how to configure user macros

1 宏函数

1 Macro functions

概述

Overview

宏函数能提供自定义宏值的功能。

Macro functions offer the ability to customize macro values.

有时宏可能会解析为一个不一定易于使用的值。它可能很长，或包含你想提取的一个特殊感兴趣的子字符串。这在宏函数中是可以使用的。

Sometimes a macro may resolve to a value that is not necessarily easy to work with. It may be long or contain a specific substring of interest that you would like to extract. This is where macro functions can be useful.

宏函数的语法为：

The syntax of a macro function is:

```
{<macro>.<func>(<params>)}
```

{<macro>.<func>(<params>)}

其中：

where:

- <macro> - 这个参数为要定义的宏（例如 {ITEM.VALUE}）；
- <func> - 要应用的函数；
- <params> - 以逗号分隔的函数参数列表。如果他们以” ”(空格)，" 或者包含 ），，这些符号开始，则参数必须要引用。

- <macro> - the macro to customize (for example {ITEM.VALUE})
- <func> - the function to apply
- <params> - a comma-delimited list of function parameters. Parameters must be quoted if they start with ” ” (space), " or contain ），，.

例如：

For example:

```
{{ITEM.VALUE}.regsub(pattern, output)}
```


```
{{ITEM.VALUE}.regsub(pattern, output)}
```


受支持的宏函数

Supported macro functions

函数	描述 *	参数 ** ** 受	持于 **
regsub (<pattern>,<output>)	通过正则表达式匹配提取的子字符串（区分大小写）。**pattern** - 匹配的正则表达式	{ITEM.VALUE} **output** - 输出的选项。**\1 - \9** 占位符支持被正则表达式匹配的组 placeholders are supported for captured groups.	{ITEM.LASTVALUE} 如果参数 pattern 是一个不正确的正则表达式，那么将返回 “UNKNOWN” 。
iregsub (<pattern>,<output>)	通过正则表达式匹配提取的子字符串（区分大小写）。**pattern** - 匹配的正则表达式	{ITEM.VALUE} <.LASTVALUE} **output** - 输出得选项 **\1 - \9** placeholders are supported for captured groups {ITE	如果参数 pattern 是一个不正确的正则表达式，那么将返回 “UNKNOWN” 。

FUNCTION	Description	Parameters	Supported for
regsub (<pattern>,<output>)	Substring extraction by a regular expression match (case sensitive).	**pattern** - the regular expression to match **output** - the output options. **\1 - \9** placeholders are supported to capture groups. **\0** returns the matched text. If `pattern` is not a correct regular expression 'UNKNOWN' is returned.	{ITEM.VALUE} {ITEM.LASTVALUE}
iregsub (<pattern>,<output>)			

FUNCTION			
	Substring extraction by a regular expression match (case insensitive).	**pattern** - the regular expression to match **output** - the output options. **\1 - \9** placeholders are supported to capture groups. **\0** returns the matched text. If `pattern` is not a correct regular expression 'UNKNOWN' is returned.	{ITEM.VALUE} {ITEM.LASTVALUE}

如果在受支持的位置使用函数，但是应用于不支持宏函数得宏，那么宏的计算结果为 “UNKNOWN”。

If a function is used in a supported location, but applied to a macro not supporting macro functions, then the macro evaluates to 'UNKNOWN'.

如果在不支持宏函数的位置将宏函数应用于宏，则忽略该函数。

If a macro function is applied to the macro in locations not supporting macro functions then the function is ignored.

示例

Examples

关于宏函数可用于自定义宏值的方法，在下面的示例中说明，其中包含的 “log line” 作为接收值：

The ways in which macro functions can be used to customize macro values is illustrated in the following examples containing log lines as received value:

接收值宏	输出
`123Log line`	`{{ITEM.VALUE}.regsub(^[0-9]+ (.*),Problem)}` `Problem`
`123 Log line`	`{{ITEM.VALUE}.regsub(^([0-9]+) (.*),"Problem")}` `Problem`
`123 Log line`	`{{ITEM.VALUE}.regsub(^([0-9]+) (.*),Problem ID: \1)}` `ID: 123`

接收值宏	输出
Log line	{{ITEM.VALUE}.regsub("^(.*", "Problem ID: \1")}
MySQL crashed errno 123	{{ITEM.VALUE}.regsub("^([A-Z]+)([0-9]+)", " Problem ID: \1_\2 ")}
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", "Problem ID: \1")}

Received value	Macro	Output
123Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", Problem)}	Problem
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", "Problem")}	Problem
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", Problem ID: \1)}	Problem ID: 123
Log line	{{ITEM.VALUE}.regsub("^(Problem)", "Problem ID: \1")}	Problem ID: Log line
MySQL crashed errno 123	{{ITEM.VALUE}.regsub("^([A-Z]+)([0-9]+)", " Problem ID: \1_\2 ")}	Problem ID: MySQL_123
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", "Problem ID: \1")}	Problem ID: 123

3 自动发现（**LLD**）宏

3 Low-level discovery macros

概述

Overview

有一种自动发现（LLD）函数中使用的宏类型为:

There is a type of macro used within the low-level discovery function:

{#MACRO}

{#MACRO}

它是一个在 LLD 规则中使用的宏，并返回文件系统名称、网络接口和 SNMP OIDs。

It is a macro that is used in an LLD rule and returns real values of file system names, network interfaces and SNMP OIDs.

这些宏可以用于创建监控项、触发器和图形原型。然后，当发现真实的文件系统、网络接口等，这些宏将被替换为真实的值，并且以这些值来创建真实的监控项、触发器和图形。

These macros can be used for creating item, trigger and graph *prototypes*. Then, when discovering real file systems, network interfaces etc., these macros are substituted with real values and are the basis for creating real items, triggers and graphs.

这些宏还用于在虚拟机自动发现中创建主机和主机组原型。

These macros are also used in creating host and host group *prototypes* in virtual machine discovery.

可支持的位置

Supported locations

LLD 宏可以用在：

LLD macros can be used:

- 用于监控项原型中：
  - names
  - key parameters
  - units
  - SNMP OIDs
  - IPMI sensor fields
  - calculated item formulas
  - SSH and Telnet scripts
  - database monitoring SQL queries
  - descriptions (从 2.2.0 开始支持)
- 用于触发器原型中：
  - names
  - expressions
  - URLs (从 3.0.0 开始支持)
  - descriptions (从 2.2.0 开始支持)
  - event tag names and values (从 3.2.0 开始支持)
- 用于图形原型中：
  - names
- 用于主机原型中 (从 2.2.0 开始支持)：
  - names
  - visible names
  - host group prototype names
  - (详细查阅全部列表)
- for item prototypes in
  - names
  - key parameters
  - units
  - update intervals
  - history storage periods
  - trend storage periods
  - SNMP OIDs
  - IPMI sensor fields
  - calculated item formulas
  - SSH and Telnet scripts
  - database monitoring SQL queries
  - JMX item endpoint fields
  - descriptions
  - since Zabbix 4.0 also in:
    - item value preprocessing steps
    - HTTP agent URL field
    - HTTP agent HTTP query fields field
    - HTTP agent request body field
    - HTTP agent required status codes field
    - HTTP agent headers field key and value
    - HTTP agent HTTP authentication username field
    - HTTP agent HTTP authentication password field
    - HTTP agent HTTP proxy field
    - HTTP agent HTTP SSL certificate file field
    - HTTP agent HTTP SSL key file field
    - HTTP agent HTTP SSL key password field
    - HTTP agent HTTP timeout field
- for trigger prototypes in
  - names
  - expressions
  - URLs
  - descriptions

    - event tag names and values
- for graph prototypes in
    - names
- for host prototypes in
    - names
    - visible names
    - host group prototype names
    - (see the full list)

在上述所有位置，LLD 宏都可以在用户宏上下文中使用。

In all those places LLD macros can be used inside user macro context.

一些自动发现(LLD)宏在 Zabbix 中是已经预先内置的，例如 {#FSNAME}、{#FSTYPE}、{#IFNAME}、{#SNMPINDEX}、{#SNMPVALUE} 这些宏。然而，当你在创建自定义自动发现规则的时候，遵守这些宏名称不是强制性的。所以，你可以使用任何其他的 LLD 宏名称并引用该名称。

Some low-level discovery macros come ”pre-packaged” with the LLD function in Zabbix - {#FSNAME}, {#FSTYPE}, {#IFNAME}, {#SNMPINDEX}, {#SNMPVALUE}. However, adhering to these names is not compulsory when creating a custom low-level discovery rule. Then you may use any other LLD macro name and refer to that name.

11 用户和用户组

11 Users and user groups 概述

Overview

Zabbix 中的所有用户都通过 Web 前端去访问 Zabbix 应用程序。并为每个用户分配唯一的登陆名和密码。

All users in Zabbix access the Zabbix application through the web-based frontend. Each user is assigned a unique login name and a password.

所有用户的密码都被加密并储存于 Zabbix 数据库中。用户不能使用其用户名和密码直接登陆到 UNIX 服务器中，除非他们也被因此建立在 UNIX 中。可以使用 SSL 来保护 Web 服务器和用户浏览器之间的通讯。

All user passwords are encrypted and stored in the Zabbix database. Users cannot use their user id and password to log directly into the UNIX server unless they have also been set up accordingly to UNIX. Communication between the web server and the user browser can be protected using SSL.

使用一个灵活的用户权限架构 可以限制和区分对以下内容的访问权限：

With a flexible user permission schema you can restrict and differentiate access to:

- 管理 Zabbix 前端的功能；
- 主机组中监视的主机。

- administrative Zabbix frontend functions
- monitored hosts in hostgroups

最初 Zabbix 安装后有两个预先定义好的用户 “Admin” 和 “guest”。其中，“guest” 用户用户未经验证身份的用户。在你使用 “Admin” 登陆前，你是 “guest” 用户。继续在 Zabbix 中配置用户。

The initial Zabbix installation has two predefined users - ’Admin’ and ’guest’. The ’guest’ user is used for unauthenticated users. Before you log in as ’Admin’, you are ’guest’. Proceed to configuring a user in Zabbix.

1 配置用户

1 Configuring a user

概述

Overview

根据以下步骤来配置一个用户：

To configure a user:

- 在 Zabbix 前端页面跳转到 管理 → 用户；
- 在当前页面点击创建用户（或在用户名中编辑现有的用户）；

- 在窗口中编辑用户属性。

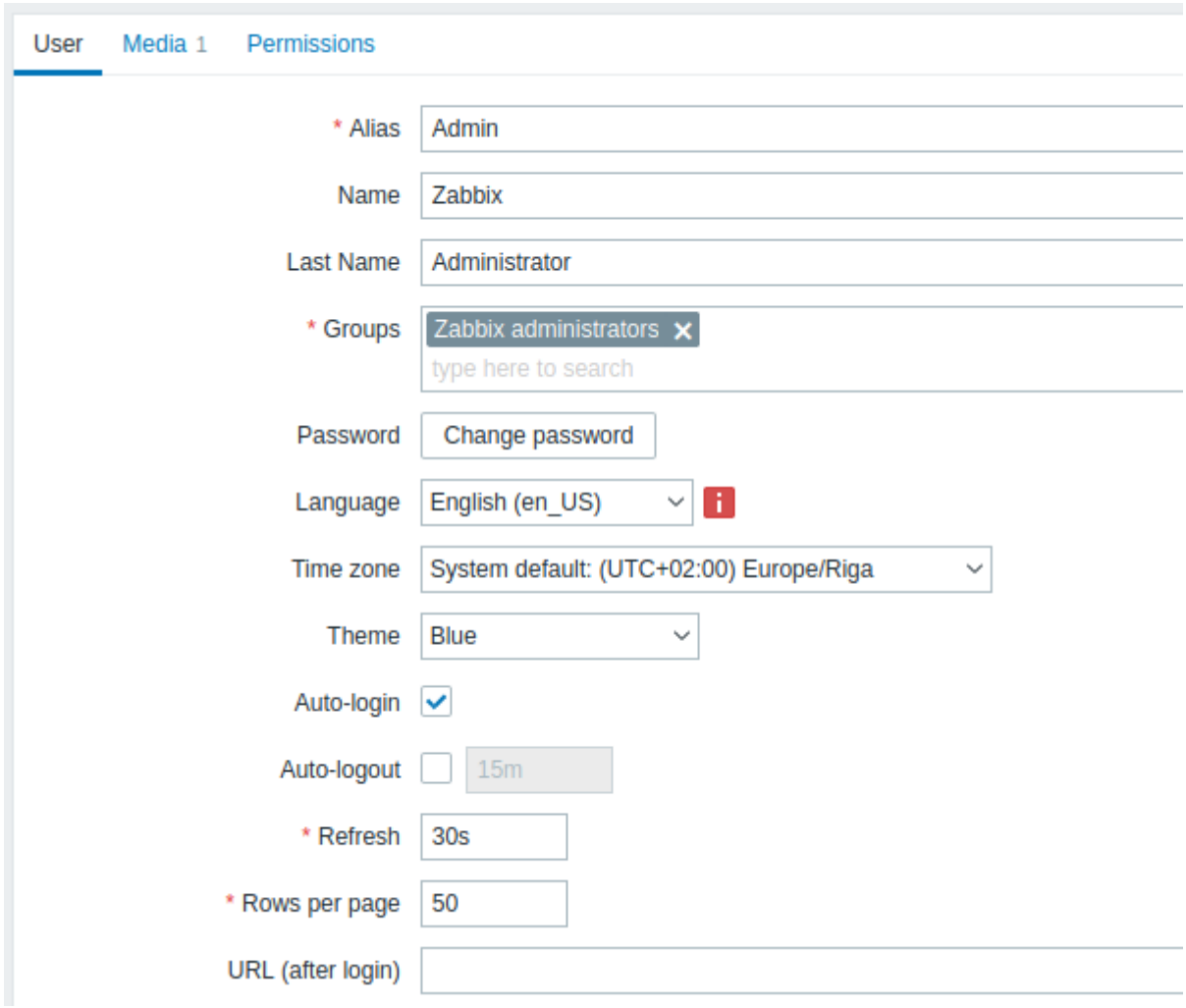
- Go to *Administration* → *Users*
- Click on *Create user* (or on the user name to edit an existing user)
- Edit user attributes in the form

常规属性

General attributes

在 用户标签页包含常规用户属性：

The *User* tab contains general user attributes:



All mandatory input fields are marked with a red asterisk.

参数描	
别名唯	的用户名，用作登陆名.

参数	描
名字用	的名字(可选的).如果此项不为空的话，则在确认信息和通知收件人信息中可见.

参数	描
姓氏用	的姓氏(可选的).如果此项不为空的话，则在确认信息和通知收件人信息中可见.

参数	描
密码输	用户密码的两个字段. With an existing password, contains a *Password* button, clicking on which opens the password fields.

参数	描
用户组用户	属用户组的列表。所属的用户组决定用户了用户可以访问的主机组和主机。点击添加进行添加用户组.
语言 Z	bbix 前端的预言.PHP 扩展插件 get-text 是翻译所必须的.

参数	描
主机定	了前端的样式：系统默认 - 使用默认的系统设置蓝 - 标准的蓝色主题深色 - 另一种深色主题

参数	描
自动登录	如果你希望 Zabbix 记住登录的信息并自动登录 30 天，请启用此选项. 此选项需要用到浏览器的 cookies.
自动登出（最少 *90* 秒）	勾选此选项以设置户在不活跃时间 (最少 90 秒) 后自动退出登录.

参数	描
刷新 *(秒)* 设置	形、聚合图形、文本数据等的刷新速率。可以设置为 0 即禁止刷新.
每页行数设置每	页面显示的行数.

参数描	
*URL (*登录后*)* 通过	置一个URL,当你登录Zab-bix后,可以跳转到此URL.例如,设置为触发器的状态页面.

Parameter	Description
Alias	Unique username, used as the login name.
Name	User first name (optional). If not empty, visible in acknowledgement information and notification recipient information.
Surname	User second name (optional). If not empty, visible in acknowledgement information and notification recipient information.
Groups	Select user groups the user belongs to. Starting with Zabbix 3.4.3 this field is auto-complete so starting to type the name of a user group will offer a dropdown of matching groups. Scroll down to select. Alternatively, click on *Select* to add groups. Click on 'x' to remove the selected. Adherence to user groups determines what host groups and hosts the user will have access to.
Password	Two fields for entering the user password. With an existing password, contains a *Password* button, clicking on which opens the password fields.
Language	Language of the Zabbix frontend. The php gettext extension is required for the translations to work.
Theme	Defines how the frontend looks like: **System default** - use default system settings **Blue** - standard blue theme **Dark** - alternative dark theme **High-contrast light** - light theme with high contrast **High-contrast dark** - dark theme with high contrast

Parameter	Description
Auto-login	Mark this checkbox to make Zabbix remember the user and log the user in automatically for 30 days. Browser cookies are used for this.
Auto-logout	With this checkbox marked the user will be logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day). Time suffixes are supported, e.g. 90s, 5m, 2h, 1d. Note that this option will not work: * If the "Show warning if Zabbix server is down" global configuration option is enabled and Zabbix frontend is kept open; * When Monitoring menu pages perform background information refreshes; * If logging in with the *Remember me for 30 days* option checked.
Refresh	Set the refresh rate used for graphs, screens, plain text data, etc. Can be set to 0 to disable.
Rows per page	You can determine how many rows per page will be displayed in lists.
URL (after login)	You can make Zabbix transfer the user to a specific URL after successful login, for example, to Problems page.

告警媒介

User media

告警媒介标签页包含用户定义的所有告警媒介。告警媒介用于发送通知。点击添加将告警媒介分配给用户。

The *Media* tab contains a listing of all media defined for the user. Media are used for sending notifications. Click on *Add* to assign media to the user.

关于配置告警媒介类型详细的信息，请参阅告警媒介类型。

See the Media types section for details on configuring media types.

权限

Permissions

权限标签页包含以下信息：

The *Permissions* tab contains information on:

- 用户类型（Zabbix User, Zabbix Admin, Zabbix Super Admin）。用户不能改变自己的用户类型。
- 用户可以访问的主机组。默认情况下,"Zabbix User" 和 "Zabbix Admin" 用户无权访问任何的主机组和主机。若要获得访问权限，需要将他们定义到访问相应主机组和主机的用户组中。

- the user type (Zabbix User, Zabbix Admin, Zabbix Super Admin). Users cannot change their own type.
- host groups the user has access to. 'Zabbix User' and 'Zabbix Admin' users do not have access to any host groups and hosts by default. To get access they need to be included in user groups that have access to respective host groups and hosts.

关于详细信息，请参阅用户权限页面。

See the User permissions page for details.

2 权限

2 Permissions

概述

Overview

您可以定义相应的用户类型，然后通过将无特权用户包含在具有访问主机组数据权限的用户组中来区分 Zabbix 中的用户权限。

You can differentiate user permissions in Zabbix by defining the respective user type and then by including the unprivileged users in user groups that have access to host group data.

用户类型

User type

用户类型定义了对前端管理菜单的访问级别以及对主机组数据的默认访问权限。

The user type defines the level of access to administrative menus and the default access to host group data.

用户类型描述	
Zabbix 用户用	可以访问“监测中”菜单页面。默认情况下，用户无权访问任何资源。必须明确分配对主机组的任何权限。

用户类型	描述
Zabbix 管理员用户	以访问“监测中和配置”菜单页面。默认情况下，用户无权访问任何主机组。必须明确给出对主机组的任何权限。

用户类型	描述
Zabbix 超级管理员用户可以	问所有内容：监测中、配置和管理菜单页面。用户对所有主机组具有读写访问权限。权限不能通过拒绝对特定主机组的访问来撤销。

User type	Description
Zabbix User	The user has access to the Monitoring menu. The user has no access to any resources by default. Any permissions to host groups must be explicitly assigned.

User type	Description
Zabbix Admin	The user has access to the Monitoring and Configuration menus. The user has no access to any host groups by default. Any permissions to host groups must be explicitly given.
Zabbix Super Admin	The user has access to everything: Monitoring, Configuration and Administration menus. The user has a read-write access to all host groups. Permissions cannot be revoked by denying access to specific host groups.

主机组权限

Permissions to host groups

只准许主机组级别的用户组访问 Zabbix 中的任何主机数据。

Access to any host data in Zabbix are granted to user groups on host group level only.

这意味着个人用户不能被直接授予对主机（或主机组）的访问权限。个人用户只能通过其归属的用户组被授予主机组访问权限，进而访问该主机组下的主机 (即个人用户------→ 用户组------→ 主机组-----→ 主机)。

That means that an individual user cannot be directly granted access to a host (or host group). It can only be granted access to a host by being part of a user group that is granted access to the host group that contains the host.

3 用户组

3 User groups

概述

Overview

用户组可以为组用户组织目的和对数据分配权限。对于主机组的监控数据权限只能分配给用户组，而不是个人用户。

User groups allow to group users both for organizational purposes and for assigning permissions to data. Permissions to monitoring data of host groups are assigned to user groups, not individual users.

将一组用户和另一组用户的可用信息单独分离开，这样做通常会更有意义。因为这样可以通过用户进行分组，然后将不同的权限分配给主机组来实现。

It may often make sense to separate what information is available for one group of users and what - for another. This can be accomplished by grouping users and then assigning varied permissions to host groups.

一个用户可以属于多个用户组。

A user can belong to any amount of groups.

配置

Configuration

通过以下步骤配置用户组：

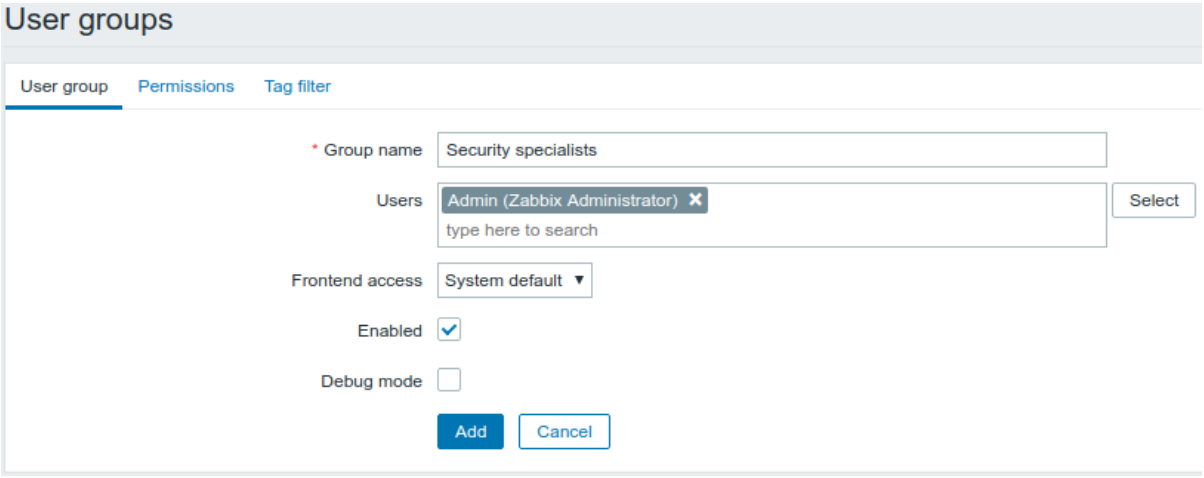
To configure a user group:

- 在 Zabbix 前端跳转到//管理 → 用户组 * * 点击 * 创建用户组// （或者在用户组名上编辑现有的用户组）
- 在表单中编辑用户组属性。

- Go to *Administration* → *User groups*
- Click on *Create user group* (or on the group name to edit an existing group)
- Edit group attributes in the form

“用户组” 标签页包含以下常规的用户组属性：

The **User group** tab contains general group attributes:



All mandatory input fields are marked with a red asterisk.

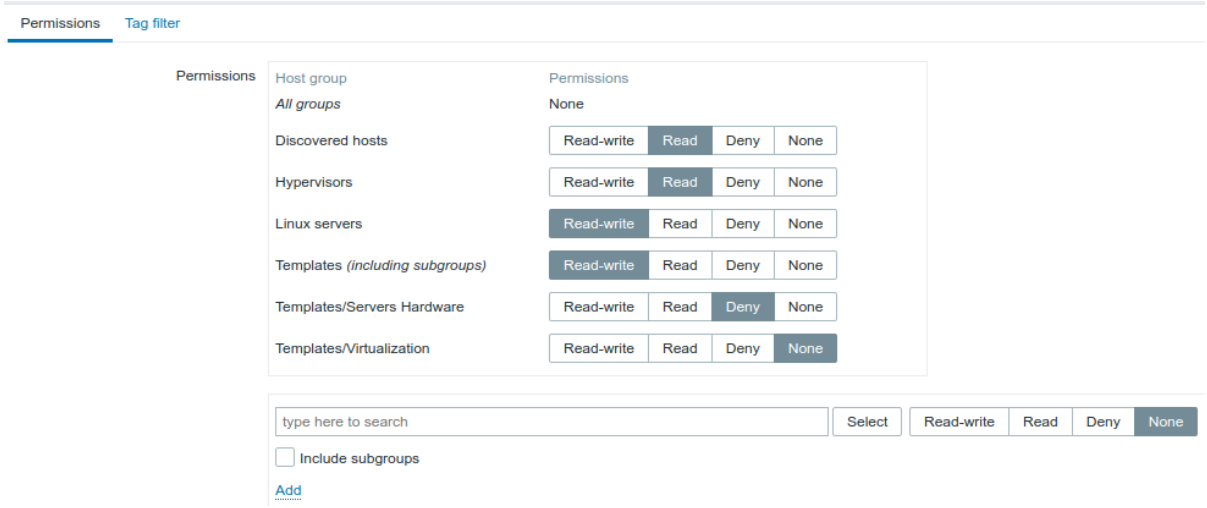
参数	描
组名	唯 的组名.
用户 *	在组中的... 这个方框内包含当前组内用户的列表. 要将其他用户添加到此组中，请在其他组这个方框下选择相应的用户，并点击«** 按钮进行添加.

参数描	
前端访问如何对	内用户进行身份验证. 系统默认 - 使用默认的验证方式 **Internal** - 使用 Zabbix 验证. 如果设置了HTTP验证，则忽略此项. 停用的 - 被禁止访问 Zabbix GUI.
已启用用户	和组成员的状态. 已选中 - 用户组和用户被启用. 未选中 - 用户组和用户被禁用.
调试模式选中此	将会激活用户的调试模式.

Parameter	Description
Group name	Unique group name.
Users	To add users to the group click *Select* button.
Frontend access	How the users of the group are authenticated. **System default** - use default authentication **Internal** - use Zabbix authentication. Ignored if HTTP authentication is set **Disabled** - access to Zabbix GUI is forbidden
Enabled	Status of user group and group members. *Checked* - user group and users are enabled *Unchecked* - user group and users are disabled
Debug mode	Mark this checkbox to activate debug mode for the users.

权限标签页允许你指定用户组访问主机组（和主机组内主机）数据：

The **Permissions** tab allows you to specify user group access to host group (and thereby host) data:



主机组的当前权限显示在权限方框内。

Current permissions to host groups are displayed in the *Permissions* block.

如果主机组的当前权限由所有嵌套主机组继承，则由主机组名称后面的括号中的包含的子组文本指示。

If current permissions of the host group are inherited by all nested host groups, that is indicated by the *including subgroups* text in the parenthesis after the host group name.

您可以更改对主机组的访问级别：

You may change the level of access to a host group:

- 读写 - 对主机组具有读写权限；
- 只读 - 对主机组具有只读权限；
- 拒绝 - 拒绝对主机组的访问；
- 无 - 不设置任何权限。

- **Read-write** - read-write access to a host group;
- **Read** - read-only access to a host group;
- **Deny** - access to a host group denied;
- **None** - no permissions are set.

使用下面的选择字段选择主机组和对它们的访问级别（请注意，如果组已经在列表中，则选择无将从列表中删除主机组）。如果要包括嵌套主机组，请选中“包含子组”复选框。该字段是自动完成的，因此开始键入主机组的名称将提供匹配组的下拉列表。如果你希望查看所有主机组，请单击选择按钮。

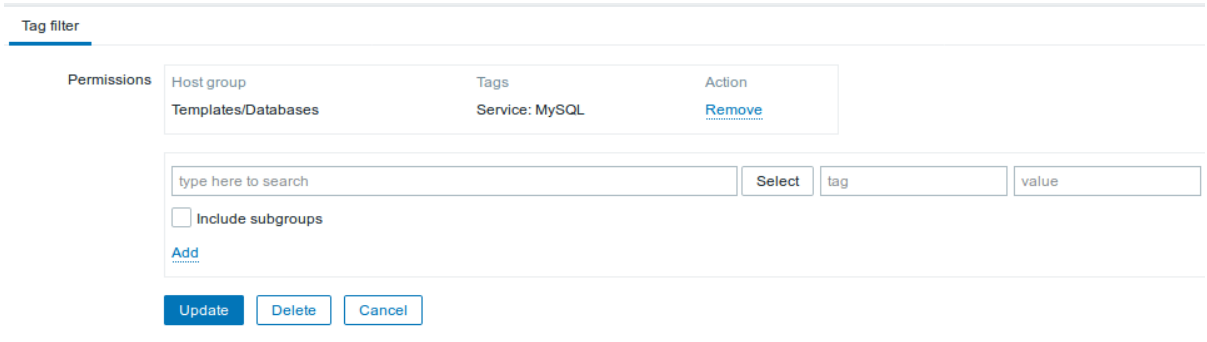
Use the selection field below to select host groups and the level of access to them (note that selecting *None* will remove host group from the list if the group is already in the list). If you wish to include nested host groups, mark the *Include subgroups* checkbox. This field is auto-complete so starting to type the name of a host group will offer a dropdown of matching groups. If you wish to see all host groups, click on *Select*.

请注意在主机组configuration Zabbix 超级管理员拥有内置主机组同等级别的权限。

Note that it is possible for Zabbix Super Admin users in host group configuration to enforce the same level of permissions to the nested host groups as the parent host group.

Tag filter 标签页允许您通过过滤标签名和标签值，来设置用户组查看问题基于标签维度的权限。

The **Tag filter** tab allows you to set tag based permissions for user groups to see problems filtered by tag name and its value:



选择一个标签过滤某个主机组，点击 *Select* 查看完整的已有的主机组列表或输入一个主机组名来获取匹配的主机组的下拉列表。如果您想使用内置的主机组标签，标记 *Include subgroups* 复选框。

To select a host group to apply a tag filter for, click *Select* to get the complete list of existing host groups or start to type the name of a host group to get a dropdown of matching groups. If you want to apply tag filters to nested host groups, mark the *Include subgroups* checkbox.

标签过滤允许分离主机组的访问可能性。

Tag filter allows to separate the access to host group from the possibility to see problems.

例如，如果一个数据库管理员需要只查看”MySQL” 数据库的问题，则需要先创建一个数据管理员用户组，然后配置”Service” 标签名的值为”MySQL”。

For example, if a database administrator needs to see only ”MySQL” database problems, it is required to create a user group for database administrators first, than specify ”Service” tag name and ”MySQL” value.

如果在左侧空白处指定标签名和值，对应的用户组将可以看到该标签下所选主机的所有问题。

If ”Service” tag name is specified and value field is left blank, corresponding user group will see all problems for selected host group with tag name ”Service”.

如果左侧空白处标签名和值都未指定，但是选择了主机组，对应的用户组将可以看到所选主机的所有问题。请确保准确地配置了标签名和标签值，否则对应的用户组将看不到任何问题。

If both tag name and value fields are left blank but host group selected, corresponding user group will see all problems for selected host group. Make sure a tag name and tag value are correctly specified otherwise a corresponding user group will not see any problems.

如下是一个用户归属于多个用户组的例子。在本例中涉及到标签过滤的说明。

Let’s review an example when a user is a member of several user groups selected. Filtering in this case will use OR condition for tags.

用户组 **A**		**	户组 B**		** 两组中	户 (组) 的可见结果 **
标签过滤						
主机组 * 标	名 * 标签值	主机组	标签名 * 标签	*		
Templates/Databases	Service	MySQL	Templates/Databases	Service	Oracle	Service: MySQL or Oracle problems visible
Templates/Databases	*blank*	*blank*	Templates/Databases	Service	Oracle	All problems visible
not selected	*blank*	*blank*	Templates/Databases	Service	Oracle	Service:Oracle problems visible

User group A			**User group B**			**Visible result for a user (member) of both groups**
Tag filter						
Host group	*Tag name*	*Tag value*	*Host group*	*Tag name*	*Tag value*	
Templates/Databases	Service	MySQL	Templates/Databases	Service	Oracle	Service: MySQL or Oracle problems visible
Templates/Databases	*blank*	*blank*	Templates/Databases	Service	Oracle	All problems visible

not selected	*blank*	*blank*	Templates/Databases	Service	Oracle	Service:Oracle problems visible

Attention:
添加过滤 (例如, 在主机组名”Templates/Databases” 中添加标签) 将导致其它的主机组的问题不能够被发现。

Attention:
Adding a filter (for example, all tags in a certain host group ”Templates/Databases”) results in not being able to see the problems of other host groups.

来自多个用户组的主机访问

Host access from several user groups

用户可以属于任意数量的用户组。这些组对主机可能具有不同的访问权限。

A user may belong to any number of user groups. These groups may have different access permissions to hosts.

因此，重要的是要知道非特权用户将能够访问哪些主机。例如，让我们考虑如何在用户组 A 和 B 中的用户的各种情况下对 “主机 ** X **” (在主机组 1 中）的访问将受到影响。

Therefore, it is important to know what hosts an unprivileged user will be able to access as a result. For example, let us consider how access to host **X** (in Hostgroup 1) will be affected in various situations for a user who is in user groups A and B.

- 如果 “用户组 A ” 没有定义权限，同时 “用户组 B ” 具有对 “主机组 1 ” 的读写权限，那么用户将获得对 “主机 X ” 的读写访问。
- If Group A has only *Read* access to Hostgroup 1, but Group B *Read-write* access to Hostgroup 1, the user will get **Read-write** access to ’X’.

Attention:
从 Zabbix 2.2 开始，” 读写 “ 权限要优先于 “只读” 权限。

Attention:
“Read-write” permissions have precedence over “Read” permissions starting with Zabbix 2.2.

- 在与上述相同的情况下，如果 “主机组 2” 中的 “主机 X ” 同时拒绝 “用户组 A ” 或 “用户组 B ”，那么 “主机 X ” 的访问将不可用，尽管 “主机组 1 ” 有读写权限。
- 如果 “用户组 A ” 没有定义权限，同时 “用户组 B ” 具有对 “主机组 1 ” 的读写权限，那么用户将获得对 “主机 X ” 的读写访问。
- 如果 “用户组 A ” 具有对 “主机组 1 ” 的拒绝权限，同时 “用户组 B” 具有对 “主机组 1 ” 的读写权限，则用户访问 “主机 X ” 将被拒绝。

- In the same scenario as above, if ’X’ is simultaneously also in Hostgroup 2 that is **denied** to Group A or B, access to ’X’ will be **unavailable**, despite a *Read-write* access to Hostgroup 1.
- If Group A has no permissions defined and Group B has a *Read-write* access to Hostgroup 1, the user will get **Read-write** access to ’X’.
- If Group A has *Deny* access to Hostgroup 1 and Group B has a *Read-write* access to Hostgroup 1, the user will get access to ’X’ **denied**.

其他细节

Other details

- 当拓扑图为空或者只有图片时，任何非 Zabbix 超级管理员 (包含’guest’) 都可以看到网络图。当主机，主机组或者触发器添加到拓扑图中，就要考虑权限问题。同样，屏幕 (screens) 和幻灯片 (slideshows) 也如此。不考虑权限的情况下，用户可以看见任何非直接或者间接链接到主机的项。
- 如果一个具有对主机具有读写权限的管理级别用户无法访问 *Templates* 主机组，则具有读写访问主机的管理级用户将无法链接或取消链接模板。使用只读访问 *Templates* 主机组，他将能够链接或取消链接到主机的模板，但是，模板列表中不会看到任何模板，也不能在其他地方使用模板。
- 具有只读访问主机的管理级用户将不会在配置页面的主机列表中看到主机; 但是，在 IT 服务配置中可以访问主机触发器。
- 只要地图为空或只有图像，任何非 Zabbix 超级管理员用户（包括 “guest”）都可以看到网络地图。当主机、主机组或触发器被添加到地图时，权限被遵守。这同样适用于屏幕和幻灯片。无论权限如何，用户将看到任何没有直接或间接链接到主机的对象。

- Any non-Zabbix Super Admin user (including ’guest’) can see network maps as long as the map is empty or has only images. When hosts, host groups or triggers are added to the map, permissions are respected. The same applies to screens and

slideshows as well. The users, regardless of permissions, will see any objects that are not directly or indirectly linked to hosts.

- An Admin level user with *Read-write* access to a host will not be able to link/unlink templates, if he has no access to the *Templates* group. With *Read* access to *Templates* group he will be able to link/unlink templates to the host, however, will not see any templates in the template list and will not be able to operate with templates in other places.
- An Admin level user with *Read* access to a host will not see the host in the configuration section host list; however, the host triggers will be accessible in IT service configuration.
- Any non-Zabbix Super Admin user (including 'guest') can see network maps as long as the map is empty or has only images. When hosts, host groups or triggers are added to the map, permissions are respected. The same applies to screens and slideshows as well. The users, regardless of permissions, will see any objects that are not directly or indirectly linked to hosts.
- Zabbix server will not send notifications to users defined as action operation recipients if access to the concerned host is explicitly "denied".

8. Service 服务监控 [ZeMing]

8. Service monitoring

> **Attention:**
> Zabbix 4.0.0 正式版目前，暂未发布，敬请期待，目前阶段 ** alpha9** 。

概述

Overview 服务 (Serivce) 监控功能是为帮助那些想要在 IT 基础设施监控之上获得更高层面（业务）监控需求的人设计。在许多情况下，我们不关心底层设施监控细节，比如磁盘空间不足，CPU 高负载等等。我们关心的是 IT 部门提供的服务（业务）整体的可用性。我们还关心在整体 IT 基础设施中最薄弱的环节，以及各种 IT 服务的 SLA 指标（SLA 服务等级协议），更关心识别现有 IT 基础设施架构薄弱环节，以及更高层面（业务）的监控信息。

Service monitoring functionality is intended for those who want to get a high-level (business) view of monitored infrastructure. In many cases, we are not interested in low-level details, like the lack of disk space, high processor load, etc. What we are interested in is the availability of service provided by our IT department. We can also be interested in identifying weak places of IT infrastructure, SLA of various IT services, the structure of existing IT infrastructure, and other information of a higher level.

Zabbix 服务监控（service）就是针对上述问题提出的解决方案。

Zabbix service monitoring provides answers to all mentioned questions.

服务监控（service）是分层表现监控数据的。

Services is a hierarchy representation of monitored data.

下面我们来看一个非常简单的服务监控（service）例子：

A very simple service structure may look like:

```
Service
|
|-Workstations
| |
| |-Workstation1
| |
| |-Workstation2
|
|-Servers
```

在结构上每个节点都具有监控属性状态。这个状态根据所选算法计算状态关联上层状态，服务监控（service）功能最底层是关联的触发器（triggers）。每个节点状态都是受其触发器（triggers）状态影响。

Each node of the structure has attribute status. The status is calculated and propagated to upper levels according to the selected algorithm. At the lowest level of services are triggers. The status of individual nodes is affected by the status of their triggers.

Note:
提示：触发器（triggers）的严重等级如：不分类（*Not classified*）或信息（*Information*）是不影响 SLA 指标（SLA 服务等级协议）计算的。

Note:
Note that triggers with a *Not classified* or *Information* severity do not impact SLA calculation.

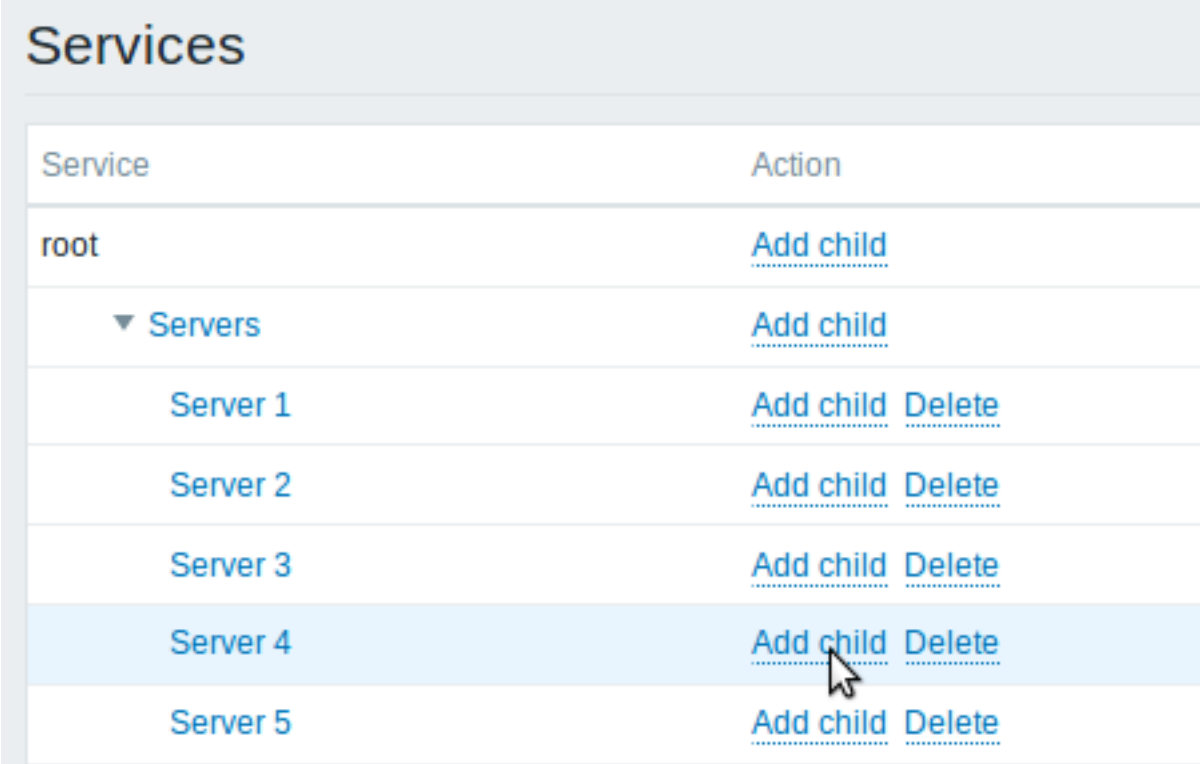
配置

Configuration 配置服务监控（service），请访问：配置（*Configuration*）→ 服务监控（*Services*）. To configure services, go to: *Configuration* → *Services*.

在该界面，您可以创建一个分层监控结构。最高的父节点服务是'root'。您可以向下创建更低层服务监控（service）子节点，实现相互层级结构。

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.

在这个屏幕上，您可以构建被监视的基础结构的层次结构。最高级的父服务是 "root"。您可以向下构建层次结构，方法是添加低级的父服务，然后向它们添加单个节点。



点击 添加子节点（*Add child*）增加服务监控（service）。点击名称可编辑已该服务监控（service），您可以通过弹出的界面编辑该服务监控（service）属性。

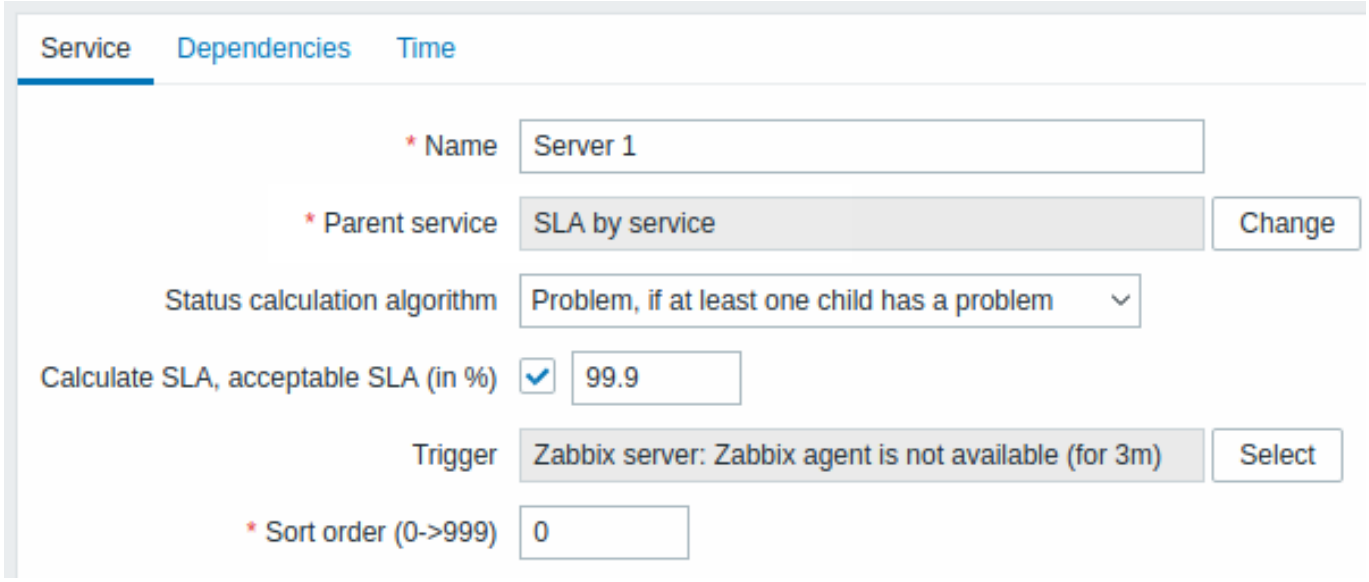
Click on *Add child* to add services. To edit an existing service, click on its name. A form is displayed where you can edit the service attributes.

配置一个服务监控（service）

Configuring a service

服务监控（**service)** 选项包含通用的服务监控（service）属性

The **Service** tab contains general service attributes:



所有必填字段都标有 * 红色星号。

All mandatory input fields are marked with a red asterisk.

参数说	
名称（*Name*）服务监	名称.ervice）
上层服务（*Parent service*）服务监控（	从属的上层父节点

参数说	
状态计算算法（*Status calculation algorithm*）服务监控（service)	* 不计算 * *, 不计算服务状态\ * * 问题, 如果至少有一个子节点服务有问题 * *—一问题状态, 如果至少一个子节点服务有问题\ * * 问题, 如果所有的孩子都

参数	说
计算 *SLA*（*Calculate SLA*）勾选 ✔	是否计算SLA指标。
可接受的 *SAL(%)*（*Acceptable SLA (in %)*）此服务监控	service)节点，可接受的SLA百分比，用于报告。

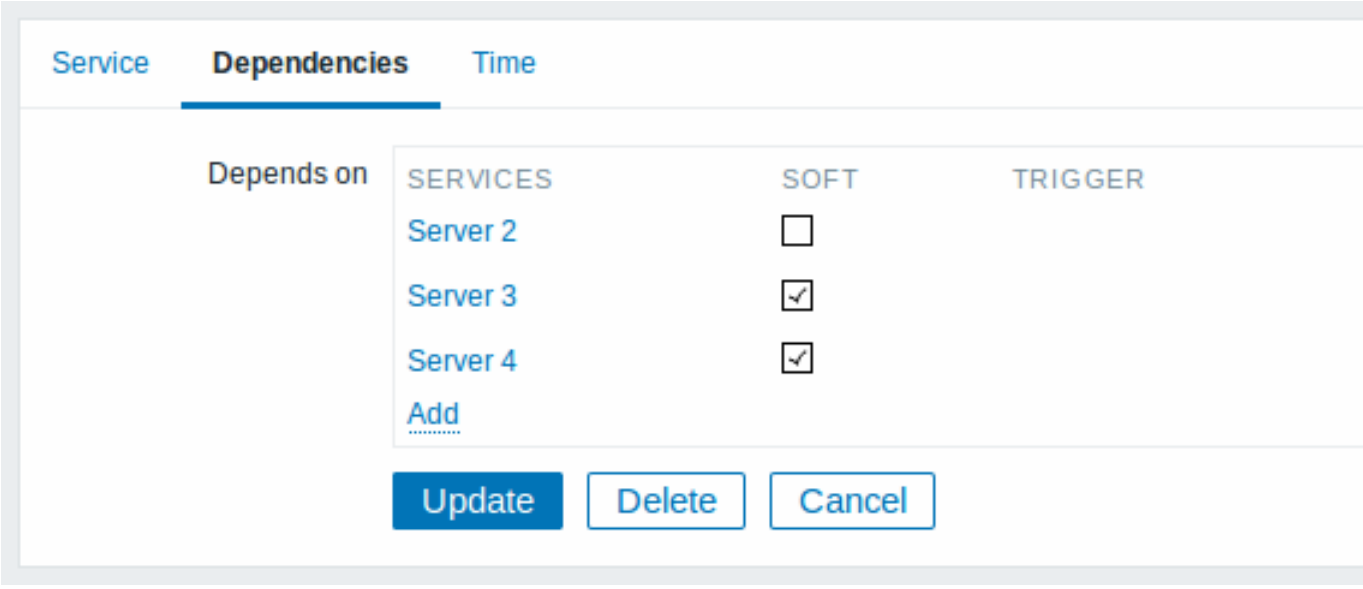
参数	说
触发器（*Trigger*）选择关联	触发器(Trigger): **None** - 没有关联的触发器(Trigger) 触发器名称(**trigger name**) - 选择绑定触发器(Trigger),因此节点状态依赖触发器(Trigger)状态. 最底层服务计算必须依赖触发

参数说	
排序（*Sort order*）显示排	的顺序，数字小的优先

Parameter	Description
Name	Service name.
Parent service	Parent service the service belongs to.
Status calculation algorithm	Method of calculating service status: **Do not calculate** - do not calculate service status **Problem, if at least one child has a problem** - problem status, if at least one child service has a problem **Problem, if all children have problems** - problem status, if all child services are having problems
Calculate SLA	Enable SLA calculation and display.
Acceptable SLA (in %)	SLA percentage that is acceptable for this service. Used for reporting.
Trigger	Linkage to trigger: **None** - no linkage **trigger name** - linked to the trigger, thus depends on the trigger status Services of the lowest level must be linked to triggers. (Otherwise their state will not be represented accurately.) When triggers are linked, their state prior to linking is not counted.
Sort order	Sort order for display, lowest comes first.

依赖关系（**Dependencies**）选项卡可以看到该服务监控（service) 所有子节点。单击 *Add* 增加一个之前配置过的服务监控（service) 节点。

The **Dependencies** tab contains services the service depends on. Click on *Add* to add a service from those that are configured.



硬依赖和软依赖（**Hard and soft dependency**） **Hard and soft dependency**

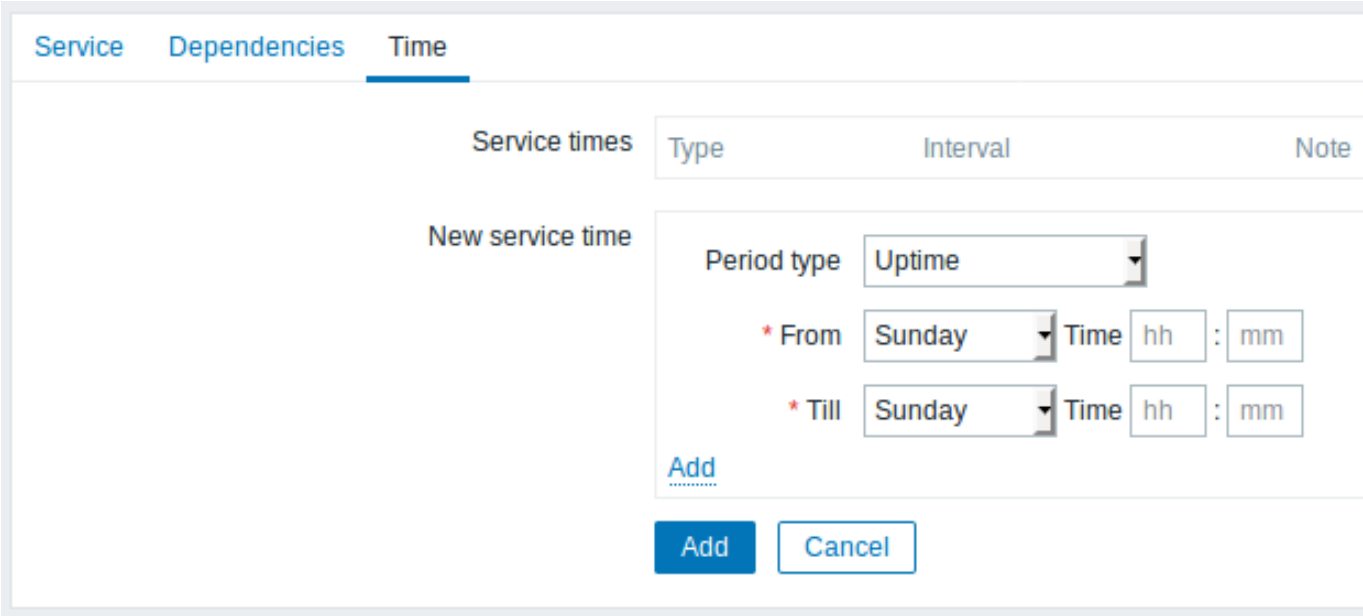
服务的可用性指标，可能取决于其他多个服务，而不仅仅是一个。界面第一个选项服务监控（service) 是直接增加子节点。

Availability of a service may depend on several other services, not just one. The first option is to add all those directly as child services.

而然，如果服务（service) 在其他节点已增加过，则不能简单的将它移动到该子节点。那该如何创建服务（service) 节点依赖？这个问题答案是” 软链接”。增加一个服务（service) 依赖是，勾选 √ 软连接 *(soft)* 选项。通过这种方式，服务（service) 可以保留节点之前原始位置，有可以绑定依赖到其他服务（service) 上。这种 “软连接” 的服务（service) 节点在服务树上显示是灰色的。另外，如果一个服务只有一个 “软连接” 节点，就可以删除此服务，而不用删除软连接的子节点。

However, if some service is already added somewhere else in the services tree, it cannot be simply moved out of there to a child service here. How to create a dependency on it? The answer is ”soft” linking. Add the service and mark the *Soft* check box. That way the service can remain in its original location in the tree, yet be depended upon from several other services. Services that are ”soft-linked” are displayed in grey in the tree. Additionally, if a service has only ”soft” dependencies, it can be deleted directly, without deleting child services first.

时间（**Time**）选型，用于设置服务（service) 的工作时间。The **Time** tab contains the service time specification.



参数	说
服务时间（*Service times*）默认，所有	务（Service）都是预设24x7x365统计时间，如有特殊需要，请增加新的服务时间（New service time）。

参数说
新的服务时间（*New service time*）服务时间（Service times）：在线时间（**Uptime**）- 服务正常运行时间故障停机时间（**Downtime**）- 故障停机时间周期内不会纳入SLA服务时间统计.单次停机（**One-time downtime**）- 单次停机时间，在该时

参数说

Parameter	Description
Service times *New service time*	By default, all services are expected to operate 24x7x365. If exceptions needed, add new service times. Service times: **Uptime** - service uptime **Downtime** - service state within this period does not affect SLA. **One-time downtime** - a single downtime. Service state within this period does not affect SLA. Add the respective hours. *Note*: Service times affect only the service they are configured for. Thus, a parent service will not take into account the service time configured on a child service (unless a corresponding service time is configured on the parent service as well). Service times are taken into account when calculating service status and SLA by the frontend. However, information on service availability is being inserted into database continuously, regardless of service times.

前端显示

Display 服务（Service），去监控中（Monitoring）-→ 服务（Service）

To monitor services, go to *Monitoring → Services*.

9. Web 监控 [ZeMing]

9. Web monitoring

Overview

概述 您可以使用 Zabbix 对多个网站进行可用性方面监控：
With Zabbix you can check several availability aspects of web sites.

Attention:
若要使用 Web 监控，必须编译编译（configured）安装是加入 cURL (libcurl) 库支持

Attention:
To perform web monitoring Zabbix server must be initially configured with cURL (libcurl) support.

要使用 Web 监控，您需要定义 web 场景。Web 场景包括一个或多个 HTTP 请求或 “步骤”。Zabbix 服务器根据预定义的命令周期性的执行这些步骤。

To activate web monitoring you need to define web scenarios. A web scenario consists of one or several HTTP requests or ”steps”. The steps are periodically executed by Zabbix server in a pre-defined order. If a host is monitored by proxy, the steps are executed by the proxy.

从 Zabbix2.2 开始，Web 场景和 Items，Triggers 等一样，是依附在 Hosts/Templates 的。这意味着 web 场景也可以创建一个模板，然后应用于多个主机。Since Zabbix 2.2 web scenarios are attached to hosts/templates in the same way as items, triggers, etc. That means that web scenarios can also be created on a template level and then applied to multiple hosts in one move.

所有的 web 场景会收集下列数据：整个场景中所有步骤的平均下载速度失败的步骤数量最后一次错误信息对于 web 场景的所有步骤，都会收集下列数据：平均下载速度响应时间 HTTP 状态码更多详情，请参见web monitoring items.

The following information is collected in any web scenario:

- average download speed per second for all steps of whole scenario

- number of the step that failed
- last error message

The following information is collected in any web scenario step:

- download speed per second
- response time
- response code

For more details, see web monitoring items.

执行 web 场景收集的数据保存在数据库中。数据自动用于图形、触发器和通知。

Data collected from executing web scenarios is kept in the database. The data is automatically used for graphs, triggers and notifications.

Zabbix 还支持获取 HTML 内容中是否存在设置的字符串。还可以模拟登陆动作和模拟鼠标单击。Zabbix can also check if a retrieved HTML page contains a pre-defined string. It can execute a simulated login and follow a path of simulated mouse clicks on the page.

Zabbix web 监控同时支持 HTTP 和 HTTPS。当运行 web 场景时，Zabbix 将选择跟踪重定向（请参见下面的选择跟踪重定向）。重定向硬编码的最大数量为 10 （使用 cURL 选项 CURLOPT_MAXREDIRS)。在执行 web 场景时，所有 Cookie 都会保存。

Zabbix web monitoring supports both HTTP and HTTPS. When running a web scenario, Zabbix will optionally follow redirects (see option *Follow redirects* below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS). All cookies are preserved during the execution of a single scenario.

web 监控使用 HTTPS 协议请参阅已知问题 See also known issues for web monitoring using HTTPS protocol.

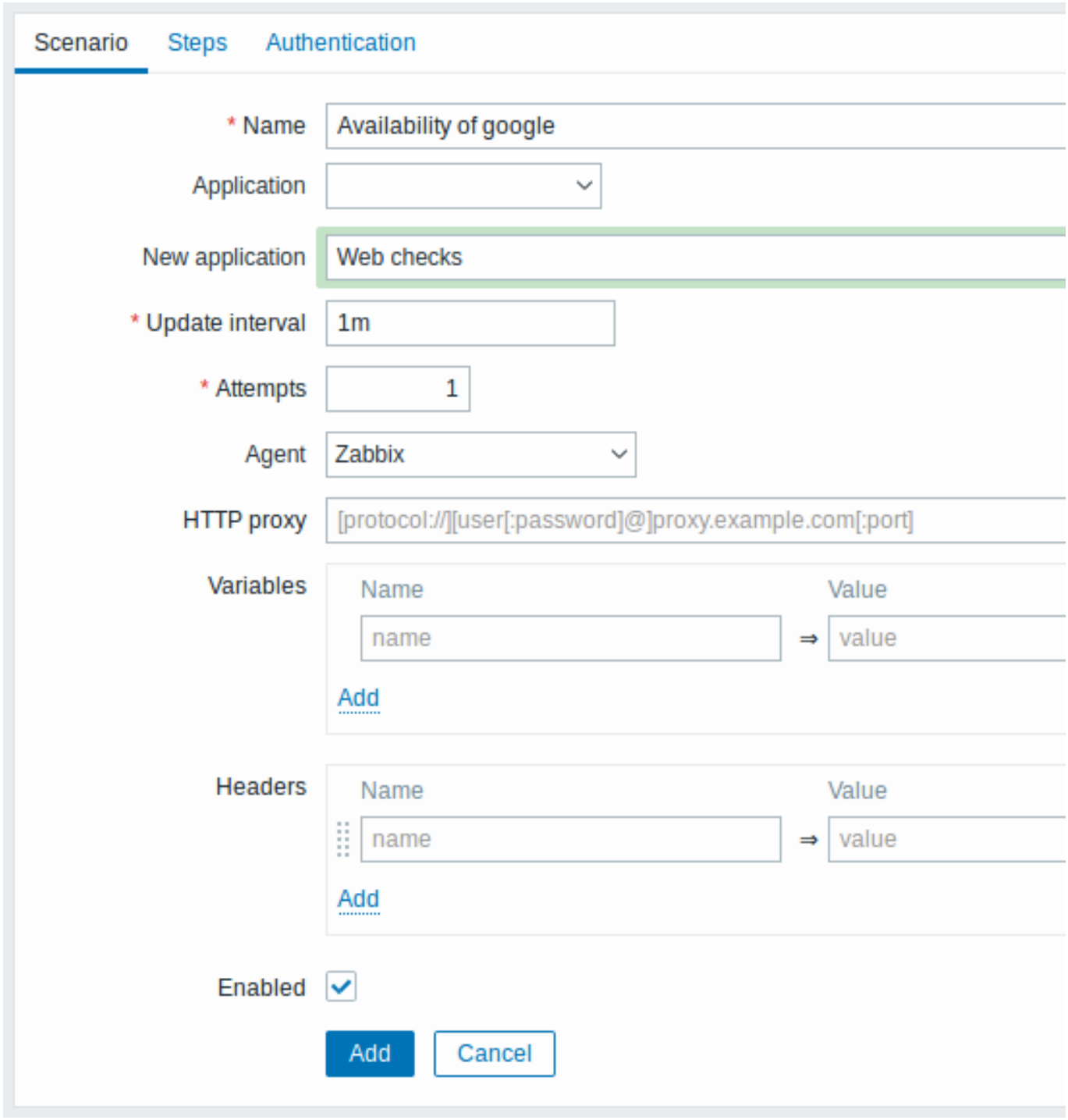
配置 **Web** 场景

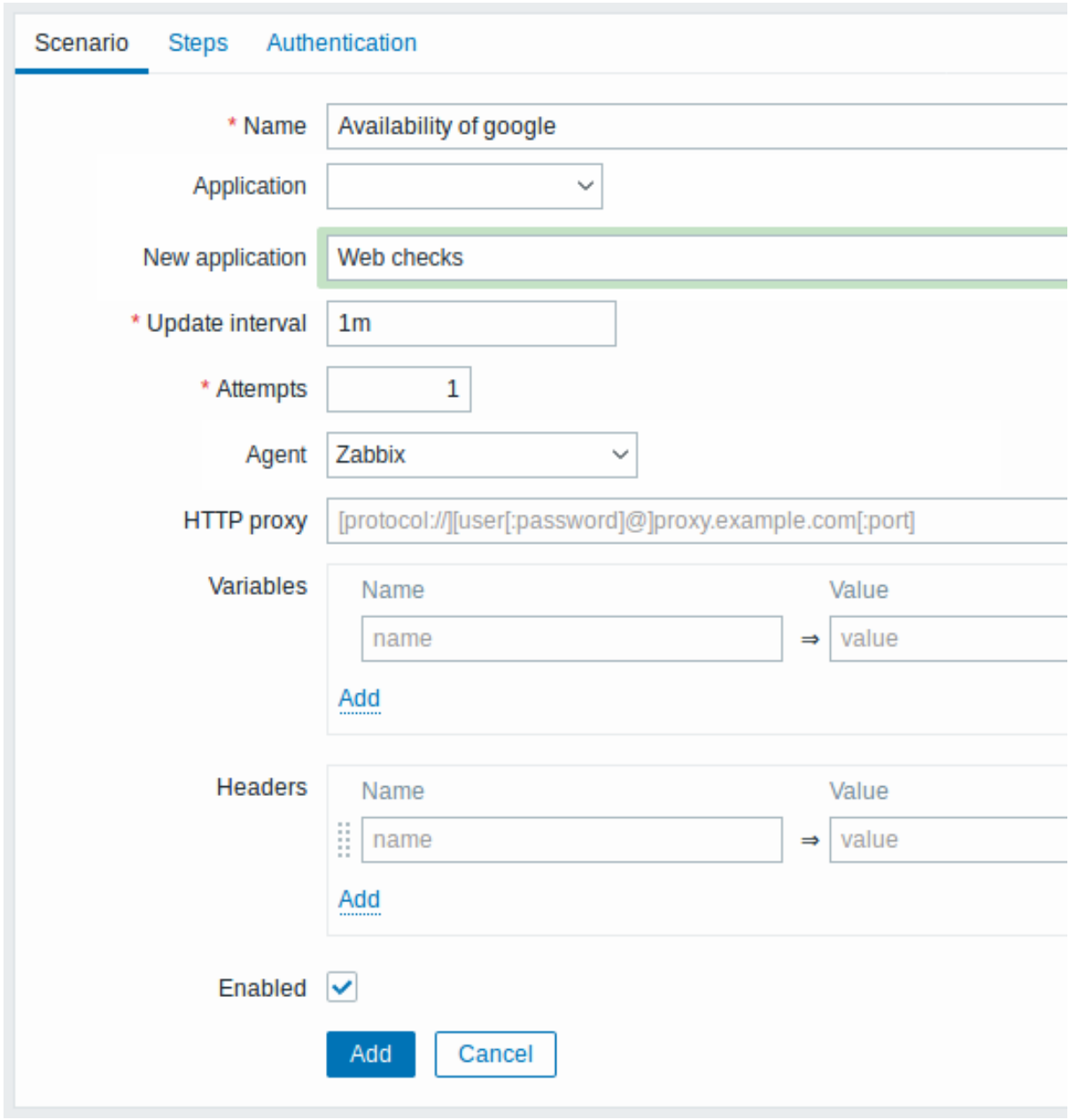
Configuring a web scenario 配置 web 场景：To configure a web scenario:

* 转到：// 配置 (Configuration)-→ 主机 //（或者 // 模板 //)

- Go to: *Configuration* → *Hosts* (or *Templates*)
- 点击主机 (host)/ 模板 (template) 行中的 *Web*
- Click on *Web* in the row of the host/template
- 点击右上角 // 创建 web 场景 //（或点击场景名字进行编辑现有的场景）
- Click on *Create scenario* to the right (or on the scenario name to edit an existing scenario)
- 在场景的表单中输入参数
- Enter parameters of the scenario in the form

场景选项卡允许您配置此 Web 场景的通用参数。The **Scenario** tab allows you to configure the general parameters of a web scenario.


Scenario
Steps
Authentication
* Name
Availability of google
Application
New application
Web checks
* Update interval
1m
* Attempts
1
Agent
Zabbix
HTTP proxy
[protocol://][user[:password]@]proxy.example.com[:port]
Variables
Name
Value
name
value
Add
Headers
Name
Value
name
value
Add
Enabled
Add
Cancel



All mandatory input fields are marked with a red asterisk.

场景参数：Scenario parameters:

场景参数：

参数	说
// 主机 (Host)// 场	所属的主机名或模板的名字。

参数	说
// 名称 (Name)// 唯	的场景名称。Zabbix 2.2 开始，这个名字支持用户宏和 {HOST.*} 宏。

参数	说
// 应用 (Application)// 选	一个场景属于的应用。Web 场景监控项在 // 监测中 (Monitoring)→ 最新数据 (Latest data)// 栏中将会分组在选择的应用中。
// 新的应用 (New application)// 对场景	建个新的应用。
// 更新间隔 (Update interval) （秒）// 执行场景时间	隔，以秒为单位。

参数	说
// 重试次数 (Attempts)// 尝试执	web场景中步骤的次数。对于网络问题(超时，没有连接，等等)Zab-bix可以多次重复执行步骤。这个数字对场景的中的所有步骤都会生效。尝试次数最大可以设置为10，

参数说
// 代理 (Agent)// 选

一个客户端。zabbix会模拟选择的浏览器，当一个网站对不同的浏览器返回不同的内容的时候是非常有用的。*zabbix 2.2*开始，这块可以使用用户自定义宏。

参数	说
HTTP 代理 *(HTTP proxy)*	您以指定要使用一个 HTTP 代理，使用格式 *http://[usernam* 默认使用 1080 端口。如果指定，代理将覆盖代理相关联的环境变量，比如 http_proxy HTTPS_PROXY。如果没有指定，那么代理将不会覆盖代理相关

参数说	
// 变量 (Variables)// 可	在场景中的步骤 (URL，POST 变量) 中使用变量。它们具有以下格式：**{macro1}**=val **{macro2}**=val **{macro3}**=reg ex- pres- sion> 例如：{username}=Al {password}=kj3 {hostid}=regex is ([0- 9]+) 然后可以在 {user- name}，{pass- word} 和 {hostid} 的步骤中引用宏。Zab- bix 将自动将其替换

参数	说
HTTP 头 (Headers)	行请求时将发送的自定义的 HTTP headers。应使用与在 HTTP 协议中出现的语法相同的语法列出标题，可选地使用 CURLOPT_HTTPHEAD cURL 选项支持的一些其他功能。例如：Accept-Charset=utf-8 Accept-Language=en-US

参数说	
// 启用 (Enabled)// 如	选中此复选框，则此场景处于启用状态，否则禁用。

Parameter	Description
Host	Name of the host/template that the scenario belongs to.
Name	Unique scenario name. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Application	Select an application the scenario will belong to. Web scenario items will be grouped under the selected application in *Monitoring* → *Latest data*.
New application	Enter the name of a new application for the scenario.
Update interval	How often the scenario will be executed. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note* that if a user macro is used and its value is changed (e.g. 5m → 30s), the next check will be executed according to the previous value (farther in the future with the example values).
Attempts	The number of attempts for executing web scenario steps. In case of network problems (timeout, no connectivity, etc) Zabbix can repeat executing a step several times. The figure set will equally affect each step of the scenario. Up to 10 attempts can be specified, default value is 1. *Note*: Zabbix will not repeat a step because of a wrong response code or the mismatch of a required string. This parameter is supported starting with *Zabbix 2.2*.
Agent	Select a client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers. User macros can be used in this field, *starting with Zabbix 2.2*.

Parameter	Description
HTTP proxy	You can specify an HTTP proxy to use, using the format: *http://[username[:password]@]proxy.mycompany.com[:port]* By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy related environment variables. The entered value is passed on "as is", no sanity checking takes place. You may also enter a SOCKS proxy address. If you specify the wrong protocol, the connection will fail and the item will become unsupported. With no protocol specified, the proxy will be treated as an HTTP proxy. *Note*: Only simple authentication is supported with HTTP proxy. User macros can be used in this field. This parameter is supported starting with *Zabbix 2.2*.
Variables	Variables that may be used in scenario steps (URL, post variables). They have the following format: **{macro1}**=value1 **{macro2}**=value2 **{macro3}**=regex:<regular expression> For example: {username}=Alexei {password}=kj3h5kJ34bd {hostid}=regex:hostid is ([0-9]+) The macros can then be referenced in the steps as {username}, {password} and {hostid}. Zabbix will automatically replace them with actual values. Note that variables with `regex:` need one step to get the value of the regular expression so the extracted value can only be applied to the step after. If the value part starts with `regex:` then the part after it is treated as a regular expression that searches the web page and, if found, stores the match in the variable. At least one subgroup must be present so that the matched value can be extracted. Regular expression match in variables is supported *since Zabbix 2.2*. User macros and {HOST.*} macros are supported, since Zabbix 2.2. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL.
Headers	Custom HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol, optionally using some additional features supported by the CURLOPT_HTTPHEADER cURL option. For example: Accept-Charset=utf-8 Accept-Language=en-US Content-Type=application/xml; charset=utf-8 User macros and {HOST.*} macros are supported. Specifying custom headers is supported *starting with Zabbix 2.4*.
Enabled	The scenario is active if this box is checked, otherwise - disabled.

注意，当编辑一个现有的场景时，会出现两个额外的按钮：

Note that when editing an existing scenario, two extra buttons are available in the form:

Clone	基于现有的场景的属性创建另一个场景。
Clear history and trends	删除场景的历史记录和趋势数据。这将使服务器在删除数据后立即执行场景。

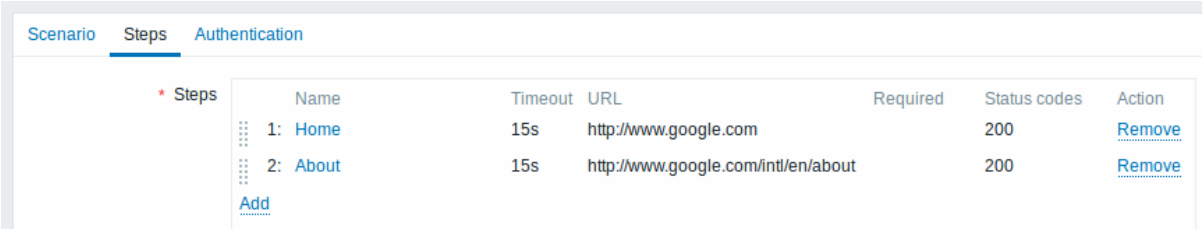
Clone	Create another scenario based on the properties of the existing one.
Clear history and trends	Delete history and trend data for the scenario. This will make the server perform the scenario immediately after deleting the data.

Note:
如果 *HTTP proxy* 字段留空，使用 HTTP 代理的另一种方法是设置代理相关的环境变量。
对于 HTTP 检查 - 为 Zabbix 服务器用户设置 **http_proxy** 环境变量。例如，//http_proxy=http:%%//%%proxy_ip:proxy_port//.
对于 HTTPS 检查 - 设置 **HTTPS_PROXY** 环境变量。例如，*HTTPS_PROXY=http:%%//%%proxy_ip:proxy_port//.* 通过运行 *shell* 命令可以获得更多详细信息：# man curl//.

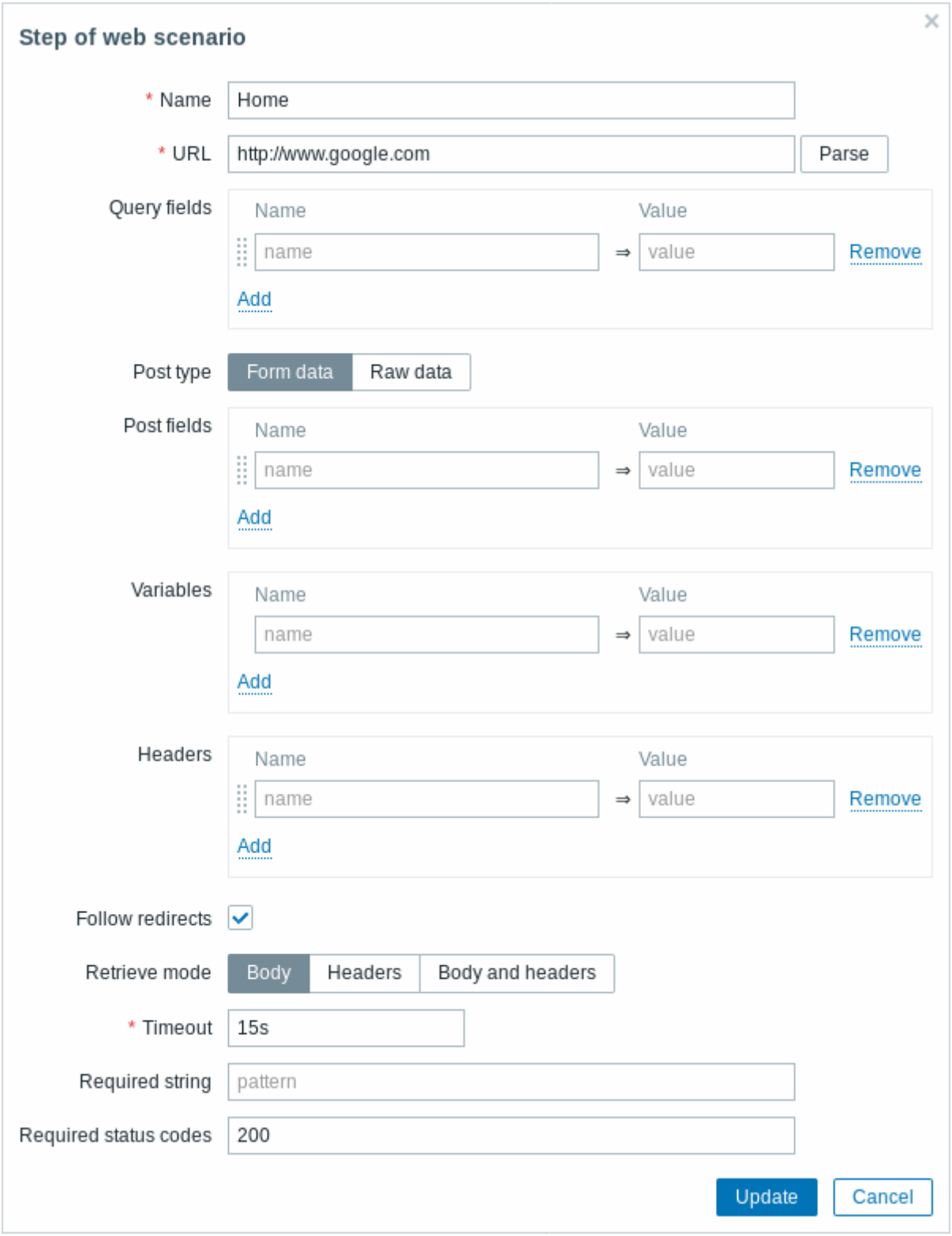
Note:
If *HTTP proxy* field is left empty, another way for using an HTTP proxy is to set proxy related environment variables.
For HTTP checks - set the **http_proxy** environment variable for the Zabbix server user. For example, //http_proxy=http:%%//%%proxy_ip:proxy_port//.
For HTTPS checks - set the **HTTPS_PROXY** environment variable. For example, //HTTPS_PROXY=http:%%//%%proxy_ip:proxy_port//. More details are available by running a shell command: *# man curl*.

“步骤” 选项卡允许您配置 Web 场景步骤。要添加 Web 场景步骤，请在 // 步骤 (Steps)// 单击 // 添加 (Add)//。

The **Steps** tab allows you to configure the web scenario steps. To add a web scenario step, click on *Add* in the *Steps* block.



配置步骤



Configuring steps

Step parameters:

步骤参数：

参数	说
// 名称 (Name)// 唯	步骤名称。*Zabbix 2.2* 开始，该名称可以支持用户宏和 {HOST.*} 宏。

参数	说
// 网址 (URL)// 用	连接和检索数据的网址。例如：https://www.goo http://www.zabb Zabbix 3.4 以后，可以以 Uni-code 编码指定域名。执行 Web 场景步骤时，它们将自动被禁止转换为 ASCII。// 解析 // 按钮可用于从 URL 中分离可选

参数说	
// 查询字段 (Query fields)// ? URL	的 HTTP GET 变量。指定属性和值对。值将自动进行 URL 编码。来自场景变量，用户宏或 {HOST。*} 宏的值将被解析，然后自动进行 URL 编码。使用 {{macro}.urlenc 语法将对其进行双重 URL 编码。从

参数说	
Post	HTTP POST 变量。在 **Form data** 模式下，指定属性和值。值被自动进行 URL 编码。来自场景变量、用户宏或 {HOST.*} 宏的值将被解析，然后自动进行 URL 编码。在 **Raw data** 模式中，属性/值显示在

参数说	
// 变量 (Variables)// 可	于 GET 和 POST 方法的步级变量。指定属性和值。步骤变量覆盖之前的场景变量或步骤变量。然而，一个步骤变量的值仅影响之后的步骤(而不是当前步骤)。它们具有以

参数	说
HTTP 头 (Headers)	行请求时将发送的自定义 HTTP headers。指定属性和值步骤级别上的 headers 将覆盖为该场景指定的 headers。例如，设置 “User-Agent：” 为空时，将覆盖在场景上设置的 User-Agent 名称。支

参数	说
// 跟踪重定向 (Follow redirects)// 选中该复	框以跟踪 HTTP 重定向。将会设置 CURLOPT_FOLLOWL cURL 选项。*Zabbix 2.4* 开始支持此选项。
// 仅检索标头 (Retrieve only headers)// 选中复选	，仅从 HTTP 响应中检索标题。这将设置 CURLOPT_NOBODY cURL 选项。*Zabbix 2.4* 开始支持此选项。

参数	说
// 超时时间 (Timeout)// Zab	ix 根据设置的秒数以内来处理 URL。实际上，此参数定义为连接到 URL 的最大时间和执行 HTTP 请求的最长时间。因此，Zab-bix 不会在步骤上花费超过 **2x** 超时时间。例如：15

参数	说
// 必需的字符串 (Required string)// 必需的正则	达式。除非检索到的内容(HTML)匹配所需的模式，否则步骤将失败。如果为空，则不执行检查。例如：Zabbix的主页、\Wel-come.*admin//注意//:在此字段中不支持引用在Zab-bix前端中创

参数说	
// 状态码 (Required status codes)/// 可以	置预期的HTTP状态代码列表。如果Zab-bix获取的HTTP状态码不在列表中，该步骤将认为为失败。如果为空，则不执行检查。例如：200,201,210-299*Zabbix 2.2*开始，支持用户宏。

Parameter	Description
Name	Unique step name. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
URL	URL to connect to and retrieve data. For example: https://www.google.com http://www.zabbix.com/download Domain names can be specified in Unicode characters since Zabbix 3.4. They are automatically punycode-converted to ASCII when executing the web scenario step. The *Parse* button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into *Query fields* for automatic URL-encoding. Variables can be used in the URL, using the {macro} syntax. Variables can be URL-encoded manually using a {{macro}.urlencode()} syntax. User macros and {HOST.*} macros are supported, since Zabbix 2.2. Limited to 2048 characters *starting with Zabbix 2.4*.
Query fields	HTTP GET variables for the URL. Specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. Using a {{macro}.urlencode()} syntax will double URL-encode them. User macros and {HOST.*} macros are supported since Zabbix 2.2.
Post	HTTP POST variables. In **Form data** mode, specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. In **Raw data** mode, attributes/values are displayed on a single line and concatenated with a **&** symbol. Raw values can be URL-encoded/decoded manually using a {{macro}.urlencode()} or {{macro}.urldecode()} syntax. For example: id=2345&userid={user} If {user} is defined as a variable of the web scenario, it will be replaced by its value when the step is executed. If you wish to URL-encode the variable, substitute {user} with {{user}.urlencode()}. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Variables	Step-level variables that may be used for GET and POST functions. Specified as attribute and value pairs. Step-level variables override scenario-level variables or variables from the previous step. However, the value of a step-level variable only affects the step after (and not the current step). They have the following format: **{macro}**=value **{macro}**=regex:<regular expression> For more information see variable description on the scenario level. Having step-level variables is supported since Zabbix 2.2. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL.
Headers	Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. Headers on the step level will overwrite the headers specified for the scenario. For example, setting a 'User-Agent' attribute with no value will remove the User-Agent value set on scenario level. User macros and {HOST.*} macros are supported. This sets the CURLOPT_HTTPHEADER cURL option. Specifying custom headers is supported *starting with Zabbix 2.4*.

Parameter	Description
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option. This option is supported *starting with Zabbix 2.4*.
Retrieve only headers	Mark the checkbox to retrieve only headers from the HTTP response. This sets the CURLOPT_NOBODY cURL option. This option is supported *starting with Zabbix 2.4*.
Timeout	Zabbix will not spend more than the set amount of time on processing the URL (maximum is 1 hour). Actually this parameter defines the maximum time for making connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than **2 x Timeout** seconds on the step. Time suffixes are supported, e.g. 30s, 1m, 1h. User macros are supported.
Required string	Required regular expressions pattern. Unless retrieved content (HTML) matches required pattern the step will fail. If empty, no check is performed. For example: Homepage of Zabbix Welcome.*admin *Note*: Referencing regular expressions created in the Zabbix frontend is not supported in this field. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Required status codes	List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the step will fail. If empty, no check is performed. For example: 200,201,210-299 User macros are supported since Zabbix 2.2.

Note:
Web 场景步骤中的任何更改只有在保存整个场景时才会保存。

Note:
Any changes in web scenario steps will only be saved when the whole scenario is saved.

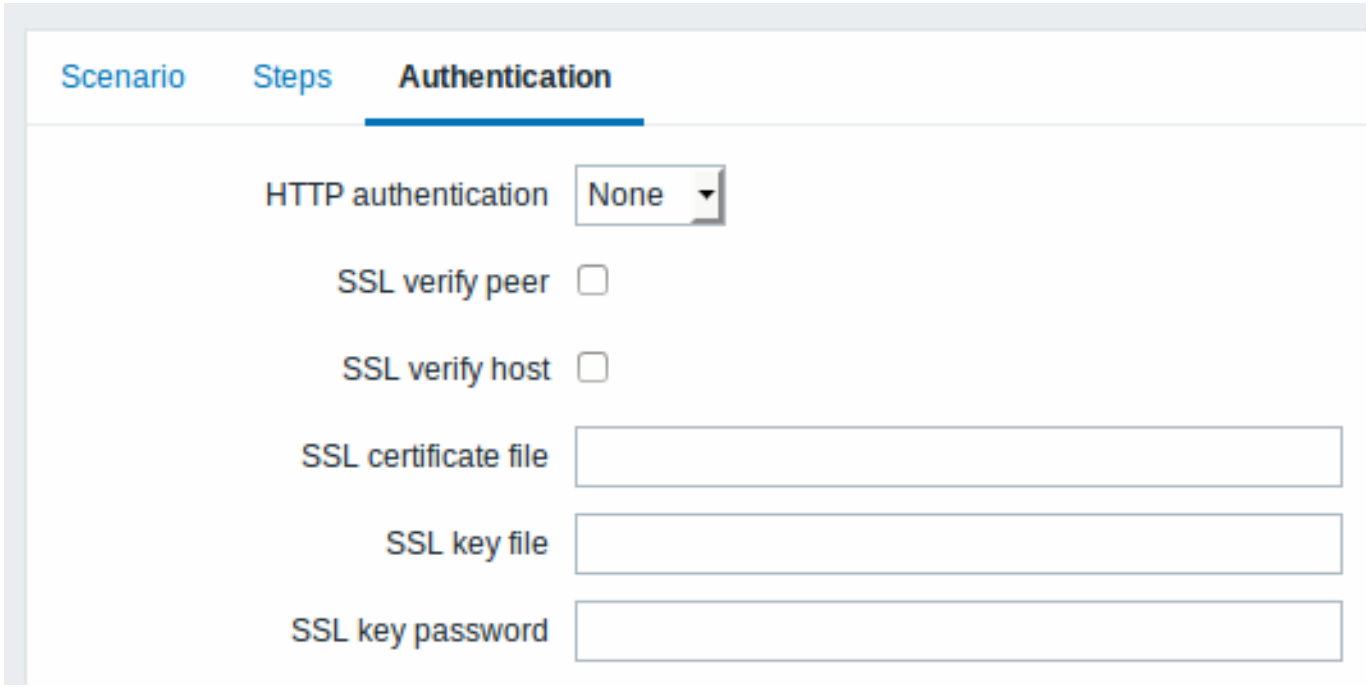
另请参见如何配置 Web 监控步骤的示例。

See also a real-life example of how web monitoring steps can be configured.

配置身份验证

Configuring authentication 身份验证选项卡允许您配置场景身份验证选项。

The **Authentication** tab allows you to configure scenario authentication options.



Authentication parameters:

认证参数：

参数说	
// 验证 (Authentication)// 验	参数。选中复选框以 **None** - 未使用身份验证。服务器证书将自基本认证 - 使用基本认证。这将设置 [CURL] **authentication** - 使用 NTLM (Windows NT LAN Manager) 身份验证。*Zabbi* 选择身份验证
	证 Web 服务器的 SSL 证书。从系统的证书颁发机构 (CA) 位置获取。您可以使用 Zabbix 服务器或代理配置参数 **NTLM** 覆盖 CA 文件的位置。PT_SSL_VERIFYP cURL 参数. 2.4* 开始支持此选项。

参数	说
SSL 验证主机 *(SSL verify host)*	选中复框以验证Web服务器证书的//公用名称(Common Name)//字段或//主题备用名称(Subject Alternate Name)//字段是否匹配。 \这将会设置CURLOPT_SSL_VERIFYHOST cURL参数。 *Zabbix 2.4*开始支持此选项。

参数说	
SSL 证书文件 *(SSL certificate file)* 用于客	端认证的SSL证书文件的名称。证书文件必须为PEM[1]格式。如果证书文件还包含私钥，请将*SSL*密钥文件*(SSL key file)*字段留空。如果密钥加密，请在*SSL*密钥密码*(SSL key pass-word)*字

参数	说
SSL 密钥文件 *(SSL key file)*	用于客端认证的SSL私钥文件的名称。私钥文件必须为PEM[1]格式。包含此文件的目录由Zab-bix服务器或代理配置参数SSLKeyLocation指定。HOST.*宏和用户宏可以在此字段中使用。这将设置CUR-

参数	说
SSL 密钥密码 *(SSL key password)* SSL	私钥文件密码。用户宏可以在此字段中使用。这将设置 CURLOPT_KEYPASSWD cURL 参数. *Zabbix 2.4* 开始支持此选项。

Parameter	Description
Authentication	Authentication options. **None** - no authentication used. **Basic authentication** - basic authentication is used. **NTLM authentication** - NTLM (Windows NT LAN Manager) authentication is used. Selecting an authentication method will provide two additional fields for entering a user name and password. User macros can be used in user and password fields, *starting with Zabbix 2.2*.
SSL verify peer	Mark the checkbox to verify the SSL certificate of the web server. The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCALocation. This sets the CURLOPT_SSL_VERIFYPEER cURL option. This option is supported *starting with Zabbix 2.4*.
SSL verify host	Mark the checkbox to verify that the *Common Name* field or the *Subject Alternate Name* field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYHOST cURL option. This option is supported *starting with Zabbix 2.4*.

Parameter	Description
SSL certificate file	Name of the SSL certificate file used for client authentication. The certificate file must be in PEM[1] format. If the certificate file contains also the private key, leave the *SSL key file* field empty. If the key is encrypted, specify the password in *SSL key password* field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLCertLocation. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLCERT cURL option. This option is supported *starting with Zabbix 2.4*.
SSL key file	Name of the SSL private key file used for client authentication. The private key file must be in PEM[1] format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLKeyLocation. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLKEY cURL option. This option is supported *starting with Zabbix 2.4*.
SSL key password	SSL private key file password. User macros can be used in this field. This sets the CURLOPT_KEYPASSWD cURL option. This option is supported *starting with Zabbix 2.4*.

Attention:
[1] Zabbix 仅支持 PEM 格式的证书和私钥文件。如果您在 PKCS # 12 格式文件（通常具有扩展名 * .p12 或 * .pfx）中具有您的证书和私钥数据，您可以使用以下命令从中生成 PEM 文件：

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Attention:
[1] Zabbix supports certificate and private key files in PEM format only. In case you have your certificate and private key data in PKCS #12 format file (usually with extention *.p12 or *.pfx) you may generate the PEM file from it using the following commands:

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Note:
Zabbix 服务器对证书的更改无需重启。

Note:
Zabbix server picks up changes in certificates without a restart.

Note:
如果在单个文件中有客户端证书和私钥，只需在 “SSL 证书文件” 字段中指定它，并将 “SSL 密钥文件” 字段留空即可。证书和密钥必须仍为 PEM 格式。组合证书和密钥很容易：

```
cat client.crt client.key > client.pem
```

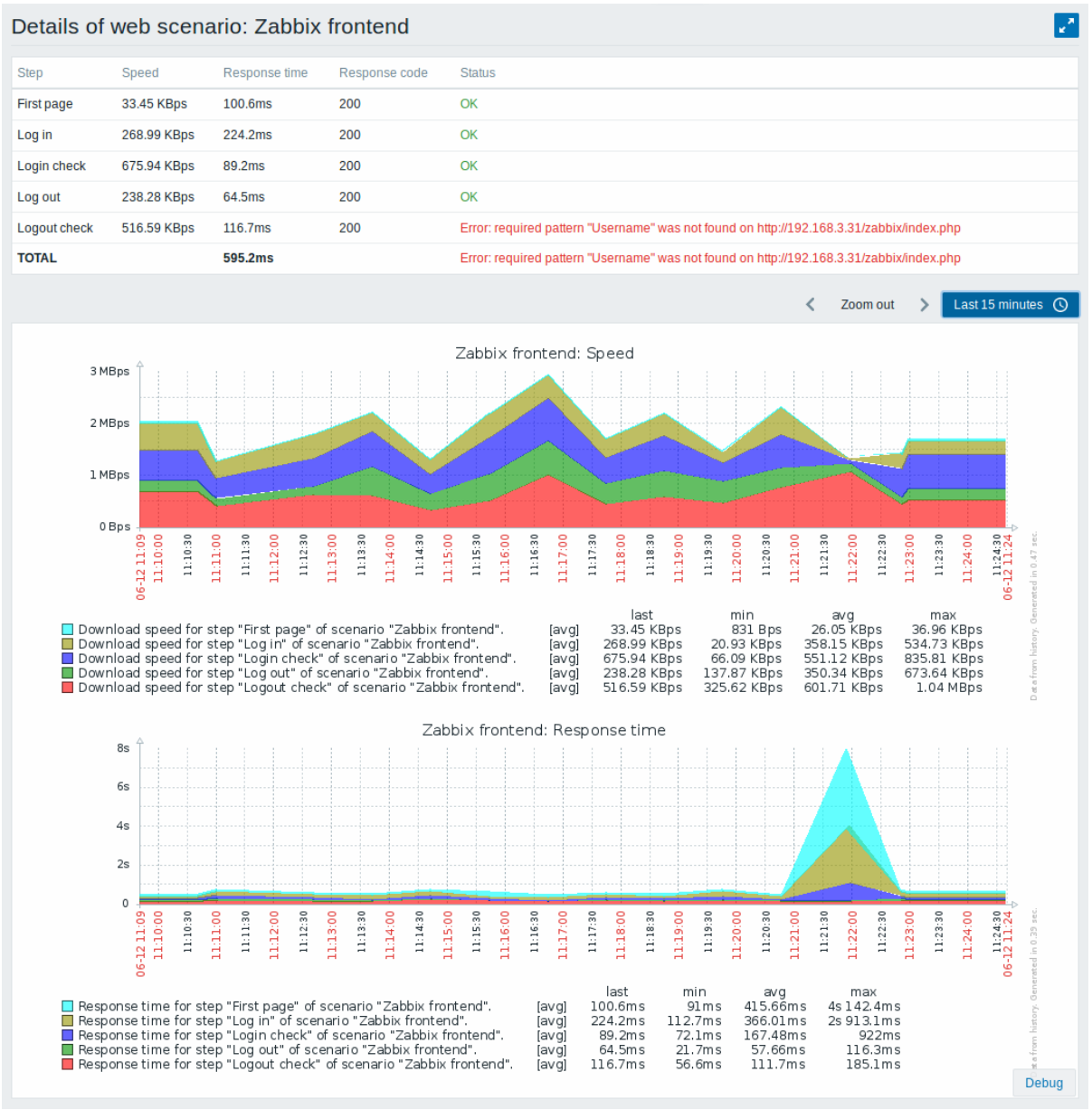
Note:
If you have client certificate and private key in a single file just specify it in a ”SSL certificate file” field and leave ”SSL key file” field empty. The certificate and key must still be in PEM format. Combining certificate and key is easy:

```
cat client.crt client.key > client.pem
```

界面

Display 要查看定义的 Web 场景的详细数据，请转到 // 监控中 (Monitoring)→Web// 或 // 最新数据 (Latest data)//。单击方案名称以查看更详细的统计信息。

To view detailed data of defined web scenarios, go to *Monitoring* → *Web* or *Latest data*. Click on the scenario name to see more detailed statistics.



可以在 // 监控中 (Monitoring)→ 仪表板 (Dashboard)// 中查看 Web 监控场景的概述。

An overview of web monitoring scenarios can be viewed in *Monitoring* → *Dashboard*.

监控扩展

Extended monitoring 有时需要记录接收的 HTML 页面内容。如果某些 Web 方案步骤失败时是非常有用的。调试级别 5（跟踪）用于此目的。此级别可以在服务端 (server) 和代理 (proxy) 代理配置文件中设置或使用运行时控制选项 (`-R log_level_increase="http poller,N"`，其中 N 是进程号)。以下示例说明如果调试级别 4 已设置，监控扩展如何启动：

```
Increase log level of all http pollers:
shell> zabbix_server -R log_level_increase="http poller"

Increase log level of second http poller:
shell> zabbix_server -R log_level_increase="http poller,2"
```

Sometimes it is necessary to log received HTML page content. This is especially useful if some web scenario step fails. Debug level 5 (trace) serves that purpose. This level can be set in server and proxy configuration files or using a runtime control option (`-R log_level_increase="http poller,N"`, where N is the process number). The following examples demonstrate how extended monitoring can be started provided debug level 4 is already set:

```
Increase log level of all http pollers:
shell> zabbix_server -R log_level_increase="http poller"
```

```
Increase log level of second http poller:
shell> zabbix_server -R log_level_increase="http poller,2"
```

如果不需要扩展 Web 监控，可以使用`-R log_level_decrease` 选项停止。

If extended web monitoring is not required it can be stopped using the `-R log_level_decrease` option.

1 Web 监控项（**items**）

1 Web monitoring items 概述

Overview

在创建 Web 场景时，会自动添加一些新监控项以进行监控。

Some new items are automatically added for monitoring when web scenarios are created.

场景监控项

Scenario items

创建场景后，Zabbix 会自动添加以下监控项进行监控，将它们链接到所选应用程序。

As soon as a scenario is created, Zabbix automatically adds the following items for monitoring, linking them to the selected application.

监控项	说明
// 场景 <Scenario> 的下载速度 // 此监控项将收	有关整个场景的下载速度(每秒字节数)的信息，即所有步骤的平均值。 监控项 key: web.test.in[Scen 类型：*Numeric(float)*

监控项说明	
// 场景 <Scenario> 的失败步骤 // 此监控项将显	场景上失败的步骤的编号。如果所有步骤成功执行，则返回0。监控项key: web.test.fail[Sce 类型：*Numeric(unsigned)*

监控项说明
// 场景 <Scenario> 的最后一个错误消息 // 此监控项返回场景的最一个错误消息文本。仅当场景具有失败步骤时，才会存储新值。如果所有步骤都正常，则不会收集新值。监控项 key: web.test.error[S 类型：*Char-ac-ter*

Item	Description
Download speed for scenario <Scenario>	This item will collect information about the download speed (bytes per second) of the whole scenario, i.e. average for all steps. Item key: web.test.in[Scenario„bps] Type: *Numeric(float)*

Item	Description
Failed step of scenario <Scenario>	This item will display the number of the step that failed on the scenario. If all steps are executed successfully, 0 is returned. Item key: web.test.fail[Scenario] Type: *Numeric(unsigned)*
Last error message of scenario <Scenario>	This item returns the last error message text of the scenario. A new value is stored only if the scenario has a failed step. If all steps are ok, no new value is collected. Item key: web.test.error[Scenario] Type: *Character*

使用实际场景名称而不是 “Scenario”。

The actual scenario name will be used instead of ”Scenario”.

Note:
添加的 Web 监控项将保留 30 天历史记录和 90 天趋势记录。

Note:
Web monitoring items are added with a 30 day history and a 90 day trend retention period.

Note:
如果场景名称以双引号开头或包含逗号或方括号，则它将在监控项键中正确引用。在其他情况下，不会执行额外的引用。

Note:
If scenario name starts with a doublequote or contains comma or square bracket, it will be properly quoted in item keys. In other cases no additional quoting will be performed.

这些监控项可用于创建触发器和定义通知条件。These items can be used to create triggers and define notification conditions.

例子 1

要创建 “Web 场景失败” 触发器，可以定义触发器表达式：

```
{host:web.test.fail[Scenario].last()}<>0
```

确保将 “Scenario” 替换为场景的真实名称。

例子 2

要在触发器名称中创建具有有用问题描述的 “Web 场景失败” 触发器，可以使用名称定义触发器：

```
Web scenario "Scenario" failed: {ITEM.VALUE}
```

和触发器表达式：

```
{host:web.test.error[Scenario].strlen()}>0 and {host:web.test.fail[Scenario].min()}>0
```

确保将 “Scenario” 替换为场景的真实名称。

例子 3

要创建 “Web application is slow” 触发器，可以定义一个触发器表达式：

```
{host:web.test.in[Scenario,,bps].last()}<10000
```

确保将 “Scenario” 替换为场景的真实名称。

Example 1

To create a ”Web scenario failed” trigger, you can define a trigger expression:

```
{host:web.test.fail[Scenario].last()}<>0
```

Make sure to replace ’Scenario’ with the real name of your scenario.

Example 2

To create a ”Web scenario failed” trigger with a useful problem description in the trigger name, you can define a trigger with name:

```
Web scenario "Scenario" failed: {ITEM.VALUE}
```

and trigger expression:

```
{host:web.test.error[Scenario].strlen()}>0 and {host:web.test.fail[Scenario].min()}>0
```

Make sure to replace 'Scenario' with the real name of your scenario.

Example 3

To create a "Web application is slow" trigger, you can define a trigger expression:

```
{host:web.test.in[Scenario,,bps].last()}<10000
```

Make sure to replace 'Scenario' with the real name of your scenario.

场景步骤项

Scenario step items

一旦创建步骤，Zabbix 会自动添加以下监控项进行监控，将它们链接到所选应用程序。

As soon as a step is created, Zabbix automatically adds the following items for monitoring, linking them to the selected application.

监控项	说明
// 场景 <Scenario> 中步骤 <Step> 的下载速度 // 此监控项将收集关于	骤的下载速度(字节每秒)的信息。监控项 key: web.test.in[Sc 类型：*Numeric(float)*

监控项说明	
// 场景 <Scenario> 中此步骤 <Step> 的响应时间 // 此监控项将收集有关步	的响应时间的信息(以秒为单位)。响应时间从请求开始计时，直到所有信息传输完毕。监控项 key: web.test.time[类型：*Numeric(float)*
// 场景 <Scenario> 的步骤 <Step> 的响应代码 // 此监控项将收集步骤	响应代码。监控项 key: web.test.rspco 类型：*Numeric(unsigne*

Item	Description
Download speed for step <Step> of scenario <Scenario>	This item will collect information about the download speed (bytes per second) of the step. Item key: web.test.in[Scenario,Step,bps] Type: *Numeric(float)*
Response time for step <Step> of scenario <Scenario>	This item will collect information about the response time of the step in seconds. Response time is counted from the beginning of the request until all information has been transferred. Item key: web.test.time[Scenario,Step,resp] Type: *Numeric(float)*
Response code for step <Step> of scenario <Scenario>	This item will collect response codes of the step. Item key: web.test.rspcode[Scenario,Step] Type: *Numeric(unsigned)*

将分别使用实际场景和步骤名称而不是 “Scenario” 和 “Step”。

Actual scenario and step names will be used instead of ”Scenario” and ”Step” respectively.

Note:
添加的 Web 监控项将保留 30 天历史记录和 90 天趋势记录。

Note:
Web monitoring items are added with a 30 day history and a 90 day trend retention period.

Note:
如果场景名称以双引号开头或包含逗号或方括号，则它将在监控项键中正确引用。在其他情况下，不会引用

Note:
If scenario name starts with a doublequote or contains comma or square bracket, it will be properly quoted in item keys. In other cases no additional quoting will be performed.

这些监控项可用于创建触发器和定义通知条件。例如，要创建一个 “Zabbix GUI 登录太慢” 触发器，可以定义一个

{zabbix:web.test.time[ZABBIX GUI,Login,resp].last()}>3

These items can be used to create triggers and define notification conditions. For example, to create a ”Zabbix GUI login is too slow” trigger, you can define a trigger expression:

```
{zabbix:web.test.time[ZABBIX GUI,Login,resp].last()}>3
```

2 Web 场景监控

2 Real life scenario 概述

Overview

本节介绍了如何使用 Web 监控的示例。

This section presents a step-by-step real-life example of how web monitoring can be used.

我们使用 Zabbix Web 监控来监控 Zabbix 的 Web 界面。我们想知道它是否可用、是否正常工作以及响应速度。为此，我们还必须使用我们的用户名和密码登录下。

Let’s use Zabbix web monitoring to monitor the web interface of Zabbix. We want to know if it is available, provides the right content and how quickly it works. To do that we also must log in with our user name and password.

场景

Scenario

第 1 步

Step 1

创建新的 Web 场景。

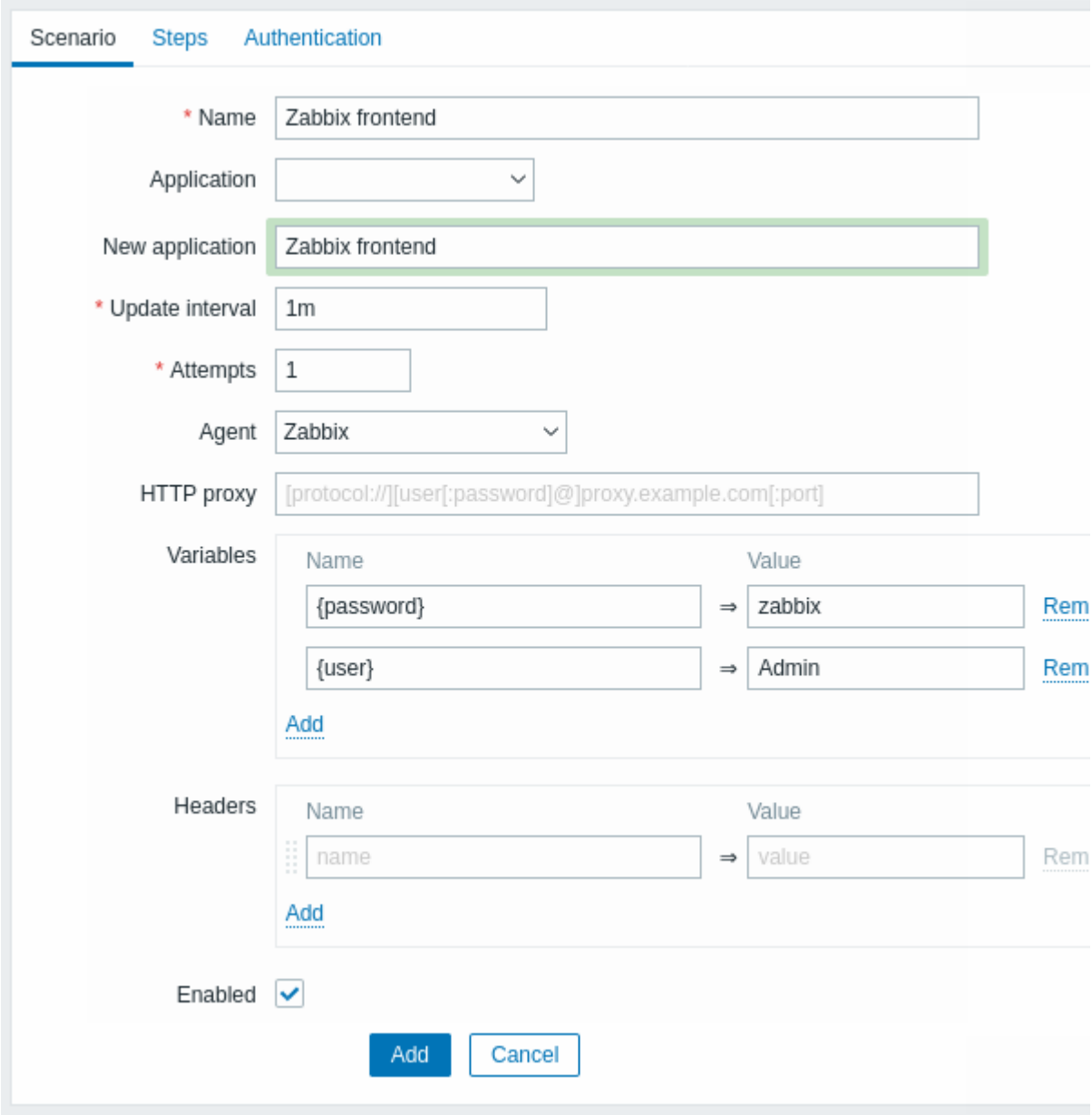
Add a new web scenario.

我们将添加一个场景来监控 Zabbix 的 Web 界面。该场景将执行多个步骤。

We will add a scenario to monitor the web interface of Zabbix. The scenario will execute a number of steps.

转到 // 配置 (Configuration) → 主机 (Hosts)//, 选择一个主机，然后在该主机行中单击 *Web*。然后单击 *Create scenario*.

Go to *Configuration* → *Hosts*, pick a host and click on *Web* in the row of that host. Then click on *Create web scenario*.



All mandatory input fields are marked with a red asterisk.

在新的场景中，我们将场景命名为 *Zabbix frontend*，并为其创建一个新的 *Zabbix frontend* 应用 (application)。

In the new scenario form we will name the scenario as *Zabbix frontend* and create a new *Zabbix frontend* application for it.

注意，我们还将创建两个变量：{user} 和 {password}。

Note that we will also create two variables: {user} and {password}.

第 2 步

Step 2

定义场景的步骤

Define steps for the scenario.

Click on *Add* button in the *Steps* tab to add individual steps.

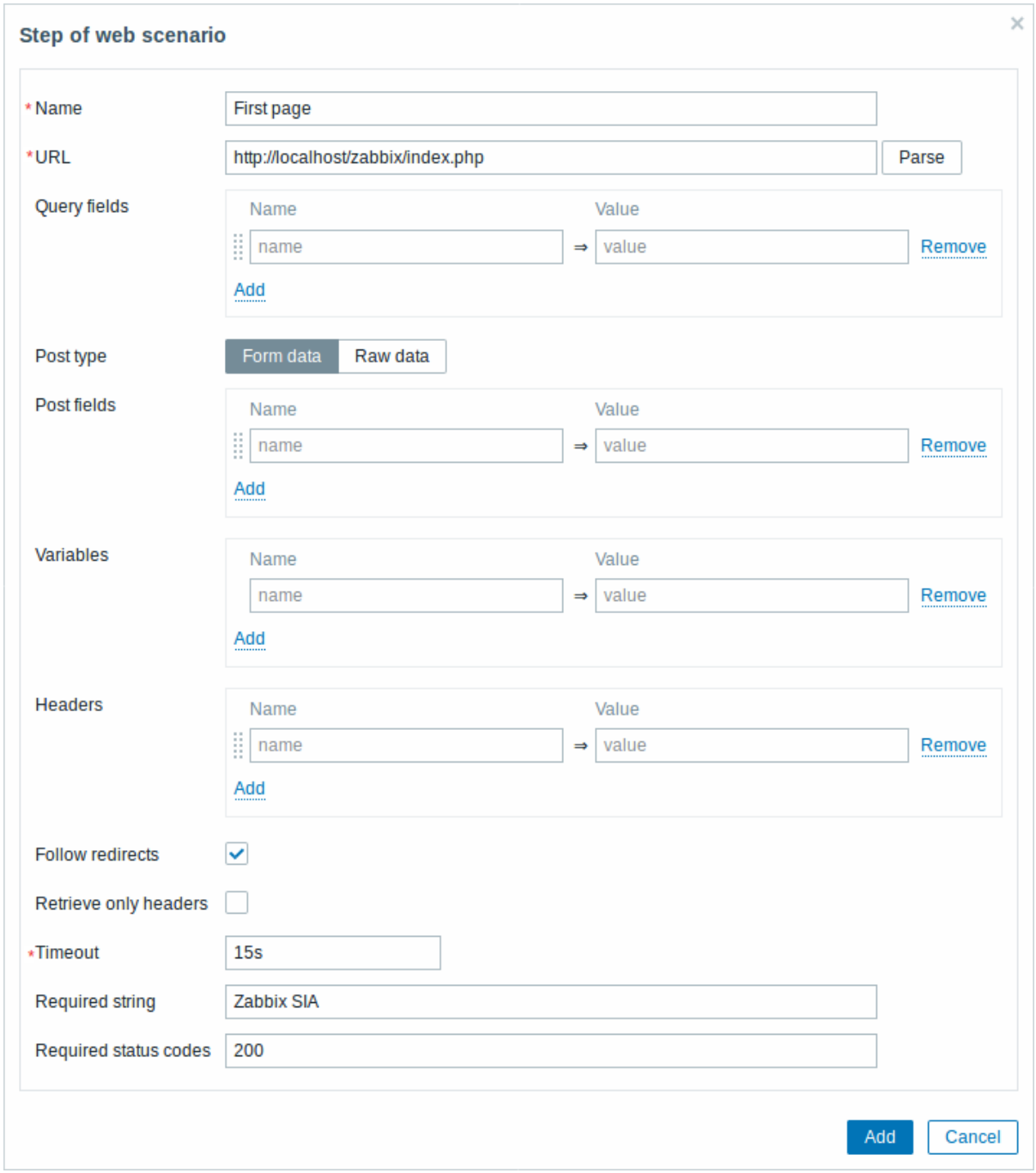
Web 场景步骤 *1*

Web scenario step 1

单击 *Steps* 选项卡中的 *Add* 按钮添加单独的步骤。

我们首先检查第一页响应是否正确，返回 HTTP 响应代码 200，并包含文本 “Zabbix SIA”。

We start by checking that the first page responds correctly, returns with HTTP response code 200 and contains text ”Zabbix SIA”.



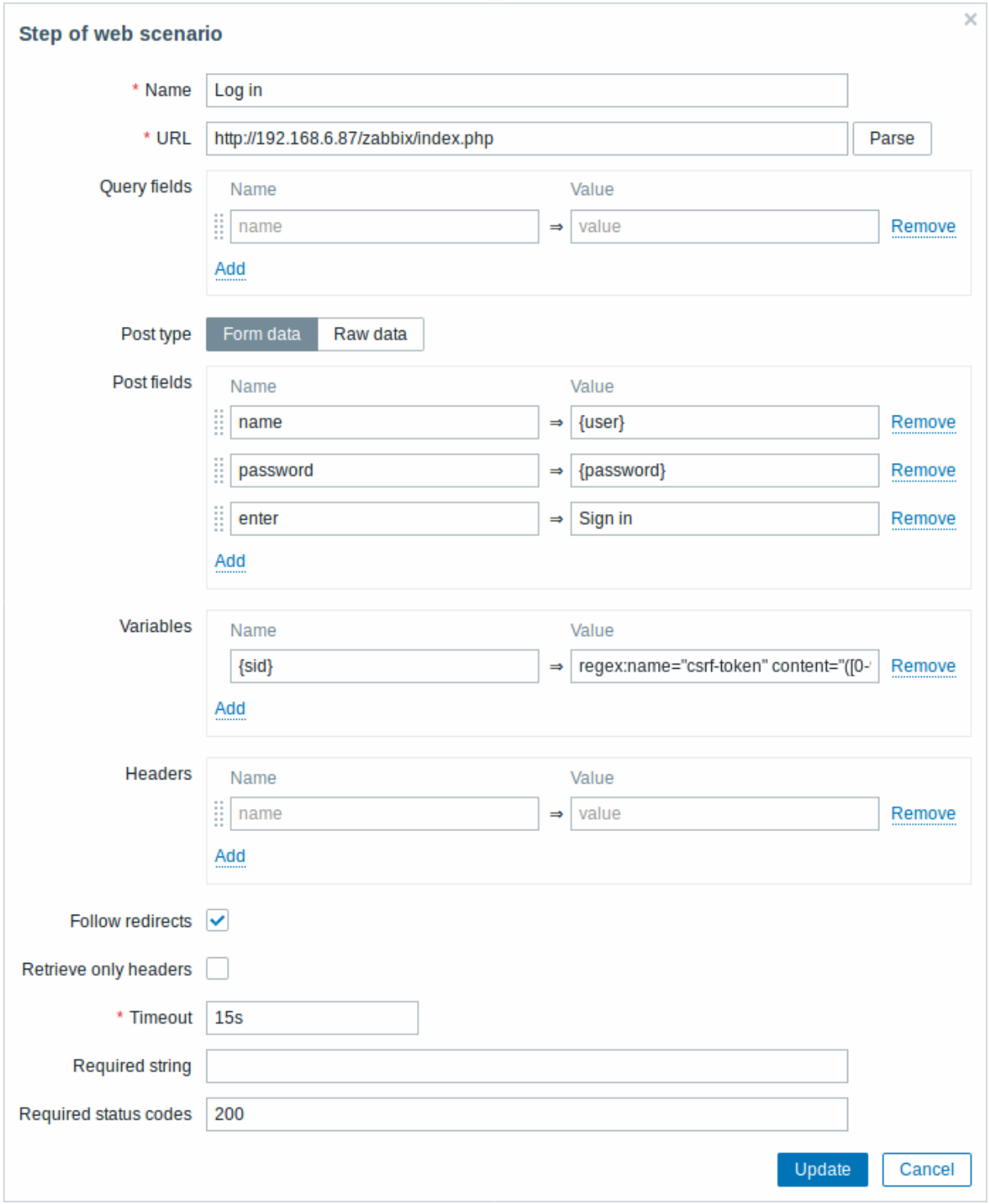
完成配置步骤后，单击 *Add*。

When done configuring the step, click on *Add*.

Web scenario step 2

我们继续登录 Zabbix 前端，我们通过我们在场景级别 {user} 和 {password} 上定义的宏（变量）来实现。

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:
注意，Zabbix 前端在登录时使用 JavaScript 重定向，因此首先我们必须登录，只有在下一步的步骤中，我们才能检查登录功能。此外，登录步骤必须使用完整的 URL 以获取 **index.php** 文件

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.

还要注意我们如何使用正则表达式的变量语法获取 `{sid}` 变量（会话 ID）的内容：`<?nowiki>?regex : name ="sid"value ="([0-9a-z] {16})"</?nowiki>`。步骤 4 中会使用此变量。

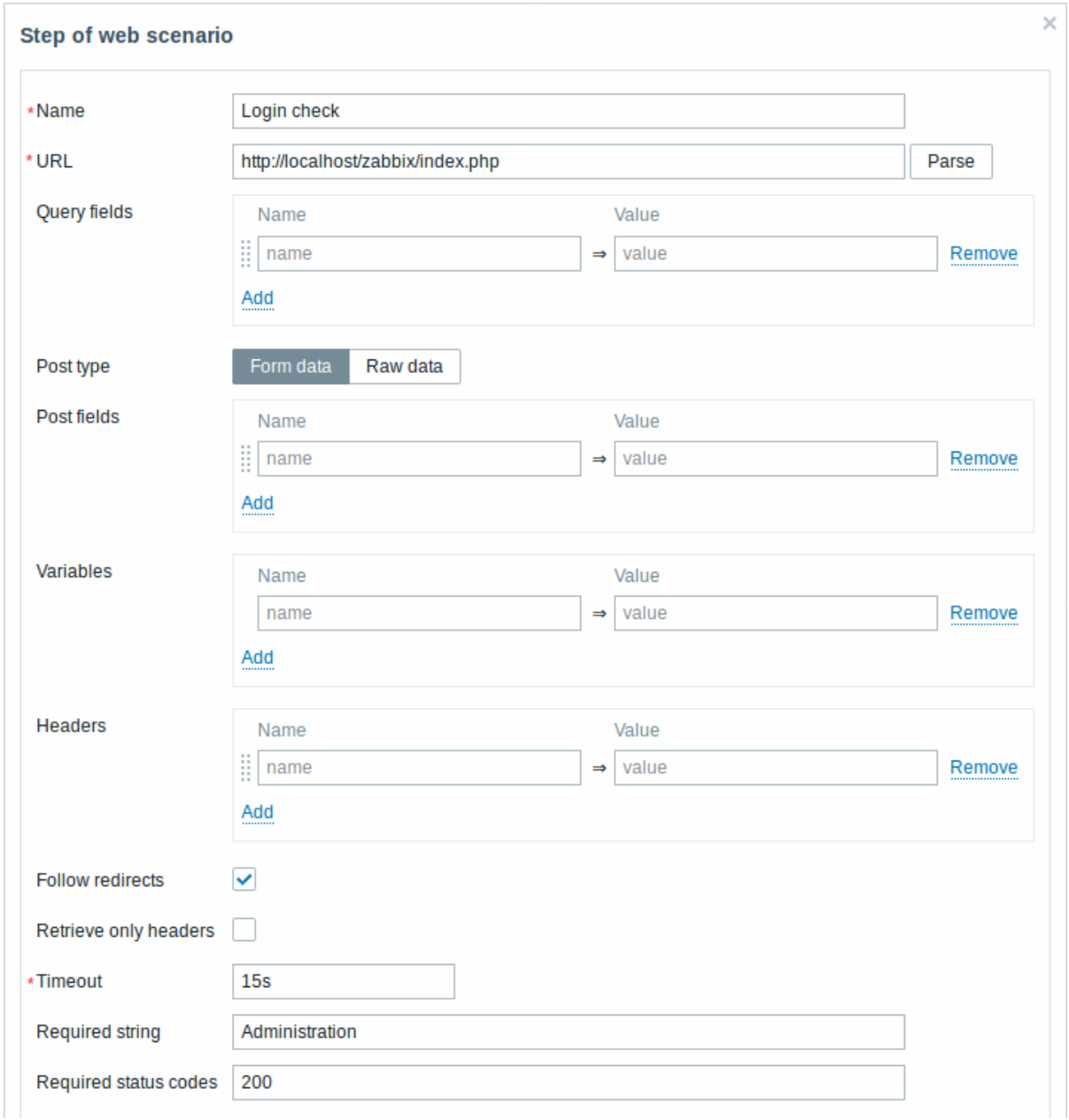
Take note also of how we are getting the content of the `{sid}` variable (session ID) using a variable syntax with regular expression: `regex:name="sid" value="([0-9a-z]{16})"`. This variable will be required in step 4.

Web 场景步骤 *3*

Web scenario step 3

登录后，我们现在应该验证一下是否登陆成功。为此，我们检查一个仅在登录后可见的字符串 - 例如 **Administration** （管理）。

Being logged in, we should now verify the fact. To do so, we check for a string that is only visible when logged in - for example, **Administration**.

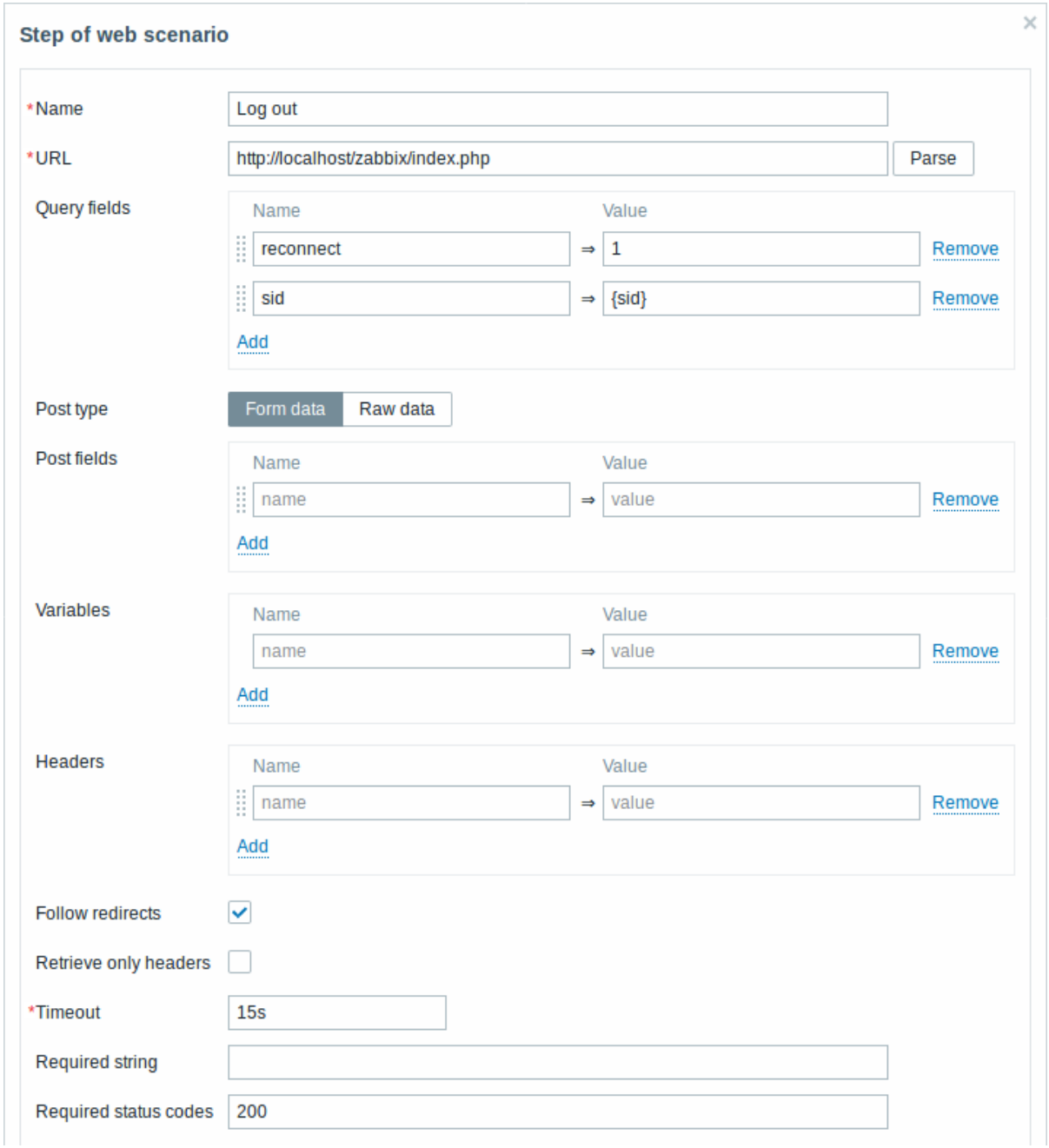


Web 场景步骤 *4*

Web scenario step 4

现在我们已经验证了前端是可访问的，我们可以登录并检索登录的内容，我们也应该注销 - 否则 Zabbix 数据库将被大量的开放会话记录所污染。

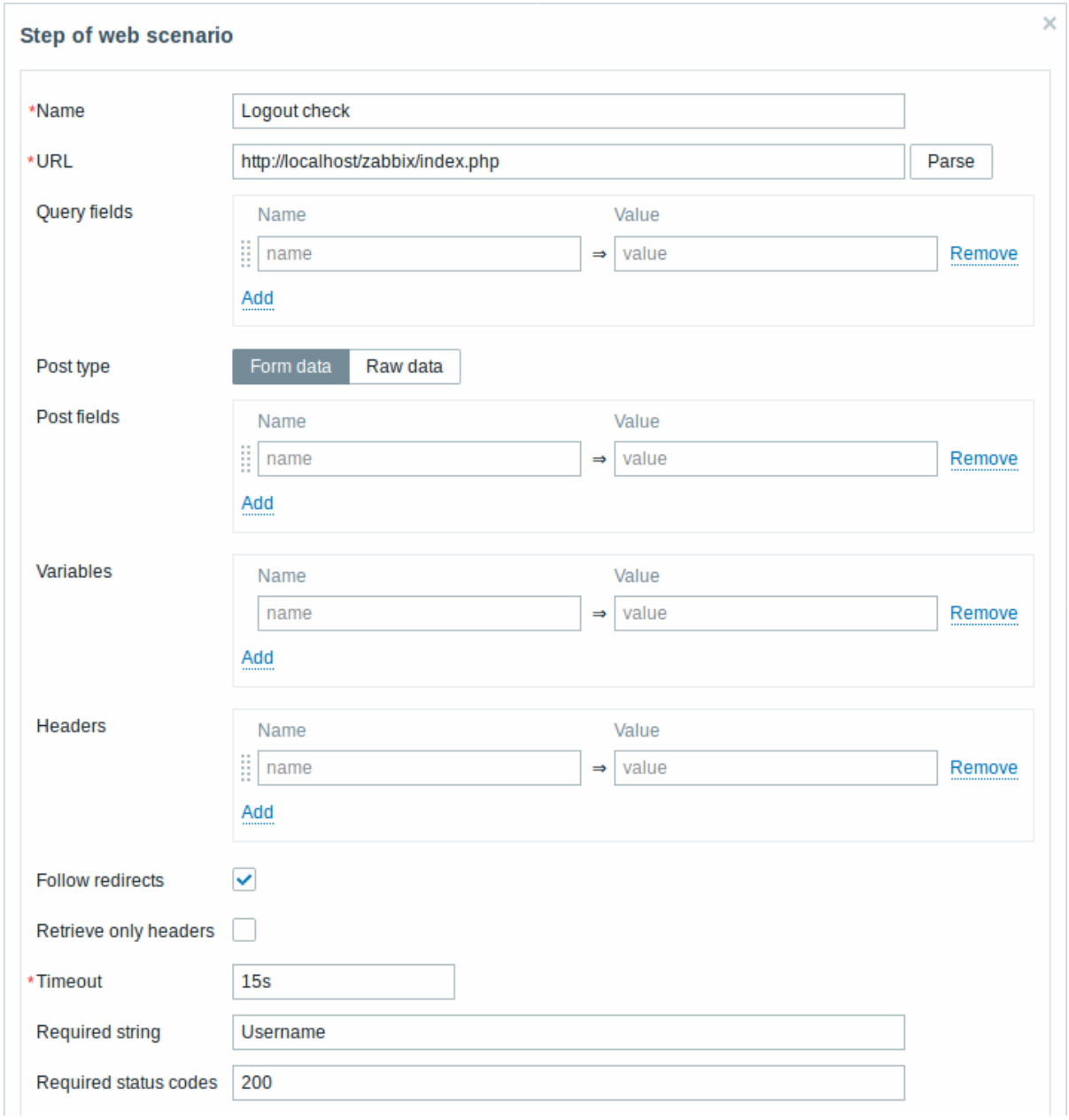
Now that we have verified that frontend is accessible and we can log in and retrieve logged-in content, we should also log out - otherwise Zabbix database will become polluted with lots and lots of open session records.



Web scenario step 5

我们可以通过查找用户名字符串来检查我们是否已经注销了。

We can also check that we have logged out by looking for the **Username** string.

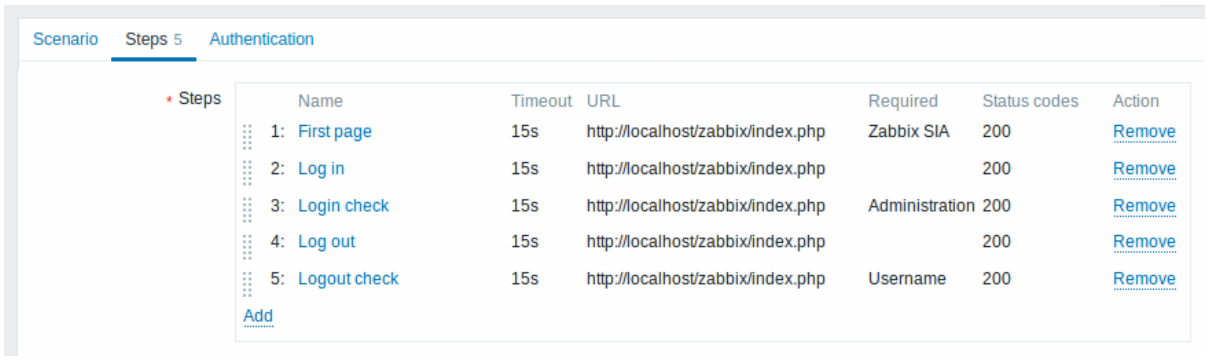


// 完成步骤配置 //

Complete configuration of steps

Web 场景步骤的完整配置应如下所示：

A complete configuration of web scenario steps should look like this:



第 3 步

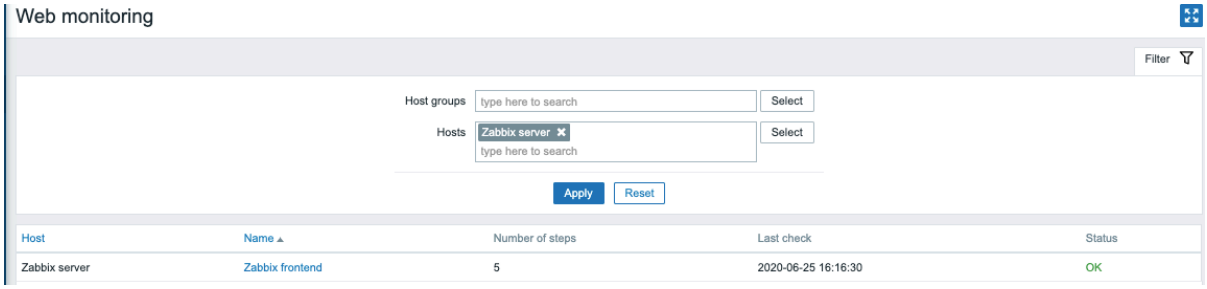
Step 3

保存 Web 监控场景。

Save the finished web monitoring scenario.

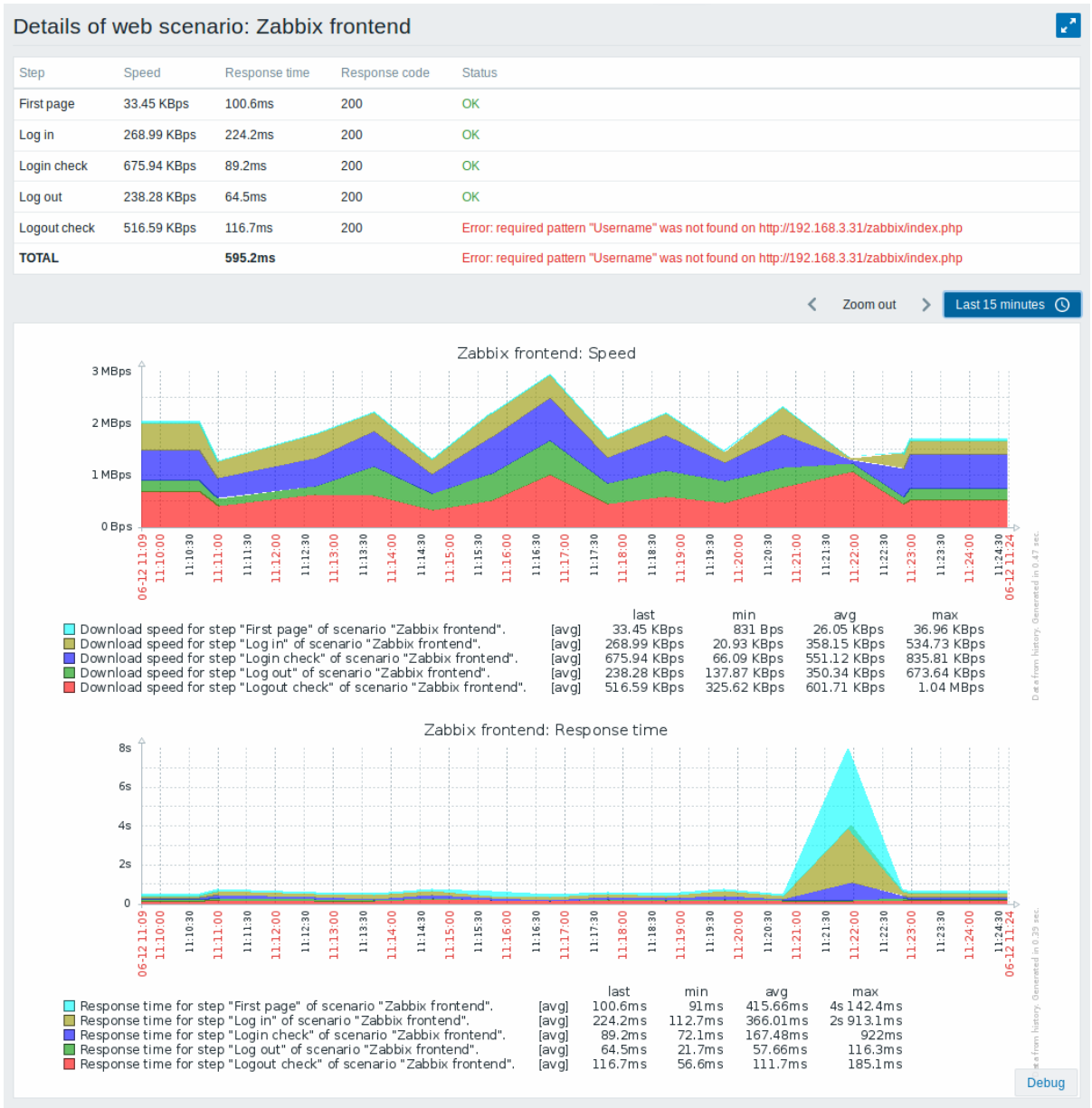
通过以下方式查看场景 *Monitoring* → *Web*:

The scenario will appear in *Monitoring* → *Web*:



单击场景名称以查看更详细的统计信息：

Click on the scenario name to see more detailed statistics:



10. 虚拟机监控

概述　从 Zabbix 2.2.0 版本开始支持对 VMware 的监控。

Zabbix 可以使用低级别自动发现（low-level discovery ）VMware hypervisors（宿主机）和虚拟机，并根据事先定义的主机原型，为这些虚拟机创建 Host 并添加监控。

Zabbix 中默认提供了几个模板，可以直接用来监控 VMware vCenter 或 ESX hypervisor。

支持 VMware vCenter 或 vSphere 版本最低为 4.1。

明细　监控虚拟机分两个步骤完成。首先，Zabbix 是通过 *vmware collector* 进程来获取虚拟机数据。这些进程通过 SOAP 协议从 VMware Web SDK 服务获取必要的信息，对其进行预处理并存储到 Zabbix server 共享内存中。然后，zabbix pollers 通过 zabbix 简单检查VMware keys 来检索这些数据。

从 Zabbix 2.4.4 开始，收集的数据分为两种类型：VMware 配置数据和 VMware 性能数据。这两种类型都由 *vmware collectors* 进程独立收集。因此，建议启用比受监控的 VMware 服务更多的收集器。否则，VMware 性能统计信息的检索可能会由于检索 VMware 配置数据而延迟（对于较大型的环境，会需要一段时间）。

目前基于 VMware 性能统计信息只有数据存储，网络接口和磁盘设备统计信息和自定义性能计数器项。

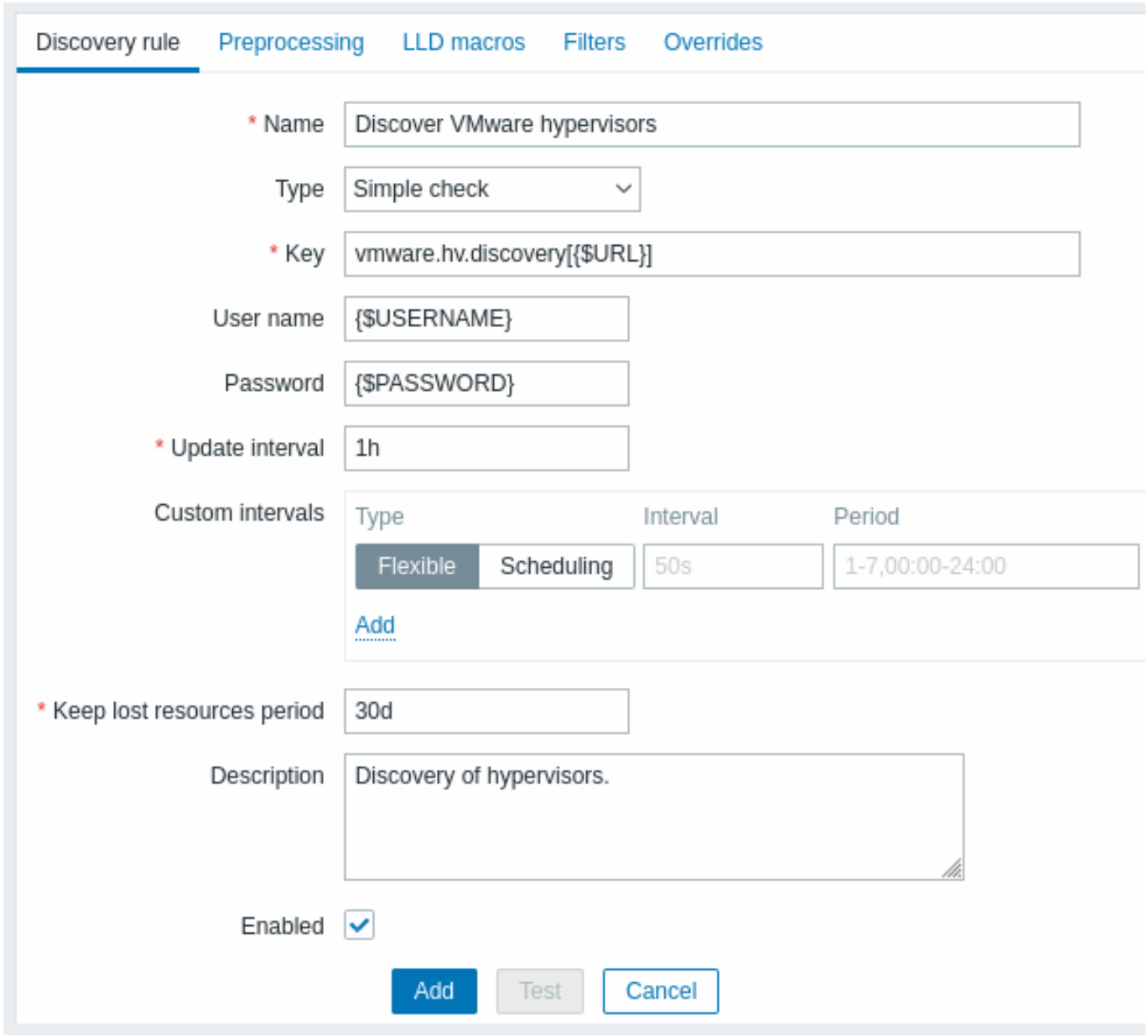
配置　要使虚拟机监控正常工作，编译安装 Zabbix 时应加上 --with-libxml2 和 --with-libcurl 编译类库选项。

以下配置文件参数可用于调整虚拟机监控：

- **StartVMwareCollectors** - 预先启动 Vmware collector 收集器实例的数量。
  此值取决于要监控的 VMware 服务的数量。在大多数情况下，这应该是：
  *servicenum < StartVMwareCollectors < (servicenum * 2)*
  其中 *servicenum* 是 VMware 服务的数量。例如：如果您有 1 个 VMware 服务要将 StartVMwareCollectors 设置为 2，那么如果您有 3 个 VMware 服务，请将其设置为 5。请注意，在大多数情况下，此值不应小于 2，不应大于 VMware 数量的 2 倍服务。还要记住，此值还取决于 VMware 环境大小和 *VMwareFrequency* 和 *VMwarePerfFrequency* 配置参数（请参阅下文）。
- **VMwareCacheSize** - 用于存储 VMware 数据的缓存容量，默认为 8M，取值范围：256K-2G。
- **VMwareFrequency** - 连接到 VMware 服务收集一个新数据的频率，默认为 60 秒，取值范围：10-86400。
- **VMwarePerfFrequency** - 连接到 VMware 服务收集性能数据的频率，默认为 60 秒，取值范围 10-86400。
- **VMwareTimeout** - VMware collector 等待 VMware 服务响应的时间，默认为 10 秒，取值范围：1-300。

有关更多详细信息，请参阅server 和proxy 的配置文件页面。

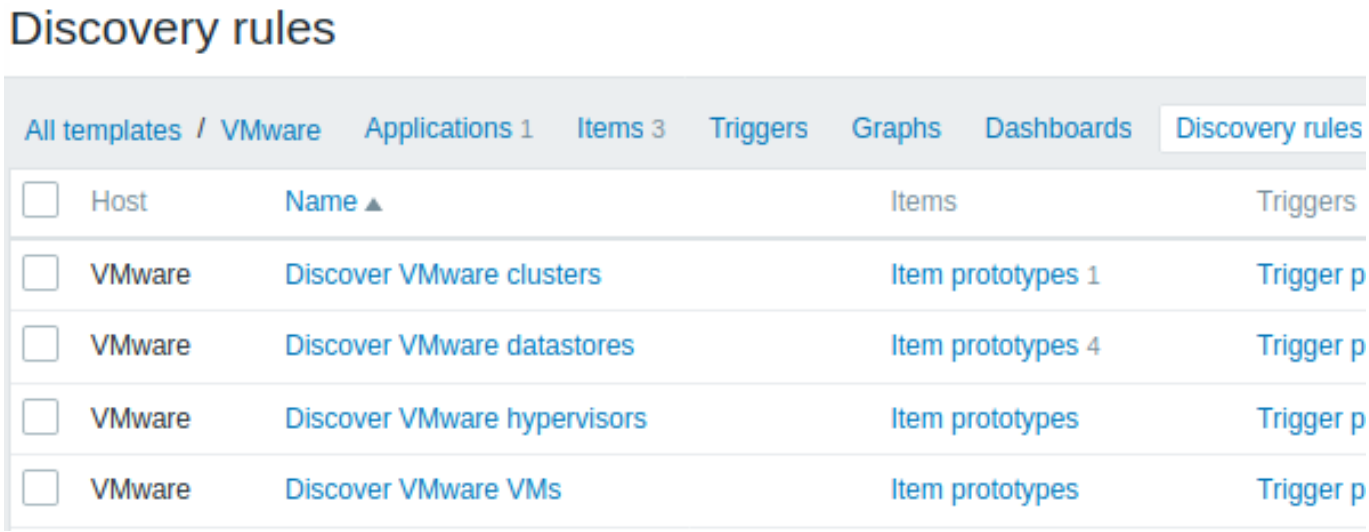
自动发现　Zabbix 可以使用低级别发现 (low-level discovery) 规则自动发现 VMware hypervisors(宿主机) 和虚拟机。



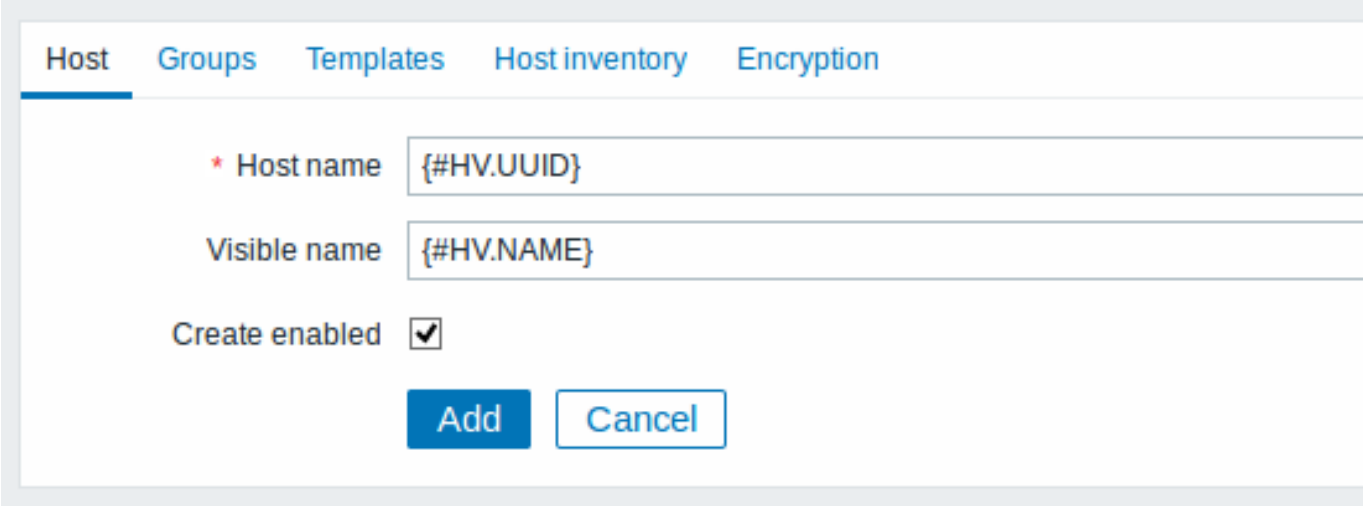
所有强制输入字段都用红色的星号标记。

以上截图中的发现规则 key 值是 *vmware.hv.discovery[{$URL}]*。

主机原型　可以使用低级别发现 (low-level discovery) 规则自动发现并创建 VMware hypervisors（宿主机）原型。当自动发现虚拟机时，这些 VMware hypervisors（宿主机）原型会关联宿主机内真实的虚拟机主机。监控主机原型在被发现之前，除了来自链接模板的监控项和触发器，不能有自己的监控项和触发器。发现的主机将属于一个现有的主机，并将根据获取的现有主机 IP 进行主机配置。



在主机原型配置中，低级别自动发现宏 (LLD macros) 用于主机名，显示名称和主机组原型字段。关联现有主机组，模板链接和加密链接等可配置选项。



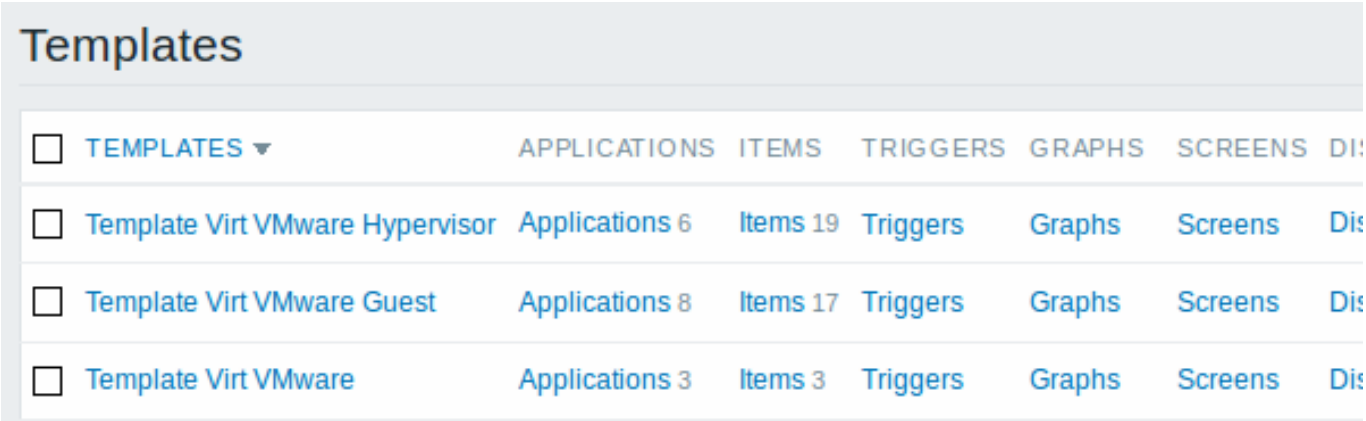
如果选中 *Create enabled*，则主机将添加为启用状态。如果未选中，将添加主机，但不启用处于禁用状态。

在主机列表中，自动发现的主机将根据它们创建的发现规则名称命名前缀。可以手动删除发现的主机。发现的主机也将根据发现规则的 // Keep lost resources period（以天为单位）// 值自动删除。除了启用 / 禁用主机和主机清单外，大多数配置选项都是只读的。发现的主机不能有自己的主机原型。

可以使用的模板　Zabbix 中默认提供了几个现成的模板，用于监控 VMware vCenter 或 ESX hypervisor。

这些模板包含事先定义的 LLD 规则以及用于监视虚拟安装的内置检查。

请注意，"*Template Virt VMware*" 监控模板应用于 VMware vCenter 和 ESX hypervisor（宿主机）监控。"*Template Virt VMware Hypervisor*" 和"*Template Virt VMware Guest*" 模板由前者自动发现关联宿主机和虚拟机，通常不应该手动链接到单个主机。



Note:
如果您的 Zabbix 从 2.2 之前的版本升级并且没有此类模板，您可以手动导入，从社区页面下载 官方模板。默认这些模板依赖于 *VMware VirtualMachinePowerState* 和 //VMware 状态值 // 映射，因此有必要首先创建这些值映射（使用 SQL 脚本，手动或从 XML 导入）

主机配置　要使用 VMware 简单检查，主机必须定义以下用户宏：

- **{$URL}** - VMware 服务 (vCenter or ESX hypervisor) SDK URL (https://servername/sdk).
- **{$USERNAME}** - VMware 服务用户名
- **{$PASSWORD}** - VMware 服务 {$ USERNAME} 用户密码

例子　以下示例演示如何在 Zabbix 上快速配置 VMware 监控：

- 编译安装 zabbix server 时添加依赖项（--with-libxml2 和 --with-libcurl）
- 将 Zabbix server 配置文件中的 StartVMwareCollectors 选项设置为 1 或更多
- 创建新主机
- 设置监控 VMware 服务所需的身份验证相关的主机宏：

{{..:..:assets:en:manual:vm_monitoring:vm_host_macros.png|}}
* 将 VMware 服务模板链接到主机：
{{..:..:assets:en:manual:vm_monitoring:vm_host_templates.png|}}
* 单击 //Add// 按钮保存主机

扩展日志 使用调试级别 5 进行详细调试时，VMware 收集器收集的数据会记录到日志中。此级别可以在server 和proxy 配置文件中设置，或使用运行时控制选项（-R log_level_increase="vmware collector,N"，其中 N 是过程编号）。以下示例说明如果配置将调试级别设置为 4 ，扩展日志启动：

提高所有 vmware 收集器的日志级别：

```
shell> zabbix_server -R log_level_increase="vmware collector"
```

提高第二个 vmware 收集器的日志级别：
```
shell> zabbix_server -R log_level_increase="vmware collector,2"
```

如果不需要对 VMware 收集器数据进行扩展日志，可以使用-R log_level_decrease 选项进行停止。

故障排查

- 如果 Vmware 监控失败，监控项不可用，请确认使用 VMware vSphere 较新版本，它们监控项是否不可用或默认被关闭，是否限制 Zabbix 访问 VMware 性能计数器数据库等配置。详情请参阅 ZBX-12094

1 虚拟机发现 **key** 值字段

下表列出了虚拟机发现 key 值返回的内容。

项目键		
描述 *	字段 ** ** 检	内容 **
vmware.cluster.discovery 执行 VMware 集群发现 discovery, 用于发现 Hypervisorv（宿主机）。	{#CLUSTER.ID}	集群 ID。
	{#CLUSTER.NAME}	集群名称。
vmware.hv.discovery 执行 hypervisor 的 discovery 。	{#V.UUID}	hypervisor 的 ID 唯一的
	{#HV.ID}	Hypervisorv（宿主机）的 ID（由 Host-System 管理）
	{#HV.NAME}	Hypervisor（宿主机）的名字
	{#CLUSTER.NAME}	VMware 群集名称，可能为空。
	{#DATACENTER.NAME}	VMware 数据中心名称。
vmware.hv.datastore.discovery 执行 hypervisor 数据存储库的 discovery 。请注意，多个 hypervisor 可以使用相同的数据存储 (datastore)。	称。	
{#DATASTORE} 数据存储		
vmware.vm.discovery		

项目键		
执行虚拟机的 discovery 。{#VM.	UID} 唯一虚拟机	D。
	{#VM.ID}	虚拟机 ID（由 Virtual-Machine 管理）。
	{#VM.NAME}	虚拟机名。
	{#HV.NAME}	Hypervisor（宿主机）名称。
	{#CLUSTER.NAME}	VMware 集群名称，可能为空。
	{#DATACENTER.NAME}	VMware 数据中心名称。
vmware.vm.net.if.discovery 执行虚拟机网络接口的 discovery 。{#IFNAME}		网络接口名称。
vmware.vm.vfs.dev.discovery 执行虚拟机磁盘设备的 discovery 。{#DISKNAM	}	磁盘设备名称。
vmware.vm.vfs.fs.discovery 执行虚拟机文件系统的 discovery 。{#FSNAME}		文件系统名称。

11. 维护

概述　在 Zabbix 里，可以为主机和主机群组定义维护期。有两种维护类型可选：一种是有数据收集；另一种是无数据收集。

在 “有数据收集” 的维护期里，触发器像往常一样正常处理，事件也会在需要的时候被创建。然而，对于正处在维护期的主机来说，如果在动作配置里勾选了 维护期暂停操作（*Pause operations while in maintenance*）选项，那么问题升级会被暂停。在这种情况下，只要维护期间持续，可能包含的发送通知或者远程命令的升级步骤会被忽略。

比如，有三个升级步骤原计划是在问题发生后的第 0 分钟、30 分钟和 60 分钟分别执行。现在定义一个半小时的维护期，持续时间刚好是从问题发生后的第 10 分钟到第 40 分钟。那么受维护期的影响，原计划在第 30 分钟和 60 分钟执行的步骤会被推迟半个小时。也就是说，步骤二会在问题发生后的第 60 分钟执行，步骤三会在问题发生后的第 90 分钟执行（假设问题仍然存在）。类似的，如果在维护期发生问题，那么问题升级会在维护期结束后开始。

如果需要在维护期间正常接收问题通知（没有延迟），必须在动作配置里取消勾选 维护期暂停操作（*Pause operations while in maintenance*）选项。

> **Note:**
> 只要有一个主机（触发器表达式中使用到的主机）不在维护模式里，Zabbix 就可能会发送问题通知。

在维护期间，Zabbix server 必须处于运行状态。Timer 进程负责在每分钟的第 0 秒进行主机是否处于维护状态的切换。Zabbix proxy 节点不论在什么维护类型（包含 “无数据收集” 维护）下都会收集数据。只不过如果是 “无数据收集” 类型，这些数据后来会被 Zabbix server 节点忽略。

当“无数据收集”维护期间刚结束的时候，使用了 nodata() 函数的触发器不会被触发。这些触发器在下一次检查以后才可能会被触发。

如果在主机处于维护状态的时候添加了一个日志相关的监控项，那么当维护结束时，只会收集自维护结束以来的新日志文件内容。

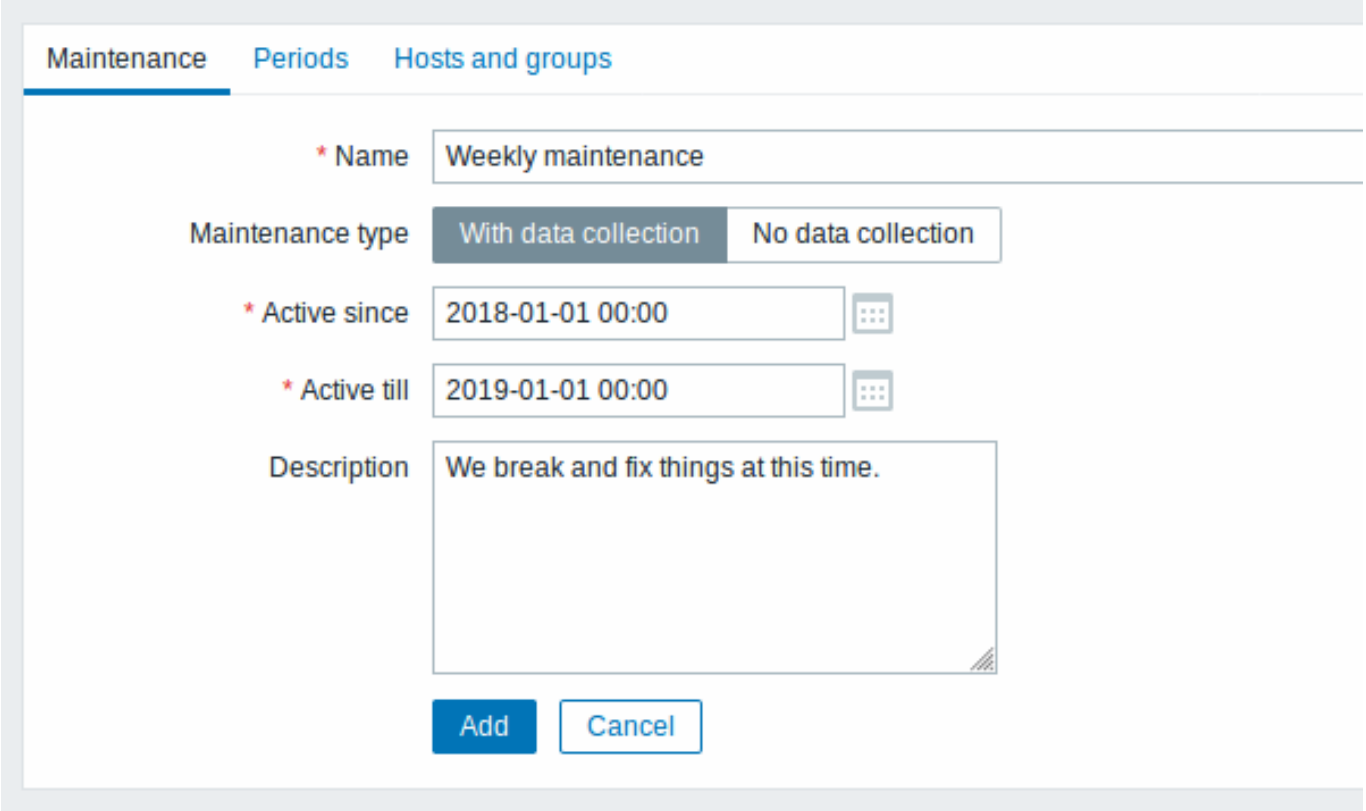
如果主机处于“无数据收集”维护类型期间，此时给它发送一个带时间戳的值（例如，使用Zabbix sender），那么这个值会被丢弃。然而，如果主机的维护期过期了，此时给它发送一个带有时间戳的值，是会被接收的。

<note important> 为确保重复性维护期（每天，每周，每月）的行为在的预期之中，Zabbix 的所有部件都应该使用相同的时区。:::

配置　配置维护期：

- 切换到：配置（*Configuration*）→ 维护（*Maintenance*）
- 单击 创建维护期间（*Create maintenance period*）（或者单击已存在的维护期的名称）

维护（**Maintenance**）选项卡包含了常见的维护期属性：



所有必填输入字段都标有红色星号。

参数说	
名称（*Name*）维护期	的名称。

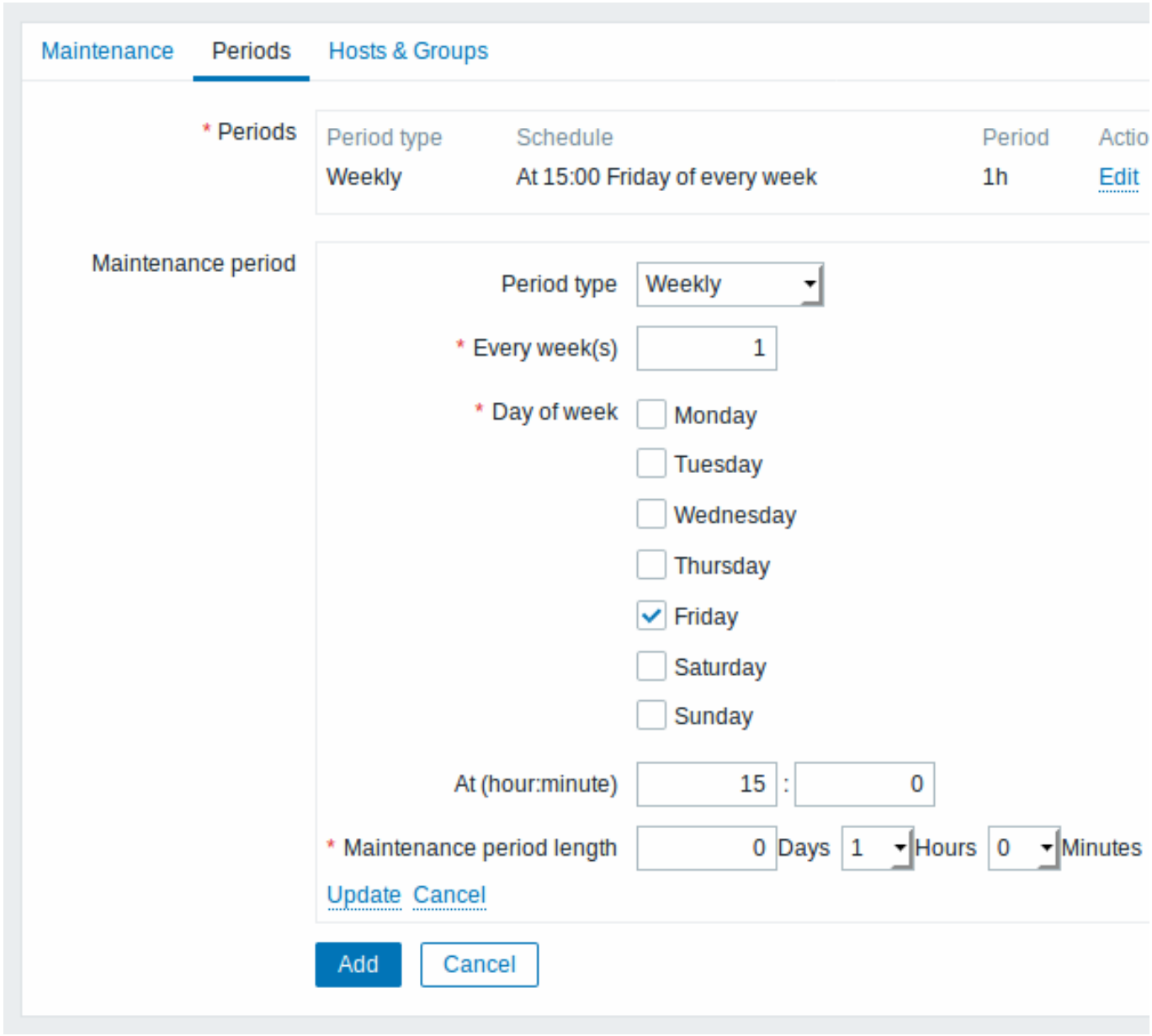
参数说	
维护类型（*Maintenance type*）有两种维护	型可以设置：有数据收集（**With data collection**）- 在维护期间数据会被server收集，触发器也会被处理无数据收集（**No data collection**）- 在维护期间数据不会被收集

参数	说
启用自从（*Active since*）执行维护期	的日期和时间变为活动状态。 注意：单独设置这个时间并不能激活维护期间；需要切换到期间（*Periods*）选项卡进行操作。

参数	说
启用直到（*Active till*）执行维护期	的日期和时间停止处于活动状态。
描述（*Description*）维护期	的描述。

The **Periods** tab allows you to define the exact days and hours when the maintenance takes place. Clicking on *New* opens a flexible *Maintenance period* form where you can define the times - for daily, weekly, monthly or one-time maintenance.

期间（**Periods**）选项卡允许您定义维护发生的确切天数和小时数。单击 新建（*New*）会打开一个 维护期间（*Maintenance period*）表单，可灵活配置维护期间的时间段 - 每天、每周、每月或者仅一次。



Daily and weekly periods have an *Every day/Every week* parameter, which defaults to 1. Setting it to 2 would make the maintenance take place every two days or every two weeks and so on. The starting day or week is the day or week that *Active since* time falls on.

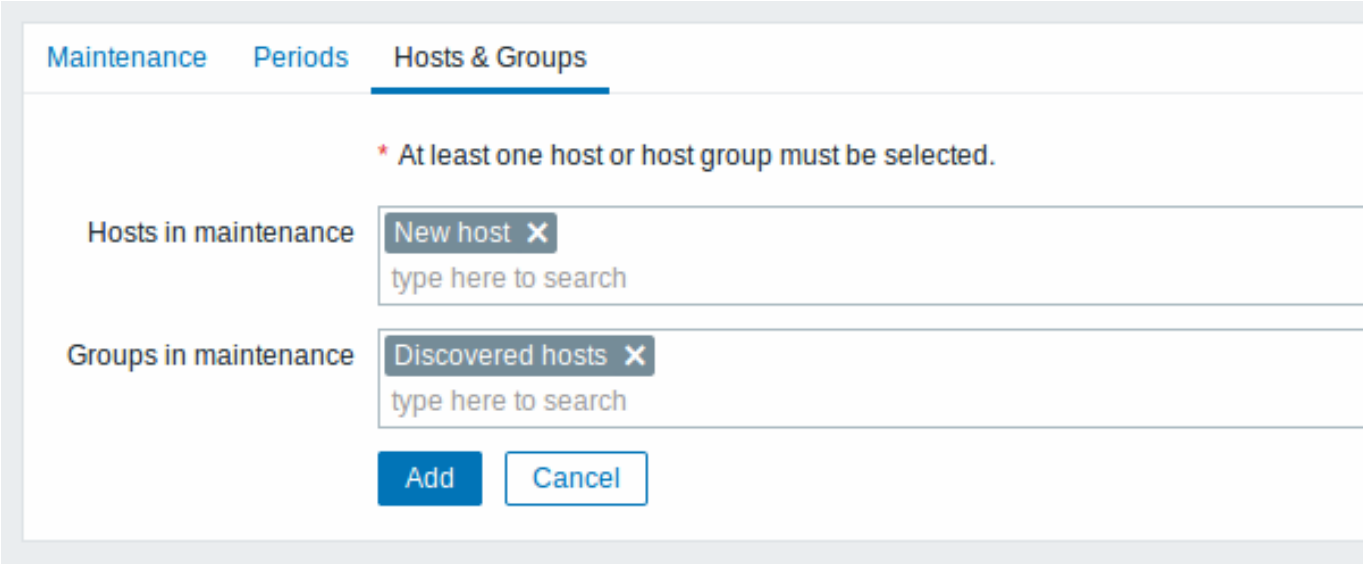
每天和每周期间有一个 每天（*Every day*）/每周（*Every week*）参数，默认值是 1。如果设置为 2，那么维护期间就是每两天或者每两周执行一次，以此类推。起始日期或星期是 启用自从（*Active since*）时间起作用时的日期或星期。

For example, having *Active since* set to 2013-09-06 12:00 and an hour long daily recurrent period every two days at 23:00 will result in the first maintenance period starting on 2013-09-06 at 23:00, while the second maintenance period will start on 2013-09-08 at 23:00. Or, with the same *Active since* time and an hour long daily recurrent period every two days at 01:00, the first maintenance period will start on 2013-09-08 at 01:00, and the second maintenance period on 2013-09-10 at 01:00.

比如，启用自从（*Active since*）设置为 2013-09-06 12:00，如果有一个在 23:00 开始的为期一个小时的维护期间，每两天执行一次，那么第一次维护期间将会开始于 2013-09-06 23:00，第二次维护期间开始于 2013-09-08 23:00。或者，再举个例子，如果还是那个相同的启用自从（*Active since*），每两天执行一次，每次一小时，开始时间设定为 01:00，那么，第一次维护期间将开始于 2013-09-08 01:00，第二次开始于 2013-09-10 01:00。

The **Hosts & Groups** tab allows you to select the hosts and host groups for maintenance.

主机（**Hosts**）**&** 主机组（**Groups**）选项卡允许选择需要维护的主机和主机组。



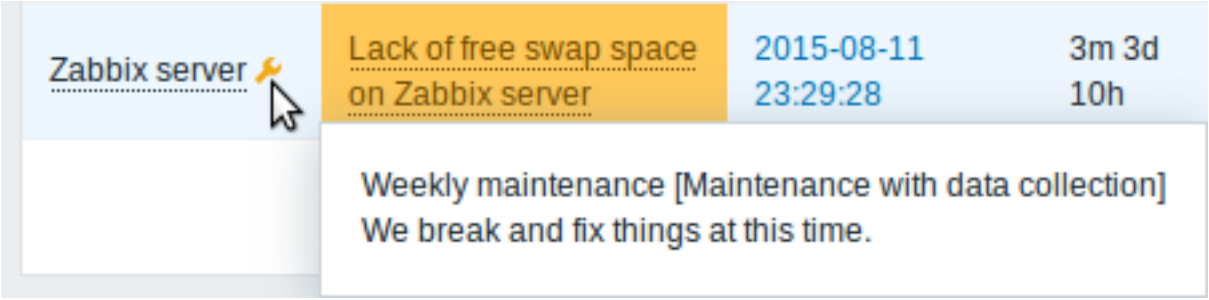
Specifying a parent host group implicitly selects all nested host groups. Thus the maintenance will also be executed on hosts from nested groups.

如果选择了某个父主机组，那么会隐式的选中其所有内嵌的主机组。因此，维护也将在内嵌的主机组的主机上执行。

Display

显示 An orange wrench icon next to the host name indicates that this host is in maintenance in the *Monitoring* → *Dashboard* and *Inventory* → *Hosts* → *Host inventory* sections.

主机名称旁边的橙色扳手图标表示该主机正处于维护状态。在 监测中（*Monitoring*）→ 仪表板（*Dashboard*）以及 资产记录（*Inventory*）→ 主机（*Hosts*）→ 主机资产记录（*Host inventory*）页面，都可能看到这个维护标志。



Maintenance details are displayed when the mouse pointer is positioned over the icon.

当鼠标指针停留在扳手图标上面的时候会显示维护的详细信息。

Note:
The display of hosts in maintenance in the Dashboard can be unset altogether with the dashboard filtering function.

Note:
可以使用仪表板过滤功能完全取消显示仪表板中处于维护状态的主机。

Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps* and in *Configuration* → *Hosts* their status is displayed as 'In maintenance'.

此外，维护中的主机在 监测中（*Monitoring*）→ 拓扑图（*Maps*）中获得橙色背景，在 配置（*Configuration*）→ 主机（*Hosts*）中其状态显示为 "维护中（In maintenance）"。

12. 正则表达式 [杨青]

概述 Perl Compatible Regular Expressions (PCRE) are supported in Zabbix.

Zabbix 支持 Perl Compatible Regular Expressions (PCRE)。

There are two ways of using regular expressions in Zabbix:

在 Zabbix 中有两种使用正则表达式的方法：

- manually entering a regular expression
- using a global regular expression created in Zabbix
- 手动输入正则表达式
- 使用在 Zabbix 中创建的全局正则表达式

Regular expressions

正则表达式 You may manually enter a regular expression in supported places. Note that the expression may not start with @ because that symbol is used in Zabbix for referencing global regular expressions.

可以在支持的位置手动输入正则表达式。请注意，表达式可能不以 @ 开头，因为该符号在 Zabbix 中用于引用全局正则表达式。

Global regular expressions

全局正则表达式 There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

在 Zabbix 前端，有一个高级的编辑器用于创建和测试复杂的正则表达式。

Once a regular expression has been created this way, it can be used in several places in the frontend by referring to its name, prefixed with @, for example, *@mycustomregexp*.

一旦以这种方式创建了正则表达式，它就可以在前端的几个地方使用，方法是加个 @ 前缀来引用它的名称，例如，*@mycustomregexp*。

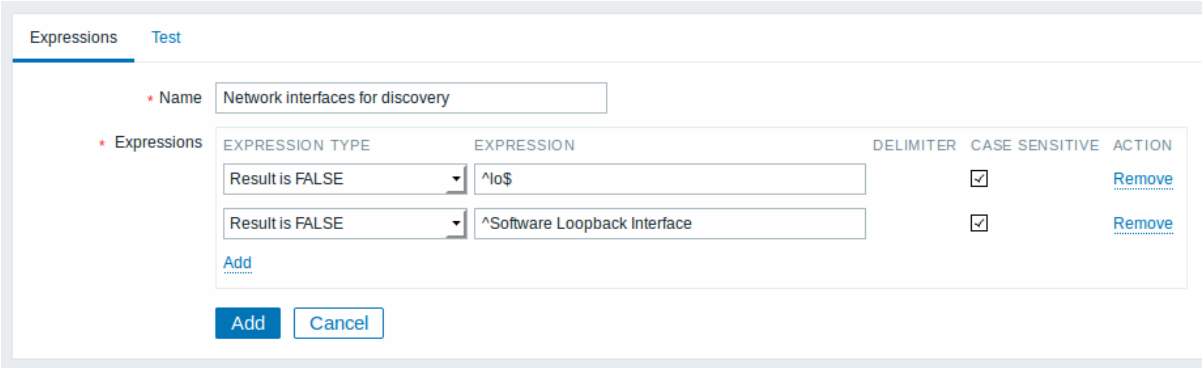
To create a global regular expression:

要创建全局正则表达式：

- Go to: *Administration* → *General*
- Select *Regular expressions* from the dropdown
- Click on *New regular expression*
- 切换到：管理（*Administration*）→ 一般（*General*）
- 从右上角的下拉列表中选择 正则表达式（*Regular expressions*）
- 单击 新的正则表达式（*New regular expression*）

The **Regular expressions** tab allows to set the regular expression name and add subexpressions.

正则表达式（**Regular expressions**）选项卡允许设置正则表达式名称并添加子表达式。



All mandatory input fields are marked with a red asterisk.

所有必填输入字段都标有红色星号。

Parameter	Description
Name	Set the regular expression name. Any Unicode characters are allowed.
Expressions	Click on *Add* in the Expressions block to add a new subexpression.

Parameter		Description
	Expression type	Select expression type: **Character string included** - match the substring **Any character string included** - match any substring from a delimited list. The delimited list includes a comma (,), a dot (.) or a forward slash (/). **Character string not included** - match any string except the substring **Result is TRUE** - match the regular expression **Result is FALSE** - do not match the regular expression
	Expression	Enter substring/regular expression.
Delimiter		A comma (,), a dot (.) or a forward slash (/) to separate text strings in a regular expression. This parameter is active only when *"Any character string included"* expression type is selected.
Case sensitive		A checkbox to specify whether a regular expression is sensitive to capitalization of letters.

参数	说	
名称（*Name*）	设置正	表达式名称。允许使用任何 Unicode 字符。
表达式（*Expressions*）	单击表达	块中的 添加（*Add*）以添加新的子表达式。

参数	说	
	表达式类型（*Expression type*）选择表达式类	： 字符串已包含（**Character string included**）- 匹配子符子串 包括任何字符串（**Any character string included**）- 匹配列表里包含的字符子串。匹配分隔列表中的任何子字符串。分隔列表包括逗号（,），点号（.）或正斜杠（/）。 字符串未包含（**Character string not included**）- 匹配除此以外的任何字符串 结果为真（**Result is TRUE**） - 匹配正则表达式 结果为假（**Result is FALSE**） - 不匹配正则表达式
	表达式（*Expression*）输入子字	串/正则表达式。
分隔符（*Delimiter*）	用逗号（	），点号（.）或正斜杠（/）分隔正则表达式中的文本字符串。仅当选择” 包括任何字符串（*Any character string included*）“表达式类型时，此参数才有效。

参数	说	
区分大小写（*Case sensitive*）	此复选框用于	定正则表达式是否对字母大小写敏感。

Since Zabbix 2.4.0, a forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, whereas previously it would produce an error.

从 Zabbix 2.4.0 开始，表达式中的正斜杠（/）按字面意思处理，而不是分隔符。这样就可以保存包含斜杠的表达式，而以前会产生错误。

Attention:
A custom regular expression name in Zabbix may contain commas, spaces, etc. In those cases where that may lead to misinterpretation when referencing (for example, a comma in the parameter of an item key) the whole reference may be put in quotes like this: *"@My custom regexp for purpose1, purpose2"*.
Regular expression names must not be quoted in other locations (for example, in LLD rule properties).

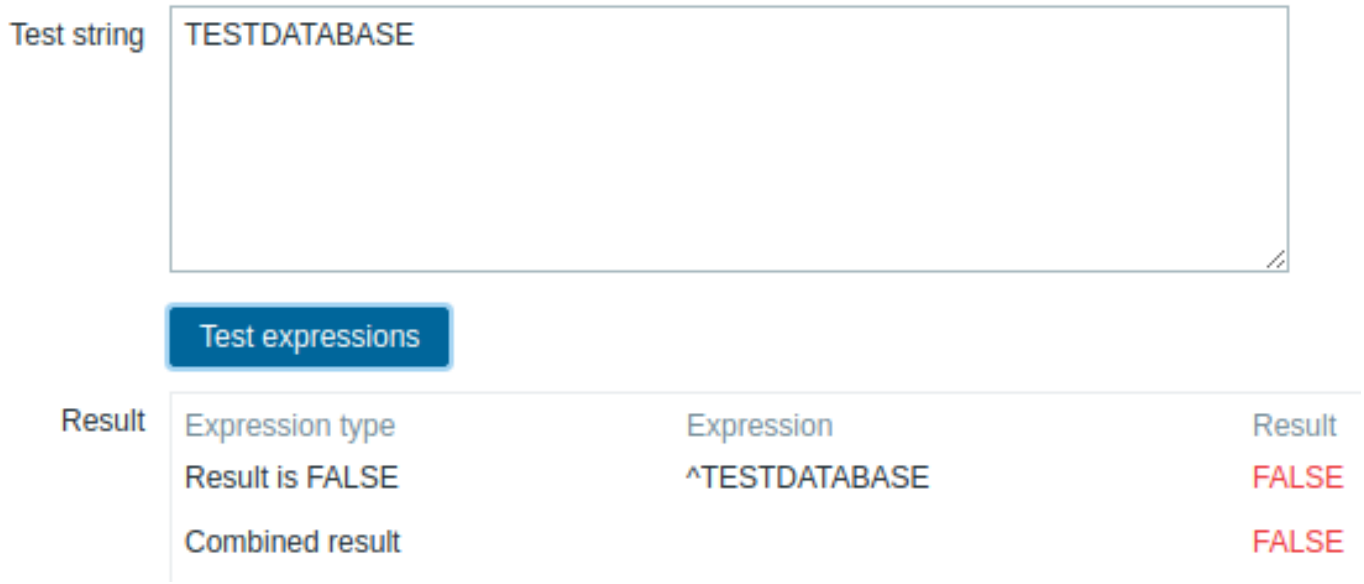
<note important>Zabbix 里自定义的表达式名称可以包含逗号，空格等。在引用时可能导致误解的情况下（例如，监控项键的参数中的逗号），整个引用可以放在引号中，如下所示：// " @My custom regexp for purpose1, purpose2"//。
不能在其他位置引用正则表达式名称（例如，在 LLD 规则属性中）。:::

Example

举例 Use of the following regular expression in LLD to discover databases not taking into consideration a database with a specific name:

在 LLD 中使用以下正则表达式来发现不考虑具有特定名称的数据库的数据库：

```
^TESTDATABASE$
```



Chosen *Expression type*: "Result is FALSE". Doesn't match name, containing string "*TESTDATABASE*".

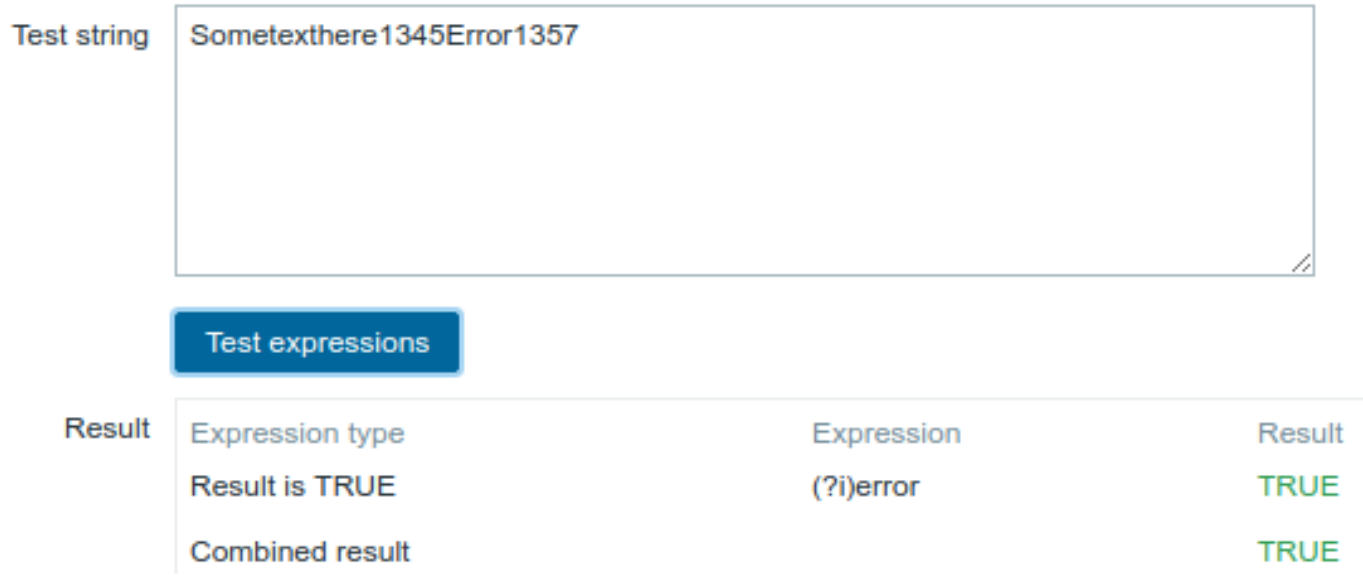
选择 表达式类型（*Expression type*）：" 结果为假（Result is FALSE）"。不匹配名称，包含字符串 "*TESTDATABASE*"。

Example with an inline regex modifier

内联正则表达式修饰符的示例 Use of the following regular expression including an inline modifier (?i) to match the characters "error":

使用如下带有内联修饰符（?i）的正则表达式匹配 "error" 字符：

```
(?i)error
```



Chosen *Expression type*: ”Result is TRUE”. Characters ”error” are matched.

选择 表达式类型（*Expression type*）：” 结果为真（Result is TRUE）”。“error” 字符被匹配到。

Another example with an inline regex modifier

内联正则表达式修饰符的另一个示例 Use of the following regular expression including multiple inline modifiers to match the characters after a specific line:

使用以下正则表达式（包括多个内联修饰符）来匹配特定行之后的字符：

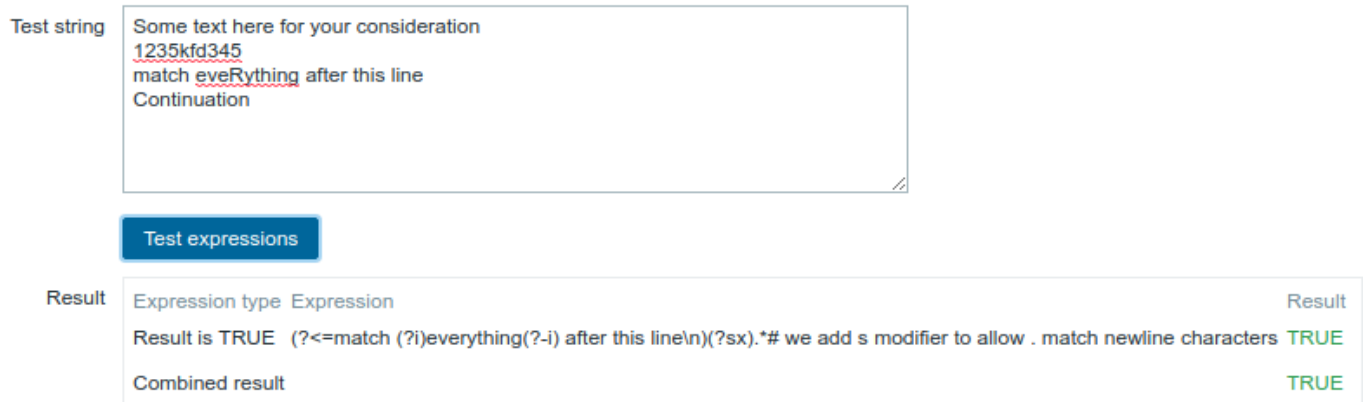
```
(?<=match (?i)everything(?-i) after this line\n)(?sx).*     #我们增加了一个修饰符(?s)来使点号（.）具备匹配
```

(?x) 打开自由间隔模式。::: <note tip>(?s) 对于 “单行模式”，使点号匹配所有字符，包括换行符。Ruby 或 JavaScript 不支持。在 Tcl 中，(?s) 使 ^ 匹配字符串的开头，$ 匹配字符串的结尾。

(?i) 使正则表达式不区分大小写。::: <note tip>(?-i) 减号后的所有模式修饰符都将被关闭。也就是说，只有 everything 是不区分大小写的。

> **Note:**
> (?<= 在正则表达式里，我们称之为 Positive Lookbehind。它告诉正则表达式引擎在字符串中暂时向后退一步，以检查 look behind 内的文本是否可以在那里匹配。

> **Note:**
> 所以，上面这个例子告诉我们，匹配 match everything after this line\n 后面的字符串，且只有 everything 不区分大小写，而且开启了 (?sx) 模式。



Chosen Expression type: ”Result is TRUE”. Characters after a specific line are matched.

选择表达式类型：” 结果为真（Result is TRUE）”。匹配特定行后的字符。

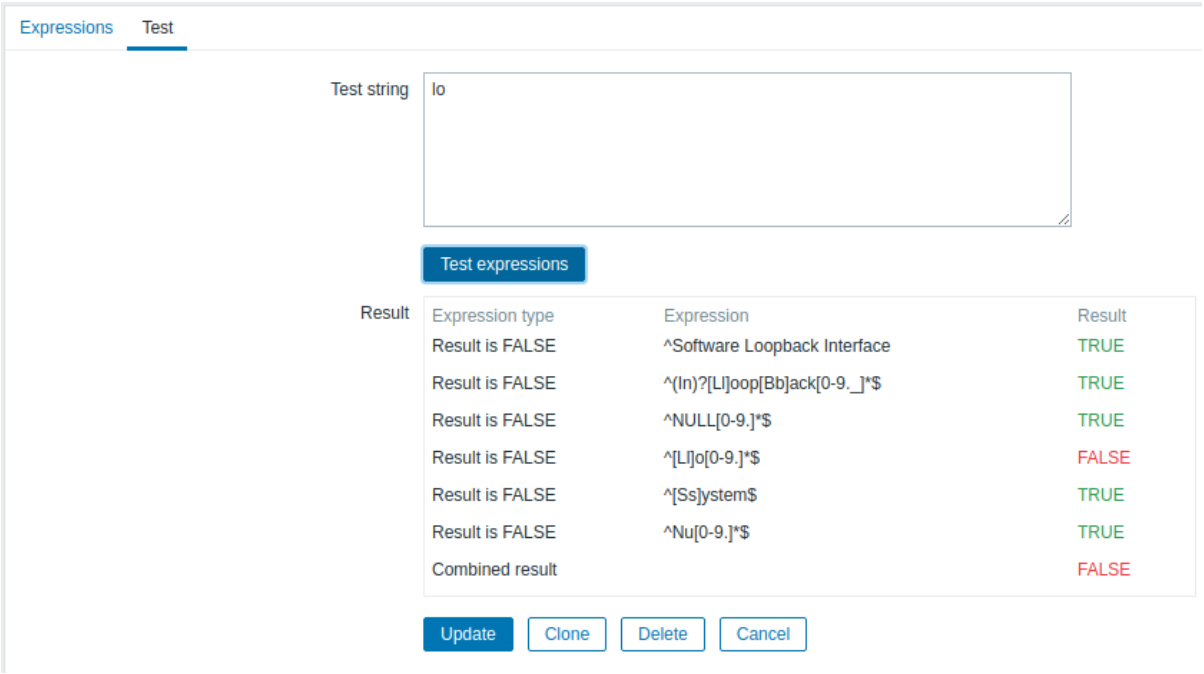
Attention:
g modifier can't be specified in line. The list of available modifiers can be found in pcresyntax man page. For more information about PCRE syntax please refer to PCRE HTML documentation.

Attention:
g 修饰符不能在行中指定。可用修饰符列表可以在 pcresyntax man page 里找到。如果想了解更多的 PCRE 正则表达式语法，请参考 PCRE HTML documentation。

More complex example

更复杂的例子 A custom regular expression may consist of multiple subexpressions, and it can be tested in the **Test** tab by providing a test string.

自定义正则表达式可能包含多个子表达式，可以通过提供测试字符串在 **Test** 选项卡中进行测试。



Results show the status of each subexpression and total custom expression status.

结果显示每个子表达式的状态和整个自定义表达式的状态。

Total custom expression status is defined as *Combined result*. If several sub expressions are defined Zabbix uses AND logical operator to calculate *Combined result*. It means that if at least one Result is False *Combined result* has also False status.

总自定义表达式状态定义为 合并的结果（*Combined result*）。如果定义了几个子表达式，Zabbix 使用 AND 逻辑运算符来计算 合并的结果（*Combined result*）。这意味着如果只要有一个结果为 False，合并的结果（*Combined result*）也为 False 状态。

Explanation of global regular expressions

全局正则表达式的说明

Global regexp	Expression	Description
File systems for discovery	`^(btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|refs|ntfs|fat32|zfs)$`	Matches strings "btrfs" or "ext2" or "ext3" or "ext4" or "jfs" or "reiser" or " xfs" or "ffs" or "ufs" or "jfs" or "jfs2" or "vxfs" or "hfs" or "refs" or "ntfs" or "fat32" or "zfs"
Network interfaces for discovery	`^Software Loopback Interface` `^lo$`	Matches strings starting with "Software Loopback Interface" Matches "lo"

Global regexp	Expression	Description
	`^(In)?[Ll]oop[Bb]ack[0-9._]*$`	Matches strings that optionally start with ”In”, then have ”L” or ”l”, then ”oop”, then ”B” or ”b”, then ”ack”, which can be optionally followed by any number of digits, dots or underscores
	`^NULL[0-9.]*$`	Matches strings staring with ”NULL” optionally followed by any number of digits or dots
	`^[Ll]o[0-9.]*$`	Matches strings starting with ”Lo” or ”lo” and optionally followed by any number of digits or dots
	`^[Ss]ystem$`	Matches ”System” or ”system”
	`^Nu[0-9.]*$`	Matches strings staring with ”Nu” optionally followed by any number of digits or dots
Storage devices for SNMP discovery	`^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$`	Matches ”Physical memory” or ”Virtual memory” or ”Memory buffers” or ”Cached memory” or ”Swap space”
Windows service names for discovery	`^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_opt`	Matches ”MMCSS” or ”gupdate” or ”SysmonLog” or strings like ”clr_optimization_v2.0.50727_32” and ”clr_optimization_v4.0.30319_32” where instead of dots you can put any character except newline.
Windows service startup states for discovery	`^(automatic|automatic delayed)$`	Matches ”automatic” or ”automatic delayed”.

全局正则表达式表达式	说明
发现文件系统（*File systems for discovery*）`^(btrf||ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|refs|ntfs|fat32|zfs)$` 匹配 “btrf	” 或 “ext2” 或 “ext3” 或 “ext4” 或 “jfs” 或 “reiser” 或 “xfs” 或 “ffs” 或 “ufs” 或 “jfs” 或 “jfs2” 或 “vxfs” 或 “hfs“或”refs“或”ntfs“或”f
发现网络接口（*Network interfaces for discovery*） `^Softw|re Loopback Interface` 匹配以”Sof	ware Loopback Interface” 开头的字符串
	`^lo$` 匹配”lo”

全局正则表达式表达式	说明
^(In)?[Ll]oop[Bb]ack	匹配以"In"开头(该项可选)，然后是"L"或者"l"字符，然后是"oop"，然后是"B"或者"b"，然后是"ack"，最后以任意长度(长度可能为0)的数字(0-9)，点号(.)或者下划线(_)结尾的字符串

全局正则表达式表达式	说明
^NULL[0-9.]*$	匹配以“NULL”开头的字符串，后面是任意长度(长度可能为0)的数字(0-9)或者点号(.)

全局正则表达式表达式	说明
^[Ll]o[0-9.]*$	匹配以”Lo”或者”lo”开头的字符串，后面是任意长度(长度可能为0)的数字(0-9)或者点号(.)
^[Ss]ystem$	匹配”System”或者”system”

全局正则表达式表达式	说明
	^Nu[0-9.]+$ 匹配以”Nu”开头的字符串，后面是任意长度(长度可能为0)的数字(0-9)或者点号(.)
使用 *SNMP* 发现存储设备（*Storage devices for SNMP discovery*）	^(Physic|l memory|Virtual memory|Memory buffers|Cached memory|Swap space)$ 匹配”Physic l memory”或”Virtual memory”或”Memory buffers”或”Cached memory”或”Swap space”

全局正则表达式表达式	说明
发现 *Windows* 服务名（*Windows service names for discovery*） `^(MMC|S|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$` 匹配 “MMC	S” 或 “gupdate” 或 “Sysmon-Log” 或类似 “clr_optimization_v2.0.50 和 “clr_optimization_v4.0.30 的字符串，而不是点号，可以放置除换行符之外的任何字符。
发现 *Windows* 服务启动状态（*Windows service startup states for discovery*） `^(automa|ic|automatic delayed)$` 匹配”automa	ic” 或”automatic de-layed”。

Regular expression support by location

支持正则表达式的位置

Location	Regular expression	Global regular expression	Comments
Agent items			
	eventlog[]	Yes	`regexp`, `severity`, `source`, `eventid` parameters
	log[]		`regexp` parameter
	log.count[]		
	logrt[]	Yes/No	`regexp` parameter supports both, `file_regexp` parameter supports non-global expressions only
	logrt.count[]		
	proc.cpu.util[]	No	`cmdline` parameter
	proc.mem[]		
	proc.num[]		
	sensor[]		`device` and `sensor` parameters on Linux 2.4
	system.hw.macaddr[]		`interface` parameter

Location	Regular expression	Global regular expression	Comments
	system.sw.packages[]		`package` parameter
	vfs.dir.count[]		`regex_incl` and `regex_excl` parameters
	vfs.dir.size[]		`regex_incl` and `regex_excl` parameters
	vfs.file.regexp[]		`regexp` parameter
	vfs.file.regmatch[]		
	web.page.regexp[]		
SNMP traps			
	snmptrap[]	Yes	`regexp` parameter
Item value preprocessing	Yes	No	`pattern` parameter
Trigger functions			
	count()	Yes	`pattern` parameter if `operator` parameter is *regexp* or *iregexp*
	logeventid()		`pattern` parameter
	iregexp()		
	regexp()		
Low-level discovery	Yes	Yes	*Filter* field
Web monitoring	Yes	No	*Variables* with a **regex:** prefix *Required string* field
Macro functions			
	regsub()	No	`pattern` parameter
	iregsub()		
Icon mapping	Yes	Yes	*Expression* field

位置	正	表达式全局正则表达	注释	
Agent 监控项（Agent items）				
	eventlog[]	Yes	Yes	`regexp`, `severity`, `source`, `eventid` 参数
	log[]			`regexp` 参数

位置	正	表达式全局正则表达	注释	
	log.count[]			
	logrt[]		Yes/No	regexp 参数两者都支持，file_regexp 参数仅支持非全局表达式
	logrt.count[]			
	proc.cpu.util[]		No	cmdline 参数
	proc.mem[]			
	proc.num[]			
	sensor[]			device 和 sensor 参数在 Linux 2.4 中
	system.hw.macaddr[]			interface 参数
	system.sw.packages[]			package 参数
	vfs.dir.count[]			regex_incl 和 regex_excl 参数
	vfs.dir.size[]			regex_incl 和 regex_excl 参数
	vfs.file.regexp[]			regexp 参数
	vfs.file.regmatch[]			
	web.page.regexp[]			
SNMP traps				
	snmptrap[]	Yes	Yes	regexp 参数

位置	正	表达式全局正则表达	注释
监控项值预处理（**Item value pre-processing**）	Yes	No	pattern| 参数 | |**[触发器函数（Trigger functions）](/manual/appendix/trigger|<|count()|Yes|Yes|pattern 参数，如果 operator 参数是 *regexp* 或者 *iregexp*| |^|logeventid()|^|^|pattern 参数 | |^|ireg-exp()|^|^|^| |^|reg-exp()|^|^|^| |**[低级别发现（Low-level discovery）](/manual/discovery/low_level_dis 字段 | |**[Web 监测（Web monitoring）](/manual/web_monitoring#config 带有 **regex:** 前缀 *Required string* 字段 | |**[宏函数（Macro functions）](/manual/config/macros/macro_fu |<|reg-sub()|Yes|No|pattern‘ 参数 iregsub()
图标映射（**Icon mapping**）	Yes	Yes	*Expr ssion* 字段

13. 问题确认

概述　Zabbix 的问题事件可由用户确认。

如果用户收到问题事件的通知，可以打开 Zabbix 的前端页面，从问题更新页面上找到对应的问题进行确认。当进行确认的时候，可以输入注释表明他们正在处理该问题，或者输入任何他们想表述的内容。

利用这种方式，如果有另一个系统管理员察觉到这个问题，就可以立刻知道该问题已经被确认过，并且看到之前留下的注释。

这样的问题处理工作流，可以让多个系统管理员协同工作。

定义动作操作（action operations） 时也会使用确认状态。例如，可以定义仅在事件一段时间后依然未被确认时才将通知发送到更高级别的管理者。

要确认事件，用户必须至少具有相应触发器的读权限。

有 两种方法访问可以进行确认操作的问题更新页面。

第一种方法，您可以单击 确认（*Ack*）列，显示问题的确认状态：

- 监测中（*Monitoring*）→ 仪表板（*Dashboard*）（问题（*Problems*）和 问题按严重性（*Problems by severity*）小部件）
- 监测中（*Monitoring* ）→ 问题（*Problems*）
- 监测中（*Monitoring* ）→ 问题（*Problems* ）→ 事件详情（*Event details*）
- 监测中（*Monitoring*）→ 聚合图形（*Screens*）（主机组问题（*Host group issues*），主机问题（*Host issues*），问题按严重性（*Problems by severity*）元素）

确认（*Ack*）按钮包含一个'Yes' 或者'No' 链接，分别代表已确认或者未确认的问题，单击这个链接将前往问题更新页面。

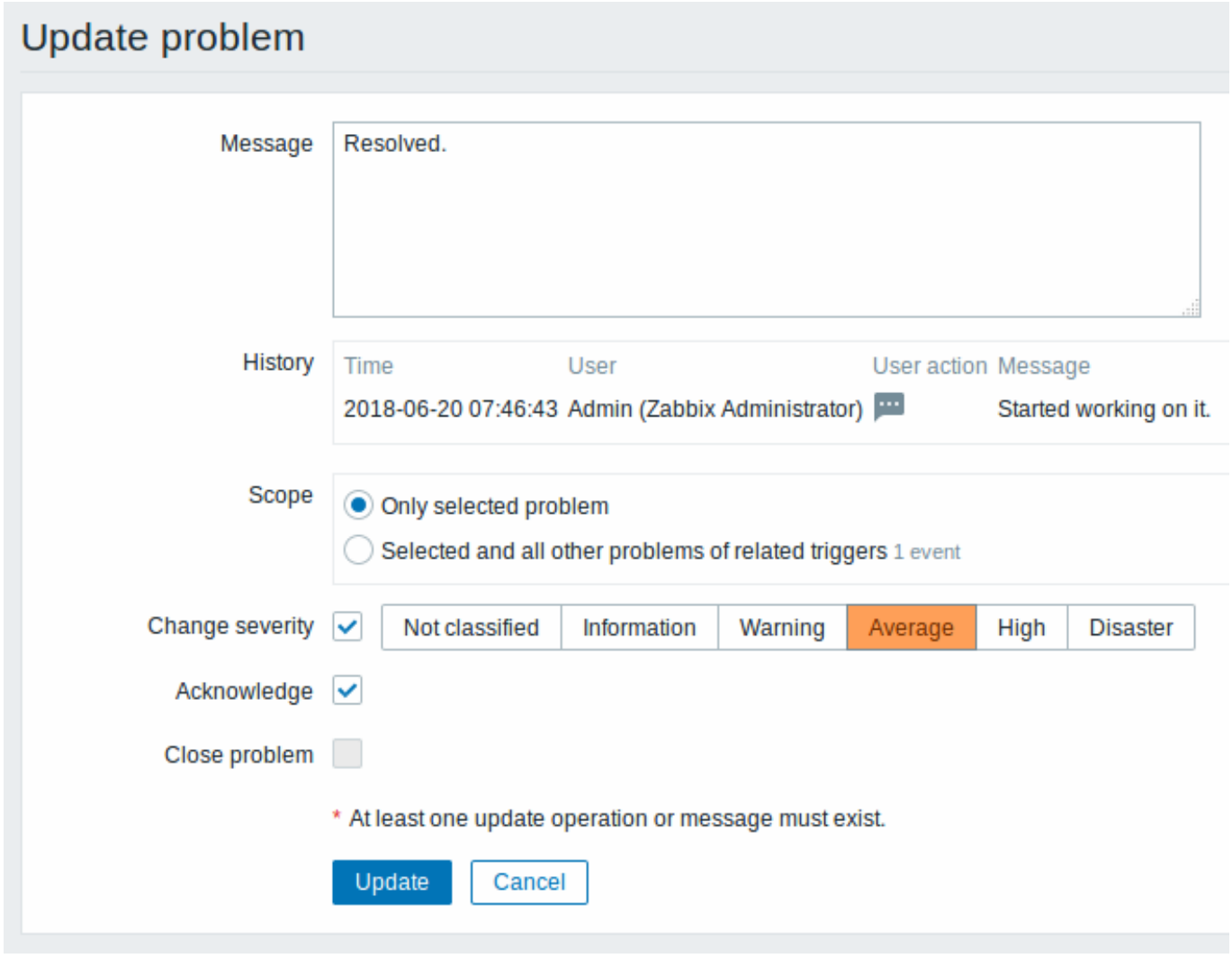
第二种方法，可以单击未解决的问题单元格：

- 监测中（*Monitoring*）→ 仪表板（*Dashboard*）（数据概览（*Data overview*）和 触发器概览（*Trigger overview*）小部件）
- 监测中（*Monitoring*）→ 概览（*Overview*）
- 监测中（*Monitoring*）→ 聚合图形（*Screens*）（数据概览（*Data overview*）和 触发器概览（*Trigger overview*）元素）

弹出菜单包含一个可以将你带到问题更新页面的选项。

问题更新　问题更新页面允许：

- 评论问题
- 查看目前为止的评论和动作
- 改变问题的级别
- 确认问题
- 手动关闭问题



所有必填输入字段都标有红色星号。

参数	描
消息（*Message*）输入文	以评论问题。
历史记录（*History*）列出了有关	问题的过去的操作和评论，以及时间和用户详细信息。有关用于表示用户操作的图标的含义，请参阅事件详情（event de-tail）页面。

参数	描
范围（*Scope*）定义此操作的范围，例如更改级别，确认或手动关闭问题： - 仅所选问题（**Only selected problem**） - 将仅影响此事件选定的和相关触发器的所有其他问题（**Selected and all other problems of related trig-**）	

参数	描
改变严重性（*Change severity*）选中该复选框	然后单击严重性按钮以更新问题级别。
确认（*Acknowledge*）选中复	框以确认问题。对于已经确认过的问题，该选项不可用。

参数	描
关闭问题（*Close problem*）选中复选框	手动关闭问题。如果在触发器配置（trigger configuration）里的允许手工关闭（*Allow manual close*）选项被勾选中，那么就可以通过此种方式去关闭问题。

显示　根据确认信息，可以在仪表板或 map 图中配置问题数量的显示方式。要做到这一点，你必须在 问题显示（*Problem display*）选项中进行选择，拓扑图配置（map configuration） 和 问题按严重性（*Problems by severity*） 仪表板小部件（dashboard widget）。可以将所有问题计数，未确认的问题计数显示为仅与总计或未确认的问题计数分开。

根据问题更新信息（确认等），可以配置更新操作 - 发送消息或执行远程命令。

14. 配置导出/导入

概述　通过 Zabbix 的导出/导入功能，你可以在不同的 Zabbix 系统之间交换配置实体。

该功能的典型使用场景如下：

- 分享模板或者网络 maps - Zabbix 用户可以分享他们的配置参数
- 在 *share.zabbix.com* 网站上分享 web 场景 - 导出带有 web 场景的模板，上传到 *share.zabbix.com* 即可。其他的用户就可以下载模板，然后往 Zabbix 导入 XML 模板文件
- 集成第三方平台 - 通用的 XML 格式，让 Zabbix 与第三方平台或者应用集成及数据导入/导出成为可能

哪些对象可以被导出/导入

可以被导出/导入的对象有：

- 主机组 (仅通过 *Zabbix API*)
- 模板
- 主机
- 网络拓扑图
- 图像
- 聚合图形
- 值映射

导出格式

可以通过 Zabbix 前端或者Zabbix API 来导出数据。支持的导出格式如下：

- XML - 在前端页面导出
- XML or JSON - 通过 Zabbix API 导出

关于导出功能的明细

- 所有支持导出的元素都在一个文件里。
- 从链接模板里继承的主机和模板实体（监控项，触发器，图表，发现规则）不会被导出。在主机层面对这些实体所做的任何更改（比如更改监控项间隔，修改正则表达式或者给低级别发现增加原型），在导出的时候都会丢失；在导入的时候，所有来自于链接模板的实体，就像在原始链接模板上一样会被重新创建。
- 由低级别发现创建的实体以及任何依赖于它们的实体都不会被导出。例如，为某个 LLD 规则生成的监控项而创建的触发器不会被导出。

关于导入功能的明细

- 一旦遇到错误导出功能就会停止
- 如果刚好在图像导入过程中更新已有的图像，“图像类型（imagetype）” 字段会被忽略。也就是说，不能通过导入来更改图像类型。
- 当导入主机/模板的时候使用 “删除不存在（Delete missing）” 选项，那么不在 XML 导入文件里的主机/模板宏（macros）也将会被删除。
- 监控项，触发器，图表，主机/模板应用，发现规则，监控项原型，触发器原型，图表原型的空标签是没有意义的，就好像不存在一样。其他的标签，比如，监控项应用是有意义的。也就是说，空标签代表监控项没有应用，丢失标签代表不需要更新应用。
- 导入支持 XML 和 JSON 两种格式，导入文件必须有正确的文件扩展名：XML 的是.xml，JSON 的是.json。
- 关于支持的 XML 版本，请查看兼容性信息

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>4.0</version>
    <date>2016-10-04T06:20:11Z</date>
</zabbix_export>
```

XML 基本格式

```
<?xml version="1.0" encoding="UTF-8"?>
```

默认 XML 文件头格式。

```
<zabbix_export>
```

Zabbix XML 导出的格式标签。

<version>4.0</version>

导出的版本。

<date>2016-10-04T06:20:11Z</date>

导出的时候，日期以 ISO 8601 长格式创建，其他的标签取决于导出的对象。

1 主机组

在前端页面上，主机组只能在主机或者模板导出的时候导出。当主机或者模板被导出的时候，它所属的所有的组都会被自动导出。

API 允许单独导出主机组而不依赖于主机或者模板。

```
<groups>
    <group>
        <name>Zabbix servers</name>
    </group>
</groups>
```

多个组/组

参数类	说明	详细
名称 *	符型 * 组名	

2 模板

概述

模板就是导出 的许多相关联的对象和对象关系。

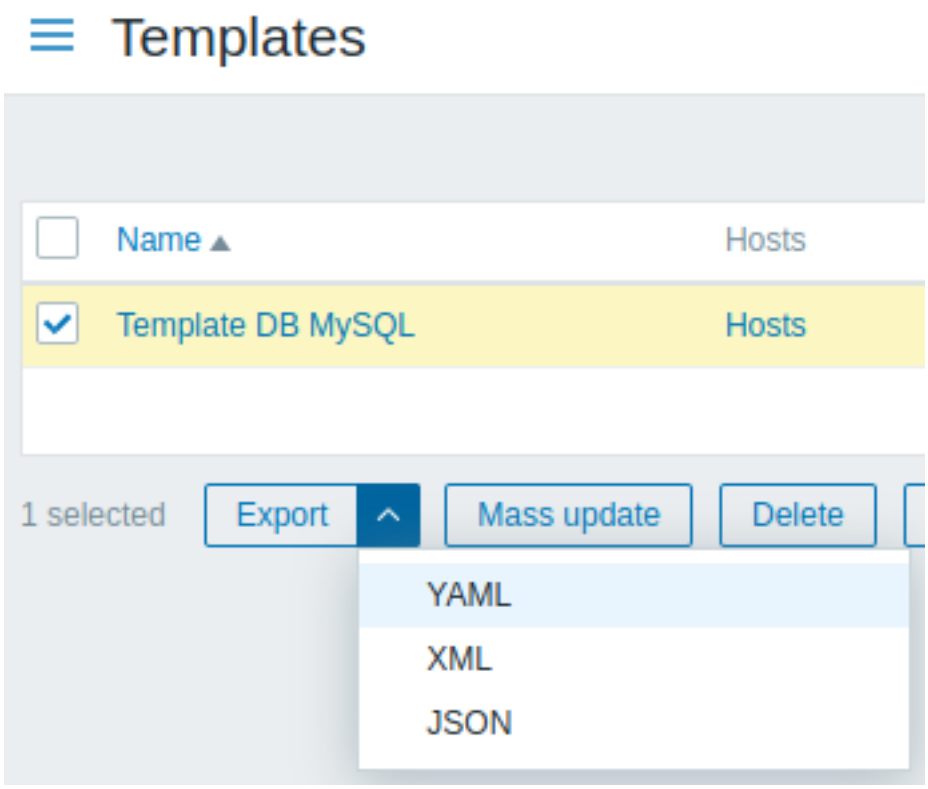
模板导出包含的内容：

- 链接的主机组
- 模板数据
- 到其他模板的链接
- 到主机组的链接
- 直接链接的应用集
- 直接链接的监控项
- 直接链接的触发器
- 直接链接的图形
- 直接链接的聚合图形
- 直接链接的带有所有原型的发现规则
- 直接链接的 web 场景
- 值映射

导出

要导出模板，按照如下的操作：

- 切换到：配置（*Configuration*） → 模板（*Templates*）
- 选中要导出模板的复选框
- 单击列表下面的 导出（*Export*）按钮

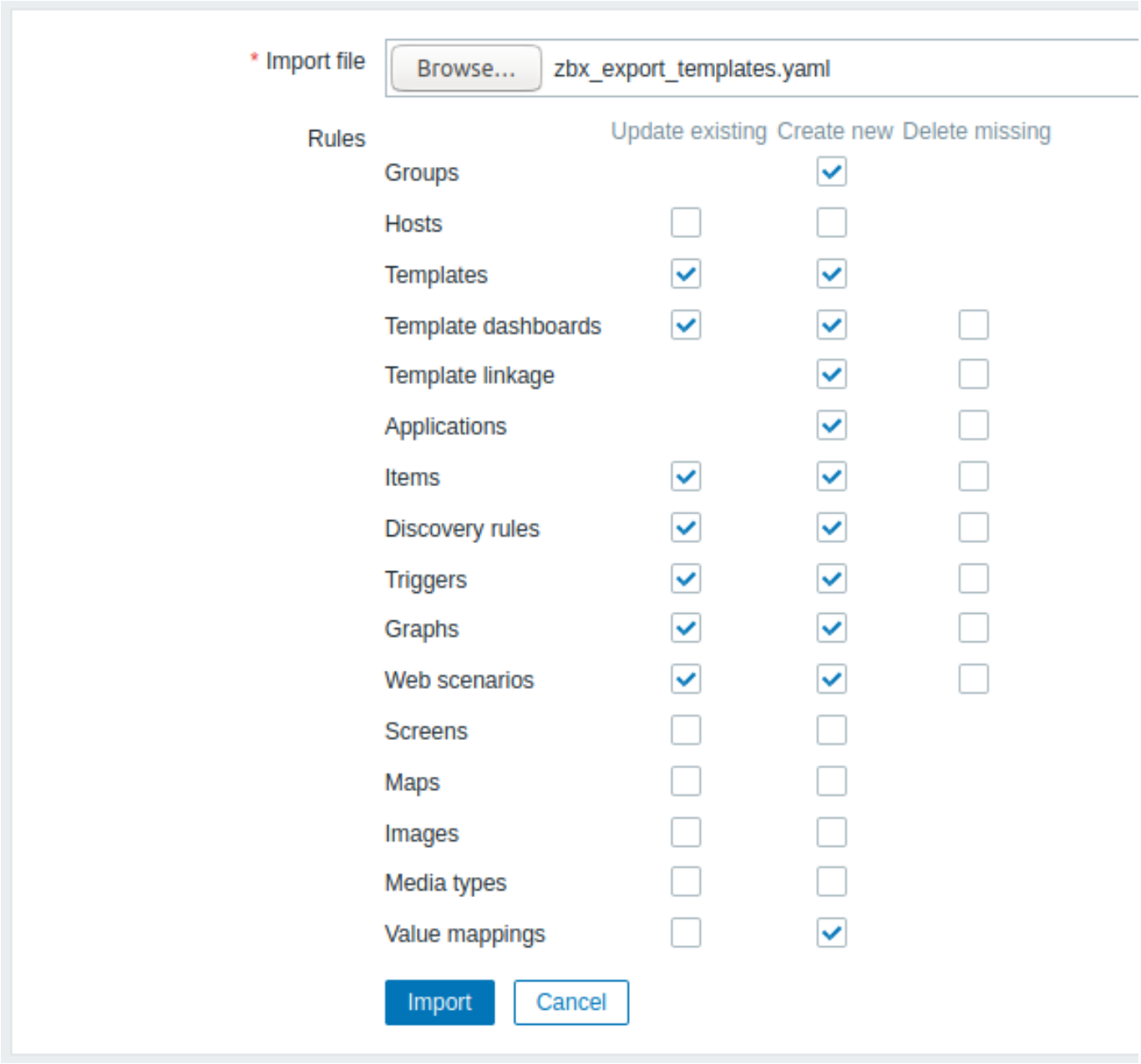


选中的模板被导出到本地的 XML 文件里，默认的名称是 *zabbix_export_templates.xml*。

导入

要导入模板，按照如下的操作：

- 切换到：配置（*Configuration*） → 模板（*Templates*）
- 单击右上角的 导入（*Import*）按钮
- 选择要导入的文件
- 标记导入规则里要求的选项
- 单击 导入（*Import*）按钮



所有必填输入字段都标有红色星号。

导入成功或者失败的消息都会在前端页面显示。

导入规则：

	规则说
更新现有的（*Update existing*）已有的元素会	导入文件里的数据更新，否则不会更新。
创建新的（*Create new*）导入会使用	入文件的里数据增加新的元素，否则不会增加。
删除不存在（*Delete missing*）导入会删除已	的但是在导入文件里没有的元素，否则不会删除。

导出格式

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>4.0</version>
    <date>2018-04-03T06:25:22Z</date>
    <groups>
        <group>
            <name>Templates/Databases</name>
        </group>
    </groups>
    <templates>
```

```
        <template>
            <template>Template DB MySQL</template>
            <name>Template DB MySQL</name>
            <description/>
            <groups>
                <group>
                    <name>Templates/Databases</name>
                </group>
            </groups>
            <applications>
                <application>
                    <name>MySQL</name>
                </application>
            </applications>
            <items>
                <item>
                    <name>MySQL status</name>
                    <type>0</type>
                    <snmp_community/>
                    <snmp_oid/>
                    <key>mysql.ping</key>
                    <delay>1m</delay>
                    <history>1w</history>
                    <trends>365d</trends>
                    <status>0</status>
                    <value_type>3</value_type>
                    <allowed_hosts/>
                    <units/>
                    <snmpv3_contextname/>
                    <snmpv3_securityname/>
                    <snmpv3_securitylevel>0</snmpv3_securitylevel>
                    <snmpv3_authprotocol>0</snmpv3_authprotocol>
                    <snmpv3_authpassphrase/>
                    <snmpv3_privprotocol>0</snmpv3_privprotocol>
                    <snmpv3_privpassphrase/>
                    <params/>
                    <ipmi_sensor/>
                    <authtype>0</authtype>
                    <username/>
                    <password/>
                    <publickey/>
                    <privatekey/>
                    <port/>
                    <description>It requires user parameter mysql.ping, which is defined in userparameter_

0 - MySQL server is down
1 - MySQL server is up</description>
                    <inventory_link>0</inventory_link>
                    <applications>
                        <application>
                            <name>MySQL</name>
                        </application>
                    </applications>
                    <valuemap>
                        <name>Service state</name>
                    </valuemap>
                    <logtimefmt/>
                    <preprocessing/>
                    <jmx_endpoint/>
                    <timeout>3s</timeout>
                    <url/>
                    <query_fields/>
```

```
            <posts/>
            <status_codes>200</status_codes>
            <follow_redirects>1</follow_redirects>
            <post_type>0</post_type>
            <http_proxy/>
            <headers/>
            <retrieve_mode>0</retrieve_mode>
            <request_method>1</request_method>
            <output_format>0</output_format>
            <allow_traps>0</allow_traps>
            <ssl_cert_file/>
            <ssl_key_file/>
            <ssl_key_password/>
            <verify_peer>0</verify_peer>
            <verify_host>0</verify_host>
            <master_item/>
        </item>
        <item>
            <name>MySQL begin operations per second</name>
            <type>0</type>
            <snmp_community/>
            <snmp_oid/>
            <key>mysql.status[Com_begin]</key>
            <delay>1m</delay>
            <history>1w</history>
            <trends>365d</trends>
            <status>0</status>
            <value_type>0</value_type>
            <allowed_hosts/>
            <units>qps</units>
            <snmpv3_contextname/>
            <snmpv3_securityname/>
            <snmpv3_securitylevel>0</snmpv3_securitylevel>
            <snmpv3_authprotocol>0</snmpv3_authprotocol>
            <snmpv3_authpassphrase/>
            <snmpv3_privprotocol>0</snmpv3_privprotocol>
            <snmpv3_privpassphrase/>
            <params/>
            <ipmi_sensor/>
            <authtype>0</authtype>
            <username/>
            <password/>
            <publickey/>
            <privatekey/>
            <port/>
            <description>It requires user parameter mysql.status[*], which is defined in userparam
            <inventory_link>0</inventory_link>
            <applications>
                <application>
                    <name>MySQL</name>
                </application>
            </applications>
            <valuemap/>
            <logtimefmt/>
            <preprocessing>
                <step>
                    <type>10</type>
                    <params/>
                </step>
            </preprocessing>
            <jmx_endpoint/>
            <timeout>3s</timeout>
```

```
                <url/>
                <query_fields/>
                <posts/>
                <status_codes>200</status_codes>
                <follow_redirects>1</follow_redirects>
                <post_type>0</post_type>
                <http_proxy/>
                <headers/>
                <retrieve_mode>0</retrieve_mode>
                <request_method>1</request_method>
                <output_format>0</output_format>
                <allow_traps>0</allow_traps>
                <ssl_cert_file/>
                <ssl_key_file/>
                <ssl_key_password/>
                <verify_peer>0</verify_peer>
                <verify_host>0</verify_host>
                <master_item/>
            </item>
            <item>
                <name>MySQL queries per second</name>
                <type>0</type>
                <snmp_community/>
                <snmp_oid/>
                <key>mysql.status[Questions]</key>
                <delay>1m</delay>
                <history>1w</history>
                <trends>365d</trends>
                <status>0</status>
                <value_type>0</value_type>
                <allowed_hosts/>
                <units>qps</units>
                <snmpv3_contextname/>
                <snmpv3_securityname/>
                <snmpv3_securitylevel>0</snmpv3_securitylevel>
                <snmpv3_authprotocol>0</snmpv3_authprotocol>
                <snmpv3_authpassphrase/>
                <snmpv3_privprotocol>0</snmpv3_privprotocol>
                <snmpv3_privpassphrase/>
                <params/>
                <ipmi_sensor/>
                <authtype>0</authtype>
                <username/>
                <password/>
                <publickey/>
                <privatekey/>
                <port/>
                <description>It requires user parameter mysql.status[*], which is defined in userparam
                <inventory_link>0</inventory_link>
                <applications>
                    <application>
                        <name>MySQL</name>
                    </application>
                </applications>
                <valuemap/>
                <logtimefmt/>
                <preprocessing>
                    <step>
                        <type>10</type>
                        <params/>
                    </step>
                </preprocessing>
```

```
                <jmx_endpoint/>
                <timeout>3s</timeout>
                <url/>
                <query_fields/>
                <posts/>
                <status_codes>200</status_codes>
                <follow_redirects>1</follow_redirects>
                <post_type>0</post_type>
                <http_proxy/>
                <headers/>
                <retrieve_mode>0</retrieve_mode>
                <request_method>1</request_method>
                <output_format>0</output_format>
                <allow_traps>0</allow_traps>
                <ssl_cert_file/>
                <ssl_key_file/>
                <ssl_key_password/>
                <verify_peer>0</verify_peer>
                <verify_host>0</verify_host>
                <master_item/>
            </item>
        </items>
        <discovery_rules/>
        <httptests/>
        <macros/>
        <templates/>
        <screens>
            <screen>
                <name>MySQL performance</name>
                <hsize>2</hsize>
                <vsize>1</vsize>
                <screen_items>
                    <screen_item>
                        <resourcetype>0</resourcetype>
                        <width>500</width>
                        <height>200</height>
                        <x>0</x>
                        <y>0</y>
                        <colspan>1</colspan>
                        <rowspan>1</rowspan>
                        <elements>0</elements>
                        <valign>1</valign>
                        <halign>0</halign>
                        <style>0</style>
                        <url/>
                        <dynamic>0</dynamic>
                        <sort_triggers>0</sort_triggers>
                        <resource>
                            <name>MySQL operations</name>
                            <host>Template DB MySQL</host>
                        </resource>
                        <max_columns>3</max_columns>
                        <application/>
                    </screen_item>
                </screen_items>
            </screen>
        </screens>
    </template>
</templates>
<triggers>
    <trigger>
        <expression>{Template DB MySQL:mysql.ping.last(0)}=0</expression>
```

```
            <recovery_mode>0</recovery_mode>
            <recovery_expression/>
            <name>MySQL is down</name>
            <correlation_mode>0</correlation_mode>
            <correlation_tag/>
            <url/>
            <status>0</status>
            <priority>2</priority>
            <description/>
            <type>0</type>
            <manual_close>0</manual_close>
            <dependencies/>
            <tags/>
        </trigger>
    </triggers>
    <graphs>
        <graph>
            <name>MySQL operations</name>
            <width>900</width>
            <height>200</height>
            <yaxismin>0.0000</yaxismin>
            <yaxismax>100.0000</yaxismax>
            <show_work_period>1</show_work_period>
            <show_triggers>1</show_triggers>
            <type>0</type>
            <show_legend>1</show_legend>
            <show_3d>0</show_3d>
            <percent_left>0.0000</percent_left>
            <percent_right>0.0000</percent_right>
            <ymin_type_1>0</ymin_type_1>
            <ymax_type_1>0</ymax_type_1>
            <ymin_item_1>0</ymin_item_1>
            <ymax_item_1>0</ymax_item_1>
            <graph_items>
                <graph_item>
                    <sortorder>0</sortorder>
                    <drawtype>0</drawtype>
                    <color>C8C800</color>
                    <yaxisside>0</yaxisside>
                    <calc_fnc>2</calc_fnc>
                    <type>0</type>
                    <item>
                        <host>Template DB MySQL</host>
                        <key>mysql.status[Com_begin]</key>
                    </item>
                </graph_item>
            </graph_items>
        </graph>
    </graphs>
    <value_maps>
        <value_map>
            <name>Service state</name>
            <mappings>
                <mapping>
                    <value>0</value>
                    <newvalue>Down</newvalue>
                </mapping>
                <mapping>
                    <value>1</value>
                    <newvalue>Up</newvalue>
                </mapping>
            </mappings>
```

```
        </value_map>
    </value_maps>
</zabbix_export>
```

元素标签

元素标签值的释义在下面的表格中。

模板标签

元素元	属性类型	范围	说明
templates			模板的根元素。
template			单独的模板。
	template	字符 唯	模板名称。
	name	字符 显	模板名称。
	description	文本 模	描述。
groups			主机组根元素。
group			单独的主机组。
	name	字符 唯	主机组名称。
applications			模板应用集的根元素。
application			单独的模板应用集。
	name		应用集名称。
macros			模板用户宏的根元素。
macro			单独的模板用户宏。
	name		用户宏名称。
	value		用户宏的值。
templates			链接模板的根元素。
template			单独的模板。
	name	字符 模	名称。

模板监控项标签

元素元	属性类型	范围	说明
items			监控项的根元素。
item			单独的监控项。
	name	字符	监 项名称。

元素元	属性类型	范围	说明
	type	整型 0	- Zabbix agent 监 1 - SNMPv1 agent 2 - Zabbix trapper 3 - simple check 4 - SNMPv2 agent 5 - internal 6 - SNMPv3 agent 7 - Zabbix agent (active) 8 - aggregate 9 - HTTP test (web monitor-ing scenario step) 10 - external 11 - database monitor 12 - IPMI agent 13 - SSH agent 14 - Telnet agent 15 - calculated 16 - JMX agent 17 - SNMP trap 18 - De-pendent item 19 - HTTP agent item 项类型。

元素元	属性类型	范围	说明
	snmp_community	字符	如 ’type’ 的值是 1 或 4，那这就是 SNMP 的团体名称。
	snmp_oid	字符	S MP 对象 ID。
	key	字符	监 项的 key。
	delay	字符	监 项的更新间隔。秒，带后缀的时间单位，自定义间隔，用户宏或者低级别发现宏。

元素元	属性类型	范围	说明	
	history	字符	决	历史数据存储时长的时间单位。带后缀的时间单位，用户宏或者低级别发现宏。

元素元	属性类型	范围	说明
	trends	字符	决 趋势数据存储时长的时间单位。带后缀的时间单位，用户宏或者低级别发现宏。
	status	整型 0	- enabled 监 1 - disabled 项状态。
	value_type	整型 0	- float 收 1 - character 2 - log 3 - unsigned integer 4 - text 值的类型。

元素元	属性类型	范围	说明	
	allowed_hosts	字符	如	’type’ 是 2 或者 19，那这就是允许发送该监控项对应值的主机 IP 地址 (逗号分隔) 列表。
	units	字符	返	值的单位 (bps, B)。
	snmpv3_contextname	字符	S	MPv3 上下文名称。
	snmpv3_securityname	字符	S	MPv3 安全名称。
	snmpv3_securitylevel	整型 0	- noAuthNo-Priv S1 - authNo-Priv 2 - authPriv	MPv3 安全级别。

元素元	属性类型	范围	说明	
	snmpv3_authprotocol	整型 0	- MD5 S1 - SHA	MPv3 认证协议。
	snmpv3_authpassphrase	字符	S	MPv3 认证密码。
	snmpv3_privprotocol	整型 0	- DES S1 - AES	MPv3 私有协议。
	snmpv3_privpassphrase	字符	S	MPv3 私有密码。

元素元	属性类型	范围	说明	
	params	文本	如	’type’ 是 13、14，这就是”执行脚本（Executed script）”的名称。如果’type’是11，这就是”SQL查询（SQL query）”字段。如果’type’是15，这是”公式（Formula）”字段。
	ipmi_sensor	字符	如	’type’ 是 12，这是IPMI传感器ID。

元素元	属性类型	范围	说明	
	authtype	整型 S	H 客户端监控项的认证类型：如果’type’是 13 或者 0 - password 1 - key HTTP 监控项认证类型： 0 - none 1 - basic 2 - NTLM	9，这是认证类型。
	username	字符	如	’type’是 11、13、14、19，这是用户名。
	password	字符	如	’type’是 11、13、14、19，这是密码。
	publickey	字符	如	’type’是 13，这是公共秘钥文件的名称。

元素元	属性类型	范围	说明	
	privatekey	字符	如	’type’是13，这是私有密钥文件的名称。
	port	字符	监	项的自定义端口。
	description	文本	监	项描述。
	inventory_link	整型 0	- 无链接 使用监控数字（*number*）- ’host_inventory’表里的字段数。	值来填充资产记录字段。
	logtimefmt	字符	日	条目的时间格式。只有日志监控项使用。

元素元	属性类型	范围	说明	
	jmx_endpoint	字符	如	’type’ 是 16，这是 JMX 端点。
	url	字符	如	’type’ 是 19，这是 URL 字符。
	allow_traps	整型 0	- 不允许 trapping。如果’ty1 - 允许 trapping。	e’ 是 19，属性允许发送数据给监控项。
	follow_redirects	整型 0	- 不跟随重定向。 如果’type’1 - 跟随重定向。	19，跟随 HTTP 重定向。
	headers	对象	如	’type’ 是 19，这是带有 HTTP(S) 请求头的对象。

元素元	属性类型	范围	说明	
	http_proxy	字符	如	’type’ 是 19，这是 HTTP(S) 代理连接字符。
	output_format	整型 0	- 保持原样存储。 如果’type’1 - 转换为 JSON。	19，怎样处理响应。
	post_type	整形 0	- 原始数据。 如果’typ2 - JSON 数据。 3 - XML 数据。	’ 是 19，这是请求体的类型。
	posts	文本	如	’type’ 是 19，这是请求体。
	query_fields	数组	如	’type’ 是 19，请求查询字段的对象数组。

元素元	属性类型	范围	说明	
	request_method	整型 0	- GET 如 1 - POST 2 - PUT 3 - HEAD	’type’ 是 19，这是请求方法。
	retrieve_mode	整型 0	- 请求体 如果’t1 - 请求头。 2 - 请求体和请求头都被存储。	pe’ 是 19，响应的什么部分将被存储。
	ssl_cert_file	字符	如	’type’ 是 19，这是公共 SSL 密钥文件的路径。
	ssl_key_file	字符	如	’type’ 是 19，这是私有 SSLK 密钥文件的路径。

元素元	属性类型	范围	说明	
	ssl_key_password	字符	如	’type’ 是 19，这是 SSL 密钥文件的密码。
	status_codes	字符	如	’type’ 是 19，这是逗号分隔的 HTTP 请求的状态码范围。
	timeout	字符	如	’type’ 是 19，监控项数据拉取请求的超时时间。

元素元	属性类型	范围	说明	
	verify_host	整型 0	- 不校验。 如果’ty1 - 校验。	e’是19，校验URL里的主机名是否在常见名称字段里，或者是否在主机证书的主题备用名称里。
	verify_peer	整型 0	- 不校验。 如果’ty1 - 校验。	e’是19，校验是否是主机证书验证。
value map				值映射。

元素元	属性类型	范围	说明
	name	字符	监项使用的值映射名称。
applications			应用集的根元素。
application			单独的应用集。
	name		应用集名称。
preprocessing			监控项值预处理。
step			单独的监控项值预处理步骤。

元素元	属性类型	范围	说明
	type	整型 1	- 自定义放大倍数 监控项值预处理步 2 - 右截断 3 - 左截断 4 - 两端截断 5 - 正则表达式匹配 6 - 二进制到十进制 7 - 八进制到十进制 8 - 十六进制到十进制 9 - 简单改变；计算为 (收到的值-之前的值) 10 - 每秒改变；计算为 (当前收到的值-上一个值)/(当前时间-上一次检查时间) 的类型
	params	字符	监 项值预处理步骤的参数。
master_item			Individual item master item data.
	key	字符	从 监控项的主监控项值。

模板低级别发现规则标签

元素元	属性类型	范围	说
discovery_rules			
discovery_rule			
	对于大部分的元素标签值来说，请查阅常规监控项的元素标签值。下面仅描述低级别发现规则特有的标签。		
	lifetime	字符	时
filter			
	evaltype	整型 0	- 2 3
	formula	字符	过
	conditions		
condition			
	macro	字符	值
	value	字符	过
	operator	整型	
	formulaid	字符	过
item_prototypes			
item_prototype			
	对于大部分元素标签值来说，请查阅常规监控项的元素标签值。下面仅描述监控项原型特有的标签。		
application_prototypes			
application_prototype			
	name		
master_item_prototype			
	key	字符	卓

模板触发器标签

元素元	属性类型	范围	说明
triggers			触发器的根元素。
trigger			单独的触发器。
	expression	字符	触发器表达式。
	recovery_mode	整型 0 - 表达式 生成 OK1 - 恢复表达式 2 - none	件的基础。
	recovery_expression	字符	触发器恢复表达式。

元素元	属性类型	范围	说明
	name	字符	触器名称。
	correlation_mode	整型 0	- 没有事件关联 关联模式。1 - 按标签的事件关联
	correlation_tag	字符	事关联使用的标签名称.
	url	字符	触器URL。
	status	整型 0	- enabled 触器状态。1 - disabled
	priority	整型 0	- 未分类 触发器严性。1 - 信息 2 - 警告 3 - 一般严重 4 - 严重 5 - 灾难
	description	文本	触器描述。
	type	整型 0	- 单个问题事件 事件生成类型。1 - 多个问题事件
	manual_close	整型 0	- 不允许 手工关闭 1 - 允许 题事件。
dependencies			依赖性的根元素
dependency			单独的依赖性。

元素元	属性类型	范围	说明	
	name	字符	依	触发的名称。
	expression	字符	依	触发器的表达式。
	recovery_expression	字符	依	触发器的恢复表达式。
tags				事件标签的根元素。
tag				单独的事件标签。
	tag	字符	标	名称。
	value	字符	标	值。

模板图形标签

元素元	属性类型	范围	说明	
graphs				图形的根元素。

元素元	属性类型	范围	说明	
graph				单独的图形。
	name	字符	图	名称。
	width	整型	用	素表示的图形宽度。饼图/爆炸图和预览使用。
	height	整型	用	素表示的图形高度。饼图/爆炸图和预览使用。
	yaxismin	双精度	如果	ymin_type_1’是1，那么这是Y轴的最小值。

元素元	属性类型	范围	说明	
	yaxismax	双精度	如果	ymax_type_1’ 是 1，那么这是 Y 轴的最大值。
	show_work_period	整型 0	- no 如 1 - yes	’type’ 是 0、1，突显非工作日。
	show_triggers	整型 0	- no 如 1 - yes	’type’ 是 0、1，以线条方式显示简单的触发器值。
	type	整型 0	- 正常 图形类 1 - 层积的 2 - 饼图 3 - 爆炸图 4 - 3D 饼图 5 - 3D 爆炸图	。
	show_legend	整型 0	- no 显 1 - yes	图形图例。

元素元	属性类型	范围	说明	
	show_3d	整型 0	- 2D 如 1 - 3D	’type’ 是 2、3，启用 3D 风格。
	percent_left	双精度	如果	type’ 是 0，显示左轴的百分位线。
	percent_right	双精度	如果	type’ 是 0，显示右轴的百分位线。
	ymin_type_1	整型 0	- 计算值 如果’t1 - 固定值 2 - 所选监控项的最新值	pe’ 是 0、1，这是 Y 轴的最小值。
	ymax_type_1	整型 0	- 计算值 如果’t1 - 固定值 2 - 所选监控项的最新值	pe’ 是 0、1，这是 Y 轴的最大值。

元素元	属性类型	范围	说明	
	ymin_item_1	字符 n	ll 或者监控项明细如果’ymin	type_1’是2，这是监控项明细。
	ymax_item_1	字符 n	ll 或者监控项明细如果’ymax	type_1’是2，这是监控项明细。
graph_items				图形监控项的根元素。
graph_item				单独的图形监控项。

元素元	属性类型	范围	说明	
	sortorder	整型	绘	顺序。先画较小的值。可以用它来画线条，或者另一个图形监控项的后面(或者前面)。
	drawtype	整型 0	- 单行 如果图 1 - 填充区域 2 - 粗线 3 - 虚线 4 - 短划线	‘type’是0，这是绘制风格。
	color	字符	元	颜色(6个符号，十六进制的)。

元素元	属性类型	范围	说明	
	yaxisside	整型 0	- 左轴 如果图 1 - 右轴	’type’ 是 0、1，这是元素所属的 Y 轴位置 (左或者右)。
	calc_fnc	整型 1	- 最小值 如果监控 2 - 平均值 4 - 最大值 7 - 所有值 (如果图形’type’ 是 0，这是最小值，平均值和最大值。) 9 - 最新值 (如果图形’type’ 不是 0 和 1)	有多个值存在，将要绘制的数据。
	type	整型 1	- 监控项的值按照比例绘制在饼图里。饼图/爆炸图的绘制类型。 2 - 监控项的值代表整个饼图 (图形求和)	
item				单独的监控项。
	host	字符	监	项的主机。
	key	字符	监	项的键。

模板 web 场景标签

元素元	属性类型	范围	说明	
httptests				web 场景的根元素。
httptest				单独的 web 场景。
	name	字符	w	b 场景名称。
	delay	字符	执	web 场景的频率。秒，带有后缀的时间单位或者用户宏。
	attempts	整型 1	10 执	Web 场景步骤的尝试次数。

元素元	属性类型	范围	说明	
	agent	字符	客	端 agent。Zab-bix 假装是所选的浏览器。当网站为不同的浏览器返回不同的内容的时候，这很有用处。
	http_proxy	字符	指	要使用的 HTTP 代理，使用这个格式：http://[user

元素元	属性类型	范围	说明	
	variables	文本	场	列表-可以在场景步骤中使用的级别变量(宏)。
	headers	文本	当	行请求的时候，要发送的HTTP头部。
	status	整型 0	- enabled w1 - disabled	b场景状态。
	authentication	整型 0	- none 认 1 - basic 2 - NTLM 认	方法。
	http_user	字符		用户名。
	http_password	字符	指	用户名的认证密码。

元素元	属性类型	范围	说明	
	verify_peer	整型 0	- no 校 1 - yes	web 服务器的 SSL 证书。
	verify_host	整型 0	- no 校 1 - yes	Web 服务器证书的 Common Name 字段或 Subject Alternate Name 字段是否匹配。
	ssl_cert_file	字符	客	端认证用到的 SSL 证书文件的名称。

元素元	属性类型	范围	说明	
	ssl_key_file	字符	客	端认证用到的SSL私钥文件的名称。
	ssl_key_password	字符	S	L私钥文件密码。
steps				web场景步骤的根元素。
step				单独的web场景步骤。
	name	字符	w	b场景步骤名称。
	url	字符	要	控的URL。
	posts	文本	’	ost’变量的列表。

元素元	属性类型	范围	说明	
	variables	文本	步	列表-这个步骤后面要应用到的级别变量(宏)。 如果变量值有'regex:'前缀，那么它的值将从按照'regex:'前缀后面的正则表达式模式而返回的数据里提取。

元素元	属性类型	范围	说明	
	headers	文本	当	行请求的时候，发送的HTTP头部。
	follow_redirects	整型 0	- no 跟 1 - yes	HTTP跳转。
	retrieve_mode	整型 0	- 内容 HTT1 - 仅 HTTP 头部	响应检索模式。
	timeout	字符	执	步骤的超时时间。秒，带后缀的时间单位或者用户宏。
	required	字符	必	字符。如果为空则忽略。

元素元	属性类型	范围	说明
	status_codes	字符	逗分隔的可接受的状态码列表。如果为空则忽略。例如：200-201,210-299。

4 网络拓扑图

概述

网络拓扑图导出（export） 包含：

- 所有相关的图片
- 拓扑图结构 - 所有拓扑图设置，所有包含元素及其设置，拓扑图链接和拓扑图链接状态指示器

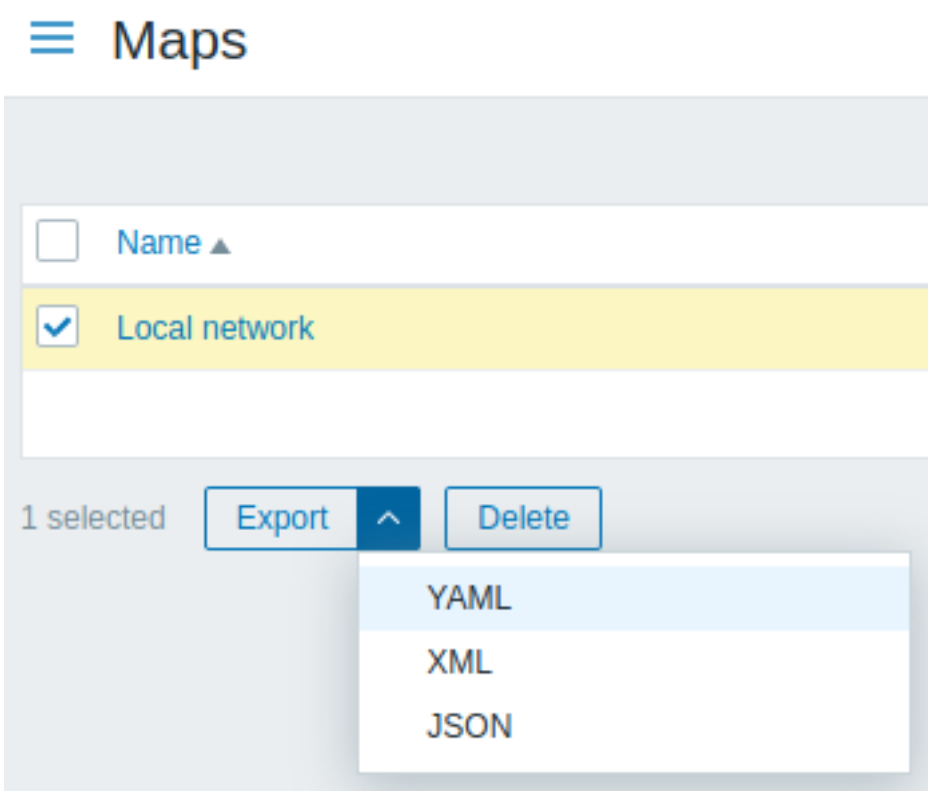
未导出的是主机组，主机，触发器，其他拓扑图或可能与导出的拓扑图相关的任何其他元素。因此，如果缺少拓扑图所引用的元素中的任何一个，导入将失败。

自 Zabbix 1.8.2 起支持网络拓扑图导出/导入。

导出

要导出网络拓扑图，请执行以下操作：

- 切换到：检测中（*Monitoring*） → 拓扑图（*Maps*）
- 标记要导出的网络拓扑图的复选框
- 单击列表下方的导出（*Export*）按钮

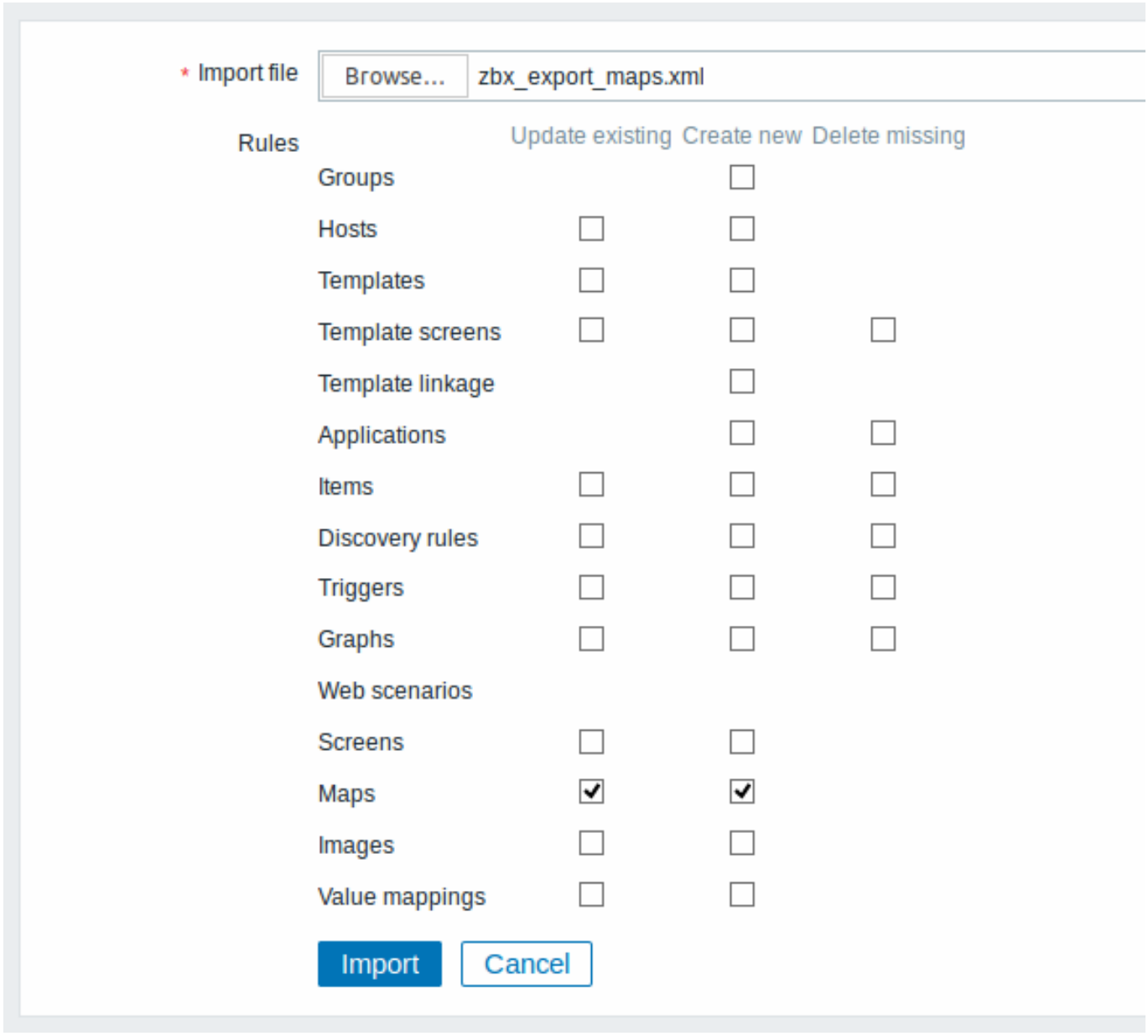


选中的拓扑图以默认名称 *zabbix_export_maps.xml* 导出到本地的 XML 文件里。

导入

要导入网络拓扑图，请执行以下操作：

- 切换到：监测中（*Monitoring*） → 拓扑图（*Maps*）
- 点击右侧的导入（*Import*）按钮
- 选择导入文件
- 在导入规则中标记所需选项
- 单击导入（*Import*）按钮



所有必填输入字段都标有红色星号。

导入成功或失败的消息将显示在前端。

导入规则：

规则说	
更新现有的（*Update existing*）将使用从导入	件中获取的数据更新现有拓扑图。否则他们将不会更新。
创建新的（*Create new*）导入将使用	入文件中的数据添加新拓扑图。否则它不会添加它们。

如果取消选中所有拓扑图选项并检查图像的相应选项，则仅导入图像。图像导入仅适用于 Zabbix Super Admin 用户。

<note warning> 如果替换现有图像，则会影响使用此图像的所有拓扑图。:::

导出格式

导出一个包含三个元素的小型网络图，它们的图像和它们之间的一些链接。请注意，图像被清空以节省空间。

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>4.0</version>
    <date>2016-10-05T08:16:20Z</date>
    <images>
        <image>
            <name>Server_(64)</name>
            <imagetype>1</imagetype>
```

```
            <encodedImage>iVBOR...SuQmCC</encodedImage>
        </image>
        <image>
            <name>Workstation_(64)</name>
            <imagetype>1</imagetype>
            <encodedImage>iVBOR...SuQmCC</encodedImage>
        </image>
        <image>
            <name>Zabbix_server_3D_(96)</name>
            <imagetype>1</imagetype>
            <encodedImage>iVBOR...ggg==</encodedImage>
        </image>
    </images>
    <maps>
        <map>
            <name>Network</name>
            <width>590</width>
            <height>400</height>
            <label_type>0</label_type>
            <label_location>0</label_location>
            <highlight>1</highlight>
            <expandproblem>0</expandproblem>
            <markelements>1</markelements>
            <show_unack>0</show_unack>
            <severity_min>2</severity_min>
            <grid_size>40</grid_size>
            <grid_show>1</grid_show>
            <grid_align>1</grid_align>
            <label_format>0</label_format>
            <label_type_host>2</label_type_host>
            <label_type_hostgroup>2</label_type_hostgroup>
            <label_type_trigger>2</label_type_trigger>
            <label_type_map>2</label_type_map>
            <label_type_image>2</label_type_image>
            <label_string_host/>
            <label_string_hostgroup/>
            <label_string_trigger/>
            <label_string_map/>
            <label_string_image/>
            <expand_macros>0</expand_macros>
            <background/>
            <iconmap/>
            <urls/>
            <selements>
                <selement>
                    <elementtype>0</elementtype>
                    <label>Host 1</label>
                    <label_location>-1</label_location>
                    <x>476</x>
                    <y>28</y>
                    <elementsubtype>0</elementsubtype>
                    <areatype>0</areatype>
                    <width>200</width>
                    <height>200</height>
                    <viewtype>0</viewtype>
                    <use_iconmap>0</use_iconmap>
                    <selementid>8</selementid>
                    <elements>
                        <element>
                            <host>Discovered host</host>
                        </element>
                    </elements>
```

```
                <icon_off>
                    <name>Server_(64)</name>
                </icon_off>
                <icon_on/>
                <icon_disabled/>
                <icon_maintenance/>
                <application/>
                <urls/>
            </selement>
            <selement>
                <elementtype>0</elementtype>
                <label>Zabbix server</label>
                <label_location>-1</label_location>
                <x>252</x>
                <y>50</y>
                <elementsubtype>0</elementsubtype>
                <areatype>0</areatype>
                <width>200</width>
                <height>200</height>
                <viewtype>0</viewtype>
                <use_iconmap>0</use_iconmap>
                <selementid>6</selementid>
                <elements>
                    <element>
                        <host>Zabbix server</host>
                    </element>
                </elements>
                <icon_off>
                    <name>Zabbix_server_3D_(96)</name>
                </icon_off>
                <icon_on/>
                <icon_disabled/>
                <icon_maintenance/>
                <application/>
                <urls/>
            </selement>
            <selement>
                <elementtype>0</elementtype>
                <label>New host</label>
                <label_location>-1</label_location>
                <x>308</x>
                <y>230</y>
                <elementsubtype>0</elementsubtype>
                <areatype>0</areatype>
                <width>200</width>
                <height>200</height>
                <viewtype>0</viewtype>
                <use_iconmap>0</use_iconmap>
                <selementid>7</selementid>
                <elements>
                    <element>
                        <host>Zabbix host</host>
                    </element>
                </elements>
                <icon_off>
                    <name>Workstation_(64)</name>
                </icon_off>
                <icon_on/>
                <icon_disabled/>
                <icon_maintenance/>
                <application/>
                <urls/>
```

```
                </selement>
            </selements>
            <links>
                <link>
                    <drawtype>0</drawtype>
                    <color>008800</color>
                    <label/>
                    <selementid1>6</selementid1>
                    <selementid2>8</selementid2>
                    <linktriggers/>
                </link>
                <link>
                    <drawtype>2</drawtype>
                    <color>00CC00</color>
                    <label>100MBps</label>
                    <selementid1>7</selementid1>
                    <selementid2>6</selementid2>
                    <linktriggers>
                        <linktrigger>
                            <drawtype>0</drawtype>
                            <color>DD0000</color>
                            <trigger>
                                <description>Zabbix agent on {HOST.NAME} is unreachable for 5 minutes</des
                                <expression>{Zabbix host:agent.ping.nodata(5m)}=1</expression>
                                <recovery_expression/>
                            </trigger>
                        </linktrigger>
                    </linktriggers>
                </link>
            </links>
        </map>
    </maps>
</zabbix_export>
```

元素标签

元素标签值在下表中说明。

元素元	属性类型	范围	说明	
images				图像的根元素。
image				单独的图像。
	name	字符	唯	图像名称。
	imagetype	整型 1	- 图像 图像类 2 - 背景	。

元素元	属性类型	范围	说明
	encodedImage		Base64 编码图像。
maps			拓扑图的根元素。
map			单独的拓扑图。
	name	字符	唯 拓扑图名称。
	width	整型	拓 图宽度，以像素为单位。
	height	整型	拓 图高度，以像素为单位。
	label_type	整型 0	- 标签 拓扑图 1 - 主机 IP 地址 2 - 元素名称 3 - 仅状态 4 - 无 素标签类型。

元素元	属性类型	范围	说明	
	label_location	整型 0	- 底部 默认情 1 - 左 2 - 右 3 - 顶部	下拓扑图元素标签位置。
	highlight	整型 0	- no 为 1 - yes	动触发器和主机状态启用图标突出显示。
	expandproblem	整型 0	- no 显 1 - yes	具有单个问题的元素的问题触发器。
	markelements	整型 0	- no 突 1 - yes	显示最近更改其状态的拓扑图元素。

元素元	属性类型	范围	说明	
	show_unack	整型 0	- 所有问题的数量 问题显示。1 - 未确认问题的数量 2 - 分别统计已确认和未确认的问题	
	severity_min	整型 0	- 未分类 默认情况 1 - 信息 2 - 警告 3 - 一般严重 4 - 严重 5 - 灾难	显示在拓扑图上的最小触发严重性。
	grid_size	整型 2	, 40, 50, 75 或者 100 如果 “	rid_show = 1”, 这是拓扑图网格的单元格大小(以像素为单位)。
	grid_show	整型 0	- yes 在 1 - no	扑图配置中显示网格。

元素元	属性类型	范围	说明	
	grid_align	整型 0	- yes 在 1 - no	扑图配置中自动对齐图标。
	label_format	整型 0	- no 使 1 - yes	高级标签配置。
	label_type_host	整型 0	- 标签 如果 “1 - 主机 IP 地址 2 - 元素名称 3 - 仅状态 4 - 无 5 - 自定义标签	abel_format = 1”，则显示为主机标签。
	label_type_hostgroup	整型 0	- 标签 如果 “2 - 元素名称 3 - 仅状态 4 - 无 5 - 自定义标签	abel_format = 1”，则显示为主机组标签
	label_type_trigger	整型 0	- 标签 如果 “2 - 元素名称 3 - 仅状态 4 - 无 5 - 自定义标签	abel_format = 1”，则显示为触发器标签

元素元	属性类型	范围	说明	
	label_type_map	整型 0	- 标签 如果 “2 - 元素名称 3 - 仅状态 4 - 无 5 - 自定义标签	abel_format = 1”，则显示为拓扑图标签
	label_type_image	整型 0	- 标签 显示为 2 - 元素名称 4 - 无 5 - 自定义标签	像标签，如果 “la-bel_format = 1”
	label_string_host	字符	如	“label_type_hos = 5”，这是主机元素的自定义标签。
	label_string_hostgroup	字符	如	“label_type_hos = 5”，这是主机组元素的自定义标签。

元素元	属性类型	范围	说明	
	label_string_trigger	字符	如	“label_type_trig = 5”，这是触发元素的自定义标签。
	label_string_map	字符	如	“label_type_ma = 5”，则是拓扑图元素的自定义标签
	label_string_image	字符	如	“label_type_ima = 5”，则是图像元素的自定义标签
	expand_macros	字符 0	- no 在 1 - yes	扑图配置中展开标签中的宏。

元素元	属性类型	范围	说明	
	background	id		如果“imagetype = 2”，则是背景图像的ID（如果有）
	iconmap	id		图标映射的ID（如果有）。
urls url				单独的URL。
	name	字符	链	名称。
	url	字符	链	URL。
	elementtype	整型 0	- 主机 链接所 1 - 拓扑图 2 - 触发器 3 - 主机组 4 - 图像	的拓扑图监控项类型。
selements selement				单独的拓扑图元素

元素元	属性类型	范围	说明	
	elementtype	整型 0	- 主机 拓扑图 1 - 拓扑图 2 - 触发器 3 - 主机组 4 - 图像	素类型。
	label	字符	图	标签。
	label_location	整型 -	- 使用拓扑图默认 0 - 底部 1 - 左 2 - 右 3 - 顶部	
	x	整型	X	上的位置。
	y	整型	Y	上的位置。
	elementsubtype	整型 0	- 单个主机组 如果 “Ele1 - 所有主机组	entType=3”，则是元素子类型
	areatype	整型 0	- 与整个拓扑图相同 如果 “ele-men1 - 自定义大小	subtype = 1”，则是区域大小
	width	整型	如	“areatype = 1”，则是面积宽度
	height	整型	如	“areatype = 1”，则是面积高度

元素元	属性类型	范围	说明	
	viewtype	整型 0	- 均匀地放在该区域如果“elemen	subtype = 1”，则是区域放置算法
	use_iconmap	整型 0	- no 使 1 - yes	此元素的图标映射。仅在拓扑图级别激活图标映射时才相关。
	selementid	id		唯一元素记录 ID。

元素元	属性类型	范围	说明	
	application	字符	应	集名称过滤器。如果给出了应用集程序名称，则只会在拓扑图上显示属于给定应用集程序的触发器问题。

elements

元素元	属性类型	范围	说明
element	host		在拓扑图上表示的单个Zabbix实体(拓扑图，主机组，主机等)。
icon_off			元素处于“正常”状态时使用的图像。
icon_on			元素处于“问题”状态时使用的图像。

元素元	属性类型	范围	说明	
icon_disabled				禁用元素时要使用的图像。
icon_maintenance				元素处于维护状态时使用的图像。
	name	字符	唯	的图像名称。
links				
link				拓扑图元素之间的个别链接。
	drawtype	整型 0	- 线条 线条类 2 - 粗线条 3 - 虚线 4 - 中划线	。

元素元	属性类型	范围	说明	
	color	字符	链	颜色(6个符号，十六进制)。
	label	字符	链	标签。
	selementid1	id		要连接的一个元素的ID。
	selementid2	id		要连接的其他元素的ID。
linktriggers linktrigger				单独的链接状态指示灯。
	drawtype	整型 0	- 线条 触发器 2 - 粗线条 3 - 虚线 4 - 中划线	于“问题”状态时的链接样式。

元素元	属性类型	范围	说明
	color	字符	当发器处于“问题”状态时，链接颜色(6个符号，十六进制)。
trigger			触发器用于指示链路状态。
	description	字符	触器名称。
	expression	字符	触器表达式。
	recovery_expression	字符	触器恢复表达式。

5 聚合图形

概述

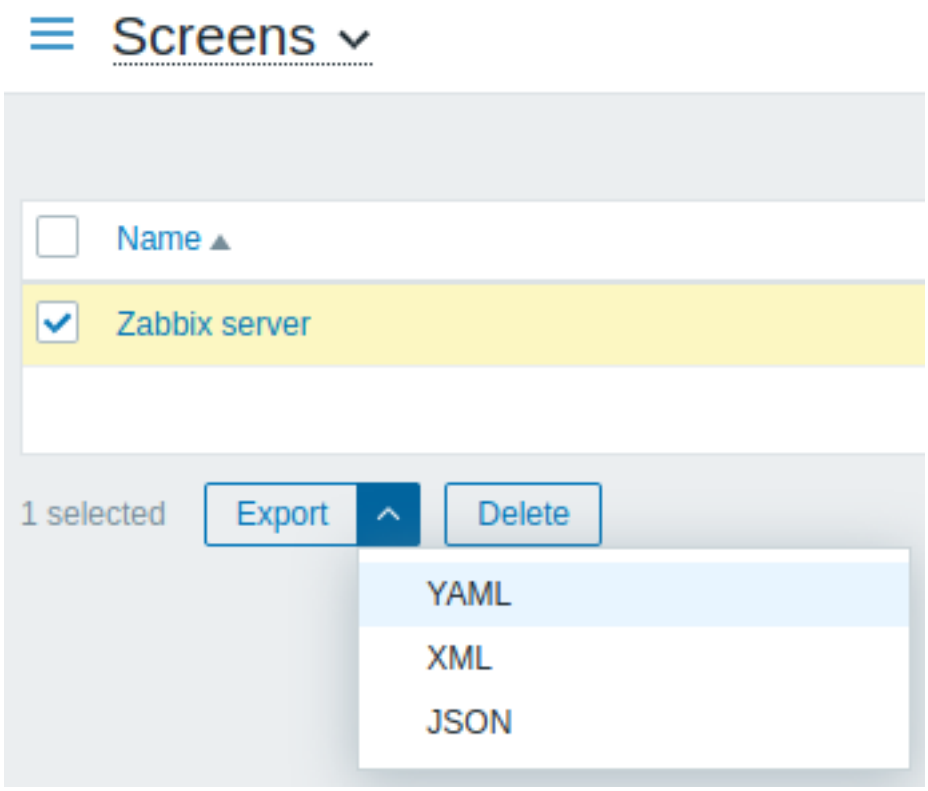
聚合图形导出（export） 包含聚合图形的结构 - 所有聚合图形设置和所有聚合图形元素及其配置。

聚合图形本身中包含的任何内容（如主机，主机组或任何其他数据）都不会导出。因此，如果聚合图形所指的元素中有任何一个缺失，则导入它将失败。

导出

要导出聚合图形，请执行以下操作：

- 切换到：监测中（*Monitoring*） → 聚合图形（*Screens*）
- 标记要导出的聚合图形的复选框
- 单击列表底部的导出（*Export*）按钮

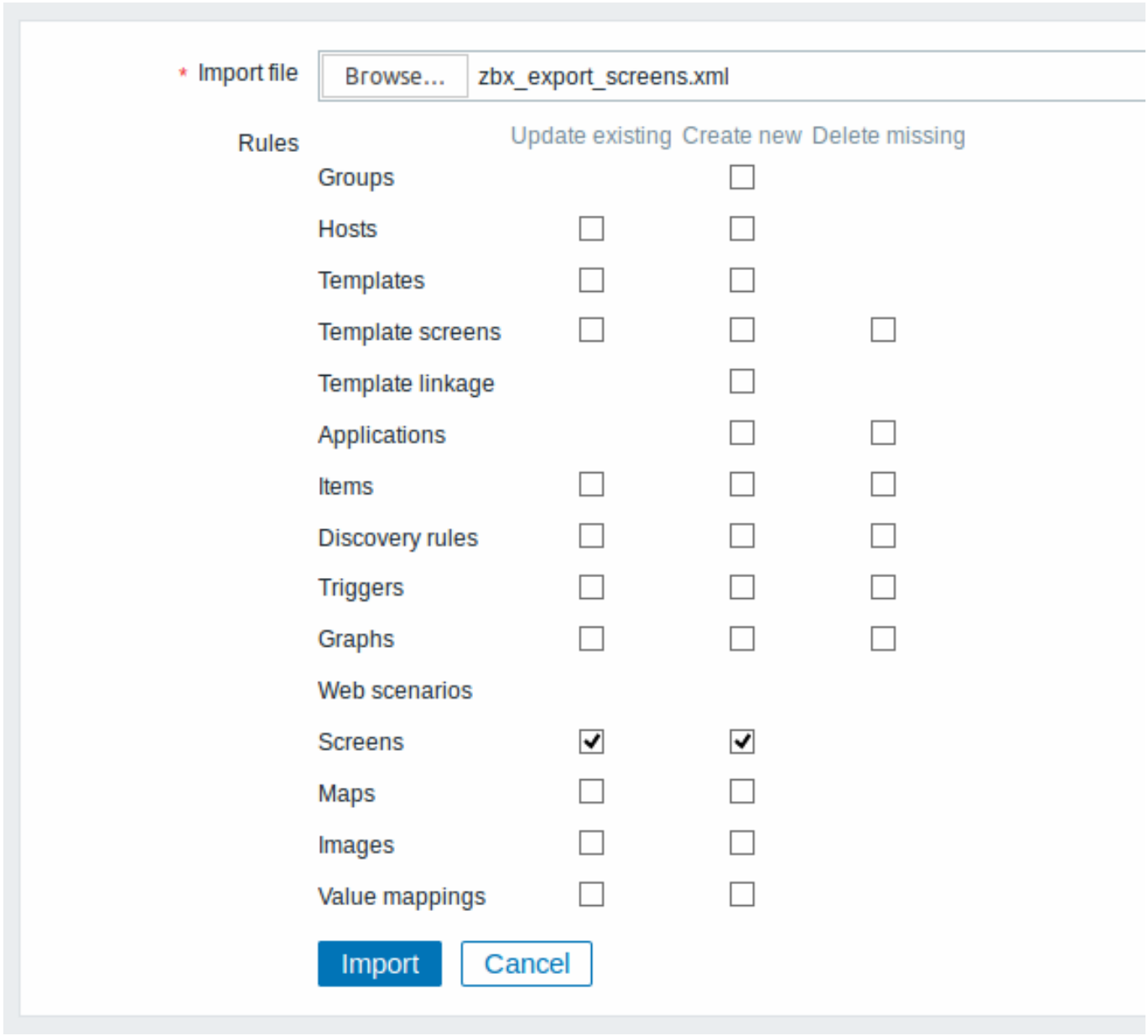


选定的聚合图形将导出到本地 XML 文件，默认名称为 *zabbix_export_screens.xml*。

导入

要导入聚合图形，请执行以下操作：

- 切换到：监测中（*Monitoring*） → 聚合图形（*Screens*）
- 单击右侧的导入（*Import*）按钮
- 选择导入文件
- 在导入规则中标记所需选项
- 单击导入（*Import*）按钮



所有必填输入字段都标有红色星号。

导入成功或失败的消息将显示在前端。

导入规则：

规则说	
更新已有的（*Update existing*）将使用从导入	件中获取的数据更新现有聚合图形。否则他们将不会更新。
创建新的（*Create new*）导入将使用	入文件中的数据添加新屏幕。否则它不会添加它们。

导出格式

导出一个由两个图形占据第一行的聚合图形。

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>4.0</version>
    <date>2016-10-07T08:02:40Z</date>
    <screens>
        <screen>
            <name>Zabbix server</name>
            <hsize>2</hsize>
            <vsize>3</vsize>
            <screen_items>
                <screen_item>
```

```
                    <resourcetype>0</resourcetype>
                    <width>300</width>
                    <height>80</height>
                    <x>0</x>
                    <y>0</y>
                    <colspan>1</colspan>
                    <rowspan>1</rowspan>
                    <elements>0</elements>
                    <valign>0</valign>
                    <halign>0</halign>
                    <style>0</style>
                    <url/>
                    <dynamic>1</dynamic>
                    <sort_triggers>0</sort_triggers>
                    <resource>
                        <name>CPU load</name>
                        <host>Zabbix host</host>
                    </resource>
                    <max_columns>3</max_columns>
                    <application/>
                </screen_item>
                <screen_item>
                    <resourcetype>0</resourcetype>
                    <width>300</width>
                    <height>80</height>
                    <x>1</x>
                    <y>0</y>
                    <colspan>1</colspan>
                    <rowspan>1</rowspan>
                    <elements>0</elements>
                    <valign>0</valign>
                    <halign>0</halign>
                    <style>0</style>
                    <url/>
                    <dynamic>1</dynamic>
                    <sort_triggers>0</sort_triggers>
                    <resource>
                        <name>CPU utilization</name>
                        <host>Zabbix host</host>
                    </resource>
                    <max_columns>3</max_columns>
                    <application/>
                </screen_item>
            </screen_items>
        </screen>
    </screens>
</zabbix_export>
```

元素标签

元素标签值在下表中说明。

元素元	属性类型	范围	说明
screens			
screen			
	name	字符	独的聚合图形名称。

元素元	属性类型	范围	说明	
	hsize	整型	水	尺寸，列数。
	vsize	整型	垂	大小，行数。
screen_items screen_item				
	resourcetype	整型 0	- 图形 资源类 1 - 简单图形 2 - 地图 3 - 纯文本 4 - 主机信息 5 - 触发器信息 6 - 服务器信息 7 - 时钟 8 - 聚合图形 9 - 触发器概述 10 - 数据概述 11 - URL 12 - 历史动作 13 - 历史事件 14 - 主机组问题 15 - 按严重性划分问题 16 - 主机问题 19 - 简单图形原型 20 - 图形原型	

元素元	属性类型	范围	说明	
	width	整型	如	“resourcetype”为0,1,7,11,19或20，则聚合图形监控项的宽度(以像素为单位)。
	height	整型	如	’resourcetype’为0,1,7,11,19或20，则聚合图形监控项的高度(以像素为单位)。

元素元	属性类型	范围	说明	
	x	整型	聚	图形上的聚合图形监控项的X坐标，从左到右。'0'表示从第一列开始。
	y	整型	聚	图形上聚合图形监控项的Y坐标，从上到下。'0'表示从第一行开始。

元素元	属性类型	范围	说明	
	colspan	整型	聚	图形监控项跨越的列数。
	rowspan	整型	聚	图形监控项将跨越的数量或行。
	elements	整型	如	’resourcetype’为3,12,13,14或16，则在聚合图形监控项上显示的行数。
	valign	整型 0	- 中（默认） 垂直对齐。 1 - 顶部 2 - 底部	
	halign	整型 0	- 中（默认） 水平对齐。 1 - 左 2 - 右	

元素元	属性类型	范围	说明	
	style	整型 0	- 纯文本 如果’r1 - HTML	sourcetype’为3，则显示聚合图形监控项的选项。
		整型 0	- 本地时间 如果’re1 - 服务器时间 2 - 主机时间	ourcetype’为7，则显示聚合图形监控项的选项。
		整型 0	- 水平 如果’1 - 垂直	esourcetype’是4,5，则显示聚合图形监控项的选项。

元素元	属性类型	范围	说明	
		整型 0	- 左侧 如果’1 - 顶部	esourcetype’ 是 9,10，则显示聚合图形监控项的选项。
	url	字符	如	’resourcetype’ 为 11，则是链接 URL。
	dynamic	整型 0	- no 如 1 - yes	’resourcetype’ 为 0,1,3,19 或 20，则使聚合图形监控项动态化。
	sort_triggers	整型 0	- 最后一次改变（降序）如果’resource1 - 严重程度（降序）2 - 主机（升序）	ype’ 是 14,16，则对触发器进行排序的选项。

元素元	属性类型	范围	说明	
		整型 3	- 时间（升序） 如果‘reso4 - 时间（降序） 5 - 类型（升序） 6 - 类型（降序） 7 - 状态（升序） 8 - 状态（降序） 9 - 剩余重试次数（升序） 10 - 剩余重试次数（降序） 11 - 收件人(升序） 12 - 收件人(降序）	rcetype’ 为 12，则对触发器进行排序的选项。

元素元	属性类型	范围	说明	
	max_columns	整型	如	’resourcetype’ 为 19 或 20，则应在聚合图形单元格中显示生成的图表数量。当有许多低级别发现生成的图表时很有用。
	application	字符	如	’resourcetype’ 为 9 或 10，则按应用集名称过滤。

resource

元素元	属性类型	范围	说明
	name	字符	资 名称。
	host	字符	资 主机。

15. 发现

请点击侧边目录栏阅读本章内容

1 网络发现

概述

Zabbix 为用户提供了高效灵活的网络自动发现功能。

适当的网络发现配置可以：

- 加快 Zabbix 部署
- 简化管理
- 无需过多管理，也能在快速变化的环境中使用 Zabbix

Zabbix 网络发现基于以下信息：

- IP 范围
- 可用的外部服务（FTP，SSH，WEB，POP3，IMAP，TCP 等）
- 来自 zabbix agent 的信息（仅支持未加密模式）
- 来自 snmp agent 的信息

不支持：

- 发现网络拓扑

网络发现由两个阶段组成: 发现（discovery）和动作（actions）。

发现

Zabbix 定期检测网络发现规则中定义的 IP 范围，并为每个规则单独配置检查的频次。

请注意，一个发现规则始终由单一发现进程处理，IP 范围主机不会被分拆到多个发现进程处理。

每个规则中都定义了一组需要检测的服务。

Note:
发现检查与其他检查独立处理。如果任何检查未找到服务（或失败），则仍会处理其他检查。

网络发现模块每次检测到 service 和 host(IP) 都会生成一个 discovery 事件

事件名称对应的	查结果
Service Discovered	服务首次被发现或者由’down’ 变’up’
Service Up	服务持续’up’
Service Lost	服务由’up’ 变’down’
Service Down	服务持续’down’
Host Discovered	在主机的所有服务都’down’ 之后，至少一个服务是’up’。
Host Up	主机至少有一个服务是’up’ 状态
Host Lost	主机的所有服务在至少一个是’up’ 之后全部是’down’。
Host Down	所有服务都持续’down’

动作

Zabbix 所有动作都是基于发现事件, 例如:

- 发送通知
- 添加/删除主机
- 启用/禁用主机
- 添加主机到组
- 从组中删除主机
- 将主机链接到/取消链接模板
- 执行远程脚本命令

基于事件的网络发现动作, 可以根据设备类型、IP 地址、状态、运行时间/停机时间等进行配置，查看操作 and 条件页面。

创建主机

如果在动作 → 操作选择添加主机操作，那么主机会被添加，即使添加主机操作未被执行，通过下列的操作仍然可以添加主机，这样的操作是：

- 启用主机
- 禁用主机
- 添加主机到主机组
- 将主机链接到模板

当添加主机时，如果反向查找失败，那么主机名就是 DNS 反向查找的结果或者是 IP 地址。查找是从 Zabbix 服务器或 Zabbix 代理执行的，具体取决于自动发现的执行。如果在 Zabbix proxy 上查找失败，则不会在 Zabbix server 上重试。如果具有相同名称的主机已经存在，那么下一个主机将会把 _2 附加在主机名后，依次附加 _3 等。

创建的主机会被添加到主机群组中的 Discovered hosts 下（默认情况下，在管理 → 一般 →其他 可以进行配置），如果希望将主机添加到另一个主机群组中，可以从动作 → 操作选择添加一个 从主机群组中删除的操作类型（需要指定 “Discovered hosts”），当然也可以选择添加到主机群组的操作类型（需要指定其他的主机群组），因为主机必须属于主机群组。

如果主机已经存在，且自动发现中同时存在已发现的 IP 地址，那么将不会创建新的主机，但是，如果自动发现的操作包含（链接模板，添加到主机群组等），则会在已经存在的主机上执行相应的操作。

移除主机

从 Zabbix 2.4.0 开始，如果已发现的实体不在自动发现规则的 IP 范围内，则由网络发现规则创建的主机将会被自动删除。主机将立即删除

添加主机时的创建接口

当网络自动发现, 添加主机时，它们的接口根据以下规律来创建的:

- 检测到服务 - 例如，如果 SNMP 检查成功，那么将会创建一个 SNMP 接口；
- 如果主机响应 Zabbix agent 和 SNMP 的请求，那么这两种类型的接口都会被创建；
- 如果唯一性准则是是 Zabbix agent 键值或是 SNMP OID 返回的数据，这第一个接口发现的主机将会被创建，而这个接口将会被作为默认接口，其他 IP 地址将会作为附加接口被添加。
- 如果主机只响应 agent 检查，则只能创建 agent 接口。如果稍后开始响应 SNMP 的检查，那么将添加 SNMP 接口为附加接口。
- 如果最初创建了 3 个独立的主机，他们都被自动发现的唯一性准则 “IP” 发现，然后修改自动发现规则，为了使 A、B 和 C 自动发现的唯一性准则结果是相同的，那么接口 B 和 C 作为接口 A 的附加接口来创建第一个主机。主机 B 和 C 作为个体主机仍然存在。在监控中 → 自动发现中，添加的接口将以黑色字体和缩进形式显示在 “已发现的设备” 这一列中，但在 “已监控的主机” 这一列将只显示第一个创建的主机 A。由于被认为附加接口的 IP，所以不测量主机 B 和 C 的 “在线时间/断线时间”。

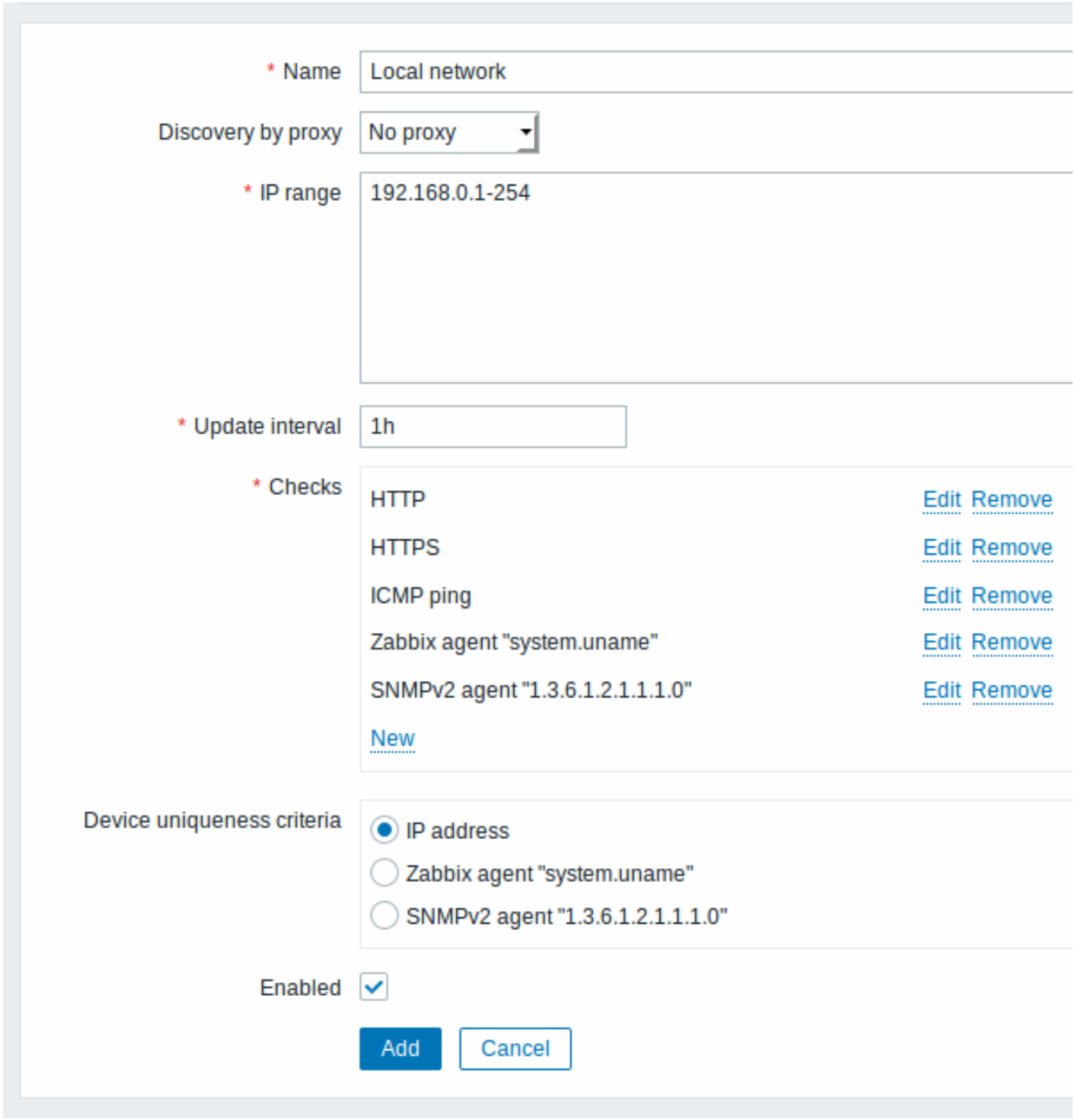
1 配置网络发现规则

概述

配置 Zabbix 的网络发现规则来发现主机和服务：

- 首先进入 配置 → 自动发现
- 单击 创建发现规则（*Create rule*）（或在自动发现规则名称上编辑现有规则）
- 编辑自动发现规则属性

规则属性



参数描	
名称（*Name*）规则名	，唯一。例如：“Local network”。

参数描	
通过代理发现 *(Discovery by proxy)* 谁执行当前	现规则:What performs discovery: **no proxy** - Zabbix server 执行发现 **<proxy name>** - 这个 proxy 执行

参数	描
IP 范围（*IP range)* 发现	则中的IP地址范围.可能的格式如下:单个IP: 192.168.1.33IP段: 192.168.1-10.1-255.范围受限于覆盖地址的总数(小于64K)。子网掩码:: 192.168.4.0/24支持的子网掩码: /16-/30 for IPv4 addresses/112-/128 for IPv6 ad-

参数描	
Update interval	This parameter defines how often Zabbix will execute the rule. The interval is measured after the execution of previous discovery instance ends so there is no overlap. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros

参数描	
检查 *(Checks)* Z	bbix 将使用这个检查列表进行发现。支持的 checks: SSH, LDAP, SMTP, FTP, HTTP, HTTPS, POP, NNTP, IMAP, TCP, Tel-net, Zab-bix agent, SN-MPv1 agent, SN-MPv2 agent, SN-MPv3 agent, ICMP ping. 基于协议的发现使用 **net.tcp.servic** f 功能测试每个主机，但不

参数	描
设备唯一标识 *(Device uniqueness criteria)*	唯一标准如： **IP 地址** - 使用 IP 地址作为设备唯一性标识，不处理多 IP 设备。如果具有相同 IP 的设备已经存在，则将认为已经发现，并且不会添加新的主机。 发现检查类型

参数描	
启用（*Enabled*）	Wit the check-box marked the rule is ac-tive and will be ex-e-cuted by Zab-bix server. If un-marked, the rule is not ac-tive. It won't be ex-e-cuted.

修改代理 (proxy) 设置

从 Zabbix 2.2.0 起，不同的代理发现的主机被认为是不同的主机。虽然这允许在不同子网使用相同的 IP 段执行发现，但是对已监测子网改变代理非常复杂，是因为代理的变化也必须应用于所有发现的主机。例如，在发现规则中替换代理的步骤如下：

1. 禁用发现规则
2. 同步代理配置
3. 替换发现规则中的代理
4. 替换由此规则发现的所有主机的代理
5. 启用发现规则

真实使用场景

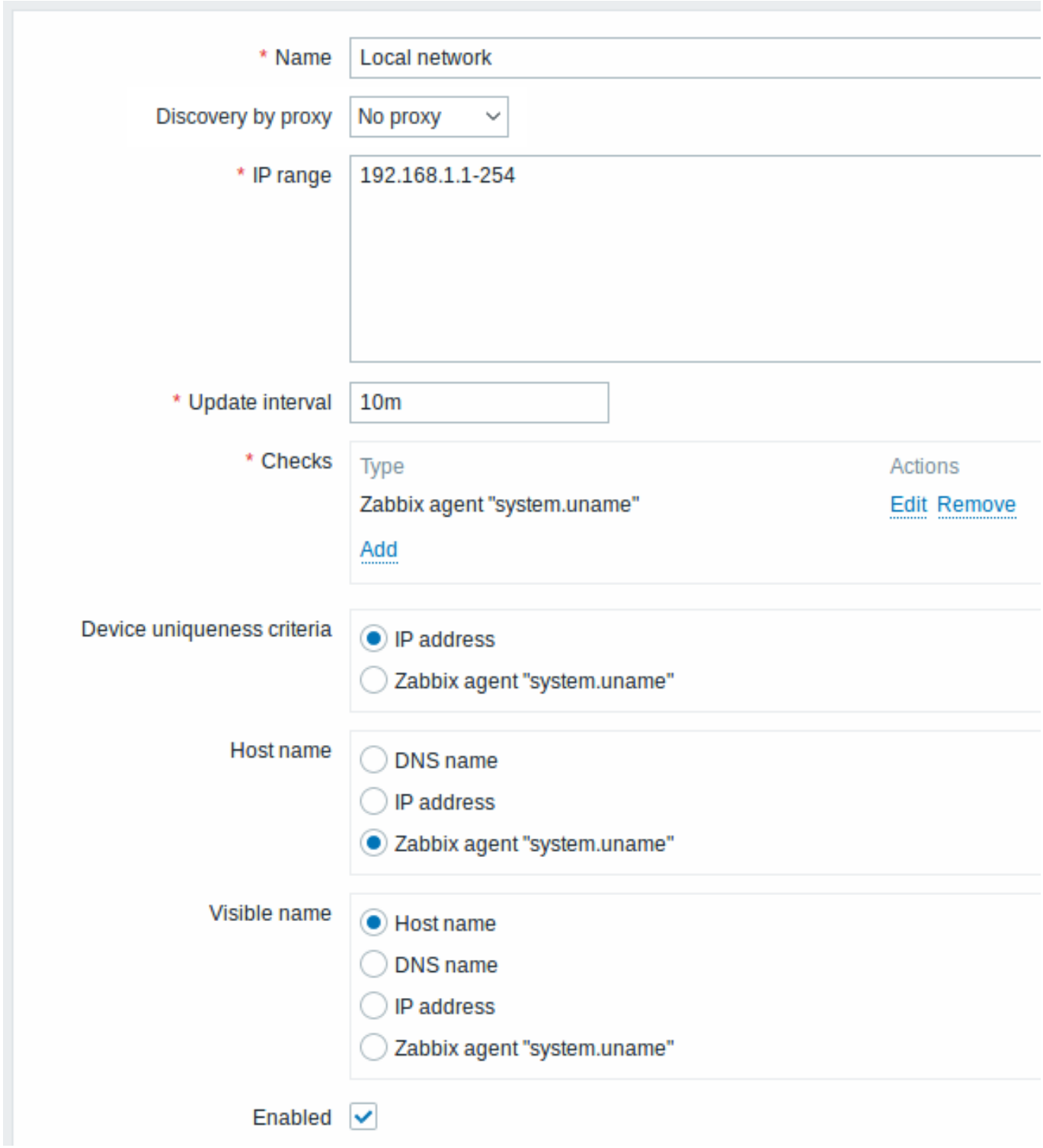
例如我们设置 IP 段为 192.168.1.1-192.168.1.254 的网络发现规则。

在我们的例子中，我们需要：

- 发现有 Zabbix agent 运行的主机
- 每 10 分钟执行一次
- 如果主机正常运行时间超过 1 小时，添加主机
- 如果主机停机时间超过 24 小时，删除主机
- 将 Linux 主机添加到 “Linux servers” 组
- 将 Windows 主机添加到 “Windows servers” 组
- 链接模板 *Template OS Linux* 到 Linux 主机
- 链接模板 *Template OS Windows* 到 Windows 主机

步骤 1

首先给我们的 IP 段定义网络发现规则。



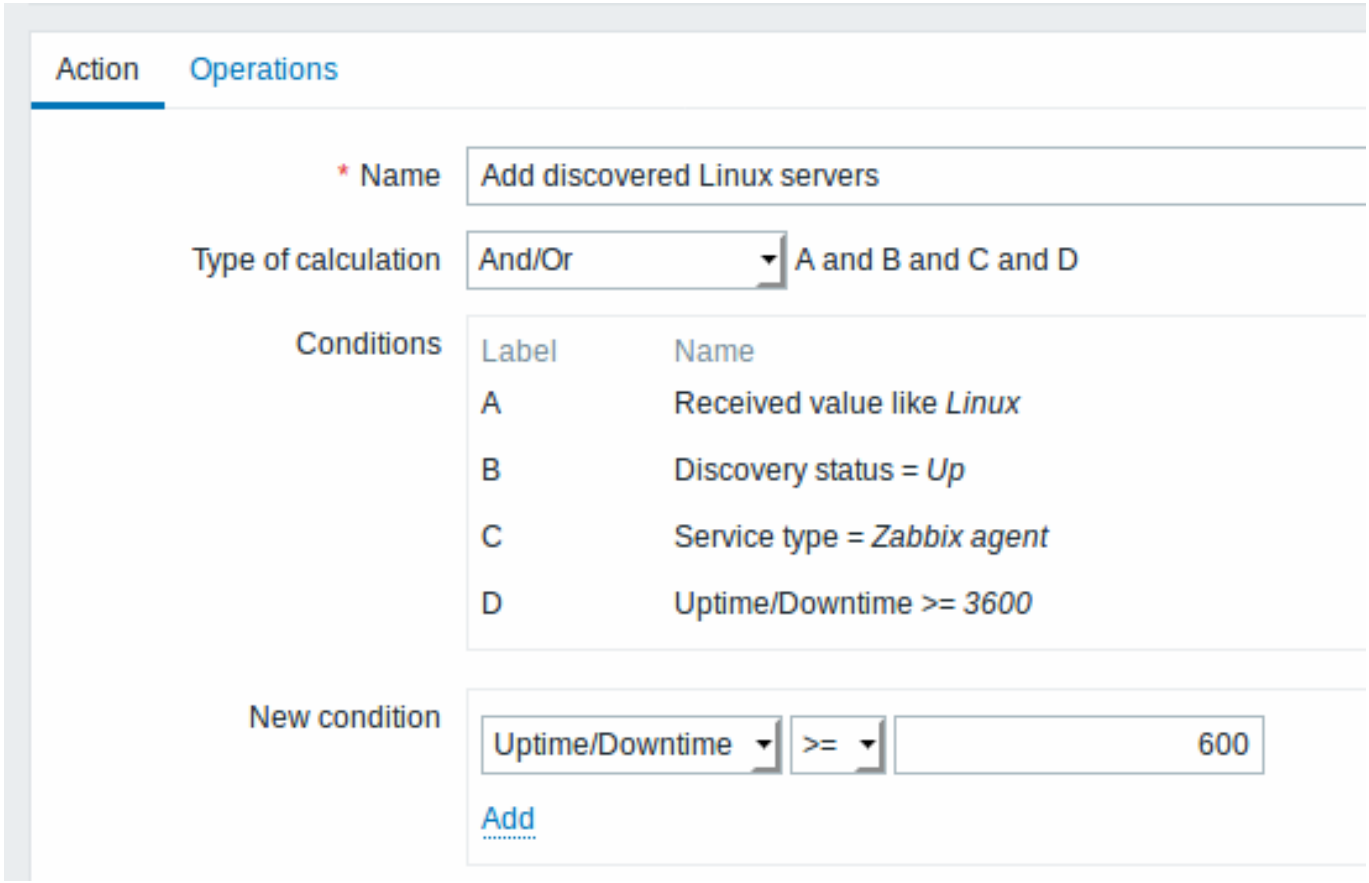
Zabbix 试图通过连接 Zabbix agents 并获取 **system.uname** 键值来发现 IP 段为 192.168.1.1-192.168.1.254 中的主机。根据不同键值来对应不同的操作系统的不同操作。根据不同键值来对应不同的操作系统的不同操作。例如将 Windows 服务器链接到 Template OS Windows，将 Linux 服务器链接到 Template OS Linux。

规则将每 10 分钟（600 秒）执行一次。

当规则添加后，Zabbix 将自动执行发现规则并生成基于发现的事件做后续处理。

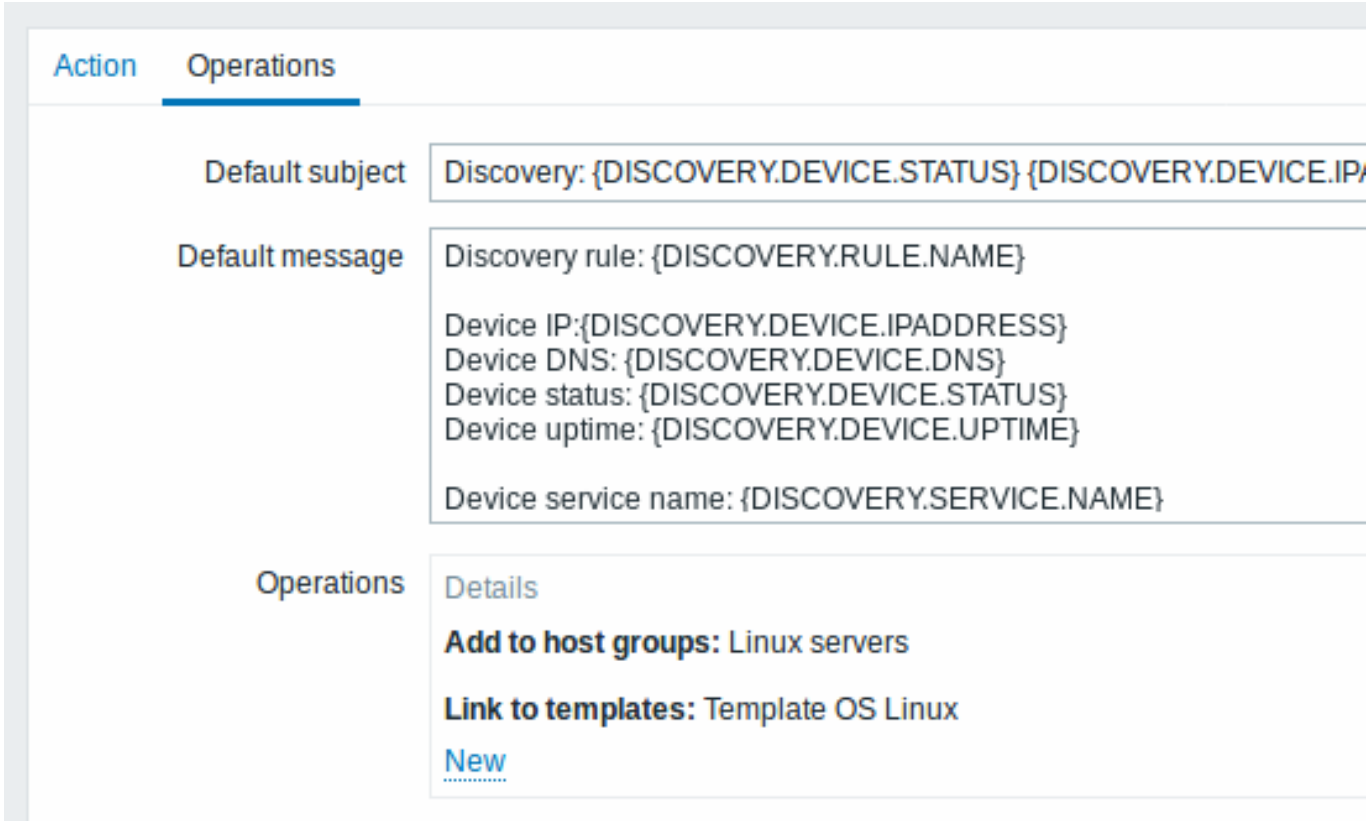
步骤 2

定义动作（action） 将所发现的 Linux 服务器添加到相应的组/模板



如果发生以下情况，动作 (action) 将被激活:

- “Zabbix agent” 服务是 “up”
- system.uname(规则中定义的 Zabbix agent 键值) 包含 “Linux”
- 正常运行时间为 1 小时（3600 秒）或更长

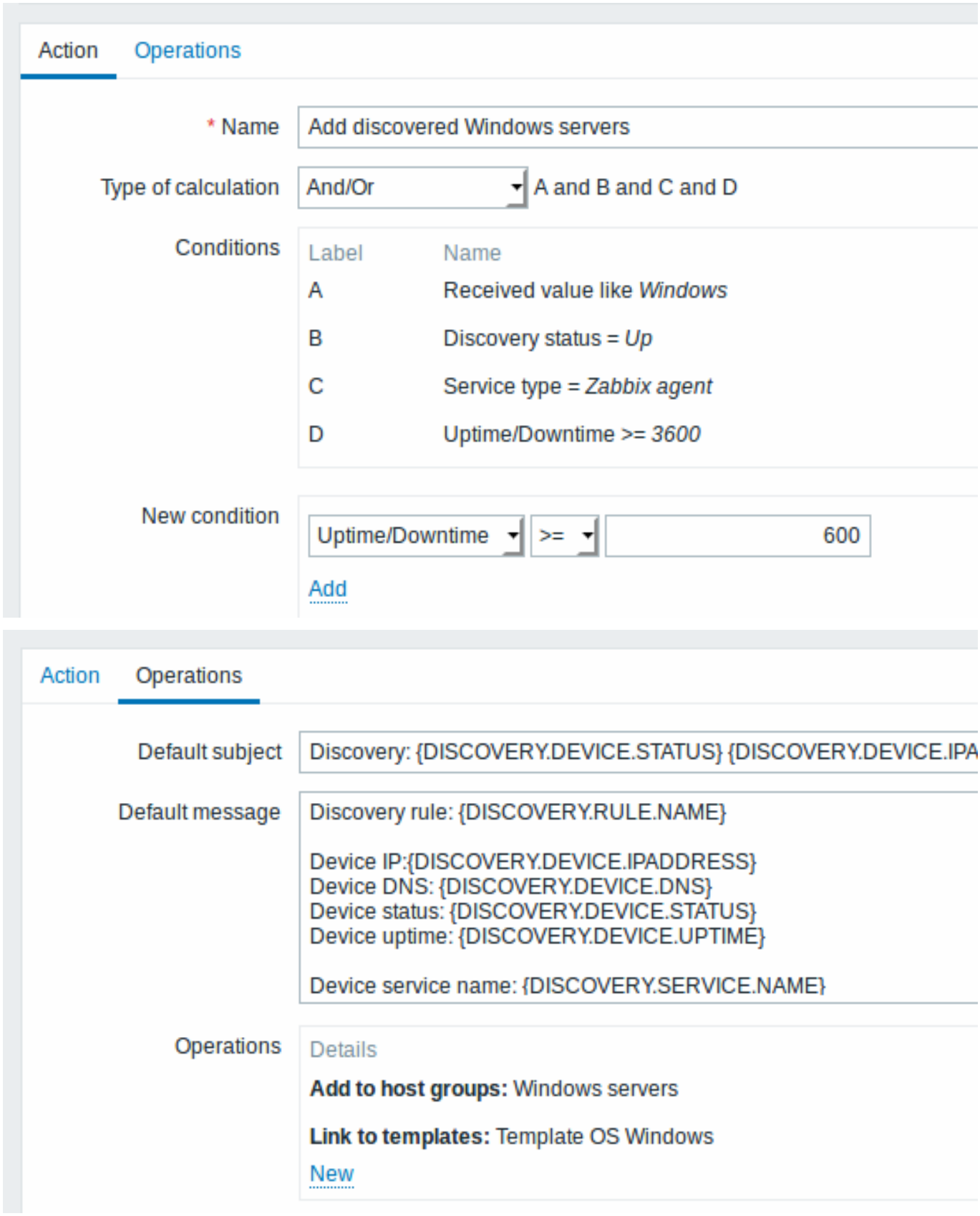


该动作 (action) 将执行以下操作：

- 将发现的主机添加到 “Linux servers” 组（如果以前未添加主机，也添加主机）
- 链接主机到 “Template OS Linux” 模板。Zabbix 将自动开始使用 “Template OS Linux” 模板中的项目和触发器来监控主机。

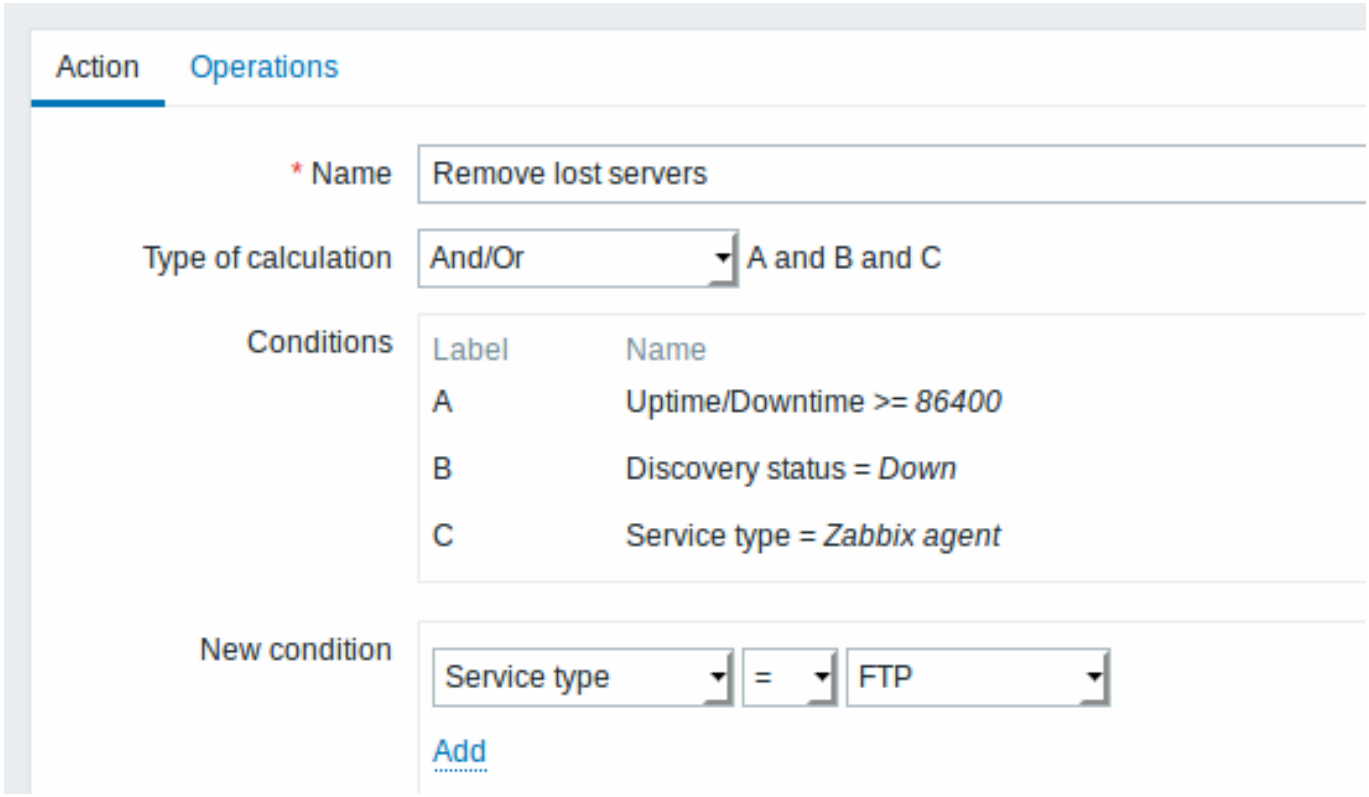
步骤 3

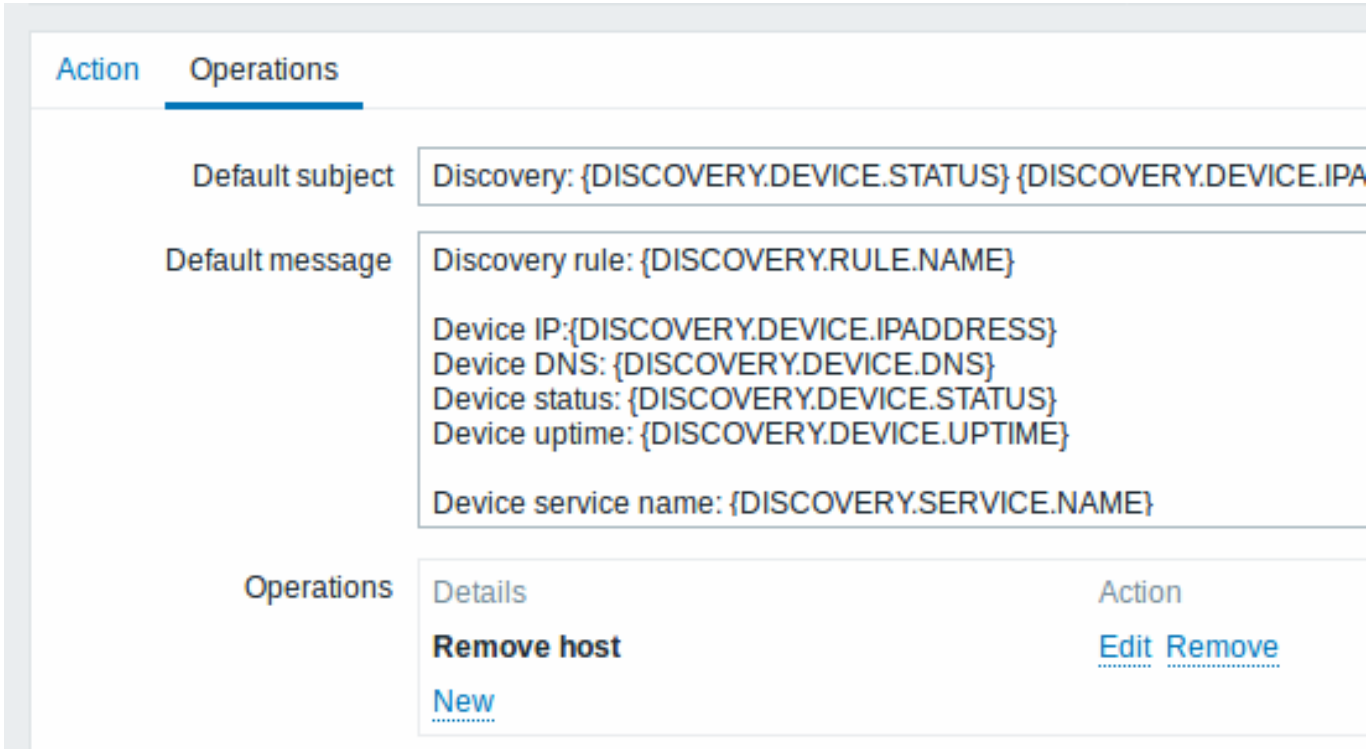
定义动作 (action) 将所发现的 Windows 服务器添加到相应的组/模板



步骤 4

定义动作删除失联主机





如果 “Zabbix agent” 服务’down’ 超过 24 小时（86400 秒），服务器将被删除。

2 Active agent 自动注册

概述

Zabbix Active agent 可以实现自动注册，进而服务器对其进行监控。通过这种方式，无需在服务器上进行手动配置便可直接启动对新 host 的监控。

当以前未知的 active agent 要求检查时，会发生自动注册。

这样功能可以非常方便的自动监控新的 Cloud 节点。一旦在 Cloud 中有一个新节点，Zabbix 将自动启动 host 的性能和可用性数据的收集。

Active agent 自动注册还支持对被添加的主机进行被动检查的监控。当 active agent 要求检查时，前提是在配置文件中已定义好了“ListenIP” 或 “ListenPort” 配置参数，这些参数将发送到服务器。(如果指定了多个 IP 地址，则第一个将被发送到服务器。)

服务器在添加新的自动注册主机时，使用接收到的 IP 地址和端口配置 agent。如果没有接收到 IP 地址值，则使用传入连接的 IP 地址。如果没有接收到端口值，则使用 10050。

以下情况下，自动注册会自动运行：

- 主机元数据信息发生变化
- 手动添加主机，元数据信息有缺失
- 手动切换主机，由另一台新的 proxy 监控
- 同一台 host 的自动注册由新的 proxy 发出

配置

指定服务器

请确保在配置文件中 指定了 Zabbix server- zabbix_agentd.conf

```
ServerActive=10.0.0.1
```

如果你没有在 zabbix_agentd.conf 中特别定义 *Hostname*, 则服务器将使用 agent 的系统主机名命名主机。Linux 中的系统主机名可以通过运行’hostname’ 命令获取。

修改配置文件后需要重启 agent

Aactive agent 自动注册动作

当服务器从 agent 收到自动注册请求时，它会调用一个动作 n. 必须要为 agent 自动注册配置一个事件源为 “自动注册” 的动作。

> **Note:**
> 设置 Setting up 网络发现 不需要使 active agents 自动注册。

在 Zabbix 前端页面，点击配置 → 动作, 选择自动注册为事件源，然后单击创建动作:

- 在动作选项卡，定义动作名称
- 可选指定条件。如果要使用 “主机元数据” 条件，请参阅下一节。
- 在 “操作” 选项卡中，需要添加关联操作，如 “添加主机”，“添加到主机组”（例如，发现的主机），“链接到模板” 等。

<note tip> 如果自动注册主机只能支持主动监视（例如由于防火墙的原因，Zabbix 服务器不允许访问的主机），则可能需要创建一个特定的模板，如 *Template_Linux-active* 来做关联。:::

使用主机元数据

当 agent 程序向服务器发送自动注册请求时，会发送其主机名。在某些情况下（例如，Amazon 云端节点），Zabbix Server 单单通过主机名区分主机。这时可以选择主机元数据将其他信息从 agent 发送到服务器。

主机元数据在 agent配置文件 - zabbix_agentd.conf 中配置。在配置文件中指定主机元数据有两种方式：

```
HostMetadata
HostMetadataItem
```

请参阅上面链接中的选项描述。

<note:important> 每当 active agent 发送刷新主动检查请求到服务器时，都会进行自动注册尝试。请求的延迟在 agent 的RefreshActiveChecks参数中指定。第一个请求在 agent 重新启动后立即发送。:::

案例 1

使用主机元数据来区分 Linux 和 Windows 主机。

假设你希望主机由 Zabbix server 自动注册，你的网络上有 active Zabbix agents（请参阅上面的 “配置” 部分），你的网络上有 Windows 主机和 Linux 主机，你有 “Template OS Linux” 和 “Template OS Windows” 模板，Zabbix 页面可以使用。在主机注册时，你希望将 Linux / Windows 模板正确的应用在正在注册的主机。默认情况下，只有主机名在自动注册时会发送到服务器，但这还不够。为了确保将正确的模板应用于主机，你应该使用主机元数据。

前段配置

第一步是配置前端，创建 2 个动作，第一个动作：

- 名称：Linux 主机自动注册
- 条件：主机元数据, 如 *Linux*
- 动作：链接到模板：Template OS Linux

Note:
在这种情况下，您可以跳过 “添加主机” 的操作。链接到模板需要首先添加主机，服务器会自动执行 “添加主机” 的操作。

第二个动作：

- 名称：Windows 主机自动注册
- 条件：主机元数据，如 *Windows*
- 操作：链接到模板：Template OS Windows

Agent 配置

第二部进行 Agent 配置，添加下行至 agent 配置文件中：

```
HostMetadataItem=system.uname
```

This way you make sure host metadata will contain ”Linux” or ”Windows” depending on the host an agent is running on. An example of host metadata in this case:

```
Linux: Linux server3 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2 i686 GNU/Linux
Windows: Windows WIN-OPXGGSTYNHO 6.0.6001 Windows Server 2008 Service Pack 1 Intel IA-32
```

Do not forget to restart the agent after making any changes to the configuration file.

Example 2

Step 1

Using host metadata to allow some basic protection against unwanted hosts registering.

Frontend configuration

Create an action in the frontend, using some hard-to-guess secret code to disallow unwanted hosts:

- Name: Auto registration action Linux
- Conditions:

```
    * Type of calculation: AND
    * Condition (A): Host metadata like //Linux//
    * Condition (B): Host metadata like //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
    * Link to templates: Template OS Linux
```

Please note that this method alone does not provide strong protection because data is transmitted in plain text. Configuration cache reload is required for changes to have an immediate effect.

Agent configuration

Add the next line to the agent configuration file:

```
HostMetadata=Linux      21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

where ”Linux” is a platform, and the rest of the string is the hard-to-guess secret text.

Do not forget to restart the agent after making any changes to the configuration file.

Step 2

It is possible to add additional monitoring for an already registered host.

Frontend configuration

Update the action in the frontend:

- Name: Auto registration action Linux
- Conditions:

```
    * Type of calculation: AND
    * Condition (A): Host metadata like Linux
    * Condition (B): Host metadata like 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
    * Link to templates: Template OS Linux
```

```
* Link to templates: Template DB MySQL
```

Agent configuration

Update the next line in the agent configuration file:

```
HostMetadata=MySQL on Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

Do not forget to restart the agent after making any changes to the configuration file.

2 Active agent auto-registration Overview

It is possible to allow active Zabbix agent auto-registration, after which the server can start monitoring them. This way new hosts can be added for monitoring without configuring them manually on the server.

Auto registration can happen when a previously unknown active agent asks for checks.

The feature might be very handy for automatic monitoring of new Cloud nodes. As soon as you have a new node in the Cloud Zabbix will automatically start the collection of performance and availability data of the host.

Active agent auto-registration also supports the monitoring of added hosts with passive checks. When the active agent asks for checks, providing it has the 'ListenIP' or 'ListenPort' configuration parameters defined in the configuration file, these are sent along to the server. (If multiple IP addresses are specified, the first one is sent to the server.)

Server, when adding the new auto-registered host, uses the received IP address and port to configure the agent. If no IP address value is received, the one used for the incoming connection is used. If no port value is received, 10050 is used.

Auto-registration is rerun:

- if host metadata information changes
- for manually created hosts with metadata missing
- if a host is manually changed to be monitored by another Zabbix proxy
- if auto-registration for the same host comes from a new Zabbix proxy

Configuration

Specify server

Make sure you have the Zabbix server identified in the agent configuration file - zabbix_agentd.conf

```
ServerActive=10.0.0.1
```

Unless you specifically define a *Hostname* in zabbix_agentd.conf, the system hostname of agent location will be used by server for naming the host. The system hostname in Linux can be obtained by running the 'hostname' command.

Restart the agent after making any changes to the configuration file.

Action for active agent auto-registration

When server receives an auto-registration request from an agent it calls an action. An action of event source "Auto registration" must be configured for agent auto-registration.

Note:
Setting up network discovery is not required to have active agents auto-register.

In the Zabbix frontend, go to *Configuration* → *Actions*, select *Auto registration* as the event source and click on *Create action*:

- In the Action tab, give your action a name
- Optionally specify conditions. If you are going to use the "Host metadata" condition, see the next section.
- In the Operations tab, add relevant operations, such as - 'Add host', 'Add to host groups' (for example, *Discovered hosts*), 'Link to templates', etc.

Note:
If the hosts that will be auto-registering are likely to be supported for active monitoring only (such as hosts that are firewalled from your Zabbix server) then you might want to create a specific template like *Template_Linux-active* to link to.

Using host metadata

When agent is sending an auto-registration request to the server it sends its hostname. In some cases (for example, Amazon cloud nodes) a hostname is not enough for Zabbix server to differentiate discovered hosts. Host metadata can be optionally used to send other information from an agent to the server.

Host metadata is configured in the agent configuration file - zabbix_agentd.conf. There are 2 ways of specifying host metadata in the configuration file:

```
HostMetadata
HostMetadataItem
```

See the description of the options in the link above.

<note:important>An auto-registration attempt happens every time an active agent sends a request to refresh active checks to the server. The delay between requests is specified in the RefreshActiveChecks parameter of the agent. The first request is sent immediately after the agent is restarted. :::

Example 1

Using host metadata to distinguish between Linux and Windows hosts.

Say you would like the hosts to be auto-registered by the Zabbix server. You have active Zabbix agents (see "Configuration" section above) on your network. There are Windows hosts and Linux hosts on your network and you have "Template OS Linux" and "Template OS Windows" templates available in your Zabbix frontend. So at host registration you would like the appropriate Linux/Windows template to be applied to the host being registered. By default only the hostname is sent to the server at auto-registration, which might not be enough. In order to make sure the proper template is applied to the host you should use host metadata.

Frontend configuration

The first thing to do is to configure the frontend. Create 2 actions. The first action:

- Name: Linux host autoregistration
- Conditions: Host metadata like *Linux*
- Operations: Link to templates: Template OS Linux

Note:
You can skip an "Add host" operation in this case. Linking to a template requires adding a host first so the server will do that automatically.

The second action:

- Name: Windows host autoregistration
- Conditions: Host metadata like *Windows*
- Operations: Link to templates: Template OS Windows

Agent configuration

Now you need to configure the agents. Add the next line to the agent configuration files:

```
HostMetadataItem=system.uname
```

This way you make sure host metadata will contain "Linux" or "Windows" depending on the host an agent is running on. An example of host metadata in this case:

```
Linux: Linux server3 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2 i686 GNU/Linux
Windows: Windows WIN-0PXGGSTYNHO 6.0.6001 Windows Server 2008 Service Pack 1 Intel IA-32
```

Do not forget to restart the agent after making any changes to the configuration file.

Example 2

Step 1

Using host metadata to allow some basic protection against unwanted hosts registering.

Frontend configuration

Create an action in the frontend, using some hard-to-guess secret code to disallow unwanted hosts:

- Name: Auto registration action Linux
- Conditions:

```
    * Type of calculation: AND
    * Condition (A): Host metadata like //Linux//
    * Condition (B): Host metadata like //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
```

```
    * Link to templates: Template OS Linux
```

Please note that this method alone does not provide strong protection because data is transmitted in plain text. Configuration cache reload is required for changes to have an immediate effect.

Agent configuration

Add the next line to the agent configuration file:

```
HostMetadata=Linux      21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

where ”Linux” is a platform, and the rest of the string is the hard-to-guess secret text.

Do not forget to restart the agent after making any changes to the configuration file.

Step 2

It is possible to add additional monitoring for an already registered host.

Frontend configuration

Update the action in the frontend:

- Name: Auto registration action Linux
- Conditions:

```
    * Type of calculation: AND
    * Condition (A): Host metadata like Linux
    * Condition (B): Host metadata like 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
    * Link to templates: Template OS Linux
    * Link to templates: Template DB MySQL
```

Agent configuration

Update the next line in the agent configuration file:

```
HostMetadata=MySQL on Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

Do not forget to restart the agent after making any changes to the configuration file.

3 Low-level discovery

Overview

Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer. For instance, Zabbix can automatically start monitoring file systems or network interfaces on your machine, without the need to create items for each file system or network interface manually. Additionally it is possible to configure Zabbix to remove unneeded entities automatically based on actual results of periodically performed discovery.

A user can define their own types of discovery, provided they follow a particular JSON protocol.

The general architecture of the discovery process is as follows.

First, a user creates a discovery rule in ”Configuration” → ”Templates” → ”Discovery” column. A discovery rule consists of (1) an item that discovers the necessary entities (for instance, file systems or network interfaces) and (2) prototypes of items, triggers, and graphs that should be created based on the value of that item.

An item that discovers the necessary entities is like a regular item seen elsewhere: the server asks a Zabbix agent (or whatever the type of the item is set to) for a value of that item, the agent responds with a textual value. The difference is that the value the agent responds with should contain a list of discovered entities in a specific JSON format. While the details of this format are only important for implementers of custom discovery checks, it is necessary to know that the returned value contains a list of macro → value pairs. For instance, item ”net.if.discovery” might return two pairs: ”{#IFNAME}” → ”lo” and ”{#IFNAME}” → ”eth0”.

These macros are used in names, keys and other prototype fields where they are then substituted with the received values for creating real items, triggers, graphs or even hosts for each discovered entity. See the full list of options for using LLD macros.

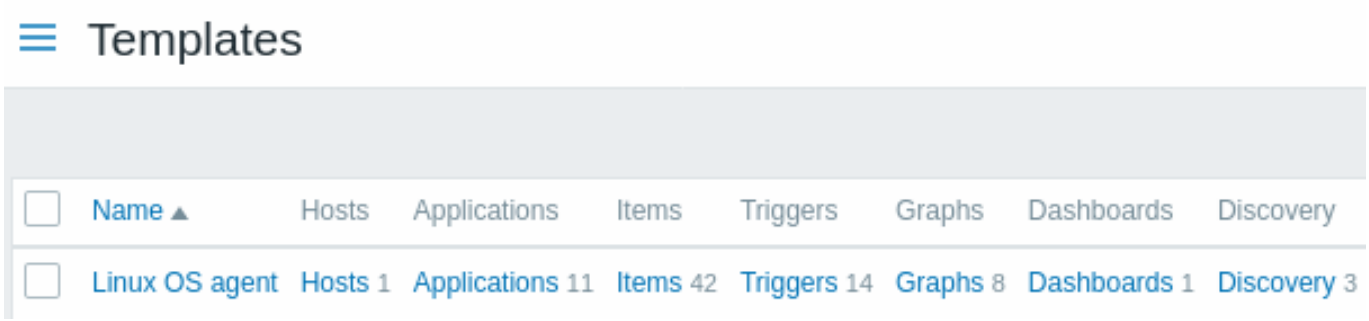
When the server receives a value for a discovery item, it looks at the macro → value pairs and for each pair generates real items, triggers, and graphs, based on their prototypes. In the example with ”net.if.discovery” above, the server would generate one set of items, triggers, and graphs for the loopback interface ”lo”, and another set for interface ”eth0”.

Configuring low-level discovery

We will illustrate low-level discovery based on an example of file system discovery.

To configure the discovery, do the following:

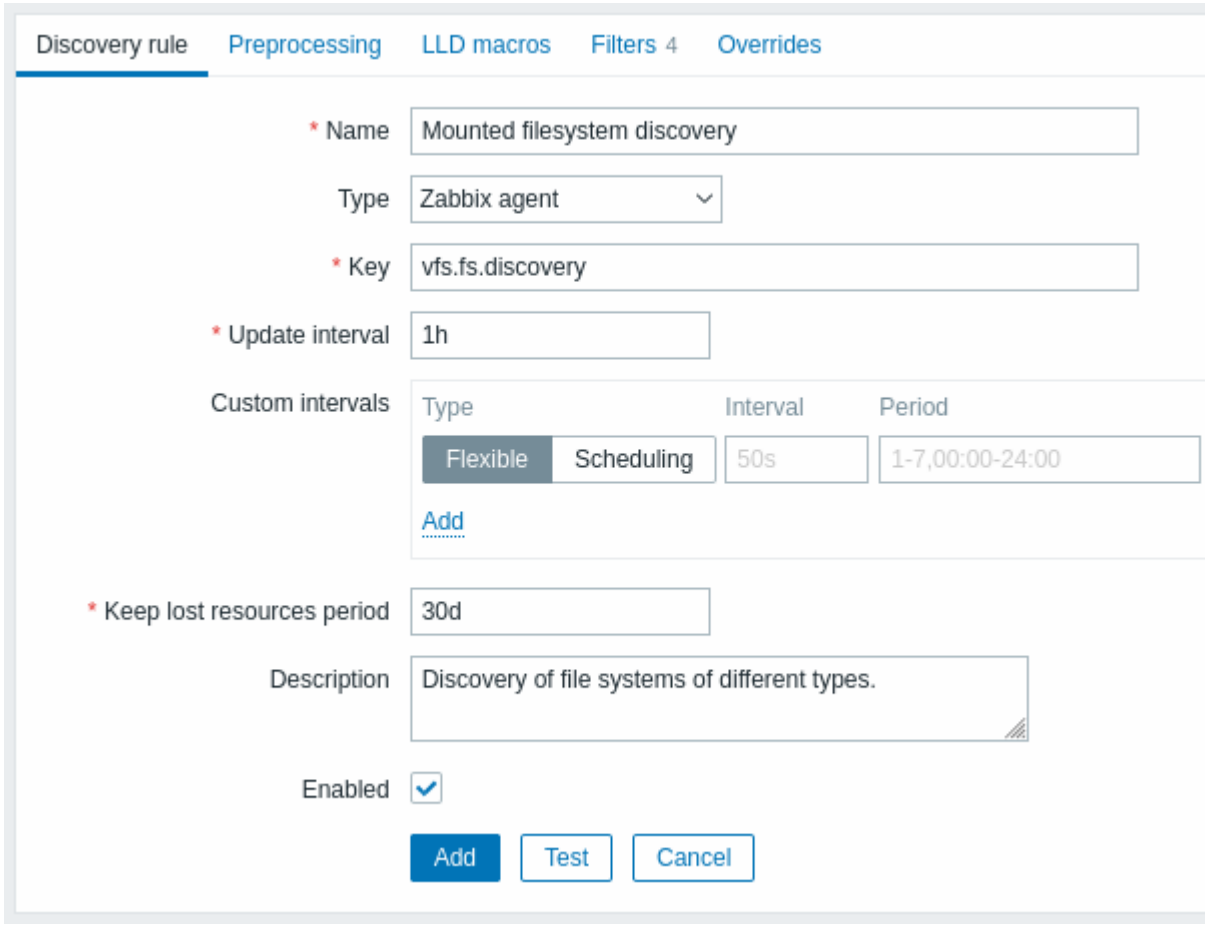
- Go to: *Configuration* → *Templates*
- Click on *Discovery* in the row of an appropriate template



- Click on *Create discovery rule* in the upper right corner of the screen
- Fill in the discovery rule form with the required details

Discovery rule

The **Discovery rule** tab contains general discovery rule attributes:



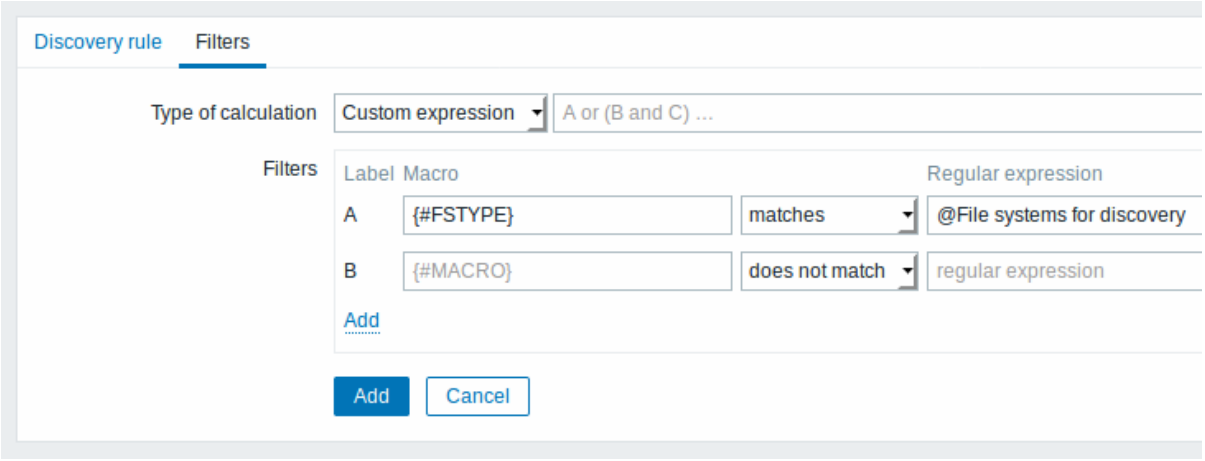
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Name of discovery rule.
Type	The type of check to perform discovery; should be *Zabbix agent* or *Zabbix agent (active)* for file system discovery.
Key	An item with "vfs.fs.discovery" key is built into Zabbix agent since version 2.0 on many platforms (see supported item key list for details), and will return a JSON with the list of file systems present on the computer and their types.

Parameter	Description
Update interval	This field specifies how often Zabbix performs discovery. In the beginning, when you are just setting up file system discovery, you might wish to set it to a small interval, but once you know it works you can set it to 30 minutes or more, because file systems usually do not change very often. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note*: 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. *Note* that for an existing discovery rule the discovery can be performed immediately by pushing the *Check now* button.
Custom intervals	You can create custom rules for checking the item: **Flexible** - create an exception to the *Update interval* (interval with different frequency) **Scheduling** - create a custom polling schedule. For detailed information see Custom intervals. Scheduling is supported since Zabix 3.0.0.
Keep lost resources period	This field allows you to specify the duration for how long the discovered entity will be retained (won't be deleted) once its discovery status becomes "Not discovered anymore" (min 1 hour, max 25 years). Time suffixes are supported, e.g. 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note:* If set to "0", entities will be deleted immediately. Using "0" is not recommended, since just wrongly editing the filter may end up in the entity being deleted with all the historical data.
Description	Enter a description.
Enabled	If checked, the rule will be processed.

Discovery rule filter

The **Filters** tab contains discovery rule filter definitions:



Parameter	Description
Type of calculation	The following options for calculating filters are available: **And** - all filters must be passed; **Or** - enough if one filter is passed; **And/Or** - uses *And* with different macro names and *Or* with the same macro name; **Custom expression** - offers the possibility to define a custom calculation of filters. The formula must include all filters in the list. Limited to 255 symbols.

Parameter	Description
Filters	A filter can be used to generate real items, triggers, and graphs only for certain file systems. It expects a Perl Compatible Regular Expression (PCRE). For instance, if you are only interested in C:, D:, and E: file systems, you could put {#FSNAME} into "Macro" and "^C|^D|^E" regular expression into "Regular expression" text fields. Filtering is also possible by file system types using {#FSTYPE} macro (e.g. "^ext|^reiserfs") and by drive types (supported only by Windows agent) using {#FSDRIVETYPE} macro (e.g., "fixed"). You can enter a regular expression or reference a global regular expression in "Regular expression" field. In order to test a regular expression you can use "grep -E", for example: `for f in ext2 nfs reiserfs smbfs; do echo $f | grep -E '^ext` macro on Windows is supported since Zabbix **3.0.0**. Defining several filters is supported since Zabbix **2.4.0**. Note that if some macro from the filter is missing in the response, the found entity will be ignored. Filter drop-down offers two values to specify whether a macro matches a regular expression or does not match.

Attention:
Zabbix database in MySQL must be created as case-sensitive if file system names that differ only by case are to be discovered correctly.

Attention:
The mistake or typo in regex used in LLD rule may cause deleting thousands of configuration elements, historical values and events for many hosts. For example, incorrect "File systems for discovery" regular expression may cause deleting thousands of items, triggers, historical values and events.

Note:
Discovery rule history is not preserved.

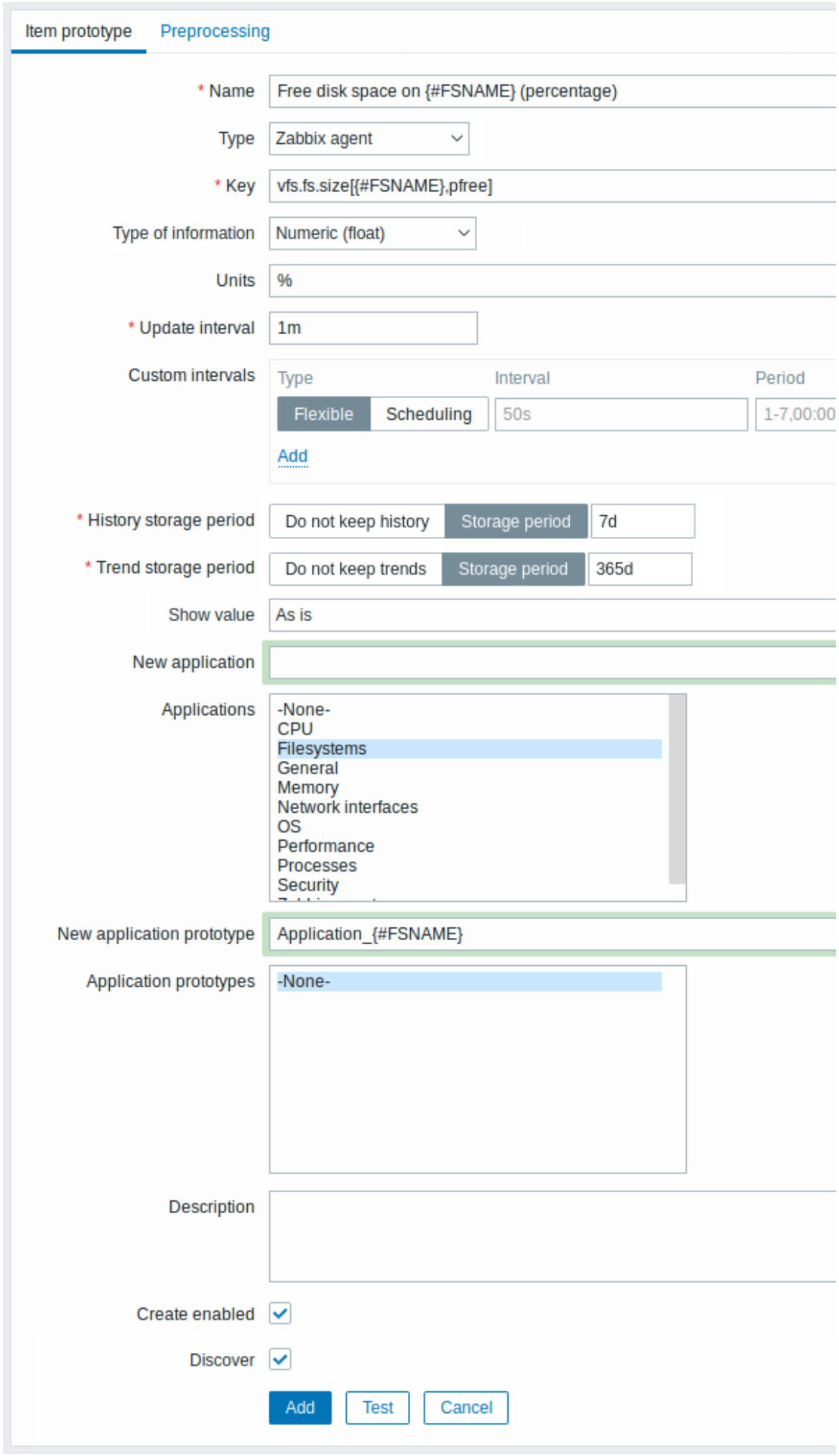
Form buttons

Buttons at the bottom of the form allow to perform several operations.

Add	Add a discovery rule. This button is only available for new discovery rules.
Update	Update the properties of a discovery rule. This button is only available for existing discovery rules.
Clone	Create another discovery rule based on the properties of the current discovery rule.
Check now	Perform discovery based on the discovery rule immediately. The discovery rule must already exist. See more details. *Note* that when performing discovery immediately, configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration.
Delete	Delete the discovery rule.
Cancel	Cancel the editing of discovery rule properties.

Item prototypes

Once a rule is created, go to the items for that rule and press ”Create prototype” to create an item prototype. Note how macro {#FSNAME} is used where a file system name is required. When the discovery rule is processed, this macro will be substituted with the discovered file system.


Item prototype
Preprocessing
* Name
Free disk space on {#FSNAME} (percentage)
Type
Zabbix agent
* Key
vfs.fs.size[{#FSNAME},pfree]
Type of information
Numeric (float)
Units
%
* Update interval
1m
Custom intervals
Type
Interval
Period
Flexible
Scheduling
50s
1-7,00:00
Add
* History storage period
Do not keep history
Storage period
7d
* Trend storage period
Do not keep trends
Storage period
365d
Show value
As is
New application
Applications
-None-
CPU
Filesystems
General
Memory
Network interfaces
OS
Performance
Processes
Security
New application prototype
Application_{#FSNAME}
Application prototypes
-None-
Description
Create enabled
Discover
Add
Test
Cancel

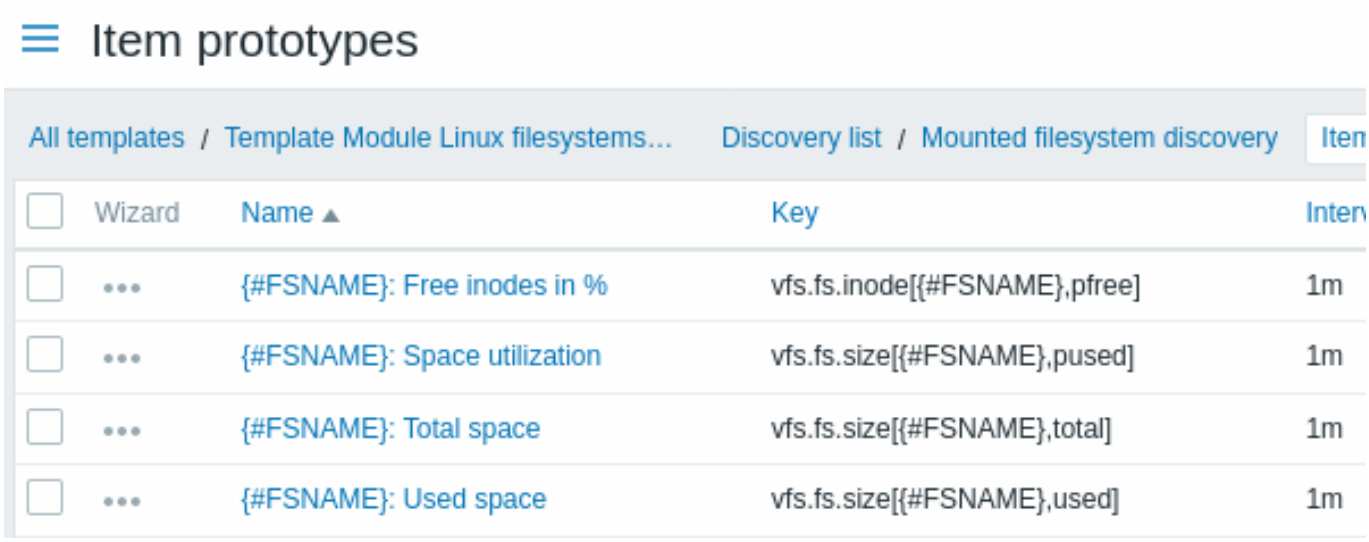
Low-level discovery macros and user macros may be used in item prototype configuration and item value preprocessing parameters.

> **Note:**
> Context-specific escaping of low-level discovery macros is performed for safe use in regular expression and XPath preprocessing parameters.

Attributes that are specific for item prototypes:

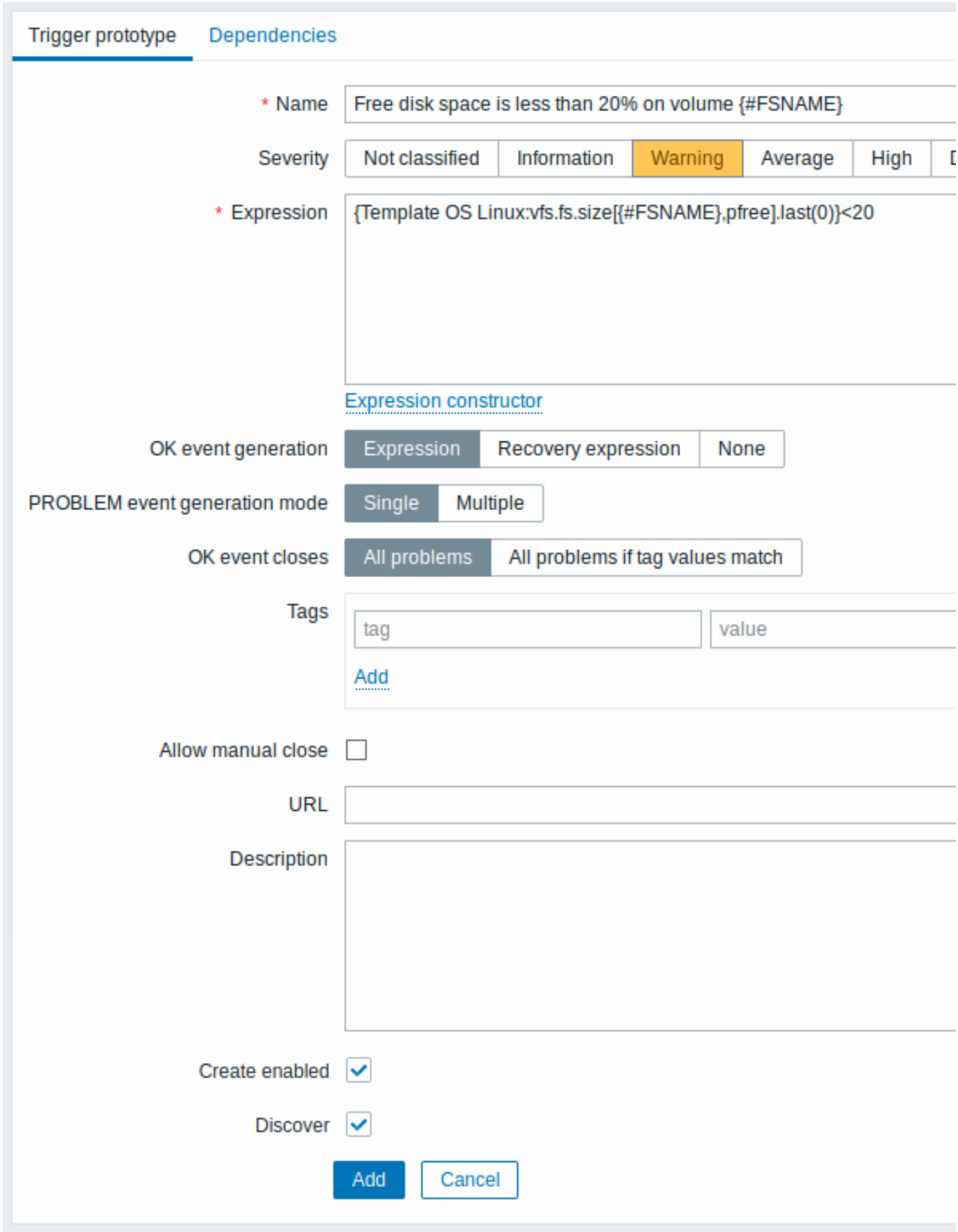
Parameter	Description
New application prototype	You may define a new application prototype. In application prototypes you can use low-level discovery macros that, after discovery, will be substituted with real values to create applications that are specific for the discovered entity. See also application discovery notes for more specific information.
Application prototypes	Select from the existing application prototypes.
Create enabled	If checked the item will be added in an enabled state. If unchecked, the item will be added to a discovered entity, but in a disabled state.

We can create several item prototypes for each file system metric we are interested in:



Trigger prototypes

We create trigger prototypes in a similar way as item prototypes:

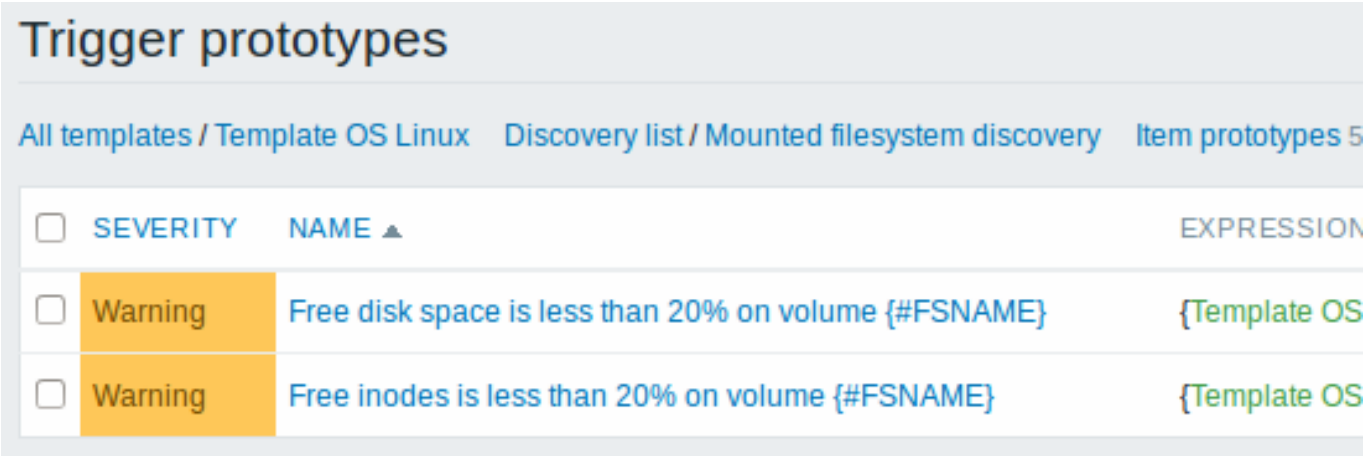


Attributes that are specific for trigger prototypes:

Parameter	Description
Create enabled	If checked the trigger will be added in an enabled state. If unchecked, the trigger will be added to a discovered entity, but in a disabled state.

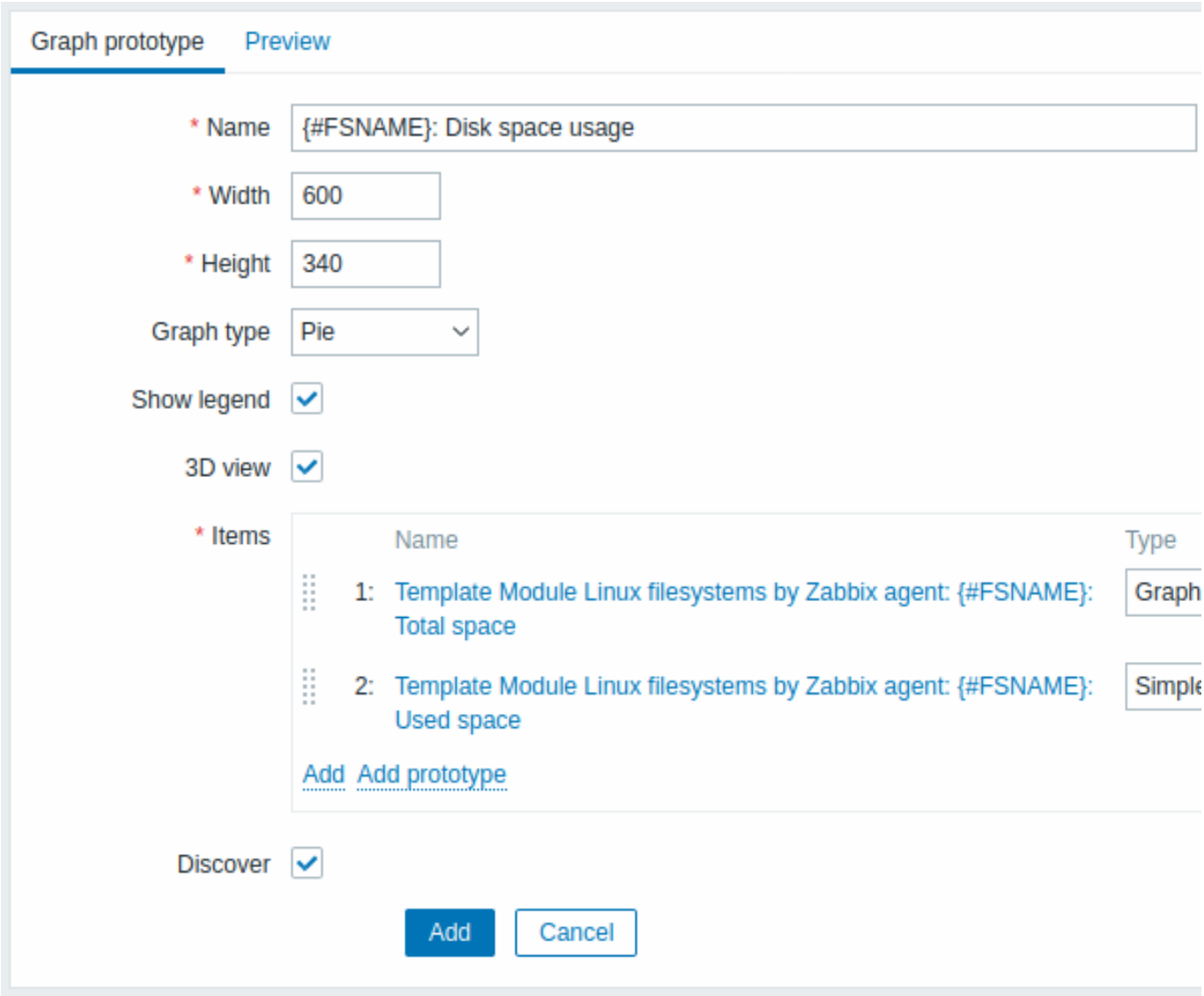
When real triggers are created from the prototypes, there may be a need to be flexible as to what constant ('20' in our example) is used for comparison in the expression. See how user macros with context can be useful to accomplish such flexibility.

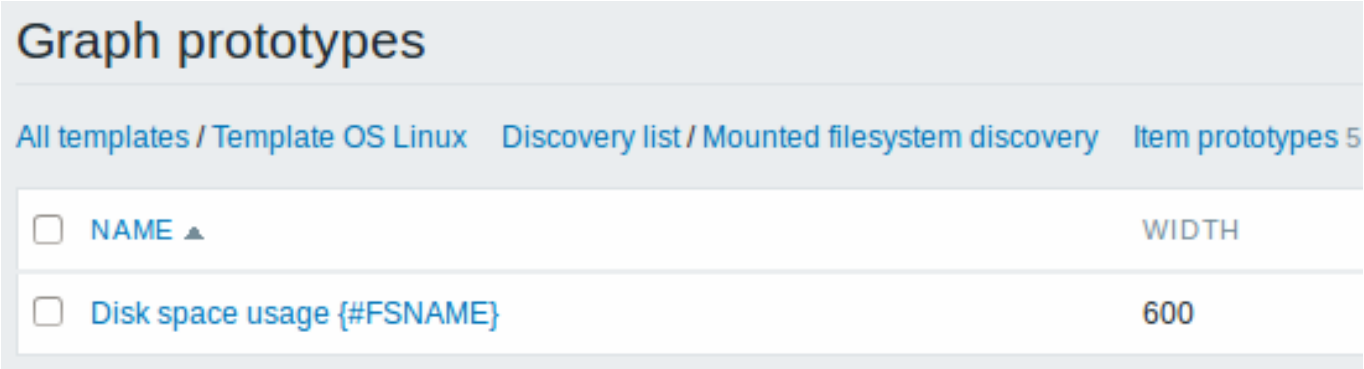
You can define dependencies between trigger prototypes as well (supported since Zabbix 3.0). To do that, go to the *Dependencies* tab. A trigger prototype may depend on another trigger prototype from the same low-level discovery (LLD) rule or on a regular trigger. A trigger prototype may not depend on a trigger prototype from a different LLD rule or on a trigger created from trigger prototype. Host trigger prototype cannot depend on a trigger from a template.



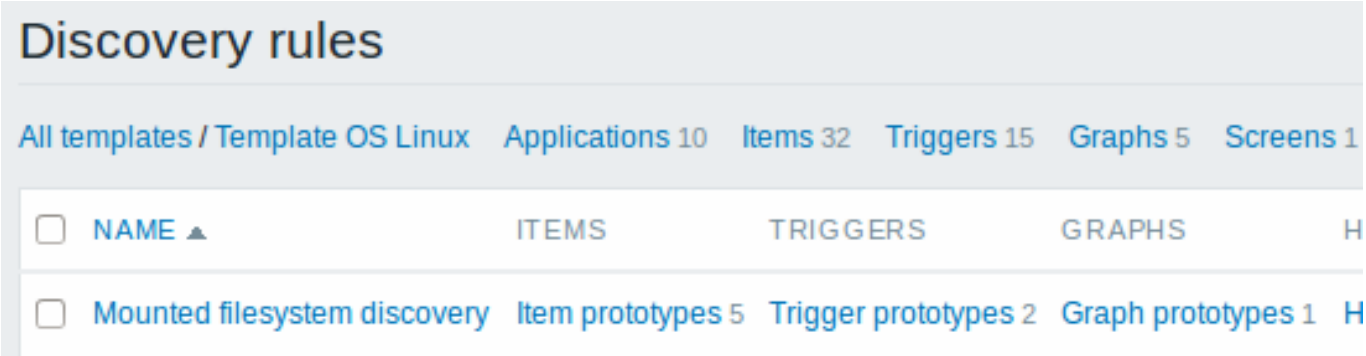
Graph prototypes

We can create graph prototypes, too:





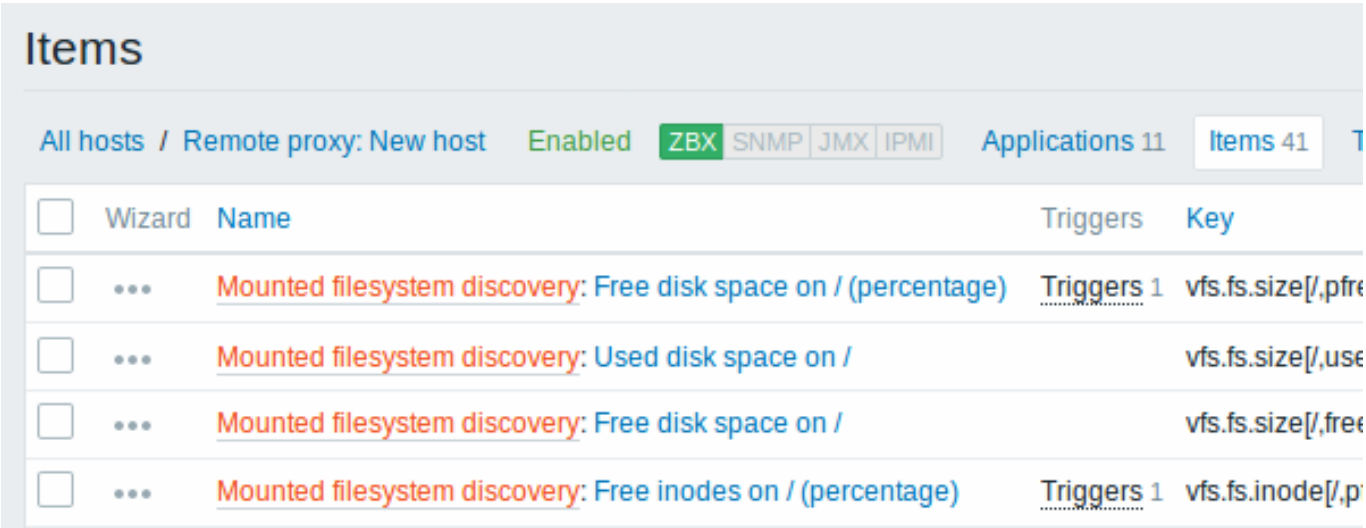
Finally, we have created a discovery rule that looks like shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.



Note: For configuring host prototypes, see the section about host prototype configuration in virtual machine monitoring.

Discovered entities

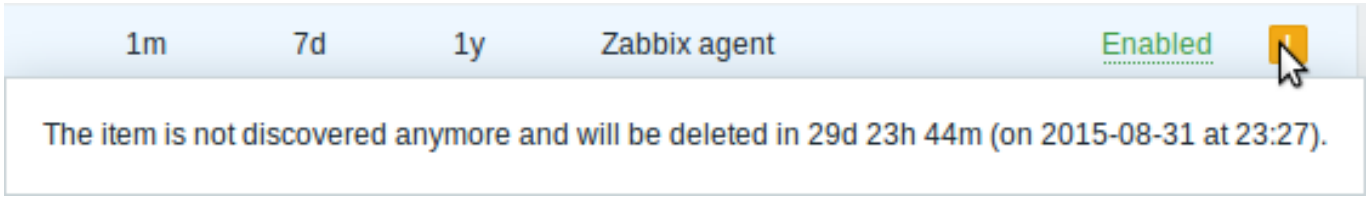
The screenshots below illustrate how discovered items, triggers, and graphs look like in the host's configuration. Discovered entities are prefixed with an orange link to a discovery rule they come from.



Note that discovered entities will not be created in case there are already existing entities with the same uniqueness criteria, for example, an item with the same key or graph with the same name.

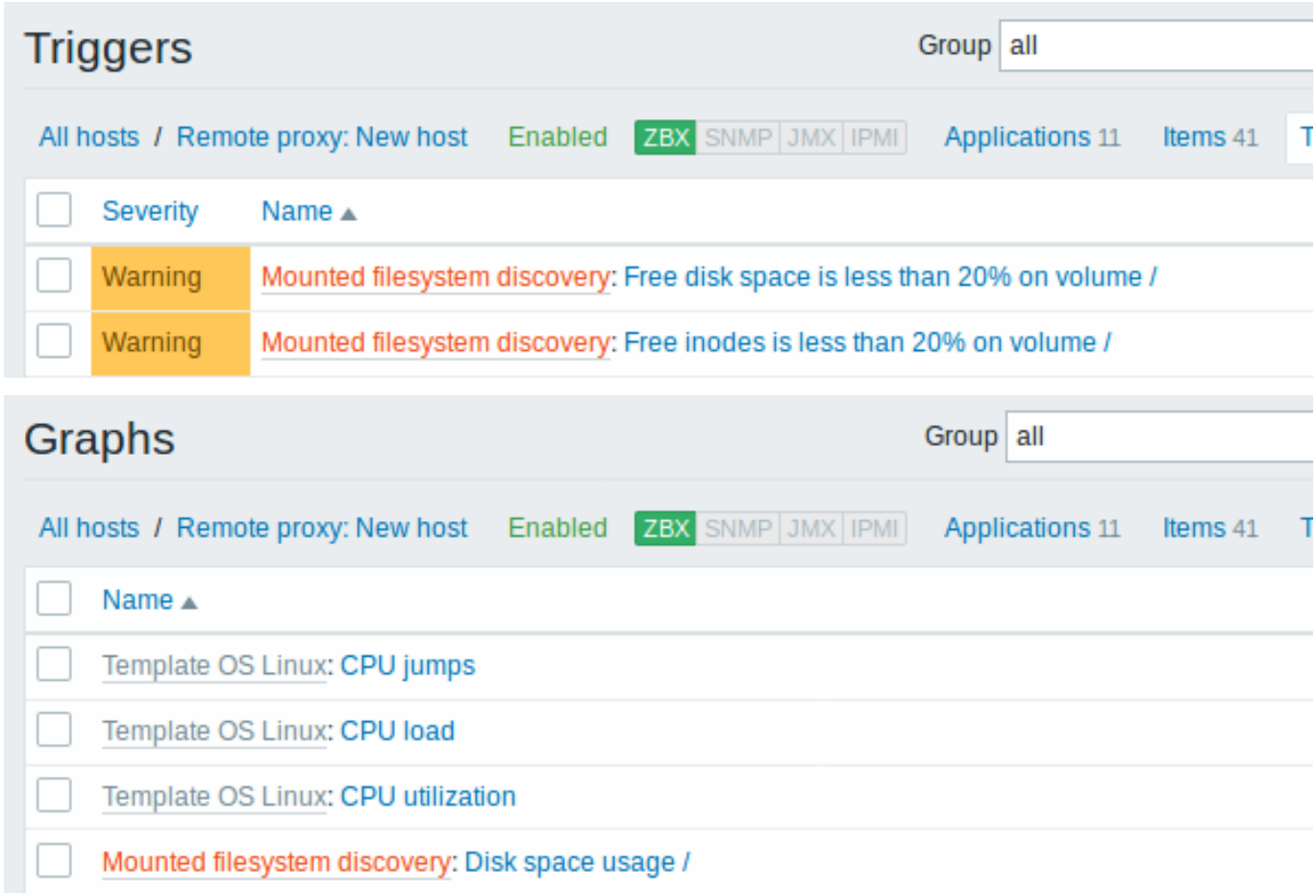
Items (similarly, triggers and graphs) created by a low-level discovery rule will be deleted automatically if a discovered entity (file system, interface, etc) stops being discovered (or does not pass the filter anymore). In this case the items, triggers and graphs will be deleted after the days defined in the *Keep lost resources period* field pass.

When discovered entities become 'Not discovered anymore', a lifetime indicator is displayed in the item list. Move your mouse pointer over it and a message will be displayed indicating how many days are left until the item is deleted.



If entities were marked for deletion, but were not deleted at the expected time (disabled discovery rule or item host), they will be deleted the next time the discovery rule is processed.

Entities containing other entities, which are marked for deletion, will not update if changed on the discovery rule level. For example, LLD-based triggers will not update if they contain items that are marked for deletion.



Other types of discovery

More detail and how-tos on other types of out-of-the-box discovery is available in the following sections:

- discovery of network interfaces;
- discovery of CPUs and CPU cores;
- discovery of SNMP OIDs;
- discovery of JMX objects;
- discovery using ODBC SQL queries;
- discovery of Windows services;
- discovery of host interfaces in Zabbix.

For more detail on the JSON format for discovery items and an example of how to implement your own file system discoverer as a Perl script, see creating custom LLD rules.

Data limits for return values

There is no limit for low-level discovery rule JSON data if it is received directly by Zabbix server, because return values are processed without being stored in a database. There's also no limit for custom low-level discovery rules, however, if it is intended to acquire custom LLD data using a user parameter, then user parameter return value limit applies (512 KB).

If data has to go through Zabbix proxy it has to store this data in database so database limits apply, for example, 2048 bytes on a Zabbix proxy run with IBM DB2 database.

Multiple LLD rules for same item

Since Zabbix agent version 3.2 it is possible to define several low-level discovery rules with the same discovery item.

To do that you need to define the Alias agent parameter, allowing to use altered discovery item keys in different discovery rules, for example `vfs.fs.discovery[foo]`, `vfs.fs.discovery[bar]`, etc.

Creating custom LLD rules

It is also possible to create a completely custom LLD rule, discovering any type of entities - for example, databases on a database server.

To do so, a custom item should be created that returns JSON, specifying found objects and optionally - some properties of them. The amount of macros per entity is not limited - while the built-in discovery rules return either one or two macros (for example, two for filesystem discovery), it is possible to return more.

The required JSON format is best illustrated with an example. Suppose we are running an old Zabbix 1.8 agent (one that does not support "vfs.fs.discovery"), but we still need to discover file systems. Here is a simple Perl script for Linux that discovers mounted file systems and outputs JSON, which includes both file system name and type. One way to use it would be as a UserParameter with key "vfs.fs.discovery_perl":

```
###!/usr/bin/perl

$first = 1;

print "{\n";
print "\t\"data\":[\n\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"{#FSNAME}\":\"$fsname\",\n";
    print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
    print "\t}\n";
}

print "\n\t]\n";
print "}\n";
```

Attention:
Allowed symbols for LLD macro names are **0-9** , **A-Z** , _ , **.**

Lowercase letters are not supported in the names.

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
{
  "data":[

  { "{#FSNAME}":"/",                          "{#FSTYPE}":"rootfs"   },
  { "{#FSNAME}":"/sys",                       "{#FSTYPE}":"sysfs"    },
  { "{#FSNAME}":"/proc",                      "{#FSTYPE}":"proc"     },
  { "{#FSNAME}":"/dev",                       "{#FSTYPE}":"devtmpfs" },
  { "{#FSNAME}":"/dev/pts",                   "{#FSTYPE}":"devpts"   },
  { "{#FSNAME}":"/lib/init/rw",               "{#FSTYPE}":"tmpfs"    },
  { "{#FSNAME}":"/dev/shm",                   "{#FSTYPE}":"tmpfs"    },
  { "{#FSNAME}":"/home",                      "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/tmp",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/usr",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/var",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/sys/fs/fuse/connections",   "{#FSTYPE}":"fusectl"  }

  ]
```

}

Then, in the discovery rule's "Filter" field, we could specify "{#FSTYPE}" as a macro and "rootfs|ext3" as a regular expression.

> **Note:**
> You don't have to use macro names FSNAME/FSTYPE with custom LLD rules, you are free to use whatever names you like.

Note that, if using a user parameter, the return value is limited to 512 KB. For more details, see data limits for LLD return values.

Using LLD macros in user macro contexts

User macros with context can be used to accomplish more flexible thresholds in trigger expressions. Different thresholds may be defined on user macro level and then used in trigger constants depending on the discovered context. Discovered context appears when the low-level discovery macros used in the macros are resolved to real values.

To illustrate we can use data from the example above and assume that the following file systems will be discovered: `/`, `/home`, `/tmp`, `/usr`, `/var`.

We may define a free-disk-space trigger prototype for a host, where the threshold is expressed by a user macro with context:

```
{host:vfs.fs.size[{#FSNAME},pfree].last()}<{$LOW_SPACE_LIMIT:"{#FSNAME}"}
```

Then add user macros:

- `{$LOW_SPACE_LIMIT}` **10**
- `{$LOW_SPACE_LIMIT:/home}` **20**
- `{$LOW_SPACE_LIMIT:/tmp}` **50**

Now, once the file systems are discovered, events will be generated if `/`, `/usr` and `/var` filesystems have less than **10**% of free disk space, the `/home` filesystem - less than **20**% of free disk space or the `/tmp` filesystem - less than **50**% of free disk space.

Notes on low-level discovery

Application discovery

Application prototypes support LLD macros.

One application prototype can be used by several item prototypes of the same discovery rule.

If created application prototype is not used by any item prototype it gets removed from 'Application prototypes' list automatically.

Like other discovered entities applications follow the lifetime defined in discovery rule ('keep lost resources period' setting) - they are removed after not being discovered for the specified number of days.

If an application is not discovered anymore all discovered items are automatically removed from it, even if the application itself is not yet removed because of the 'lost resources period' setting.

Application prototypes defined by one discovery rule can't discover the same application. In this situation only the first prototype discovery will succeed, the rest will report appropriate LLD error. Only application prototypes defined in different discovery rules can result in discovering the same application.

16. Distributed monitoring

Overview Zabbix provides an effective and reliable way of monitoring a distributed IT infrastructure using Zabbix proxies.

Proxies can be used to collect data locally on behalf of a centralized Zabbix server and then report the data to the server.

Proxy features

When making a choice of using/not using a proxy, several considerations must be taken into account.

	Proxy
Lightweight	**Yes**
GUI	No
Works independently	**Yes**
Easy maintenance	**Yes**
Automatic DB creation[1]	**Yes**

	Proxy
Local administration	No
Ready for embedded hardware	**Yes**
One way TCP connections	**Yes**
Centralised configuration	**Yes**
Generates notifications	No

Note:
[1] Automatic DB creation feature only works with SQLite. Other databases require a manual setup.

1 Proxies

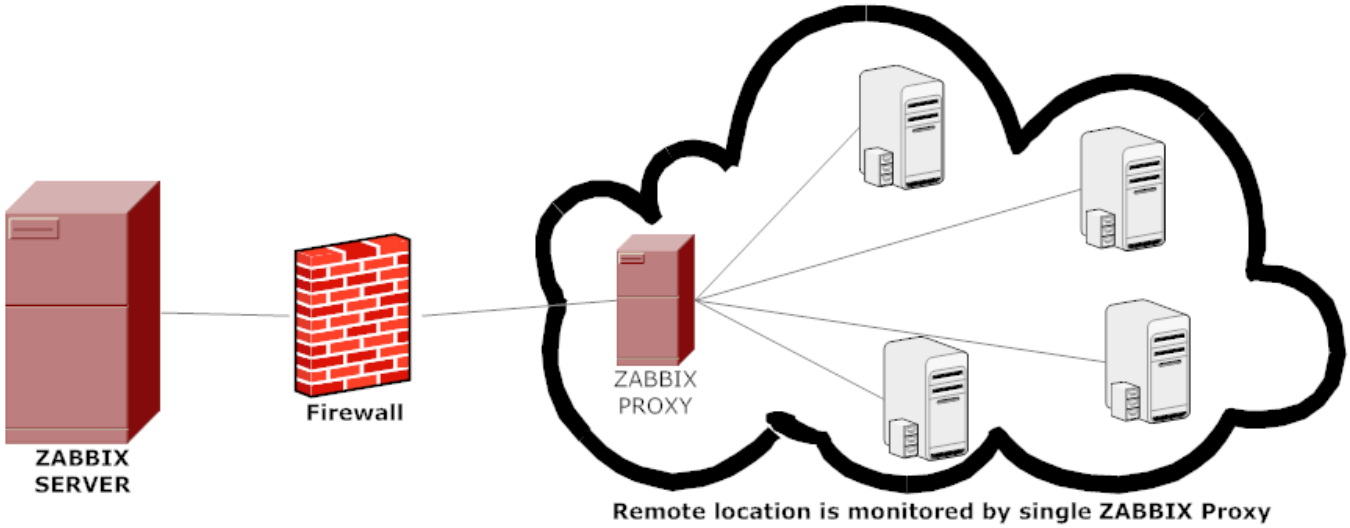
Overview

A Zabbix proxy can collect performance and availability data on behalf of the Zabbix server. This way, a proxy can take on itself some of the load of collecting data and offload the Zabbix server.

Also, using a proxy is the easiest way of implementing centralized and distributed monitoring, when all agents and proxies report to one Zabbix server and all data is collected centrally.

A Zabbix proxy can be used to:

- Monitor remote locations
- Monitor locations having unreliable communications
- Offload the Zabbix server when monitoring thousands of devices
- Simplify the maintenance of distributed monitoring



The proxy requires only one TCP connection to the Zabbix server. This way it is easier to get around a firewall as you only need to configure one firewall rule.

Attention:
Zabbix proxy must use a separate database. Pointing it to the Zabbix server database will break the configuration.

All data collected by the proxy is stored locally before transmitting it over to the server. This way no data is lost due to any temporary communication problems with the server. The *ProxyLocalBuffer* and *ProxyOfflineBuffer* parameters in the proxy configuration file control for how long the data are kept locally.

Attention:
It may happen that a proxy, which receives the latest configuration changes directly from Zabbix server database, has a more up-to-date configuration than Zabbix server whose configuration may not be updated as fast due to the value of CacheUpdateFrequency. As a result, proxy may start gathering data and send them to Zabbix server that ignores these data.

Zabbix proxy is a data collector. It does not calculate triggers, process events or send alerts. For an overview of what proxy functionality is, review the following table:

Function		Supported by proxy
Items		
	Zabbix agent checks	**Yes**
	Zabbix agent checks (active)	**Yes** [1]
	Simple checks	**Yes**
	Trapper items	**Yes**
	SNMP checks	**Yes**
	SNMP traps	**Yes**
	IPMI checks	**Yes**
	JMX checks	**Yes**
	Log file monitoring	**Yes**
	Internal checks	**Yes**
	SSH checks	**Yes**
	Telnet checks	**Yes**
	External checks	**Yes**
	Dependent items	**Yes** [2]
Built-in web monitoring		**Yes**
Network discovery		**Yes**
Low-level discovery		**Yes**
Remote commands		**Yes**
Calculating triggers		*No*
Processing events		*No*
Event correlation		*No*
Sending alerts		*No*
Item value preprocessing		*No*

Note:
[1] To make sure that an agent asks the proxy (and not the server) for active checks, the proxy must be listed in the **ServerActive** parameter in the agent configuration file.

Note:
[2] Item value preprocessing by Zabbix server is required to extract the required value from the master item data.

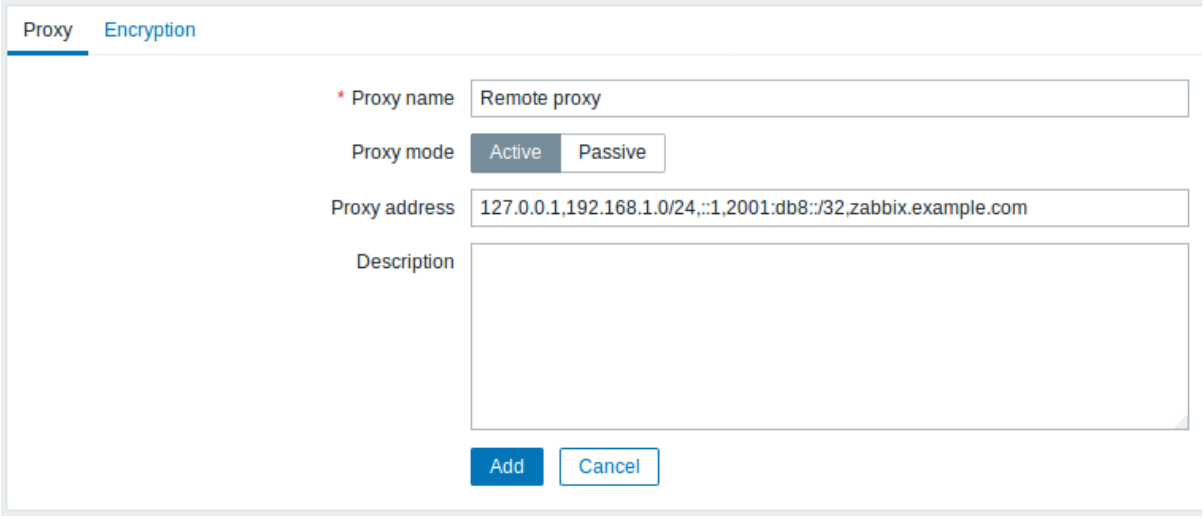
Configuration

Once you have installed and configured a proxy, it is time to configure it in the Zabbix frontend.

Adding proxies

To configure a proxy in Zabbix frontend:

- Go to: *Administration* → *Proxies*
- Click on *Create proxy*



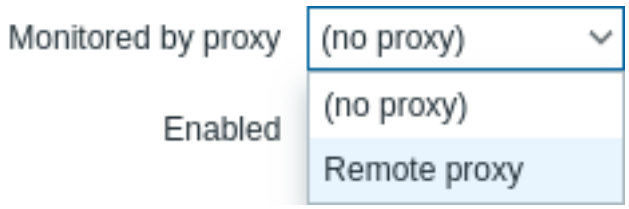
Parameter		Description
Proxy name		Enter the proxy name. It must be the same name as in the *Hostname* parameter in the proxy configuration file.
Proxy mode		Select the proxy mode. **Active** - the proxy will connect to the Zabbix server and request configuration data **Passive** - Zabbix server connects to the proxy *Note* that without encrypted communications (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data if authentication does not take place or proxy addresses are not limited in the *Proxy address* field.
Proxy address		If specified then active proxy requests are only accepted from this list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of active Zabbix proxy. This field is only available if an active proxy is selected in the *Proxy mode* field. Macros are not supported. This option is supported since Zabbix 4.0.0.
Interface		Enter interface details for the passive proxy. This field is only available if a passive proxy is selected in the *Proxy mode* field.
	IP address	IP address of the passive proxy (optional).
	DNS name	DNS name of the passive proxy (optional).
	Connect to	Clicking the respective button will tell Zabbix server what to use to retrieve data from proxy: **IP** - Connect to the proxy IP address (recommended) **DNS** - Connect to the proxy DNS name
	Port	TCP/UDP port number of the passive proxy (10051 by default).
Description		Enter the proxy description.

The **Encryption** tab allows you to require encrypted connections with the proxy.

Parameter	Description
Connections to proxy	How the server connects to the passive proxy: no encryption (default), using PSK (pre-shared key) or certificate.
Connections from proxy	Select what type of connections are allowed from the active proxy. Several connection types can be selected at the same time (useful for testing and switching to other connection type). Default is "No encryption".
Issuer	Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the *Issuer* field can be used to further restrict allowed CA. This field is optional, intended to use if your Zabbix installation uses certificates from multiple CAs.
Subject	Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the *Subject* field can be used to allow only one value of *Subject* string. If this field is empty then any valid certificate signed by the configured CA is accepted.
PSK identity	Pre-shared key identity string.
PSK	Pre-shared key (hex-string). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

Host configuration

You can specify that an individual host should be monitored by a proxy in the host configuration form, using the *Monitored by proxy* field.



Host mass update is another way of specifying that hosts should be monitored by a proxy.

17. Encryption

Overview Zabbix supports encrypted communications between Zabbix server, Zabbix proxy, Zabbix agent, zabbix_sender and zabbix_get utilities using Transport Layer Security (TLS) protocol v.1.2. Encryption is supported starting with Zabbix 3.0. Certificate-based and pre-shared key-based encryption is supported.

Encryption is optional and configurable for individual components (e.g. some proxies and agents can be configured to use certificate-based encryption with the server, while others can use pre-shared key-based encryption, and yet others continue with unencrypted communications as before).

Server (proxy) can use different encryption configurations for different hosts.

Zabbix daemon programs use one listening port for encrypted and unencrypted incoming connections. Adding an encryption does not require opening new ports on firewalls.

Limitations

- Private keys are stored in plain text in files readable by Zabbix components during startup.
- Pre-shared keys are entered in Zabbix frontend and stored in Zabbix database in plain text.
- Built-in encryption does not protect communications:

```
   * between web server running Zabbix frontend and user web browser,
   * between Zabbix frontend and Zabbix server,
   * between Zabbix server (proxy) and Zabbix database.
* Currently each encrypted connection opens with a full TLS handshake, no session caching and tickets are
* Adding encryption increases time of checks and actions, depending on network latency.\\ For example, if
* Encryption is not supported by [[/manual/discovery/network_discovery|network discovery]]. Zabbix agent c
```

Compiling Zabbix with encryption support To support encryption Zabbix must be compiled and linked with one of three crypto libraries:

- *mbed TLS* (formerly *PolarSSL*)(version 1.3.9 and later 1.3.x). *mbed TLS* 2.x is not currently supported, it is not a drop-in replacement for 1.3 branch, Zabbix will not compile with *mbed TLS* 2.x.
- *GnuTLS* (from version 3.1.18)
- *OpenSSL* (from version 1.0.1)

The library is selected by specifying an option to "configure" script:

- `--with-mbedtls[=DIR]`
- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]`

For example, to configure the sources for server and agent with *OpenSSL* you may use something like:
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2 --with-openssl

Different Zabbix components may be compiled with different crypto libraries (e.g. a server with *OpenSSL*, an agent with *GnuTLS*).

Attention:
If you plan to use pre-shared keys (PSK) consider using *GnuTLS* or *mbed TLS* libraries in Zabbix components using PSKs. *GnuTLS* and *mbed TLS* libraries support PSK ciphersuites with Perfect Forward Secrecy. *OpenSSL* library (versions 1.0.1, 1.0.2c) does support PSKs but available PSK ciphersuites do not provide Perfect Forward Secrecy.

Connection encryption management Connections in Zabbix can use:

- no encryption (default)
- RSA certificate-based encryption
- PSK-based encryption

There are two important parameters used to specify encryption for connections between Zabbix components:

- TLSConnect
- TLSAccept

TLSConnect specifies what encryption to use for outgoing connections and can take one of 3 values (unencrypted, PSK, certificate). TLSConnect is used in configuration files for Zabbix proxy (in active mode, specifies only connections to server) and Zabbix agentd (for active checks). In Zabbix frontend the TLSConnect equivalent is *Connections to host* field in *Configuration→Hosts→<some host>→Encryption* tab and *Connections to proxy* field in *Administration→Proxies→<some proxy>→Encryption* tab. If the configured encryption type for connection fails, no other encryption types will be tried.

TLSAccept specifies what types of connections are allowed for incoming connections. Connection types are: unencrypted, PSK, certificate. One or more values can be specified. TLSAccept is used in configuration files for Zabbix proxy (in passive mode, specifies only connections from server) and Zabbix agentd (for passive checks). In Zabbix frontend the TLSAccept equivalent is *Connections from host* field in *Configuration→Hosts→<some host>→Encryption* tab and *Connections from proxy* field in *Administration→Proxies→<some proxy>→Encryption* tab.

Normally you configure only one type of encryption for incoming encryptions. But you may want to switch encryption type, e.g. from unencrypted to certificate-based with minimum downtime and rollback possibility.
To achieve this you can set TLSAccept=unencrypted,cert in agentd configuration file and restart Zabbix agent.
Then you can test connection with zabbix_get to the agent using certificate. If it works, you can reconfigure encryption for that agent in Zabbix frontend in *Configuration→Hosts→<some host>→Encryption* tab by setting *Connections to host* to "Certificate".
When server configuration cache gets updated (and proxy configuration is updated if the host is monitoring by proxy) then connections to that agent will be encrypted.
If everything works as expected you can set TLSAccept=cert in agent configuration file and restart Zabbix agent.
Now the agent will be accepting only encrypted certificate-based connections. Unencrypted and PSK-based connections will be rejected.

In a similar way it works on server and proxy. If in Zabbix frontend in host configuration *Connections from host* is set to "Certificate" then only certificate-based encrypted connections will be accepted from agent (active checks) and zabbix_sender (trapper items).

Most likely you will configure incoming and outgoing connections to use the same encryption type or no encryption at all. But technically it is possible to configure it asymmetrically, e.g. certificate-based encryption for incoming and PSK-based for outgoing connections.

For overview, encryption configuration for each host is displayed in Zabbix frontend *Configuration→Hosts* on the right side, in column *AGENT ENCRYPTION*. Configuration display examples:

Example	Connections TO host	Allowed connections FROM host	Rejected connections FROM host
NONE	Unencrypted	Unencrypted	Encrypted certificate and PSK-based
CERT NONE PSK CERT	Encrypted, certificate-based	Encrypted certificate-based	Unencrypted and PSK-based
PSK NONE PSK CERT	Encrypted, PSK-based	Encrypted PSK-based	Unencrypted and certificate-based
PSK NONE PSK CERT	Encrypted, PSK-based	Unencrypted and PSK-based encrypted	Certificate-based
CERT NONE PSK CERT	Encrypted, certificate-based	Unencrypted, PSK or certificate-based encrypted	-

Attention:
Default is unencrypted connections. Encryption must be configured for each host and proxy individually.

zabbix_get and zabbix_sender with encryption See man-pages zabbix_get and zabbix_sender for using them with encryption.

Ciphersuites Ciphersuites are configured internally during Zabbix startup and depend on crypto library, currently they are not user-configurable.

Configured ciphersuites by library type in order from higher to lower priority:

Library	Certificate ciphersuites	PSK ciphersuites
mbed TLS (PolarSSL) 1.3.9	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA256 TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA TLS-RSA-WITH-AES-128-GCM-SHA256 TLS-RSA-WITH-AES-128-CBC-SHA256 TLS-RSA-WITH-AES-128-CBC-SHA	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256 TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA TLS-PSK-WITH-AES-128-GCM-SHA256 TLS-PSK-WITH-AES-128-CBC-SHA256 TLS-PSK-WITH-AES-128-CBC-SHA
GnuTLS 3.1.18	TLS_ECDHE_RSA_AES_128_GCM_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA1 TLS_RSA_AES_128_GCM_SHA256 TLS_RSA_AES_128_CBC_SHA256 TLS_RSA_AES_128_CBC_SHA1	TLS_ECDHE_PSK_AES_128_CBC_SHA256 TLS_ECDHE_PSK_AES_128_CBC_SHA1 TLS_PSK_AES_128_GCM_SHA256 TLS_PSK_AES_128_CBC_SHA256 TLS_PSK_AES_128_CBC_SHA1
OpenSSL 1.0.2c	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-SHA256 AES128-SHA	PSK-AES128-CBC-SHA
OpenSSL 1.1.0	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA

Cipher suites using certificates:

	TLS server		
TLS client	*mbed TLS (PolarSSL)*	*GnuTLS*	*OpenSSL 1.0.2*
mbed TLS (PolarSSL)	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
GnuTLS	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
OpenSSL 1.0.2	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256

Cipher suites using PSK:

	TLS server		
TLS client	*mbed TLS (PolarSSL)*	*GnuTLS*	*OpenSSL 1.0.2*
mbed TLS (PolarSSL)	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
GnuTLS	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
OpenSSL 1.0.2	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA

1 Using certificates

Overview

Zabbix can use RSA certificates in PEM format, signed by a public or in-house certificate authority (CA). Certificate verification is done against a pre-configured CA certificate. Self-signed certificates are not supported. Optionally certificate revocation lists (CRL) can be used. Each Zabbix component can have only one certificate configured.

For more information how to set up and operate internal CA, how to generate certificate requests and sign them, how to revoke certificates you can find numerous online how-tos, for example, OpenSSL PKI Tutorial v1.1 .

Carefully consider and test your certificate extensions - see Limitations on using X.509 v3 certificate extensions.

Certificate configuration parameters

Parameter	Mandatory	Description
TLSCAFile	*	Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification. In case of certificate chain with several members they must be ordered: lower level CA certificates first followed by certificates of higher level CA(s). Certificates from multiple CA(s) can be included in a single file.
TLSCRLFile		Full pathname of a file containing Certificate Revocation Lists. See notes in Certificate Revocation Lists (CRL).
TLSCertFile	*	Full pathname of a file containing certificate (certificate chain). In case of certificate chain with several members they must be ordered: server, proxy, or agent certificate first, followed by lower level CA certificates then certificates of higher level CA(s).
TLSKeyFile	*	Full pathname of a file containing private key. Set access rights to this file - it must be readable only by Zabbix user.
TLSServerCertIssuer		Allowed server certificate issuer.
TLSServerCertSubject		Allowed server certificate subject.

Configuring certificate on Zabbix server

1. In order to verify peer certificates, Zabbix server must have access to file with their top-level self-signed root CA certificates. For example, if we expect certificates from two independent root CAs, we can put their certificates into file `/home/zabbix/zabbix_ca_file` like this:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
            ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            ...
-----BEGIN CERTIFICATE-----
MIID2jCCAsKgAwIBAgIBATANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
....
9wEzdN8uTrqoyU78gi12npLj08LegRKjb5hFTVmO
-----END CERTIFICATE-----
Certificate:
    Data:
```

```
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
            ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ....
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            ....
-----BEGIN CERTIFICATE-----
MIID3DCCAsSgAwIBAgIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQB
...
vdGNYoSfvu41GQAR5Vj5FnRJRzv5XQOZ3B6894GY1zY=
-----END CERTIFICATE-----
```

2. Put Zabbix server certificate chain into file, for example, `/home/zabbix/zabbix_server.crt`:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix server
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints:
                CA:FALSE
            ...
-----BEGIN CERTIFICATE-----
MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixk
...
h02u1GHiy46GI+xfR3LsPwFKlkTaaLaL/6aaoQ==
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
```

```
        ...
-----BEGIN CERTIFICATE-----
MIID4TCCAsmgAwIBAgIBAjANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
...
dyCeWnvL7u5sd6ffo8iRny0QzbHKmQt/wUtcVIvWXdMIFJM0Hw==
-----END CERTIFICATE-----
```

Here the first is Zabbix server certificate, followed by intermediate CA certificate.

3. Put Zabbix server private key into file, for example, `/home/zabbix/zabbix_server.key`:

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXDl
...
IJLkhbybBYEf47MLhffWa7XvZTY=
-----END PRIVATE KEY-----
```

4. Edit TLS parameters in Zabbix server configuration file like this:

```
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSCertFile=/home/zabbix/zabbix_server.crt
TLSKeyFile=/home/zabbix/zabbix_server.key
```

Configuring certificate-based encryption for Zabbix proxy

1. Prepare files with top-level CA certificates, proxy certificate (chain) and private key as described in Configuring certificate on Zabbix server. Edit parameters `TLSCAFile`, `TLSCertFile`, `TLSKeyFile` in proxy configuration accordingly.

2. For active proxy edit `TLSConnect` parameter:

```
TLSConnect=cert
```

For passive proxy edit `TLSAccept` parameter:

```
TLSAccept=cert
```

3. Now you have a minimal certificate-based proxy configuration. You may prefer to improve proxy security by setting `TLSServerCertIssuer` and `TLSServerCertSubject` parameters (see Restricting allowed certificate Issuer and Subject).

4. In final proxy configuration file TLS parameters may look like:

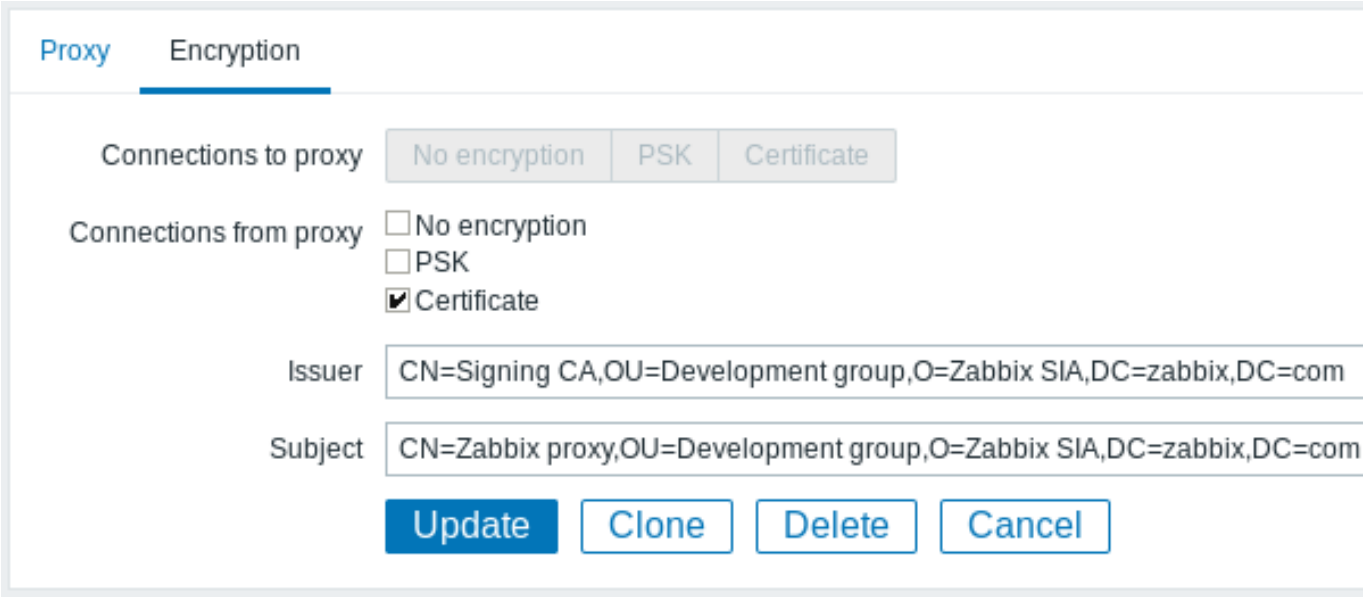
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_proxy.crt
TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

5. Configure encryption for this proxy in Zabbix frontend:

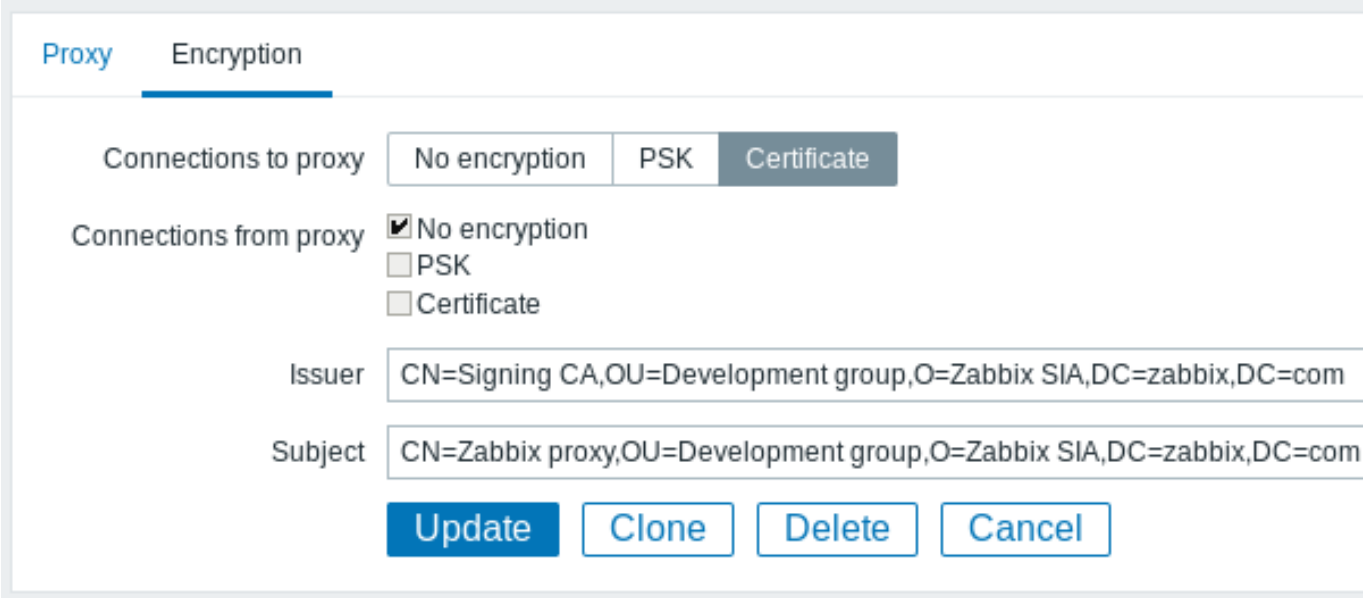
- Go to: *Administration* → *Proxies*
- Select proxy and click on **Encryption** tab

In examples below Issuer and Subject fields are filled in - see Restricting allowed certificate Issuer and Subject why and how to use these fields.

For active proxy



For passive proxy



Configuring certificate-based encryption for Zabbix agent

1. Prepare files with top-level CA certificates, agent certificate (chain) and private key as described in Configuring certificate on Zabbix server. Edit parameters `TLSCAFile`, `TLSCertFile`, `TLSKeyFile` in agent configuration accordingly.

2. For active checks edit `TLSConnect` parameter:

```
TLSConnect=cert
```

For passive checks edit `TLSAccept` parameter:

```
TLSAccept=cert
```

3. Now you have a minimal certificate-based agent configuration. You may prefer to improve agent security by setting `TLSServerCertIssuer` and `TLSServerCertSubject` parameters. (see Restricting allowed certificate Issuer and Subject).

4. In final agent configuration file TLS parameters may look like:

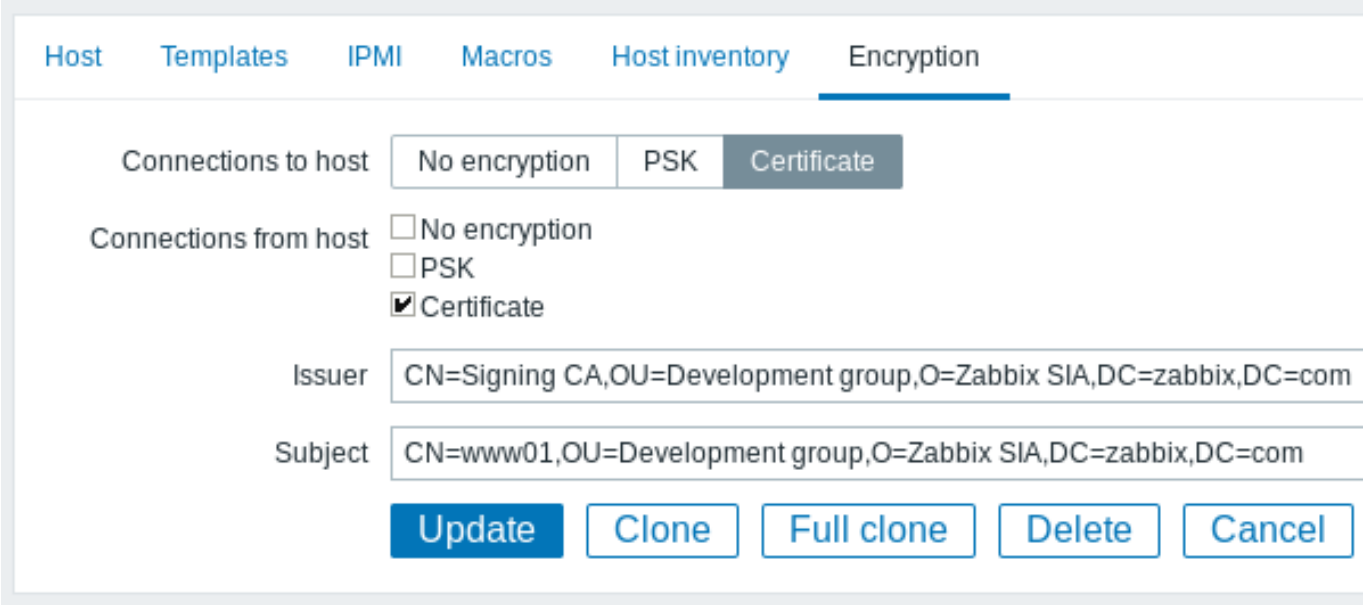
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_agentd.crt
TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

(Example assumes that host is monitored via proxy, hence proxy certificate Subject.)

5. Configure encryption for this agent in Zabbix frontend:

- Go to: *Configuration* → *Hosts*
- Select host and click on **Encryption** tab

In example below Issuer and Subject fields are filled in - see Restricting allowed certificate Issuer and Subject why and how to use these fields.



Restricting allowed certificate Issuer and Subject

When two Zabbix components (e.g. server and agent) establish a TLS connection they both check each others certificates. If a peer certificate is signed by a trusted CA (with pre-configured top-level certificate in `TLSCAFile`), is valid, has not expired and passes some other checks then communication can proceed. Certificate issuer and subject are not checked in this simplest case.

Here is a risk - anybody with a valid certificate can impersonate anybody else (e.g. a host certificate can be used to impersonate server). This may be acceptable in small environments where certificates are signed by a dedicated in-house CA and risk of impersonating is low.

If your top-level CA is used for issuing other certificates which should not be accepted by Zabbix or you want to reduce risk of impersonating you can restrict allowed certificates by specifying their Issuer and Subject strings.

For example, you can write in Zabbix proxy configuration file:

```
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

With these settings, an active proxy will not talk to Zabbix server with different Issuer or Subject string in certificate, a passive proxy will not accept requests from such server.

A few notes about Issuer or Subject string matching:

1. Issuer and Subject strings are checked independently. Both are optional.
2. UTF-8 characters are allowed.
3. Unspecified string means any string is accepted.
4. Strings are compared "as-is", they must be exactly the same to match.
5. Wildcards and regexp's are not supported in matching.
6. Only some requirements from RFC 4514 Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names are implemented:

```
  - escape characters '"' (U+0022), '+' U+002B, ',' U+002C, ';' U+003B, '<' U+003C, '>' U+003E, '\' U+00
  - escape characters space (' ' U+0020) or number sign ('#' U+0023) at the beginning of string.
  - escape character space (' ' U+0020) at the end of string.
- Match fails if a null character (U+0000) is encountered ([[http://tools.ietf.org/html/rfc4514|RFC 4514]]
- Requirements of [[http://tools.ietf.org/html/rfc4517| RFC 4517 Lightweight Directory Access Protocol (LD
```

Order of fields in Issuer and Subject strings and formatting are important! Zabbix follows RFC 4514 recommendation and uses "reverse" order of fields.

The reverse order can be illustrated by example:

```
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

Note that it starts with low level (CN), proceeds to mid-level (OU, O) and ends with top-level (DC) fields.

OpenSSL by default shows certificate Issuer and Subject fields in "normal" order, depending on additional options used:

```
$ openssl x509 -noout -in /home/zabbix/zabbix_proxy.crt -issuer -subject
issuer= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Signing CA
subject= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Zabbix proxy

$ openssl x509 -noout -text -in /home/zabbix/zabbix_proxy.crt
Certificate:
        ...
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
    ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix proxy
```

Here Issuer and Subject strings start with top-level (DC) and end with low-level (CN) field, spaces and field separators depend on options used. None of these values will match in Zabbix Issuer and Subject fields!

Attention:
To get proper Issuer and Subject strings usable in Zabbix invoke OpenSSL with special options `-nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname`:

```
$ openssl x509 -noout -issuer -subject \
        -nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname \
        -in /home/zabbix/zabbix_proxy.crt
issuer= CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
subject= CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

Now string fields are in reverse order, fields are comma-separated, can be used in Zabbix configuration files and frontend.

Limitations on using X.509 v3 certificate extensions

- **Subject Alternative Name (*subjectAltName*)** extension.
  Alternative subject names from *subjectAltName* extension (like IP address, e-mail address) are not supported by Zabbix. Only value of "Subject" field can be checked in Zabbix (see Restricting allowed certificate Issuer and Subject).
  If certificate uses the *subjectAltName* extension then result depends on particular combination of crypto toolkits Zabbix components are compiled with (it may or may not work, Zabbix may refuse to accept such certificates from peers).
- **Extended Key Usage** extension.
  If used then generally both *clientAuth* (TLS WWW client authentication) and *serverAuth* (TLS WWW server authentication) are necessary.
  For example, in passive checks Zabbix agent acts in a TLS server role, so *serverAuth* must be set in agent certificate. For active checks agent certificate needs *clientAuth* to be set.
  GnuTLS issues a warning in case of key usage violation but allows communication to proceed.
- **Name Constraints** extension.
  Not all crypto toolkits support it. This extension may prevent Zabbix from loading CA certificates where this section is marked as *critical* (depends on particular crypto toolkit).

Certificate Revocation Lists (CRL)

If a certificate is compromised CA can revoke it by including in CRL. CRLs can be configured in server, proxy and agent configuration file using parameter `TLSCRLFile`. For example:

```
TLSCRLFile=/home/zabbix/zabbix_crl_file
```

where `zabbix_crl_file` may contain CRLs from several CAs and look like:

```
-----BEGIN X509 CRL-----
MIIB/DCB5QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv
...
treZeUPjb7LSmZ3K2hpbZN7SoOZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t
...
CAEebS2CND3ShBedZ8YSil59O6JvaDP61lR5lNs=
-----END X509 CRL-----
```

CRL file is loaded only on Zabbix start. CRL update requires restart.

Attention:
If Zabbix component is compiled with *OpenSSL* and CRLs are used then each top and intermediate level CA in certificate chains must have a corresponding CRL (it can be empty) in `TLSCRLFile`.

Limitations on using CRL extensions

- **Authority Key Identifier** extension.
  CRLs for CAs with identical names may not work in case of *mbedTLS* (*PolarSSL*), even with "Authority Key Identifier" extension.

2 Using pre-shared keys

Overview

Each pre-shared key (PSK) in Zabbix actually is a pair of:

- non-secret PSK identity string,
- secret PSK string value.

PSK identity string is a non-empty UTF-8 string. For example, "PSK ID 001 Zabbix agentd". It is a unique name by which this specific PSK is referred to by Zabbix components. Do not put sensitive information in PSK identity string - it is transmitted over the network unencrypted.

PSK value is a hard to guess string of hexadecimal digits, for example, "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327b

Size limits

There are size limits for PSK identity and value in Zabbix, in some cases a crypto library can have lower limit:

Component	PSK identity max size	PSK value min size	PSK value max size
Zabbix	128 UTF-8 characters	128-bit (16-byte PSK, entered as 32 hexadecimal digits)	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
GnuTLS	128 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
mbed TLS (PolarSSL)	128 UTF-8 characters	-	256-bit (default limit) (32-byte PSK, entered as 64 hexadecimal digits)
OpenSSL	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)

Attention:
Zabbix frontend allows configuring up to 128-character long PSK identity string and 2048-bit long PSK regardless of crypto libraries used.
If some Zabbix components support lower limits it is a user responsibility to configure PSK identity and value with allowed length for these components.
Exceeding length limits results in communication failures between Zabbix components.

Before Zabbix server connects to agent using PSK, the server looks up the PSK identity and PSK value configured for that agent in database (actually in configuration cache). Upon receiving a connection the agent uses PSK identity and PSK value from its configuration file. If both parties have the same PSK identity string and PSK value the connection may succeed.

Attention:
It is a user responsibility to ensure that there are no two PSKs with the same identity string but different values. Failing to do so may lead to unpredictable disruptions of communication between Zabbix components using PSKs with this PSK identity string.

Generating PSK

For example, a 256-bit (32 bytes) PSK can be generated using the following commands:

- with *OpenSSL*:

```
$ openssl rand -hex 32
af8ced32dfe8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
```

- with *GnuTLS*:

```
$ psktool -u psk_identity -p database.psk -s 32
Generating a random key for user 'psk_identity'
Key stored to database.psk

$ cat database.psk
psk_identity:9b8eafedfaae00cece62e85d5f4792c7d9c9bcc851b23216a1d300311cc4f7cb
```

Note that "psktool" above generates a database file with a PSK identity and its associated PSK. Zabbix expects just a PSK in the PSK file, so the identity string and colon (':') should be removed from the file.

Configuring PSK for server-agent communication (example)

On the agent host, write the PSK value into a file, for example, `/home/zabbix/zabbix_agentd.psk`. The file must contain PSK in the first text string, for example:

```
1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

Set access rights to PSK file - it must be readable only by Zabbix user.

Edit TLS parameters in agent configuration file `zabbix_agentd.conf`, for example, set:

```
TLSConnect=psk
TLSAccept=psk
TLSPSKFile=/home/zabbix/zabbix_agentd.psk
TLSPSKIdentity=PSK 001
```

The agent will connect to server (active checks) and accept from server and `zabbix_get` only connections using PSK. PSK identity will be "PSK 001".

Restart the agent. Now you can test the connection using `zabbix_get`, for example:

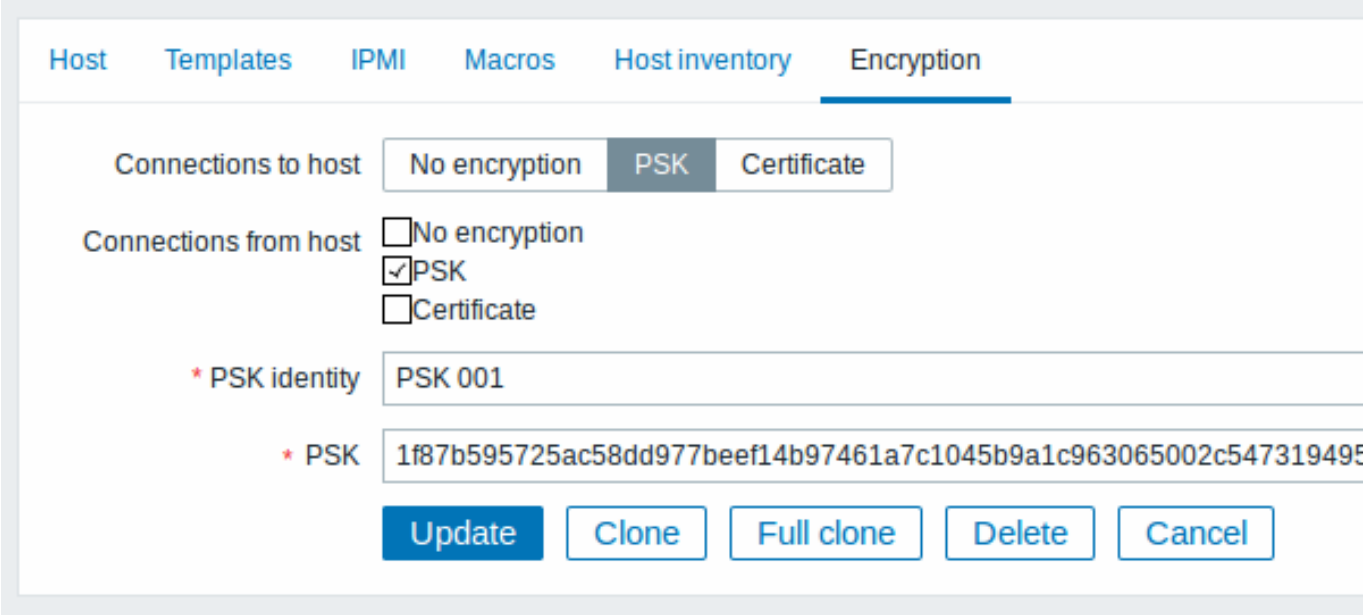
```
$ zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk \
            --tls-psk-identity="PSK 001" --tls-psk-file=/home/zabbix/zabbix_agentd.psk
```

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK encryption for this agent in Zabbix frontend:

- Go to: *Configuration* → *Hosts*
- Select host and click on **Encryption** tab

Example:



All mandatory input fields are marked with a red asterisk.

When configuration cache is synchronized with database the new connections will use PSK. Check server and agent logfiles for error messages.

Configuring PSK for server - active proxy communication (example)

On the proxy, write the PSK value into a file, for example, `/home/zabbix/zabbix_proxy.psk`. The file must contain PSK in the first text string, for example:

```
e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9
```

Set access rights to PSK file - it must be readable only by Zabbix user.

Edit TLS parameters in proxy configuration file `zabbix_proxy.conf`, for example, set:

```
TLSConnect=psk
TLSPSKFile=/home/zabbix/zabbix_proxy.psk
TLSPSKIdentity=PSK 002
```

The proxy will connect to server using PSK. PSK identity will be "PSK 002".

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK for this proxy in Zabbix frontend. Go to *Administration→Proxies*, select the proxy, go to "Encryption" tab. In "Connections from proxy" mark PSK. Paste into "PSK identity" field "PSK 002" and "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d083 into "PSK" field. Click "Update".

Restart proxy. It will start using PSK-based encrypted connections to server. Check server and proxy logfiles for error messages.

For a passive proxy the procedure is very similar. The only difference - set `TLSAccept=psk` in proxy configuration file and set "Connections to proxy" in Zabbix frontend to PSK.

3 Troubleshooting

General recommendations

- Start with understanding which component acts as a TLS client and which one acts as a TLS server in problem case.
  Zabbix server, proxies and agents, depending on interaction between them, all can work as TLS servers and clients.
  For example, Zabbix server connecting to agent for a passive check, acts as a TLS client. The agent is in role of TLS server.
  Zabbix agent, requesting a list of active checks from proxy, acts as a TLS client. The proxy is in role of TLS server.
  `zabbix_get` and `zabbix_sender` utilities always act as TLS clients.
- Zabbix uses mutual authentication.
  Each side verifies its peer and may refuse connection.
  For example, Zabbix server connecting to agent can close connection immediately if agent's certificate is invalid. And vice versa - Zabbix agent accepting a connection from server can close connection if server is not trusted by agent.
- Examine logfiles in both sides - in TLS client and TLS server.
  The side which refuses connection may log a precise reason why it was refused. Other side often reports rather general error (e.g. "Connection closed by peer", "connection was non-properly terminated").
- Sometimes misconfigured encryption results in confusing error messages in no way pointing to real cause.
  In subsections below we try to provide a (far from exhaustive) collection of messages and possible causes which could help in troubleshooting.
  Please note that different crypto toolkits (OpenSSL, GnuTLS, mbed TLS (PolarSSL)) often produce different error messages in same problem situations.
  Sometimes error messages depend even on particular combination of crypto toolkits on both sides.

1 Connection type or permission problems

Server is configured to connect with PSK to agent but agent accepts only unencrypted connections

In server or proxy log (with *mbed TLS* (*PolarSSL*) 1.3.11)

```
Get value from agent failed: ssl_handshake(): SSL - The connection indicated an EOF
```

In server or proxy log (with *GnuTLS* 3.3.16)

```
Get value from agent failed: zbx_tls_connect(): gnutls_handshake() failed: \
    -110 The TLS connection was non-properly terminated.
```

In server or proxy log (with *OpenSSL* 1.0.2c)

```
Get value from agent failed: TCP connection successful, cannot establish TLS to [[127.0.0.1]:10050]: \
    Connection closed by peer. Check allowed connection types and access rights
```

One side connects with certificate but other side accepts only PSK or vice versa

In any log (with *mbed TLS* (*PolarSSL*)):

```
failed to accept an incoming connection: from 127.0.0.1: ssl_handshake():\
    SSL - The server has no ciphersuites in common with the client
```

In any log (with *GnuTLS*):

```
failed to accept an incoming connection: from 127.0.0.1: zbx_tls_accept(): gnutls_handshake() failed:\
    -21 Could not negotiate a supported cipher suite.
```

In any log (with *OpenSSL* 1.0.2c):

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake returned error code 1:\
    file .\ssl\s3_srvr.c line 1411: error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher:\
    TLS write fatal alert "handshake failure"
```

2 Certificate problems

OpenSSL used with CRLs and for some CA in the certificate chain its CRL is not included in `TLSCRLFile`

In TLS server log in case of *mbed TLS (PolarSSL)* and *OpenSSL* peers:

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
    file s3_srvr.c line 3251: error:14089086: SSL routines:ssl3_get_client_certificate:certificate verify
    TLS write fatal alert "unknown CA"
```

In TLS server log in case of *GnuTLS* peer:

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
    file rsa_pk1.c line 103: error:0407006A: rsa routines:RSA_padding_check_PKCS1_type_1:\
    block type is not 01 file rsa_eay.c line 705: error:04067072: rsa routines:RSA_EAY_PUBLIC_DECRYPT:padd
```

CRL expired or expires during server operation

OpenSSL, in server log:

- before expiration:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
    SSL routines:ssl3_get_server_certificate:certificate verify failed:\
    TLS write fatal alert "certificate revoked"
```

- after expiration:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
    SSL routines:ssl3_get_server_certificate:certificate verify failed:\
    TLS write fatal alert "certificate expired"
```

The point here is that with valid CRL a revoked certificate is reported as "certificate revoked". When CRL expires the error message changes to "certificate expired" which is quite misleading.

GnuTLS, in server log:

- before and after expiration the same:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
      invalid peer certificate: The certificate is NOT trusted. The certificate chain is revoked.
```

mbed TLS (*PolarSSL*), in server log:

- before expiration:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    invalid peer certificate: revoked
```

- after expiration:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
      invalid peer certificate: revoked, CRL expired
```

3 PSK problems

PSK contains an odd number of hex-digits

Proxy or agent does not start, message in the proxy or agent log:

```
invalid PSK in file "/home/zabbix/zabbix_proxy.psk"
```

PSK identity string longer than 128 bytes is passed to GnuTLS

In TLS client side log:

```
gnutls_handshake() failed: -110 The TLS connection was non-properly terminated.
```

In TLS server side log.

```
gnutls_handshake() failed: -90 The SRP username supplied is illegal.
```

PSK longer than 32 bytes is passed to mbed TLS (PolarSSL)

In any Zabbix log:

```
ssl_set_psk(): SSL - Bad input parameters to function
```

18. Web 界面

Overview For an easy access to Zabbix from anywhere and from any platform, the web-based interface is provided.

> **Note:**
> Trying to access two Zabbix frontend installations on the same host, on different ports, simultaneously will fail. Logging into the second one will terminate the session on the first one - unless the default frontend session name is adjusted for the second frontend in frontend definitions (see ZBX_SESSION_NAME).

1 前端

1 监测

简介

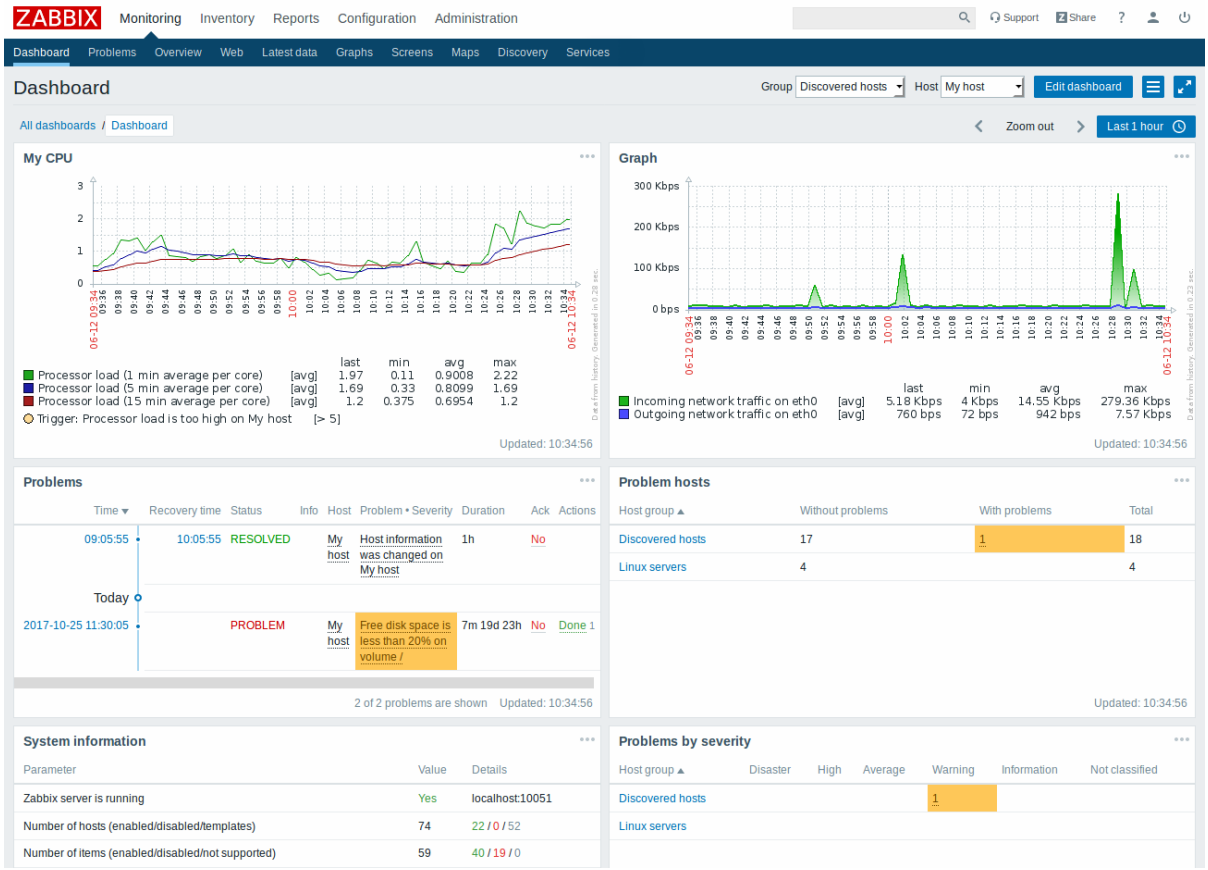
所有的监控数据都会在此模块中展示。你可以通过进行简单的配置把你需要展现的拓扑图、告警、聚合图形在此模块进行展示。

1 仪表板

简介

访问方式 监测 → 仪表板这里是监控信息的一个汇总。方便你快速总览当前全局监控状态。

仪表板是由多个小模块组成，可以有服务器信息、拓扑图、摘要、告警项、局和图形、时钟等模块进行组合展示。



在仪表板编辑模式下可以添加和编辑窗口模块。在仪表板查看模式下展示窗口模块。

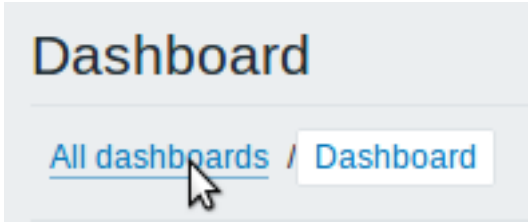
在定义单个仪表板中，您可以对来自各种来源的模块进行分组以便快速浏览，同时你还可以创建包含不同模块、内容的多个仪表板并在它们之间切换。

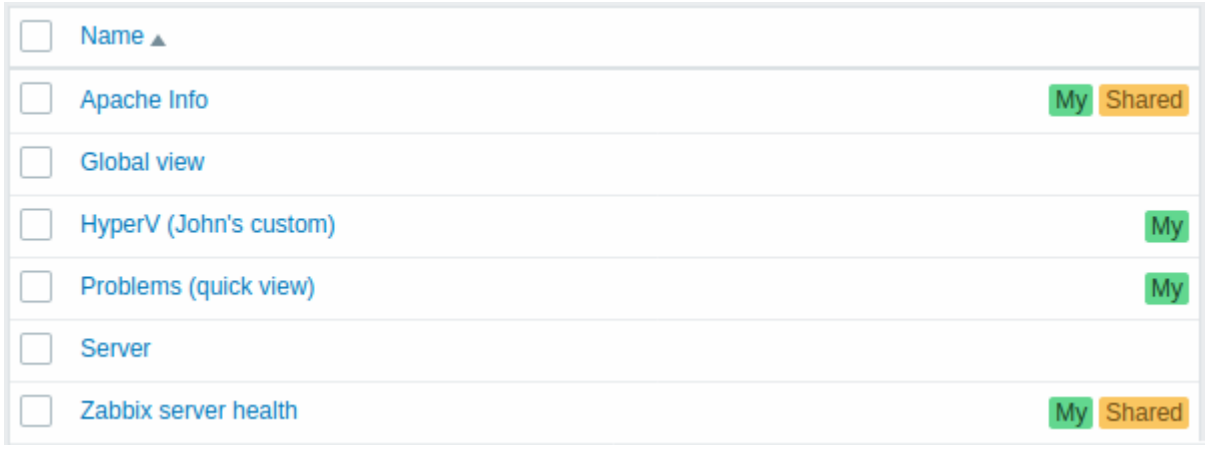
仪表板展示的时间范围你可以通过右上角的时间控制器来进行选择time period selector 单机时间控制器，来选择要查询的历史时间或者最近一段时间的数据。

提示：当使用仪表板的图形、局和图形等图形模块的时候，你可以通过双击、拉取等方式选取图形的时间展示周期，当你双击的时候会缩小显示时间范围，如果你需要选择摸个周期则可以直接拉取选取对应的时间段。

查看仪表板

要在访问或者管理已配置的所有仪表板时你需要这么操作 添加仪表板



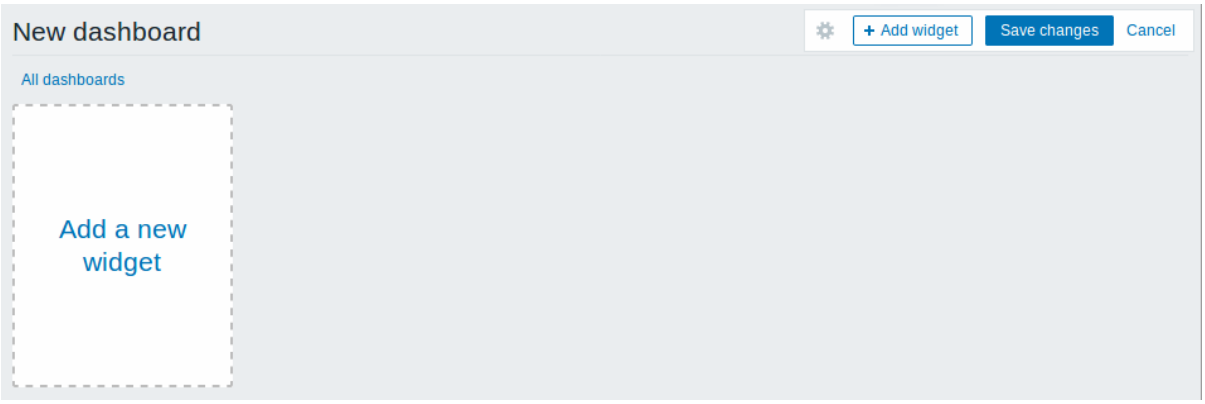


在仪表板列表中选择对应的仪表板连接。

如果你不在需要某个仪表板你同样可以在此界面单机左侧复选框选择 删除按钮后进行删除。

创建仪表板

如果想查看所有仪表板或者创建新的仪表板可以选择 添加仪表板这个链接来创建或者管理新的仪表板:



第一次访问的时候，仪表板为空，你可以通过以下两种方式创建新的仪表板:

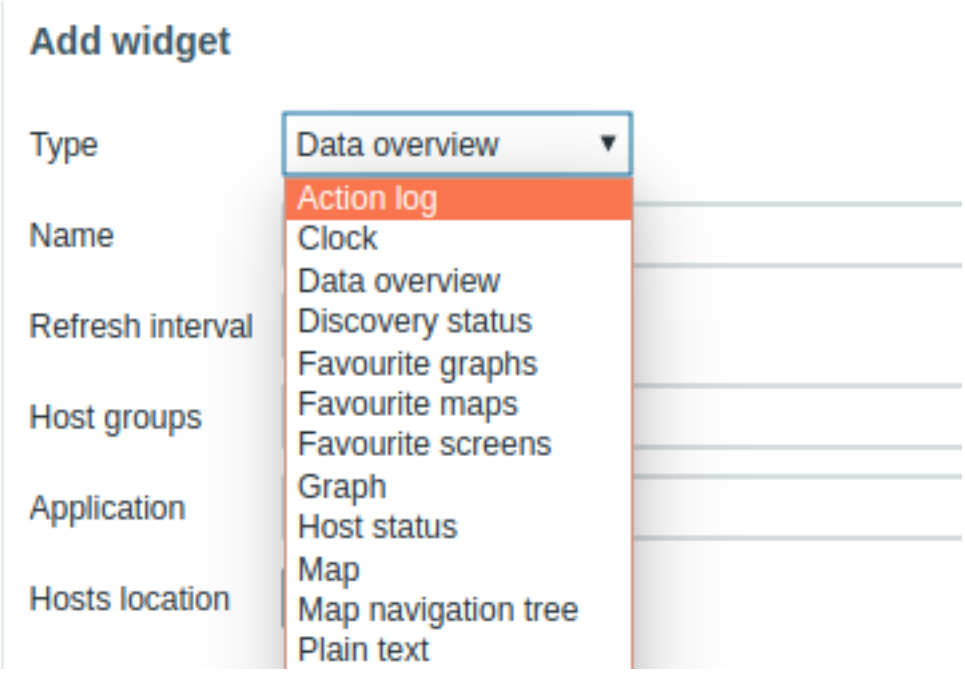
- 单击右上角 添加新构件按钮
- 单击页面左边 添加新构件的连接

在弹出的选项框中单击 添加即可创建新的仪表板. 如果你想取消创建可以选择 取消按钮来结束创建新的仪表板。

添加小构件

你可以通过以下方式添加小构件到仪表板:

- 单击选择 小构件按钮或者链接并且选择小构件的类型
- 选择 类型
- 根据自己的需要填写小构件的相关参数
- 单击 添加



可以添加到仪表板的小构件类型有以下内容:

- Action log
- Clock
- Data overview
- Discovery status
- Favourite graphs
- Favourite maps
- Favourite screens
- Graph
- Problem hosts

- Map
- Map navigation tree
- Plain text
- Problems
- System information
- Problems by severity
- Trigger overview
- URL
- Web monitoring

在仪表板编辑模式中，可以通过单击小构件标题栏并将其拖动到新位置。此外，您可以单击窗口小构件中的以下按钮:

- - 编辑构件;
- - 删除构件;

单击 添加来保存你刚刚在仪表板对小部件进行任何更改。

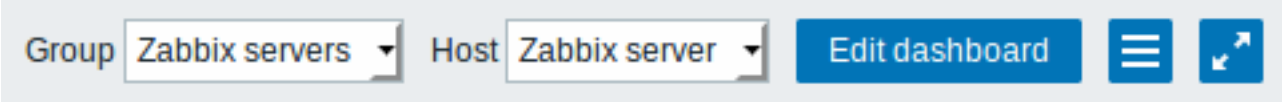
动态小构件

在配置 一些小部件的时:

- 图形 (或者简单图形)
- 纯文本
- URL

还有一个特别的选项 动态监控项。如果你勾选此项，便可以根据选择不同的主机展示不同的内容，或者持续展示更新最新的数据信息。

当你勾选动态监控后，在保存仪表板时，您会注意到仪表板上方出现了两个新的下拉列表，用于选择主机组/主机:



因此，您有一个窗口小构件，它可以显示基于下拉列表中所选主机的数据的内容。这样做的好处是您不需要创建额外的小构件，例如，您希望看到包含来自不同主机的数据的相同图形。

查看或编辑仪表板

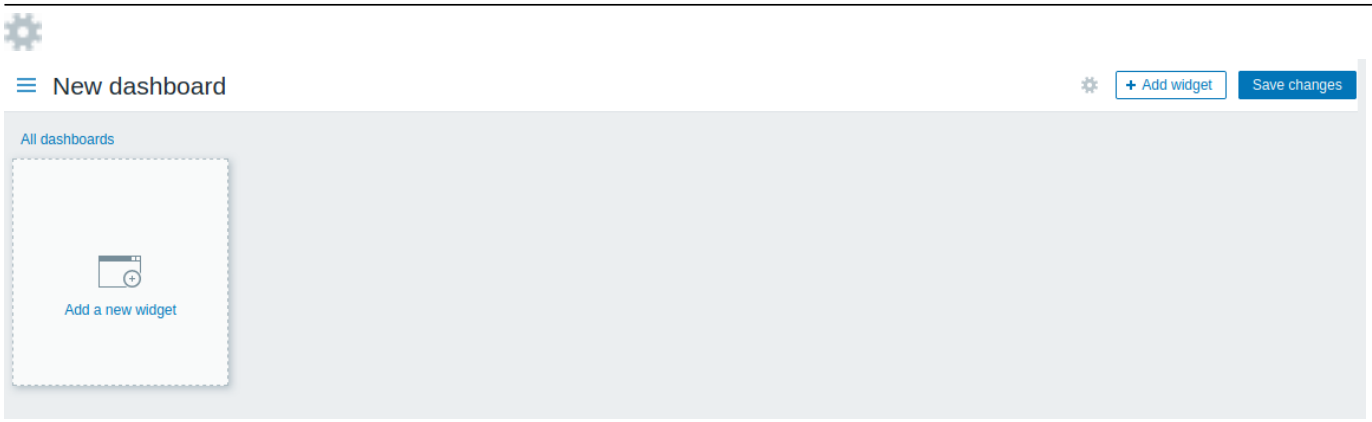
你可以通过以下选项查看单个仪表板:

Edit dashboard		切换到编辑模式。
		打开操作菜单。
	Sharing	编辑仪表板可以被哪些用户访问。\\默认的情况下所有用户都可以看到该仪表板。私有仪表板默认仅能自己查看，当然你可以选择分享给其他用户或者组查看 如果你想了解更多分享的配置信息可以参考配置 连接。
	Create new	创建一个新的仪表板。 首先，系统会提示您输入新仪表板的常规属性 - 所有者和名称。然后，新仪表板将以编辑模式打开，您可以添加小部件。
	Clone	克隆并且创建一个新的仪表板 首先，系统会提示您输入新仪表板的常规属性 - 所有者和名称。然后，新仪表板将以编辑模式打开，其中包含原始仪表板的所有小构件。
	Delete	删除当前仪表板
		全屏展示仪表板
		在自助终端下显示仪表板. 这个模式仅展示小构件。

编辑仪表板模式:

- 选择你要编辑的仪表板
- 选择按钮 编辑仪表板

在仪表板编辑模式中，可以使用以下选项:



编辑常规

添加新的

保存更改

取消更改

仪表板权限

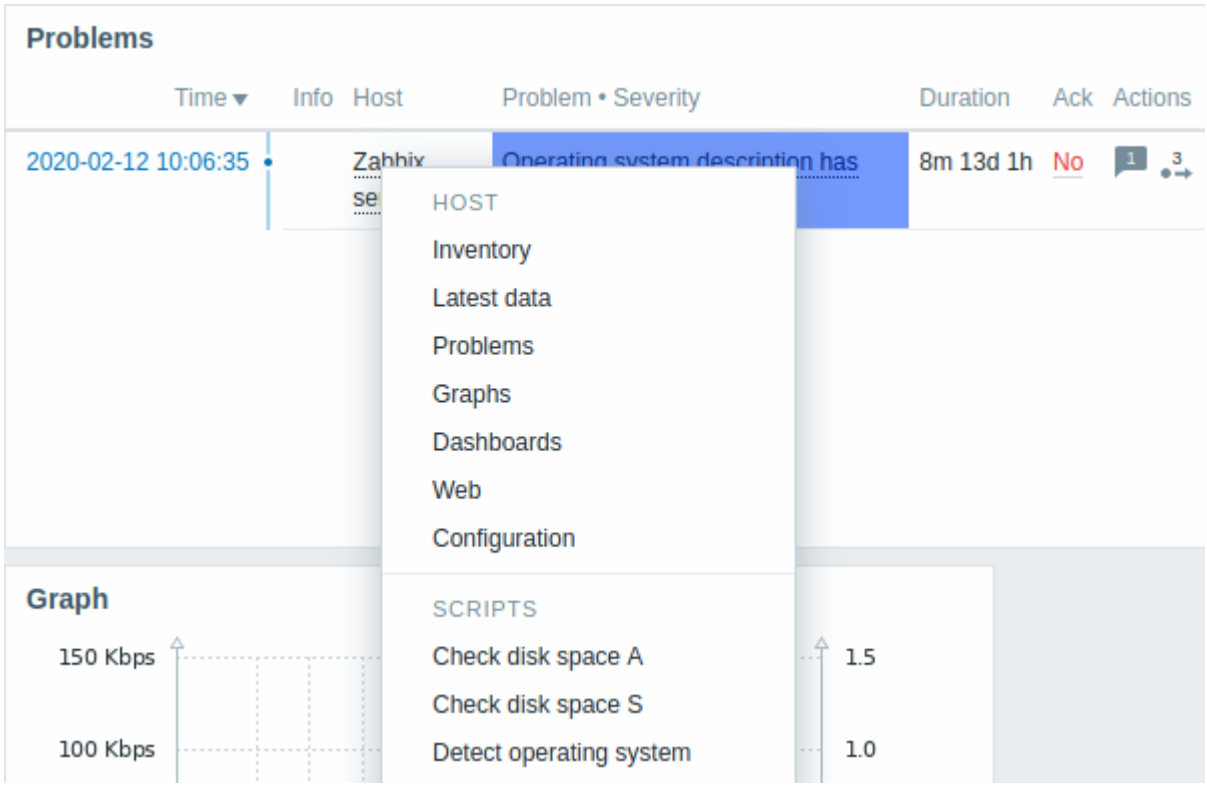
从 Zabbix 3.4.2 以后，普通和 Zabbix Admin 用户的仪表板权限受到以下限制：:

- 如果他们拥有 READ 权限，他们可以查看和克隆仪表板;
- 如果要编辑和删除仪表板需要 READ 以及 WRITE 的权限;
- 他们无法更改仪表板的所属用户.

在 Zabbix 3.4.2 之前管理员权限的用户不受到此规则限制。

主机菜单

单击// 问题 //小构件中的主机将显示主机菜单。它包含指向主机的自定义脚本，最新数据，触发器，库存，图形和屏幕的链接。

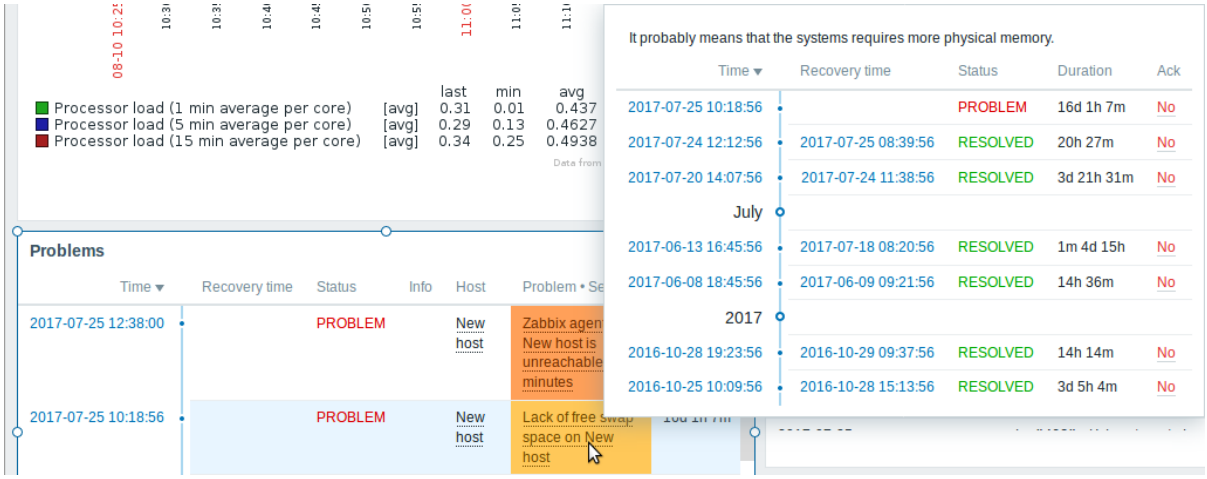


单击 WEB 前端中其他位置中的主机名，也可以访问此主机菜单。

- Monitoring → 问题 (Problems)
- Monitoring → 问题 (Problems) → 事件详情 (Event details)
- Monitoring → 概述 (Overview) (主机位置选择为左侧)
- Monitoring → 最新数据 (Latest data)
- Monitoring → 聚合图形 (Screens) (在 主机问题以及主机组问题小构件)
- Monitoring → 拓扑图 (Maps)
- Reports → 触发器 TOP 100 (Triggers top 100)

触发器弹出窗口

点击近 20 个问题窗口中的问题（issue），会调出触发器事件弹出式菜单。它包括该事件的列表，事先定义好的触发器描述和可点击的URL。



1 仪表板小构件

简介

本文列出了一些可用的dashboard 小构建，并且提供了一些配置方法。

以下参数对于每个小构建都是通用的:

Name	输入小构件名称。
Refresh interval	配置默认刷新间隔。窗口小构件的默认刷新间隔范围为无刷新 *(No refresh)* 到// 15 分钟 (15 minutes) 取决于窗口小构件的类型。例如：没有刷新用于 *URL* 小构件， 1 分钟用于操作日志小构件， 15 分钟//用于时钟小构件。

可以将窗口小构件的刷新间隔设置为统一的一个默认值，另外每个用户也可以设置自己的刷新间隔值:

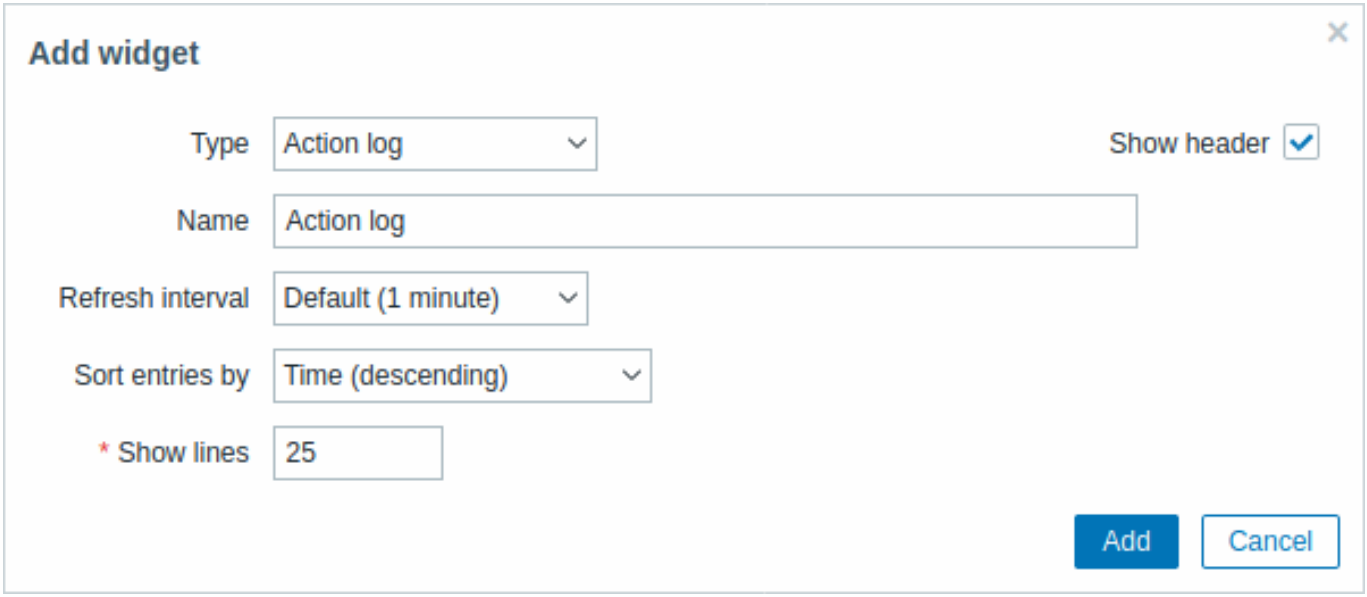
- 如果要设置全局用户的默认刷新值，请切换到编辑模式（单击 “编辑仪表板” 按钮，找到要设置的小构件，单击 “编辑” 按钮，编辑小构件表单是现在打开）并从下拉列表中选择所需的刷新间隔。
- 通过单击某个窗口小构件的 按钮，可以在视图模式下单独为某个用户设置唯一的刷新间隔。

注意的是，单独针对某个用户刷新值的优先级大于全局默认刷新值的优先级。设置后会如果不再更改的话会一直保留。

动作日志

In the action log widget you can display details of action operations (notifications, remote commands). It replicates information from *Administration* → *Audit*.

To configure, select *Action log* as type:



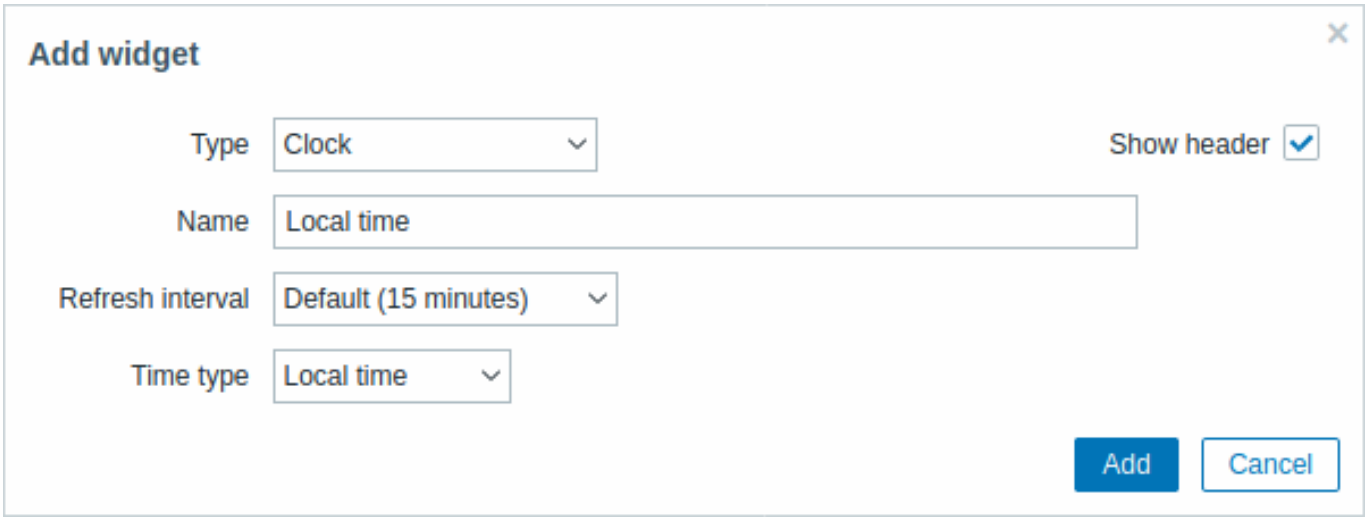
你可以设置以下特定选项:

以目标排序以目标排	: 时间（**Time**）（升序或者降序） 类型（**Type**）（升序或者降序） 状态（**Status**）（升序或者降序） 接收（**Recipient**）（升序或者降序）。
展示行设置	口小构件中将显示的操作日志行数。

时钟

在时钟小构件中，您可以显示本地、服务器或指定的主机时间。

如果要配置, 请选择 时钟 *(Clock)* 类型：



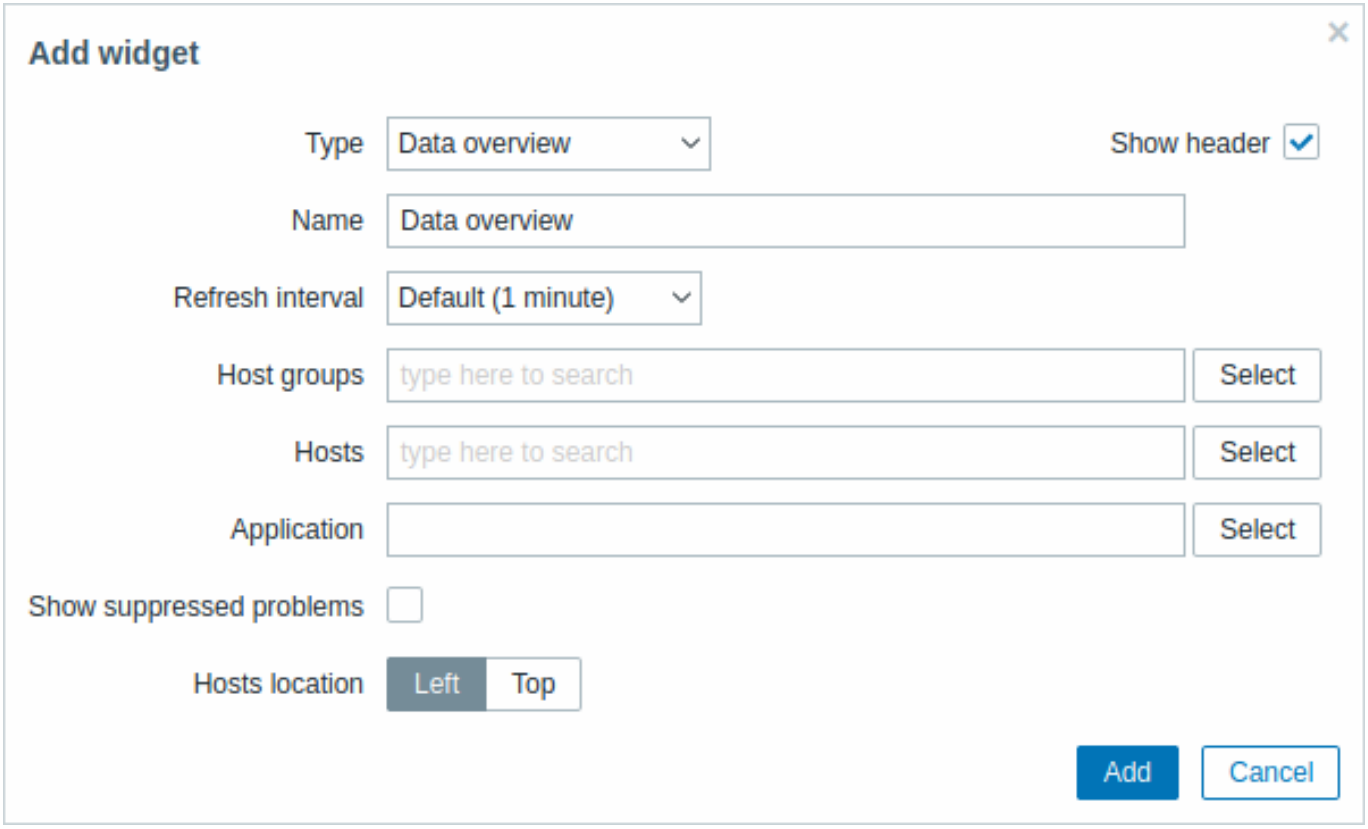
你可以设置以下特定选项:

时间类型本地时	、服务器时间或者指定某个服务器的时间。
项目选	显示时间的项目。要显示主机时间，请使用 `system.localtime[local]` 这个功能仅允许在设置显示服务器时间时使用。主机时间（*Host time*）选中。

数据概述

在数据概述窗口小构件中，您可以显示一组主机的最新数据。窗口内容数据源取自 监测中 → 概述 (数据类型选择 数据（*Data*）).

如果你要配置 //数据概述 // 类型:



你可以使用以下特定配置选项:

主机组输入	机组关键字即可触发自动匹配，选择需要设置的主机组即可，如果想要删除则单击主机组旁边的 X 来进行删除。
应用输	应用名称。
主机地址选择主	位置，左侧或者顶部。

自动发现状态

此小构件显示启用的网络发现规则状态摘要。

喜欢的图表

此小构件包含最需要的图表的快捷方式。当您查看图形时单击其添加到收藏夹按钮时，将填充快捷方式列表。

喜欢的拓扑图

此小构件包含最需要的拓扑图的快捷方式。当您查看图形时单击其添加到收藏夹按钮时，将填充快捷方式列表。

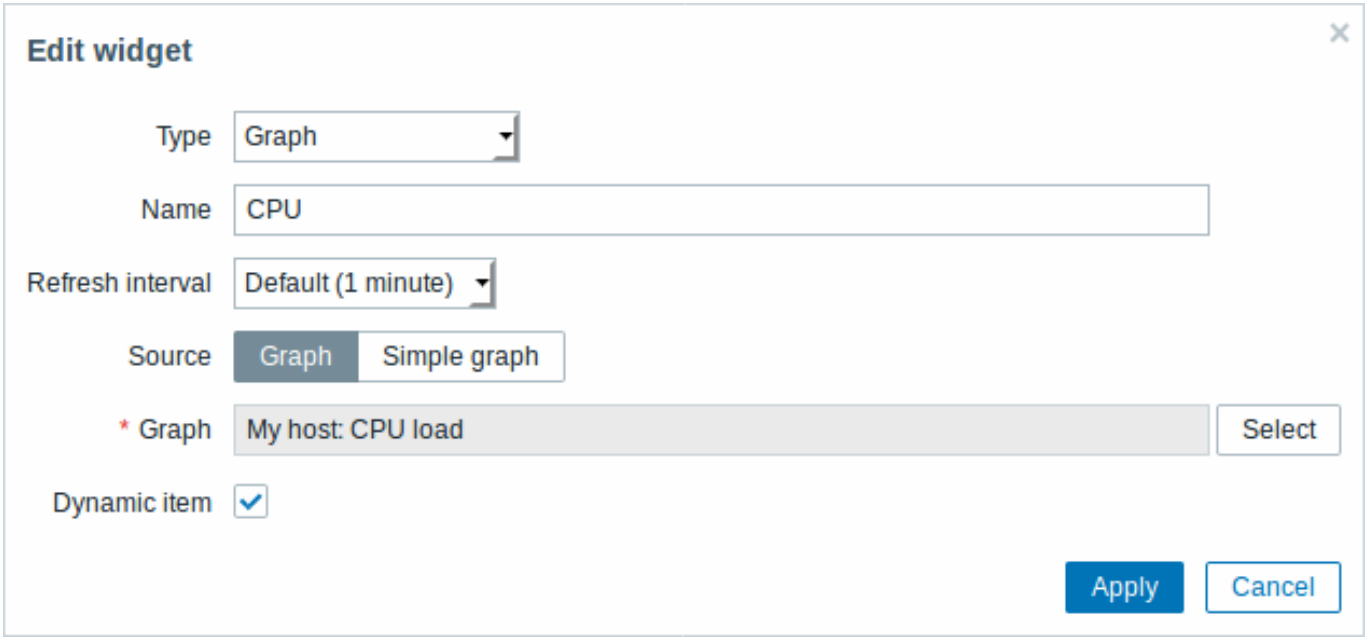
喜欢的聚合图形

此小构件包含最需要的聚合图形的快捷方式。当您查看图形时单击其添加到收藏夹按钮时，将填充快捷方式列表。

图形

在图表小构件中，您可以展示某个自定义图形或简单图形。

如果要配置, 请选择 图形 *(Graph)* 类型：



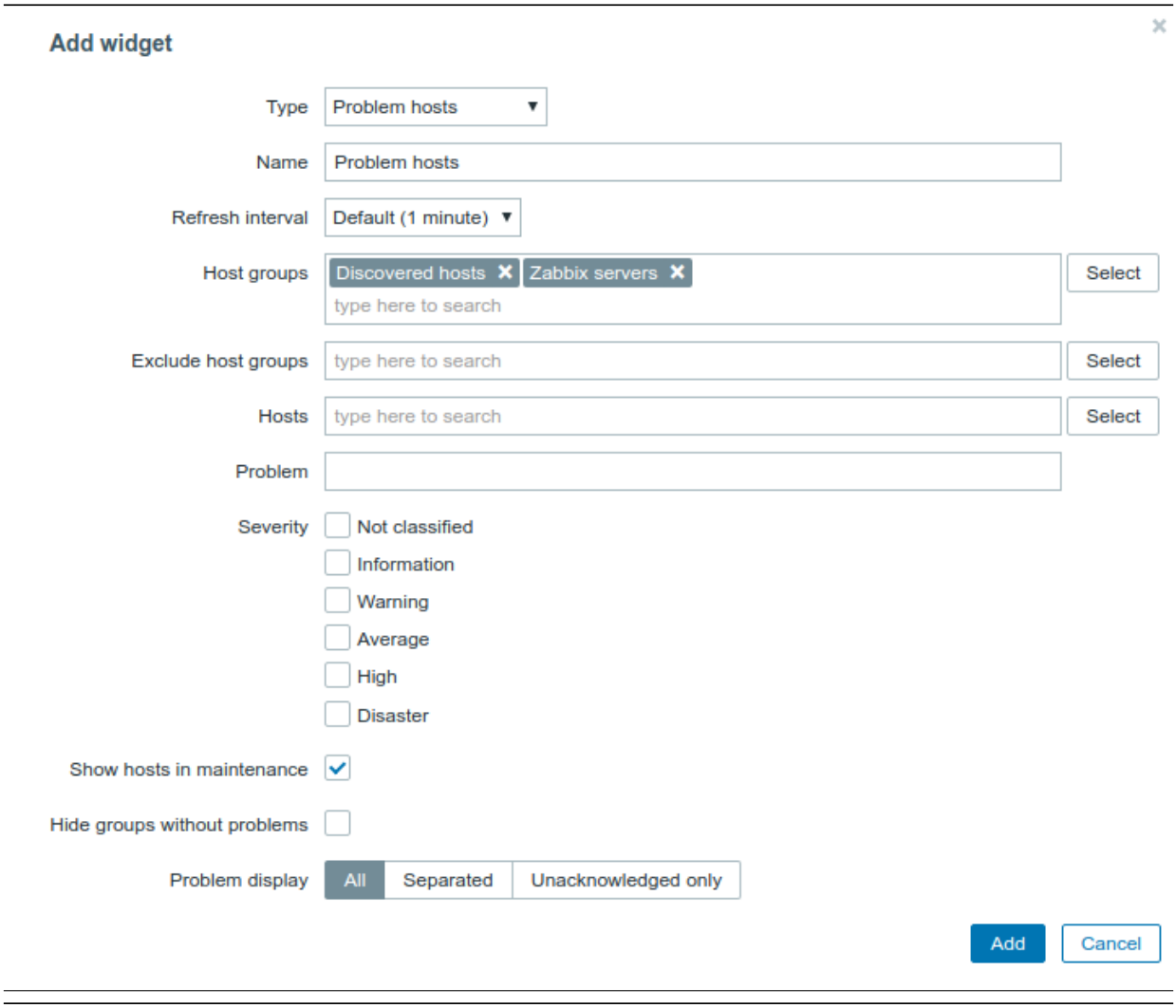
你可以设置以下特定选项:

来源选	图形类型： 图形 - 自定义 简单图形 - 简单图
图形要	示自定义图形时。源应该选择为\\“图形”。
选项要	示简单图形时源应该选择为\\“简单图形”。
动态监控项 设置图表	据所选主机显示不同的数据。

异常主机

在主机信息窗口小构件中，可以展示有关主机可用性的高级信息。

如果你需要设置，请选择 异常主机（*Problem hosts*）类型:



你可以设置以下特定选项：

参数类型	功能说
主机组输入	机组关键字即可触发自动匹配，选择需要设置的主机组即可，如果想要删除则单击主机组旁边的X来进行删除。\\指定父主机组会隐式选择所有嵌套的主机组。

参数类型功能说	
排除主机组输入要从	口小构件隐藏的主机组。输入主机组关键字即可触发自动匹配，选择需要设置的主机组即可，如果想要删除则单击主机组旁边的x来进行删除。\指定父主机组

参数	类型	功能说
主机输		要展示的主机，仅输入关键词即可实现自动匹配后进行选择。如果你不输入任何内容默认显示所有主机。

参数	类型	功能说
问题	你	以设置过滤仅展示哪些问题，或者展示哪些主机的哪些问题。如果想进行关键字过滤可以再like或者Equal输入字符。宏不会触发匹配。

参数类型	功能说
问题等级选择哪	问题等级可以在此窗口中展示。
显示维护中的主机选择此项后，如	主机处于异常并且同时又在维护中的时候依然会显示，默认勾选。如果不需要可以取消勾选。

参数	类型	功能说
Hide groups without problems		Mark the *Hide groups without problems* option to hide data from host groups without problems in the widget.

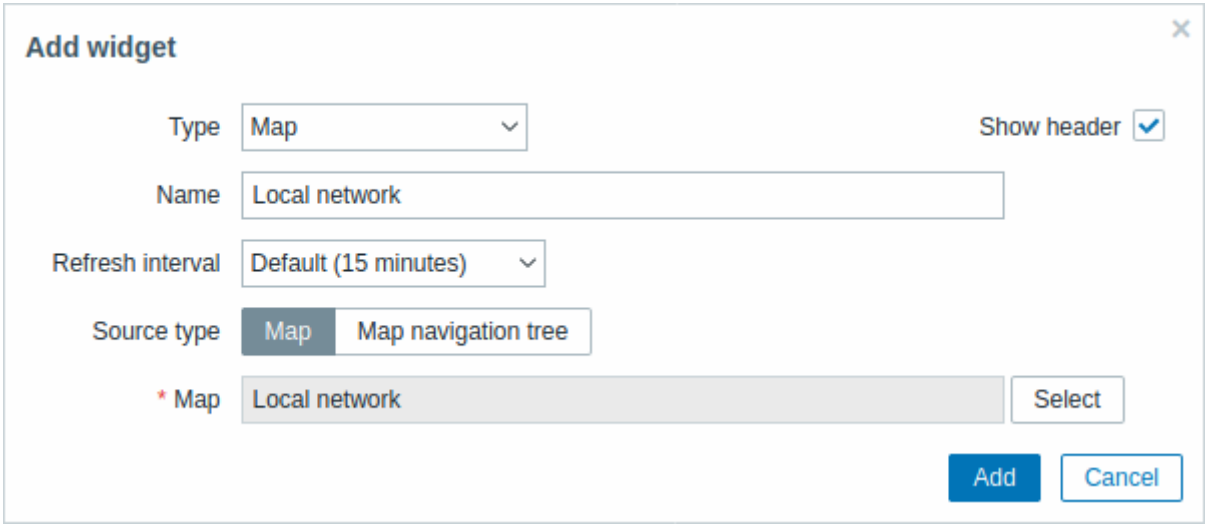
参数	类型	功能说
显示问题	显示问	统计：所有 - 展示所有问题数。分隔 - 未确认的问题数将以总问题数分开显示未确认问题 - 仅显示未确认的问题计数。

拓扑图

使用拓扑图小构件展示:

- 单独的网络映射
- 拓扑图导航树中配置的网络拓扑图（单击树中的拓扑图名称时）。

配置中选择 拓扑图（*Map*）类型:



你可以设置以下选项:

来源类型选择显	： 拓扑图 - 网络拓扑图 拓扑图导航树 - 所选拓扑图导航树中的一个拓扑图
地图选	要显示的拓扑图。当选择 “拓扑图” 类型作为源类型时可用。
过滤选	拓扑图导航树显示。当选择 “拓扑图导航树” 作为源类型时可用。

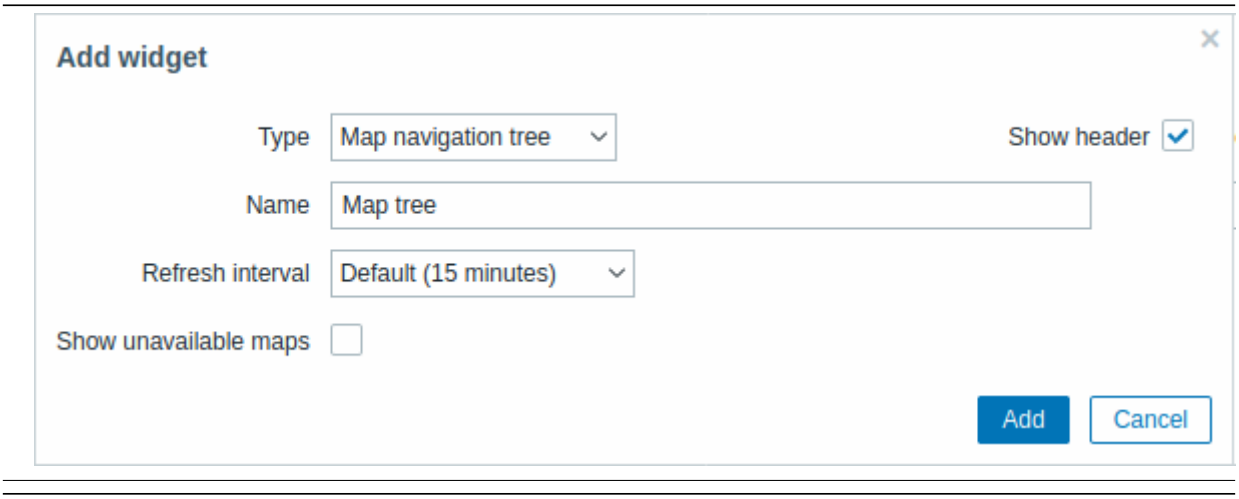
拓扑图导航树

此窗口小构件允许构建现有拓扑图的层次结构，同时还显示每个包含的拓扑图和拓扑图组的问题统计信息。

如果将// 拓扑图 小构件链接到导航树，它会变得更加强大。在这种情况下，单击导航树中的拓扑图名称会在拓扑图 //小构件中完整显示拓扑图。

在分层展示中，最上面一层将会展示所有问题的总和。

要配置导航树窗口小构件，请选择拓扑图导航树（*Map navigation tree*）作为类型：



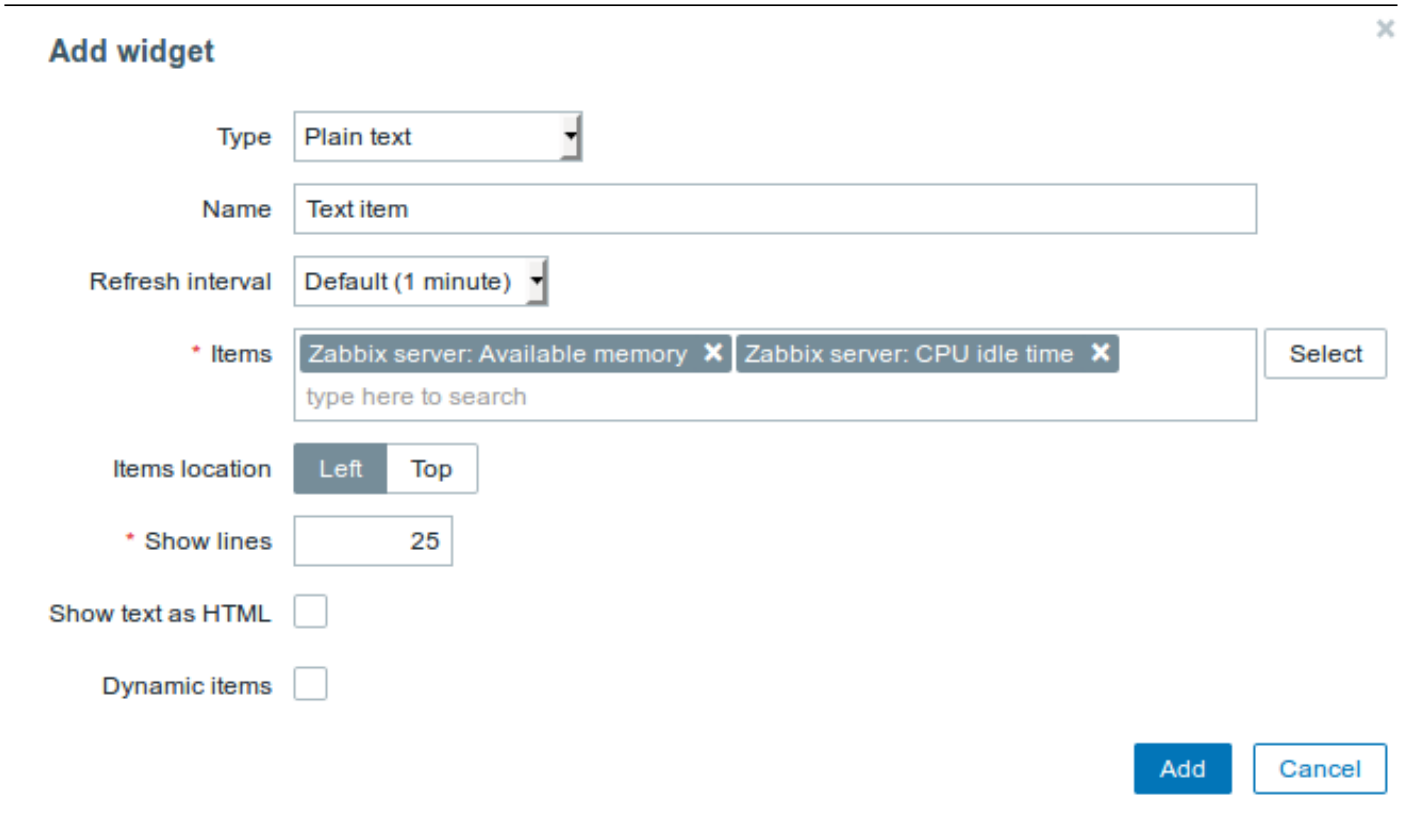
你还需要配置以下选项:

展示不可用的拓扑图选中此复选框以显	用户没有读取权限的拓扑图。 导航树中的不可用拓扑图将显示为带有灰色图标。 注意，如果标记了此复选框，则即使显示可用的子拓扑图，也会显示父级别拓扑图是无关紧要的。如果未标记，则根本不会显示不可用父图的可用子图。 问题计数是根据可用的拓扑图和可用的拓扑图元素计算的。

文本

在此小构件中，您可以以纯文本格式展示最新的项目数据

配置方法, 选择文本（*Plain text*）类型:



你还需要设置以下选项:

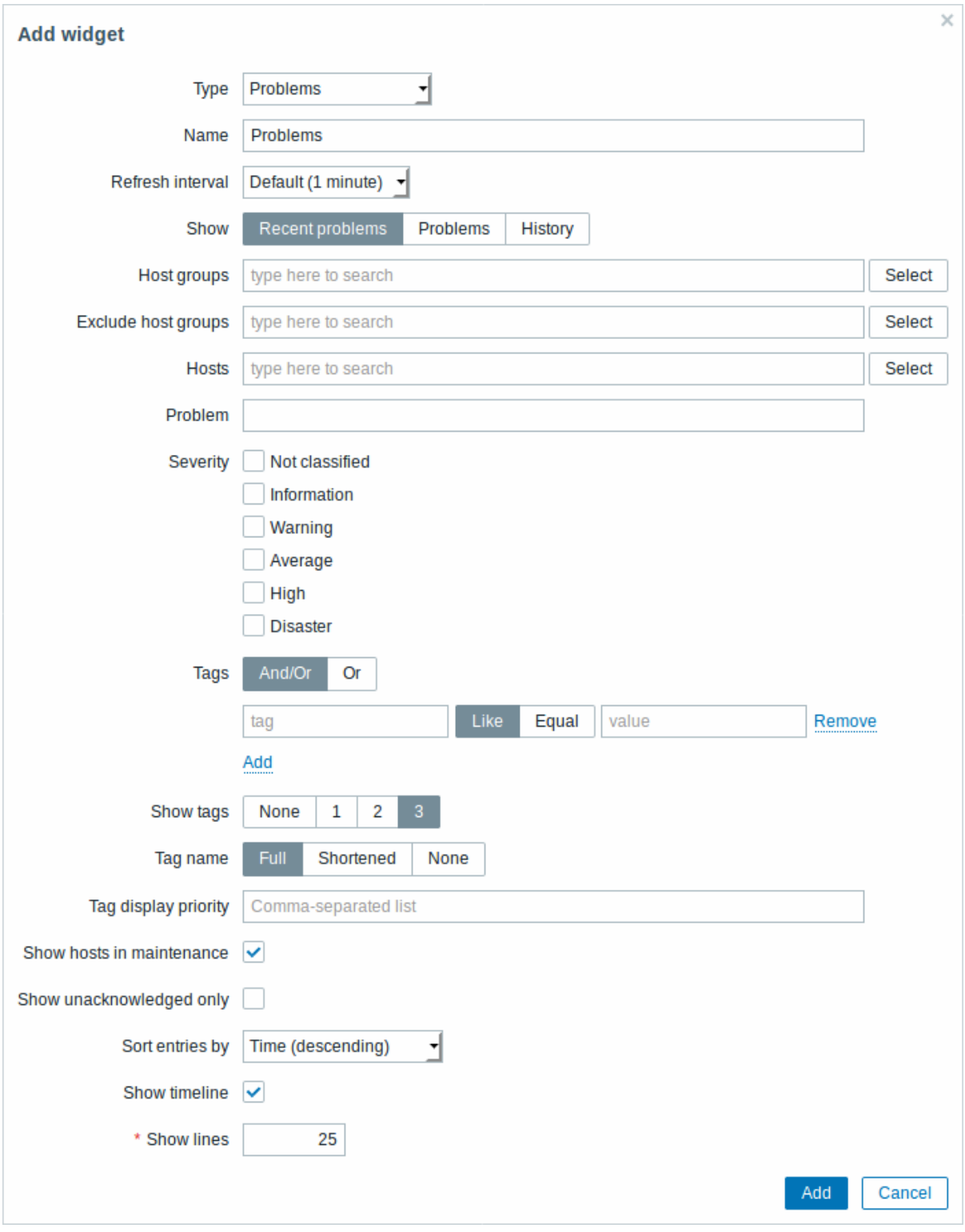
监控项选择	应的监控项。

监控项位置设置监控	的位置
展示行设置	示多少行数据
查看 *HTML* 文字展示 H	ML 文字
动态监控项 针对不同	机展示不同内容。

异常

在小构件中展示异常信息. 此构件中内容类似 监测中（*Monitoring*） → 异常（*Problems*）.

配置方法，选择 异常（*Problems*）类型:



您可以通过各种方式限制窗口小部件中显示的问题数量 - 问题状态，问题名称，严重性，主机组，主机，事件标记，确认状态等。

参数	类型	功能说
查看根		问题过滤，最近的问题：显示未解决的问题以及最近发生的问题。(默认选项)；问题：显示所有未解决的问题。历史记录：显示所有事件记录

参数类型	功能说
主机组输入	机组以显示窗口小构件中的问题。此字段是自动完成的，可以根据关键字快速搜索。\\指定父主机组会隐式选择所有嵌套的主机组。\\来自这些主机组的问题将显

参数类型功能说	
排除主机组输入要排	的主机组。此字段是自动完成的，可以根据关键字快速搜索。\\指定父主机组会隐式选择所有嵌套的主机组。\\这些主机组中的问题不会显示在小构件。例如，主机001,002,003

参数	类型	功能说
主机	E	ter hosts to dis- play prob- lems of in the wid- get. This field is auto- complete so start- ing to type the name of a host will of- fer a drop- down of match- ing hosts. If no hosts are en- tered, prob- lems of all hosts will be dis- played.

参数	类型	功能说
监控项名称	You	an limit the number of problems displayed by their name. If you enter a string here, only those problems whose name LIKE matches the entered string will be displayed. Macros are not expanded.
异常等级	Mar	the problem severities to be displayed in the widget.

参数	类型	功能说
标签 S		ecify event tag name and value to limit the number of problems displayed. To add more event tag names and values, click on *Add*. There are two calculation types for several conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or con-

参数类型功能说	
Show tags	Select the number of displayed tags: **None**- no *Tags* column in *Problems* widget **1**- *Tags* column contains one tag **2**- *Tags* column contains two tags **3**- *Tags* column contains three tags To see all tags for the problem roll your mouse over the three dots icon.

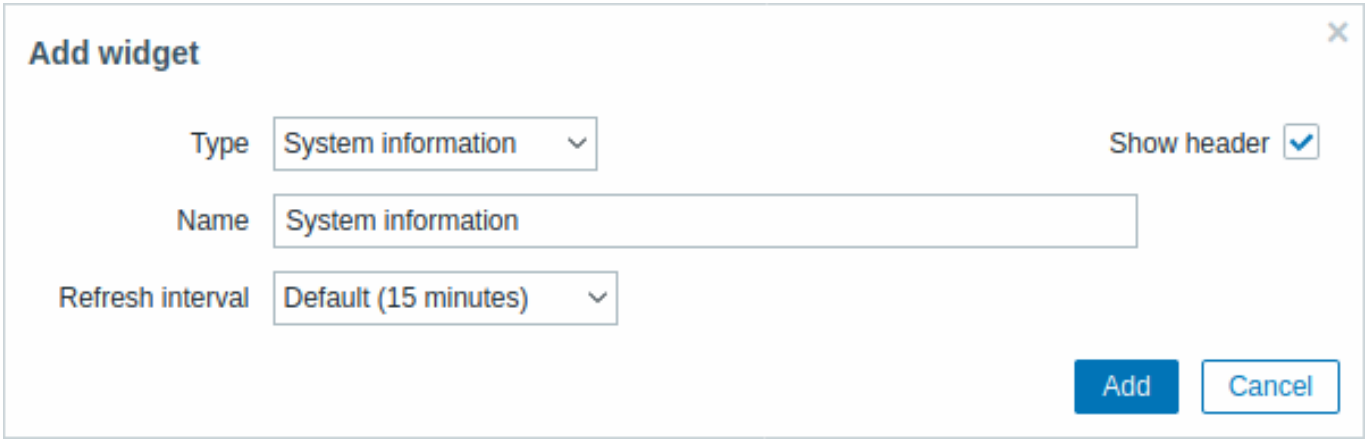
参数类型功能说	
Show hosts in maintenance	Mark the check-box to dis-play prob-lems of hosts in main-te-nance, too.
Show unacknowledged only	Mark the check-box to dis-play un-ac-knowl-edged prob-lems only.

参数类型功能说	
Sort entries by	Sort entries by: **Time** (descending or ascending) **Severity** (descending or ascending) **Problem name** (descending or ascending) **Host** (descending or ascending).
Show timeline	Mark the checkbox to display a visual timeline.
Show lines	Specify the number of problem lines to display.

System information

In the System information widget you can display high-level Zabbix and Zabbix server information.

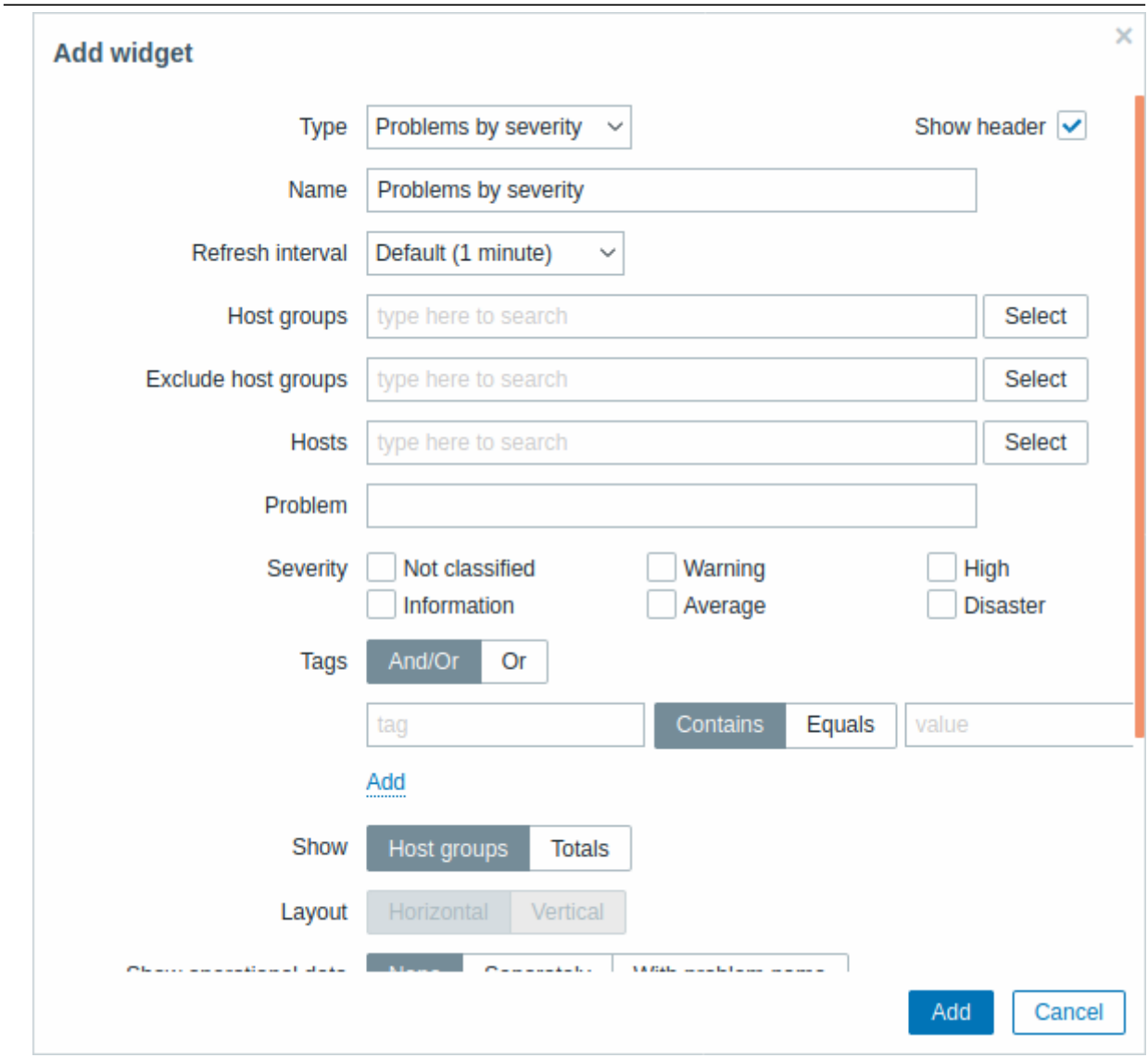
To configure, select *System information* as type:



Problems by severity

In this widget you can display problems by severity. You can limit what hosts and triggers are displayed in the widget and define how the problem count is displayed.

To configure, select *Problems by severity* as type:



You may set the following specific options:

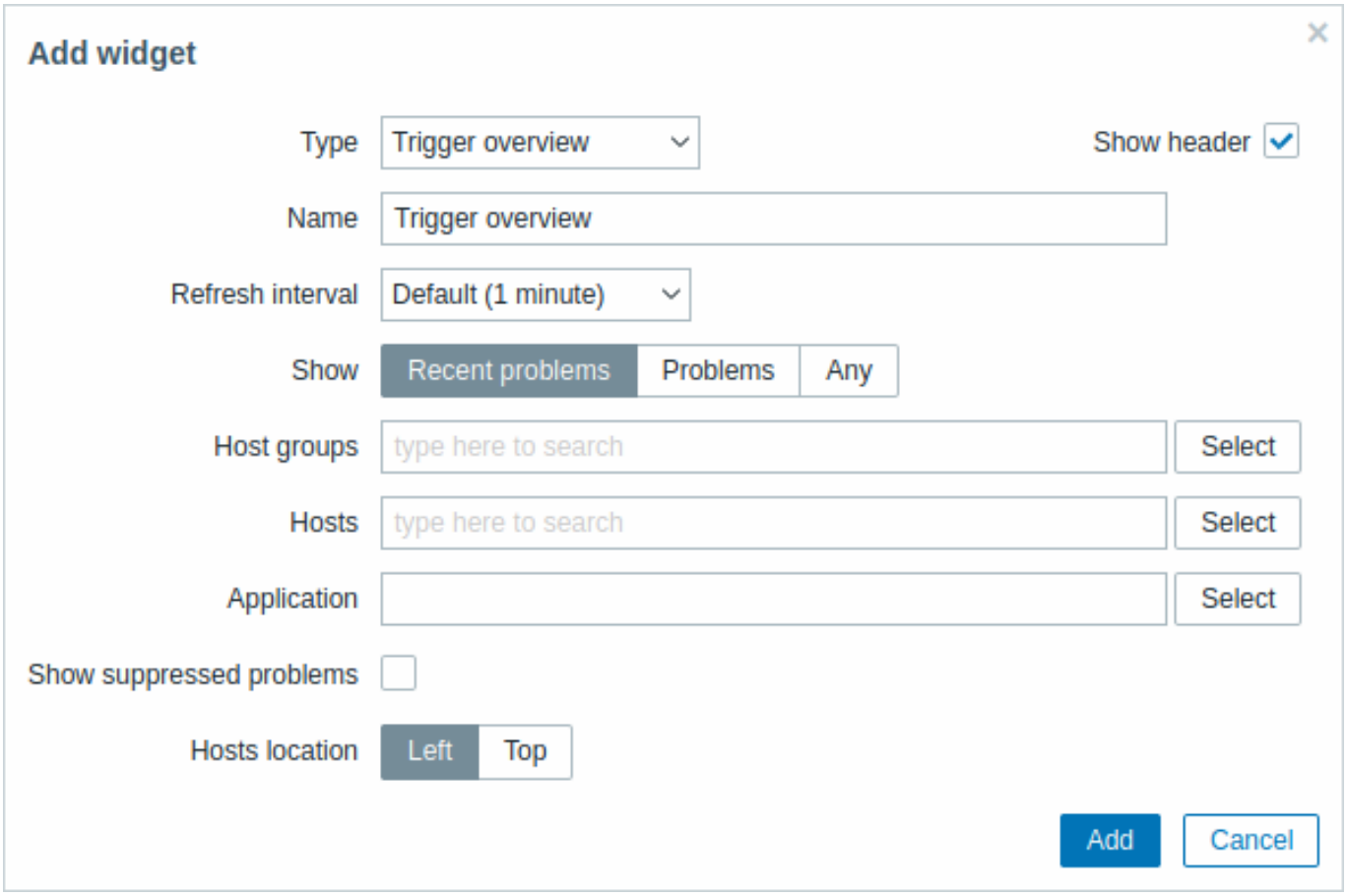
Parameter	Description
Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.
Exclude host groups	Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to *show* Group A and *exclude* Group B at the same time, only data from host 001 will be displayed in the Dashboard.
Hosts	Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.

Parameter	Description
Problem	You can limit the number of problem hosts displayed by the problem name. If you enter a string here, only those hosts with problems whose name LIKE matches the entered string will be displayed. Macros are not expanded.
Severity	Mark the problem severities to be displayed in the widget.
Show hosts in maintenance	Mark the *Show hosts in maintenance* option to display data from hosts in maintenance in the widget.
Hide groups without problems	Mark the *Hide groups without problems* option to hide data from host groups without problems in the widget.
Problem display	Display problem count as: **All** - full problem count will be displayed **Separated** - unacknowledged problem count will be displayed separated as a number of the total problem count **Unacknowledged only** - only the unacknowledged problem count will be displayed.

Trigger overview

In the trigger overview widget you can display the trigger states for a group of hosts. It replicates information from *Monitoring* → *Overview* (when *Triggers* is selected as Type there).

To configure, select *Trigger overview* as type:



You may set the following specific options:

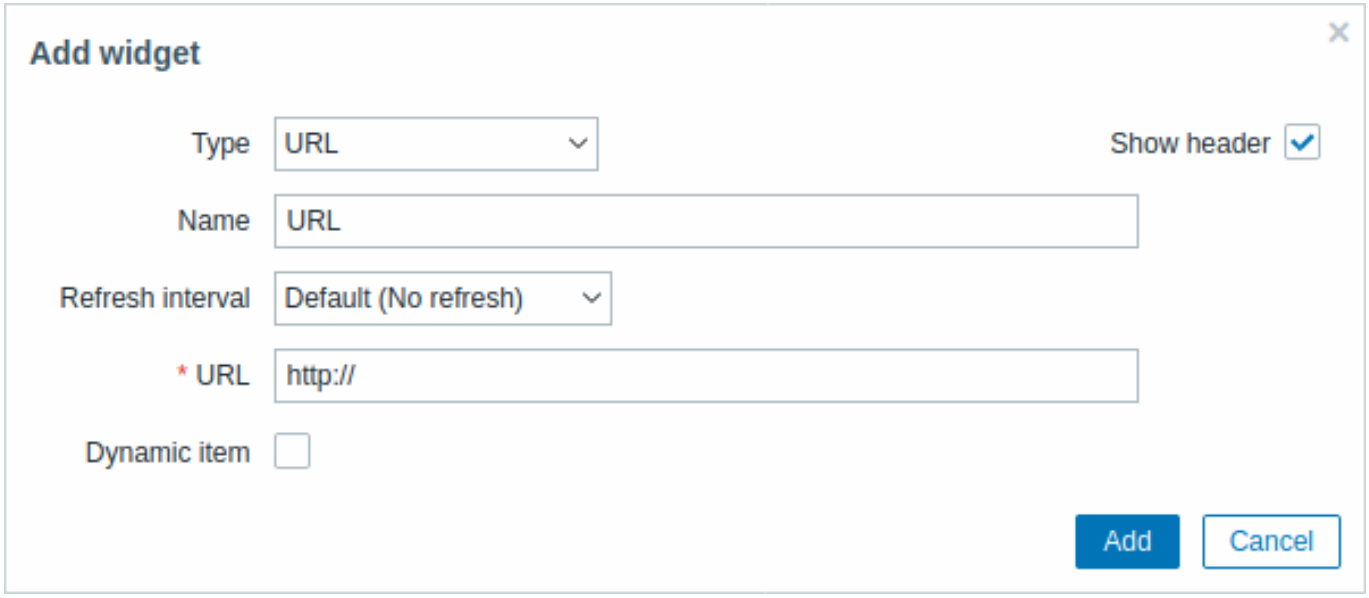
Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **Any** - history of all events is displayed

Host groups	Select the host group(s). This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups.
Application	Enter the application name.
Hosts location	Select host location - left or top.

URL

In the URL widget you can display a URL content from an external resource.

To configure, select *URL* as type:



You may set the following specific options:

URL	Enter the URL to display. The URL must start with `http://`. {HOST.*} macros are supported.
Dynamic item	Set to display different URL content depending on the selected host. This can work if {HOST.*} macros are used in the URL.

Attention:
Browsers might not load an HTTP page included in the widget, if Zabbix frontend is accessed over HTTPS.

Web monitoring

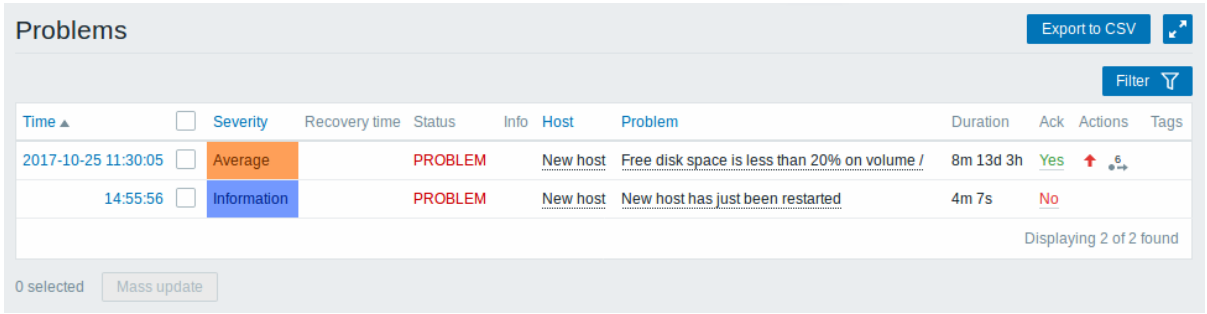
This widget displays a status summary of the active web monitoring scenarios.

Note:
In cases when a user does not have permission to access certain widget elements, that element's name will appear as *Inaccessible* during the widget's configuration. This results in *Inaccessible Item, Inaccessible Host, Inaccessible Group, Inaccessible Map and Inaccessible Graph* appearing instead of the "real" name of the element.

2 问题

简介

在 //监测中 → 问题中 // ，你可看到当前存在什么问题。问题指处在 "问题" 状态下的触发器。



参数功	说明
时间（*Time*）显示问 //严重等级 (Severity) // 显示异	开始时间。 严重等级。 显示问题恢复时间。问题严重等级取决于其触发器的严重等级。触发器严重等级的颜色用作单元背景色。已处理过的问题，其背景颜色是绿色。在问题发生后，你可以使用//“确认事件” screen 更新问题. | | 恢复时间（Recovery time）//
状态（*Status*）显示问	状态被显示为： 问题（**Problem**）- 未解决的问题 已恢复（**Resolved**） - 近期已解决问题. 你可通过使用过滤器来隐藏近期已解决问题。 新解决的和近期解决的问题会闪烁 2 分钟。已解决问题共显示 5 分钟。触发器显示时间的配置在 管理 → 通用 → 触发器显示选项 *(Trigger displaying options)*.
信息（*Info*）如果通	全局关联关闭问题或在更新问题时手动显示绿色信息图标。在图标上滚动鼠标将显示更多详细信息：
主机（*Host*）显示异	的主机。
问题（*Problem*）显示问	名称。 问题名称取决于其触发器的问题名称。
持续时间（*Duration*）显示问题持	时间 也可以参考这里: 异常问题持续时间

参数功	说明
问题确认 *(ack)* 显示问	确认状态： ** 已确认 (Yes) ** - 绿色字体表明问题已确认。如果一项问题的所有事件都已被确认，则此项问题被认为已被确认。 未确认（**NO**） - 红色链接表明有未被确认的事件。 如果你点击链接将会被带到问题确认可以对显示的问题进行简单的处置。
动作（*Actions*）使用符	标记有关问题的活动的历史记录： - 显示已经更新的描述数量信息。 -问题的告警级别提高 (例如：信息级别 → 告警级别) - 问题严重级别下降 (例如：警告 → 信息) - 问题的严重级别发生过变化，但是目前回归到初始问题级别。(例如：警告 → 信息 → 警告) - 已经触发动作，并且显示当前触发的动作数。 - 动作操作正在进行中，显示当前操作数量进度。 - 动作进行过程中至少有 1 次的动作发生失败。 当鼠标移动到图标时会显示当前的动作信息，更多内容请参见查看详情
标记（*Tags*） [时间	签](/manual/config/triggers/event_tags 显示时间标签 (如果存在)。

负的问题持续时间

在某些情况下，可能会出现具有负的持续时间，即问题解决时间早于问题创建时间，例如：

- 在使用代理收集数据的时候，发生网络错误，导致代理暂时接收不到数据。同时主机触发器里有用到 item.nodata() 时，这时此触发器会自动触发。但等到链接恢复后，代理节点重新把积累数据传送给服务器时，问题将会得到解决。并且会出现负数的问题持续

时间。；

- 当解决问题事件的项目数据由 Zabbix 发送并包含早于问题创建时间的时间戳时，还将显示负问题持续时间。

批量编辑选项

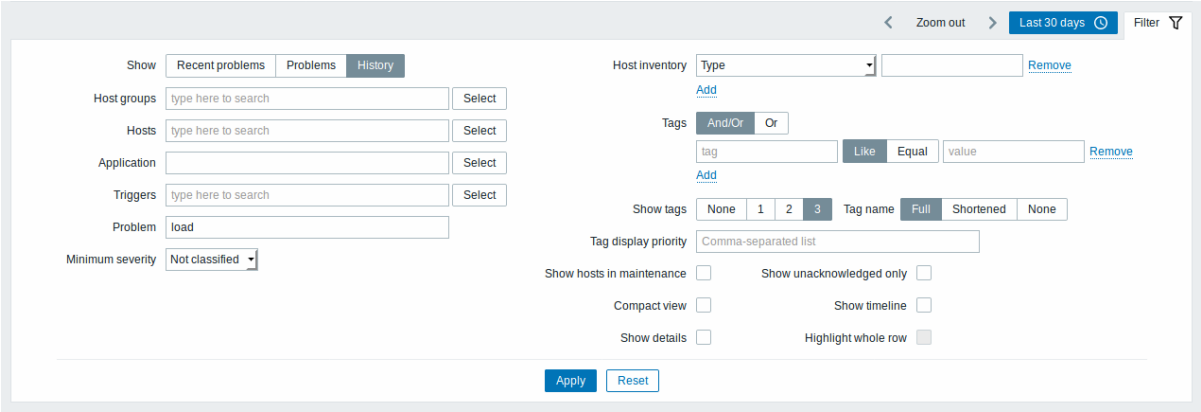
列表下方的按钮提供了一些批量编辑选项:

- 批量更新（*Mass update*） - 通过导航到问题来更新问题（problem update） 屏幕

要使用此选项，请在出现相应问题之前选中复选框，然后单击 批量更新（*Mass update*）按钮。

使用过滤器

您可以使用过滤器只显示你感兴趣的问题。过滤器位于目录上方。



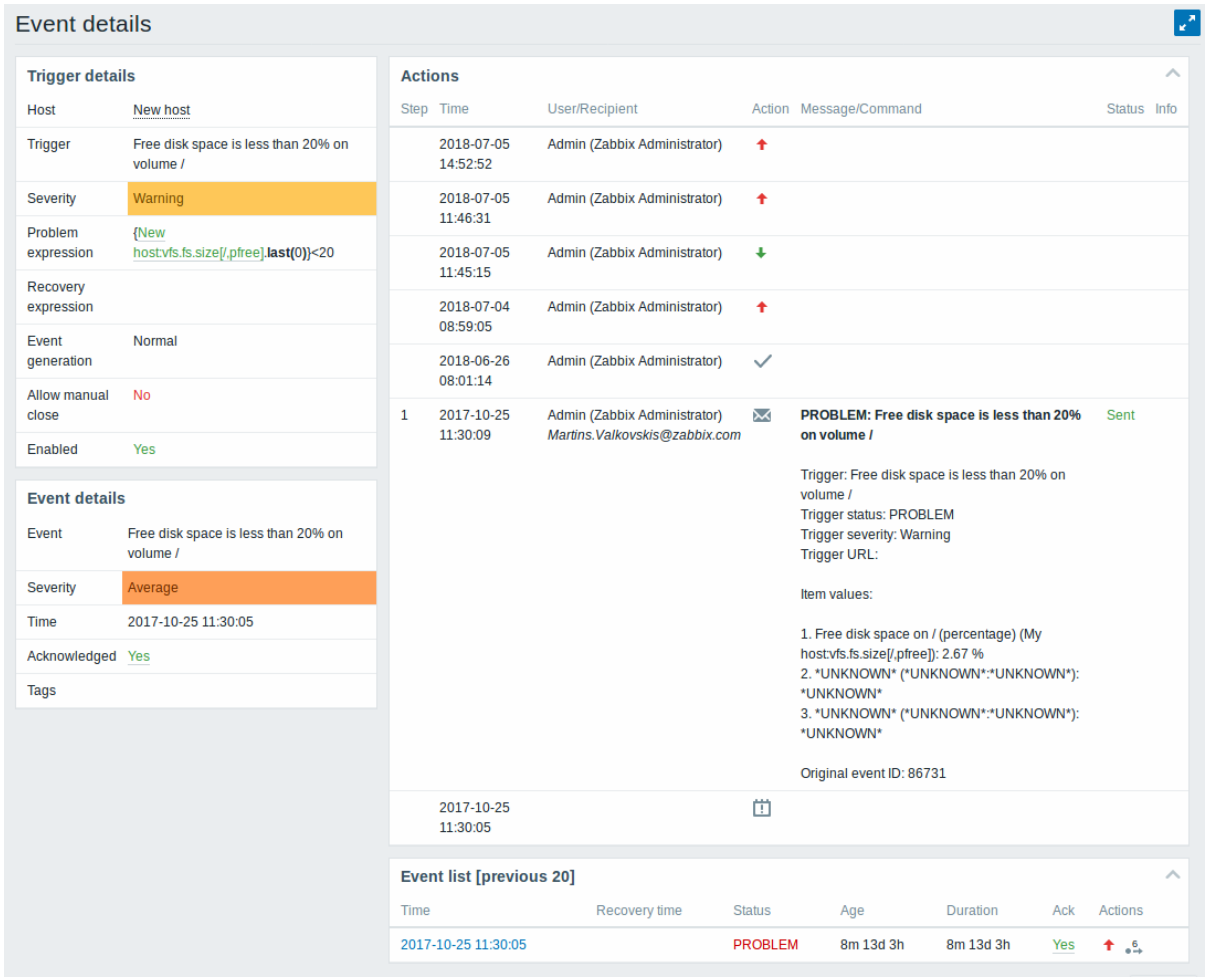
参数功	说明
//显示 (Show) // 按	题状态进行筛选： 最近的问题 **(Recent problems)** - 显示未解决以及近期已解决异常(默认) 问题 **(Problems)** - 显示未解决的问题 历史记录 **(History)** - 显示所有事件的历史记录
//主机群组 (Host Group) // 按一个	多个主机群组筛选。 指定一个父主机群组， 指定一个父主机群组， 隐式选择全部嵌套主机群组。
主机 *(Hosts)* 按	个或多个主机进行筛选。
应用集（*Application*）按应用集	称筛选。
触发器（*Triggers*）按一个或	个触发器筛选。
问题（*Problem*）按问题	称筛选。
//最低触发器严重等级 (Minimum trigger severity) // 按最低触发器严重	级筛选。

参数功	说明
//主机资产记录 (Host inventory) // 按资产记录	型和值进行筛选。
//标签 (Tags) // 按	件标签名称和值进行筛选。 可以设置多个条件，条件中可以增加判断。 和（**And**）/或者（**Or**） - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 或者（**Or**） - 满足其中一条即可。 匹配表标记值的方法有两种： （类似）**Like** - 模糊类型的字段匹配 等于（**Equal**） - 精确匹配
显示标签（*Show tags*）选择显示的	签数量： 无或空（**None**） - 没有 标签的监控问题 监测 → 问题 **1**- 标签列包含一个标签 **2**- 标签列包含两个标签 **3**- 标签栏包含三个标签 要查看问题的所有标记，请将鼠标悬停在三个点图标上。
//显示维护中的主机异常 (Show hosts in maintenance) // 标记复选框，以显示	护中的主机异常。
精简视图（*Compact view*）选中复选框	启用精简、紧凑视图。
展示详细信息（*Show details*）选中复选框以显	问题的基础触发表达式。 需要禁用精简视图（*Compact view*）。
//仅显示未确认的异常 (Show unacknowledged only) // 标记复选框，仅显	未确认的异常。

参数功	说明
时间轴显示（*Show timeline*）选中复选框以	示可视时间轴和分组。需要禁用精简视图（*Compact view*）。
整行突出显示（*Highlight whole row*）选中复选框以突	显示未解决问题的完整行。问题严重性颜色用于突出显示。仅在官方蓝色、黑色的主题中使用精简视图并启用。高对比度主题中无法突出显示整行。

查看详细信息

在 监测 → 问题异常开始和恢复的时间都有链接，单击链接可以打开更多事件细节。



触发器和问题时间的严重性是有区别的。问题事件需要到 问题确认中进行更新。细节

在操作列表中，以下图标用于表示活动类型:

- - 生成问题事件
- - 信息已发送

- - 已确认问题事件
- - 有评论添加
- - 问题严重程度已经升级 (例如：信息 → 警告)
- - 问题严重度已经下降 (e.g. 警告 → 信息)
- - 问题严重性发生变化，回到初始问题级别。(例如：(最初为）警告级别 → （降级为）信息级别 → （又升级为）警告级别)
- - 执行了远程命令
- - 问题事件已恢复
- - 问题被手动关闭

3 概述

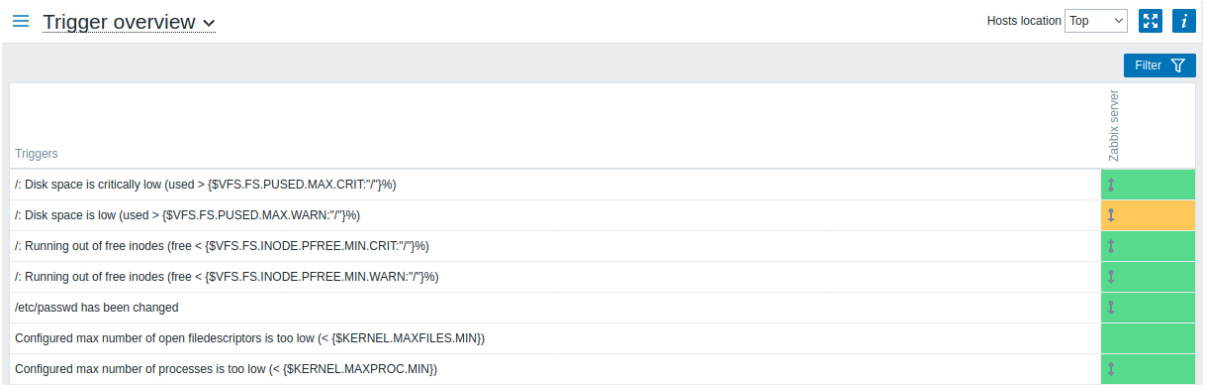
简介

在 监测 → 概述提供了总览触发器状态一个平台，或者将不同主机的数据放在一起进行比较。可用下列显示的选项：

- 在主机群组下拉菜单中选择全部或指定主机群组
- 在类型下拉菜单中选择要显示的信息类型（触发器或数据）
- 在主机位置下拉菜单中选择主机名位于表格顶端或表格左侧显示

触发器概述

在下一个屏幕截图中，在类型下拉列表中选择触发器。因此，两个本地主机的触发状态显示为彩色块，(问题触发器的颜色取决于问题严重性颜色，可以在确认问题 屏幕中调整):

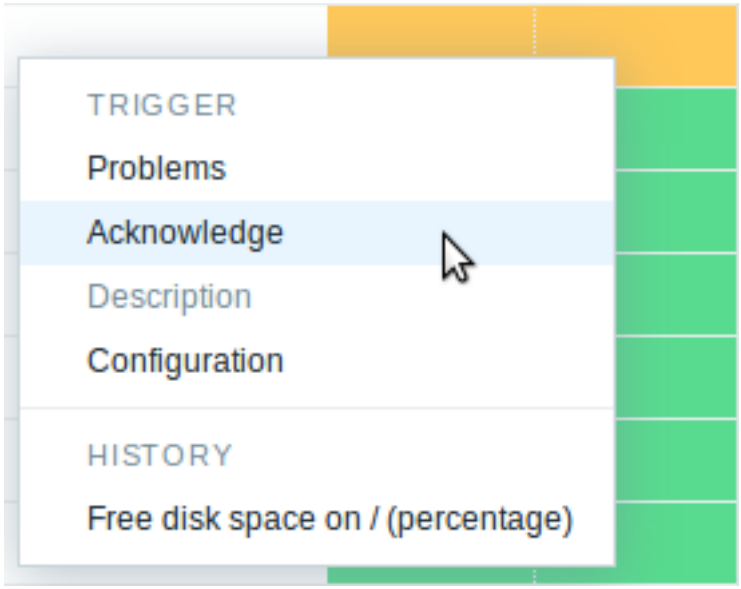


请注意。显示为闪烁块，仅显示（最后 2 分钟内）变化的触发器。

蓝色向上和向下箭头表示具有依赖关系的触发器。在鼠标悬停时，显示依赖性详细信息。

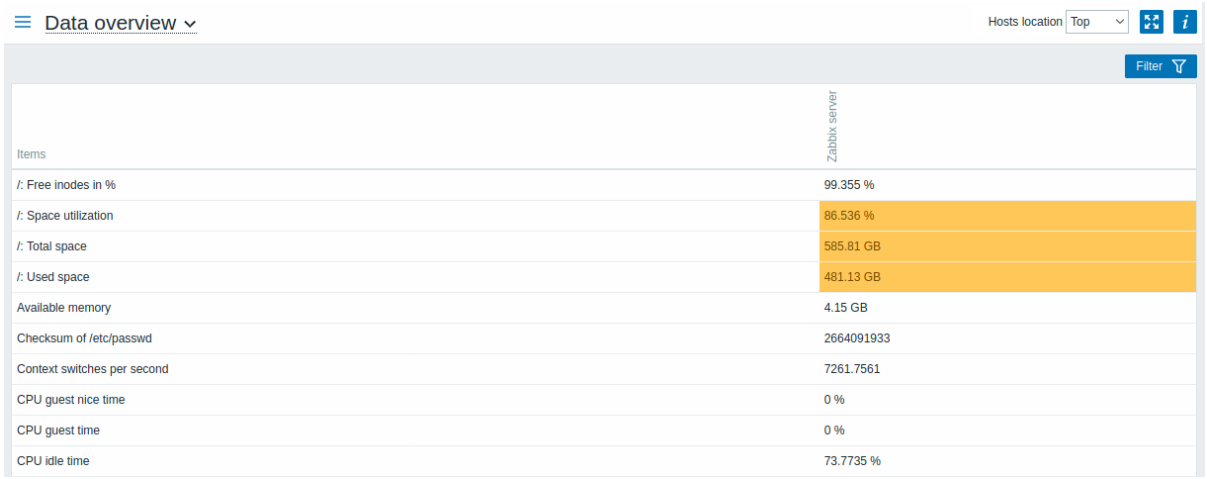
复选框图标表示已确认的问题。

单击触发器块可提供与触发器的问题事件，问题确认屏幕，触发器配置，触发器 URL 或简单图表/最新值列表的上下文相关链接。



数据概述

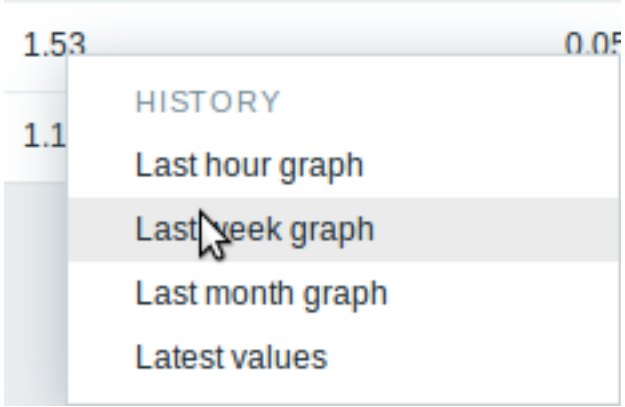
在下一个屏幕截图中，在类型下拉列表中选择数据。结果为显示两个本地主机的性能项数据。



问题项的颜色基于问题严重性颜色，可以在问题确认 中调整.

默认情况下仅显示最近 24 小时内的数据。这样设置是为了优化页面加载数据的时间。如果你想查看更多数据也可以到前端文件 *include/defines.inc.php* 中更改 ZBX_HISTORY_PERIOD 的常量值 。

单击一条数据可提供指向某些预定义图形或最新值的链接。

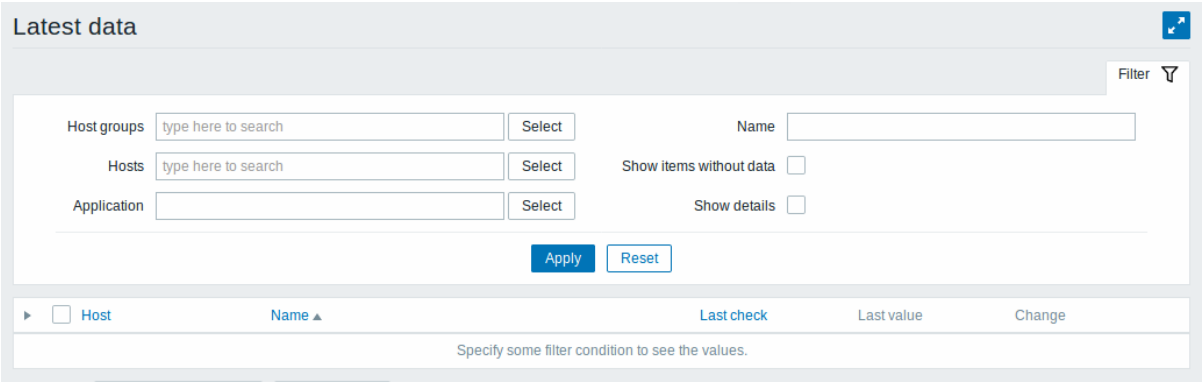


5 最新数据

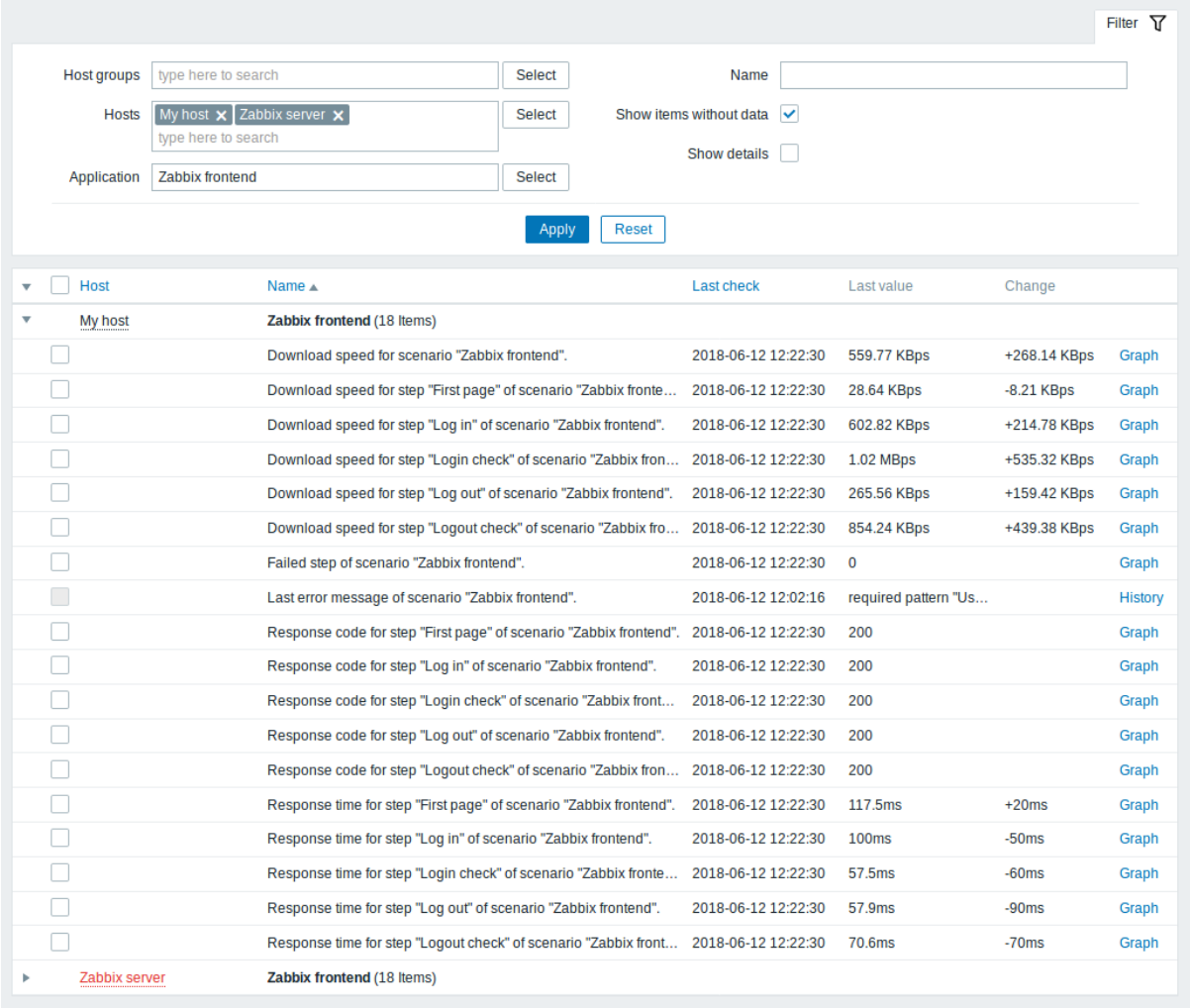
简介

监测 → 最新数据可以用来查看监控项收集的最新值，以及访问各种项目图表。

第一次打开此页面时，不显示任何内容。



要访问数据，您需要在过滤器中进行选择，例如主机组，主机，应用程序或项目名称。



在显示的列表中，单击 ▶ 在主机和相关应用程序之前显示该主机和应用程序的最新值。您可以展开所有主机和所有应用程序，从而通过单击显示所有项目 ▶ 在标题行中。

注意：被禁用主机的名称显示为红色。从 Zabbix2.2.0 起，被禁用主机数据（包括图表和项目值列表）在最新数据页可以被访问。

本页面的列表针对监控项展示以下列：监控项名称、最近检查记录、最后一个值、更改量 (Change) 以及一个跳转到项目值的简单图表/历史记录的链接。

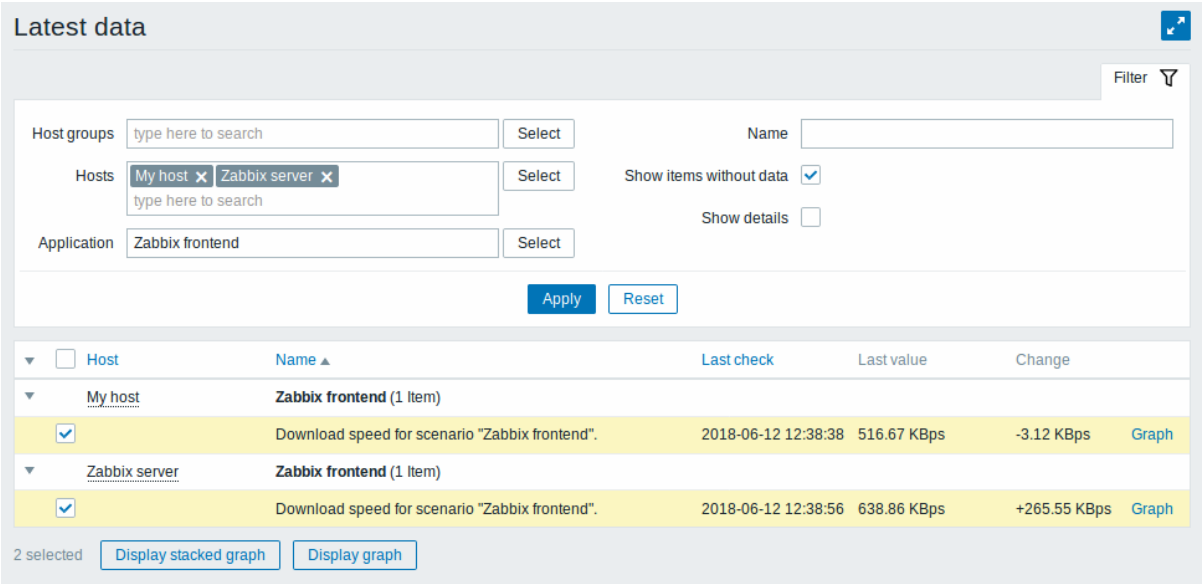
默认情况下仅显示最近 24 小时内的数据。这样设置是为了优化页面加载数据的时间。如果你想查看更多数据也可以到前端文件 *include/defines.inc.php* 中更改 ZBX_HISTORY_PERIOD 的常量值 。

使用筛选器

您可以使用筛选器只显示您感兴趣的监控项。筛选器链接位于表格上方中部。您可以使用它来过滤主机群组、主机、应用集、取自监控项名称中的字符串；还可以选择是否显示没有收集到数据的项目（查看无资料项目）。

指定一个父主机组，隐式选择所有嵌套的主机组。

显示详细信息会增加显示监控项相关的以下项目：该监控项的键值、间隔设置、历史记录及趋势的保存时间设置、监控项的类型和监控项的错误（良好/不支持）等详细信息，同时键值是一个链接到监控项配置的超链接。



默认情况下，会显示没有数据的项目，但不显示详细内容。

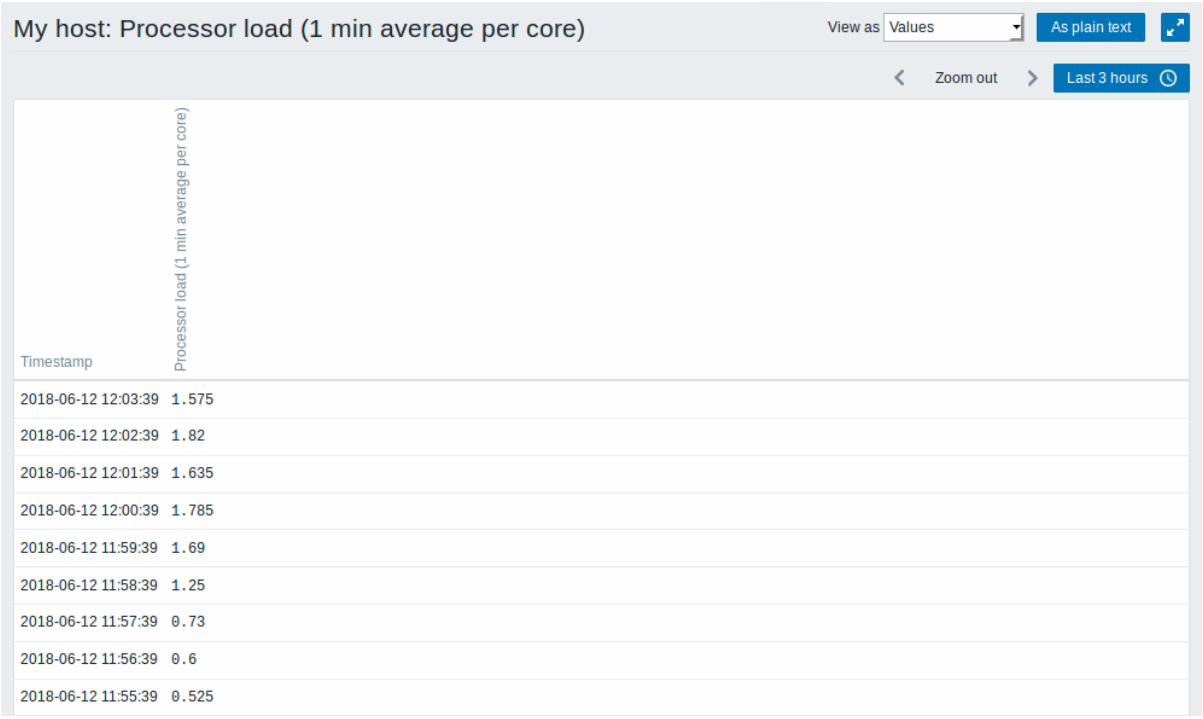
比较项目图表

可以在第二列的复选框选择几个监控，然后用简单图形或堆叠图比较它们的数据。选择感兴趣的监控项，然后单击表下所需图形的按钮，即可查看图形。

链接到值的历史/简单图形

提供最新值列表中的最后一列：

- 一个 ** 历史链接 ** （用于所有文本项）-链接到的列表（values/ 500 个最新 values）显示前一个项目值的历史记录。
- 一个 图表连接（用于所有数字项）-链接到一个简单图形。图形被调用出来后，从右上角的下拉框也可以切换为显示值 (values) 或最新 *500* 个值 (500 latest values)。



此列表中显示的是“原始的”值，即指未经处理的指。

Note:
显示的总值由“搜索限制”和“过滤结果”参数值定义，在管理 → 一般中设置。

7 聚合图形

Overview

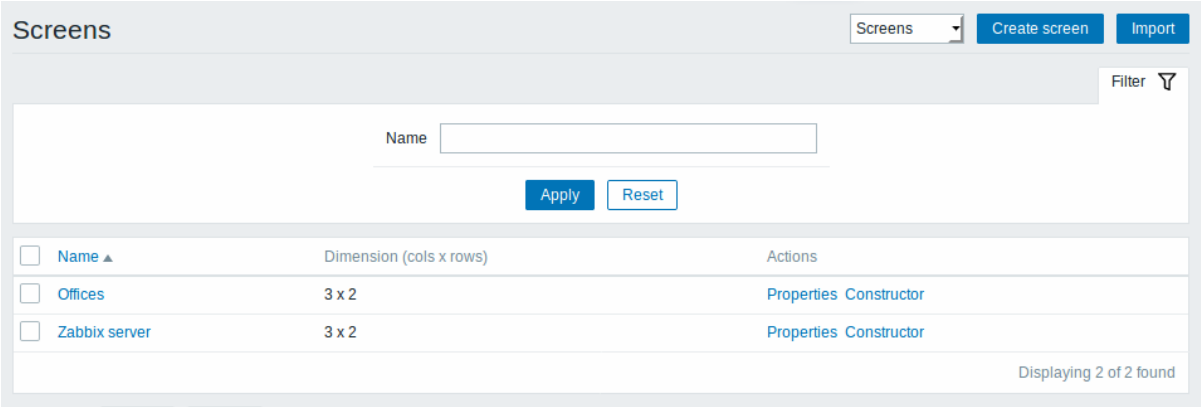
进入方法 监测 → 聚合图形您可以配置，管理和查看 Zabbix 聚合图形 (screens) 以及幻灯片演示 (slide shows)。

当您打开此部分时，您将看到您访问的最后一个聚合图形/幻灯片演示或您可以访问的所有实体的列表。聚合图形/幻灯片演示列表可以按名称过滤。

Zabbix 3.0 之后的版本中，所有的聚合图形/幻灯片演示可以是公共的或私人的. 公共的可用于所有用户，而私有的用户只能由其所有者和实体共享的用户访问。

使用标题栏中的下拉菜单在聚合图形/幻灯片演示之间切换。

聚合图形列表



展示数据:

参数功	说明
名称（*Name*）聚合图	的名称. 点击名称查看对应的聚合图形。
尺寸（*Dimensions*）聚合图	的列数和行数。
操作（*Actions*）目前仅	持两种操作方式：属性 - 编辑一般// 聚合图形 // 的属性 (名称和尺寸)\\构造函数 - 访问网格化的聚合图形元素来做编辑

如果要创建新的聚合图形, 点击屏幕右上角的创建聚合图形（*Create screen*）。要从 XML 文件导入屏幕，请单击右上角的导入按钮。导入聚合图形的用户将被设置为其所有者。

批量编辑选项

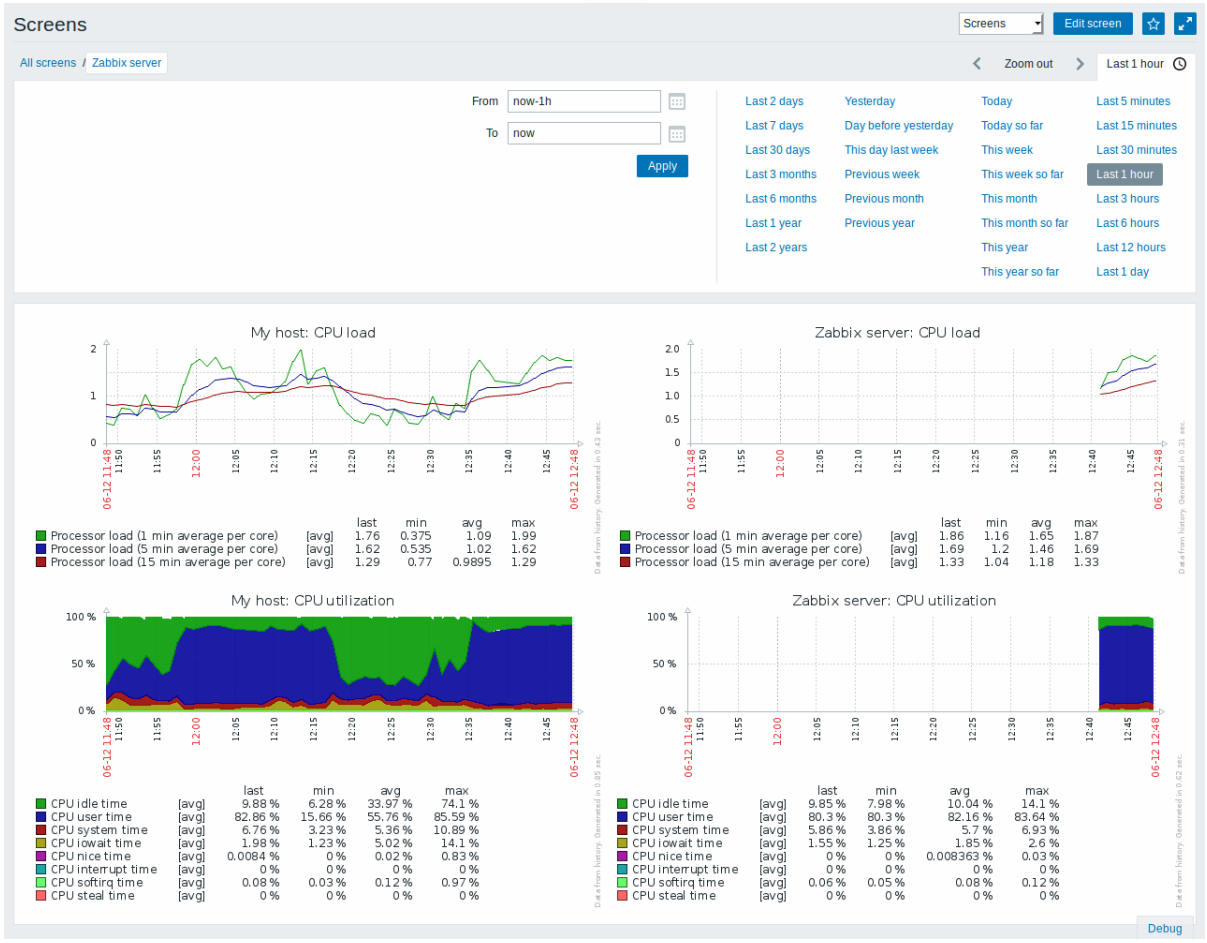
列表下方的按钮提供了一些批量编辑选项：

- 导出（*Export*） - 将/聚合图形导出到 XML 文件
- 删除（*Delete*） - 删除聚合图形

要使用这些选项，请在相应聚合图形之前标记复选框，然后单击所需的按钮。

查看聚合图形

要查看聚合图形，请在所有聚合图形列表中单击其名称。



时间选择器

聚合图形上方的过滤器部分包含时间段选择器。它允许您轻松选择所需的时间段，影响图形中显示的数据等。

更多参见: 时间选择器

控制

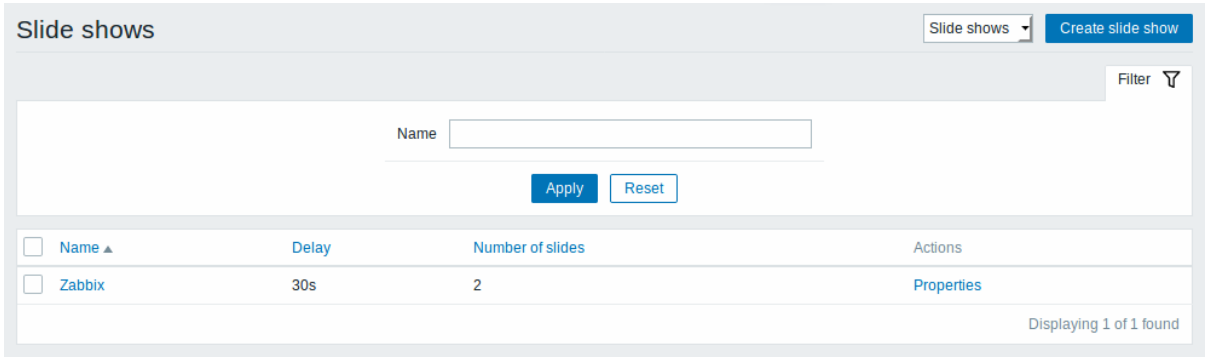
标题栏中有三个控制按钮：

-

- 去聚合图形构造函数编辑聚合图形
- - 把聚合图形添加到仪表板中
- - 使用浏览器窗口全屏展示聚合图形

幻灯片演示列表

使用标题栏中的下拉菜单从聚合图形切换到幻灯片演示。



显示数据：

参数功	说明
名称（*Name*）幻灯片	示的名称. 点击名称以幻灯片方式演示 。
翻页时间（*Delay*）每个幻灯片	示 & 翻页的间隔时间。
幻灯片数量（*Number of slides*）幻灯片的数量	页数。
操作（*Actions*）仅支持	个操作式: 属性 - 编辑幻灯片属性

创建一个新的幻灯片，点击右上角的 创建幻灯片按钮。

批量操作选项

列表下方的按钮提供了一个批量编辑选项：

- 删除（*Delete*） - 批量删除演示幻灯片

要使用此选项，请在相应幻灯片显示之前标记复选框，然后单击 删除（*Delete*）。

查看幻灯片

要查看幻灯片演示，请在所有幻灯片演示列表中单击其名称。

控件

标题栏中有四个控制按钮：

-

 - 幻灯片演示属性编辑
- - 将幻灯片演示添加至仪表板
- - 浏览器全屏显示幻灯片演示
- - 放慢或者加快幻灯片演示播放速度

引用聚合图形

聚合图形可以由’elementid 和 screenname”GET 参数引用。例如，

```
http://zabbix/zabbix/screens.php?screenname=Zabbix%20server
```

将使用该名称打开聚合图形（Zabbix 服务器）。

如果同时指定 “elementid”（聚合图形 ID）和 “screenname”（聚合图形名），则 “screenname”（聚合图形名）具有较高的优先级。

8 拓扑图

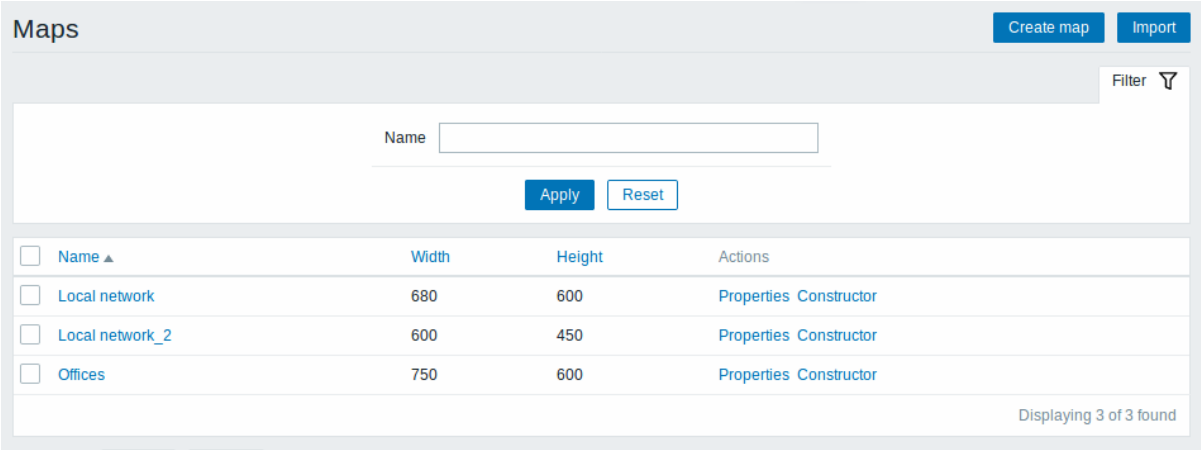
简介

进入方法 监测 → 拓扑图您可以配置，管理和查看拓扑图。

当您打开此部分时，您将看到您访问的最后一张拓扑图或您可以访问的所有拓扑图的列表。拓扑图列表可以按名称过滤。

自 Zabbix 3.0 以来，所有拓扑图都可以是公共的或私有的。所有用户都可以使用公共拓扑图，而私人拓扑图只能由其所有者和拓扑图共享的用户访问。

拓扑图列表



配置信息：

参数功	说明
名称（*Name*）拓扑图	名称. 点击名称查看 对应的拓扑图.
宽度（*Width*）显示拓	图宽度
高度（*Height*）显示拓	图的高度
操作（*Actions*）两项操	可做: 属性 - 编辑拓扑图整体属性 结构 - 访问网格化的拓扑图元素来编辑

配置创建新的拓扑图, 点击右上角的// 创建拓扑图按钮。要从 *XML* 文件导入拓扑图，请单击右上角的导入//按钮。导入拓扑图的用户将被设置为其所有者。

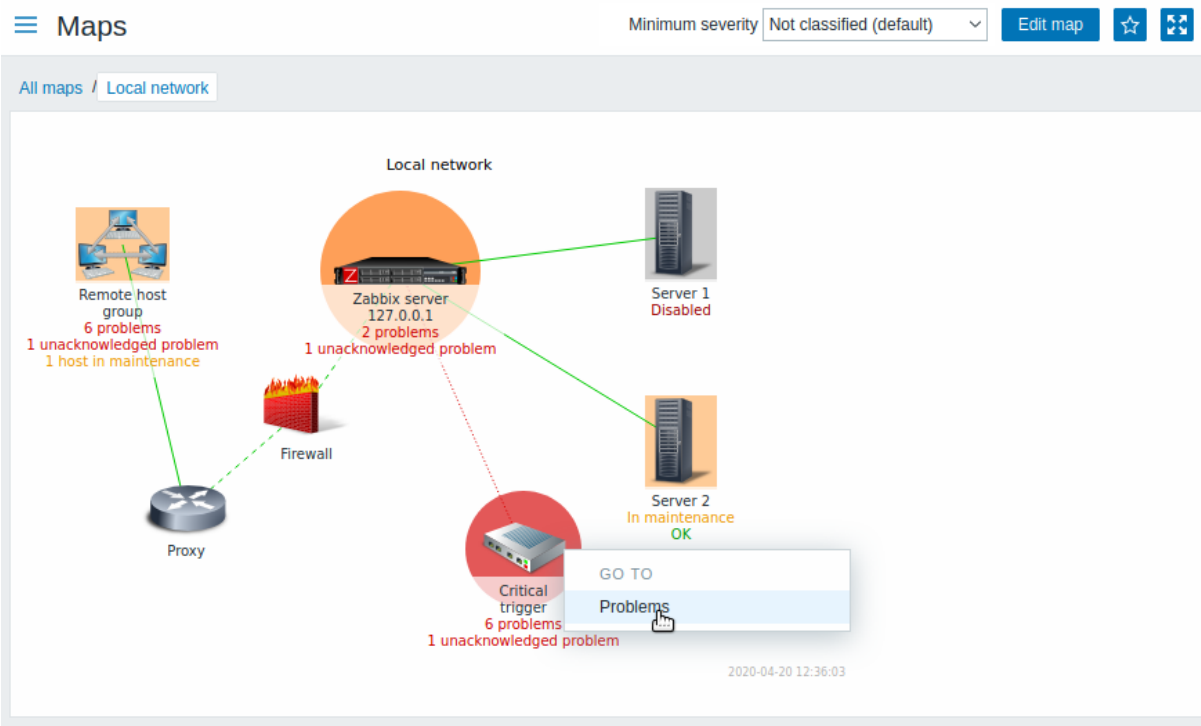
列表下方的两个按钮提供了一些批量编辑选项：

- 导出（*Export*） - 将拓扑图导出为 XML 文件
- 删除（*Delete*） - 删除拓扑图

要使用这些选项，请在各个拓扑图之前标记复选框，然后单击所需的按钮。

查看拓扑图

要查看拓扑图，请在所有拓扑图列表中单击其名称。



您可以使用拓扑图标题栏中的下拉列表来选择要显示的问题触发器的最低严重性级别。标记为 default 的严重性是映射配置中设置的级别。如果拓扑图包含子拓扑图，则导航到子拓扑图将保留较高级别的拓扑图严重性。

图标突出显示

如果一个拓扑图元素处于问题状态，则以圆圈突出显示。圆的填充颜色对应于问题触发器的严重性颜色。所选严重性级别以上的问题只会与元素一起显示。如果所有问题都得到承认，则会显示圆圈周围的粗绿色边框。另外，如果一个主机在维护状态，则突出显示橙色的填充方块，禁用（未监视）主机以灰色突出显示，填充方块和着重显示只有在图标突出显示复选框被标记在拓扑图的配置中时，才会显示。

最近更改的标记

向内指向元素周围的红色三角形表示最近的触发状态变化 -最近 30 分钟内发生的最近的触发状态更改。如果触发器状态上的标记元素更改复选框在拓扑图配置中标记，则会显示这些三角形。

链接

点击拓扑图元素会打开一些包含一些可用链接的菜单。

控件

标题栏中有三个控制按钮：

-

- 转到拓扑图构造器来编辑地图内容
- - 添加拓扑图到仪表板的小构件功能中
- - 全屏展示拓扑图

Readable summary in maps

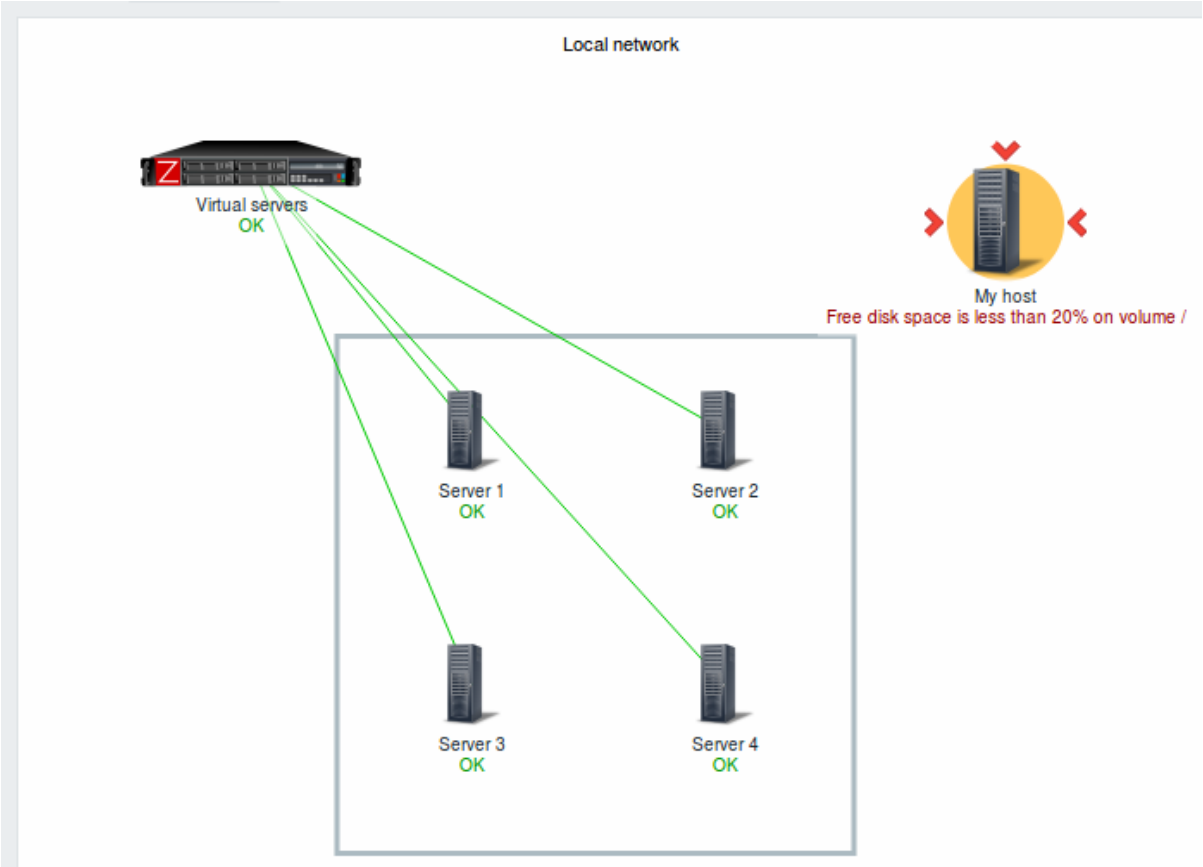
A hidden ”aria-label” property is available allowing map information to be read with a screen reader. Both general map description and individual element description is available, in the following format:

- for map description: `<Map name>, <* of * items in problem state>, <* problems in total>.`
- for describing one element with one problem: `<Element type>, Status <Element status>, <Element name>, <Problem description>.`
- for describing one element with multiple problems: `<Element type>, Status <Element status>, <Element name>, <* problems>.`
- for describing one element without problems: `<Element type>, Status <Element status>, <Element name>.`

For example, this description is available:

```
'Local network, 1 of 6 elements in problem state, 1 problem in total. Host, Status problem, My host, Free
```

for the following map:



Referencing a network map

Network maps can be referenced by both `sysmapid` and `mapname` GET parameters. For example,

`http://zabbix/zabbix/maps.php?mapname=Local%20network`

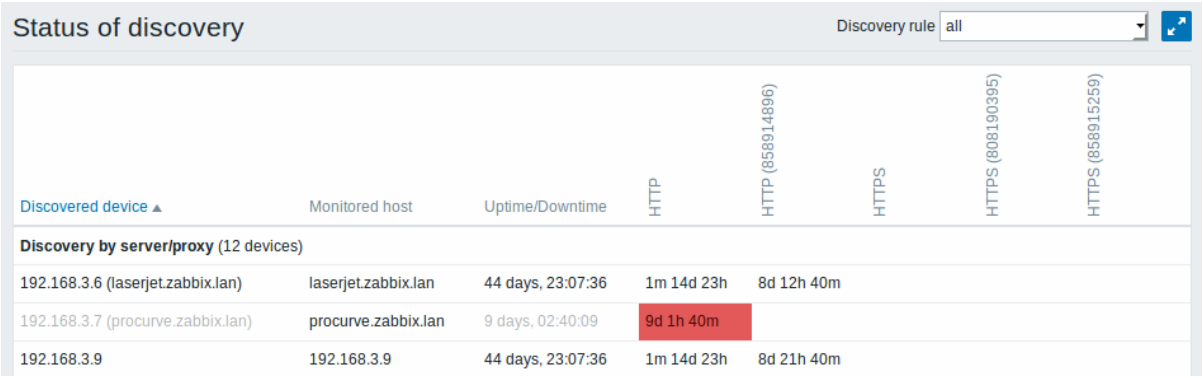
will open the map with that name (Local network).

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

9 自动发现

简介

进入方法 检测 → 自动发现显示了网络发现的部分结果。发现的设备按发现规则排序。



如果设备已被监控，主机名将列在监控的主机列中，并且在上次发现后发现或丢失的设备的持续时间显示在正常运行/停机时间列中。

随后，显示每个发现的设备的各个服务状态的列。每个单元格的工具提示将显示单独的服务正常运行时间或停机时间。

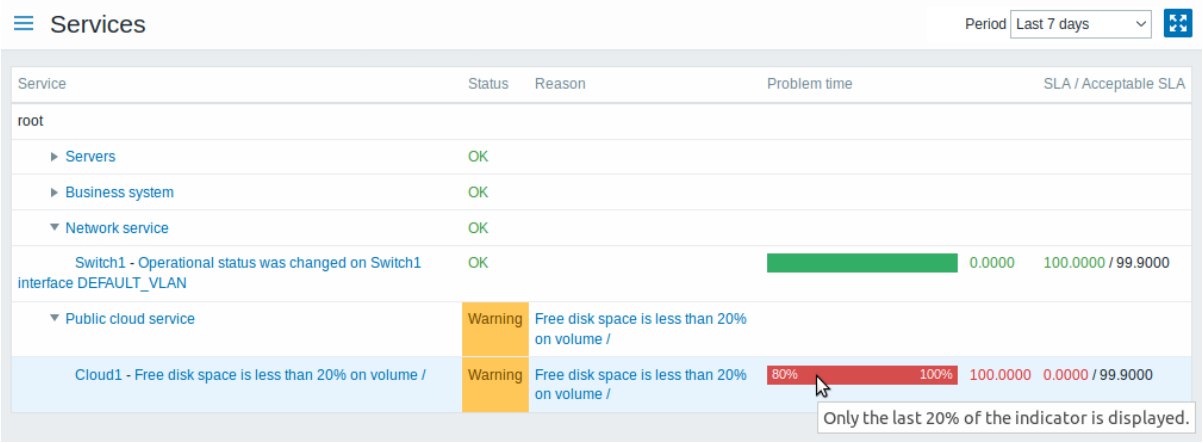
<note important> 只有在至少一台设备上找到的那些服务才会有一列显示其状态的列。

:::

11 IT 服务

简介

在 “检测中 - > IT 服务” 部分中，显示IT 服务的状态。

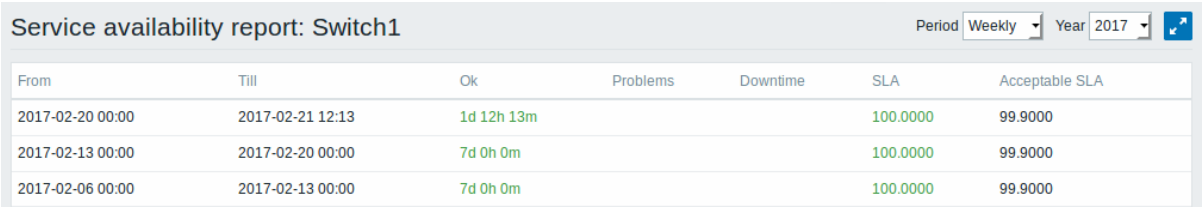


显示现有 IT 服务的列表以及其状态和 SLA 的数据。从右上角的下拉菜单中，您可以选择所需的显示周期。

显示数据：

参数功	说明
Service	服务名称。
Status	服务状态: **OK** - 正常 **(**触发颜色和严重性**)** - 表示一个问题及其严重性
Reason	表示问题的原因（如果有）.
Problem time	显示 SLA 栏. 绿/红比表示可用性/问题的比例。 显示栏显示 SLA 的最后 20 %（从 80 %到 100 %)。 该栏包含可用性数据图表的链接。
SLA/可接受的 *SLA* 显示当	SLA /预期 SLA 值。如果当前值低于可接受水平，则显示为红色。

您还可以单击服务名称以访问// IT 服务可用性报告//。



在这里，您可以在更长的时间内按日/每周/每月/每年评估 IT 服务可用性数据

2 库存

概述

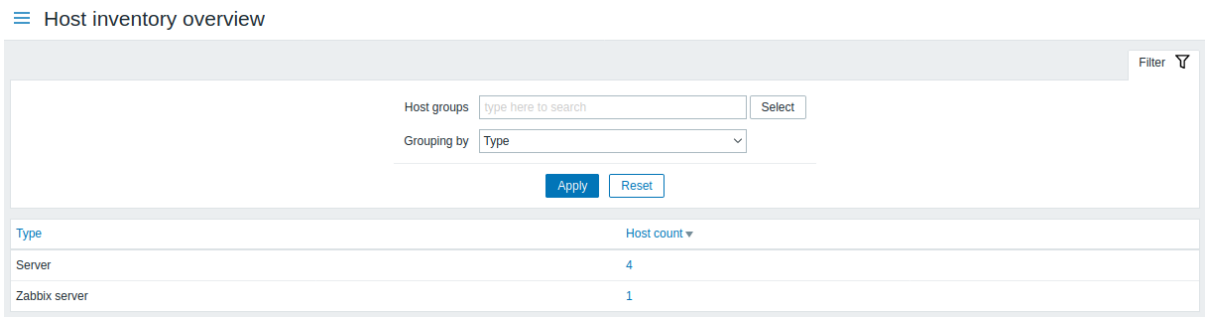
“库存” 菜单具有部分，根据所选参数概述主机库存数据，以及查看主机库存明细的功能。

1 概述

概述

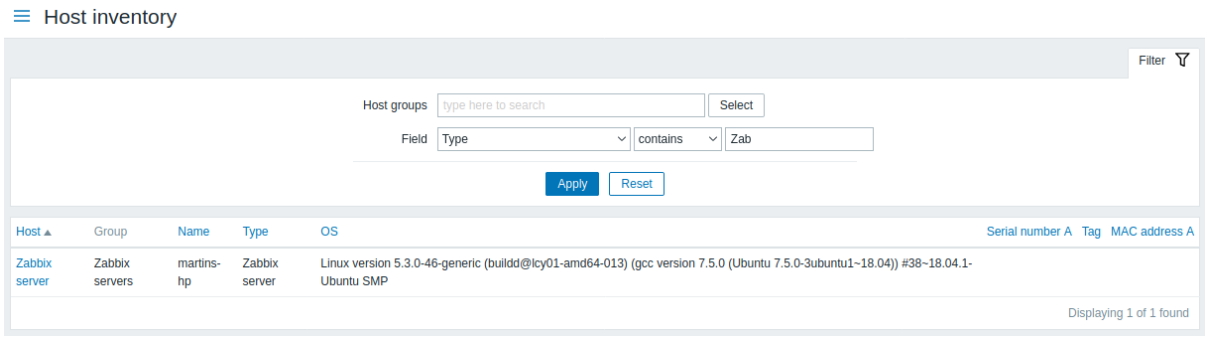
库存 **-** > 概述部分提供了有关主机盘点数据概述的方法。

要显示的概述，请选择一个主机组（或所有组）和显示数据的库存字段。将显示与所选字段的每个条目相对应的主机数量。



概述的完整性取决于主机维护多少库存信息。

主机数列中的数字是链接; 它们导致这些主机在主机库存表中被过滤掉。

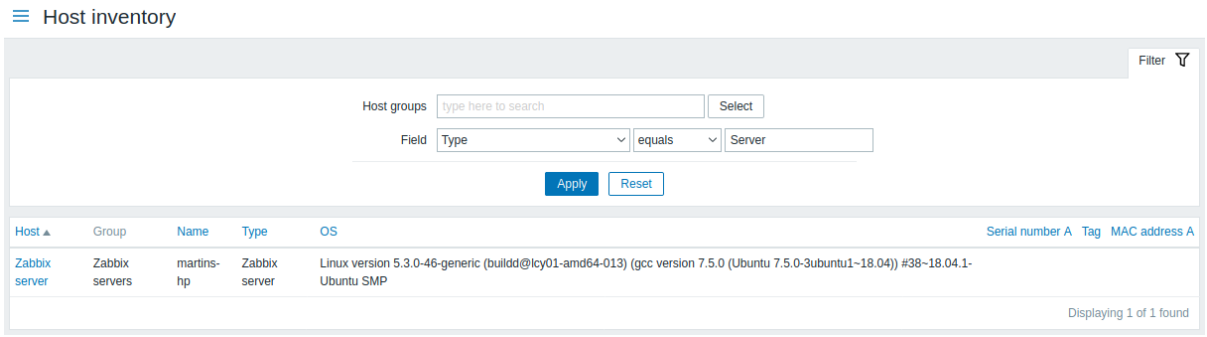


2 主机

简介

进入方法 资产记录 → 主机，资产记录信息 。

通过右上角的下拉菜单选择要查看的主机组信息，当然你也可以通过页面上方中部的关键词过滤来细化操作。

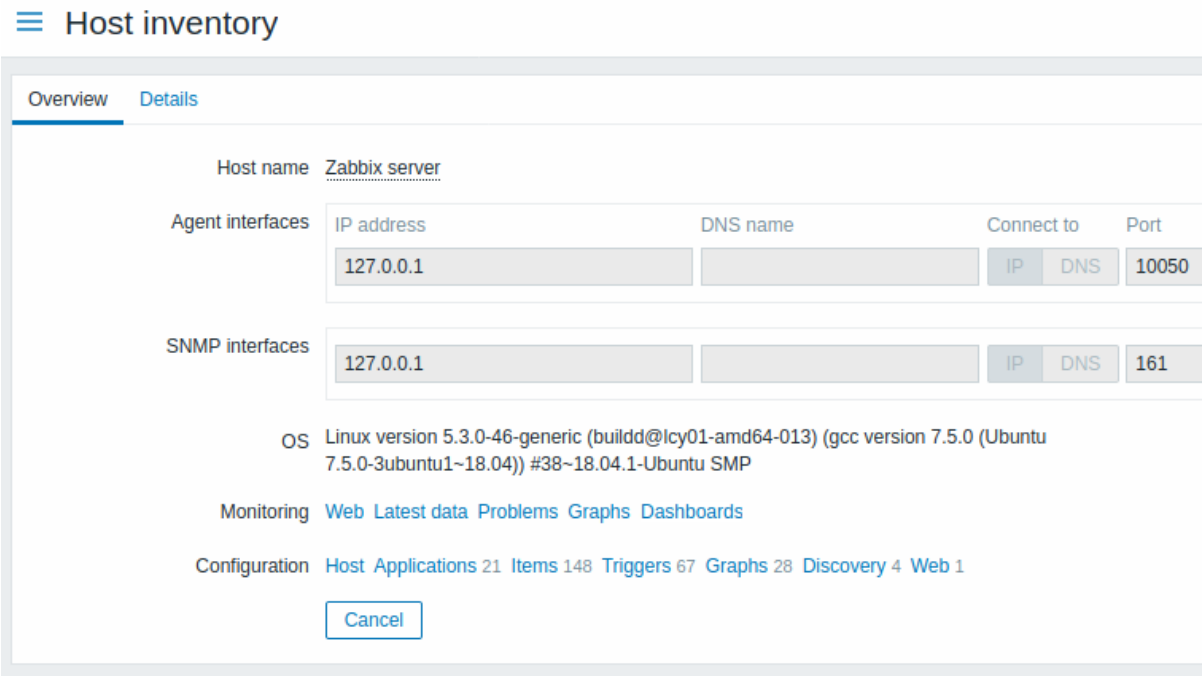


要显示所有主机清单，在组下拉列表中选择全部，清除过滤器中的比较字段，然后按过滤器。

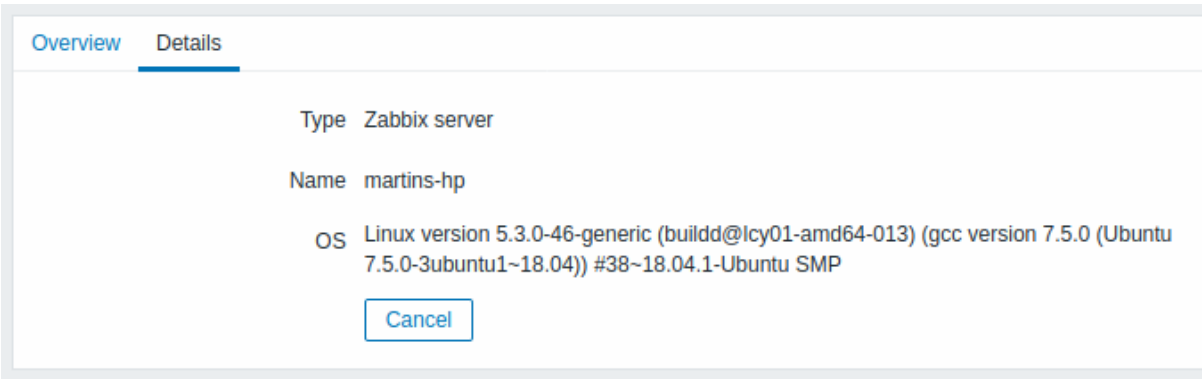
虽然表中只显示了一些关键的库存字段，但您也可以查看该主机的所有可用库存信息。如果想这么查看请单击列表中第一个主机的名字。

资产详情

在 概览选项卡包含有关主机的一些一般信息以及预定义脚本的链接，最新的监视数据和主机配置选项：



在 明细选项卡包含主机的所有可用库存明细：



资产数据的完整性取决于与主机保持多少库存信息。如果没有维护信息，则详细信息禁用。

3 报告

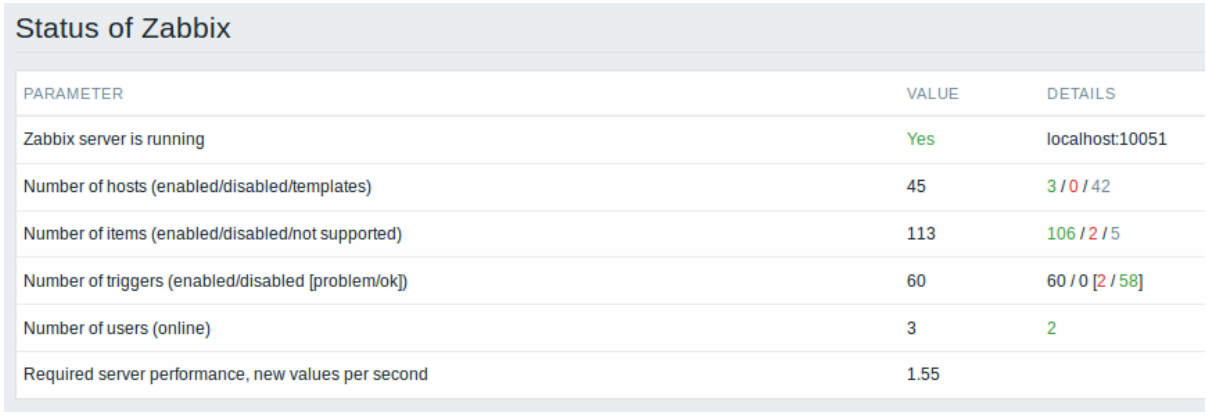
简介

“报告” 菜单包含多个部分，其中包含各种预定义和用户可自定义的报告，这些报告侧重于显示系统信息，触发器和收集数据等参数的概括。

1 系统信息

概述

在报表 **-** > *Zabbix* 的状态显示关键系统数据的摘要。



此报告也显示为仪表板中的小构件

显示数据

参数值	功能	绍
Zabbix 服务器正在运行（*Zabbix server is running*） Zabbix 服务	的状态：Zabbix 服务器的位置和端口 **Yes** - 服务器正在运行 **No** - 服务器没有运行 *Note:* 为了确保 Web 前端知道服务器正在运行，服务器上必须至少有一个 Trapper 进程（zabbix_server.conf文件中的 StartTrappers 参数）	

参数值	功能 绍
主机数量（*Number of hosts*）显示配置的	机总数。\\模板也算作主机类型。受监控主机数量/未监控主机/模板数。
监控项的个数（*Number of items*）显示监控项总数	仅指的是分配来启用主机的项目。受监控/禁用/不受支持的项目数。
触发器数量（*Number of triggers*）显示触发总数	只有分配给启用的主机和根据启用的项目的触发计数。启用/禁用触发器。[触发问题/ok 状态.]
用户数量（*Number of users*）显示配置的	户总数。N 在线人数。

参数值	功能	绍
所需的服务器性能，每秒新的值（*Required server performance, new values per second*）显示 Zabbix 服务器每秒处理	新值的预期数量。所需服务器性能一个估计值，可以作为指导。要精确处理的数值，请	用’zabbix [wcache，val- ues，all]” \\计算中包含受监控主机的启用项目。日志项目被计为每个项目更新间隔的一个值.定期间隔值被计数;灵活和调度间隔值不是。“节点”维护期间的计

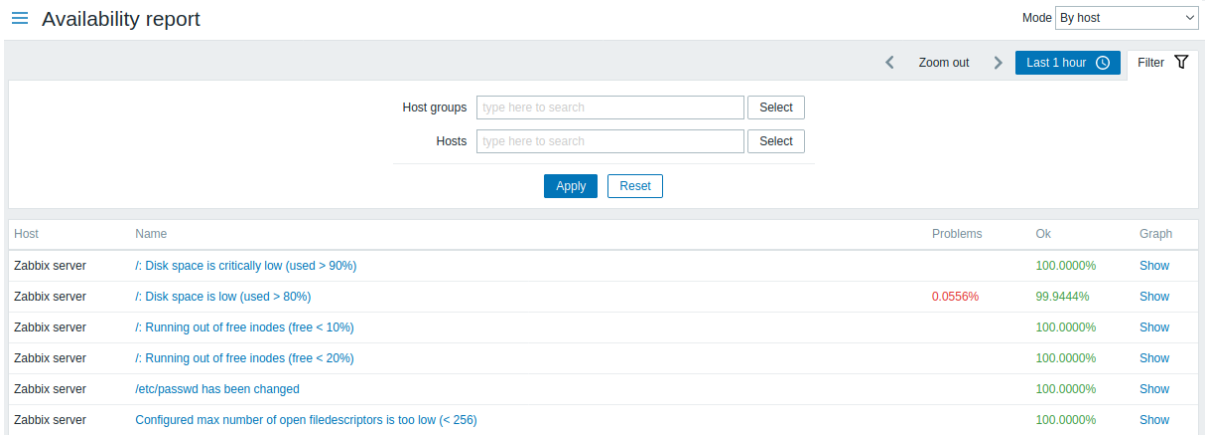
参数值	功能	绍

2 可用性报告

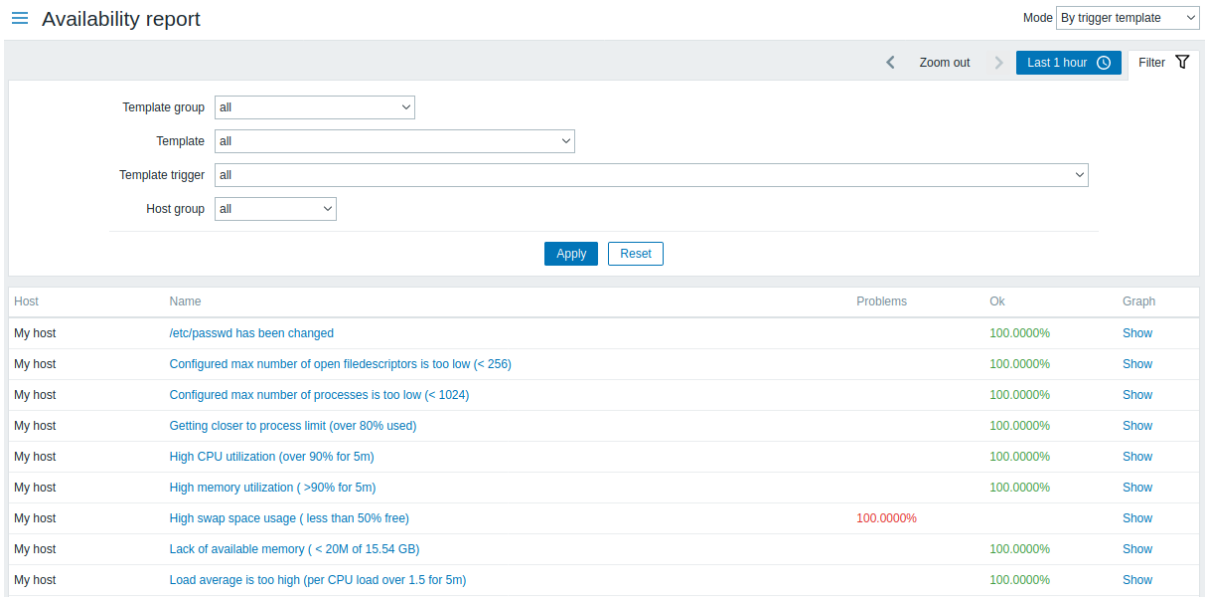
简介

在报表 - > 可用性报表中，您可以看到每个触发器在问题/状态中的时间比例。显示每个状态的时间百分比。

因此，很容易确定系统上各种元素的可用性情况。



从右上角的下拉菜单中，您可以选择选择模式 - 是否显示主机或属于模板的触发器。然后在过滤器中，您可以将选择范围缩小到所需的选项和时间段。



触发器的名称是指向该触发器的最新事件的链接。

过滤器

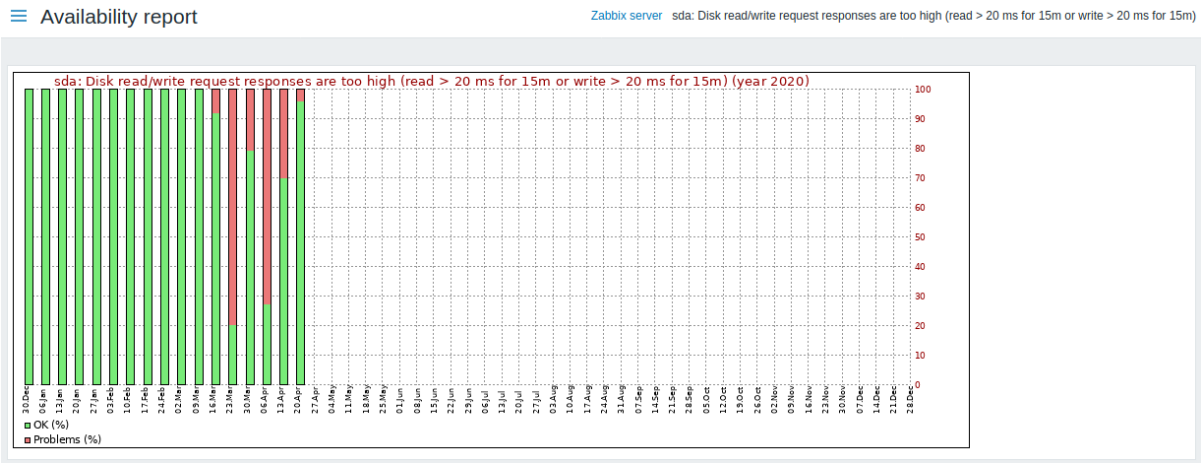
您可以使用过滤器来缩小选择范围。指定父主机组会连同选择所有嵌套的主机组。

过滤器位于可用性报表栏下方。可以通过单击左侧的// 过滤 //选项卡打开和折叠它。

时间选择器

时间选择器 允许通过单击鼠标选择经常需要的时间段。单击过滤器旁边的时间段选项卡可以打开时间段选择器。

点击 “图形” 列中的显示显示一个条形图，其中可用性信息以条形格式显示，表示当前年份过去一周的每个条。

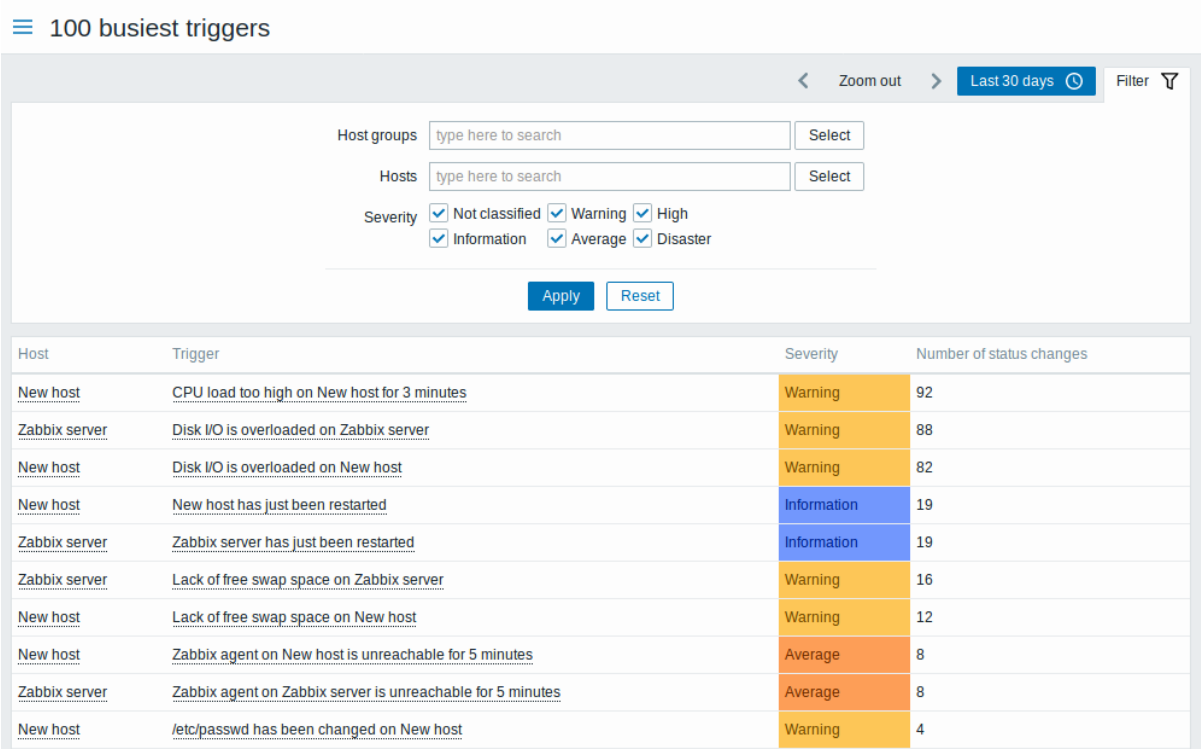


绿色部分代表 OK 时间，红色表示异常时间。

3 触发器前 **100**

简介

进入方法 报表 → 触发器 *top100* 您可以看到在评估期间最多次更改其状态的触发器，按状态更改次数排序。



Host	Trigger	Severity	Number of status changes
New host	CPU load too high on New host for 3 minutes	Warning	92
Zabbix server	Disk I/O is overloaded on Zabbix server	Warning	88
New host	Disk I/O is overloaded on New host	Warning	82
New host	New host has just been restarted	Information	19
Zabbix server	Zabbix server has just been restarted	Information	19
Zabbix server	Lack of free swap space on Zabbix server	Warning	16
New host	Lack of free swap space on New host	Warning	12
New host	Zabbix agent on New host is unreachable for 5 minutes	Average	8
Zabbix server	Zabbix agent on Zabbix server is unreachable for 5 minutes	Average	8
New host	/etc/passwd has been changed on New host	Warning	4

主机和触发器列条目都是提供一些有用选项的链接：

- 以主机为标准 - 链接到用户定义的脚本，最新数据，库存，主图的图形和屏幕
- 以触发器为标准 - 链接到最新事件，触发器配置表单和简单图表

使用过滤器

您可以使用过滤器按主机组，主机或触发器严重级别显示触发器。指定父主机组会隐式选择所有嵌套的主机组。

过滤器位于触发器 *top100* 栏下方。可以通过单击左侧的// 过滤 //选项卡打开和折叠它。时间选择器

时间选择器 允许通过单击鼠标选择经常需要的时间段。单击过滤器旁边的时间段选项卡可以打开时间段选择器。

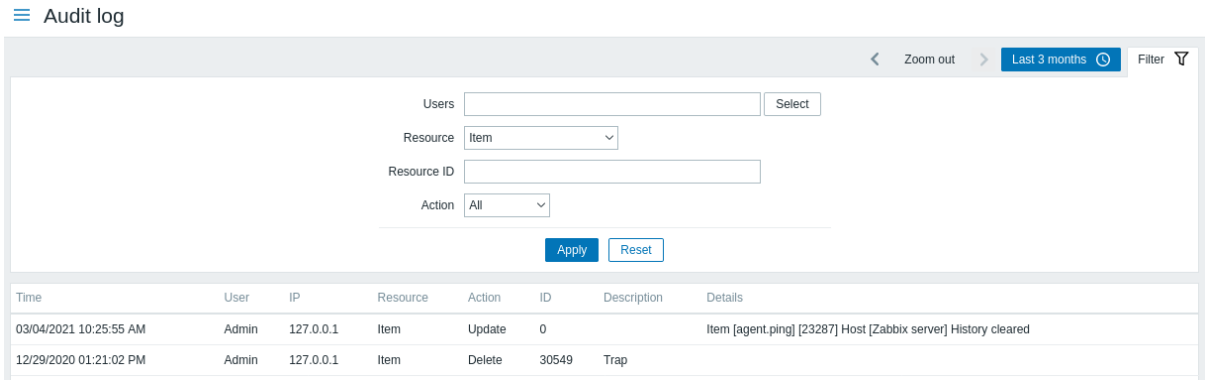
4 审计

概述

在 报告 - > 审核部分，用户可以查看在前端所做更改的记录。

审计日志

在此屏幕中，可以看到在前端进行的各种更改的审核日志。你可以使用过滤器，位于审计日志栏之下，缩小用户，活动类型，受影响资源和时间段的记录。



显示数据：

参数功	介绍
时间戳（*Time*）审计记录	时间戳.
用户（*User*）活动用	。
IP	在活动中使用的 IP。
资源（*Resource*）将显示	影响的资源。
操作（*Action*）活动类	显示 - 登录，注销，添加，更新，删除，启用或者 禁止.
ID	显示受影响资源的 ID。
描述（*Description*）显示资	的描述。
细节（*Details*）显示执	活动的详细信息。

使用过滤器

你可以通过过滤器使用，用户、影响周期、以及影响的资源来精确过滤出想获得的数据

过滤器在 审计日志导航条下方。通过左侧的过滤按钮来使用它。

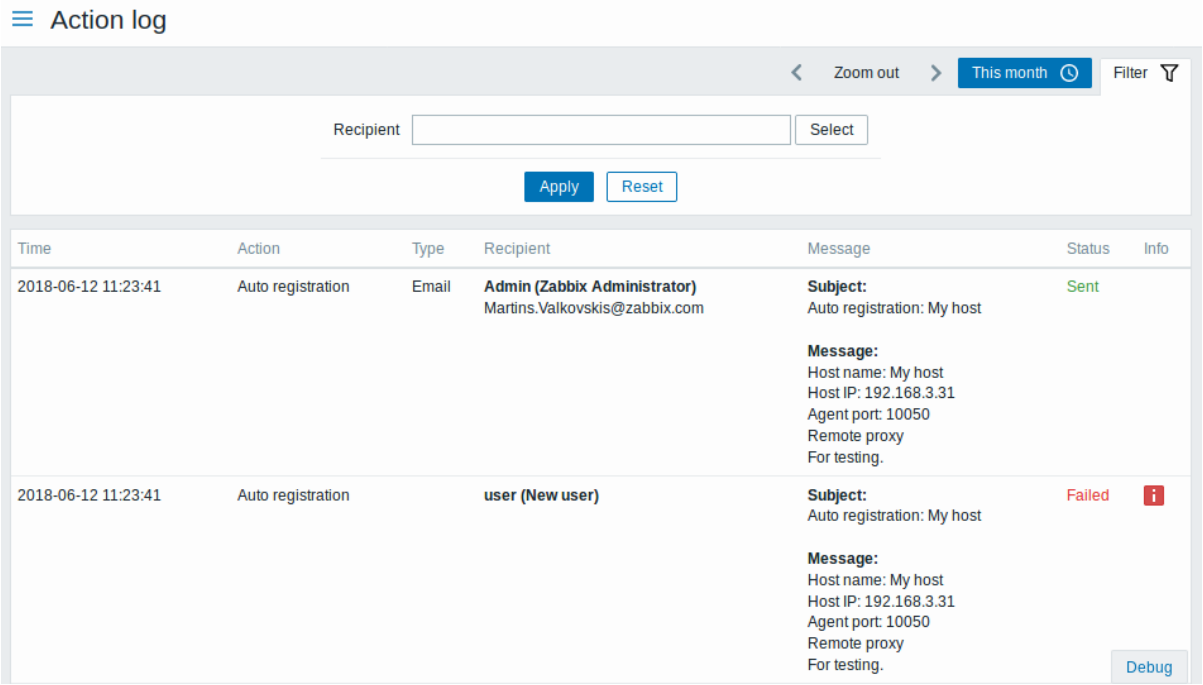
时间选择器

时间选择器 允许通过单击鼠标选择经常需要的时间段。单击过滤器旁边的时间段选项卡可以打开时间段选择器。

5 动作日志

简介

在此页面中，将显示在操作中执行的操作（通知，远程命令）的详细信息。



设置信息:

参数功	说明
时间戳（*Time*）操作时间	
操作（*Action*）显示导	operation 的 action 名称。 Zabbix 2.4.0 之后的版本可以显示动作名称。
类型（*Type*）显示操	类型 - 邮件或者 命令.
别名（*Recipient(s)*）显示用	别名，姓名（括号中）和通知收件人的电子邮件地址。 Zabbix 2.4.0 之后的版本可以显示显示用户别名，姓名
消息（*Message*）将显示	息/远程命令的内容。
状态（*Status*）显示操	状态： 进行中 - actions 正在进行中\\对于正在进行的 actions，显示剩余的重试次数 - 服务器将尝试发送通知的剩余次数。 已发送 - 通知已发送 已执行 - 命令已执行 未发送 - action 并未结束
信息（*Info*）显示有	操作执行的错误信息（如果有）。

使用过滤器

你可以通过过滤收件人、时间等信息来缩小查看范围。

过滤器位于动作日志导航栏下方。可以通过单击左侧的// 过滤 //选项卡打开和折叠它。

时间选择器

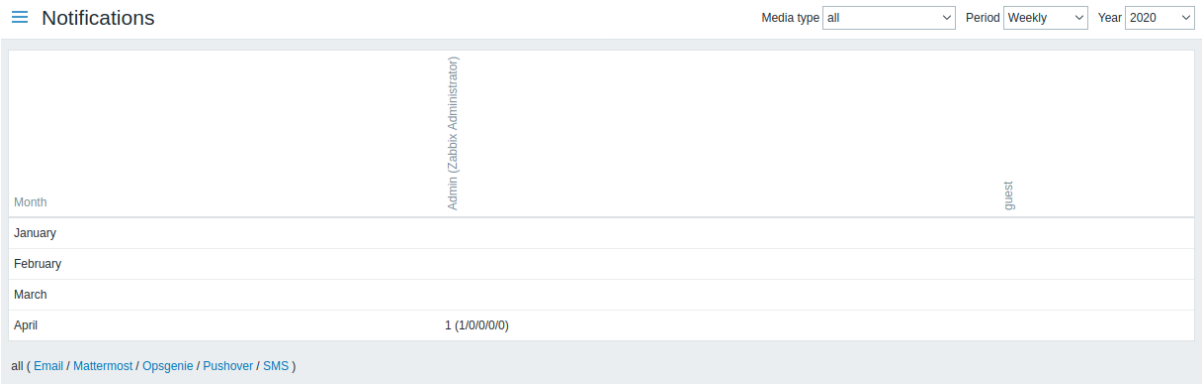
时间选择器 允许通过单击鼠标选择经常需要的时间段。单击过滤器旁边的时间段选项卡可以打开时间段选择器。

6 通知

概述

在 报告 - > 通知栏中，将显示发送给每个用户的通知数量的报告。

从右上角的下拉菜单中，您可以选择媒体类型（或全部），周期（每天/周/月/年的数据）和发送通知的年份。



每列显示每个系统用户的总计数。

4 配置

概述

The Configuration menu contains sections for setting up major Zabbix functions, such as hosts and host groups, data gathering, data thresholds, sending problem notifications, creating data visualisation and others.

“配置” 菜单包含用于设置主要 Zabbix 功能的部分，例如主机和主机组，数据收集，数据阈值，发送问题通知，创建数据可视化等。

1 主机组

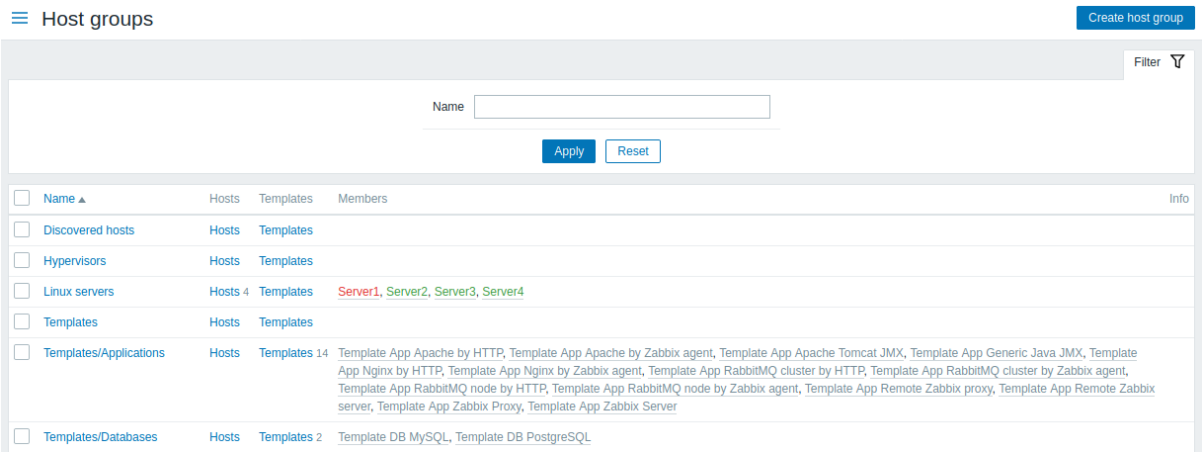
概述

In the *Configuration* → *Host groups* section users can configure and maintain host groups. A host group can contain both templates and hosts.

在 Configuration→Host groups 部分中，用户可以配置和维护主机组。主机组可以包含模板和主机。

A listing of existing host groups with their details is displayed. You can search and filter host groups by name.

列表将显示现有主机组及其详细信息。您可以按名称搜索和过滤主机组。



Displayed data: 数据展示：

Column	Description
Name	Name of the host group. Clicking on the group name opens the host group configuration form.

Column	Description
Hosts	Number of hosts in the group (displayed in grey). Clicking on ”Hosts” will, in the whole listing of hosts, filter out those that belong to the group.
Templates	Number of templates in the group (displayed in grey). Clicking on ”Templates” will, in the whole listing of templates, filter out those that belong to the group.
Members	Names of group members. Template names are displayed in grey, monitored host names in blue and non-monitored host names in red. Clicking on a name will open the template/host configuration form.
Info	Error information (if any) regarding the host group is displayed.

列名描	
主机组名主机组	字. 点击主机组名打开主机组配置表格.
主机主	组中的主机个数 (显示灰色)。单击 “主机” 将呈现属于该组的主机列表。
模板主	组中的模板个数 (显示灰色)。单击 “主机” 将呈现属于该组的模板列表。
成员主	。模板名称显示灰色，监控状态的主机名显示为蓝色，非监控状态的主机名显示为红色。单击可显示 template/host 配置信息.
信息显	有关主机组的错误信息（如果有）。

批量编辑选项

Buttons below the list offer some mass-editing options:

- *Enable hosts* - change the status of all hosts in the group to ”Monitored”
- *Disable hosts* - change the status of all hosts in the group to ”Not monitored”
- *Delete* - delete the host groups

To use these options, mark the checkboxes before the respective host groups, then click on the required button.

列表下方的按钮提供了一些批量编辑选项：

* 启用主机 - 将组中所有主机的状态更改为 “已监控”

* 禁用主机 - 将组中所有主机的状态更改为 “未监控”

* 删除 - 删除主机组

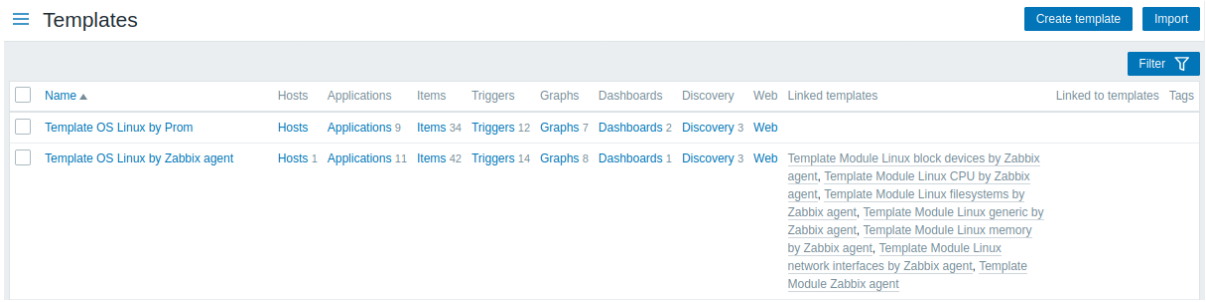
要使用这些选项，请在相应主机组之前选中复选框，然后单击所需按钮。

2 模板

概述

在配置 **-** > 模板部分，用户可以配置和维护模板。

显示现有模板及其详细信息的列表如下：



从标题栏中的右侧的下拉列表中，您可以选择是显示所有模板还是仅显示属于组的模板。您也可以按名称搜索和过滤模板。

显示数据：

列	述
模板模 实体（应用程序，项目，触发器，图形，屏幕，发现，*Web*）模板中各个实体的数量（以灰色显示）。单击实体名	名称，单击模板名称打开模板配置表单. 将在该实体的整个列表中过滤掉属于该模板的那些实体。

列	述
链接的模板链接到特 链接到模板链接到的主机和模板.	模板的模板，在嵌套设置中，模板将共享所链接模板的所有实体。

点击右上角创建模板按钮配置新的模板。点击右上角导入按钮，从 XML 文件导入模板。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

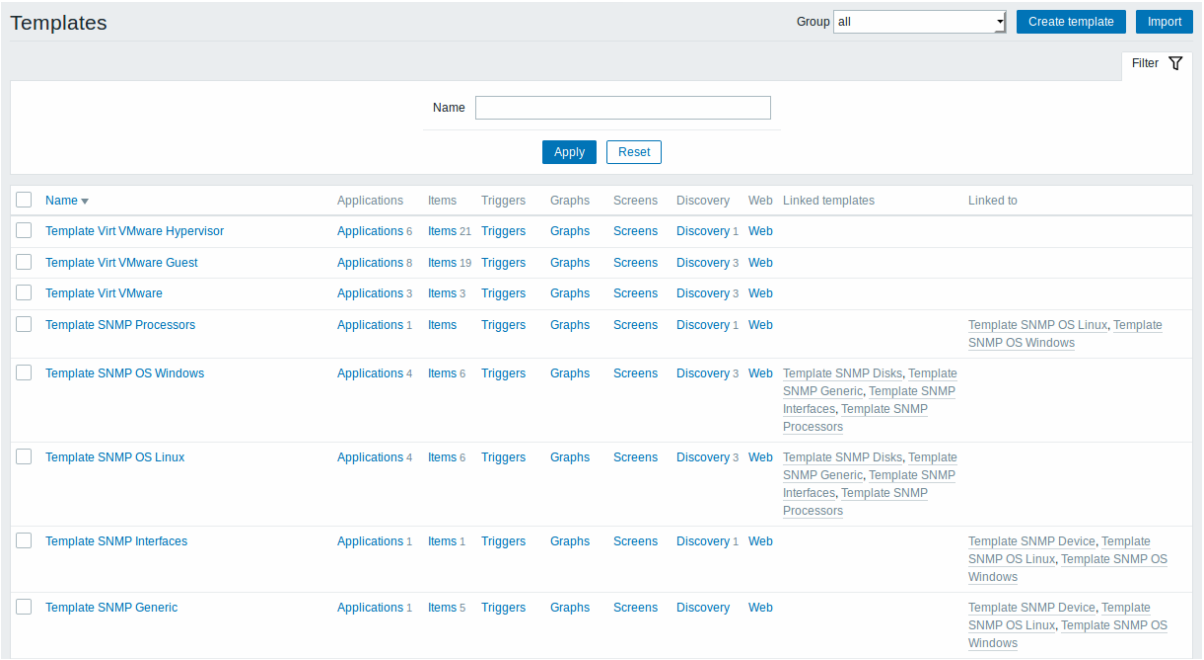
- *Export* - 将模板导出到 XML 文件
- *Delete* - 删除模板，将其链接的实体（项目，触发器等）与主机保持同步
- *Delete and clear* - 从主机中删除模板及其链接的实体

要使用这些选项，请在相应模板之前标记复选框，然后单击所需的按钮。

Overview

In the *Configuration* → *Templates* section users can configure and maintain templates.

A listing of existing templates with their details is displayed.



From the dropdown to the right in the title bar you can choose whether to display all templates or only those belonging to a group. You can also search and filter templates by name.

Displayed data:

Column	Description
Templates	Name of the template. Clicking on the template name opens the template configuration form.
Entities (Applications, Items, Triggers, Graphs, Screens, Discovery, Web)	Number of the respective entities in the template (displayed in grey). Clicking on the entity name will, in the whole listing of that entity, filter out those that belong to the template.
Linked templates	Templates that are linked to the template, in a nested setup where the template will inherit all entities of the linked templates.
Linked to	The hosts and templates that the template is linked to.

To configure a new template, click on the *Create template* button in the top right-hand corner. To import a template from an XML file, click on the *Import* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Export* - export the template to an XML file
- *Delete* - delete the template while leaving its linked entities (items, triggers etc.) with the hosts
- *Delete and clear* - delete the template and its linked entities from the hosts

To use these options, mark the checkboxes before the respective templates, then click on the required button.

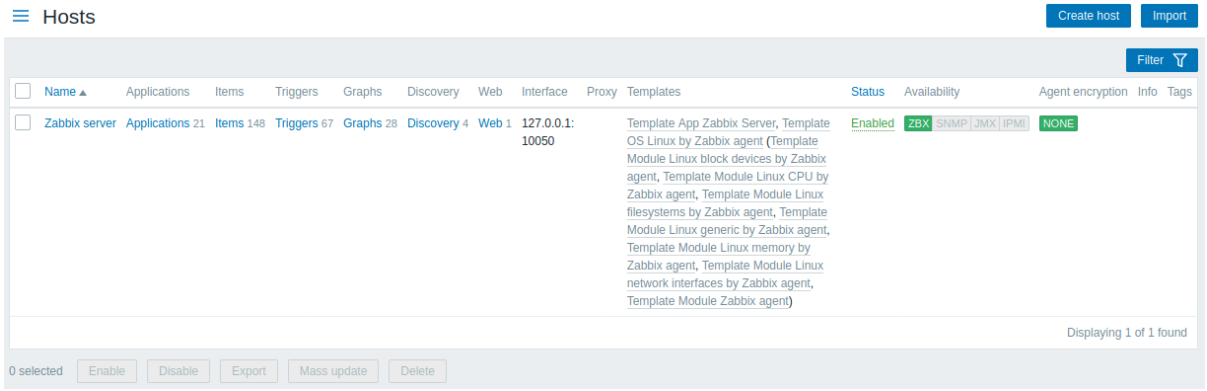
3 主机

概览

在配置 **-** > 主机中，用户可以配置和维护主机。

有一个显示现有主机及其详细信息的列表。

从右边的下拉菜单中有 主机栏，您可以选择是显示所有主机还是仅显示属于一个特定组的主机。



显示数据：

列	述
名称主 元素（应用程序，监控项，触发器，图形，发现，*Web*）单击元素名称将显示主机的项目，触发器等。各	名称，单击主机名打开主机配置表. 元素的数量以灰色显示。
界面显	主机的主界面。
模板显	与主机链接的模板。如果链接的模板中包含其他模板，那么它们将显示在括号中，以逗号分隔。单击模板名称将打开其配置表单。
状态显	主机状态 - 启用或是 禁止. 点击状态可以更改。
可用性显示	机的可用性。四个图标各自表示支持的接口（Zabbix 代理，SNMP，IPMI，JMX）。\\界面的当前状态由相应的颜色显示: 绿色 - 可用 红色 - 不可用（在鼠标悬停时，显示无法访问接口的原因的详细信息） 灰色 - 知或未配置
代理加密显示与	机连接的加密状态: **None** - 没有加密 **PSK** - 使用预共享密钥 **Cert** - 使用证书
Info	显示有关主机的错误信息（如果有）。

点击右上角创建主机按钮配置一个新的主机。点击右上角 导入按钮从 XML 文件导入主机

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

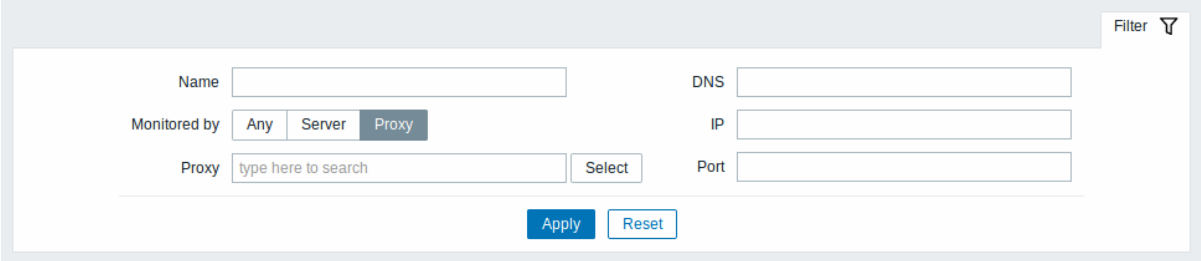
- *Enable* - 将主机状态更改为 已监控
- *Disable* - 将主机状态更改为 未被监控
- *Export* - 将主机导出为 XML 文件
- *Mass update* - 多个主机多个属性更新。
- *Delete* - 删除主机

要使用这些选项，请在相应的主机之前标记复选框，然后单击所需的按钮。

过滤器

由于该列表可能包含很多主机，可能需要过滤出您真正需要的主机。

过滤器链接在主机列表之上. 如果您点击它，则可以使用过滤器，您可以通过名称，DNS，IP 或端口号过滤主机。



阅读主机可用性

主机可用性图标反映了 Zabbix 服务器上的当前主机接口状态。因此，在前台：

- 如果禁用主机，可用性图标将不会立即变为灰色（未知状态），因为服务器必须首先同步配置更改;
- 如果启用主机，则可用性图标将不会立即变为绿色（可用），因为服务器必须同步配置更改并开始首先轮询主机。

未知主机状态

Zabbix 服务器将主机可用性图标设置为相应代理接口（Zabbix，SNMP，IMP，JMX）的灰色（未知状态），如果：

- 界面上没有启用的项目（它们被删除或禁用）；
- 主机已禁用;
- 主机被设置为由代理监视，不同的代理或由服务器监视，如果它被代理监视;
- 主机由看起来处于脱机状态的代理进行监控（在最大心跳间隔（1 小时）内没有从代理收到更新）。

*

请参阅有关主机的更多细节unreachability.

Reading host availability

Host availability icons reflect the current host interface status on Zabbix server. Therefore, in the frontend:

- If you disable a host, availability icons will not immediately turn gray (unknown status), because the server has to synchronize the configuration changes first;
- If you enable a host, availability icons will not immediately turn green (available), because the server has to synchronize the configuration changes and start polling the host first.

Unknown host status

Zabbix server sets the host availability icon to gray (unknown status) for the corresponding agent interface (Zabbix, SNMP, IMP, JMX) if:

- there are no enabled items on the interface (they were removed or disabled);
- host is disabled;
- host is set to be monitored by proxy, a different proxy or by server if it was monitored by proxy;
- host is monitored by a proxy that appears to be offline (no updates received from the proxy during the maximum heartbeat interval - 1 hour).

Setting host availability to unknown is done after server configuration cache synchronization. Restoring host availability (available/unavailable) on hosts monitored by proxies is done after proxy configuration cache synchronization.

See also more details about host unreachability.

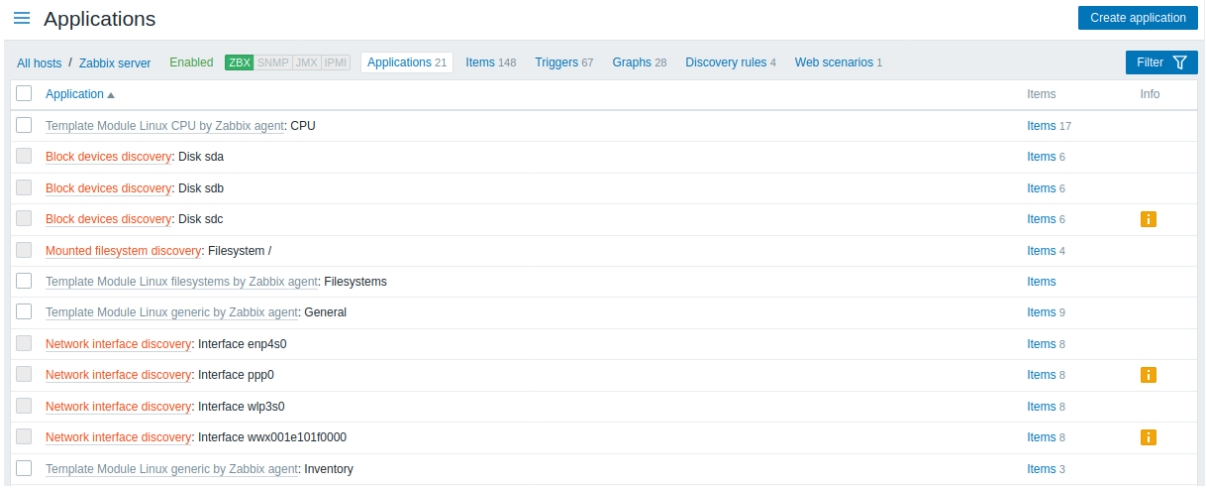
1 应用集

概览

可以从配置 - > 模板中访问模板的应用程序列表，然后单击相应模板的应用程序。

可以在 配置 - > 主机中访问主机的应用程序列表，然后单击相应主机的应用程序。

显示现有应用程序的列表：



显示数据：

列	述
应用应	的名称, 显示为直接创建的应用程序的蓝色链接。 单击应用程序名称链接将打开该应用程序配置表. 如果主机应用程序属于模板，则模板名称将以灰色链接的形式显示在应用程序名称之前。单击模板链接将打开模板级别的应用程序列表。
项	击项目以查看应用程序中包含的项目。项目数量显示为灰色。
Info	显示有关应用程序的错误信息（如果有）。

点击右上角创建应用程序按钮配置新的应用程序。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将应用状态更改为 启用
- *Disable* - 将应用状态更改为 禁用
- *Delete* - 删除应用

要使用这些选项，请在各个应用程序之前标记复选框，然后单击所需的按钮。

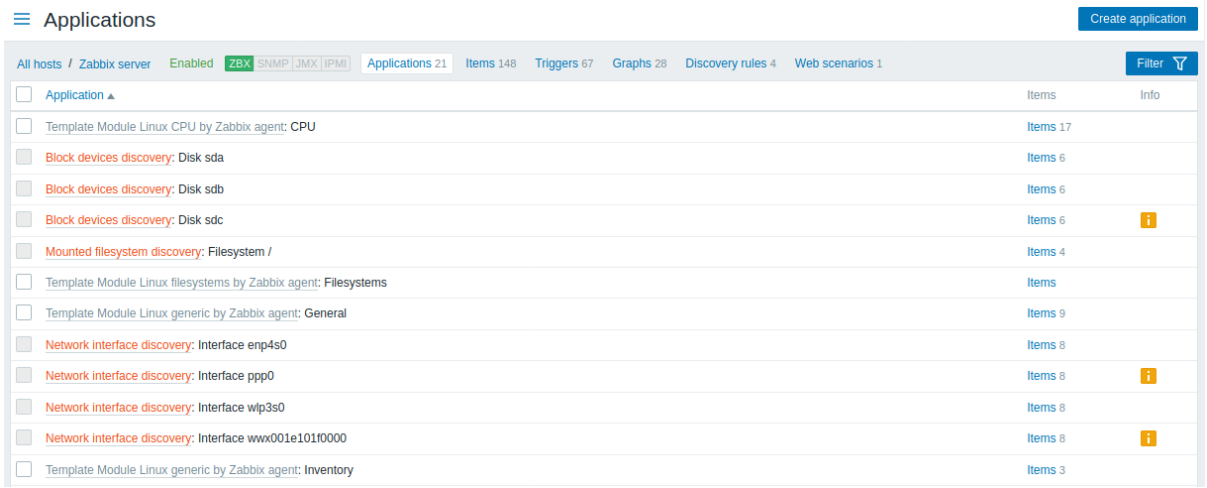
1 Applications

Overview

The application list for a template can be accessed from *Configuration* → *Templates* and then clicking on Applications for the respective template.

The application list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Applications for the respective host.

A list of existing applications is displayed.



Displayed data:

Column	Description
Application	Name of the application, displayed as a blue link for directly created applications. Clicking on the application name link opens the application configuration form. If the host application belongs to a template, the template name is displayed before the application name, as a grey link. Clicking on the template link will open the application list on the template level.
Items	Click on Items to view the items contained in the application. The number of items is displayed in grey.
Info	Error information (if any) regarding the application is displayed.

To configure a new application, click on the *Create application* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change application status to *Enabled*
- *Disable* - change application status to *Disabled*
- *Delete* - delete the applications

To use these options, mark the checkboxes before the respective applications, then click on the required button.

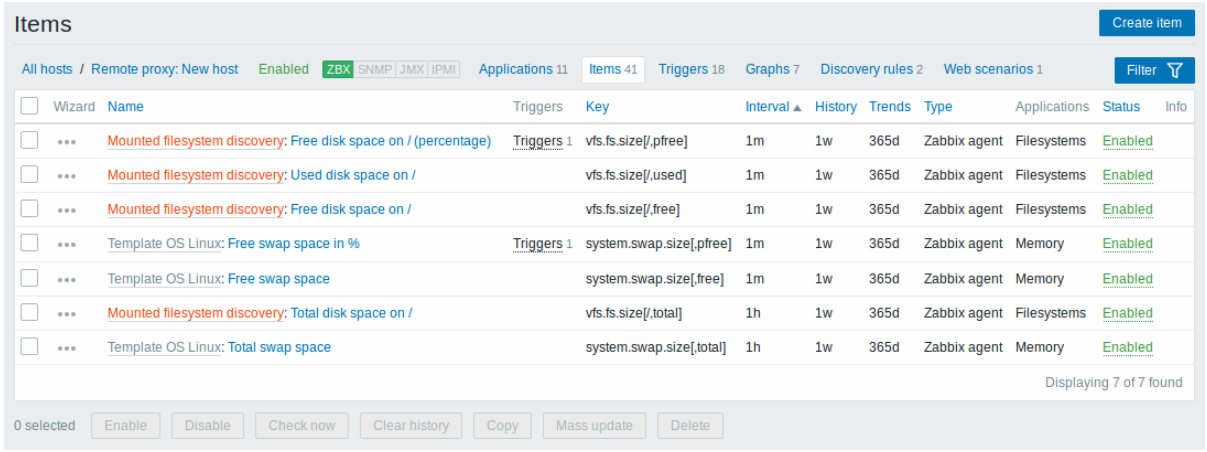
2 监控项

概览

可以从配置 - > 模板中访问模板的项目列表，然后单击相应模板的项目。

在配置 - > 主机中可以访问主机的项目列表，然后单击相应主机的项目。

现有项目的列表：



显示数据：

列	述
Wizard	向导图标是指向向导的链接，用于根据项目创建触发器。
Name	项目的名称，显示为项目详细信息的蓝色链接。 单击项目名称链接将打开该项目配置表. 如果主机项目属于模板，模板名称将显示在项目名称之前，作为灰色链接。单击模板链接将打开模板级别的项目列表。 如果项目是从项目原型创建的，则其名称前面是低级别的发现规则名称，以橙色显示。单击发现规则名称将打开项目原型列表。
触发器将鼠	移动到触发器上将显示一个信息框，显示与该项目相关联的触发器。 触发器的数量以灰色显示。
Key	显示项目键。
间隔显	检查频率。
历史将	示项目数据记录将保留多少天。

列	述
趋势将	示将保留多少天的项目趋势记录。
类型显	项目类型（Zabbix 代理，SNMP 代理，简单检查等）。
应用程序项目应	程序显示。
状态显	项目状态 - 启用, 禁用或者 不支持. 通过点击状态，您可以更改它 - 从启用到禁用（反之亦然）；从不支持到禁用（反之亦然）。
Info	如果一切正常，此列中不会显示图标。如果有错误，将显示带有十字架的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的工具提示。

点击右上角 创建项目按钮配置新项目。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

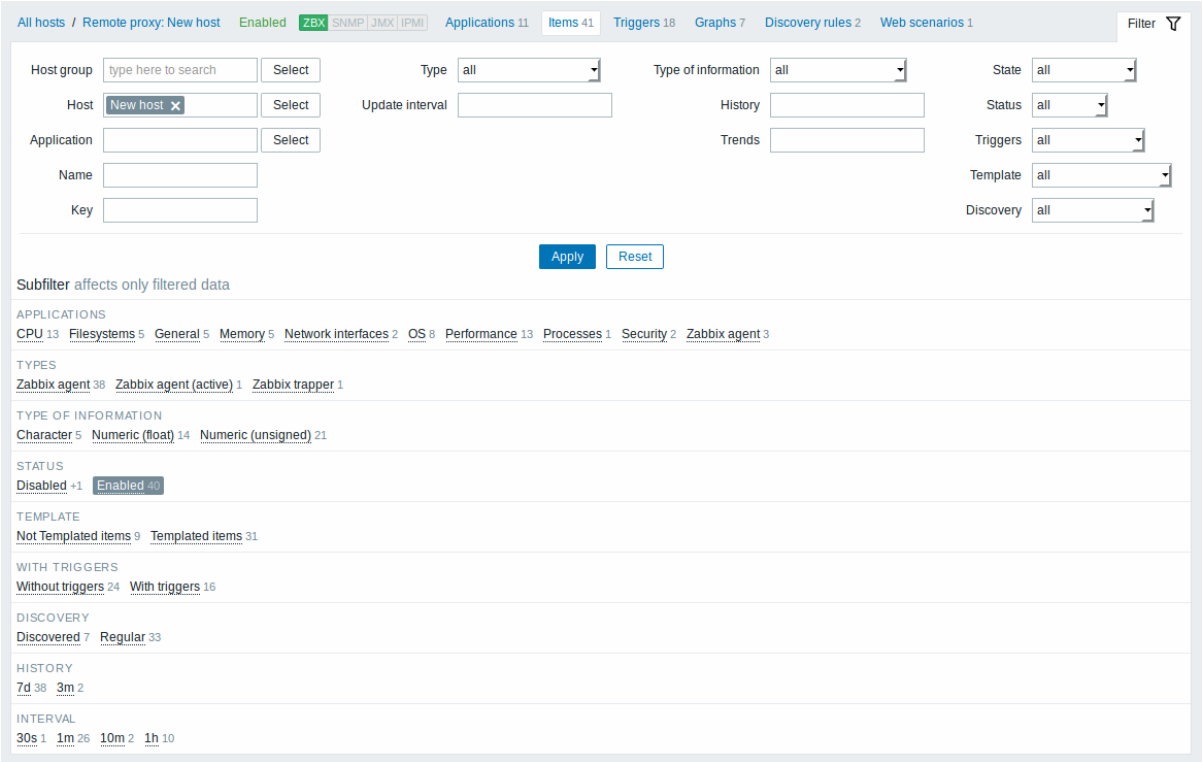
- *Enable* - 将项目状态更改为 启用
- *Disable* - 将项目状态更改为 禁用
- *Clear history* -删除项目的历史记录和趋势数据
- *Copy* - 将项目复制到其他主机或模板
- *Mass update* - 多项目多属性 同时升级
- *Delete* - 删除项

要使用这些选项，请在相应项目之前标记复选框，然后单击所需的按钮。

过滤器

由于列表可能包含很多项目，可能需要过滤出您真正需要的项目。

过滤器链接在列表上方可用。如果您点击它，则可以使用过滤器，您可以通过多个属性过滤项目。



在过滤器下方的子滤波器 below the filter 提供进一步的过滤选项（已经过滤的数据）。您可以选择具有公共参数值的项目组。如果单击一个组，它将突出显示，只有具有此参数值的项目保留在列表中。

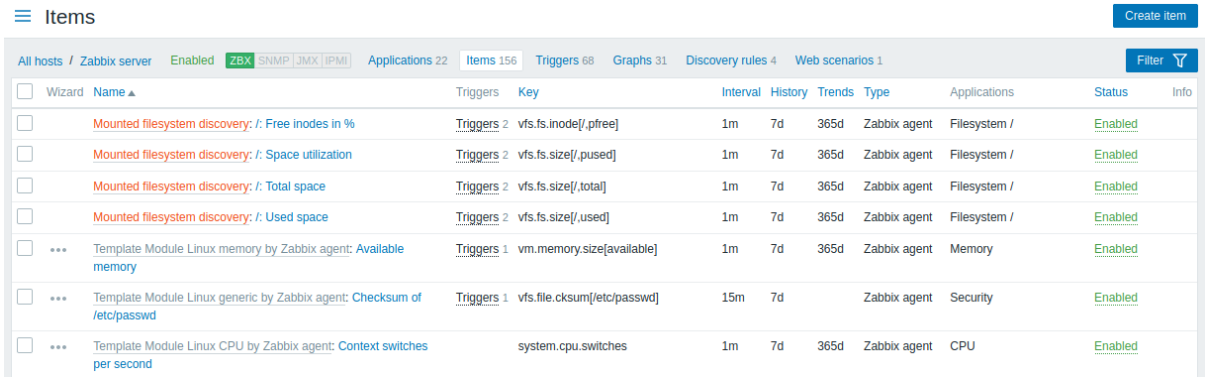
2 Items

Overview

The item list for a template can be accessed from *Configuration* → *Templates* and then clicking on Items for the respective template.

The item list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Items for the respective host.

A list of existing items is displayed.



Displayed data:

Column	Description
Wizard	The wizard icon is a link to a wizard for creating a trigger based on the item.
Name	Name of the item, displayed as a blue link to item details. Clicking on the item name link opens the item configuration form. If the host item belongs to a template, the template name is displayed before the item name, as a grey link. Clicking on the template link will open the item list on the template level. If the item has been created from an item prototype, its name is preceded by the low level discovery rule name, in orange. Clicking on the discovery rule name will open the item prototype list.
Triggers	Moving the mouse over Triggers will display an info box displaying the triggers associated with the item. The number of the triggers is displayed in grey.
Key	Item key is displayed.
Interval	Frequency of the check is displayed. *Note* that passive items can also be checked immediately by pushing the *Check now* button.
History	How many days item data history will be kept is displayed.
Trends	How many days item trends history will be kept is displayed.
Type	Item type is displayed (Zabbix agent, SNMP agent, simple check, etc).
Applications	Item applications are displayed.
Status	Item status is displayed - *Enabled*, *Disabled* or *Not supported*. By clicking on the status you can change it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new item, click on the *Create item* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

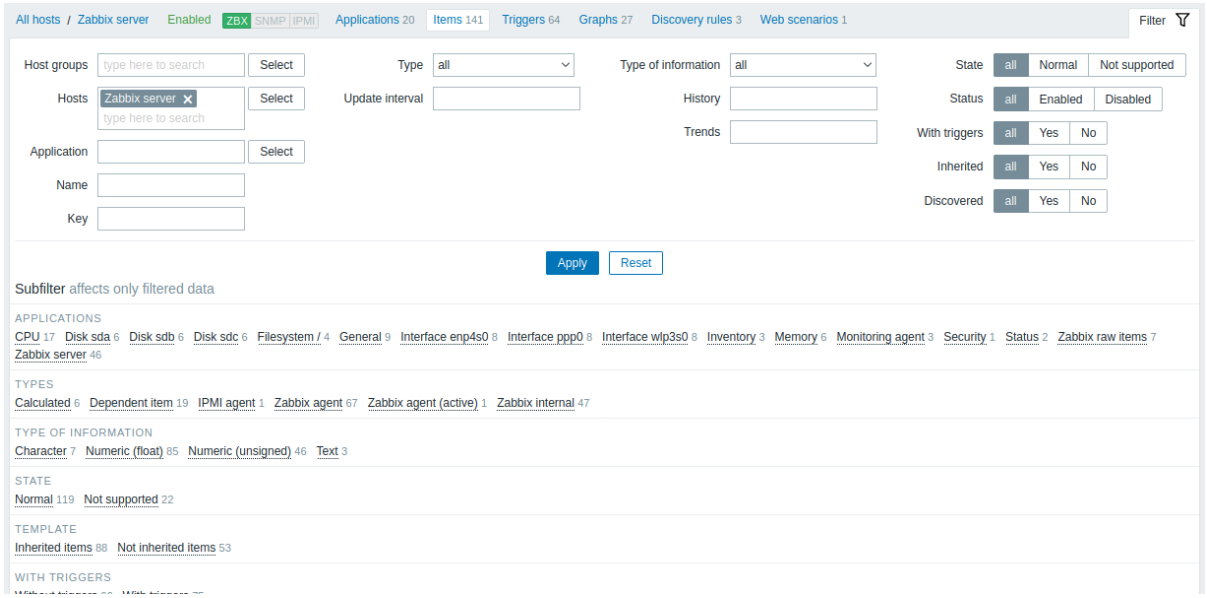
- *Enable* - change item status to *Enabled*
- *Disable* - change item status to *Disabled*
- *Check now* - execute a check for new item values immediately. Supported for **passive** checks only (see more details). Note that when checking for values immediately, configuration cache is not updated, thus the values will not reflect very recent changes to item configuration.
- *Clear history* - delete history and trend data for items
- *Copy* - copy the items to other hosts or templates
- *Mass update* - update several properties for a number of items at once
- *Delete* - delete the items

To use these options, mark the checkboxes before the respective items, then click on the required button.

Filter

As the list may contain very many items, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list. If you click on it, a filter becomes available where you can filter items by several properties.



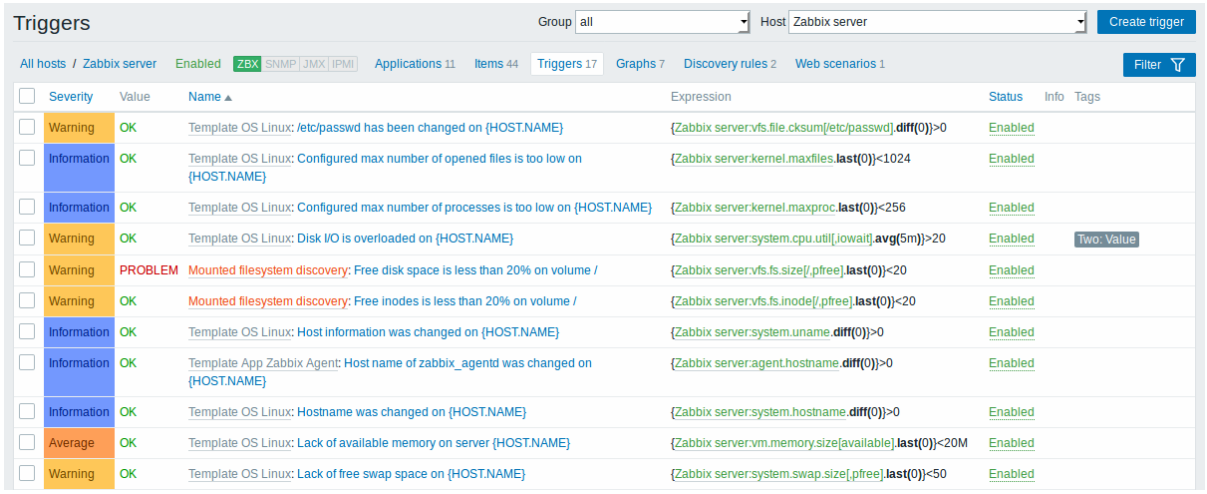
The **Subfilter** below the filter offers further filtering options (for the data already filtered). You can select groups of items with a common parameter value. If you click on a group it gets highlighted and only the items with this parameter value remain in the list.

3 触发器

概述

可以从 配置 - > 模板中访问模板的触发器列表，然后单击相应模板的触发器。

可以从配置 - > 主机访问主机的触发器列表，然后单击相应主机的触发器。



显示数据：

列	述
Severity	触发器的严重性由名称和单元格背景颜色显示。
Name	触发器的名称，显示为蓝色链接以触发细节。 单击触发器名称链接将打开触发器配置表.\\如果主机触发器属于模板，则模板名称将在触发器名称之前显示为灰色链接。单击模板链接将打开模板级别的触发器列表。 如果触发器是从触发器原型创建的，则其名称前面是低级别的发现规则名称，以橙色显示。单击发现规则名称将打开触发器原型列表。
Expression	显示触发表达式。表达式的 host-item 部分显示为链接，链接到项目配置表单。
Status	显示触发状态 - 启用，禁用或者未知. 通过点击状态，您可以更改它 - 从启用到禁用（反之亦可）；从未知到已禁用（反之亦可）。

列	述
Info	如果一切正常，此列中不会显示图标。如果有错误，将显示带有十字架的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的工具提示。

点击右上角创建触发器配置新的触发器。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将触发状态更改为 启用
- *Disable* - 将触发状态更改为 禁用
- *Copy* - 将触发器复制到其他主机或模板
- *Mass update* - 一次更新多个触发器的几个属性
- *Delete* - 删除触发器

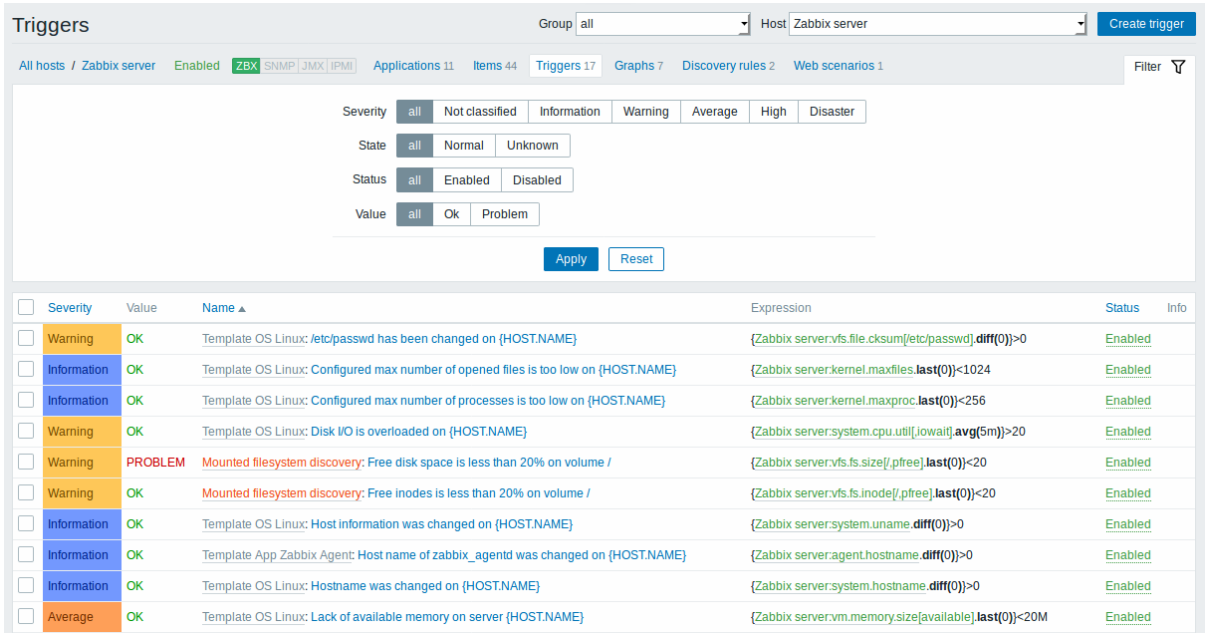
要使用这些选项，请在相应的触发器之前标记复选框，然后单击所需的按钮。

3 Triggers

Overview

The trigger list for a template can be accessed from *Configuration* → *Templates* and then clicking on Triggers for the respective template.

The trigger list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Triggers for the respective host.



Displayed data:

Column	Description
Severity	Severity of the trigger is displayed by both name and cell background colour.
Name	Name of the trigger, displayed as a blue link to trigger details. Clicking on the trigger name link opens the trigger configuration form. If the host trigger belongs to a template, the template name is displayed before the trigger name, as a grey link. Clicking on the template link will open the trigger list on the template level. If the trigger has been created from a trigger prototype, its name is preceded by the low level discovery rule name, in orange. Clicking on the discovery rule name will open the trigger prototype list.
Expression	Trigger expression is displayed. The host-item part of the expression is displayed as a link, leading to the item configuration form.
Status	Trigger status is displayed - *Enabled*, *Disabled* or *Unknown*. By clicking on the status you can change it - from Enabled to Disabled (and back); from Unknown to Disabled (and back).

Column	Description
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new trigger, click on the *Create trigger* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change trigger status to *Enabled*
- *Disable* - change trigger status to *Disabled*
- *Copy* - copy the triggers to other hosts or templates
- *Mass update* - update several properties for a number of triggers at once
- *Delete* - delete the triggers

To use these options, mark the checkboxes before the respective triggers, then click on the required button.

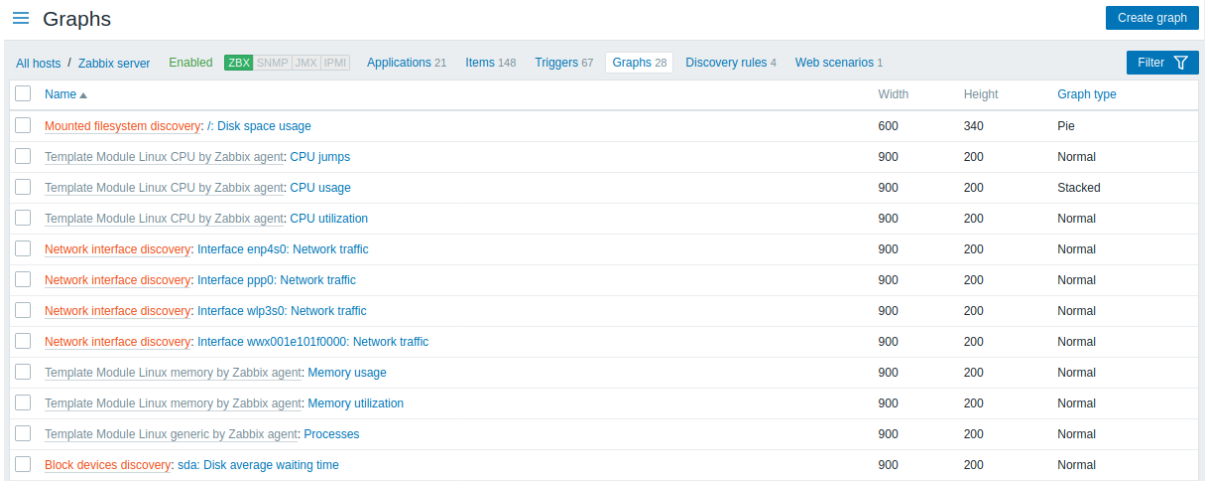
4 图形

概述

可以从 Configuration → Templates （配置 → 模板）访问模板的自定义图列表，然后单击相应模板的 Graphs（图）。

可以从 Configuration → Hosts （配置 → 主机）访问主机的自定义图列表，然后单击相应主机的图。

显示现有图的列表。.



显示数据：

列	述
Name	自定义图的名称，显示图细节的蓝色链接。 点击图名的链接来打开图.configuration form. 如果主机图属于模板，则模板名称将在图名称之前，以灰色链接显示。单击模板链接打开模板级的图列表。 如果图是从图原型创建的，则其名称前面是低级别发现规则名，并以橙色显示。单击发现规则名将打开图原型列表
Width	图显示的宽度
Height	图显示的长度
Graph type	图显示的类型 - *Normal,* （正常图形）, *Stacked,*（叠加图形）, *Pie* （饼状图形）或者 *Exploded.*（分解饼状图形）.

配置新的图，可以点击顶部右上角的 Create graph 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Copy* - 将图复制到其他主机或模板上。
- *Delete* - 删除图

要使用这些选项，请在各个图之前标记复选框，然后单击所需的按钮

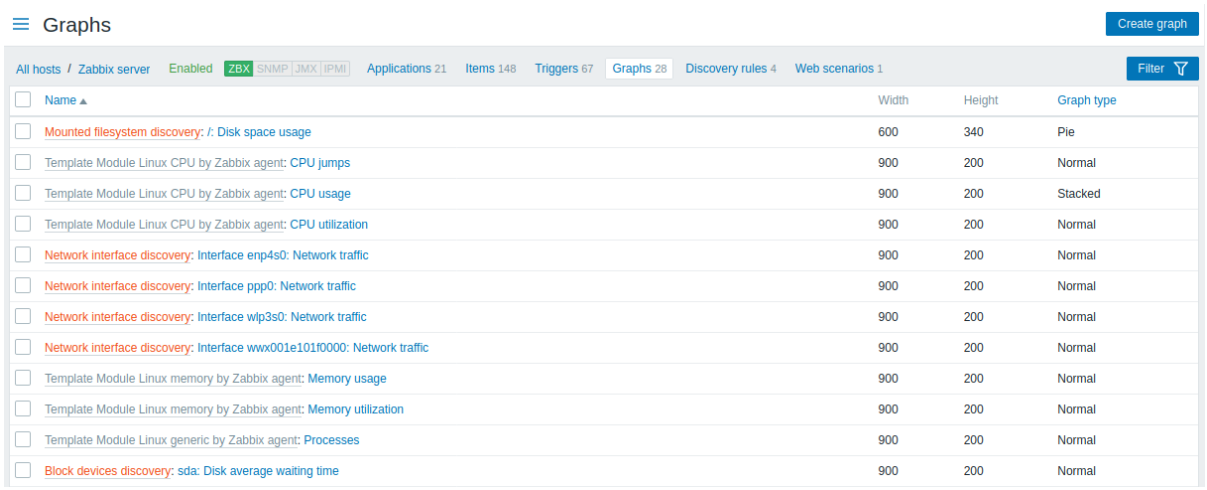
4 Graphs

Overview

The custom graph list for a template can be accessed from *Configuration* → *Templates* and then clicking on Graphs for the respective template.

The custom graph list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Graphs for the respective host.

A list of existing graphs is displayed.



Displayed data:

Column	Description
Name	Name of the custom graph, displayed as a blue link to graph details. Clicking on the graph name link opens the graph configuration form. If the host graph belongs to a template, the template name is displayed before the graph name, as a grey link. Clicking on the template link will open the graph list on the template level. If the graph has been created from a graph prototype, its name is preceded by the low level discovery rule name, in orange. Clicking on the discovery rule name will open the graph prototype list.
Width	Graph width is displayed.
Height	Graph height is displayed.
Graph type	Graph type is displayed - *Normal*, *Stacked*, *Pie* or *Exploded*.

To configure a new graph, click on the *Create graph* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Copy* - copy the graphs to other hosts or templates
- *Delete* - delete the graphs

To use these options, mark the checkboxes before the respective graphs, then click on the required button.

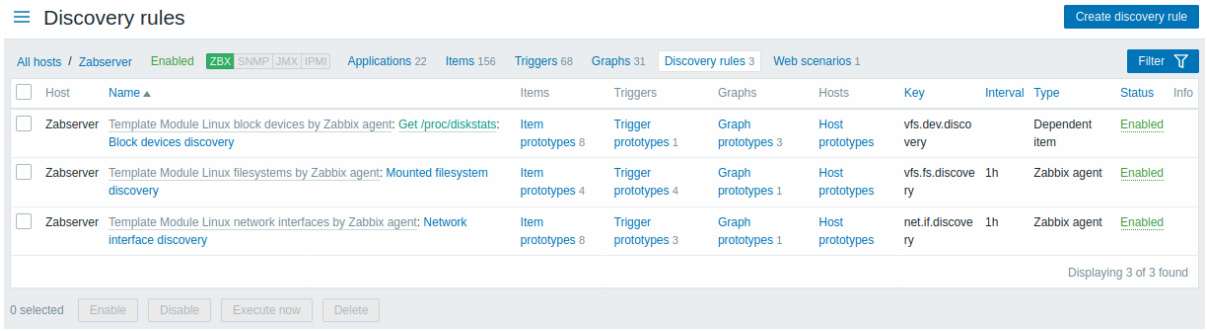
5 发现规则

概述

从 Configuration → Templates （配置 → 模板）访问模板的低级别发现规则，随后点击相应模板的 Discovery （发现）。

从 Configuration → Hosts （配置 → 主机）访问主机的低级别发现规则列表，随后点击相应主机的 Discovery （发现）。

显示现有的低级别发现规则列表：



显示数据:

列	述
Name	规则名，用蓝色链接来显示。 点击规则名打开低级别发现规则。configuration form. 如果发现规则属于模板，模板名将以灰色链接，显示在规则名前面。点击模板链接将会在模板级打开规则列表。
Items	显示监控项原型列表的链接 现有监控项原型的数量用灰色来显示。
Triggers	显示触发器原型列表的链接。 现有触发器原型的数量用灰色来显示。
Graphs	显示图原型列表的链接。 现有图原型的数量用灰色来显示。
Hosts	显示主机原型列表的链接。 现有主机原型的数量用灰色来显示。
Key	显示用于发现的监控项值。
Interval	显示执行发现的频率。
Type	显示用于发现的监控项类型 (Zabbix agent, SNMP agent, 等).
Status	D 显示发现规则的状态 - Enabled,（可用）Disabled （不可用）or Not supported.（不支持）通过点击状态可以改变它。从可用到不可用（反之亦然）；从不支持到不可用（反之亦然）
Info	如果一切顺利，没有任何图标显示在这一列。但要有错误的话，如果有错误，将显示带有十字的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的提示

配置新的低等级发现规则，可以点击顶部右上角的 Create discovery rule 按钮。and corner.

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Enable* - 将低等级发现规则的状态改为 Enabled 可用
- *Disable* - 将低等级发现规则的状态改为 Disabled 禁用。
- *Delete* - 删除低等级发现规则。

要使用这些选项，请在各个发现规则之前标记复选框，然后单击所需的按钮

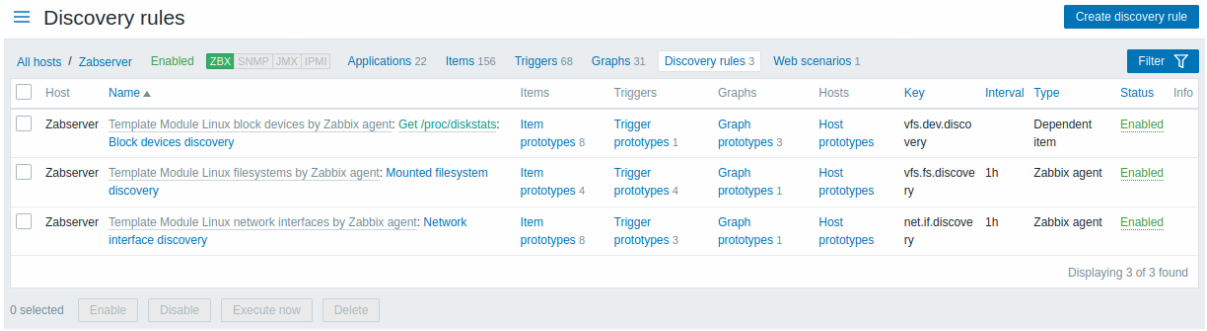
5 Discovery rules

Overview

The list of low-level discovery rules for a template can be accessed from *Configuration* → *Templates* and then clicking on Discovery for the respective template.

The list of low-level discovery rules for a host can be accessed from *Configuration* → *Hosts* and then clicking on Discovery for the respective host.

A list of existing low-level discovery rules is displayed.



Displayed data:

Column	Description
Name	Name of the rule, displayed as a blue link. Clicking on the rule name opens the low-level discovery rule configuration form. If the discovery rule belongs to a template, the template name is displayed before the rule name, as a grey link. Clicking on the template link will open the rule list on the template level.
Items	A link to the list of item prototypes is displayed. The number of existing item prototypes is displayed in grey.
Triggers	A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in grey.
Graphs	A link to the list of graph prototypes displayed. The number of existing graph prototypes is displayed in grey.
Hosts	A link to the list of host prototypes displayed. The number of existing host prototypes is displayed in grey.
Key	The item key used for discovery is displayed.
Interval	The frequency of performing discovery is displayed. *Note* that discovery can also be performed immediately by pushing the *Check now* button below the list.
Type	The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).
Status	Discovery rule status is displayed - *Enabled*, *Disabled* or *Not supported*. By clicking on the status you can change it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new low-level discovery rule, click on the *Create discovery rule* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change the low-level discovery rule status to *Enabled*
- *Disable* - change the low-level discovery rule status to *Disabled*
- *Check now* - perform discovery based on the discovery rules immediately. See more details. Note that when performing discovery immediately, the configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration
- *Delete* - delete the low-level discovery rules

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

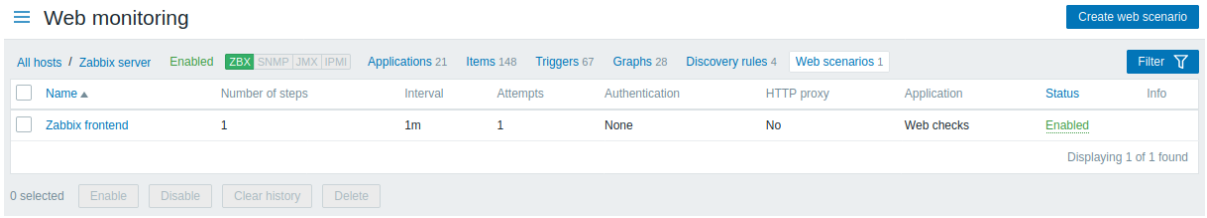
6 Web 场景

概述

从 Configuration →Templates （配置 → 模板）访问模板的 web 场景列表，随后点击相应模板的 Web 。

从 Configuration →Hosts （配置 → 主机）访问主机的 web 场景列表，随后点击相应主机的 Web 。

显示现有的 web 场景。从 Scenarios 栏中的右下方的下拉列表中，您可以选择是显示所有 Web 场景或仅显示属于一个特定组和主机的场景。此外，您可以选择隐藏已禁用的方案（或再次显示），方法是单击相应的链接。



显示数据:

列	述
Name	Web 场景名称。点击 web 场景名称来打开 web 场景。configuration form.
Number of steps	场景里包含的步骤数。
Update interval	场景多久执行一次。
Attempts	执行 web 场景步骤的尝试有多少次
Authentication	显示验证方法- Basic, NTLM on None.
HTTP proxy	显示 HTTP proxy 或者在不应用的情况下选择'No'。
Application	Web 场景应用已被显示
Status	Web 场景状态的显示 Enabled （可用）或者 Disabled.（禁用）通过点击状态来改变它。
Info	如果一切顺利，没有任何图标显示在这一列。但要有错误的话，如果有错误，将显示带有十字的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的提示

配置新的 web 场景，可以点击顶部右上角的 Create web scenario 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Enable* - 改变场景的状态至 Enabled 可用
- *Disable* - 改变场景的状态至 Disabled 不可用
- *Clear history* - 为场景清楚历史和趋势数据。
- *Delete* - 删除 web 场景。

要使用这些选项，请在各个 web 场景之前标记复选框，然后单击所需的按钮

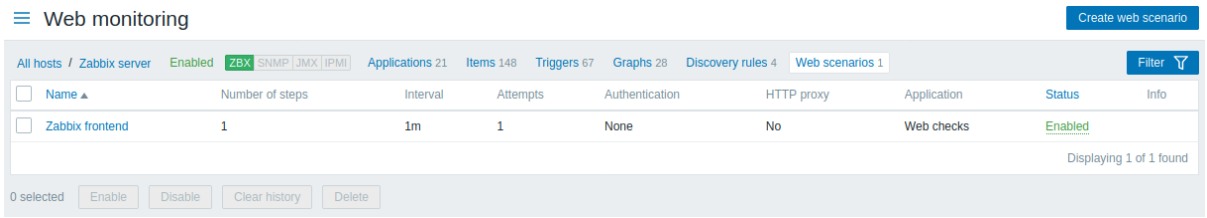
6 Web scenarios

Overview

The web scenario list for a template can be accessed from *Configuration* → *Templates* and then clicking on Web for the respective template.

The web scenario list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Web for the respective host.

A list of existing web scenarios is displayed. From the dropdown to the right in the *Scenarios* bar you can choose whether to display all web scenarios or only those belonging to one particular group and host. Additionally you can choose to hide disabled scenarios (or show them again) by clicking on the respective link.



Displayed data:

Column	Description
Name	Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form.
Number of steps	The number of steps contained in the scenario.
Update interval	How often the scenario is performed.

Column	Description
Attempts	How many attempts for executing web scenario steps are performed.
Authentication	Authentication method is displayed - Basic, NTLM on None.
HTTP proxy	Displays HTTP proxy or 'No' if not used.
Application	Web scenario application is displayed.
Status	Web scenario status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new web scenario, click on the *Create web scenario* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change the scenario status to *Enabled*
- *Disable* - change the scenario status to *Disabled*
- *Clear history* - clear history and trend data for the scenarios
- *Delete* - delete the web scenarios

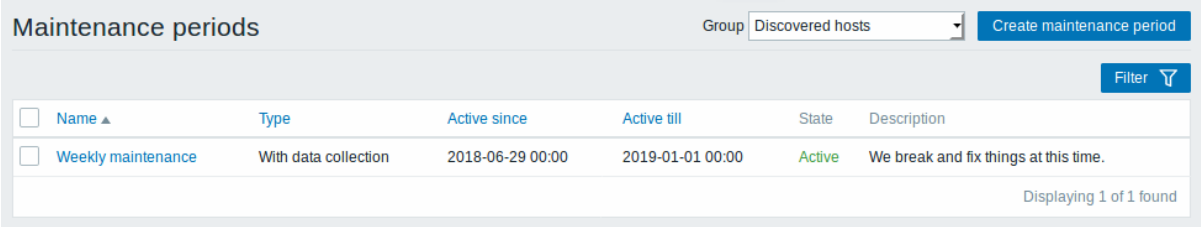
To use these options, mark the checkboxes before the respective web scenarios, then click on the required button.

4 维护期

概述

在 Configuration → Maintenance 里，用户可以为主机维护和配置维护时段。

现有的维护时段和其细节的列表展示从 Maintenance periods 中的右侧的下拉列表中，您可以选择是显示所有维护周期或是仅显示属于一个特定组的维护时段。



显示数据:

列	述
Name	维护时段的名称。点击维护时段名称打开维护时段。configuration form.
Type	显示维护时段的类型：With data collection 或 No data collection
Active since	执行维护时段的开始时间和数据。
Active till	执行维护时段的结束时间和数据
State	维护时段的状态： **Approaching** - 将会被激活。 **Active** - 已激活。 **Expired** -不再激活
Description	显示维护时段的描述。

Name, Type, Active since 和 Active till 可以按照升序/降序的方式排列。为了排序，请点击列名。

To 配置新的维护时段，点击顶部右上角的 Create maintenance period 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

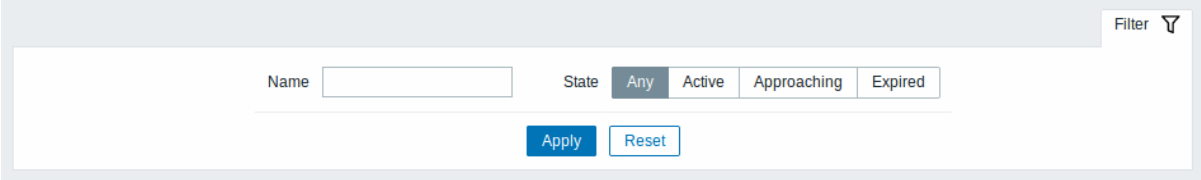
- *Delete* - 删除维护时段。

要使用这些选项，请在各个维护时段之前标记复选框，然后单击所需的按钮

过滤器

A 当一个列表包含多个维护时段时，可以使用过滤器功能找到真正您想要的。

Filter 链接在维护时段列表下可用。如果您点击它，过滤器就会可用，您可以用名称和状态进行过滤。



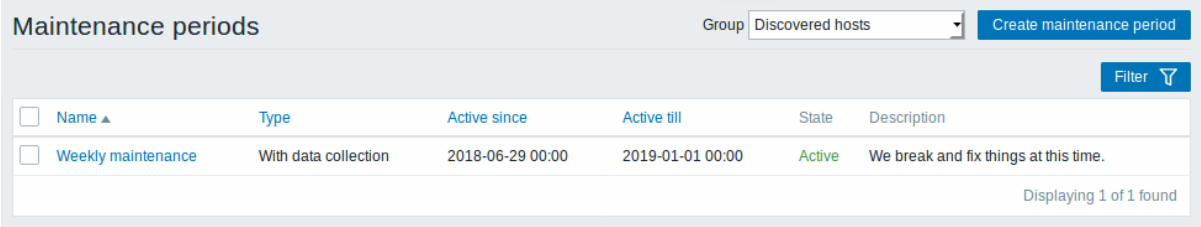
4 Maintenance

Overview

In the *Configuration* → *Maintenance* section users can configure and maintain maintenance periods for hosts.

A listing of existing maintenance periods with their details is displayed.

From the dropdown to the right in the *Maintenance periods* bar you can choose whether to display all maintenance periods or only those belonging to one particular group.



Displayed data:

Column	Description
Name	Name of the maintenance period. Clicking on the maintenance period name opens the maintenance period configuration form.
Type	The type of maintenance is displayed: *With data collection* or *No data collection*
Active since	The date and time when executing maintenance periods becomes active.
Active till	The date and time when executing maintenance periods stops being active.
State	The state of the maintenance period: **Approaching** - will become active soon **Active** - is active **Expired** - is not active any more
Description	Description of the maintenance period is displayed.

Name, *Type*, *Active since* and *Active till* are sortable columns that can be sorted in ascending/descending order. To sort, click on the column name.

To configure a new maintenance period, click on the *Create maintenance period* button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

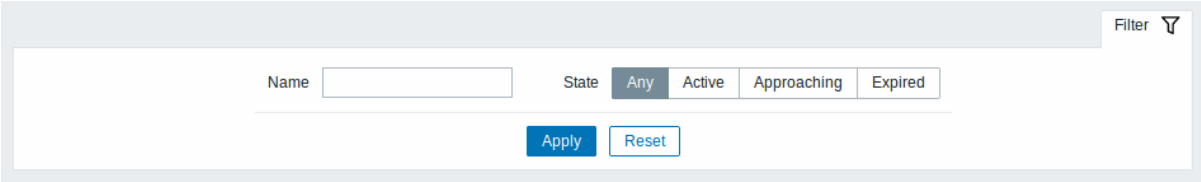
- *Delete* - delete the maintenance periods

To use this option, mark the checkboxes before the respective maintenance periods and click on *Delete*.

Filter

As the list may contain a number of maintenance periods, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of maintenance periods. If you click on it, a filter becomes available where you can filter maintenance periods by name and state.

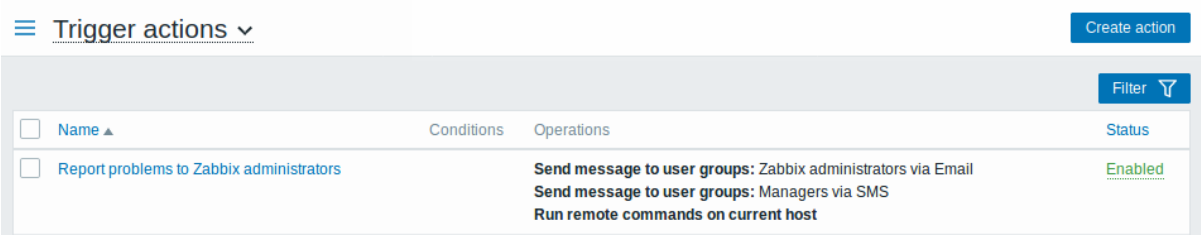


5 动作

概述

在 Configuration → Actions 里，用户可以维护和配置动作。显示的操作是分配给所选事件源（触发器，发现，自动注册）的操作。显示现有的动作和它们的描述。显示的操作是分配给所选事件源（触发器，发现，自动注册）的操作。现有的动作和其细节的列表展示。

要查看分配给不同事件源的操作，请将源从下拉菜单改到 Actions 栏中的右键。



显示数据:

列	述
Name	动作名称。点击动作名称来打开动作。configuration form.
Conditions	显示动作条件。
Operations	显示动作操作。 从 Zabbix 2.2 开始, 操作列表还显示通知收件人用于通知的媒介类型（电子邮件，短信，Jabber 等）以及名字和姓氏（在别名之后的括号中）。
Status	显示动作状态。- *Enabled* 或者 *Disabled*. 通过点击状态来修改它。 参见Escalations 获取更多细节，如在升级过程中，动作被禁用该怎么办。

配置新的动作，点击顶部右上角的 Create action 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

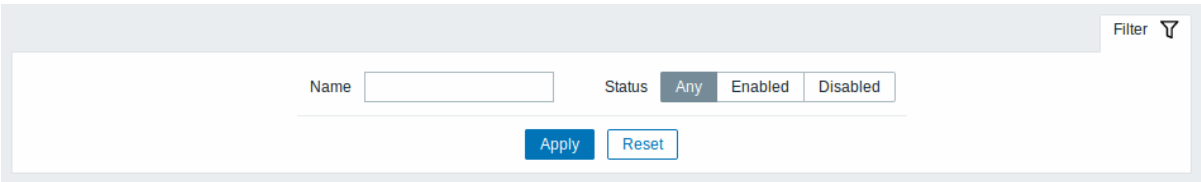
- *Enable* - 改变动作的状态至 Enabled 可用
- *Disable* - 改变动作的状态至 Disabled 不可用
- *Delete* - 删除动作

要使用这些选项，请在各个动作之前标记复选框，然后单击所需的按钮

过滤器

因为该列表可能包含多个动作，您可能需要过滤器来找到您真正需要的。

动作列表上面的 Filter （过滤器）连接是可用的。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤操作。



5 Actions

Overview

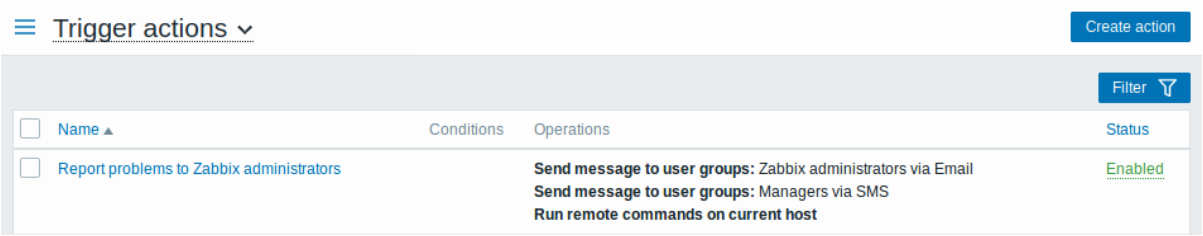
In the *Configuration* → *Actions* section users can configure and maintain actions.

A listing of existing actions with their details is displayed. The actions displayed are actions assigned to the selected event source (triggers, discovery, auto-registration).

To view actions assigned to a different event source, change the source from the dropdown to the right in the *Actions* bar.

For users without Super-admin rights actions are displayed according to permission settings. That means in some cases a user without Super-admin rights isn't able to view the complete action list because of certain permission restrictions. An action is displayed to the user without Super-admin rights if the following conditions are fulfilled:

- The user has read-write access to host groups, hosts, templates and triggers in action conditions
- The user has read-write access to host groups, hosts and templates in action operations, recovery operations and update operations
- The user has read access to user groups and users in action operations, recovery operations and update operations



Displayed data:

Column	Description
Name	Name of the action. Clicking on the action name opens the action configuration form.
Conditions	Action conditions are displayed.
Operations	Action operations are displayed. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS, Jabber, etc) used for notification as well as the name and surname (in parentheses after the alias) of a notification recipient. Action operation can both be a notification or a remote command depending on the selected type of operation.
Status	Action status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it. See the Escalations section for more details as to what happens if an action is disabled during an escalation in progress.

To configure a new action, click on the *Create action* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

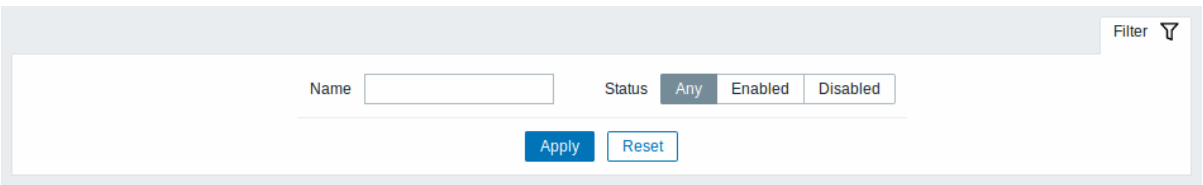
- *Enable* - change the action status to *Enabled*
- *Disable* - change the action status to *Disabled*
- *Delete* - delete the actions

To use these options, mark the checkboxes before the respective actions, then click on the required button.

Filter

As the list may contain a number of actions, it may be needed to filter out the ones you really need.

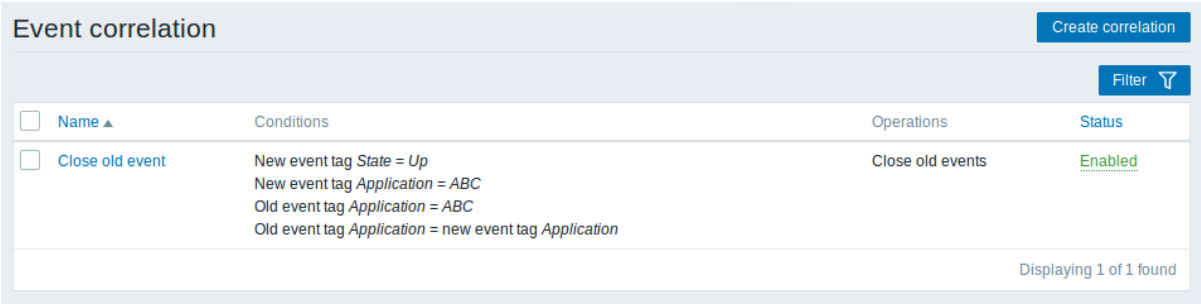
The *Filter* link is available above the list of actions. If you click on it, a filter becomes available where you can filter actions by name and status.



6 事件关联

概述

在配置 **-** > 事件关联中，用户可以配置和维护 Zabbix 事件的全局关联规则。



显示数据：

列	述
Name	关联规则的名称。单击关联规则名称打开规则配置表.
Conditions	显示关联规则条件。
Operations	显示关联规则操作。
Status	显示关联规则状态 - 启用或者禁用. 点击状态可以更改。

点击右上角 建立关联配置新的关联规则。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

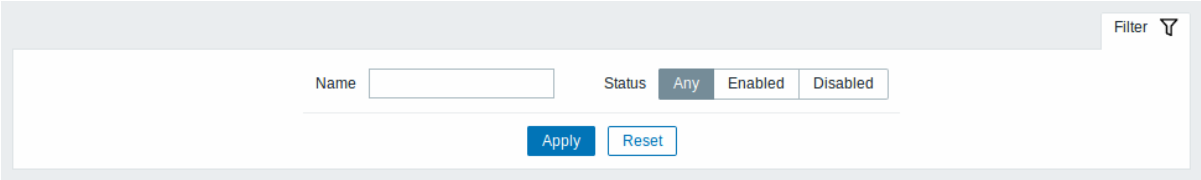
- *Enable* - 将相关关联状态更改为启用
- *Disable* -将相关关联状态更改为 禁用
- *Delete* - 删除关联规则

要使用这些选项，请在相应的关联规则之前标记复选框，然后单击所需的按钮。

过滤器

由于列表可能包含多个关联规则，可能需要过滤出您真正需要的那些。

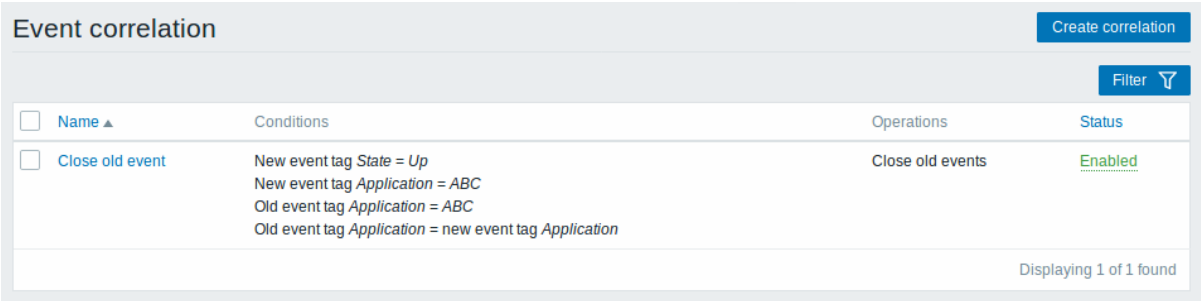
过滤器链接在相关规则列表上方可用。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤关联规则。



6 Event correlation

Overview

In the *Configuration* → *Event correlation* section users can configure and maintain global correlation rules for Zabbix events.



Displayed data:

Column	Description
Name	Name of the correlation rule. Clicking on the correlation rule name opens the rule configuration form.
Conditions	Correlation rule conditions are displayed.
Operations	Correlation rule operations are displayed.
Status	Correlation rule status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

To configure a new correlation rule, click on the *Create correlation* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

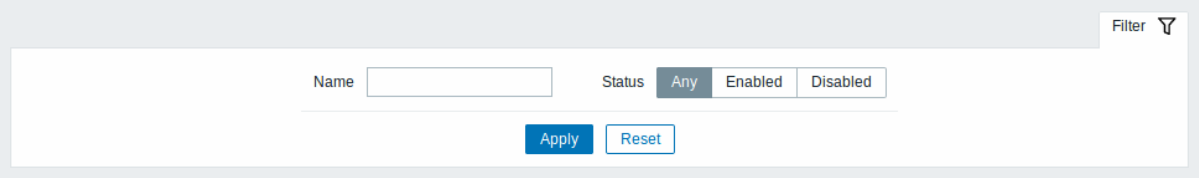
- *Enable* - change the correlation rule status to *Enabled*
- *Disable* - change the correlation rule status to *Disabled*
- *Delete* - delete the correlation rules

To use these options, mark the checkboxes before the respective correlation rules, then click on the required button.

Filter

As the list may contain a number of correlation rules, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of correlation rules. If you click on it, a filter becomes available where you can filter correlation rules by name and status.

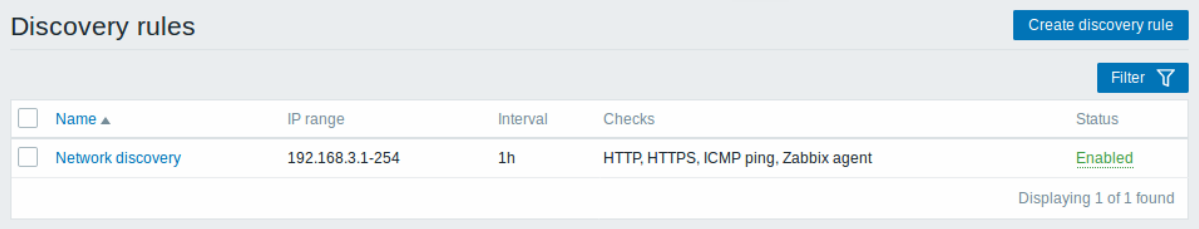


7 自动发现

概述

在 配置 **-** > 发现中用户可以配置和维护发现规则。

显示现有发现规则及其详细信息的列表。



显示数据：

列	述
名称发	规则的名称。单击发现规则名称将打开发现规则配置表.
IP 范围显	用于网络扫描的 IP 地址范围。
延迟显	执行发现的频率。
检查显	用于发现的检查类型。
状态显	动作状态 - 启用或者 禁止.\\点击状态可以更改。

点击右上角 创建发现规则按钮配置新的发现规则。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- 启用 - 将发现规则状态更改为 启用
- 禁用 - 将发现规则状态更改为 禁用

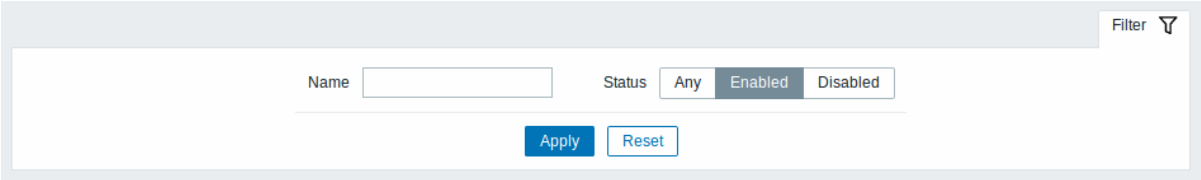
- 删除 - 删发现规则

要使用这些选项，请在相应的发现规则之前标记复选框，然后单击所需的按钮。

过滤器

由于列表可能包含许多发现规则，可能需要过滤出您真正需要的那些。

过滤器链接在发现规则列表之上。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤发现规则。

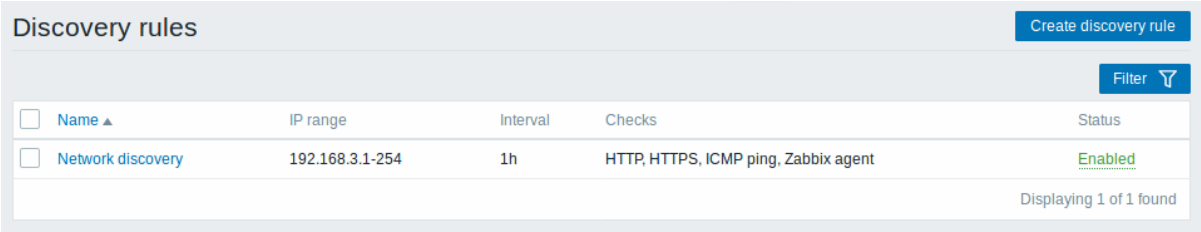


7 Discovery

Overview

In the *Configuration* → *Discovery* section users can configure and maintain discovery rules.

A listing of existing discovery rules with their details is displayed.



Displayed data:

Column	Description
Name	Name of the discovery rule. Clicking on the discovery rule name opens the discovery rule configuration form.
IP range	The range of IP addresses to use for network scanning is displayed.
Delay	The frequency of performing discovery displayed.
Checks	The types of checks used for discovery are displayed.
Status	Action status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

To configure a new discovery rule, click on the *Create discovery rule* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

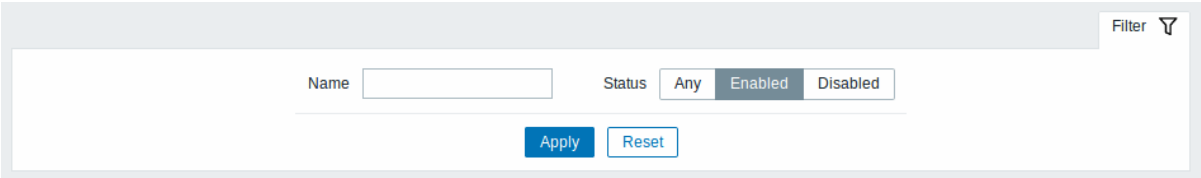
- *Enable* - change the discovery rule status to *Enabled*
- *Disable* - change the discovery rule status to *Disabled*
- *Delete* - delete the discovery rules

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Filter

As the list may contain a number of discovery rules, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of discovery rules. If you click on it, a filter becomes available where you can filter discovery rules by name and status.



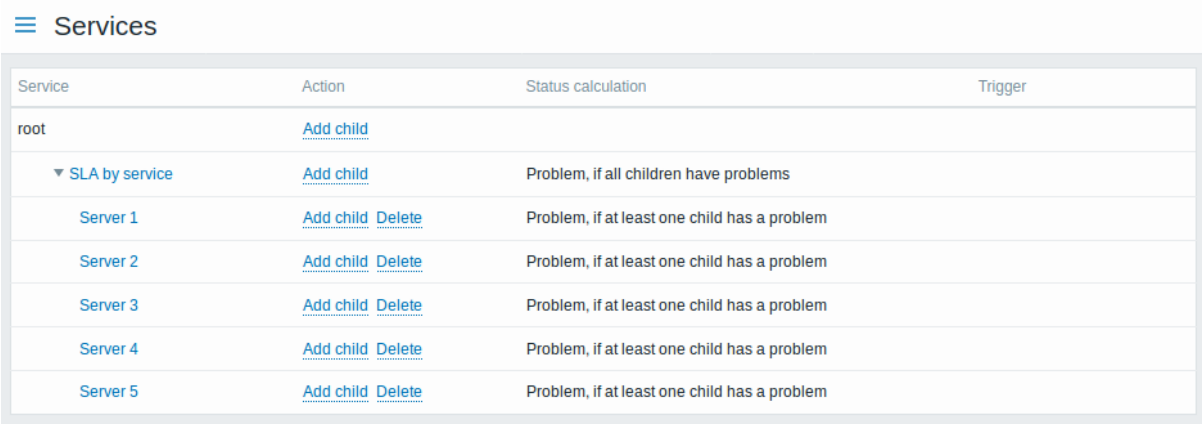
8 IT 服务

概述

在配置 **-** > *IT* 服务中用户可以配置和维护 IT 服务层次结构。

第一次打开此部分时，它只包含一个 *root* 入口.

您可以将其用作构建受监视基础架构层次结构的起点。点击 *Add child* 添加服务，然后在您添加的服务下添加其他服务。



F 有关添加服务的详细信息，请参阅IT 服务 模块.

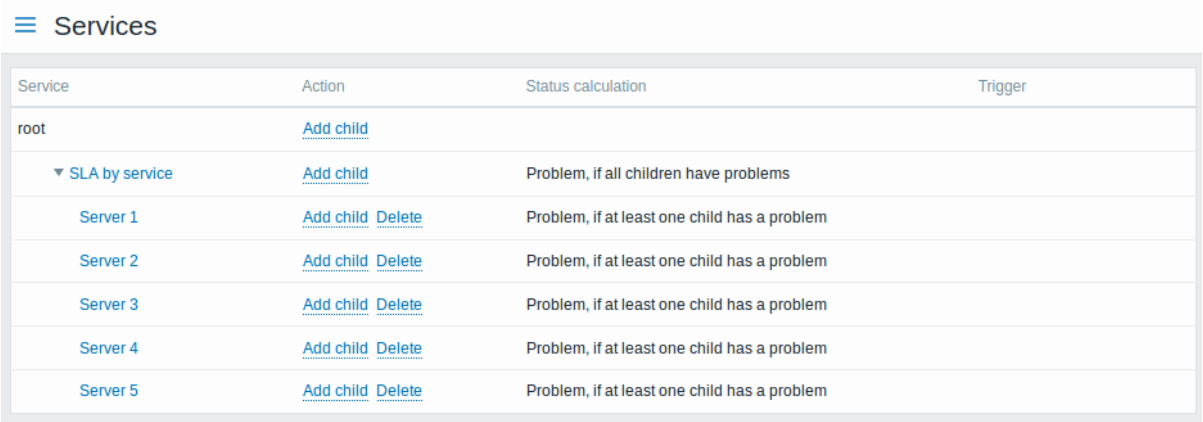
8 Services

Overview

In the *Configuration* → *Services* section users can configure and maintain an IT services hierarchy.

When you first open this section it only contains a *root* entry.

You can use it as a starting point of building the hierarchy of monitored infrastructure. Click on *Add child* to add services and then other services below the ones you have added.



For details on adding services, see the Service monitoring section.

5 管理

5 Administration 概述

“管理” 菜单用于 Zabbix 的管理功能。此菜单仅Super Administrators 适用。

Overview

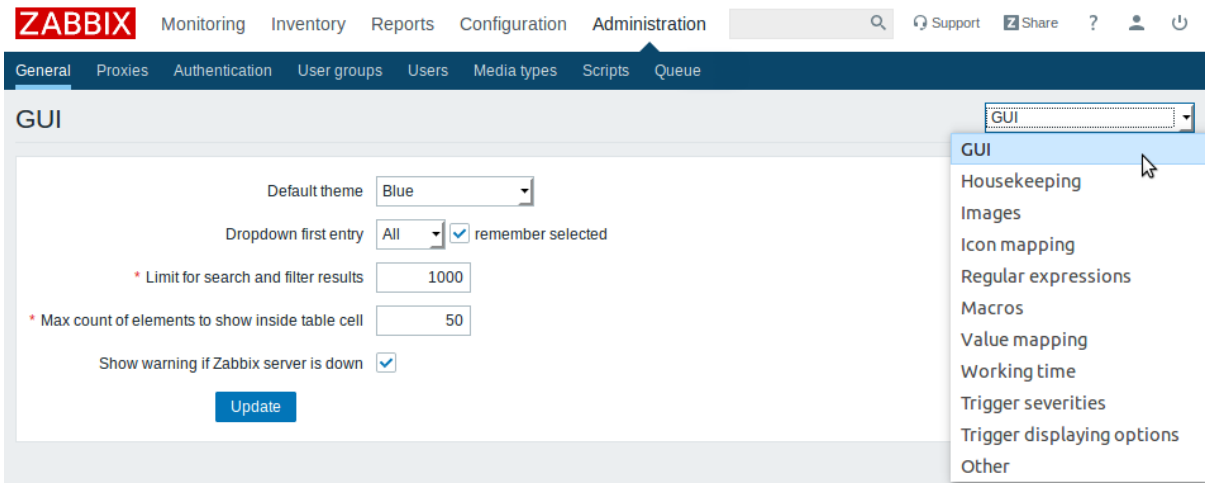
The Administration menu is for administrative functions of Zabbix. This menu is available to users of Super Administrators type only.

1 常规设置

概述

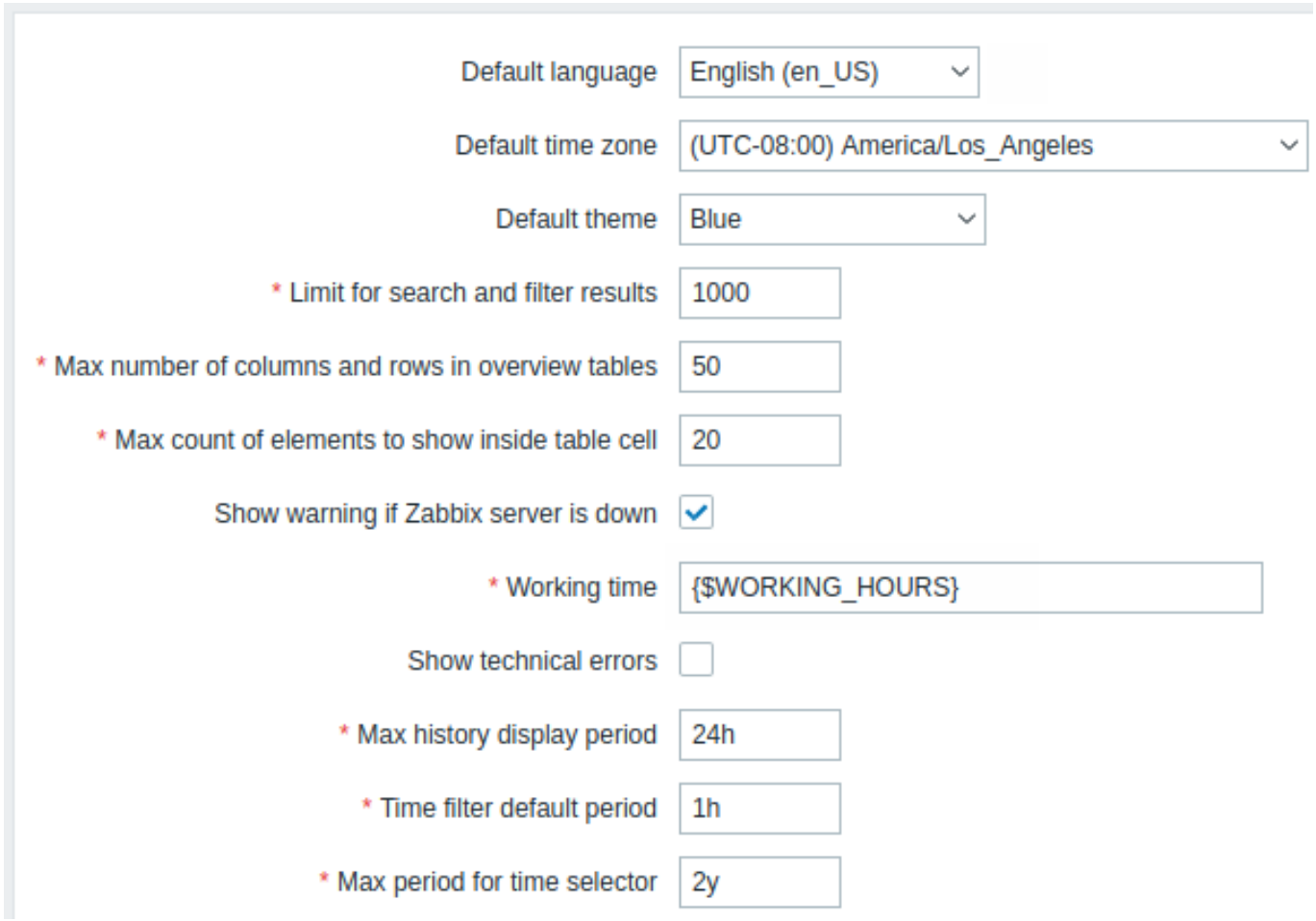
管理 - > 常规部分包含多个用于设置前端相关默认值和自定义 Zabbix 的屏幕。

右侧的下拉菜单允许您在不同的配置屏幕之间切换。



1 GUI

此屏幕提供了与前端相关的默认值的定制。



配置参数：

参数	描
默认主题没有在	个人资料中设置特定的用户的默认主题.

参数	描
下拉第一个入口不管元素选择	拉列表中的第一个条目是全部或是无.并且勾选了记住所选项,当导航到另一个页面时，下拉列表中的最后一个选定的元素将被记住(而不是默认值)。

参数	描
限制搜索和过滤结果将在 Web 界面列	中显示的元素(行)的最大数量，例如，在监控 -> 触发器或者配置 -> 主机.注意:如果设置为例如“50”，则前50个元素将仅显示在所有受影响的前端列表中。如果一些列

参数	描
最大元素数	对于单个在表格内显示格单元格中显示的条目，将不再显示此处配置的条目。
启用事件确认	此参数定义Zabbix界面中是否激活了事件确认发器状态”屏幕中显示多少天事件。默认为7天。
显示不久于.. 的时间	此参数定义在 “于 *(天数)*

参数	描
每个触发器显示的最大事件计数	触发状态屏幕中每个触发器的最大事件数。默认值为100。

参数	描
如果 *Zabbix* 服务器关闭，则显示警告	如果无法访问 Zabbix 务器(可能会关闭)，此参数将使浏览器窗口中显示警告消息。即使用户向下滚动页面，邮件仍然可见。如果鼠标移过它，该信息将被暂时隐藏以显示下面的内容。

参数描

2 Housekeeper

家是由 Zabbix 服务器执行的定期流程。该过程消除用户删除的过时信息和信息。

Events and alerts

Enable internal housekeeping ☑

* Trigger data storage period: 365d

* Internal data storage period: 1d

* Network discovery data storage period: 1d

* Auto-registration data storage period: 1d

Services

Enable internal housekeeping ☑

* Data storage period: 365d

Audit

Enable internal housekeeping ☑

* Data storage period: 365d

User sessions

Enable internal housekeeping ☑

* Data storage period: 365d

History

Enable internal housekeeping ☑

Override item history period ☐

* Data storage period: 90d

Trends

Enable internal housekeeping ☑

Override item trend period ☐

* Data storage period: 365d

Update | Reset defaults

在本节中，内容任务可以单独启用或禁用每个任务: 事件和警报/ IT 服务/审核/用户会话/历史/趋势。如果启用了家政管理，可以设置管理员被删除之前数据记录将保留多少天。

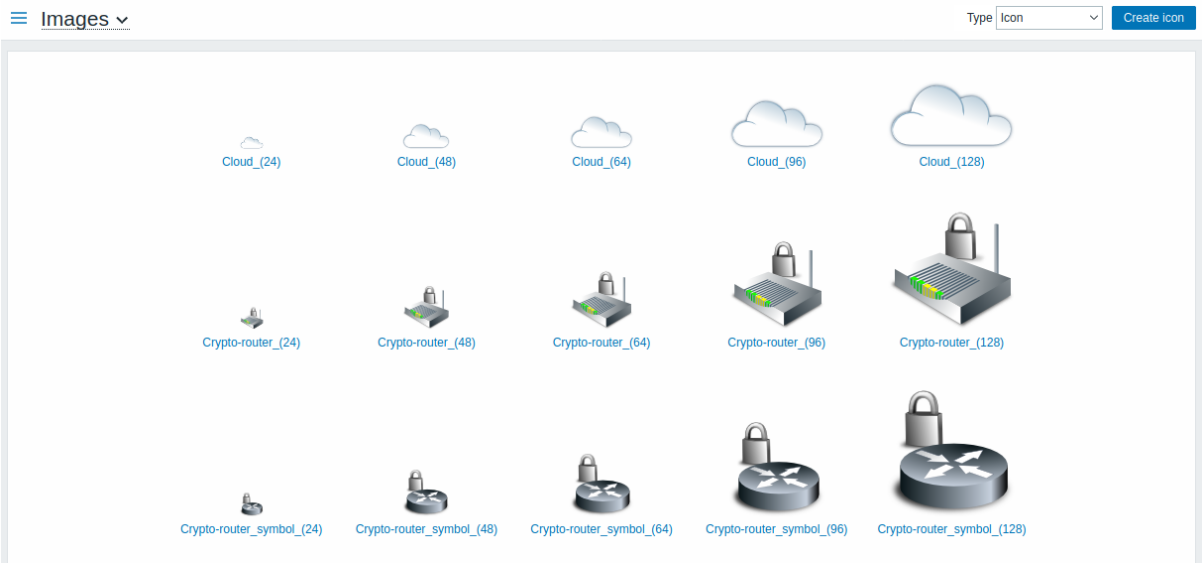
对于历史和趋势，还有其他选择：覆盖项目历史记录周期以及 覆盖项目趋势期. 此选项允许全局设置项目历史/趋势将保留多少天，在这种情况下，覆盖项目配置中保留历史记录/保留趋势字段中各个项目的值集。

即使禁用内部管家，也可以覆盖历史/趋势存储期。因此，当使用外部管家时，可以使用历史记录数据存储期间字段设置历史存储期。

恢复默认值按钮可以恢复所做的任何更改。

3 图片

图像部分显示 Zabbix 中可用的所有图像。图像存储在数据库中。

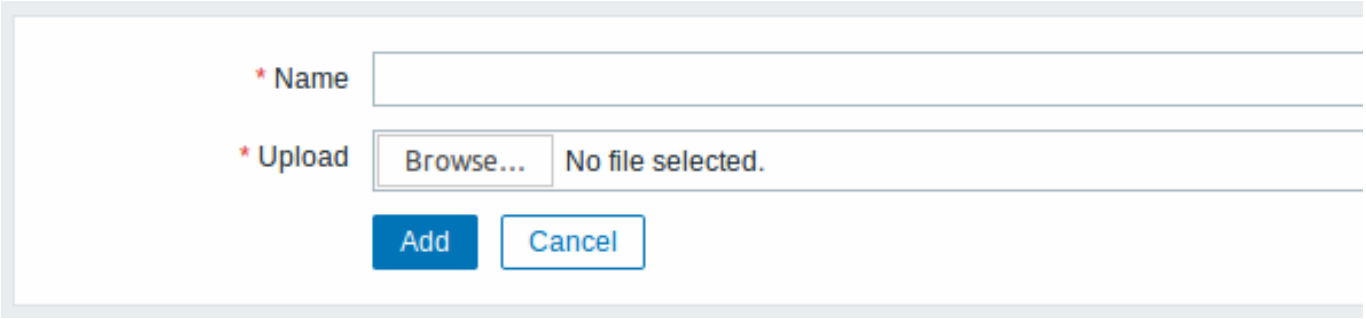


类型下拉菜单允许您在图标和背景图像之间切换：

- 标用于显示网络图 元素
- 背景用作网络图的背景图像

添加图像

您可以通过点击右上角创建图标或者创建背景按钮添加自己的图像。



图像属性：

参数描	n
名称图	的唯一名称。
上传从	地系统中选择要上传到 Zabbix 的文件（PNG，JPEG）。

Note:
上传文件的最大大小受 ZBX_MAX_IMAGE_SIZE 值的限制，为 1024x1024 字节或 1 MB。

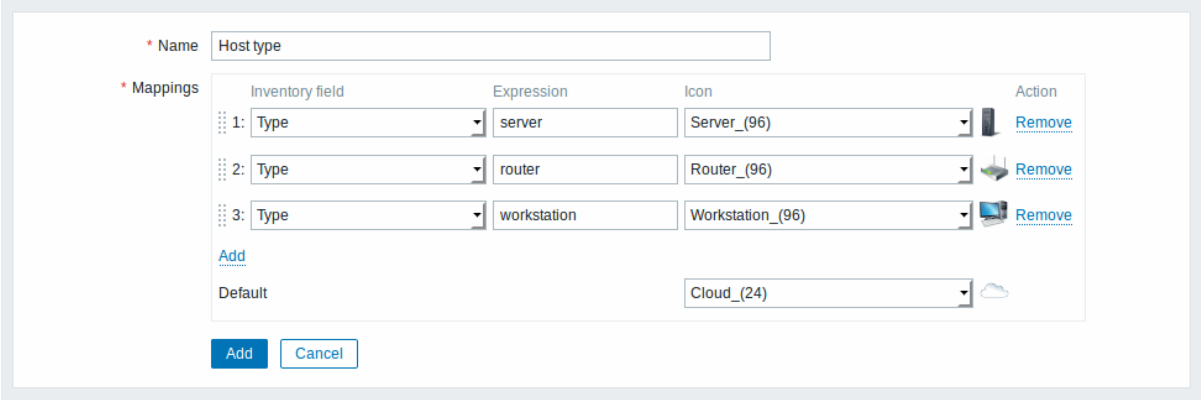
如果图像大小接近 1 MB，“max_allowed_packet” 的 MySQL 配置参数的默认值为 1MB，则图像的上传可能会失败。在这种情况下，增加 max_allowed_packet 参数。

4 图标映射

本部分允许使用某些图标创建某些主机的映射。主机清单字段信息用于创建映射。

然后可以使用映射网络地图配置 自动为匹配的主机分配适当的图标。

创建一个新的图标图，点击右上角的 创建图标地图。



配置参数：

参数描	
名称图	地图的唯一名称。
映射映	列表。映射顺序决定哪一个优先级。您可以使用拖放方式在列表上下移动映射。
库存字段将要查	一个匹配的主机库存字段。
表达式描述	配的正则表达式。
图标如	找到表达式的匹配，则使用图标。
默认要	用的默认图标。

5 正则表达式

此部分允许创建可在前端的多个位置使用的自定义正则表达式。参见正则表达式 细节。

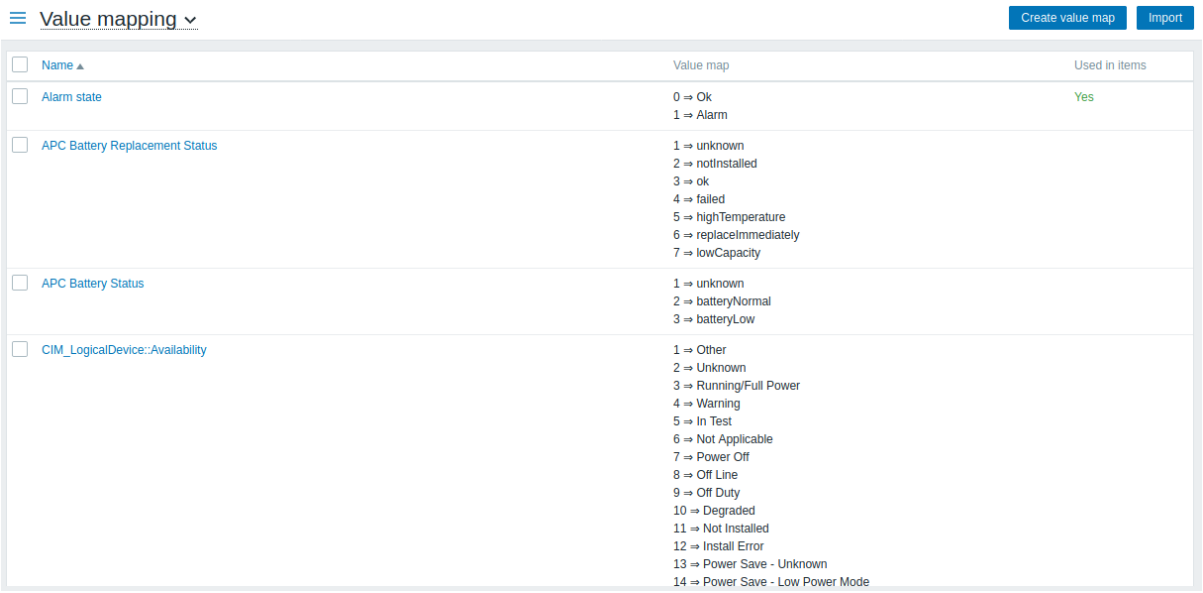
6 宏

本节允许定义系统范围的宏。

更多细节，参见用户宏。

7 值映射

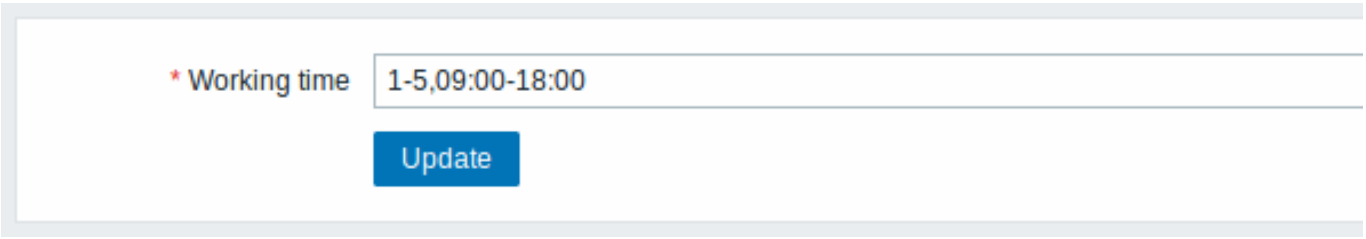
本部分允许管理对于 Zabbix 前端中输入数据的可读表示有用的值映射。



更多细节，参见值映射。

8 工作时间

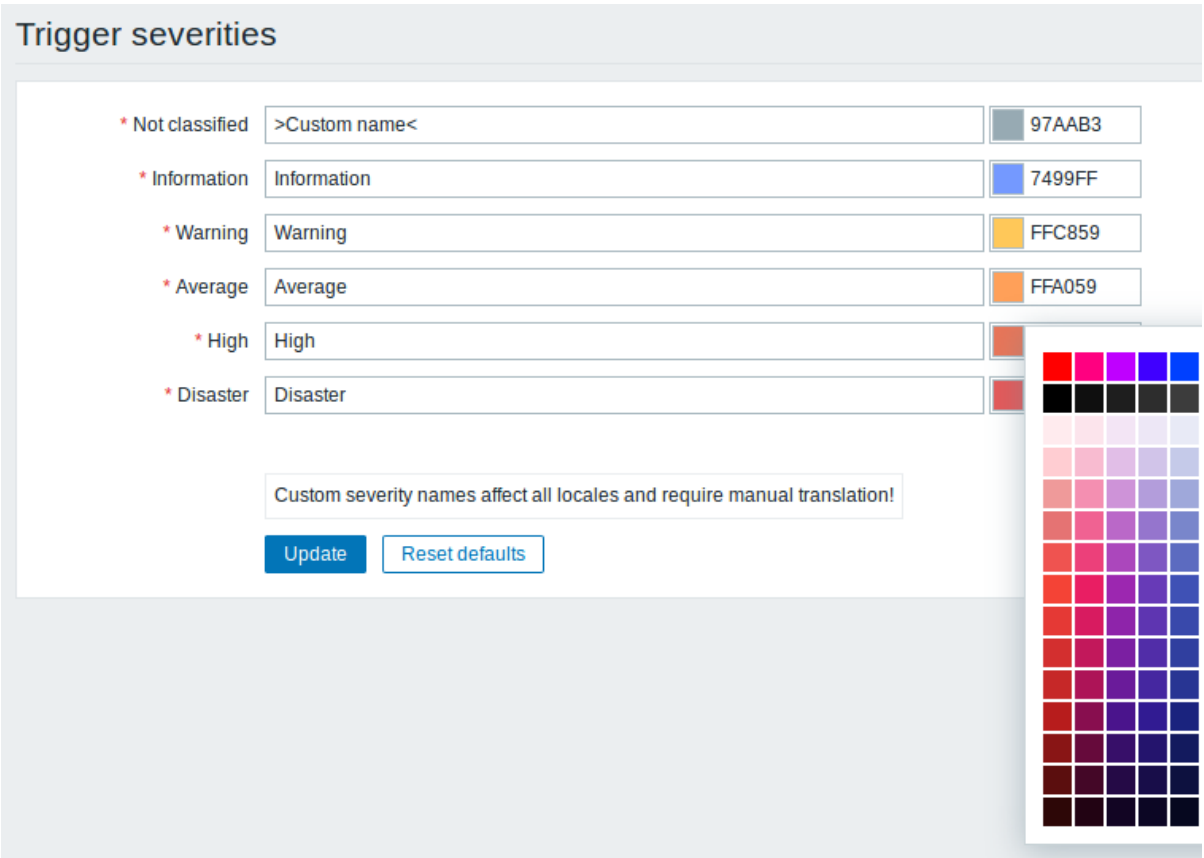
工作时间是系统范围的参数，用于定义工作时间。工作时间显示为图形中的白色背景，而非工作时间显示为灰色。



时间格式描述请参见时间段规格 页面。

9 触发严重级

此部分允许自定义触发严重级名称和颜色

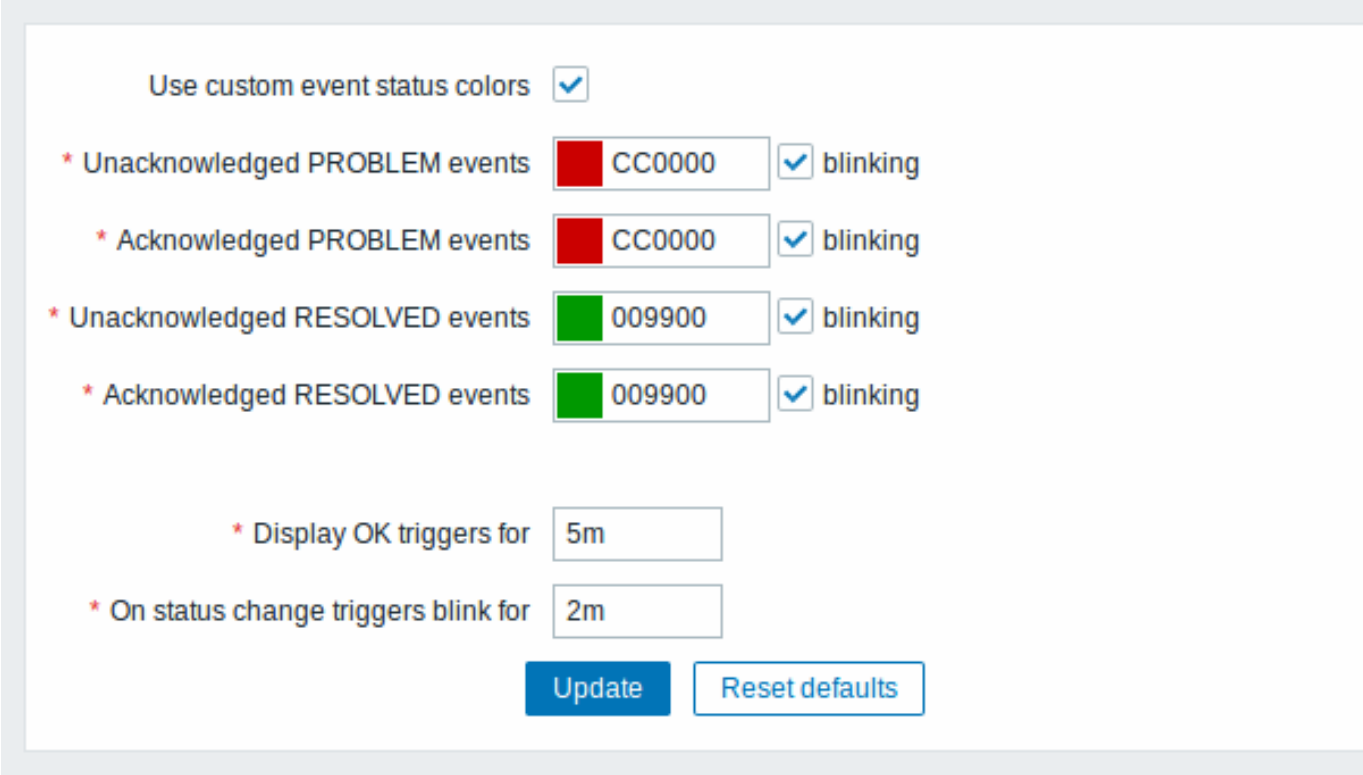


此部分允许自定义触发严重性您可以输入新的名称和颜色代码，或单击颜色以从提供的调色板中选择另一个。

更多信息，请参见自定义触发严重级页面。

10 触发显示选项

此部分允许自定义触发状态在前端中的显示方式。

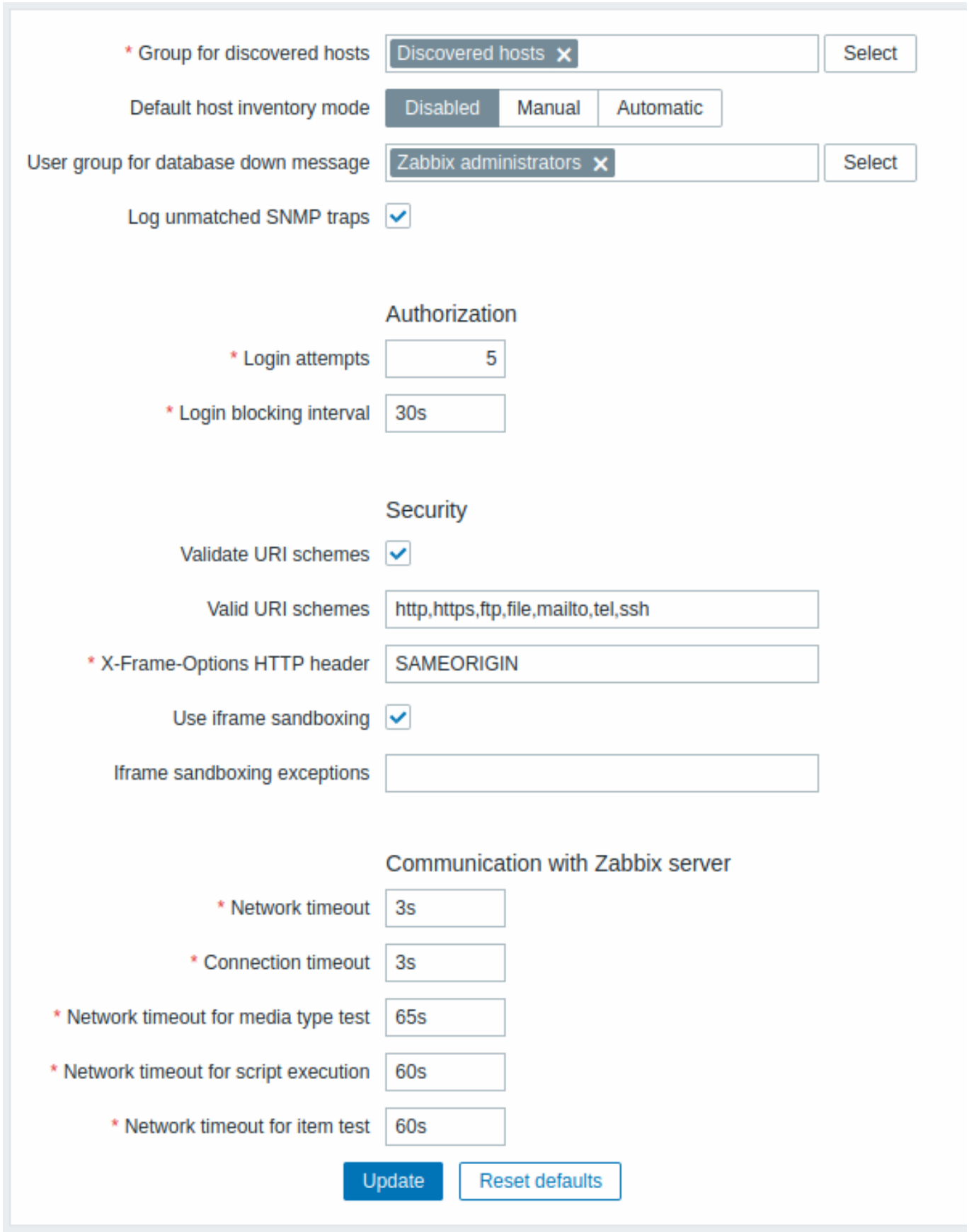


确认/未确认事件的颜色可以自定义并启用或禁用闪烁。

此外，可以定制显示 OK 触发的时间段和触发状态更改时的闪烁时间。最大值为 86400 秒（24 小时）。

11 其他参数

此部分允许配置其他前端参数。


* Group for discovered hosts
Discovered hosts
Select
Default host inventory mode
Disabled
Manual
Automatic
User group for database down message
Zabbix administrators
Select
Log unmatched SNMP traps
Authorization
* Login attempts
5
* Login blocking interval
30s
Security
Validate URI schemes
Valid URI schemes
http,https,ftp,file,mailto,tel,ssh
* X-Frame-Options HTTP header
SAMEORIGIN
Use iframe sandboxing
Iframe sandboxing exceptions
Communication with Zabbix server
* Network timeout
3s
* Connection timeout
3s
* Network timeout for media type test
65s
* Network timeout for script execution
60s
* Network timeout for item test
60s
Update
Reset defaults

参数	描
刷新不支持项（以秒为单位）由于用户参数错误或代理商	支持某些项目，某些项目可能会不受支持。Zabbix 可以配置为定期使不受支持的项目处于活动状态。Zabbix 将在此处设置 N 秒钟激活不受支持的项目。如果设置为 0，

参数	描
发现主机组被 [网络	现](/manual/discovery/和agent自动注册发现的主机将自动放置在主机组中，此处选择。

参数	描
默认主机库存模式	主机库存默认模式](/manual/config/hosts/host#configuration)。

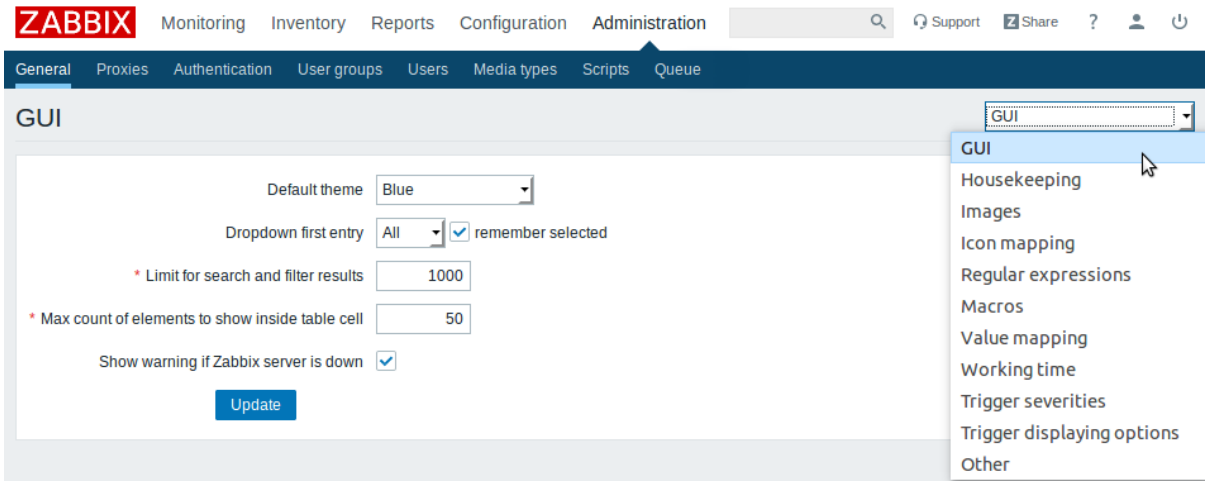
参数	描
Log unmatched SNMP traps	如果没有找到相应的SNMP接口。查看日志SNMP trap。

1 General

Overview

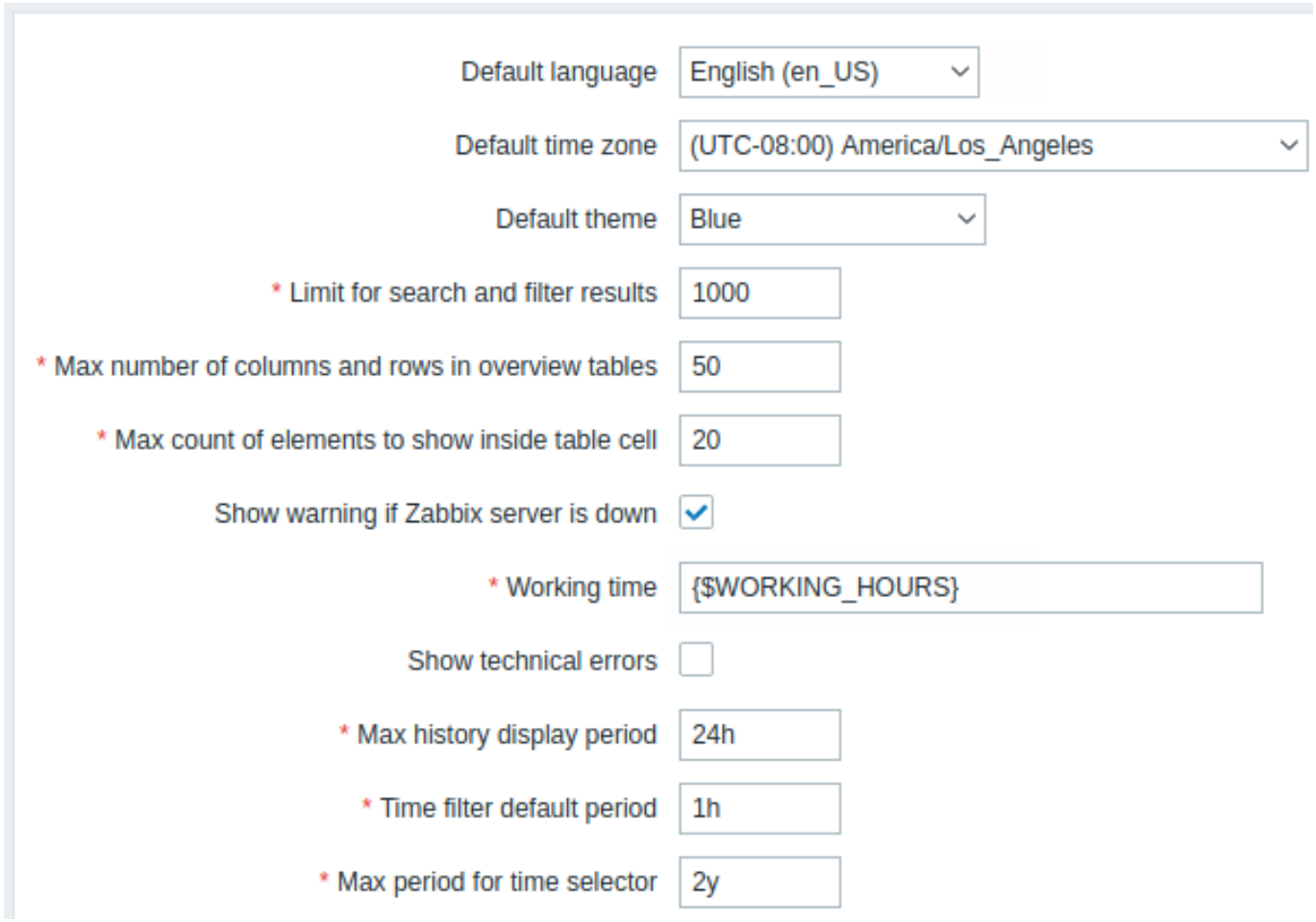
The *Administration* → *General* section contains a number of screens for setting frontend-related defaults and customizing Zabbix.

The dropdown to the right allows you to switch between different configuration screens.



12 GUI

This screen provides customization of several frontend-related defaults.



Configuration parameters:

Parameter	Description
Default theme	Default theme for users who have not set a specific one in their profiles.
Dropdown first entry	Whether first entry in element selection dropdowns should be *All* or *None*. With *remember selected* checked, the last selected element in the dropdown will be remembered (instead of the default) when navigating to another page.
Limit for search and filter results	Maximum amount of elements (rows) that will be displayed in a web-interface list, like, for example, in *Configuration* → *Hosts*. *Note*: If set to, for example, '50', only the first 50 elements will be displayed in all affected frontend lists. If some list contains more than fifty elements, the indication of that will be the '+' sign in *"Displaying 1 to 50 of **50+** found"*. Also, if filtering is used and still there are more than 50 matches, only the first 50 will be displayed.
Max count of elements to show inside table cell	For entries that are displayed in a single table cell, no more than configured here will be shown.
Show warning if Zabbix server is down	This parameter enables a warning message to be displayed in the browser window if Zabbix server cannot be reached (may be down). The message remains visible even if the user scrolls down the page. If the mouse is moved over it, the message is temporarily hidden to reveal the contents below. This parameter is supported since Zabbix 2.0.1.

13 Housekeeper

The housekeeper is a periodical process, executed by Zabbix server. The process removes outdated information and information deleted by user.

Events and alerts

Enable internal housekeeping ☑

* Trigger data storage period: 365d

* Internal data storage period: 1d

* Network discovery data storage period: 1d

* Auto-registration data storage period: 1d

Services

Enable internal housekeeping ☑

* Data storage period: 365d

Audit

Enable internal housekeeping ☑

* Data storage period: 365d

User sessions

Enable internal housekeeping ☑

* Data storage period: 365d

History

Enable internal housekeeping ☑

Override item history period ☐

* Data storage period: 90d

Trends

Enable internal housekeeping ☑

Override item trend period ☐

* Data storage period: 365d

Update | Reset defaults

In this section housekeeping tasks can be enabled or disabled on a per-task basis separately for: events and alerts/IT services/audit/user sessions/history/trends. If housekeeping is enabled, it is possible to set for how many days data records will be kept before being removed by the housekeeper.

Deleting an item/trigger will also delete problems generated by that item/trigger.

Also, an event will only be deleted by the housekeeper if it is not associated with a problem in any way. This means that if an event is either a problem or recovery event, it will not be deleted until the related problem record is removed. The housekeeper will delete problems first and events after, to avoid potential problems with stale events or problem records.

For history and trends an additional option is available: *Override item history period* and *Override item trends period*. This option allows to globally set for how many days item history/trends will be kept, in this case overriding the values set for individual items in *Keep history/Keep trends* fields in item configuration.

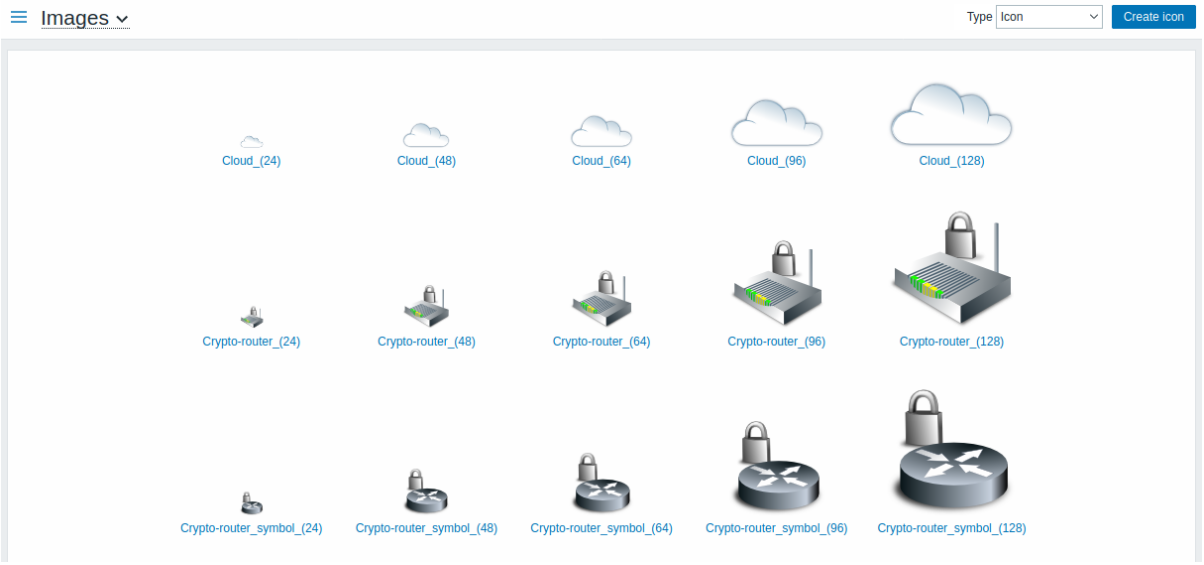
It is possible to override the history/trend storage period even if internal housekeeping is disabled. Thus, when using an external housekeeper, the history storage period could be set using the history *Data storage period* field.

Time suffixes are supported in the period fields, e.g. 1d (one day), 1w (one week). Minimum is 1 day (1 hour for history), maximum 25 years.

Reset defaults button allows to revert any changes made.

14 Images

The Images section displays all the images available in Zabbix. Images are stored in the database.



The *Type* dropdown allows you to switch between icon and background images:

- Icons are used to display network map elements
- Backgrounds are used as background images of network maps

Adding image

You can add your own image by clicking on the *Create icon* or *Create background* button in the top right corner.

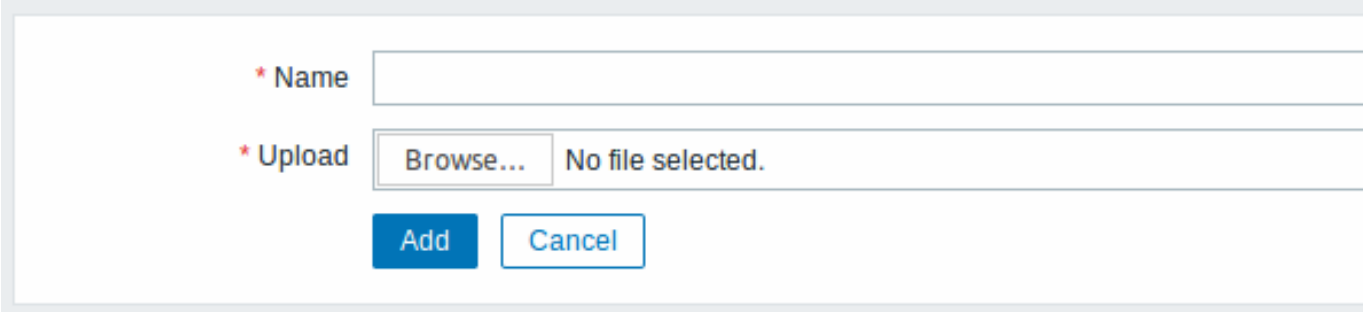


Image attributes:

Parameter	Description
Name	Unique name of an image.

Parameter	Description
Upload	Select the file (PNG, JPEG) from a local system to be uploaded to Zabbix.

Note:
Maximum size of the upload file is limited by value of ZBX_MAX_IMAGE_SIZE that is 1024x1024 bytes or 1 MB.

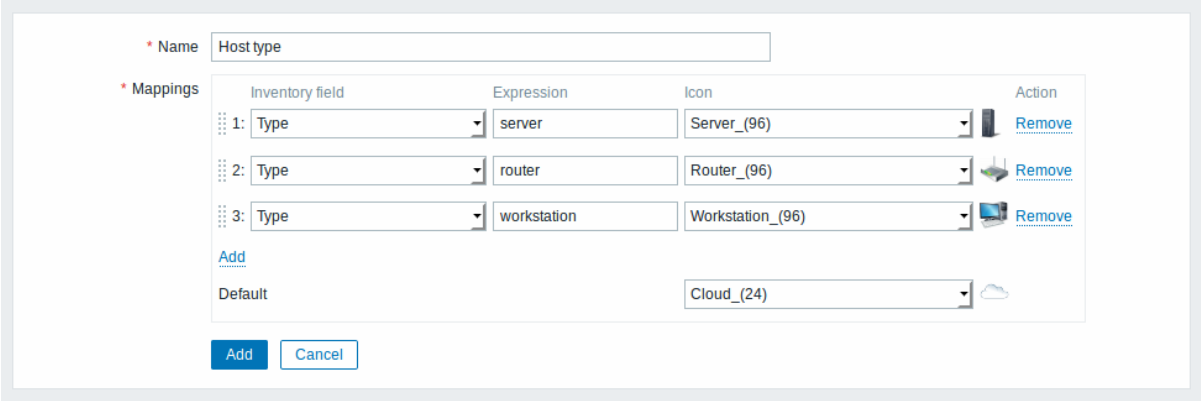
The upload of an image may fail if the image size is close to 1 MB and the `max_allowed_packet` MySQL configuration parameter is at a default of 1MB. In this case, increase the max_allowed_packet parameter.

15 Icon mapping

This section allows to create the mapping of certain hosts with certain icons. Host inventory field information is used to create the mapping.

The mappings can then be used in network map configuration to assign appropriate icons to matching hosts automatically.

To create a new icon map, click on *Create icon map* in the top right corner.



Configuration parameters:

Parameter	Description
Name	Unique name of icon map.
Mappings	A list of mappings. The order of mappings determines which one will have priority. You can move mappings up and down the list with drag-and-drop.
Inventory field	Host inventory field that will be looked into to seek a match.
Expression	Regular expression describing the match.
Icon	Icon to use if a match for the expression is found.
Default	Default icon to use.

16 Regular expressions

This section allows to create custom regular expressions that can be used in several places in the frontend. See Regular expressions section for details.

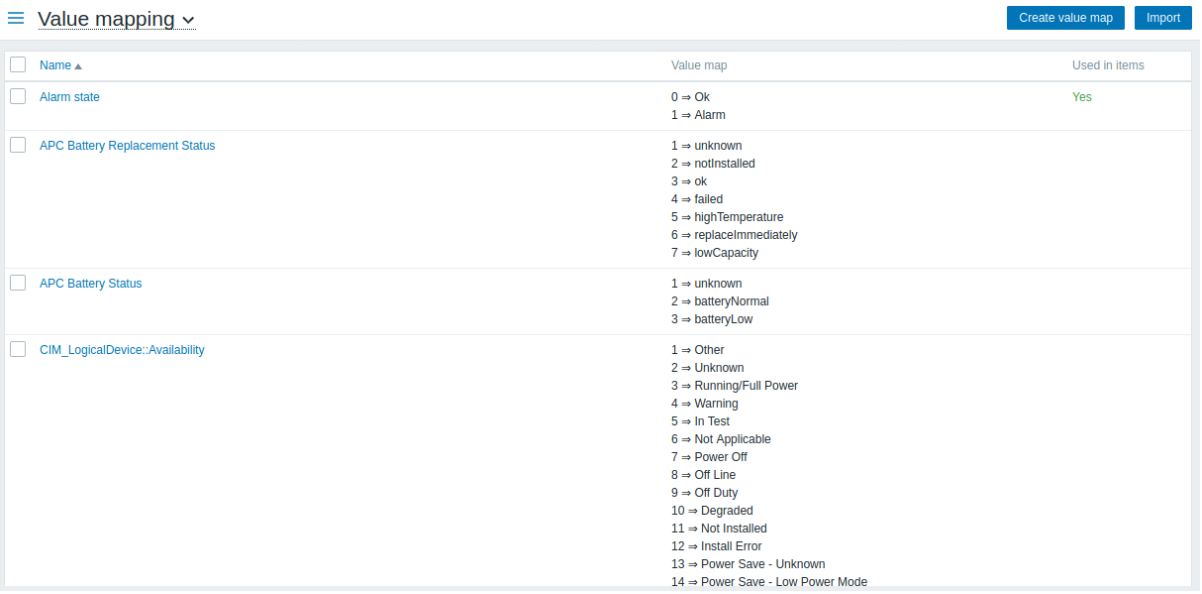
17 Macros

This section allows to define system-wide macros.

See User macros section for more details.

18 Value mapping

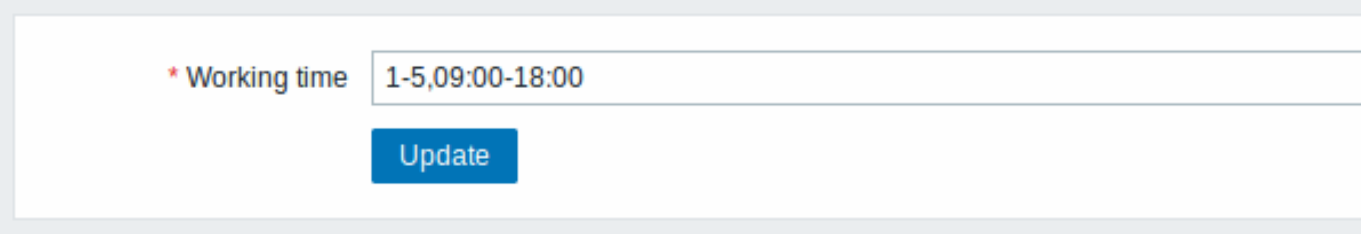
This section allows to manage value maps that are useful for human-readable representation of incoming data in Zabbix frontend.



See Value mapping section for more details.

19 Working time

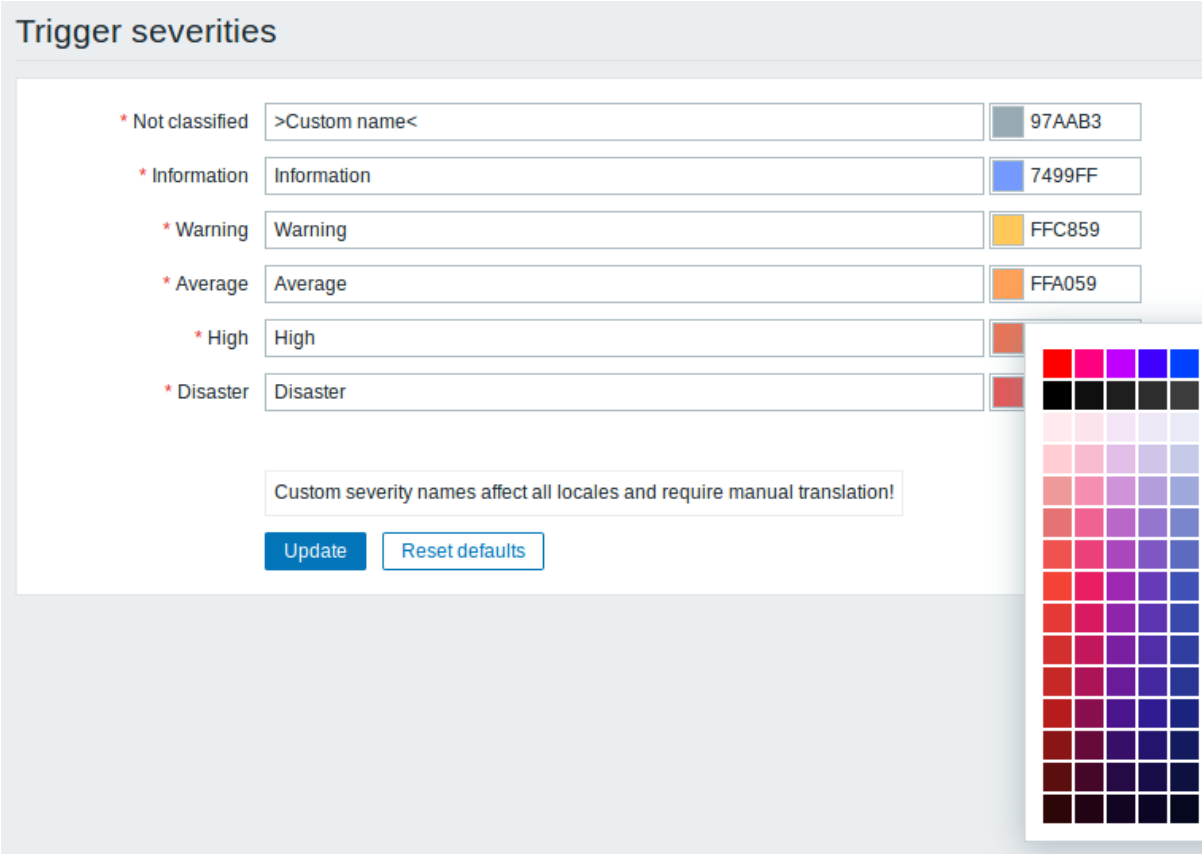
Working time is system-wide parameter, which defines working time. Working time is displayed as a white background in graphs, while non-working time is displayed in grey.



See Time period specification page for description of the time format. User macros are supported (since Zabbix 3.4.0).

20 Trigger severities

This section allows to customize trigger severity names and colors.

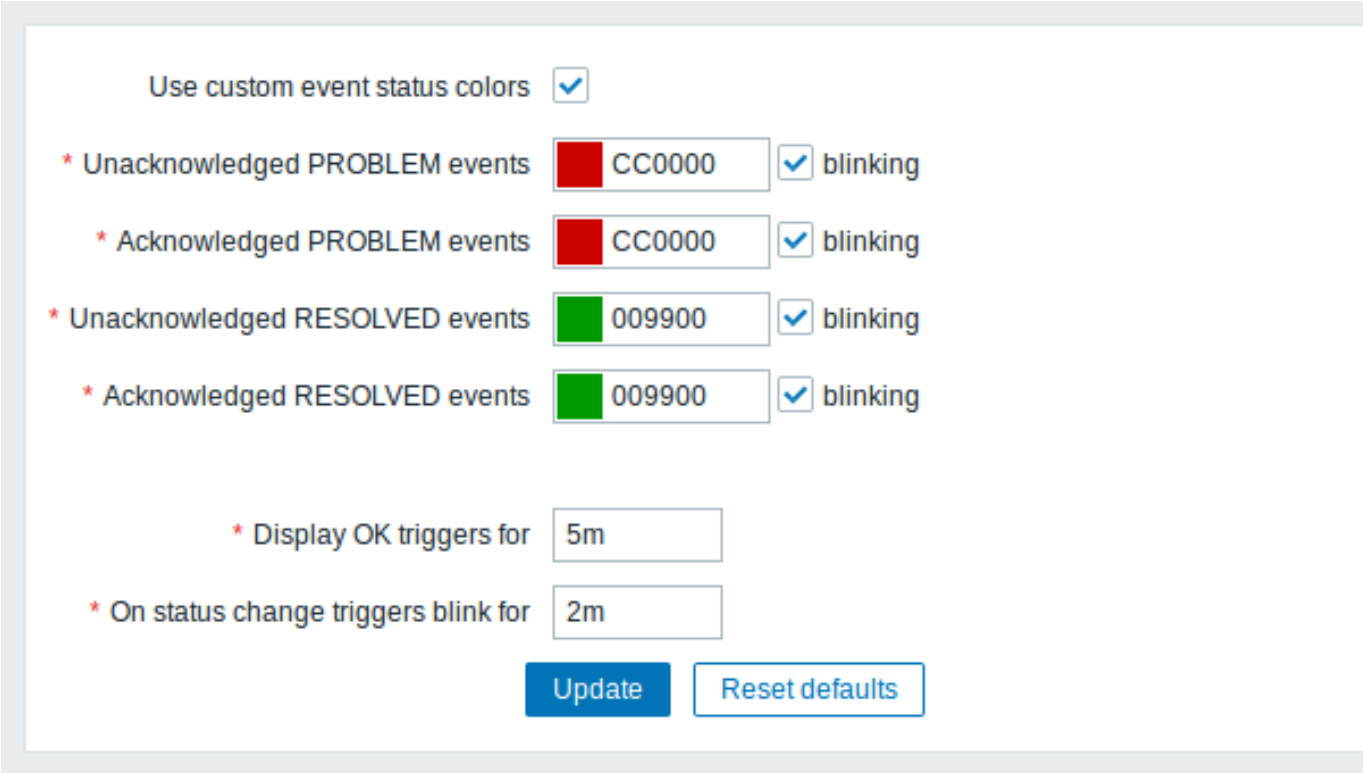


You can enter new names and color codes or click on the color to select another from the provided palette.

See Customising trigger severities page for more information.

21 Trigger displaying options

This section allows to customize how trigger status is displayed in the frontend.

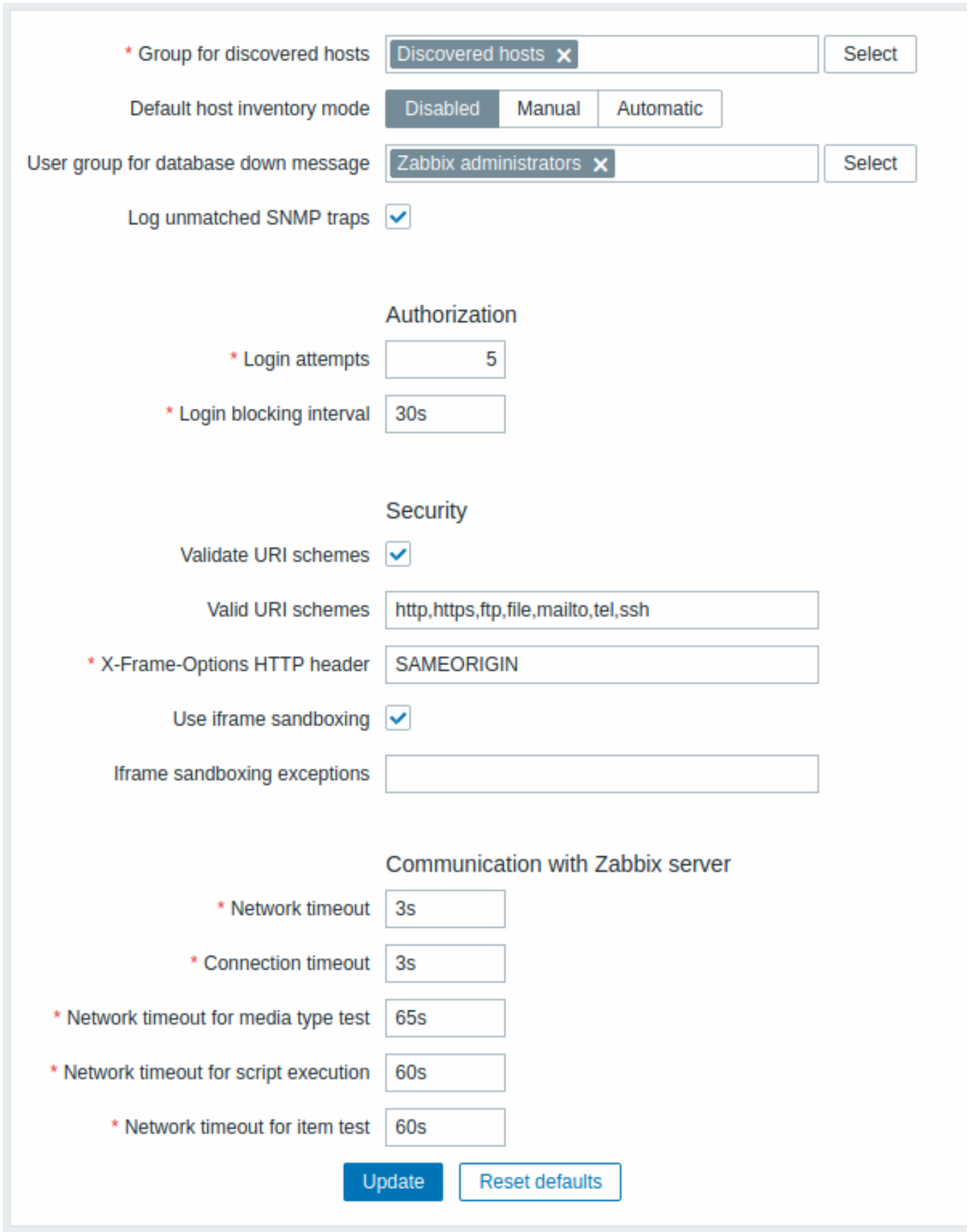


Checking *"Use custom event status colors"* checkbox enables customization of the colors for acknowledged/unacknowledged events. Unchecking this checkbox disables this customization, respectively. Blinking isn't affected by this checkbox.

Also the time period for displaying OK triggers and for blinking upon trigger status change can be customized. The maximum value is 86400 seconds (24 hours). Time suffixes are supported in the period fields, e.g. 5m, 2h, 1d.

22 Other parameters

This section allows to configure several other frontend parameters.



Parameter	Description
Refresh unsupported items	Some items may become unsupported due to errors in user parameters or because of an item not being supported by agent. Zabbix can be configured to periodically make unsupported items active. Zabbix server will activate unsupported item every N period set here (1 day maximum). If set to 0, the automatic activation will be disabled. Time suffixes are supported, e.g. 60s, 5m, 2h, 1d. The configured value also applies to how often Zabbix proxies reactivate unsupported items.
Group for discovered hosts	Hosts discovered by network discovery and agent auto-registration will be automatically placed in the host group, selected here.

Parameter	Description	
Default host inventory mode	Default mode for host inventory. It will be followed whenever a new host or host prototype is created by server or frontend, unless overriden during host discovery/auto registration by the //Set host inventory mode *operation.* | |User group for database down message//	User group for sending alarm message or 'None'. Zabbix server depends on the availability of backend database. It cannot work without a database. If the database is down, selected users can be notified by Zabbix. Notifications will be sent to the user group set here using all configured user media en-

Parameter	Description
Log unmatched SNMP traps	Log SNMP trap if no corresponding SNMP interfaces have been found.

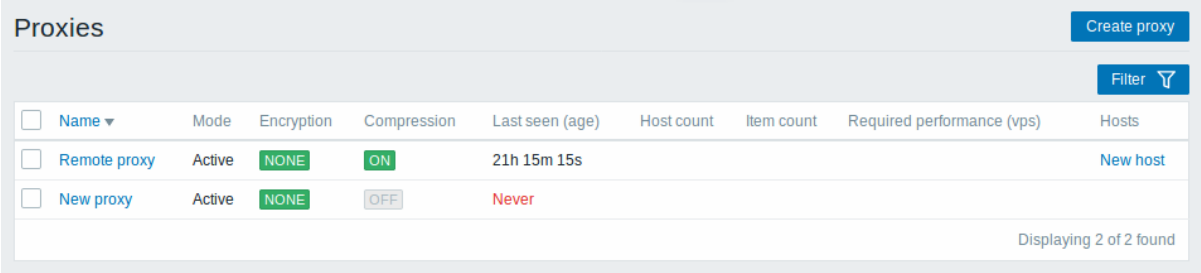
2 Proxies

概述

在 Administration → Proxies 里，分布式监控可以在 Zabbix 前端进行配置。

Proxies

显示现有 proxy 列表及其详细信息



显示的信息：

Column	描述
Name	Proxy 名称。点击 proxy 名可以打开当前 proxy 配置表单.
Mode	显示 Proxy 的模式 - Active 或者 Passive.
Encryption	显示来自 proxy 的连接的加密状态: **None** - 不加密 **PSK** -使用 PSK 方式 **Cert** -使用证书
Last seen (age)	显示 sever 上次看到 agent 的时间
Host count	显示被 proxy 监控的 host 数量
Item count	显示被 proxy 监控的监控项的数量。
Required performance (vps)	显示所需的 proxy 性能（每秒需要收集的值的数量）。
Hosts	列出由 proxy 监控的所有主机。单击主机名将打开主机配置表单。

配置新的 proxy，请单击顶部右上角的 Create proxy 按键。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

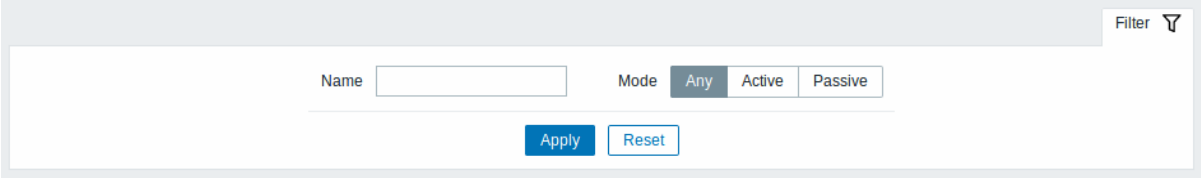
- *Enable hosts* - 将被 proxy 监控的 host 的状态改为 Monitored （监控）
- *Disable hosts* -将被 proxy 监控的 host 的状态改为 Not monitored （不监控）
- *Delete* - 删除 proxy

要使用这些选项，请在各个 proxy 之前标记复选框，然后单击您需要的按键。

过滤器

因为列表中可能包含许多 proxy，所以可能需要通过过滤得到您需要的内容。

Filter 过滤器链接位于 agent 列表之上。如果您点击它，则可以使用过滤器，您可以通过名称和模式过滤 proxy。



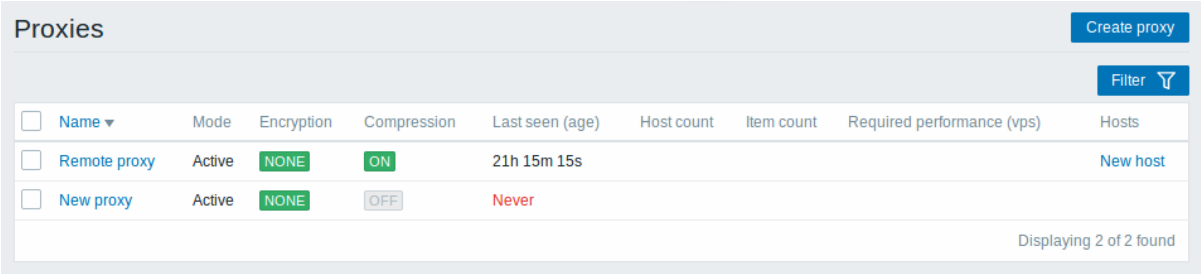
2 Proxies

Overview

In the *Administration* → *Proxies* section proxies for distributed monitoring can be configured in the Zabbix frontend.

Proxies

A listing of existing proxies with their details is displayed.



Displayed data:

Column	Description
Name	Name of the proxy. Clicking on the proxy name opens the proxy configuration form.
Mode	Proxy mode is displayed - *Active* or *Passive*.
Encryption	Encryption status for connections from the proxy is displayed: **None** - no encryption **PSK** - using pre-shared key **Cert** - using certificate
Last seen (age)	The time when the proxy was last seen by the server is displayed.
Host count	The number of enabled hosts assigned to the proxy is displayed.
Item count	The number of enabled items on enabled hosts assigned to the proxy is displayed.
Required performance (vps)	Required proxy performance is displayed (the number of values that need to be collected per second).
Hosts	All hosts monitored by the proxy are listed. Clicking on the host name opens the host configuration form.

To configure a new proxy, click on the *Create proxy* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

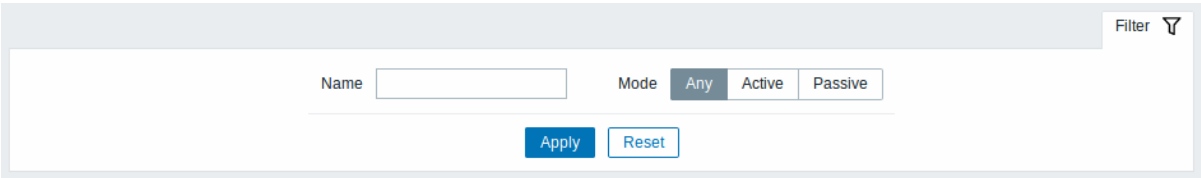
- *Enable hosts* - change the status of hosts monitored by the proxy to *Monitored*
- *Disable hosts* - change the status of hosts monitored by the proxy to *Not monitored*
- *Delete* - delete the proxies

To use these options, mark the checkboxes before the respective proxies, then click on the required button.

Filter

As the list may contain many proxies, it may be needed to filter out the ones you really need.

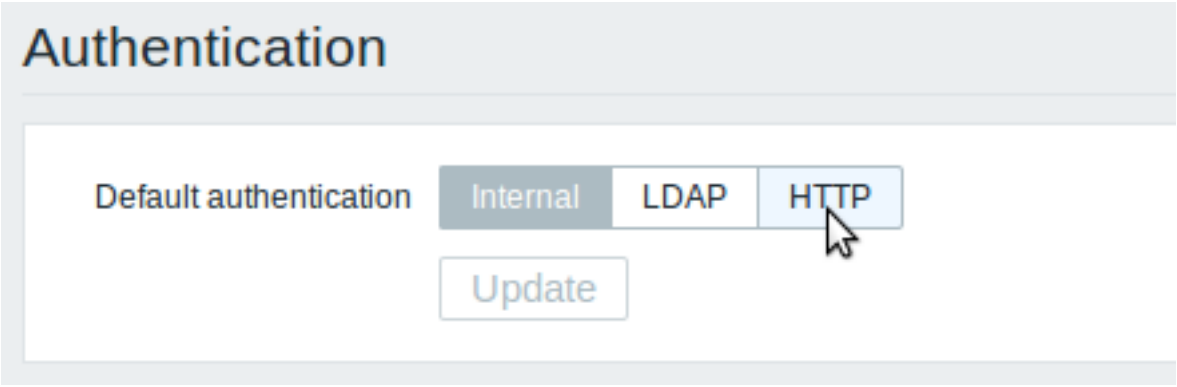
The *Filter* link is available above the list of proxies. If you click on it, a filter becomes available where you can filter proxies by name and mode.



3 身份认证

概述

在 Administration → Authentication 中，可以改变 Zabbix 用户身份认证方法。可用的方法为：内部认证（internal），LDAP 和 HTTP 认证。



默认情况下，使用内部 Zabbix 认证。要更改的话，请点击认需要选择的证方法按钮，然后按 Update 更新

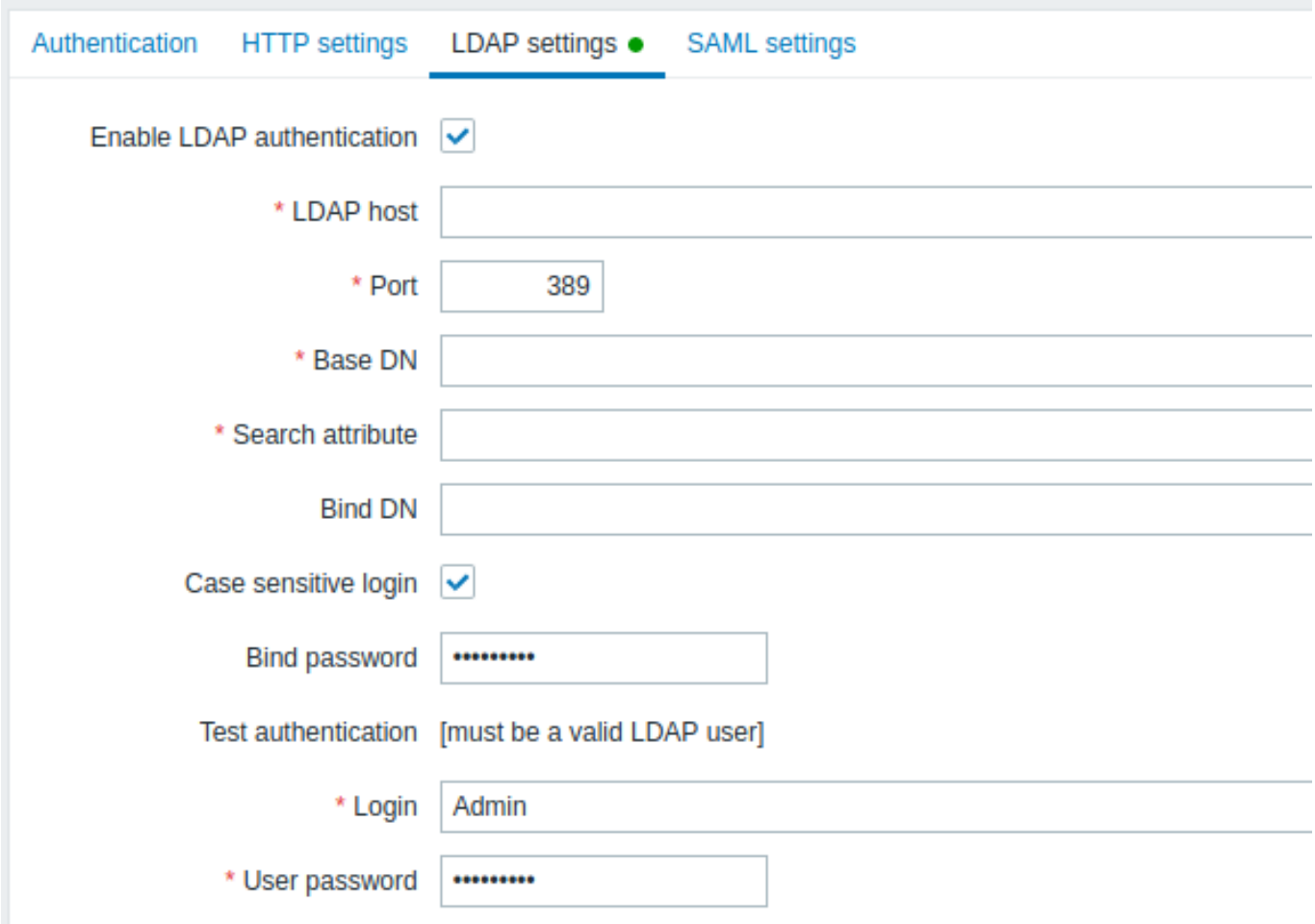
Internal

使用内部 Zabbix 认证。

LDAP

外部 LDAP 认证可用于检查用户名和密码。请注意，该用户也必须存在于 Zabbix 中，但是它的 Zabbix 密码将不会被使用

Zabbix LDAP 验证至少要与 Microsoft Active Directory 和 OpenLDAP 一起工作



配置参数：

参数 r 描	
LDAP host	LDAP 服务器名称。例如：ldap://ldap.zabbix 安全 LDAP 服务器使用 ldaps 协议。ldaps://ldap.zabb
Port	LDAP 服务器接口，默认为 389。\\安全 LDAP 连接端口号一般为 636。

参数 r 描	
Base DN	寻找账户的基本路径：ou=Users,ou=sys (for OpenLDAP), DC=company,DC (for Microsoft Active Directory)
Search attribute	用户搜索的 LDAP 账户属性：uid (for OpenLDAP), sAMAccountNam (for Microsoft Active Directory)

参数 r 描	
Bind DN	通过 LDAP 服务器进行绑定和搜索的 LDAP 帐户，例如：uid=ldap_search, (for OpenL-DAP), CN=ldap_search, (for Mi-crosoft Ac-tive Di-rec-tory) Required, 匿名绑定目前不支持。
Bind password	通过 LDAP 服务器进行绑定和搜索的 LDAP 账户密码。

参数 r 描	
Test authentication	测试部分的标题
Login	测试用户名称(当前 Zabbix 前端登录的). 用户名必须在 LDAP 服务器上存在。. 如果无法验证测试用户，Zabbix 将不会激活 LDAP 身份验证。

参数 r 描	
User password	测试用户的 LDAP 密码。

<note tip> 建议创建一个单独的 LDAP 帐户（绑定 DN），以 LDAP 中的最小权限执行绑定和搜索，而不使用真正的用户帐户（用于登录 Zabbix 前端）。
这种方法提供更多的安全性，并且用户在 LDAP 服务器中更改密码时，不需要更改 Bind password 绑定密码。在上表中，ldap_search 是帐号名。:::

Note:
某些用户组仍然可以由 Zabbix 授权。这些组必须具有内部的前端访问设置为内部认证）将被 Apache 授权，而不是由 Zabbix 授权！

HTTP

可以使用基于 Apache（HTTP）的身份验证来检查用户名和密码。请注意，用户也必须存在于 Zabbix 中，但是它的 Zabbix 密码将不会被使用。

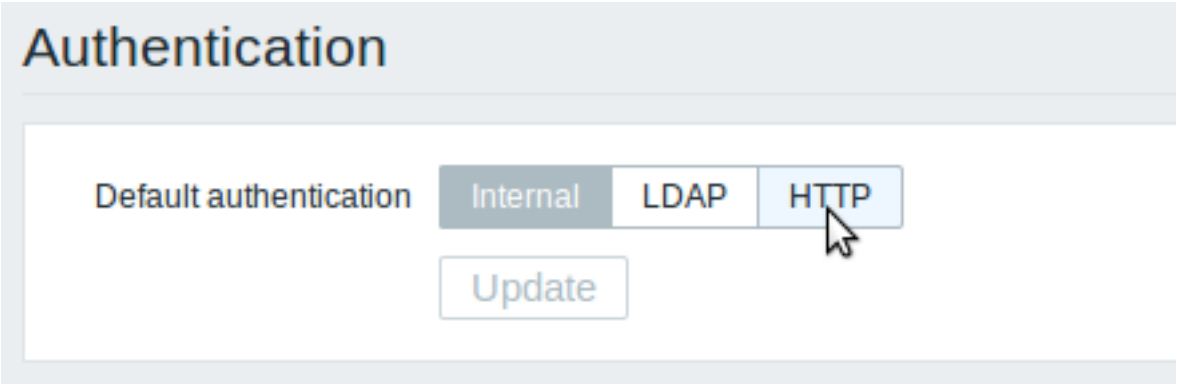
Attention:
小心！确保 Apache 身份验证已配置并正常工作，然后再打开它。

Note:
在 Apache 身份认证验证的情况下，所有用户（即使前端访问 设置为内部认证）将被 Apache 授权，而不是由 Zabbix 授权！

3 Authentication

Overview

In *Administration* → *Authentication* the user authentication method to Zabbix can be changed. The available methods are internal, LDAP and HTTP authentication.



By default, internal Zabbix authentication is used. To change, click on the button with the method name and press *Update*.

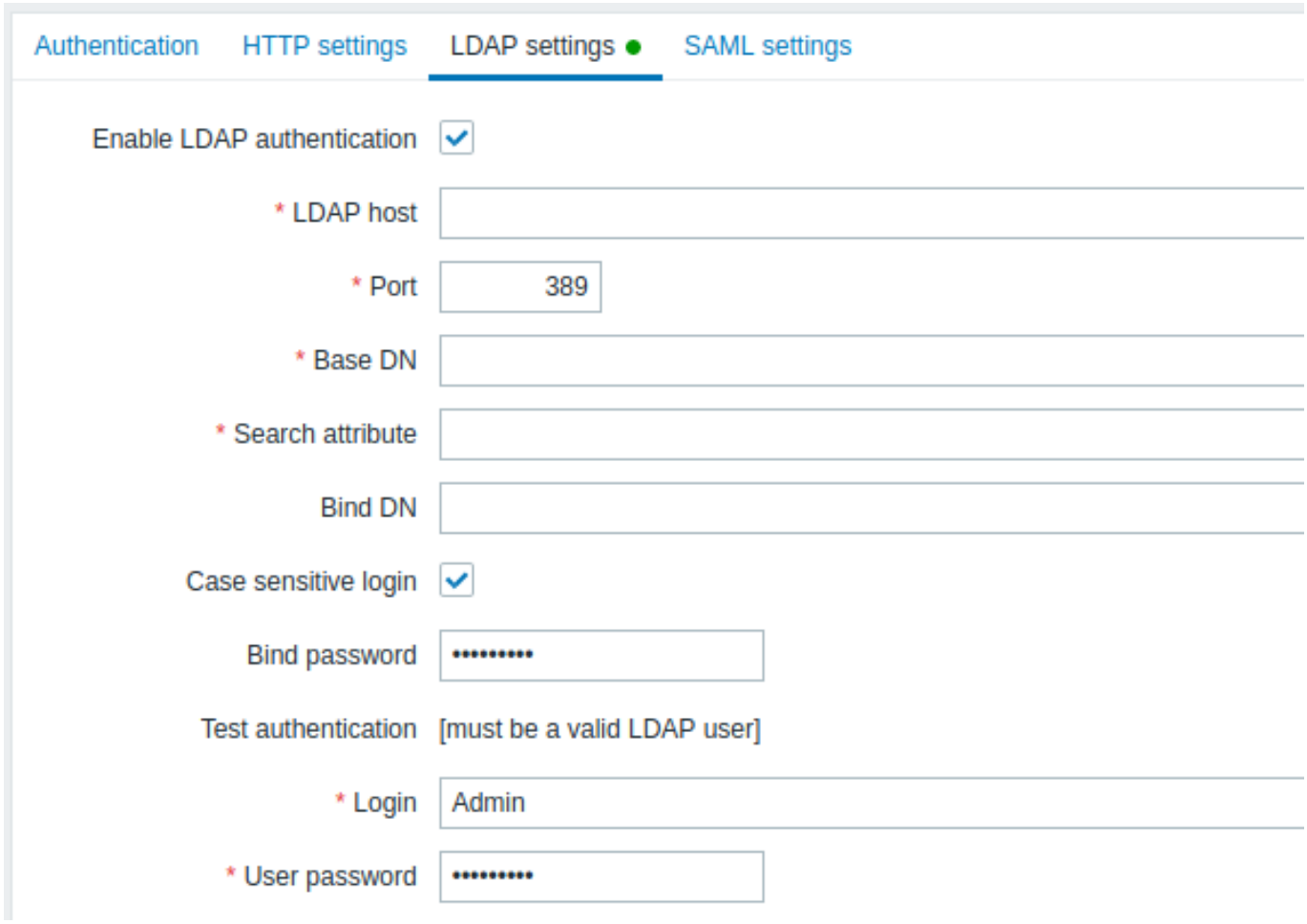
Internal

Internal Zabbix authentication is used.

LDAP

External LDAP authentication can be used to check user names and passwords. Note that a user must exist in Zabbix as well, however its Zabbix password will not be used.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.



Configuration parameters:

Parameter	Description
LDAP host	Name of LDAP server. For example: ldap://ldap.zabbix.com For secure LDAP server use *ldaps* protocol. ldaps://ldap.zabbix.com With OpenLDAP 2.x.x and later, a full LDAP URI of the form ldap://hostname:port or ldaps://hostname:port may be used.
Port	Port of LDAP server. Default is 389. For secure LDAP connection port number is normally 636. Not used when using full LDAP URIs.
Base DN	Base path to search accounts: ou=Users,ou=system (for OpenLDAP), DC=company,DC=com (for Microsoft Active Directory)
Search attribute	LDAP account attribute used for search: uid (for OpenLDAP), sAMAccountName (for Microsoft Active Directory)
Bind DN	LDAP account for binding and searching over the LDAP server, examples: uid=ldap_search,ou=system (for OpenLDAP), CN=ldap_search,OU=user_group,DC=company,DC=com (for Microsoft Active Directory) Required, anonymous binding is not supported.
Bind password	LDAP password of the account for binding and searching over the LDAP server.
Test authentication	Header of a section for testing
Login	Name of a test user (which is currently logged in the Zabbix frontend). This user name must exist in the LDAP server. Zabbix will not activate LDAP authentication if it is unable to authenticate the test user.

Parameter	Description
User password	LDAP password of the test user.

Warning:
In case of trouble with certificates, to make a secure LDAP connection (ldaps) work you may need to add a `TLS_REQCERT allow` line to the /etc/openldap/ldap.conf configuration file. It may decrease the security of connection to the LDAP catalog.

Note:
It is recommended to create a separate LDAP account (*Bind DN*) to perform binding and searching over the LDAP server with minimal privileges in the LDAP instead of using real user accounts (used for logging in the Zabbix frontend).
Such an approach provides more security and does not require changing the *Bind password* when the user changes his own password in the LDAP server.
In the table above it's *ldap_search* account name.

Note:
Some user groups can still be authenticated by Zabbix. These groups must have frontend access set to Internal.

HTTP

Apache-based (HTTP) authentication can be used to check user names and passwords. Note that a user must exist in Zabbix as well, however its Zabbix password will not be used.

Attention:
Be careful! Make sure that Apache authentication is configured and works properly before switching it on.

Note:
In case of Apache authentication all users (even with frontend access set to Internal) will be authenticated by Apache, not by Zabbix!

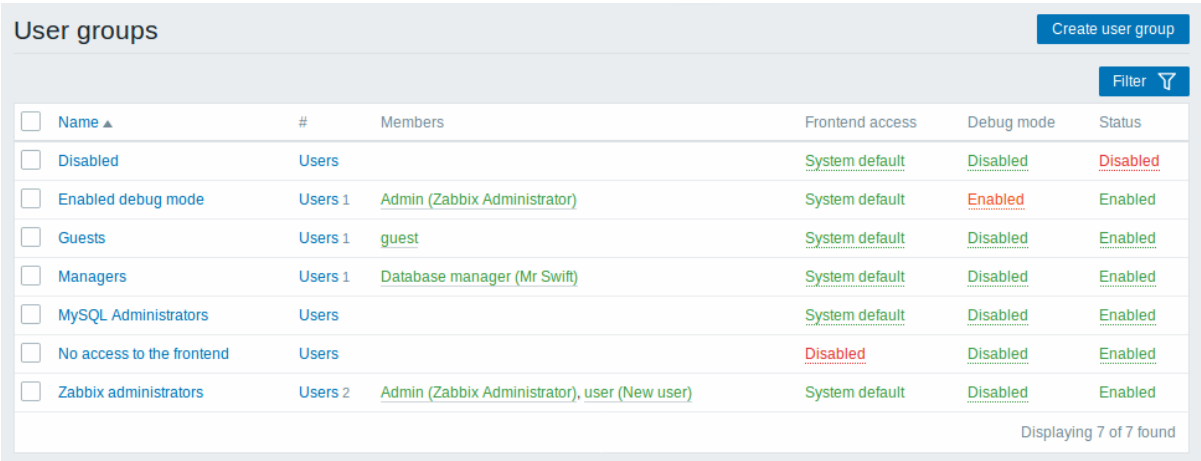
4 用户组

概述

在 Administration → User groups 中，维护系统中的用户组

用户组

显示现有用户组及其详细信息的列表。



显示的数据：

Column	描述
Name	用户组的名称。单击该用户组名来打开用户组的配置列表.
#	该组中用户数量。单击 Users 将会显示过滤出的对应的用户。

Column	描述
Members	用户组中独立用户的别名 (括号内会有名字和姓氏) 单击别名将会打开用户配置列表，来自禁用组的用户会以红色显示。
Frontend access	显示前端访问级别： **System default** - Zabbix, LDAP 或 HTTP 身份认证；取决于选择的身份验证的方法 **Internal** - 方法 **Disabled** - 禁止用户进行前端访问。 单击当前用户级别可以改变用户访问级别。
Debug mode	显示 Debug 模式的状态 Enabled 或 Disabled. 通过点击状态可以改变它。
Status	显示用户组状态。- Enabled 或 Disabled. 通过点击状态可以改变它。

配置新的用户组，点击顶部右上角的 Create user group 。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

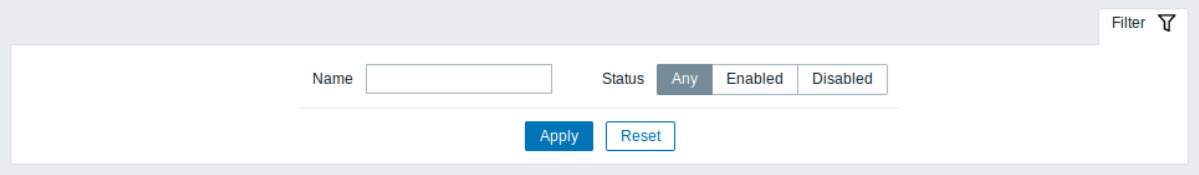
- *Enable* - 将用户组状态改为 Enabled
- *Disable* - 将用户组状态改为 Disabled
- *Enable debug mode* - 激活该用户组的 debug 模式
- *Disable debug mode* - 禁用该用户组的 debug 模式
- *Delete* - 删除用户组

要使用这些选项，请在各个用户组之前标记复选框，然后单击您需要的按键。

过滤器

因为列表中可能包含许多用户组，所以可能需要通过过滤得到您需要的内容。

Filter 过滤器链接位于用户组列表之上。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤用户组。



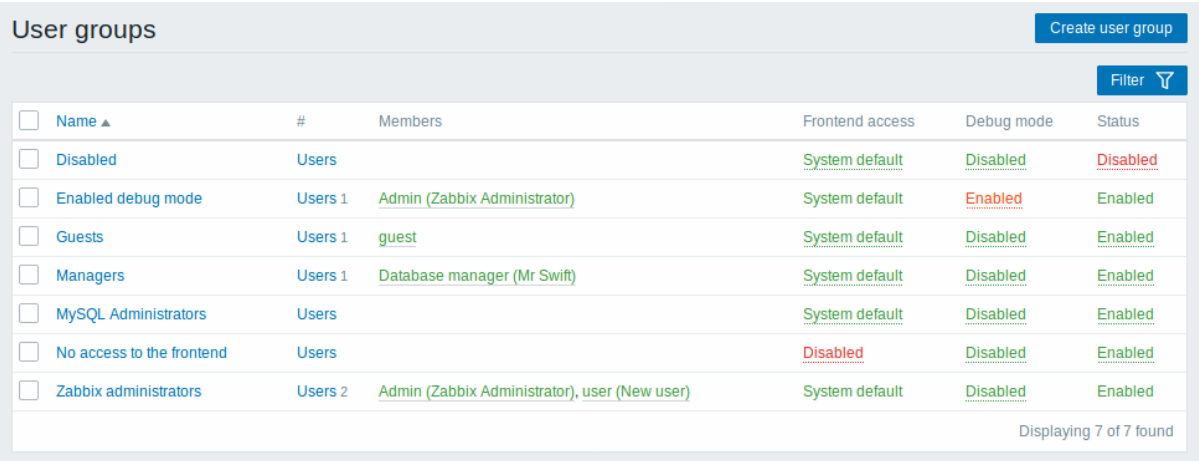
4 User groups

Overview

In the *Administration* → *User groups* section user groups of the system are maintained.

User groups

A listing of existing user groups with their details is displayed.



Displayed data:

Column	Description
Name	Name of the user group. Clicking on the user group name opens the user group configuration form.
#	The number of users in the group. Clicking on *Users* will display the respective users filtered out in the user list.
Members	Aliases of individual users in the user group (with name and surname in parentheses). Clicking on the alias will open the user configuration form. Users from disabled groups are displayed in red.
Frontend access	Frontend access level is displayed: **System default** - Zabbix, LDAP or HTTP authentication; depending on the chosen authentication method **Internal** - the user is authenticated by Zabbix regardless of system settings **Disabled** - frontend access for this user is disabled. By clicking on the current level you can change it.
Debug mode	Debug mode status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Status	User group status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

To configure a new user group, click on the *Create user group* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

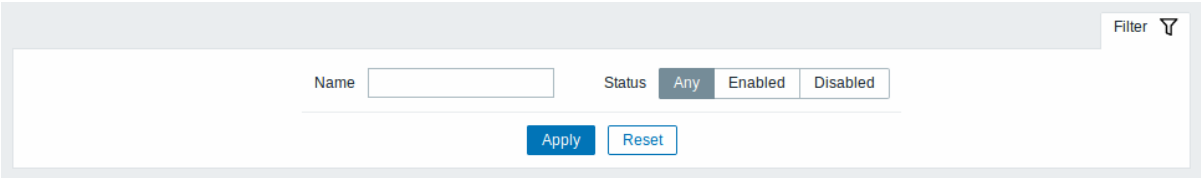
- *Enable* - change the user group status to *Enabled*
- *Disable* - change the user group status to *Disabled*
- *Enable debug mode* - enable debug mode for the user groups
- *Disable debug mode* - disable debug mode for the user groups
- *Delete* - delete the user groups

To use these options, mark the checkboxes before the respective user groups, then click on the required button.

Filter

As the list may contain many user groups, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of user groups. If you click on it, a filter becomes available where you can filter user groups by name and status.



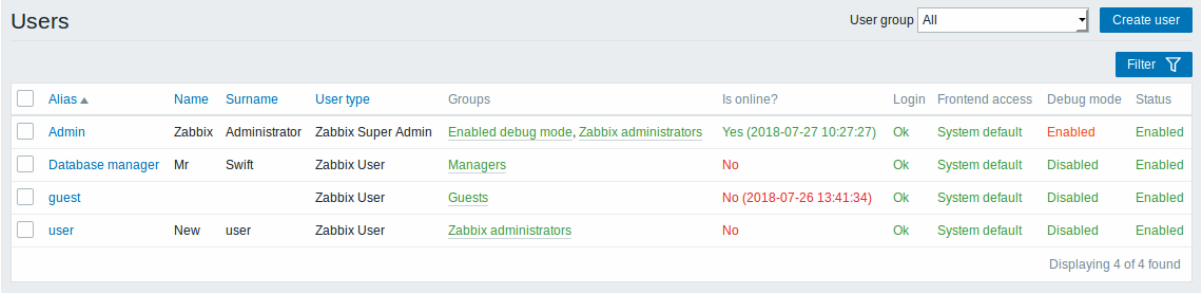
5 用户

概述

在 Administration → Users 中，维护系统中的用户。

用户

显示现有用户及其详细信息的列表。



从 Users 栏中的右侧的下拉列表中，您可以选择是显示所有用户或是只显示属于一个特定组的用户。

显示的数据：

列	述
Alias	用户的别名，用于登录 Zabbix。点击该别名打开用户配置表单.
Name	用户的名字。
Surname	用户的姓氏。
User type	显示用户类型 - Zabbix Super Admin, （Zabbix 超级管理员）Zabbix Admin （Zabbix 管理员）或者 Zabbix User （Zabbix 用户）。
Groups	显示用户类型 - Zabbix Super Admin, （Zabbix 超级管理员）Zabbix Admin （Zabbix 管理员）或者 Zabbix User （Zabbix 用户）。
Is online?	显示用户在线状态- Yes 或 No. 用户最后活动时间显示在括号内。
Login	显示用户的登录状态。- Ok or Blocked. 用户会因连续五次失败登录而被暂时锁定。点击 Blocked 您可以解除该用户的锁定。
Frontend access	显示前端访问级别。- System default, Internal 或 Disabled, 这取决于整个用户组的设置。
Debug mode	显示 Debug 模式 - Enabled 或 Disabled, 这取决于整个用户组的设置。
Status	显示用户状态 - Enabled 或 Disabled, 这取决于整个用户组的设置。

配置新的用户，点击顶部右上角的 Create user

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

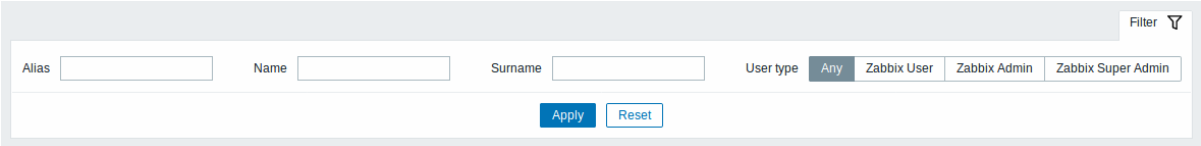
- *Unblock* - 重新启用被阻止用户对系统的访问
- *Delete* - 删除用户

要使用这些选项，请在各个用户之前标记复选框，然后单击您需要的按键

过滤器

因为列表中可能包含许多用户，所以可能需要通过过滤得到您需要的内容。

Filter 过滤器链接位于用户列表之上。如果您点击它，则可以使用过滤器，您可以通过别名，名字，姓氏和用户类型过滤用户。



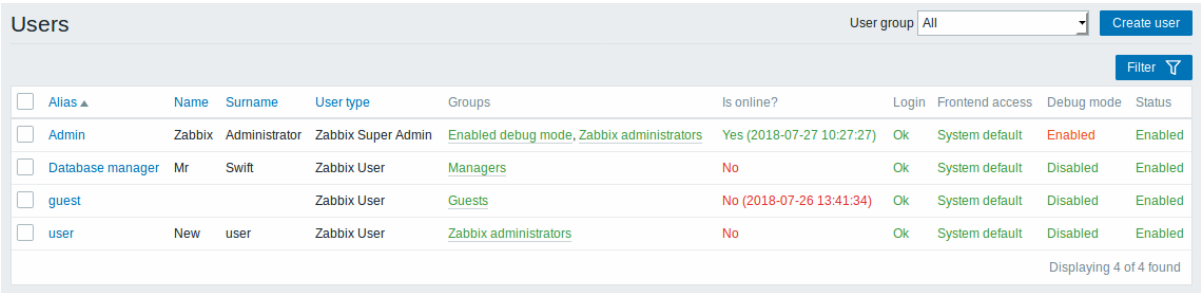
5 Users

Overview

In the *Administration* → *Users* section users of the system are maintained.

Users

A listing of existing users with their details is displayed.



From the dropdown to the right in the *Users* bar you can choose whether to display all users or those belonging to one particular group.

Displayed data:

Column	Description
Alias	Alias of the user, used for logging into Zabbix. Clicking on the alias opens the user configuration form.
Name	First name of the user.
Surname	Second name of the user.
User type	User type is displayed - *Zabbix Super Admin*, *Zabbix Admin* or *Zabbix User*.
Groups	Groups that the user is member of are listed. Clicking on the user group name opens the user group configuration form. Disabled groups are displayed in red.
Is online?	The on-line status of the user is displayed - *Yes* or *No*. The time of last user activity is displayed in parentheses.
Login	The login status of the user is displayed - *Ok* or *Blocked*. A user can become temporarily blocked upon more than five unsuccessful login attempts. By clicking on *Blocked* you can unblock the user.
Frontend access	Frontend access level is displayed - *System default*, *Internal* or *Disabled*, depending on the one set for the whole user group.
Debug mode	Debug mode status is displayed - *Enabled* or *Disabled*, depending on the one set for the whole user group.
Status	User status is displayed - *Enabled* or *Disabled*, depending on the one set for the whole user group.

To configure a new user, click on the *Create user* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

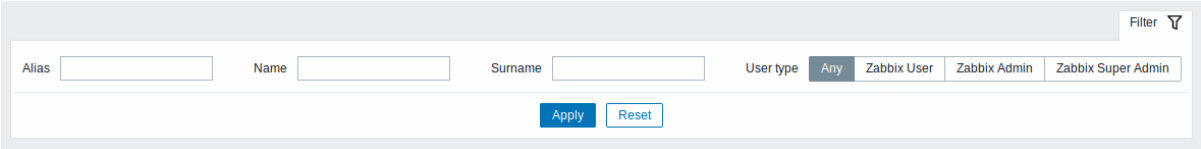
- *Unblock* - re-enable system access to blocked users
- *Delete* - delete the users

To use these options, mark the check-boxes before the respective users, then click on the required button.

Filter

As the list may contain many users, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of users. If you click on it, a filter becomes available where you can filter users by alias, name, surname and user type.



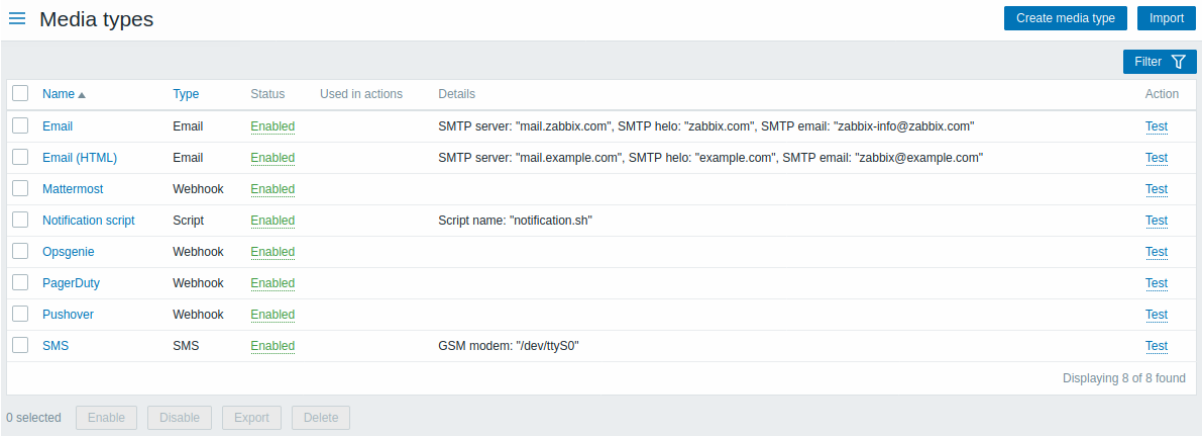
6 媒介类型

概述

在 Administration → Media types 部分，用户可以配置和维护媒介类型信息。

媒介类型信息包含使用媒介作为通知的传送通道的一般说明。具体细节，比如发送通知的个人电子邮件地址与个人用户保持一致。

显示现有媒介类型及其详细信息的列表。



显示的信息：

列	述
Name	媒介类型名称。. 单击名称打开媒介类型配置表单.
Type	显示媒介类型 (电子邮件，短信等)
Status	显示媒介类型状态- Enabled or Disabled. 您可用过单击来改变其状态。
Used in actions	将显示直接使用媒介类型的所有动作（仅在 Send only to 下拉菜单中选择）。单击动作名称打开动作配置表单。
Details	显示媒介类型的详细信息。

配置新的媒介类型，点击顶部右上角的 Create media type

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

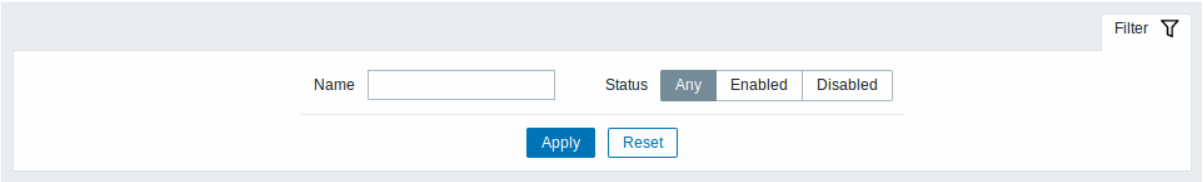
- *Enable* - 将媒介类型状态改为 Enabled
- *Disable* - 将媒介类型状态改为 Disabled
- *Delete* - 删除媒介类型

要使用这些选项，请在各个媒介类型之前标记复选框，然后单击您需要的按键。

过滤器

因为列表中可能包含许多媒介类型，所以可能需要通过过滤得到您需要的内容。

Filter 过滤器链接位于媒介类型列表之上。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤媒介类型。



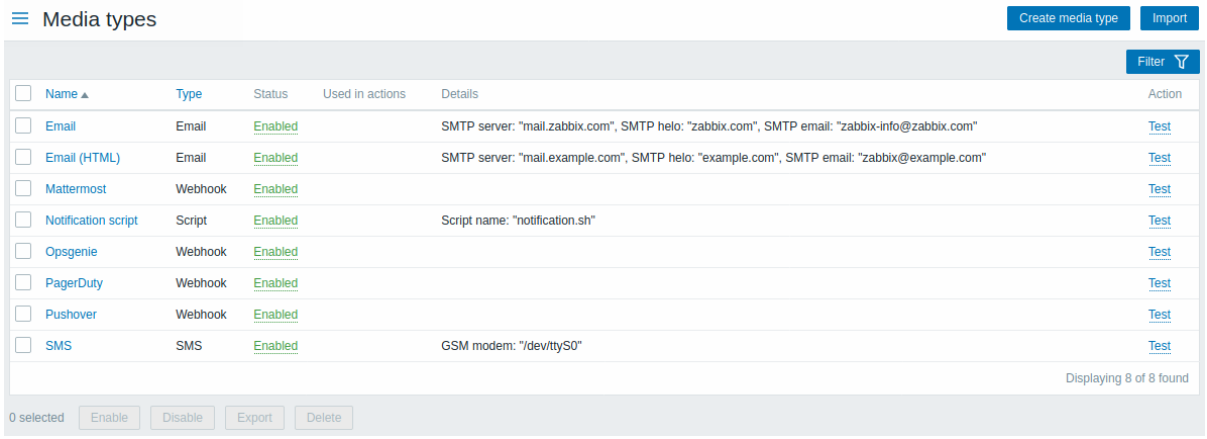
6 Media types

Overview

In the *Administration* → *Media types* section users can configure and maintain media type information.

Media type information contains general instructions for using a medium as delivery channel for notifications. Specific details, such as the individual e-mail addresses to send a notification to are kept with individual users.

A listing of existing media types with their details is displayed.



Displayed data:

Column	Description
Name	Name of the media type. Clicking on the name opens the media type configuration form.
Type	Type of the media (e-mail, SMS, etc) is displayed.
Status	Media type status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Used in actions	All actions where the media type is used directly (selected in the *Send only to* dropdown) are displayed. Clicking on the action name opens the action configuration form.
Details	Detailed information of the media type is displayed.

To configure a new media type, click on the *Create media type* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

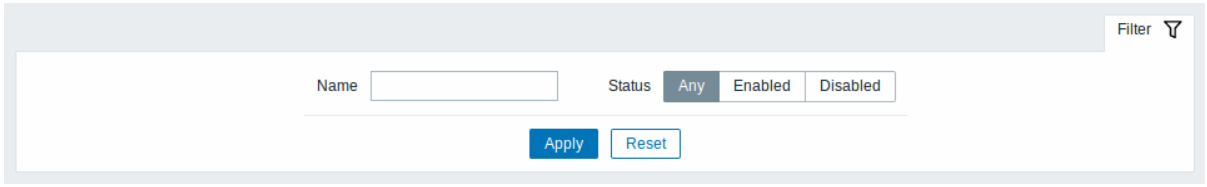
- *Enable* - change the media type status to *Enabled*
- *Disable* - change the media type status to *Disabled*
- *Delete* - delete the media types

To use these options, mark the checkboxes before the respective media types, then click on the required button.

Filter

As the list may contain a number of media types, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of media types. If you click on it, a filter becomes available where you can filter media types by name and status.

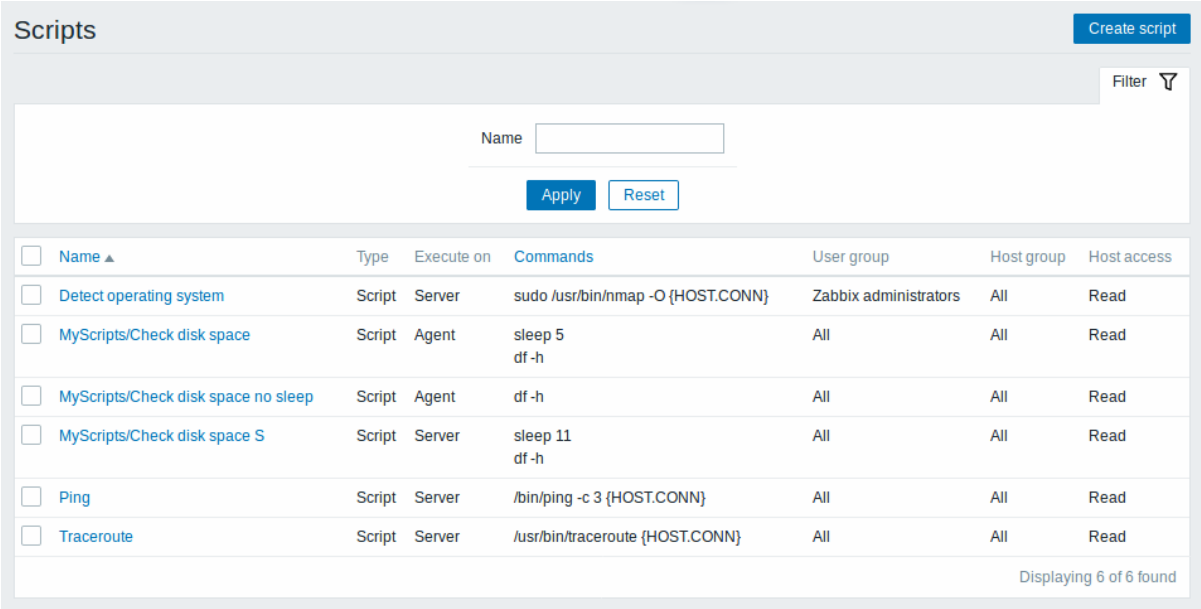


7 脚本

概述

在 Administration → Scripts 中，可以配置和维护用户定义的全局脚本。

这些脚本取决于设置的用户权限，之后可以通过单击主机上各个前端位置 (Dashboard, Problems, Latest data, Status of triggers, Maps) 便可执行，同时也可以用作行动操作来运行。脚本在 Zabbix sever 或 agent 上执行。显示现有脚本及其详细信息的列表



显示的数据：

列	述
Name	脚本名。点击该脚本名打开脚本配置表格..
Type	显示脚本类型- Script 或者 IPMI 命令
Execute on	显示脚本执行在 Zabbix sever 或者 agent 上。
Commands	显示在脚本中执行的所有命令。
User group	显示该脚本可用的用户组 (或者 All 针对所有用户组)。
Host group	将显示该脚本可用的主机组 (或者 All 针对所有主机组)。
Host access	显示主机组的权限级别 Read 或者 Write. 只有具备所需权限级别的用户才能访问执行脚本。

配置新的脚本，请单击顶部右上角的 Create script 按键。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

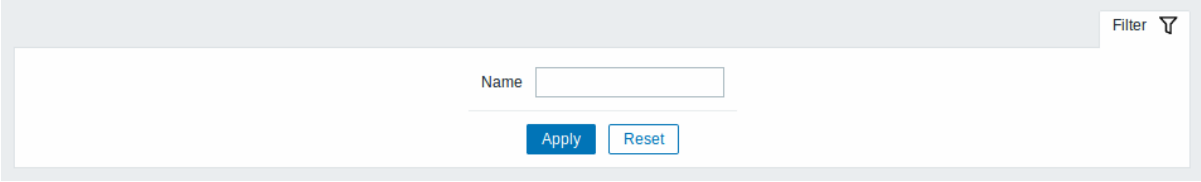
- *Delete* -删除脚本

要使用这个选项，请在各个脚本之前标记复选框，然后单击 Delete.

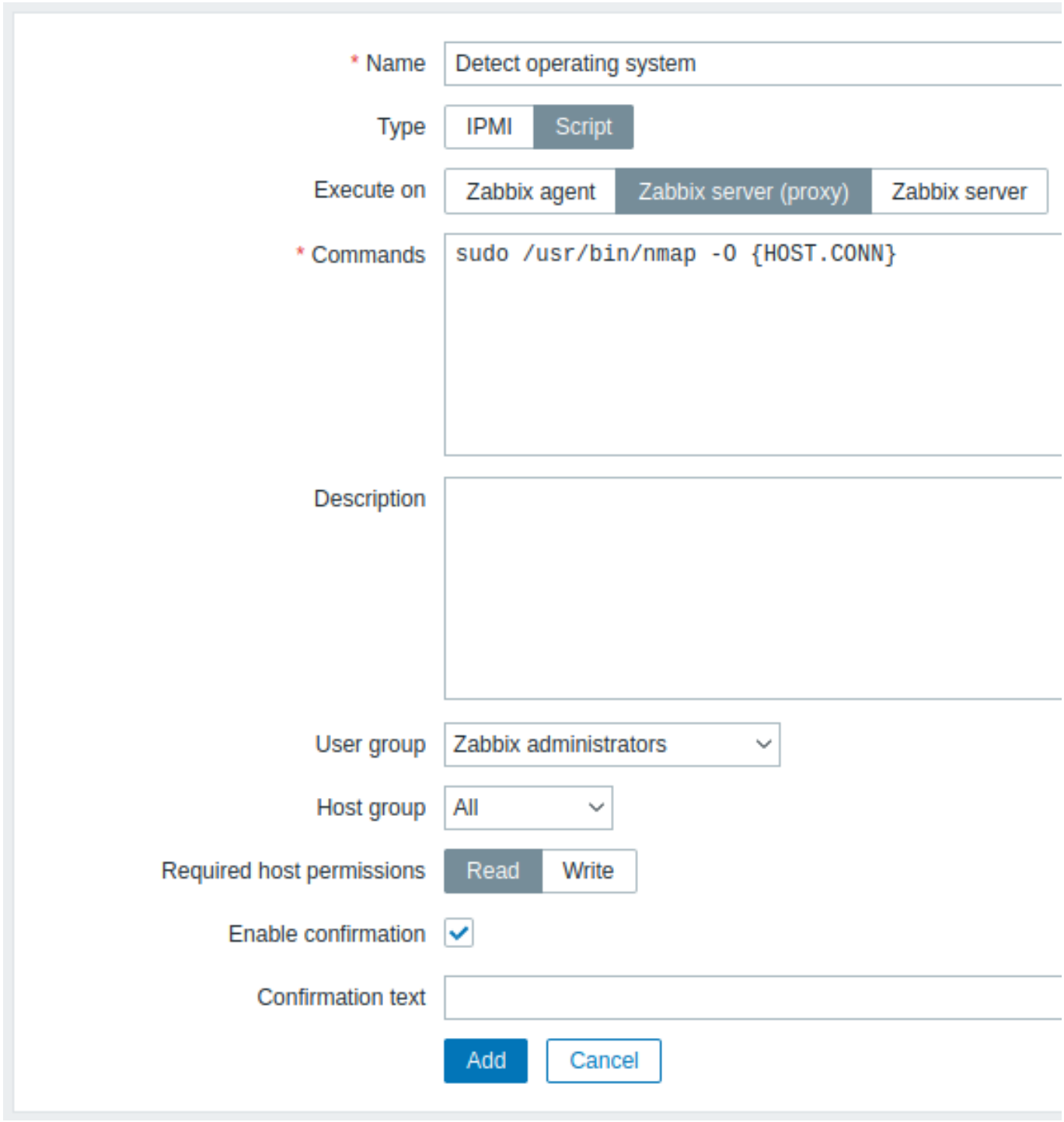
过滤器

因为列表中可能包含许多脚本，所以可能需要通过过滤得到您需要的内容。

Filter 过滤器链接位于脚本列表之上。如果您点击它，则可以使用过滤器，您可以通过名称过滤媒脚本。



配置全局脚本


* Name
Detect operating system
Type
IPMI
Script
Execute on
Zabbix agent
Zabbix server (proxy)
Zabbix server
* Commands
sudo /usr/bin/nmap -O {HOST.CONN}
Description
User group
Zabbix administrators
Host group
All
Required host permissions
Read
Write
Enable confirmation
Confirmation text
Add
Cancel

脚本属性：

参数	描
Name	脚本的唯一名称。\ 从 Zabbix 2.2 起，名称可以以所需的路径为前缀，例如 Default/，将脚本放入相应的目录。通过监控部分中的菜单访问脚本时，将根据给定的目录进行组

参数	描
Type	点击相应的按键，来选择脚本类型 - IPMI command或者 Script.

参数	描
Execute on	单击对应的按键来在 Zabbix server 或 agent 上执行脚本。从 Zabbix 2.0 版本起，(在 Zabbix agent配置文件中的 EnableRemoteCommands 参数中启用远程命令)，可以使用 Zabbix agent 执行脚本的选项。

参数描	
Commands	输入脚本执行命令的完整路径。命令中支持以下宏：{HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}. 如果宏可能解析为具有空格的值(例如, host name)，不要忘记使用引号。从 Zab-bix 2.2 起，脚本命令开始支持用

参数	描
Description	为脚本添加描述
User group	选择脚本可用的用户组（All是对所有的用户组）。
Host group	选择脚本可用的主机组（All是对所有主机组）。

参数	描
Required host permissions	选择主机组的权限级别 -- Read 或 Write。只有具有所需权限级别的用户才能访问执行脚本。

参数	描
Enable confirmation	在执行脚本之前选中复选框以显示确认消息。对于潜在的危险操作(如重新启动脚本)可能需要很长的操作时间，此功能因此会特别有用。

参数描	
Confirmation text	使用复选框，输入确认弹出窗口的自定义确认文本(例如，Remote system-tem 远程系统将要重启，您确定吗？). 要查看文字的效果，请点击该字段旁边的 Test con-fir-ma-tion 。

从 Zab-

参数描

脚本的执行和结果

由 Zabbix sever 运行的脚本由命令执行 部分中描述的顺序执行，包括退出代码检查。脚本结果将显示在运行脚本后显示在弹窗中。

Note: 脚本的返回值是标准输出以及标准错误。

请参见下面的脚本和结果窗口示例:

```
uname
uname --non-existing-flag
/tmp/non_existing_script.sh
```

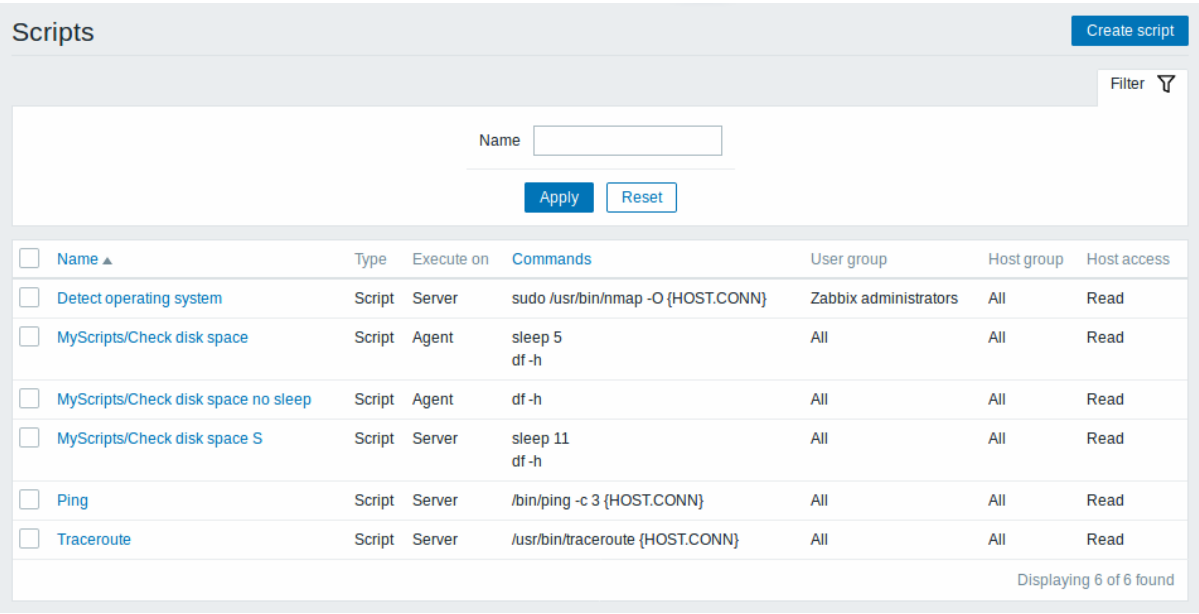
7 Scripts

Overview

In the *Administration* → *Scripts* section user-defined global scripts can be configured and maintained.

These scripts, depending on the set user permissions, then become available for execution by clicking on the host in various frontend locations (*Dashboard*, *Problems*, *Latest data*, *Maps*) and can also be run as an action operation. The scripts are executed on the Zabbix server or agent.

A listing of existing scripts with their details is displayed.



Displayed data:

Column	Description
Name	Name of the script. Clicking on the script name opens the script configuration form.
Type	Script type is displayed - *Script* or *IPMI* command.
Execute on	It is displayed whether the script will be executed on Zabbix server or agent.
Commands	All commands to be executed within the script are displayed.
User group	The user group that the script is available to is displayed (or *All* for all user groups).
Host group	The host group that the script is available for is displayed (or *All* for all host groups).
Host access	The permission level for the host group is displayed - *Read* or *Write*. Only users with the required permission level will have access to executing the script.

To configure a new script, click on the *Create script* button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

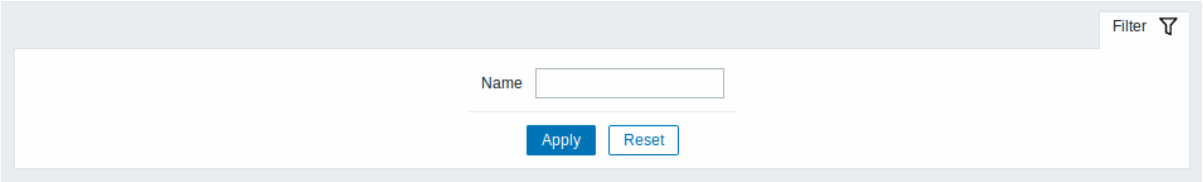
- *Delete* - delete the scripts

To use this option, mark the checkboxes before the respective scripts and click on *Delete*.

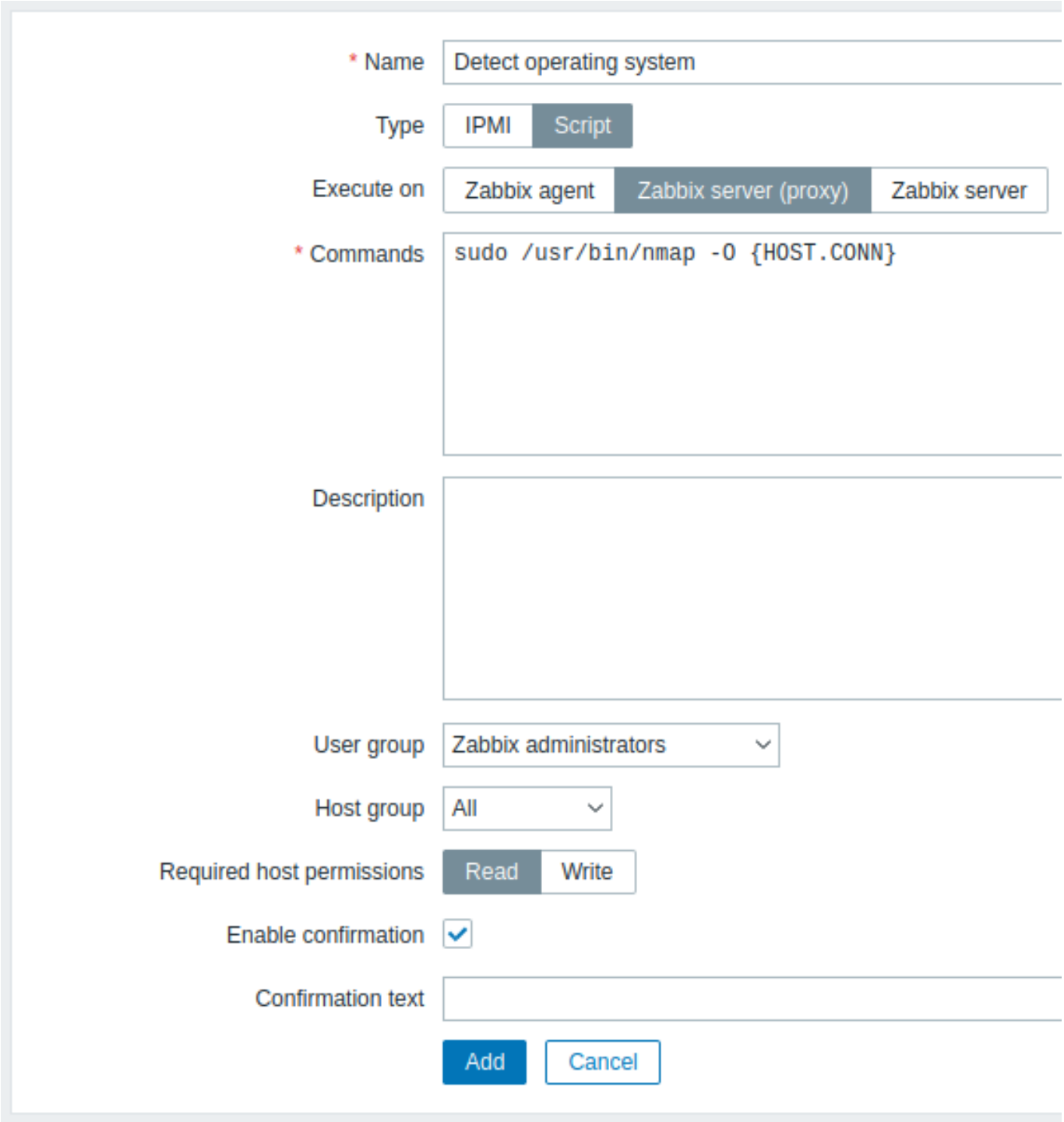
Filter

As the list may contain a number of scripts, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of scripts. If you click on it, a filter becomes available where you can filter scripts by name.



Configuring a global script



Script attributes:

Parameter	Description
Name	Unique name of the script. Since Zabbix 2.2 the name can be prefixed with the desired path, for example, `Default/`, putting the script into the respective directory. When accessing scripts through the menu in monitoring sections, they will be organized according to the given directories. A script cannot have the same name as an existing directory (and vice versa). A script name must be unique within its directory. Unescaped script names are validated for uniqueness, i.e. "Ping" and "\Ping" cannot be added in the same folder. A single backslash escapes any symbol directly after it. For example, characters '/' and '\' can be escaped by backslash, i.e. \/ or \\.
Type	Click the respective button to select script type - IPMI command or Script.

Parameter	Description
Execute on	Click the respective button to execute the script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only The option to execute scripts on Zabbix agent is available since Zabbix 2.0 version (providing remote commands are enabled in the EnableRemoteCommands parameter in Zabbix agent configuration file).
Commands	Enter full path to the commands to be executed within the script. The following macros are supported in the commands: {HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}. If a macro may resolve to a value with spaces (for example, host name), don't forget to quote as needed. Since Zabbix 2.2, user macros are supported in script commands.
Description	Enter a description for the script.
User group	Select the user group that the script will be available to (or *All* for all user groups).
Host group	Select the host group that the script will be available for (or *All* for all host groups).
Required host permissions	Select the permission level for the host group - *Read* or *Write*. Only users with the required permission level will have access to executing the script.
Enable confirmation	Mark the checkbox to display a confirmation message before executing the script. This feature might be especially useful with potentially dangerous operations (like a reboot script) or ones that might take a long time.
Confirmation text	Enter a custom confirmation text for the confirmation popup enabled with the checkbox above (for example, *Remote system will be rebooted. Are you sure?*). To see how the text will look like, click on *Test confirmation* next to the field. Since Zabbix 2.2, the confirmation text will expand host name macros - {HOST.HOST}, {HOST.NAME}, host connection macros - {HOST.IP}, {HOST.DNS}, {HOST.CONN} and user macros. *Note:* The macros will not be expanded when testing the confirmation message.

Script execution and result

Scripts run by Zabbix server are executed by the order described in Command execution section including exit code checking. The script result will be displayed in a pop-up window that will appear after the script is run.

Note: The return value of the script is standard output together with standard error.

See example of a script and the result window below:

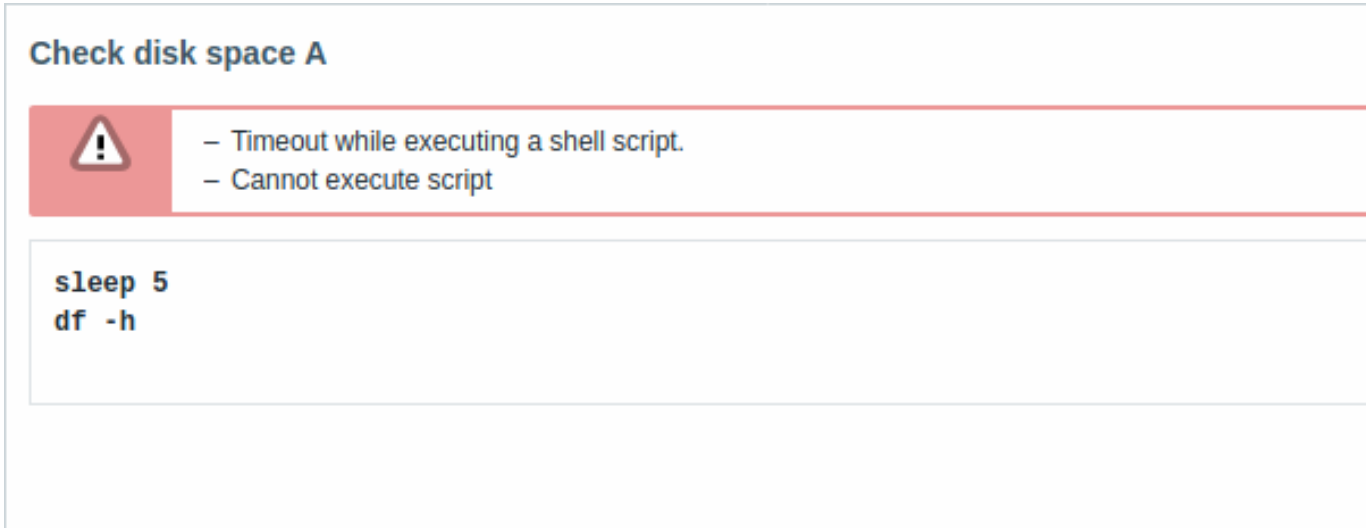
```
uname
uname --non-existing-flag
/tmp/non_existing_script.sh
```

Script timeout

Zabbix agent

You may encounter a situation when timeout occurs while executing a script.

See example of a script running on Zabbix agent and the result window below:



Error message in this case is the following:

```
Timeout while executing a shell script.
```

In order to avoid such a situation, it is advised to optimize the script itself (instead of adjusting Timeout parameter to a corresponding value (in our case, > '5') by modifying the Zabbix agent configuration and Zabbix server configuration).

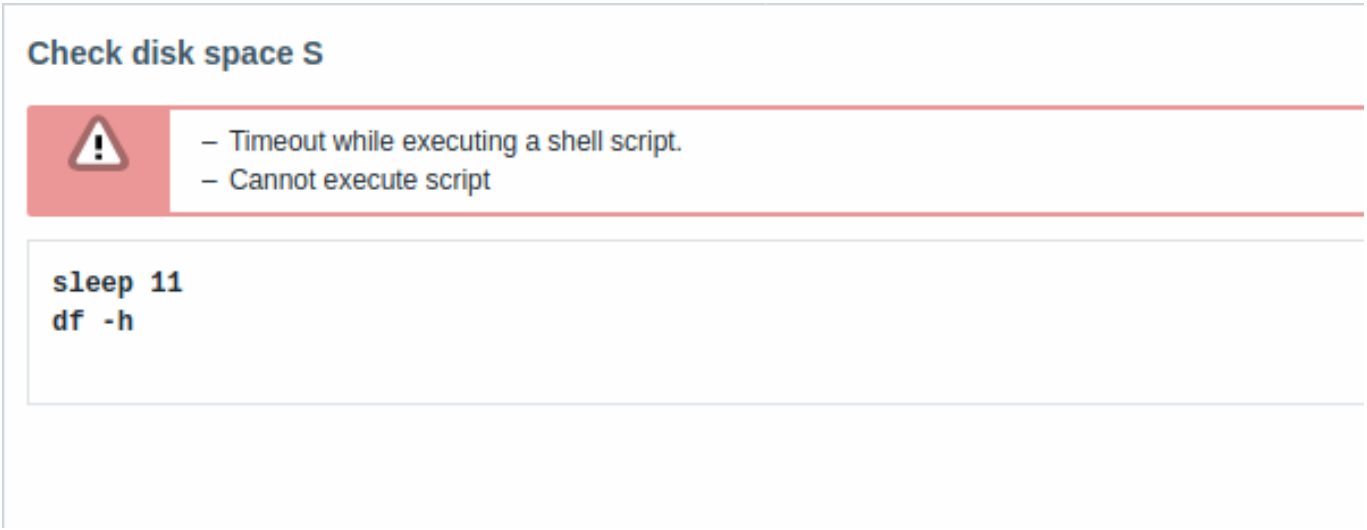
In case still the Timeout parameter is changed in Zabbix agent configuration following error message appears:

```
Get value from agent failed: ZBX_TCP_READ() timed out.
```

It means that modification was made in Zabbix agent configuration and it is required to modify Timeout setting also in Zabbix server configuration.

Zabbix server/proxy

See example of a script running on Zabbix server and the result window below:



It is also advised to optimize the script itself (instead of adjusting TrapperTimeout parameter to a corresponding value (in our case, > '11') by modifying the Zabbix server configuration).

8 队列

概述

在 Administration → Queue 中，显示等待升级的监控项。

理想情况下，当您打开此部分时，应该都是 "绿色" 的，表示队列中没有任何监控项。如果所有监控项都没有延迟更新，则没有等待。但是，由于服务器性能匮乏，连接问题或 proxy 问题，有些监控项可能会延迟，并且在该区域中显示信息。有关详细信息，请参阅Queue 队列 。

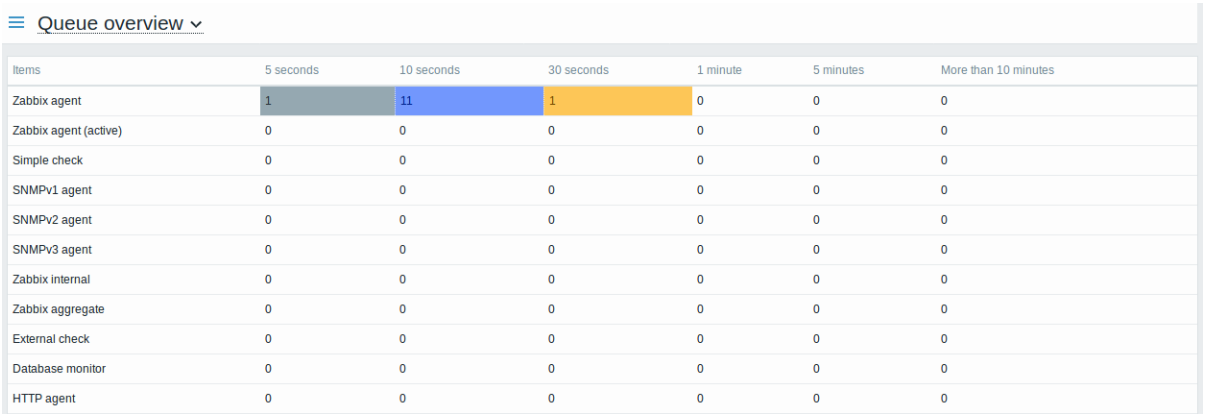
Note:
队列仅在 Zabbix 服务器运行时可用。

从右上角的下拉菜单中您可以选择：

- 监控项类型队列概述
- Proxy 队列概述
- 延时监控项列表

监控项类型概述

在此屏幕中，如果异常与一个或多个监控项类型相关，则可以轻松找到它。

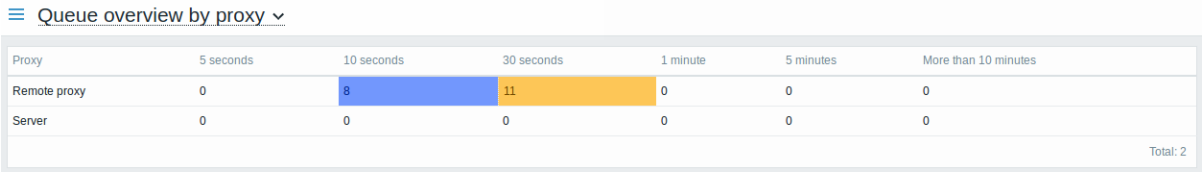


Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

每行包含一个监控项类型。每列显示等待监控项的数量 - 等待 5-10 秒/ 10-30 秒/ 30-60 秒/ 1-5 分钟/ 5-10 分钟或超过 10 分钟。

Proxy 概述

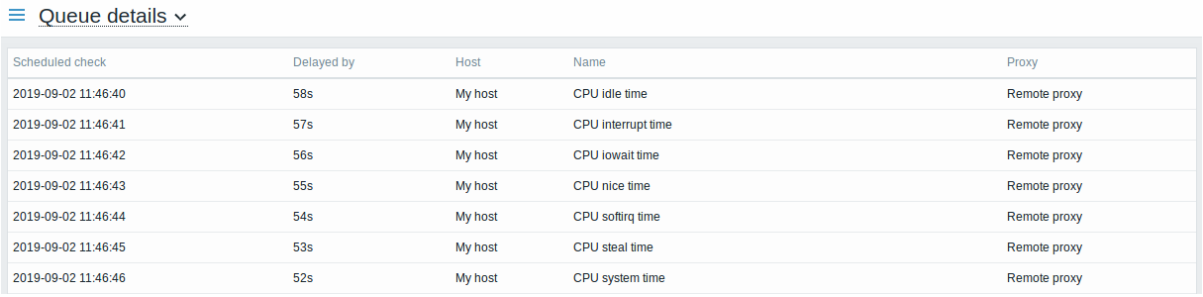
在此屏幕中，如果异常与 agent 或 sever 之一相关，则可以轻松找到它。



每行包含一个 proxy，sever 在列表的最后一个。每列显示等待监控项的数量 - 等待 5-10 秒/ 10-30 秒/ 30-60 秒/ 1-5 分钟/ 5-10 分钟或超过 10 分钟。

等待监控项列表

在下屏中，每个等待监控项被列了出来。



在主机列中，由 proxy 监视的主机以 proxy 名称为前缀（从 Zabbix 2.4.0 起）。

显示的数据：

列	述
Next check	显示检查到期的时间。
Delayed by	显示延迟的长度。
Host	显示监控项的主机。
Name	显示等待监控项的名称。

可能的错误消息

您可能会遇到没有数据显示的情况，并显示以下错误消息：

在这种情况下的错误消息如下：

```
Cannot display item queue. Permission denied
```

当 zabbix.conf.php 中的 PHP 配置参数 $ZBX_SERVER_PORT 或 $ZBX_SERVER 指向使用不同数据库的现有 Zabbix 服务器时，会发生这种情况。

8 Queue

Overview

In the *Administration* → *Queue* section items that are waiting to be updated are displayed.

Ideally, when you open this section it should all be ”green” meaning no items in the queue. If all items are updated without delay, there are none waiting. However, due to lacking server performance, connection problems or problems with agents, some items may get delayed and the information is displayed in this section. For more details, see the Queue section.

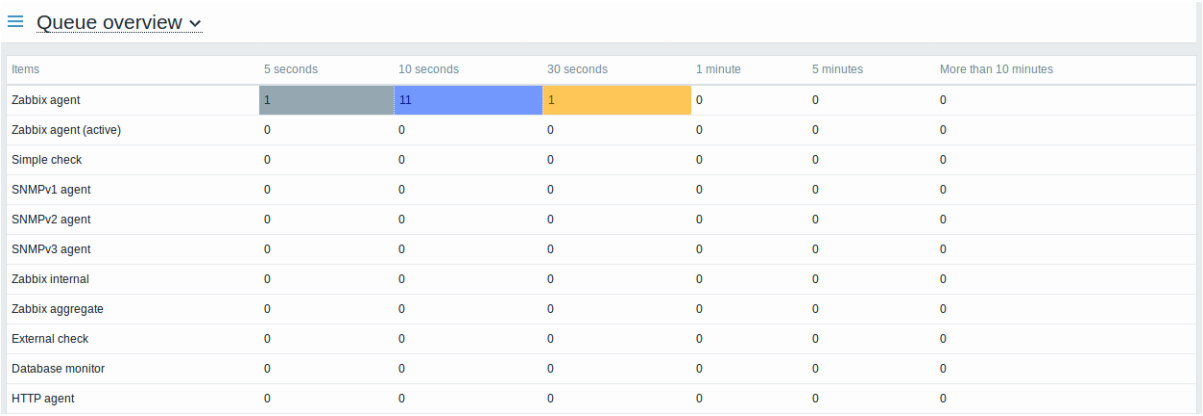
Note:
Queue is available only if Zabbix server is running.

From the dropdown in the upper right corner you can select:

- queue overview by item type
- queue overview by proxy
- list of delayed items

Overview by item type

In this screen it is easy to locate if the problem is related to one or several item types.

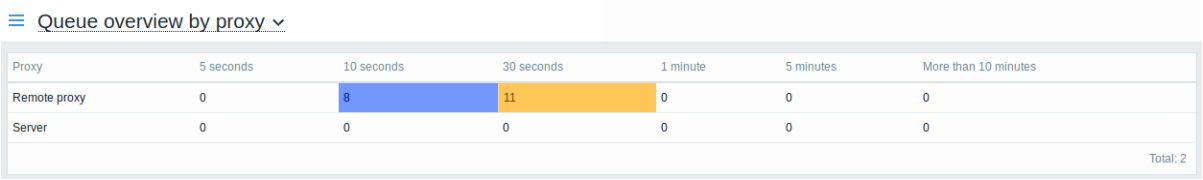

Queue overview

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

Each line contains an item type. Each column shows the number of waiting items - waiting for 5-10 seconds/10-30 seconds/30-60 seconds/1-5 minutes/5-10 minutes or over 10 minutes respectively.

Overview by proxy

In this screen it is easy to locate if the problem is related to one of the proxies or the server.


Queue overview by proxy

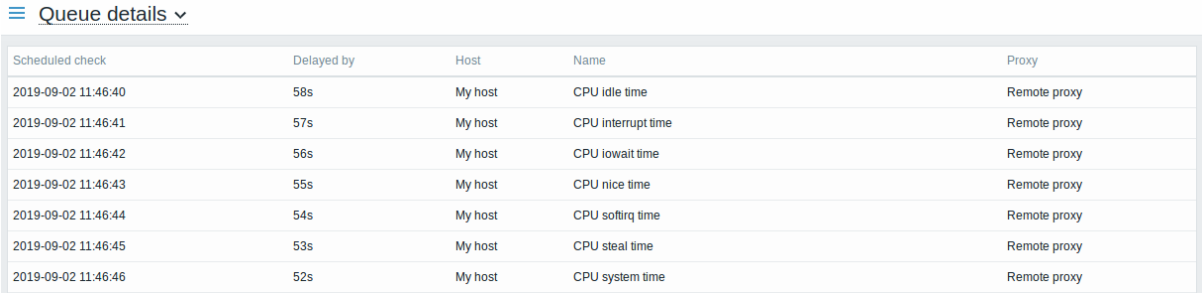
Proxy	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Remote proxy	0	8	11	0	0	0
Server	0	0	0	0	0	0

Total: 2

Each line contains a proxy, with the server last in the list. Each column shows the number of waiting items - waiting for 5-10 seconds/10-30 seconds/30-60 seconds/1-5 minutes/5-10 minutes or over 10 minutes respectively.

List of waiting items

In this screen, each waiting item is listed.


Queue details

Scheduled check	Delayed by	Host	Name	Proxy
2019-09-02 11:46:40	58s	My host	CPU idle time	Remote proxy
2019-09-02 11:46:41	57s	My host	CPU interrupt time	Remote proxy
2019-09-02 11:46:42	56s	My host	CPU iowait time	Remote proxy
2019-09-02 11:46:43	55s	My host	CPU nice time	Remote proxy
2019-09-02 11:46:44	54s	My host	CPU softirq time	Remote proxy
2019-09-02 11:46:45	53s	My host	CPU steal time	Remote proxy
2019-09-02 11:46:46	52s	My host	CPU system time	Remote proxy

In the host column, hosts monitored by proxy are prefixed with the proxy name (since Zabbix 2.4.0).

Displayed data:

Column	Description
Next check	The time when the check was due is displayed.
Delayed by	The length of the delay is displayed.
Host	Host of the item is displayed.
Name	Name of the waiting item is displayed.

Possible error messages

You may encounter a situation when no data is displayed and the following error message appears:

Details

Cannot display item queue.

Permission denied.

Error message in this case is the following:

```
Cannot display item queue. Permission denied
```

This happens when PHP configuration parameters $ZBX_SERVER_PORT or $ZBX_SERVER in zabbix.conf.php point to existing Zabbix server which uses different database.

2 User profile 用户资料

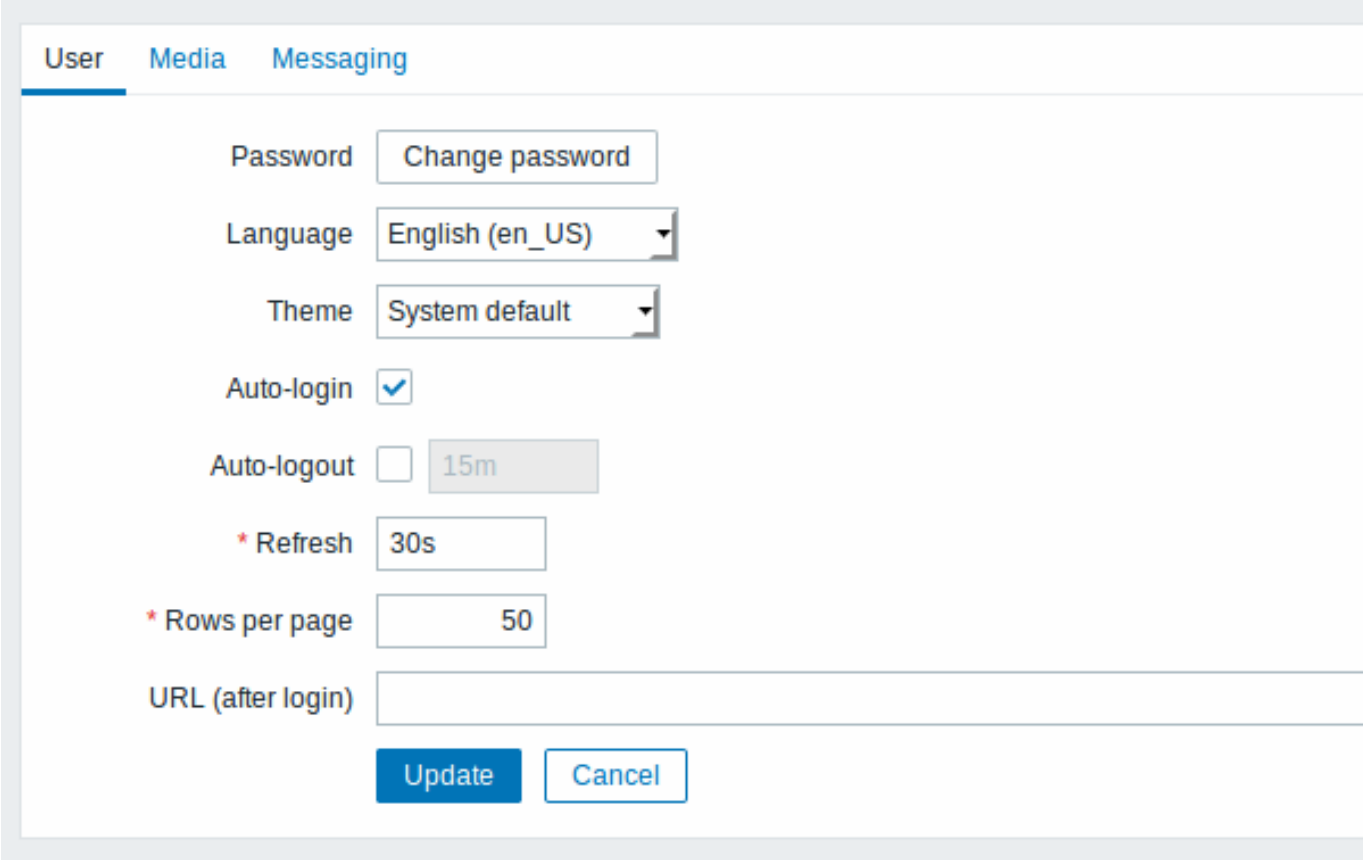
Overview 概述

In the user profile you can customize some Zabbix frontend features, such as the interface language, color theme, number of rows displayed in the lists etc. The changes made here will apply for the user only. 在用户资料中，你可以自定义一些 Zabbix 的前端特性，比如：界面语言，主题颜色，列表中显示的行数等等。此改变只针对当前用户。

To access the user profile configuration form, click on the user profile link in the upper right corner of Zabbix window. 点击 zabbix 窗口右上角的 来访问用户信息.

Configuration 配置

The **User** tab allows you to set various user preferences. User 选项卡允许您设置关于用户相关配置.

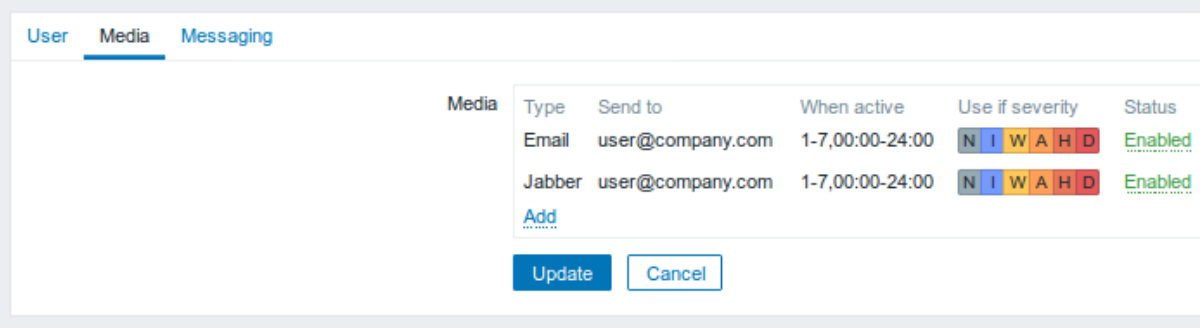


Parameter 参数 D	scription 描述
Password	Click on the link to display two fields for entering a new password. 点击链接显示两个字段，来输入新的密码.
Language	Select the interface language of your choice. 选择您想要的界面语言。 The php gettext extension is required for the translations to work.PHP 的 gettext 扩展是翻译正常运作所必需的。

Parameter 参数 D	scription 描述
Theme	Select a color theme specifically for your profile 为您的资料选择一种特殊的颜色主题: **System default** - use default system settings 系统默认 **Blue** - standard blue theme 标准蓝色主题 **Dark** - alternative dark theme 暗黑主题 **High-contrast light** - light theme with high contrast 高对比的浅色主题 **High-contrast dark** - dark theme with high contrast 高对比的暗黑主题
Auto-login	Mark this checkbox to make Zabbix remember you and log you in automatically for 30 days. Browser cookies are used for this. 选择复选框来标记自动登录，无需再次输入用户名和密码。
Auto-logout	With this checkbox marked you will be logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day). 勾选了这个复选框后，您将在设定的秒数后自动注销 (最少 90 秒). Time suffixes are supported, e.g. 90s, 5m, 2h, 1d. 支持的。如 90s, 5m, 2h, 1d. Note that this option will not work: 请注意，以下选项不起作用： * If the ”Show warning if Zabbix server is down” global configuration option is enabled and Zabbix frontend is kept open; 当 Zabbix 服务器宕机时显示告警，全局配置选项启用且 Zabbix 前端持续打开时； * When Monitoring menu pages perform background information refreshes; 当监控菜单页面一直在后台进行信息刷新时，会无法正常工作； * If logging in with the *Remember me for 30 days* option checked. 当选中”30 天记住我” 选项，请登录.
Refresh	You can set how often the information in the pages will be refreshed on the Monitoring menu, except for Dashboard, which uses its own refresh parameters for every widget. 您可以设置监控目录下信息刷新的频率。只有 Dashboard 例外, 它使用为自己的每个部件使用自有的刷新参数。 Time suffixes are supported, e.g. 30s, 5m, 2h, 1d. 支持的。如 30s, 5m, 2h, 1d.
Rows per page	You can set how many rows will be displayed per page in the lists. Fewer rows (and fewer records to display) mean faster loading times. 可设置每页显示的行数。行数越少（显示的记录越少）加载速度越快。
URL (after login)	You can set a specific URL to be displayed after the login. Instead of the default *Monitoring → Dashboard* it can be, for example, the URL of *Monitoring → Triggers*. 您可设置在登录后显示的自定义 URL 不同于默认的 Monitoring → Dashboard 例如，它甚至可以成 Monitoring 的 URL → Triggers.

Note:
If some language is not available for selection in the user profile it means that a locale for it is not installed on the web server. See the link at the bottom of this page to find out how to install them. 如果某些语言在用户资料中无法选择，则意味着它的区域设置未安装在 Web 服务器上。请参阅链接link ，了解如何安装。

The **Media** tab allows you to specify the media details for the user, such as the types, the addresses to use and when to use them to deliver notifications. 媒介 Media 选项卡允许您指定给用户以media 细节，例如类型、地址的使用以及何时使用它们来发送通知。



Note:
Only admin level users (Admin and Super Admin) can change their own media details. 只有管理员级别admin level 用户（管理员和超级管理员）可以更改他们自己的 media 细节。

The **Messaging** tab allows you to set global notifications. 可通过 **Messaging** 选项卡, 设置全局通知global notifications.

See also 参考

1. How to install additional locales to be able to select unavailable languages in the user profile

1 Global notifications 全局通知

Overview 概述

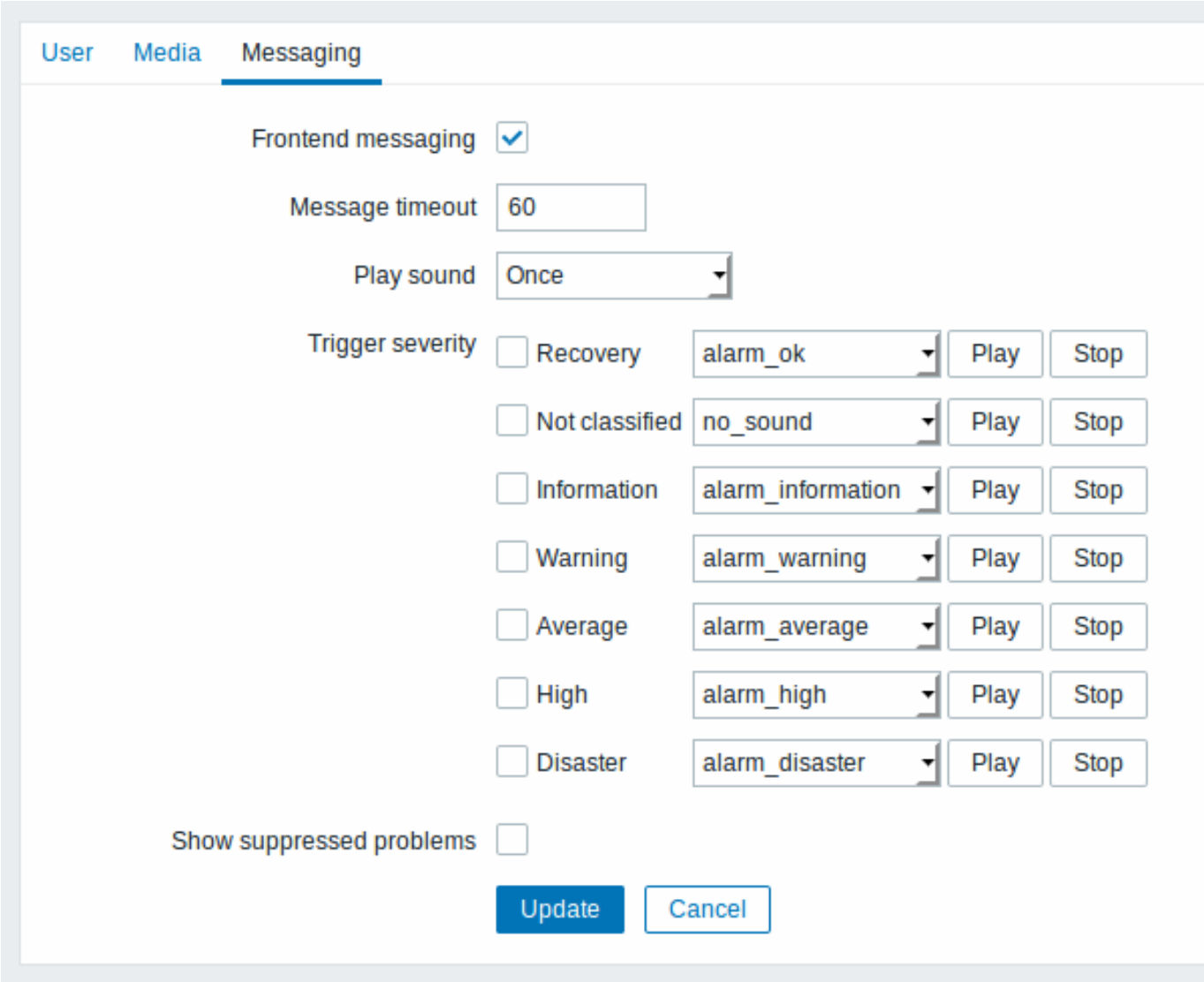
Global notifications are a way of displaying issues that are currently happening right on the screen you're at in Zabbix frontend. 全局通知是一种在 Zabbix 前端屏幕上显示当前正在发生的问题的方法。

Without global notifications, working in some other location than *Host groups* or *Dashboard* pages would not show any information regarding issues that are currently happening. Global notifications will display this information regardless of where you are. 没有全局通知，触发器状态 or Dashboard 页面，将不会显示任何有关当前正在发生的问题的信息。不论您在哪里，全局通知会显示这些信息。

Global notifications involve both showing a message and playing a sound. 全局通知涉及到信息的显示和playing a sound.

Configuration 配置

Global notifications can be enabled per user in the *Messaging* tab of profile configuration. 可以在profile configuration的// Messaging //选项卡中为每个用户启用全局通知。

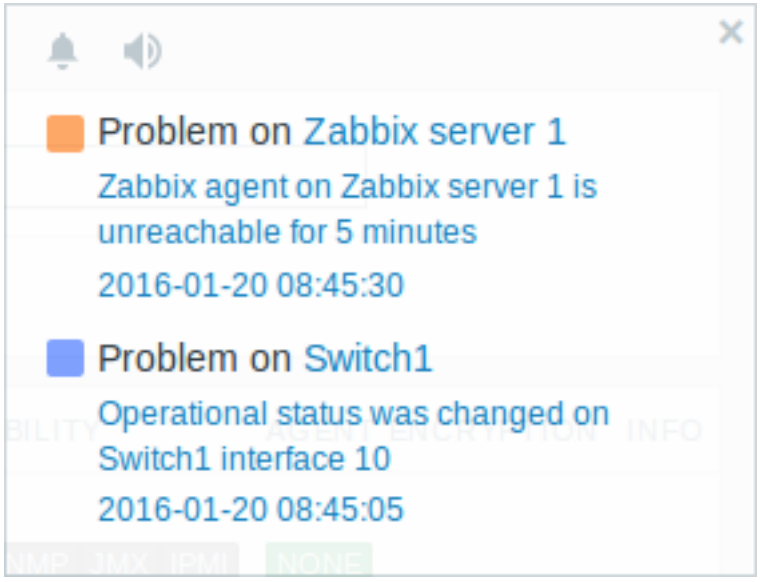


Parameter 参数 D	scription 描述
Frontend messaging	Mark the checkbox to enable global notifications. 选中该复选框以启用全局通知。
Message timeout	You can set for how long the message will be displayed. By default, messages will stay on screen for 60 seconds. 您可以设置消息显示的时间。默认情况下，消息将在屏幕上显示 60 秒。 Time suffixes are supported, e.g. 30s, 5m, 2h, 1d. 支持的，如 30s, 5m, 2h, 1d.
Play sound	You can set how long the sound will be played. 您可以设置声音的播放长度。 **Once** - sound is played once and fully. 声音完整播放一次。 **10 seconds** - sound is repeated for 10 seconds. 声音重复播放 10 秒。 **Message timeout** - sound is repeated while the message is visible. 当消息显示时，声音一直播放。

Parameter 参数 D	scription 描述
Trigger severity	You can set the trigger severities that global notifications and sounds will be activated for. You can also select the sounds appropriate for various severities. 您可以设置全局通知和声音的触发器的严苛度，同时，您还可以针对不同的严苛度选择合适的声音。 If no severity is marked then no messages will be displayed at all. Also, recovery messages will only be displayed for those severities that are marked. So if you mark *Recovery* and *Disaster*, global notifications will be displayed for the problems and the recoveries of disaster severity triggers. 如果没有标记严苛度，那么就不会显示任何消息。而且，只有标记的严苛性才会显示恢复信息。因此，如果标记 Recovery 和 Disaster, 全局通知将会显示问题，以及灾难严苛度触发器的恢复。

Global messages displayed 全局信息显示

As the messages arrive, they are displayed in a floating section on the right hand side. This section can be repositioned freely by dragging the section header. 当消息到达时，它们显示在右侧的浮动部分中。通过拖动节标题可以自由地重新定位此部分。



For this section, several controls are available 在这个区域内，一些控件是可用的：:

- **Snooze** button silences currently active alarm sound 键将会静音当前的警报音;
- **Mute/Unmute** button switches between playing and not playing the alarm sounds 键在播放与不播放警报音之间切换.

2 Sound in browsers 浏览器中的声音

Overview 概述

For the sounds to be played in Zabbix frontend, *Frontend messaging* must be enabled in the user profile *Messaging* tab, with all trigger severities checked, and sounds should also be enabled in the global notification pop-up window. 为了在 Zabbix 前端播放声音，Frontend messaging 必须在用户档案里的 Messaging 选项卡里被启用，并检查所有触发器的严重程度，同时声音也必须在全局通知的弹窗里被启用。

The sounds of Zabbix frontend have been successfully tested in the following web browser versions and no additional configuration was required: Zabbix 前端的声音已经在以下 Web 浏览器版本中成功测试，且不需要其的配置:

- Firefox 3.5.16 on Linux Linux 上的 Firefox 3.5.16

- Opera 11.01 on Linux Linux 上的 Opera 11.01
- Google Chrome 9.0 on Windows Windows 上的 Google Chrome 9.0
- Firefox 3.5.16 on Windows Windows 上的 Firefox 3.5.16
- IE7 browser on Windows Windows 上的 IE7
- Opera v11.01 on Windows Windows 上的 Opera v11.01
- Chrome v9.0 on Windows Windows 山的 Chrome v9.0
- Safari v5.0 on Windows, but this browser requires *Quick Time Player* to be installed Windows 上的 Safari v5.0, 但该浏览器需要安装 Quick Time Player

Additional requirements 附加要求

Firefox v 3.5.16

For playing `wav` files in the Firefox browser you can use one of the following applications: 要在 Firefox 浏览器中播放 wav 文件，您可以使用以下应用：

- Windows Media Player
- Quick Time plug-in. Quick Time 插件

Then, in *Tools* → *Options* → *Applications*, in ”Wave sound (audio/wav)” set Windows Media Player to play these files. 然后，在 Tools（工具）→ Options（选项）→ Applications（应用程序），中，“Wave sound（audio / wav）” 中设置 Windows Media Player 播放这些文件

Safari 5.0

Quick Time Player is required. 需要 Quick Time Player.

Microsoft Internet Explorer

To play sounds in MSIE7 and MSIE8: 在 IE7 和 IE8 中播放声音:

- In *Tools* → *Internet Options* → *Advanced* enable *Play sounds in webpages*
- In *Tools* → *Manage Add-ons...* enable **Windows Media Player**
- In the Windows Media Player, in *Tools→Options→File Types* enable *Windows audio file (wav)*

In the Windows Media Player, in Tools→Options tab, ”File Types” is only available if the user is a member of ”Power Users” or ”Administrators” group, i.e. a regular user does not have access to this tab and does not see it. 在 Windows Media Player 的 “Tools”→“Options” 选项卡中，“File Types” 仅在用户是 “高级用户” 或 “管理员” 组的成员时可用，即普通用户无权访问此选项卡而不是” 看见”。

An additional thing - if IE does not have some *.wav file in the local cache directory (%userprofile%\Local Settings\Temporary Internet Files) the sound will not play the first time. 另外 - 如果 IE 在本地缓存目录（%userprofile%\ Local Settings \ Temporary Internet Files）中没有一些 * .wav 文件，则声音将不会在第一次播放。

Known not to work 已知无法正常工作的

Browsers where the sound did not work 声音不起作用的浏览器:

- Opera 10.11 on Linux. 在 Linux 下的 Opera 10.11.

3 Global search 全局搜索

It is possible to search Zabbix frontend for hosts, host groups and templates.

The search input box is located in the upper right corner. The search can be started by pressing *Enter* or clicking on the

search icon. 在 Zabbix 前端，可以搜索多种实体。实体搜索输入框在右上角。搜索可以通过摁回车键或者点击 搜索图标来开始搜索。

If there is a host that starts with the entered string, a dropdown will appear, listing all such hosts. 如果有一个以输入的字符串开头的主机，将显示一个下拉列表，列出所有这样的主机:

Properties searched 实体搜索

Hosts can be searched by the following properties: 可以搜索这些实体及其属性：

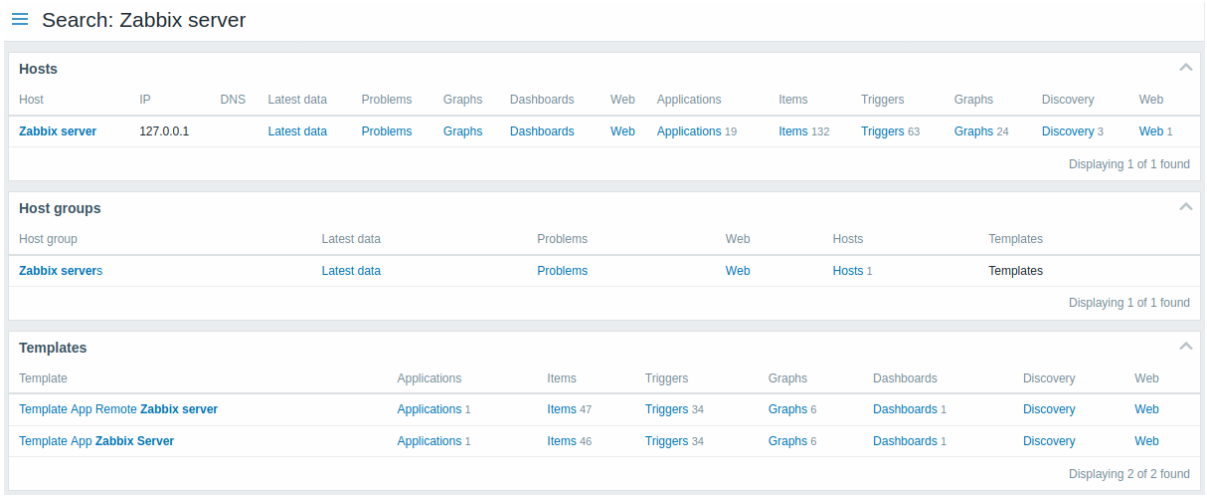
- Host name 主机名
- Visible name 可见名
- IP address IP 地址
- DNS name DNS 名

Host groups can be searched by name. Specifying a parent host group implicitly selects all nested host groups. 指定父主机组间接地选择所有嵌套的主机组

Templates can be searched by name or visible name. If you search by a name that is different from the visible name (of a template/host), in the search results it is displayed below the visible name in parentheses. 可以按名称或可见名搜索模板。如果使用与（模板/主机的）可见名不同的名称进行搜索，则在搜索结果中，它将显示在括号中的可见名称下方。

Search results 搜索结果

Search results consist of three separate blocks for hosts, host groups and templates. 搜索结果包含三个单独的块，用于主机，主机组和模板。



It is possible to collapse/expand each individual block. The entry count is displayed at the bottom of each block, for example, *Displaying 13 of 13 found*. Total entries displayed within one block are limited to 100. 可以折叠/展开每个单独的块。条目计数显示在每个块的底部，例如，显示 *13* 中的 *13* 个找到。一个块内显示的条目总数限制为 100。

Each entry provides links to monitoring and configuration data. See links available. 每个实体都提供指向监视和配置数据的链接。参见links available。

For all configuration data (such as items, triggers, graphs) the amount of entities found is displayed by a number next to the entity name, in grey. **Note** that if there are zero entities, no number is displayed. 对于所有配置数据（例如项目，触发器，图形），找到的实体数量由实体名称旁边的数字显示，灰色。注意如果实体为零，则不显示任何数字。

Enabled hosts are displayed in blue, disabled hosts in red. 已启用的主机以蓝色显示，已禁用的主机以红色显示。

Links available 可用链接

For each entry the following links are available: 对于找到的实体，下列链接均可用：

- Hosts 主机
  - Monitoring 监控
    - Latest data 最新数据
    - Triggers 触发器
    - Problems 异常
    - Graphs 图
    - Host screens 主机聚合图形
    - Web scenarios Web 场景
  - Configuration 配置
    - Host properties 主机属性
    - Applications 应用
    - Items 监控项
    - Triggers 触发器
    - Graphs 图
    - Discovery rules 发现规则
    - Web scenarios Web 场景
- Host groups 主机组
  - Monitoring 监控
    - Latest data 最新数据
    - Triggers 监控项
    - Problems 异常
    - Graphs 图
    - Web scenarios Web 场景
  - Configuration 配置
    - Host group properties 主机组属性
    - Host group members (hosts and templates) 主机组成员（主机和模板）
- Templates 模板
  - Configuration 配置
    - Template properties 模板属性
    - Applications 应用
    - Items 监控项
    - Triggers 触发器
    - Graphs 图
    - Template screens 模板聚合图形
    - Discovery rules 发现规则
    - Web scenarios Web 场景

4 Frontend maintenance mode 前端维护模式

Overview 概述

Zabbix web frontend can be temporarily disabled in order to prohibit access to it. This can be useful for protecting the Zabbix database from any changes initiated by users, thus protecting the integrity of database. Zabbix web 前端可以暂时禁用，以禁止访问它。这对于保护 Zabbix 数据库免受用户发起的任何更改非常有用，从而保护了数据库的完整性。

Zabbix database can be stopped and maintenance tasks can be performed while Zabbix frontend is in maintenance mode. Zabbix 数据库可以被停止，并且维护任务可以在 Zabbix 前端在维护模式中进行。

Users from defined IP addresses will be able to work with the frontend normally during maintenance mode. 来自指定 IP 地址的用户将能够在维护模式期间正常工作。

Configuration 配置

In order to enable maintenance mode, the `maintenance.inc.php` file (located in /conf of the Zabbix HTML document directory on the webserver) must be modified to uncomment the following lines: 为了启用维护模式，必须以取消注释的方法修改 `maintenance.inc.php` 文件（位于 web 服务器上的 Zabbix HTML 文档目录的/ conf 中）：

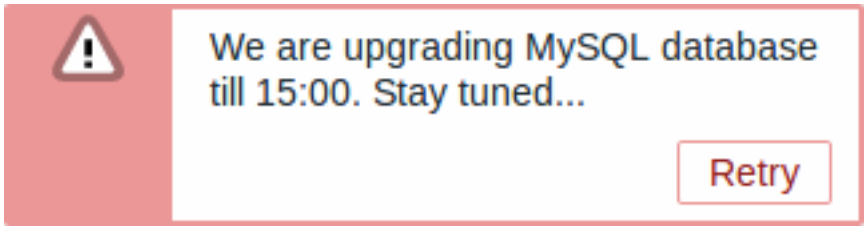
```
// Maintenance mode. 维护模式
define('ZBX_DENY_GUI_ACCESS', 1);

// Array of IP addresses, which are allowed to connect to frontend (optional). 一列包含IP地址的数组，它们可
```

```
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// Message shown on warning screen (optional). 警告屏幕上所显示的信息（可选）
$ZBX_GUI_ACCESS_MESSAGE = 'We are upgrading MySQL database till 15:00. Stay tuned...';
```

Parameter 参数 D	tails 详情
ZBX_DENY_GUI_ACCESS	Enable maintenance mode: 打开维护模式: 1 – maintenance mode is enabled, disabled otherwise 维护模式已打开，其他的数字表示未打开
ZBX_GUI_ACCESS_IP_RANGE	Array of IP addresses, which are allowed to connect to frontend (optional). 允许连接到前端的 IP 地址数组（可选） For example 例: `array('192.168.1.1', '192.168.1.2')`
ZBX_GUI_ACCESS_MESSAGE	A message you can enter to inform users about the maintenance (optional). 您可以输入的消息以通知用户维护（可选）

Display 显示

The following screen will be displayed when trying to access the Zabbix frontend while in maintenance mode. The screen is refreshed every 30 seconds in order to return to a normal state without user intervention when the maintenance is over. 下图显示了在维护模式下访问 Zabbix 前端的情况。屏幕每 30 秒刷新一次，以便在维护结束后，无需用户干预即可恢复正常状态。



IP addresses defined in ZBX_GUI_ACCESS_IP_RANGE will be able to access the frontend as always. 在 ZBX_GUI_ACCESS_IP_RANGE 中定义的 IP 地址也可以一直访问前端。

5 Page parameters 页面参数

Overview 概述

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 (&). 大多数 Zabbix Web 界面页面都支持各种 HTTP GET 参数来控制将要显示的内容。可以通过在 URL 之后指定参数 = 值对来传递它们，通过问号 (?) 与 URL 分隔，并通过＆符号（＆）彼此分隔。

6 Definitions 定义

Overview 概述

While many things in the frontend can be configured using the frontend itself, some customisations are currently only possible by editing a definitions file. 虽然可以使用前端本身配置前端中的许多内容，但目前只能通过编辑定义文件来进行某些自定义。

This file is `defines.inc.php` located in /include of the Zabbix HTML document directory. 该文件是位于/包含 Zabbix HTML 文档目录的’define.inc.php’。

Parameters 参数

Parameters in this file that could be of interest to users 用户可能感兴趣的此文件中的参数:

- ZBX_LOGIN_ATTEMPTS

Number of unsuccessful login attempts that is allowed to an existing system user before a login block in applied (see ZBX_LOGIN_BLOCK). By default 5 attempts. Once the set number of login attempts is tried unsuccessfully, each additional unsuccessful attempt results in a login block. Used with internal authentication only. 应用登录块之前允许现有系统用户的不成功登录尝试次数（请参阅 ZBX_LOGIN_BLOCK）。默认为 5 次尝试。一旦尝试了设置的登录尝试次数失败，则每次额外的不成功尝试都会导致登录阻止。仅与internal身份验证一起使用。

- ZBX_LOGIN_BLOCK

Number of seconds for blocking a user from accessing Zabbix frontend after a number of unsuccessful login attempts (see ZBX_LOGIN_ATTEMPTS). By default 30 seconds. Used with internal authentication only. 在多次登录尝试失败后阻止用户访问 Zabbix 前端的秒数（请参阅 ZBX_LOGIN_ATTEMPTS）。默认为 30 秒。仅与internal身份验证一起使用。

- ZBX_PERIOD_DEFAULT

Default graph period, in seconds. One hour by default. 默认图表周期，以秒为单位。默认为一小时。

- ZBX_MIN_PERIOD

Minimum graph period, in seconds. One hour by default. 最短图表周期，以秒为单位。默认为一小时。

- ZBX_MAX_PERIOD

Maximum graph period, in seconds. Two years by default since 1.6.7, one year before that. 最大图形周期，以秒为单位。一年前，两年后默认为 1.6.7。

- ZBX_HISTORY_PERIOD

The maximum period to display history data in *Latest data*, *Web*, *Overview* pages and *Data overview* screen element in seconds. By default set to 86400 seconds (24 hours). Unlimited period, if set to 0 seconds. 在 *Latest data*, *Web*, *Overview* 页面和 *Data overview* 屏幕元素中以秒显示历史数据的最长期限。默认设置为 86400 秒（24 小时）。无限期，如果设置为 0 秒。

- 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. 在将监控项添加到自定义图形时，简单图形中的 Y 轴的默认位置和下拉框的默认值。可能的值：0 - 左，1 - 右。

Default 默认值: 0

- SCREEN_REFRESH_TIMEOUT (available since 2.0.4)

Used in screens and defines the timeout seconds for a screen element update. When the defined number of seconds after launching an update pass and the screen element has still not been updated, the screen element will be darkened. 用于聚合图形并定义聚合图形元素更新的超时秒数。当启动更新过程后定义的秒数且聚合图形元素仍未更新时，聚合图形元素将变暗。

Default 默认值: 30

- SCREEN_REFRESH_RESPONSIVENESS (available since 2.0.4)

Used in screens and defines the number of seconds after which query skipping will be switched off. Otherwise, if a screen element is in update status all queries on update are skipped until a response is received. With this parameter in use, another update query might be sent after N seconds without having to wait for the response to the first one. 在聚合图形中使用，并定义将关闭查询跳过的秒数。否则，如果聚合图形元素处于更新状态，则会跳过所有更新查询，直到收到响应。使用此参数后，可能会在 N 秒后发送另一个更新查询，而不必等待对第一个的响应。

Default 默认值: 10

- QUEUE_DETAIL_ITEM_COUNT

Defines retrieval limit of the total items queued. Since Zabbix 3.2.4 may be set higher than default value. 定义排队的总监控项的检索限制。由于 Zabbix 3.2.4 可能设置为高于默认值。

Default 默认值: 500

- ZBX_SHOW_SQL_ERRORS (available since 3.4.0)

Show SQL errors in the frontend, if 'true'. If changed to 'false' then SQL errors will still be displayed to all users with *Debug mode* enabled. With *Debug mode* disabled, only Zabbix Super Admin users will see SQL errors. Others will see a generic message: "SQL error. Please contact Zabbix administrator." 如果为'true'，则在前端显示 SQL 错误。如果更改为 "false"，则仍会以调试模式enabled向所有用户显示 SQL 错误。在调试模式禁用的情况下，只有 Zabbix Super Admin 用户会看到 SQL 错误。其他人会看到一条通用消息："SQL 错误。请联系 Zabbix 管理员。"

Default 默认值: true

- VALIDATE_URI_SCHEMES (available since 3.4.5)

Validate a URI against the scheme whitelist defined in ZBX_URI_VALID_SCHEMES. 根据 ZBX_URI_VALID_SCHEMES 中定义的方案白名单验证 URI。

Default 默认值: true

- ZBX_URI_VALID_SCHEMES (available since 3.4.2)

A comma-separated list of allowed URI schemes. Affects all places in the frontend where URIs are used, for example, in map element URLs. 逗号分隔的允许 URI 方案列表。影响使用 URI 的前端中的所有位置，例如，在地图元素 URL 中。

Default 默认值: http,https,ftp,file,mailto,tel,ssh

- ZBX_SHOW_TECHNICAL_ERRORS (available since 3.4.4)

Show technical errors (PHP/SQL) to non-Zabbix Super admin users and to users that are not part of user groups with debug mode enabled. 向非 Zabbix 超级管理员用户以及不启用debug mode的用户组的用户显示技术错误（PHP / SQL）。

Default 默认值: false

- ZBX_SESSION_NAME (available since 4.0.0)

String used as the name of the Zabbix frontend session cookie. 用作 Zabbix 前端会话 cookie 名称的字符串。

Default 默认值: zbx_sessionid

7 Creating your own theme 制定自己的主题

Overview 概述

By default, Zabbix provides a number of predefined themes. You may follow the step-by-step procedure provided here in order to create your own. Feel free to share the result of your work with Zabbix community if you created something nice. 默认情况下，Zabbix 预置了许多主题。您还可以按照以下提供的步骤，制作自定义主题。如果您创作了一些很好的主题，欢迎随时与 Zabbix 社区分享您的工作成果。

Step 1 步骤 1

To define your own theme you'll need to create a CSS file and save it in the *styles/* folder (for example, *custom-theme.css*). You can either copy the files from a different theme and create your theme based on it or start from scratch. 为了制作属于您自己的主题，您需要在 *styles/* 文件夹下创建一个 CSS 文件 (例如：*custom-theme.css*)。您可以从不同的主题复制文件，并据此创建主题，或从头开始创作。

Step 2 步骤 2

Add your theme to the list of themes returned by the Z::getThemes() method. You can do this by overriding the ZBase::getThemes() method in the Z class. This can be done by adding the following code before the closing brace in *include/classes/core/Z.php*: 您可以通过 Z::getThemes() 方法将您的主题添加到主题列表中。您可以通过覆盖 Z 类中的 ZBase::getThemes() 方法来执行此操作。这可以通过在 *include/classes/core/Z.php*: 中的关闭括号之前添加以下代码：

```
public static function getThemes() {
    return array_merge(parent::getThemes(), array(
        'custom-theme' => _('Custom theme')
    ));
}
```

Attention:
Note that the name you specify within the first pair of quotes must match the name of the theme file without extension.
需要注意的是：您在第一对引号内指定的名称必须与没有扩展名的主题文件的名称相匹配。

To add multiple themes, just list them under the first theme, for example: 添加多个主题，只需要将它们罗列在第一个主题下面即可，例如：

```
public static function getThemes() {
    return array_merge(parent::getThemes(), array(
        'custom-theme' => _('Custom theme'),
        'anothertheme' => _('Another theme'),
        'onemoretheme' => _('One more theme')
```

```
    ));
}
```

Note that every theme except the last one must have a trailing comma. 注意：除了最后的一个主题外，其他主题必须用逗号分隔。

Note:
To change graph colours, the entry must be added in the *graph_theme* database table. 为了改变图形颜色，必须在 *graph_theme* 数据库表格中添加该条目。

Step 3 步骤 3

Activate the new theme. 激活新主题

In Zabbix frontend, you may either set this theme to be the default one or change your theme in the user profile. 在 Zabbix 前端，您可以将此主题设置为默认主题或在用户资料改主题。

Enjoy the new look and feel! 享受新的外观吧！

8 Debug mode 调试模式

Overview 概述

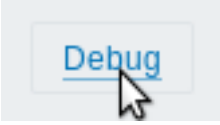
Debug mode may be used to diagnose performance problems with frontend pages. 调试模式可用于诊断前端页面的性能问题。

Configuration 配置

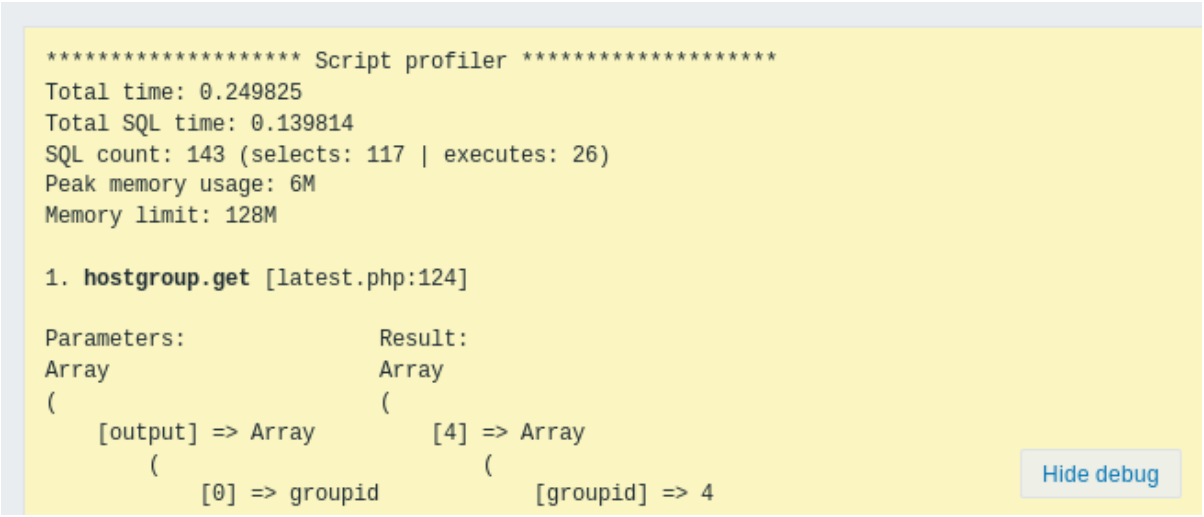
Debug mode can be activated for individual users who belong to a user group: 可为所属用户组的各个用户激活调试模式：

- when configuring a user group; 当配置user group时；
- when viewing configured user groups. 当查看配置user groups时；

When *Debug mode* is enabled for a user group, its users will see a *Debug* button in the lower right corner of the browser window: 当 *Debug mode* 为用户组启用时，其用户将在浏览器窗口的右下角看到 *Debug* 按钮：



Clicking on the *Debug* button opens a new window below the page contents which contains the SQL statistics of the page, along with a list of API calls and individual SQL statements: 单击 *Debug* 按钮将在页面内容下方打开一个新窗口，其中包含页面的 SQL 统计信息，以及 API 调用和各个 SQL 语句的列表：



In case of performance problems with the page, this window may be used to search for the root cause of the problem. 如果页面出现性能问题，可以使用此窗口搜索问题的根本原因。

Warning:
Enabled *Debug mode* negatively affects frontend performance. 启用 *Debug mode* 会对前端造成一定的性能影响。

19. API

Overview Zabbix API allows you to programmatically retrieve and modify the configuration of Zabbix and provides access to historical data. It is widely used to:

- Create new applications to work with Zabbix;
- Integrate Zabbix with third party software;
- Automate routine tasks.

The Zabbix API is a web based API and is shipped as part of the web frontend. It uses the JSON-RPC 2.0 protocol which means two things:

- The API consists of a set of separate methods;
- Requests and responses between the clients and the API are encoded using the JSON format.

More info about the protocol and JSON can be found in the JSON-RPC 2.0 specification and the JSON format homepage.

Structure The API consists of a number of methods that are nominally grouped into separate APIs. Each of the methods performs one specific task. For example, the `host.create` method belongs to the *host* API and is used to create new hosts. Historically, APIs are sometimes referred to as "classes".

> **Note:**
> Most APIs contain at least four methods: `get`, `create`, `update` and `delete` for retrieving, creating, updating and deleting data respectively, but some of the APIs may provide a totally different set of methods.

Performing requests Once you've set up the frontend, you can use remote HTTP requests to call the API. To do that you need to send HTTP POST requests to the `api_jsonrpc.php` file located in the frontend directory. For example, if your Zabbix frontend is installed under *http://company.com/zabbix*, the HTTP request to call the `apiinfo.version` method may look like this:

```
POST http://company.com/zabbix/api_jsonrpc.php HTTP/1.1
Content-Type: application/json-rpc

{"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{}}
```

The request must have the `Content-Type` header set to one of these values: `application/json-rpc`, `application/json` or `application/jsonrequest`.

> **Note:**
> You can use any HTTP client or a JSON-RPC testing tool to perform API requests manually, but for developing applications we suggest you use one of the community maintained libraries.

Example workflow The following section will walk you through some usage examples in more detail.

Authentication Before you can access any data inside of Zabbix you'll need to log in and obtain an authentication token. This can be done using the user.login method. Let us suppose that you want to log in as a standard Zabbix Admin user. Then your JSON request will look like this:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1,
    "auth": null
}
```

Let's take a closer look at the request object. It has the following properties:

- `jsonrpc` - the version of the JSON-RPC protocol used by the API; the Zabbix API implements JSON-RPC version 2.0;
- `method` - the API method being called;
- `params` - parameters that will be passed to the API method;

- `id` - an arbitrary identifier of the request;
- `auth` - a user authentication token; since we don't have one yet, it's set to `null`.

If you provided the credentials correctly, the response returned by the API will contain the user authentication token:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

The response object in turn contains the following properties:

- `jsonrpc` - again, the version of the JSON-RPC protocol;
- `result` - the data returned by the method;
- `id` - identifier of the corresponding request.

Retrieving hosts We now have a valid user authentication token that can be used to access the data in Zabbix. For example, let's use the host.get method to retrieve the IDs, host names and interfaces of all configured hosts:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2,
    "auth": "0424bd59b807674191e7d77572075f33"
}
```

Attention:
Note that the `auth` property is now set to the authentication token we've obtained by calling `user.login`.

The response object will contain the requested data about the hosts:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
    "id": 2
}
```

Note:
For performance reasons we recommend to always list the object properties you want to retrieve and avoid retrieving everything.

Creating a new item Let's create a new item on "Zabbix server" using the data we've obtained from the previous `host.get` request. This can be done by using the item.create method:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on $1",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "10084",
        "type": 0,
        "value_type": 3,
        "interfaceid": "1",
        "delay": 30
    },
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 3
}
```

A successful response will contain the ID of the newly created item, which can be used to reference the item in the following requests:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 3
}
```

Note:
The `item.create` method as well as other create methods can also accept arrays of objects and create multiple items with one API call.

Creating multiple triggers So if create methods accept arrays, we can add multiple triggers like so:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": [
        {
            "description": "Processor load is too high on {HOST.NAME}",
            "expression": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
        },
        {
            "description": "Too many processes on {HOST.NAME}",
            "expression": "{Linux server:proc.num[].avg(5m)}>300",
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 4
}
```

A successful response will contain the IDs of the newly created triggers:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
```

```
    "id": 4
}
```

Updating an item Enable an item, that is, set its status to "0":

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 5
}
```

A successful response will contain the ID of the updated item:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 5
}
```

Note:
The `item.update` method as well as other update methods can also accept arrays of objects and update multiple items with one API call.

Updating multiple triggers Enable multiple triggers, that is, set their status to 0:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": [
        {
            "triggerid": "13938",
            "status": 0
        },
        {
            "triggerid": "13939",
            "status": 0
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 6
}
```

A successful response will contain the IDs of the updated triggers:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938",
            "13939"
        ]
    },
    "id": 6
}
```

Note:
This is the preferred method of updating. Some API methods like `host.massupdate` allow to write more simple code, but it's not recommended to use those methods, since they will be removed in the future releases.

Error handling Up to that point everything we've tried has worked fine. But what happens if we try to make an incorrect call to the API? Let's try to create another host by calling host.create but omitting the mandatory `groups` parameter.

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050"
            }
        ]
    },
    "id": 7,
    "auth": "0424bd59b807674191e7d77572075f33"
}
```

The response will then contain an error message:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "No groups for host \"Linux server\"."
    },
    "id": 7
}
```

If an error occurred, instead of the `result` property, the response object will contain an `error` property with the following data:

- `code` - an error code;
- `message` - a short error summary;
- `data` - a more detailed error message.

Errors can occur in different cases, such as, using incorrect input values, a session timeout or trying to access unexisting objects. Your application should be able to gracefully handle these kinds of errors.

API versions To simplify API versioning, since Zabbix 2.0.4, the version of the API matches the version of Zabbix itself. You can use the apiinfo.version method to find out the version of the API you're working with. This can be useful for adjusting your application to use version-specific features.

We guarantee feature backward compatibility inside of a major version. When making backward incompatible changes between major releases, we usually leave the old features as deprecated in the next release, and only remove them in the release after that. Occasionally, we may remove features between major releases without providing any backward compatibility. It is important that you never rely on any deprecated features and migrate to newer alternatives as soon as possible.

Note:
You can follow all of the changes made to the API in the API changelog.

Further reading You now know enough to start working with the Zabbix API, but don't stop here. For further reading we suggest you have a look at the list of available APIs.

Method reference

This section provides an overview of the functions provided by the Zabbix API and will help you find your way around the available classes and methods.

Monitoring The Zabbix API allows you to access history and other data gathered during monitoring.

History

Retrieve historical values gathered by Zabbix monitoring processes for presentation or further processing.

History API

Trends

Retrieve trend values calculated by Zabbix server for presentation or further processing.

Trend API

Events

Retrieve events generated by triggers, network discovery and other Zabbix systems for more flexible situation management or third-party tool integration.

Event API

Problems

Retrieve problems according to the given parameters.

Problem API

Service monitoring

Retrieve detailed service layer availability information about any service.

Service SLA calculation

Tasks

Task manager allows to check items or low-level discovery rules without config reload.

Task API

Configuration The Zabbix API allows you to manage the configuration of your monitoring system.

Hosts and host groups

Manage host groups, hosts and everything related to them, including host interfaces, host macros and maintenance periods.

Host API | Host group API | Host interface API | User macro API | Maintenance API

Items and applications

Define items to monitor. Create or remove applications and assign items to them.

Item API | Application API

Triggers

Configure triggers to notify you about problems in your system. Manage trigger dependencies.

Trigger API

Graphs

Edit graphs or separate graph items for better presentation of the gathered data.

Graph API | Graph item API

Templates

Manage templates and link them to hosts or other templates.

Template API

Export and import

Export and import Zabbix configuration data for configuration backups, migration or large-scale configuration updates.

Configuration API

Low-level discovery

Configure low-level discovery rules as well as item, trigger and graph prototypes to monitor dynamic entities.

LLD rule API | Item prototype API | Trigger prototype API | Graph prototype API | Host prototype API

Event correlation

Create custom event correlation rules.

Correlation API

Actions and alerts

Define actions and operations to notify users about certain events or automatically execute remote commands. Gain access to information about generated alerts and their receivers.

Action API | Alert API

Services

Manage services for service-level monitoring and retrieve detailed SLA information about any service.

Service API

Dashboards

Manage dashboards.

Dashboard API

Screens

Edit global and template-level screens or each screen item individually.

Screen API | Screen item API | Template screen API | Template screen item API

Maps

Configure maps to create detailed dynamic representations of your IT infrastructure.

Map API

Web monitoring

Configure web scenarios to monitor your web applications and services.

Web scenario API

Network discovery

Manage network-level discovery rules to automatically find and monitor new hosts. Gain full access to information about discovered services and hosts.

Discovery rule API | Discovery check API | Discovery host API | Discovery service API

Administration With the Zabbix API you can change administration settings of your monitoring system.

Users

Add users that will have access to Zabbix, assign them to user groups and grant permissions. Configure media types and the ways users will receive alerts.

User API | User group API | Media type API

General

Change certain global configuration options.

Icon map API | Image API | User macro API

Proxies

Manage the proxies used in your distributed monitoring setup.

Proxy API

Scripts

Configure and execute scripts to help you with your daily tasks.

Script API

API information Retrieve the version of the Zabbix API so that your application could use version-specific features.

API info API

Action 动作

这个类用于操作动作。

对象引用:

- 动作
- 触发动作需要的条件
- 动作触发后联动的操作

相关方法:

- action.create - 创建新的动作
- action.delete - 删除动作
- action.get - 检索动作
- action.update - 更新动作

This class is designed to work with actions.

Object references:

- Action
- Action condition
- Action operation

Available methods:

- action.create - create new actions
- action.delete - delete actions
- action.get - retrieve actions
- action.update - update actions

> Action object

下面是动作（`action`）API 相关的对象。

动作

动作对象具有以下属性。

属性类	描述	
actionid	string	*(readonly)* 动作的ID。

属性类	描述	
esc_period (required)	string	默认操作步骤持续时间。必须大于60秒。接受秒，带后缀的时间单位和用户宏。
eventsource (required)	integer	*(constant)* 动作将处理的事件的类型。 参见event "source" property以获取支持的事件类型列表。

属性类	描述	
name (required)	string	动作的名称。
def_longdata	string	异常消息文本。
def_shortdata	string	异常消息主题。
r_longdata	string	恢复消息文本。
r_shortdata	string	恢复消息主题。
ack_longdata	string	确认操作消息文本。
ack_shortdata	string	确认操作消息主题。

属性类	描述	
status	integer	动作是启动还是禁用。 取值：0 - (默认)启用；1 - 禁用。
pause_suppressed	integer	是否在维护期间暂停升级。 可能的值：0 - 不要暂停升级; 1 - (默认)暂停升级。

动作操作

动作操作对象定义执行动作时执行的操作。它具有以下属性。

属性类	描述	
operationid	string	*(readonly)* 动作操作的ID。

属性类	描述	
operationtype (required)	integer	操作类型 可能的值： 0 - 发送消息; 1 - 远程命令; 2 - 添加主机; 3 - 删除主机; 4 - 添加到主机组; 5 - 从主机组删除; 6 - 链接到模板; 7 - 取消与模板的关联; 8 - 启用主机; 9 - 禁

属性类	描述	
actionid	string	操作所属的动作的ID。

属性类	描述	
esc_period	string	以秒为单位的升级步骤的持续时间。必须大于60秒。接受秒，时间单位后缀和用户宏。如果设置为0或0s，则将使用默认的动作升级周期。 默认: 0s.

属性类	描述	
esc_step_from	integer	步骤开始升级。 默认: 1.
esc_step_to	integer	步骤结束升级。 默认: 1.
evaltype	integer	运行状态计算方法。 可能的值: 0 - (默认) AND / OR; 1 - AND; 2 - OR.

属性类	描述	
opcommand	object	包含操作所运行的命令的数据。 操作命令对象是described in detail below. 远程命令操作所需的。

属性类	描述
opcommand_grp	array 运行远程命令的主机组。 每个对象具有以下属性： opcommand_grp - *(string, read-only)* 对象的ID; operationid - *(string)* 操作ID； groupid - *(string)* 主机组的ID。 如果没有设置opcommand_hst则需要远程命令操作。

属性类	描述
opcommand_hst	array 主机上运行远程命令。 每个对象具有以下属性： opcommand_hst - *(string, read-only)* 对象的ID; operationid - *(string)* 操作ID; hostid - *(string)* 主机ID; 如果设置为0，则命令将在当前主机上运行。 如果没有设

属性类	描述	
opconditions	array	用于触发动作的操作条件 操作条件对象是下面详细描述.

属性类	描述	
opgroup	array	用于添加主机的主机组。 每个对象都具有以下属性: `operationid` - *(string)* ID of the operation; `groupid` - *(string)* 主机组的ID。 添加到主机组和从主机组中删除操作所必需的。

属性类	描述	
opmessage	object	包含有关操作发送的消息的数据的对象。 操作消息对象是下面详细描述. 消息操作必需。

属性类	描述	
opmessage_grp	array	要发送消息的用户组。 每个对象都具有以下属性： operationid - *(string)* 操作ID; usrgrpid - *(string)* 用户组的ID。 如果未设置opmessage_usr则消息操作必需。

属性类	描述	
opmessage_usr	array	发送消息给的用户。 每个对象都具有以下属性： operationid - *(string)* 操作ID; userid - *(string)* 用户的ID。 如果未设置opmessage_grp则消息操作必需。

属性类	描述	
optemplate	array	用于将主机链接到的模板。 每个对象都具有以下属性： `operationid` - *(string)* 操作ID; `templateid` - *(string)* 模板ID. 必须有”link to template”和”unlink from template”操作

属性类	描述	
opinventory	object	库存模式设置主机。 每个对象都具有以下属性: `operationid` - *(string)* 操作ID; `inventory_mod` - *(string)* Inventory mode. 需要有”Set host inventory mode”操作。

动作操作命令

操作命令对象包含有关运行操作命令的数据。

属性类	说明	
operationid	string	*(readonly)* 操作ID.

属性类	说明	
command	string	要运行的命令。当类型为(0,1,2,3)时，此项是必须的
type (required)	integer	操作命令的类型 可能的值： 0 - custom script; 1 - IPMI; 2 - SSH; 3 - Telnet; 4 - global script.
authtype	integer	SSH命令的认证方法。 可能的值： 0 - password; 1 - public key. Required for SSH commands.

属性类	说明	
execute_on	integer	将要执行自定义脚本操作命令的目标。 可能的值: 0 - Zabbix agent; 1 - Zabbix server; 2 - Zabbix server (proxy). 自定义脚本命令所需的。
password	string	密码验证和 telnet 命令时用于 SSH 命令的密码。
port	string	用于 SSH 和 telnet 命令的端口号。

属性类	说明	
privatekey	string	使用公钥认证的SSH命令的私钥文件的名称。 具有密钥验证的SSH命令所必需的。
publickey	string	用于SSH公钥和公钥认证的公钥名称。 具有密钥验证的SSH命令所必需的。
scriptid	string	用于全局脚本命令的脚本ID。 需要全局脚本命令。

属性类	说明	
username	string	用于登录认证的用户名 使用SSH和Telnet命令时是必须的.

动作操作消息

操作消息对象包含有关将由操作发送的消息的数据。

属性类	说明	
operationid	string	*(readonly)* 动作操作的ID

属性类	说明	
default_msg	integer	是否使用默认动作消息文本和主题。 可能的值: 0 - *(default)* 使用操作中的消息文本和主题 1 - 使用动作中的消息文本和主题

属性类	说明	
mediatypeid	string	将用于发送消息的媒体类型ID。
message	string	操作消息文本。
subject	string	操作消息主题。

动作操作条件

动作操作条件对象定义了一个必须满足的条件来执行当前操作。它具有以下属性。

属性类	说明	
opconditionid	string	*(readonly)* 动作操作条件的ID
conditiontype (required)	integer	条件的类型。 可能的值: 14 - event ac-knowl-edged.
value (required)	string	与之比较的值。

属性类	说明	
operationid	string	*(readonly)* 动作操作的ID
operator	integer	条件运算符 可能的值： 0 - *(default)* =.

每个操作条件类型都支持以下运算符和值。

条件条	名称支持的运算	期望值	
14	Event acknowledged	=	件是否被确认。 可能的值: 0 - 没有确认; 1 - 已确认。

Action recovery operation

The action recovery operation object defines an operation that will be performed when a problem is resolved. Recovery operations are possible for trigger actions and internal actions. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 11 - notify all involved. Possible values for internal actions: 0 - send message; 11 - notify all involved.
actionid	string	ID of the action that the recovery operation belongs to.
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above. Required for remote command operations.

Property	Type	Description
opcommand_grp	array	Host groups to run remote commands on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for remote command operations if `opcommand_hst` is not set.
opcommand_hst	array	Host to run remote commands on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host. Required for remote command operations if `opcommand_grp` is not set.
opmessage	object	Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above. Required for message operations.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `usrgrpid` - *(string)* ID of the user group. Required for message operations if `opmessage_usr` is not set.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `opmessage_grp` is not set.

Action acknowledge operation

The action acknowledge operation object defines an operation that will be performed when a problem is acknowledged. Acknowledge operations are possible for trigger actions. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 12 - notify all involved.

Property	Type	Description
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above.
opcommand_grp	array	Required for remote command operations. Host groups to run remote commands on. Each object has the following properties: `groupid` - *(string)* ID of the host group.
opcommand_hst	array	Required for remote command operations if `opcommand_hst` is not set. Host to run remote commands on. Each object has the following properties: `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command operations if `opcommand_grp` is not set. Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `usrgrpid` - *(string)* ID of the user group. Required only for `send message` operations if `opmessage_usr` is not set. Is ignored for `send acknowledge message` operations.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `userid` - *(string)* ID of the user. Required only for `send message` operations if `opmessage_grp` is not set. Is ignored for `send acknowledge message` operations.

Action filter

The action filter object defines a set of conditions that must be met to perform the configured action operations. It has the following properties.

Property	Type	Description
conditions (required)	array	Set of filter conditions to use for filtering results.

Property	Type	Description
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(readonly)* Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression.
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Required for custom expression filters.

Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

Property	Type	Description
conditionid	string	*(readonly)* ID of the action condition.

Property	Type	Description
conditiontype (required)	integer	Type of condition. Possible values for trigger actions: 0 - host group; 1 - host; 2 - trigger; 3 - trigger name; 4 - trigger severity; 6 - time period; 13 - host template; 15 - application; 16 - maintenance status; 25 - event tag; 26 - event tag value. Possible values for discovery actions: 7 - host IP; 8 - discovered service type; 9 - discovered service port; 10 - discovery status; 11 - uptime or downtime duration; 12 - received value; 18 - discovery rule; 19 - discovery check; 20 - proxy; 21 - discovery object. Possible values for auto-registration actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions: 0 - host group; 1 - host; 13 - host template; 15 - application; 23 - event type.
value (required)	string	Value to compare with.
value2	string	Secondary value to compare with. Requried for trigger actions when condition type is *26*.
actionid	string	*(readonly)* ID of the action that the condition belongs to.
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 0 - *(default)* =; 1 - <>; 2 - like; 3 - not like; 4 - in; 5 - >=; 6 - <=; 7 - not in.

Note:
To better understand how to use filters with various types of expressions, see examples on the action.get and action.create method pages.

The following operators and values are supported for each condition type.

Condition	Condition name	Supported operators	Expected value
0	Host group	=, <>	Host group ID.
1	Host	=, <>	Host ID.
2	Trigger	=, <>	Trigger ID.
3	Trigger name	like, not like	Trigger name.
4	Trigger severity	=, <>, >=, <=	Trigger severity. Refer to the trigger "severity" property for a list of supported trigger severities.
5	Trigger value	=	Trigger value. Refer to the trigger "value" property for a list of supported trigger values.
6	Time period	in, not in	Time when the event was triggered as a time period.
7	Host IP	=, <>	One or several IP ranges to check separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.
8	Discovered service type	=, <>	Type of discovered service. The type of service matches the type of the discovery check used to detect the service. Refer to the discovery check "type" property for a list of supported types.
9	Discovered service port	=, <>	One or several port ranges separated by commas.
10	Discovery status	=	Status of a discovered object. Possible values: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost.
11	Uptime or downtime duration	>=, <=	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	=, <>, >=, <=, like, not like	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	=, <>	Linked template ID.
15	Application	=, like, not like	Name of the application.

Condition	Condition name	Supported operators	Expected value
16	Maintenance status	in, not in	No value required: using the ”in” operator means that the host must be in maintenance, ”not in” - not in maintenance.
18	Discovery rule	=, <>	ID of the discovery rule.
19	Discovery check	=, <>	ID of the discovery check.
20	Proxy	=, <>	ID of the proxy.
21	Discovery object	=	Type of object that triggered the discovery event. Possible values: 1 - discovered host; 2 - discovered service.
22	Host name	like, not like	Host name.
23	Event type	=	Specific internal event. Possible values: 0 - item in ”not supported” state; 1 - item in ”normal” state; 2 - LLD rule in ”not supported” state; 3 - LLD rule in ”normal” state; 4 - trigger in ”unknown” state; 5 - trigger in ”normal” state.
24	Host metadata	like, not like	Metadata of the auto-registered host.
25	Tag	=, <>, like, not like	Event tag.
26	Tag value	=, <>, like, not like	Event tag value.

The following objects are directly related to the `action` API.

Action

The action object has the following properties.

Property	Type	Description
actionid	string	*(readonly)* ID of the action.
esc_period (required)	string	Default operation step duration. Must be greater than 60 seconds. Accepts seconds, time unit with suffix and user macro.
eventsource (required)	integer	*(constant)* Type of events that the action will handle. Refer to the event ”source” property for a list of supported event types.
name (required)	string	Name of the action.
def_longdata	string	Problem message text.
def_shortdata	string	Problem message subject.
r_longdata	string	Recovery message text.
r_shortdata	string	Recovery message subject.
ack_longdata	string	Acknowledge operation message text.
ack_shortdata	string	Acknowledge operation message subject.
status	integer	Whether the action is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

Property	Type	Description
pause_suppressed	integer	Whether to pause escalation during maintenance periods or not. Possible values: 0 - Don't pause escalation; 1 - *(default)* Pause escalation.

Action operation

The action operation object defines an operation that will be performed when an action is executed. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values: 0 - send message; 1 - remote command; 2 - add host; 3 - remove host; 4 - add to host group; 5 - remove from host group; 6 - link to template; 7 - unlink from template; 8 - enable host; 9 - disable host; 10 - set host inventory mode.
actionid	string	ID of the action that the operation belongs to.
esc_period	string	Duration of an escalation step in seconds. Must be greater than 60 seconds. Accepts seconds, time unit with suffix and user macro. If set to 0 or 0s, the default action escalation period will be used. Default: 0s.
esc_step_from	integer	Step to start escalation from. Default: 1.
esc_step_to	integer	Step to end escalation at. Default: 1.
evaltype	integer	Operation condition evaluation method. Possible values: 0 - *(default)* AND / OR; 1 - AND; 2 - OR.
opcommand	object	Object containing the data about the command run by the operation. The operation command object is described in detail below. Required for remote command operations.

Property	Type	Description
opcommand_grp	array	Host groups to run remote commands on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for remote command operations if `opcommand_hst` is not set.
opcommand_hst	array	Host to run remote commands on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host. Required for remote command operations if `opcommand_grp` is not set.
opconditions	array	Operation conditions used for trigger actions. The operation condition object is described in detail below.
opgroup	array	Host groups to add hosts to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for ”add to host group” and ”remove from host group” operations.
opmessage	object	Object containing the data about the message sent by the operation. The operation message object is described in detail below. Required for message operations.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `usrgrpid` - *(string)* ID of the user group. Required for message operations if `opmessage_usr` is not set.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `opmessage_grp` is not set.
optemplate	array	Templates to link the hosts to to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `templateid` - *(string)* ID of the template. Required for ”link to template” and ”unlink from template” operations.

Property	Type	Description
opinventory	object	Inventory mode set host to. Object has the following properties: `operationid` - *(string)* ID of the operation; `inventory_mode` - *(string)* Inventory mode. Required for ”Set host inventory mode” operations.

Action operation command

The operation command object contains data about the command that will be run by the operation.

Property	Type	Description
operationid	string	*(readonly)* ID of the operation.
command	string	Command to run. Required when type IN (0,1,2,3)
type (required)	integer	Type of operation command. Possible values: 0 - custom script; 1 - IPMI; 2 - SSH; 3 - Telnet; 4 - global script.
authtype	integer	Authentication method used for SSH commands. Possible values: 0 - password; 1 - public key. Required for SSH commands.
execute_on	integer	Target on which the custom script operation command will be executed. Possible values: 0 - Zabbix agent; 1 - Zabbix server; 2 - Zabbix server (proxy). Required for custom script commands.
password	string	Password used for SSH commands with password authentication and Telnet commands.
port	string	Port number used for SSH and Telnet commands.
privatekey	string	Name of the private key file used for SSH commands with public key authentication. Required for SSH commands with public key authentication.
publickey	string	Name of the public key file used for SSH commands with public key authentication. Required for SSH commands with public key authentication.
scriptid	string	ID of the script used for global script commands. Required for global script commands.
username	string	User name used for authentication. Required for SSH and Telnet commands.

Action operation message

The operation message object contains data about the message that will be sent by the operation.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
default_msg	integer	Whether to use the default action message text and subject. Possible values: 0 - *(default)* use the data from the operation; 1 - use the data from the action.
mediatypeid	string	ID of the media type that will be used to send the message.
message	string	Operation message text.
subject	string	Operation message subject.

Action operation condition

The action operation condition object defines a condition that must be met to perform the current operation. It has the following properties.

Property	Type	Description
opconditionid	string	*(readonly)* ID of the action operation condition
conditiontype (required)	integer	Type of condition. Possible values: 14 - event acknowledged.
value (required)	string	Value to compare with.
operationid	string	*(readonly)* ID of the operation.
operator	integer	Condition operator. Possible values: 0 - *(default)* =.

The following operators and values are supported for each operation condition type.

Condition	Condition name	Supported operators	Expected value
14	Event acknowledged	=	Whether the event is acknowledged. Possible values: 0 - not acknowledged; 1 - acknowledged.

Action recovery operation

The action recovery operation object defines an operation that will be performed when a problem is resolved. Recovery operations are possible for trigger actions and internal actions. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.

Property	Type	Description
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 11 - notify all involved. Possible values for internal actions: 0 - send message; 11 - notify all involved.
actionid	string	ID of the action that the recovery operation belongs to.
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above.
opcommand_grp	array	Required for remote command operations. Host groups to run remote commands on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group.
opcommand_hst	array	Required for remote command operations if `opcommand_hst` is not set. Host to run remote commands on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command operations if `opcommand_grp` is not set. Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above.
opmessage_grp	array	Required for message operations. User groups to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `usrgrpid` - *(string)* ID of the user group.
opmessage_usr	array	Required for message operations if `opmessage_usr` is not set. Users to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `opmessage_grp` is not set.

Action acknowledge operation

The action acknowledge operation object defines an operation that will be performed when a problem is acknowledged. Acknowledge operations are possible for trigger actions. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 12 - notify all involved.
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above.
opcommand_grp	array	Required for remote command operations. Host groups to run remote commands on. Each object has the following properties: `groupid` - *(string)* ID of the host group.
opcommand_hst	array	Required for remote command operations if `opcommand_hst` is not set. Host to run remote commands on. Each object has the following properties: `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command operations if `opcommand_grp` is not set. Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `usrgrpid` - *(string)* ID of the user group. Required only for `send message` operations if `opmessage_usr` is not set. Is ignored for `send acknowledge message` operations.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `userid` - *(string)* ID of the user. Required only for `send message` operations if `opmessage_grp` is not set. Is ignored for `send acknowledge message` operations.

Action filter

The action filter object defines a set of conditions that must be met to perform the configured action operations. It has the following properties.

Property	Type	Description
conditions (required)	array	Set of filter conditions to use for filtering results.
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(readonly)* Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression.
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Required for custom expression filters.

Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

Property	Type	Description
conditionid	string	*(readonly)* ID of the action condition.

Property	Type	Description
conditiontype (required)	integer	Type of condition. Possible values for trigger actions: 0 - host group; 1 - host; 2 - trigger; 3 - trigger name; 4 - trigger severity; 6 - time period; 13 - host template; 15 - application; 16 - maintenance status; 25 - event tag; 26 - event tag value. Possible values for discovery actions: 7 - host IP; 8 - discovered service type; 9 - discovered service port; 10 - discovery status; 11 - uptime or downtime duration; 12 - received value; 18 - discovery rule; 19 - discovery check; 20 - proxy; 21 - discovery object. Possible values for auto-registration actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions: 0 - host group; 1 - host; 13 - host template; 15 - application; 23 - event type.
value (required)	string	Value to compare with.
value2	string	Secondary value to compare with. Requried for trigger actions when condition type is *26*.
actionid	string	*(readonly)* ID of the action that the condition belongs to.
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 0 - *(default)* =; 1 - <>; 2 - like; 3 - not like; 4 - in; 5 - >=; 6 - <=; 7 - not in.

Note:
To better understand how to use filters with various types of expressions, see examples on the action.get and action.create method pages.

The following operators and values are supported for each condition type.

Condition	Condition name	Supported operators	Expected value
0	Host group	=, <>	Host group ID.
1	Host	=, <>	Host ID.
2	Trigger	=, <>	Trigger ID.
3	Trigger name	like, not like	Trigger name.
4	Trigger severity	=, <>, >=, <=	Trigger severity. Refer to the trigger "severity" property for a list of supported trigger severities.
5	Trigger value	=	Trigger value. Refer to the trigger "value" property for a list of supported trigger values.
6	Time period	in, not in	Time when the event was triggered as a time period.
7	Host IP	=, <>	One or several IP ranges to check separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.
8	Discovered service type	=, <>	Type of discovered service. The type of service matches the type of the discovery check used to detect the service. Refer to the discovery check "type" property for a list of supported types.
9	Discovered service port	=, <>	One or several port ranges separated by commas.
10	Discovery status	=	Status of a discovered object. Possible values: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost.
11	Uptime or downtime duration	>=, <=	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	=, <>, >=, <=, like, not like	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	=, <>	Linked template ID.
15	Application	=, like, not like	Name of the application.

Condition	Condition name	Supported operators	Expected value
16	Maintenance status	in, not in	No value required: using the ”in” operator means that the host must be in maintenance, ”not in” - not in maintenance.
18	Discovery rule	=, <>	ID of the discovery rule.
19	Discovery check	=, <>	ID of the discovery check.
20	Proxy	=, <>	ID of the proxy.
21	Discovery object	=	Type of object that triggered the discovery event. Possible values: 1 - discovered host; 2 - discovered service.
22	Host name	like, not like	Host name.
23	Event type	=	Specific internal event. Possible values: 0 - item in ”not supported” state; 1 - item in ”normal” state; 2 - LLD rule in ”not supported” state; 3 - LLD rule in ”normal” state; 4 - trigger in ”unknown” state; 5 - trigger in ”normal” state.
24	Host metadata	like, not like	Metadata of the auto-registered host.
25	Tag	=, <>, like, not like	Event tag.
26	Tag value	=, <>, like, not like	Event tag value.

action.create

说明

`object action.create(object/array actions)`

此方法用于创建新动作。

参数

`(object/array)` 创建新动作

除此之外标准动作属性, 该方法接受以下参数。

参数类	说明	
filter	object	动作的动作过滤器对象。
operations	array	为动作创建的动作操作。
recovery_operations	array	为动作创建动作恢复操作。
acknowledge_operations	array	为动作创建动作确认操作。

返回值

`(object)` 返回一个对象，其中 `actionids` 属性下包含已创建动作的 ID。返回的 ID 的顺序与传递的操作的顺序相匹配。

范例

创建触发器动作

创建一个动作，动作如下描述，当主机 30045 ，它的触发器中的 `memory` 进入问题状态时。该动作必须首先向用户组 7 中的所有用户发送消息。如果事件在 4 分钟内未被解决，它将在 2 组中的所有主机上运行脚本 3。在触发恢复中，它将通知所有接收到关于该问题的消息的用户。在触发器确认中，带有自定义主体和主体的消息将通过所有媒体类型发送给所有确认和评论的所有人。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
        "esc_period": "2m",
        "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084"
                },
                {
                    "conditiontype": 3,
                    "operator": 2,
                    "value": "memory"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_period": "0s",
                "esc_step_from": 1,
                "esc_step_to": 2,
                "evaltype": 0,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            },
            {
                "operationtype": 1,
                "esc_step_from": 3,
                "esc_step_to": 4,
                "evaltype": 0,
                "opconditions": [
                    {
                        "conditiontype": 14,
                        "operator": 0,
                        "value": "0"
                    }
                ],
                "opcommand_grp": [
                    {
                        "groupid": "2"
                    }
                ],
                "opcommand": {
                    "type": 4,
```

```
                    "scriptid": "3"
                }
            }
        ],
        "recovery_operations": [
            {
                "operationtype": "11",
                "opmessage": {
                    "default_msg": 1
                }
            }
        ],
        "acknowledge_operations": [
            {
                "operationtype": "12",
                "opmessage": {
                    "message": "Custom acknowledge operation message body",
                    "subject": "Custom acknowledge operation message subject"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17"
        ]
    },
    "id": 1
}
```

创建发现动作

创建一个将发现的主机链接到模板 30085 的动作。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Discovery action",
        "eventsource": 1,
        "status": 0,
        "esc_period": "0s",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "value": "2"
                }
            ]
        },
```

```
        "operations": [
            {
                "esc_step_from": 1,
                "esc_period": "0s",
                "optemplate": [
                    {
                        "templateid": "10091"
                    }
                ],
                "operationtype": 6,
                "esc_step_to": 1
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

使用自定义表达式筛选器

创建使用自定义筛选器条件的触发器动作。该动作必须为每个触发器发送一个消息，其严重程度高于或等于主机 10084 和 10106 的 警告。公式 ID A、B 和 C 都是任意选择的。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
        "esc_period": "2m",
        "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.
        "filter": {
            "evaltype": 3,
            "formula": "A and (B or C)",
            "conditions": [
                {
                    "conditiontype": 4,
                    "operator": 5,
                    "value": "2",
                    "formulaid": "A"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084",
                    "formulaid": "B"
                },
                {
                    "conditiontype": 1,
```

```
                    "operator": 0,
                    "value": "10106",
                    "formulaid": "C"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_period": "0s",
                "esc_step_from": 1,
                "esc_step_to": 2,
                "evaltype": 0,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

参见

- Action filter
- Action operation

来源

CAction::create() in *frontends/php/include/classes/api/services/CAction.php*.

Description

`object action.create(object/array actions)`

This method allows to create new actions.

Parameters

`(object/array)` Actions to create.

Additionally to the standard action properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	Action filter object for the action.
operations	array	Action operations to create for the action.
recovery_operations	array	Action recovery operations to create for the action.

Parameter	Type	Description
acknowledge_operations	array	Action acknowledge operations to create for the action.

Return values

(object) Returns an object containing the IDs of the created actions under the actionids property. The order of the returned IDs matches the order of the passed actions.

Examples

Create a trigger action

Create an action that will be run when a trigger from host "30045" that has the word "memory" in its name goes into problem state. The action must first send a message to all users in user group "7". If the event is not resolved in 4 minutes, it will run script "3" on all hosts in group "2". On trigger recovery it will notify all users who received any messages regarding the problem before. On trigger acknowledge, message with custom subject and body will be sent to all who left acknowledgements and comments via all media types.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
        "esc_period": "2m",
        "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084"
                },
                {
                    "conditiontype": 3,
                    "operator": 2,
                    "value": "memory"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_period": "0s",
                "esc_step_from": 1,
                "esc_step_to": 2,
                "evaltype": 0,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            },
            {
                "operationtype": 1,
```

```
                "esc_step_from": 3,
                "esc_step_to": 4,
                "evaltype": 0,
                "opconditions": [
                    {
                        "conditiontype": 14,
                        "operator": 0,
                        "value": "0"
                    }
                ],
                "opcommand_grp": [
                    {
                        "groupid": "2"
                    }
                ],
                "opcommand": {
                    "type": 4,
                    "scriptid": "3"
                }
            }
        ],
        "recovery_operations": [
            {
                "operationtype": "11",
                "opmessage": {
                    "default_msg": 1
                }
            }
        ],
        "acknowledge_operations": [
            {
                "operationtype": "12",
                "opmessage": {
                    "message": "Custom acknowledge operation message body",
                    "subject": "Custom acknowledge operation message subject"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17"
        ]
    },
    "id": 1
}
```

Create a discovery action

Create an action that will link discovered hosts to template ”30085”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
```

```
        "name": "Discovery action",
        "eventsource": 1,
        "status": 0,
        "esc_period": "0s",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "value": "2"
                }
            ]
        },
        "operations": [
            {
                "esc_step_from": 1,
                "esc_period": "0s",
                "optemplate": [
                    {
                        "templateid": "10091"
                    }
                ],
                "operationtype": 6,
                "esc_step_to": 1
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

Using a custom expression filter

Create a trigger action that will use a custom filter condition. The action must send a message for each trigger with severity higher or equal to ”Warning” for hosts ”10084” and ”10106”. The formula IDs ”A”, ”B” and ”C” have been chosen arbitrarily.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
        "esc_period": "2m",
        "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.
        "filter": {
```

```
            "evaltype": 3,
            "formula": "A and (B or C)",
            "conditions": [
                {
                    "conditiontype": 4,
                    "operator": 5,
                    "value": "2",
                    "formulaid": "A"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084",
                    "formulaid": "B"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10106",
                    "formulaid": "C"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_period": "0s",
                "esc_step_from": 1,
                "esc_step_to": 2,
                "evaltype": 0,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

See also

- Action filter
- Action operation

Source

CAction::create() in *frontends/php/include/classes/api/services/CAction.php*.

action.delete

说明

`object action.delete(array actionIds)`

此方法用于删除动作。

参数

(array) 要删除的动作的 ID。

返回值

(object) 返回一个对象，该对象在 actionids 属性下包含要删除的动作的 ID。

范例

删除多个动作

删除两个动作。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.delete",
    "params": [
        "17",
        "18"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17",
            "18"
        ]
    },
    "id": 1
}
```

来源

CAction::delete() in *frontends/php/include/classes/api/services/CAction.php*.

Description

`object action.delete(array actionIds)`

This method allows to delete actions.

Parameters

(array) IDs of the actions to delete.

Return values

(object) Returns an object containing the IDs of the deleted actions under the actionids property.

Examples

Delete multiple actions

Delete two actions.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.delete",
    "params": [
        "17",
        "18"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17",
            "18"
        ]
    },
    "id": 1
}
```

Source

CAction::delete() in *frontends/php/include/classes/api/services/CAction.php*.

action.get

说明

`integer/array action.get(object parameters)`

该方法允许根据给定的参数检索动作。

参数

`(object)` 定义期望输出的参数。

该方法支持以下参数。

参数类	说明	
actionids	string/array	只返回给定ID的动作。

参数类	说明	
groupids	string/array	只返回在操作条件下使用给定主机组的动作。
hostids	string/array	只返回在操作条件下使用给定主机的动作。
triggerids	string/array	只返回在操作条件下使用给定触发器的动作。

参数类	说明	
mediatypeids	string/array	只返回使用给定媒体类型发送消息的动作。
usrgrpids	string/array	仅返回配置为向给定用户组发送消息的动作。
userids	string/array	仅返回配置为向给定用户发送消息的动作。

参数类	说明	
scriptids	string/array	只返回配置为运行给定脚本的动作。
selectFilter	query	返回`filter`属性中的动作筛选器。
selectOperations	query	在操作属性中返回操作操作。
selectRecoveryOperations	query	在恢复操作属性中返回动作恢复操作。

参数类	说明	
selectAcknowledgeOperations	query	在确认操作属性中返回动作确认操作。
sortfield	string/array	根据给定的属性排序结果。 可能的值是: `actionid`, `name` and `status`.
countOutput	boolean	这些参数对于所有 `get` 方法都是常见的。在reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	

参数类	说明
limit	integer
output	query
preservekeys	boolean
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	boolean

返回值

(integer/array) 也返回:

- 对象数组;
- 如果使用了 curtOutlook 参数，则检索对象的计数。

范例

检索发现动作

Retrieve all configured discovery actions together with action conditions and operations. The filter uses the "and" evaluation type, so the formula property is empty and eval_formula is generated automatically. 检索所有配置的发现动作以及操作条件和操作。筛选器使用 and 评估类型，因此 formula 属性为空，自动生成 eval_formula 。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectRecoveryOperations": "extend",
        "selectFilter": "extend",
        "filter": {
            "eventsource": 1
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "2",
            "name": "Auto discovery. Linux servers.",
            "eventsource": "1",
            "status": "1",
            "esc_period": "0s",
            "def_shortdata": "",
            "def_longdata": "",
            "r_shortdata": "",
            "r_longdata": "",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "conditiontype": "10",
                        "operator": "0",
```

```
                    "value": "0",
                    "value2": "",
                    "formulaid": "B"
                },
                {
                    "conditiontype": "8",
                    "operator": "0",
                    "value": "9",
                    "value2": "",
                    "formulaid": "C"
                },
                {
                    "conditiontype": "12",
                    "operator": "2",
                    "value": "Linux",
                    "value2": "",
                    "formulaid": "A"
                }
            ],
            "eval_formula": "A and B and C"
        },
        "operations": [
            {
                "operationid": "1",
                "actionid": "2",
                "operationtype": "6",
                "esc_period": "0s",
                "esc_step_from": "1",
                "esc_step_to": "1",
                "evaltype": "0",
                "opconditions": [],
                "optemplate": [
                    {
                        "operationid": "1",
                        "templateid": "10001"
                    }
                ]
            },
            {
                "operationid": "2",
                "actionid": "2",
                "operationtype": "4",
                "esc_period": "0s",
                "esc_step_from": "1",
                "esc_step_to": "1",
                "evaltype": "0",
                "opconditions": [],
                "opgroup": [
                    {
                        "operationid": "2",
                        "groupid": "2"
                    }
                ]
            }
        ],
        "recoveryOperations": [
            {
                "operationid": "585",
                "actionid": "2",
                "operationtype": "11",
                "evaltype": "0",
                "opconditions": [],
```

```
                "opmessage": {
                    "operationid": "585",
                    "default_msg": "1",
                    "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
                    "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger
                    "mediatypeid": "0"
                }
            }
        ],
        "acknowledgeOperations": [
            {
                "operationid": "585",
                "operationtype": "12",
                "evaltype": "0",
                "opmessage": {
                    "default_msg": "1",
                    "subject": "Acknowledged: {TRIGGER.NAME}",
                    "message": "{USER.FULLNAME} acknowledged problem at {ACK.DATE} {ACK.TIME} with the
                    "mediatypeid": "0"
                }
            },
            {
                "operationid": "586",
                "operationtype": "0",
                "evaltype": "0",
                "opmessage": {
                    "default_msg": "1",
                    "subject": "Acknowledged: {TRIGGER.NAME}",
                    "message": "{USER.FULLNAME} acknowledged problem at {ACK.DATE} {ACK.TIME} with the
                    "mediatypeid": "0"
                },
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage_usr": []
            },
            {
                "operationid": "587",
                "operationtype": "1",
                "evaltype": "0",
                "opcommand": {
                    "type": "0",
                    "scriptid": "0",
                    "execute_on": "0",
                    "port": "",
                    "authtype": "0",
                    "username": "",
                    "password": "",
                    "publickey": "",
                    "privatekey": "",
                    "command": "notify.sh"
                },
                "opcommand_hst": [
                    {
                        "hostid": "0"
                    }
                ],
                "opcommand_grp": []
            }
        ]
```

```
        }
    ],
    "id": 1
}
```

参见

- Action filter
- Action operation

来源

CAction::get() in *frontends/php/include/classes/api/services/CAction.php*.

Description

`integer/array action.get(object parameters)`

The method allows to retrieve actions according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
actionids	string/array	Return only actions with the given IDs.
groupids	string/array	Return only actions that use the given host groups in action conditions.
hostids	string/array	Return only actions that use the given hosts in action conditions.
triggerids	string/array	Return only actions that use the given triggers in action conditions.
mediatypeids	string/array	Return only actions that use the given media types to send messages.
usrgrpids	string/array	Return only actions that are configured to send messages to the given user groups.
userids	string/array	Return only actions that are configured to send messages to the given users.
scriptids	string/array	Return only actions that are configured to run the given scripts.
selectFilter	query	Returns the action filter in the `filter` property.
selectOperations	query	Return action operations in the `operations` property.
selectRecoveryOperations	query	Return action recovery operations in the `recoveryOperations` property.
selectAcknowledgeOperations	query	Return action acknowledge operations in the `acknowledgeOperations` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `actionid`, `name` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve discovery actions

Retrieve all configured discovery actions together with action conditions and operations. The filter uses the "and" evaluation type, so the formula property is empty and eval_formula is generated automatically.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectRecoveryOperations": "extend",
        "selectFilter": "extend",
        "filter": {
            "eventsource": 1
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "2",
            "name": "Auto discovery. Linux servers.",
            "eventsource": "1",
            "status": "1",
            "esc_period": "0s",
            "def_shortdata": "",
            "def_longdata": "",
            "r_shortdata": "",
            "r_longdata": "",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "conditiontype": "10",
                        "operator": "0",
                        "value": "0",
                        "value2": "",
                        "formulaid": "B"
                    },
                    {
                        "conditiontype": "8",
                        "operator": "0",
                        "value": "9",
                        "value2": "",
                        "formulaid": "C"
                    },
                    {
                        "conditiontype": "12",
                        "operator": "2",
```

```
                "value": "Linux",
                "value2": "",
                "formulaid": "A"
            }
        ],
        "eval_formula": "A and B and C"
    },
    "operations": [
        {
            "operationid": "1",
            "actionid": "2",
            "operationtype": "6",
            "esc_period": "0s",
            "esc_step_from": "1",
            "esc_step_to": "1",
            "evaltype": "0",
            "opconditions": [],
            "optemplate": [
                {
                    "operationid": "1",
                    "templateid": "10001"
                }
            ]
        },
        {
            "operationid": "2",
            "actionid": "2",
            "operationtype": "4",
            "esc_period": "0s",
            "esc_step_from": "1",
            "esc_step_to": "1",
            "evaltype": "0",
            "opconditions": [],
            "opgroup": [
                {
                    "operationid": "2",
                    "groupid": "2"
                }
            ]
        }
    ],
    "recoveryOperations": [
        {
            "operationid": "585",
            "actionid": "2",
            "operationtype": "11",
            "evaltype": "0",
            "opconditions": [],
            "opmessage": {
                "operationid": "585",
                "default_msg": "1",
                "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
                "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger
                "mediatypeid": "0"
            }
        }
    ],
    "acknowledgeOperations": [
        {
            "operationid": "585",
            "operationtype": "12",
            "evaltype": "0",
```

```
                    "opmessage": {
                        "default_msg": "1",
                        "subject": "Acknowledged: {TRIGGER.NAME}",
                        "message": "{USER.FULLNAME} acknowledged problem at {ACK.DATE} {ACK.TIME} with the
                        "mediatypeid": "0"
                    }
                },
                {
                    "operationid": "586",
                    "operationtype": "0",
                    "evaltype": "0",
                    "opmessage": {
                        "default_msg": "1",
                        "subject": "Acknowledged: {TRIGGER.NAME}",
                        "message": "{USER.FULLNAME} acknowledged problem at {ACK.DATE} {ACK.TIME} with the
                        "mediatypeid": "0"
                    },
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ],
                    "opmessage_usr": []
                },
                {
                    "operationid": "587",
                    "operationtype": "1",
                    "evaltype": "0",
                    "opcommand": {
                        "type": "0",
                        "scriptid": "0",
                        "execute_on": "0",
                        "port": "",
                        "authtype": "0",
                        "username": "",
                        "password": "",
                        "publickey": "",
                        "privatekey": "",
                        "command": "notify.sh"
                    },
                    "opcommand_hst": [
                        {
                            "hostid": "0"
                        }
                    ],
                    "opcommand_grp": []
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Action filter
- Action operation

Source

CAction::get() in *frontends/php/include/classes/api/services/CAction.php*.

action.update

说明

`object action.update(object/array actions)`

此方法允许更新现有的动作。

参数

`(object/array)` 要更新的动作属性。

必须为每个动作定义 `actionid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

除此之外standard action properties, 该方法接受以下参数。

参数类	说明	
filter	object	动作筛选器对象以替换当前筛选器。
operations	array	动作操作替换现有操作。
recovery_operations	array	动作恢复操作以替换现有恢复操作。
acknowledge_operations	array	动作确认操作以替换现有的确认操作。

返回值

`(object)` 返回一个对象，该对象在 `actionids` 属性下包含要更新动作的 ID。

范例

禁用动作

禁用动作，也就是说，将其状态设置为 1。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.update",
    "params": {
        "actionid": "2",
        "status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "2"
        ]
    },
    "id": 1
}
```

参见

- Action filter
- Action operation

来源

CAction::update() in *frontends/php/include/classes/api/services/CAction.php*.

Description

`object action.update(object/array actions)`

This method allows to update existing actions.

Parameters

`(object/array)` Action properties to be updated.

The `actionid` property must be defined for each action, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard action properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	Action filter object to replace the current filter.
operations	array	Action operations to replace existing operations.
recovery_operations	array	Action recovery operations to replace existing recovery operations.
acknowledge_operations	array	Action acknowledge operations to replace existing acknowledge operations.

Return values

`(object)` Returns an object containing the IDs of the updated actions under the `actionids` property.

Examples

Disable action

Disable action, that is, set its status to "1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.update",
    "params": {
        "actionid": "2",
        "status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Action filter
- Action operation

Source

CAction::update() in *frontends/php/include/classes/api/services/CAction.php*.

Alert-告警

这个对象用于告警模块。

对象引用:

- Alert

相关方法::

- alert.get - 获取告警

This class is designed to work with alerts.

Object references:

- Alert

Available methods:

- alert.get - retrieve alerts

> Alert object

以下是 `alert` API 的使用方法

告警

> **Note:**
> Alerts 是由 Zabbix server 创建，无法通过 API 修改。

alert 对象包含有关某些 action 操作是否已成功执行的信息，它具有以下特性。

特性类	描述	
alertid	string	Alert ID 值。
actionid	string	Alert 生成的 Action ID.
alerttype	integer	Alert 类型。 可能的值： 0 - 信息; 1 - 远程命令。
clock	timestamp	Alert 生成的时间。

特性类	描述	
error	string	Alert 发送信息或者执行一个命令产生的报错信息。
esc_step	integer	生成 Alert 后 Action 的处理步骤。
eventid	string	触发 Action 的事件 ID。
mediatypeid	string	用于发送消息的报警媒介类型的 ID。

特性类	描述	
message	text	消息文本。用于消息告警。
retries	integer	Zabbix尝试发送消息的次数。
sendto	string	地址，用户名或接收者的其他标识符。用于消息告警。

特性类	描述	
status	integer	显示 action 操作是否已执行成功的状态。 消息告警的可能值：0 - 消息未发送。1 - 消息已发送。2 - 经多次重试后失败。3 - action 管理器尚未处理新警报。 命令告警的

特性类	描述	
subject	string	消息主题。用于消息告警。
userid	string	邮件发送到的用户的ID。
p_eventid	string	生成告警的异常事件ID。
acknowledgeid	string	生成告警的确认ID。

The following objects are directly related to the `alert` API.

Alert

Note:
Alerts are created by the Zabbix server and cannot be modified via the API.

The alert object contains information about whether certain action operations have been executed successfully. It has the following properties.

Property	Type	Description
alertid	string	ID of the alert.
actionid	string	ID of the action that generated the alert.
alerttype	integer	Alert type. Possible values: 0 - message; 1 - remote command.
clock	timestamp	Time when the alert was generated.
error	string	Error text if there are problems sending a message or running a command.

Property	Type	Description
esc_step	integer	Action escalation step during which the alert was generated.
eventid	string	ID of the event that triggered the action.
mediatypeid	string	ID of the media type that was used to send the message.
message	text	Message text. Used for message alerts.
retries	integer	Number of times Zabbix tried to send the message.
sendto	string	Address, user name or other identifier of the recipient. Used for message alerts.
status	integer	Status indicating whether the action operation has been executed successfully. Possible values for message alerts: 0 - message not sent. 1 - message sent. 2 - failed after a number of retries. 3 - new alert is not yet processed by alert manager. Possible values for command alerts: 0 - command not run. 1 - command run. 2 - tried to run the command on the Zabbix agent but it was unavailable.
subject	string	Message subject. Used for message alerts.
userid	string	ID of the user that the message was sent to.
p_eventid	string	ID of problem event, which generated the alert.
acknowledgeid	string	ID of acknowledgement, which generated the alert.

alert.get

描述

整数/数组 `alert.get(object parameters)`

该方式允许根据给定的参数检索警报.

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数.

参数类	描述	
alertids	string/array	只返回给定ID的alerts。

参数类	描述	
actionids	string/array	只返回给定 actions 生成的 alerts。
eventids	string/array	只返回给定事件生成的 alerts。
groupids	string/array	只返回来自指定主机组的对象生成的 alerts。
hostids	string/array	只返回来自指定主机的对象生成的 alerts。

参数类	描述	
mediatypeids	string/array	只返回用于指定报警媒介类型的消息警报。
objectids	string/array	只返回指定对象生成的alerts。
userids	string/array	只返回发送给指定用户的消息警报。

参数类	描述	
eventobject	integer	仅返回与给定类型的对象相关的事件生成的警报。 参考event "object" property获取受支持的对象类型列表。 默认值: 0 - trigger.

参数类	描述	
eventsource	integer	仅返回由给定类型的事件生成的警报。 参考event "source" property获取受支持的对象类型列表。 默认值: 0 - trigger events.
time_from	timestamp	仅返回在给定时间后生成的警报。

参数类	描述	
time_till	timestamp	仅返回在给定时间之前生成的警报。
selectHosts	query	在 `hosts` 属性中返回触发 action 操作的主机。
selectMediatypes	query	Return the media type that was used for the message alert as an array in the `mediatypes` property.

参数类	描述	
selectUsers	query	Return the user that the message was addressed to as an array in the `users` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `alertid`, `clock`, `eventid` and `status`.

参数类	描述	
countOutput	boolean	These parameters being common for all get methods are described in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

范例

通过动作 ID 检索警报

Retrieve all alerts generated by action "3".

请求:

```
{
    "jsonrpc": "2.0",
    "method": "alert.get",
    "params": {
        "output": "extend",
        "actionids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

返回值:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "alertid": "1",
            "actionid": "3",
            "eventid": "21243",
            "userid": "1",
            "clock": "1362128008",
            "mediatypeid": "1",
            "sendto": "support@company.com",
            "subject": "PROBLEM: Zabbix agent on Linux server is unreachable for 5 minutes: ",
            "message": "Trigger: Zabbix agent on Linux server is unreachable for 5 minutes: \nTrigger stat
            "status": "0",
            "retries": "3",
            "error": "",
            "esc_step": "1",
            "alerttype": "0",
            "p_eventid": "0",
            "acknowledgeid": "0"
        }
    ],
    "id": 1
}
```


参见

- Host
- Media type
- User

Source

CAlert::get() in *frontends/php/include/classes/api/services/CAlert.php*.

Description

`integer/array alert.get(object parameters)`

The method allows to retrieve alerts according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
alertids	string/array	Return only alerts with the given IDs.
actionids	string/array	Return only alerts generated by the given actions.
eventids	string/array	Return only alerts generated by the given events.
groupids	string/array	Return only alerts generated by objects from the given host groups.
hostids	string/array	Return only alerts generated by objects from the given hosts.
mediatypeids	string/array	Return only message alerts that used the given media types.
objectids	string/array	Return only alerts generated by the given objects
userids	string/array	Return only message alerts that were sent to the given users.
eventobject	integer	Return only alerts generated by events related to objects of the given type. Refer to the event "object" property for a list of supported object types. Default: 0 - trigger.

Parameter	Type	Description
eventsource	integer	Return only alerts generated by events of the given type. Refer to the event "source" property for a list of supported event types. Default: 0 - trigger events.
time_from	timestamp	Return only alerts that have been generated after the given time.
time_till	timestamp	Return only alerts that have been generated before the given time.
selectHosts	query	Return the hosts that triggered the action operation in the `hosts` property.
selectMediatypes	query	Return the media type that was used for the message alert as an array in the `mediatypes` property.
selectUsers	query	Return the user that the message was addressed to as an array in the `users` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `alertid`, `clock`, `eventid` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve alerts by action ID

Retrieve all alerts generated by action "3".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "alert.get",
    "params": {
        "output": "extend",
        "actionids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": [
        {
            "alertid": "1",
            "actionid": "3",
            "eventid": "21243",
            "userid": "1",
            "clock": "1362128008",
            "mediatypeid": "1",
            "sendto": "support@company.com",
            "subject": "PROBLEM: Zabbix agent on Linux server is unreachable for 5 minutes: ",
            "message": "Trigger: Zabbix agent on Linux server is unreachable for 5 minutes: \nTrigger stat
            "status": "0",
            "retries": "3",
            "error": "",
            "esc_step": "1",
            "alerttype": "0",
            "p_eventid": "0",
            "acknowledgeid": "0"
        }
    ],
    "id": 1
}
```


See also

- Host
- Media type
- User

Source

CAlert::get() in *frontends/php/include/classes/api/services/CAlert.php*.

API 信息

这个类用于检索 API 相关信息

相关方法:

- apiinfo.version - 获取 Zabbix API 版本

apiinfo.version

说明

`string apiinfo.version(array)`

该方法用于获取 Zabbix API 版本。

参数

> **Attention:**
> 此方法可用于未经身份验证的用户，必须在发送 JSON-RPC 请求中不加 `auth` 参数的情况下调用。

(`array`) 该方法接受一个空的数组。

返回值

(`string`) 返回 Zabbix API 的版本。

> **Note:**
> 从 Zabbix 2.0.4 版本开始，API 的版本与 Zabbix 的版本相匹配。

范例

获取 API 版本

获取 Zabbix API 版本。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "apiinfo.version",
    "params": [],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "4.0.0",
    "id": 1
}
```

来源

CAPIInfo::version() in *frontends/php/include/classes/api/services/CAPIInfo.php*.

Description

`string apiinfo.version(array)`

This method allows to retrieve the version of the Zabbix API.

Parameters

Attention:
This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

(`array`) The method accepts an empty array.

Return values

(`string`) Returns the version of the Zabbix API.

Note:
Starting from Zabbix 2.0.4 the version of the API matches the version of Zabbix.

Examples

Retrieving the version of the API

Retrieve the version of the Zabbix API.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "apiinfo.version",
    "params": [],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "4.0.0",
    "id": 1
}
```

Source

CAPIInfo::version() in *frontends/php/include/classes/api/services/CAPIInfo.php*.

Application 应用集

这个类用于管理应用集。

对象引用:

- Application(应用集))

相关方法：

- application.create - 创建新的应用集
- application.delete - 删除应用集
- application.get - 检索应用集
- application.massadd - 更新应用集
- application.update - 添加监控项到应用集

This class is designed to work with applications.

Object references:

- Application

Available methods:

- application.create - creating new applications
- application.delete - deleting applications
- application.get - retrieving application
- application.massadd - updating application
- application.update - adding items to applications

> Application object

以下是 `application` API 的使用方法。

应用集

应用集对象包含以下属性。

属性类	说明	
applicationid	string	*(readonly)* 应用集的ID。
hostid (required)	string	应用集所属主机的ID。 不能进行更新。

属性类	说明	
name (required)	string	应用集名称。
flags	integer	*(readonly)* 应用集的来源。 可能的值为：0 - 普通应用集；4 - 自动发现的应用集。
templateids	array	*(readonly)* 上级模板应用集的ID。

The following objects are directly related to the `application` API.

Application

The application object has the following properties.

Property	Type	Description
applicationid	string	*(readonly)* ID of the application.
hostid (required)	string	ID of the host that the application belongs to. Cannot be updated.
name (required)	string	Name of the application

Property	Type	Description
flags	integer	*(readonly)* Origin of the application. Possible values: 0 - a plain application; 4 - a discovered application.
templateids	array	*(readonly)* IDs of the parent template applications.

application.create

说明

`object application.create(object/array applications)`

此方法允许创建新的应用集。

参数

`(object/array)` 需要去创建的应用集。

此方法接受创建的应用集带有标准应用集属性.

返回值

返回一个包含”applicationID” 属性的应用程序 ID 的对象。返回的 ID 的顺序与传递的应用程序的顺序相匹配

范例

创建一个应用集

创建一个应用集来存储 SNMP 监控项。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "application.create",
    "params": {
        "name": "SNMP Items",
        "hostid": "10050"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "356"
        ]
    },
    "id": 1
}
```

来源

CApplication::create() in *frontends/php/include/classes/api/services/CApplication.php*.

Description

`object application.create(object/array applications)`

This method allows to create new applications.

Parameters

`(object/array)` Applications to create.

The method accepts applications with the standard application properties.

Return values

(object) Returns an object containing the IDs of the created applications under the applicationids property. The order of the returned IDs matches the order of the passed applications.

Examples

Creating an application

Create an application to store SNMP items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.create",
    "params": {
        "name": "SNMP Items",
        "hostid": "10050"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "356"
        ]
    },
    "id": 1
}
```

Source

CApplication::create() in *frontends/php/include/classes/api/services/CApplication.php*.

application.delete

说明

`object application.delete(array applicationIds)`

此方法用于删除应用集。

参数

(array) 需要去删除的应用集 ID

返回值

(object) 返回一个"applicationid" 属性下的要删除的应用程序 ID 的对象。

范例

删除多个应用集

删除两个应用集

请求:

```
{
    "jsonrpc": "2.0",
    "method": "application.delete",
    "params": [
        "356",
        "358"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "356",
            "358"
        ]
    },
    "id": 1
}
```

来源

CApplication::delete() in *frontends/php/include/classes/api/services/CApplication.php*.

Description

`object application.delete(array applicationIds)`

This method allows to delete applications.

Parameters

`(array)` IDs of the applications to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted applications under the `applicationids` property.

Examples

Deleting multiple applications

Delete two applications.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.delete",
    "params": [
        "356",
        "358"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "356",
            "358"
        ]
    },
    "id": 1
}
```

Source

CApplication::delete() in *frontends/php/include/classes/api/services/CApplication.php*.

application.get

说明

`integer/array application.get(object parameters)`

该方法用于根据规定的参数获取应用集。

参数

`(object)` 定义所需输出的参数。

该方法提供以下参数。

Parameter	Type	Description
applicationids	string/array	只返回指定 ID 的应用集。
groupids	string/array	只返回指定主机组所属主机的应用集。
hostids	string/array	只返回指定主机所属的应用集。
inherited	boolean	如果设定为 `true`，只返回继承该模板的应用集。
itemids	string/array	只返回包含特定监控项的应用集。
templated	boolean	如果设定为 `true`，只返回属于该模板的应用集。
templateids	string/array	只返回指定模板的应用集。
selectHost	query	返回值中会包括应用集所属的主机名属性。
selectItems	query	返回值中会应用集包含的监控项属性。
selectDiscoveryRule	query	返回值中会包括应用集的底层自动发现规则属性。
selectApplicationDiscovery	query	返回值中会包括应用集发现的对象属性。
sortfield	string/array	使用规定的属性将结果分类。 可能的值：`applicationid` 和 `name`。
countOutput	boolean	在reference commentary 中详细描述了所有”get” 方法的相关参数.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回两者其中之一:

- an array of objects;
- 如果已经使用了 “countOutput” 参数，则检索对象的计数.

范例

从主机中检索应用集

从主机中根据名称排序检索所有的应用集。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "application.get",
    "params": {
        "output": "extend",
        "hostids": "10001",
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "applicationid": "13",
            "hostid": "10001",
            "name": "CPU",
            "templateids": []
        },
        {
            "applicationid": "5",
            "hostid": "10001",
            "name": "Filesystems",
            "templateids": []
        },
        {
            "applicationid": "21",
            "hostid": "10001",
            "name": "General",
            "templateids": []
        },
        {
            "applicationid": "15",
            "hostid": "10001",
            "name": "Memory",
            "templateids": []
        },
    ],
    "id": 1
}
```


See also

- Host
- Item

Source

CApplication::get() in *frontends/php/include/classes/api/services/CApplication.php*.

Description

`integer/array application.get(object parameters)`

The method allows to retrieve applications according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
applicationids	string/array	Return only applications with the given IDs.
groupids	string/array	Return only applications that belong to hosts from the given host groups.
hostids	string/array	Return only applications that belong to the given hosts.
inherited	boolean	If set to `true` return only applications inherited from a template.
itemids	string/array	Return only applications that contain the given items.
templated	boolean	If set to `true` return only applications that belong to templates.
templateids	string/array	Return only applications that belong to the given templates.

Parameter	Type	Description
selectHost	query	Return the host that the application belongs to in the `host` property.
selectItems	query	Return the items contained in the application in the `items` property.
selectDiscoveryRule	query	Return the LLD rule that created the application in the `discoveryRule` property.
selectApplicationDiscovery	query	Return the application discovery object in the `applicationDiscovery` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `applicationid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving applications from a host

Retrieve all applications from a host sorted by name.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.get",
    "params": {
        "output": "extend",
        "hostids": "10001",
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "applicationid": "13",
            "hostid": "10001",
            "name": "CPU",
            "templateids": []
        },
        {
```

```
            "applicationid": "5",
            "hostid": "10001",
            "name": "Filesystems",
            "templateids": []
        },
        {
            "applicationid": "21",
            "hostid": "10001",
            "name": "General",
            "templateids": []
        },
        {
            "applicationid": "15",
            "hostid": "10001",
            "name": "Memory",
            "templateids": []
        },
    ],
    "id": 1
}
```


See also

- Host
- Item

Source

CApplication::get() in *frontends/php/include/classes/api/services/CApplication.php*.

application.massadd

说明

`object application.massadd(object parameters)`

此方法用于同时添加多个监控项到指定的应用集。

参数

(object) 参数包含更新应用集和加入应用集监控项的 ID。

该方法接受以下参数。

参数类	说明	
applications (required)	array/object	需要更新的应用集。 应用集必须已定义好 applicationei 属性。
items	array/object	监控项加入到指定的应用集。 监控项必须已定义好 itemid 属性。

返回值

（object）返回一个其中在 applicationid 属性下已更新应用集的 ID 的对象。

范例

添加监控项到多个应用集。

添加指定的监控项到两个应用集。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "application.massadd",
    "params": {
        "applications": [
```

```
            {
                "applicationid": "247"
            },
            {
                "applicationid": "246"
            }
        ],
        "items": [
            {
                "itemid": "22800"
            },
            {
                "itemid": "22801"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "247",
            "246"
        ]
    },
    "id": 1
}
```

参见

- Item

来源

CApplication::massAdd() in *frontends/php/include/classes/api/services/CApplication.php*.

Description

`object application.massadd(object parameters)`

This method allows to simultaneously add multiple items to the given applications.

Parameters

`(object)` Parameters containing the IDs of the applications to update and the items to add to the applications.

The method accepts the following parameters.

Parameter	Type	Description
applications (required)	array/object	Applications to be updated. The applications must have the `applicationid` property defined.
items	array/object	Items to add to the given applications. The items must have the `itemid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated applications under the `applicationids` property.

Examples

Adding items to multiple applications

Add the given items to two applications.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.massadd",
    "params": {
        "applications": [
            {
                "applicationid": "247"
            },
            {
                "applicationid": "246"
            }
        ],
        "items": [
            {
                "itemid": "22800"
            },
            {
                "itemid": "22801"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "247",
            "246"
        ]
    },
    "id": 1
}
```

See also

- Item

Source

CApplication::massAdd() in *frontends/php/include/classes/api/services/CApplication.php*.

application.update

说明

`object application.update(object/array applications)`

此方法用于更新目前的应用集。

Parameters

(object/array) 需要被更新的应用集属性。

Applicationid 属性必须在每个应用集中已定义，其他所有属性为可选项。只有传递过去的属性会被更新，其他所有属性仍然保持不变。

返回值

(object) 返回一个 applicationids 属性下已更新应用集的 ID 的对象。

范例

更新应用集的名称。

更新应用集的名称为 “Processes and performance”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "application.update",
    "params": {
        "applicationid": "13",
        "name": "Processes and performance"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "13"
        ]
    },
    "id": 1
}
```

来源

CApplication::update() in *frontends/php/include/classes/api/services/CApplication.php*.

Description

`object application.update(object/array applications)`

This method allows to update existing applications.

Parameters

`(object/array)` Application properties to be updated.

The `applicationid` property must be defined for each application, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated applications under the `applicationids` property.

Examples

Changing the name of an application

Change the name of the application to ”Processes and performance”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.update",
    "params": {
        "applicationid": "13",
        "name": "Processes and performance"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "applicationids": [
            "13"
        ]
    },
    "id": 1
}
```

Source

CApplication::update() in *frontends/php/include/classes/api/services/CApplication.php*.

Configuration 配置

这个类用于导入和导出 Zabbix 的配置数据。

相关方法:

- configuration.export - 导出配置
- configuration.import - 导入配置

This class is designed to export and import Zabbix configuration data.

Available methods:

- configuration.export - exporting the configuration
- configuration.import - importing the configuration

configuration.export

说明

`string configuration.export(object parameters)`

此方法允许将配置数据导出并序列化为字符串。

参数

`(object)` 参数定义了导出的对象以及使用的格式。

参数类	说明	
format (必须)	string	导出数据的格式。 可能的值为: `json` - JSON; `xml` - XML.

参数类	说明	
options (必须)	object	导出的对象。 options 对象有以下参数 groups - (array) 主机组 ID 的导出; hosts - (array) 主机 ID 的导出; images - (array) 图表 ID 的导出; maps - (array) 拓扑图 ID 的导出. screens - (array) 屏幕 ID 的导出; templates - (array) 模板

参数类	说明

返回值

(string) 返回一个包含请求配置数据的序列化字符串

范例

导出一个主机

导出一个 XML 字符串的主机配置。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "hosts": [
                "10161"
            ]
        },
        "format": "xml"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>4.0</version><date>2018
    "id": 1
}
```

来源

CConfiguration::export() in *frontends/php/include/classes/api/services/CConfiguration.php*.

Description

`string configuration.export(object parameters)`

This method allows to export configuration data as a serialized string.

Parameters

`(object)` Parameters defining the objects to be exported and the format to use.

Parameter	Type	Description
format (required)	string	Format in which the data must be exported. Possible values: `json` - JSON; `xml` - XML.
options (required)	object	Objects to be exported. The `options` object has the following parameters: `groups` - `(array)` IDs of host groups to export; `hosts` - `(array)` IDs of hosts to export; `images` - `(array)` IDs of images to export; `maps` - `(array)` IDs of maps to export. `screens` - `(array)` IDs of screens to export; `templates` - `(array)` IDs of templates to export; `valueMaps` - `(array)` IDs of value maps to export;

Return values

(string) Returns a serialized string containing the requested configuration data.

Examples

Exporting a host

Export the configuration of a host as an XML string.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "hosts": [
                "10161"
            ]
        },
        "format": "xml"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>4.0</version><date>2018
    "id": 1
}
```

Source

CConfiguration::export() in *frontends/php/include/classes/api/services/CConfiguration.php*.

configuration.import

说明

boolean configuration.import(object parameters)

此方法允许使用序列化字符串导入配置数据。

参数

(object) 参数包含导入的数据以及如何处理数据的规则。

参数类	说明	
format (required)	string	关于如何导入应用集的规则。 可能的值: json - JSON; xml - XML.

参数类	说明	
source (required)	string	包含配置数据的序列化字符串。
rules (required)	object	如何导入新的和现有的对象的规则。 `rules` 参数在下表详细描述。

Note:
如果没有规则，配置将不被更新。

`rules` 对象提供以下参数。

参数类	说明
applications	object 如何导入应用程序的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的应用集将会被创建；默认：false； deleteMissing - (boolean) 如果设置为 true，不在导入数据中的应用集将会从数据库

参数类	说明	
discoveryRules	object	关于如何导入底层自动发现规则(LLD)的规则。 支持的参数： createMissing - (boolean) 如果设置为true，新的底层自动发现规则(LLD)将会被创建；默认：false； updateExistin - (boolean) 如果设置为true，已有的底

参数类	说明
graphs	object 关于如何导入图表的规则。 支持的参数： createMissing - (boolean) 如果设置为true，新的图表将会被创建；默认：false； updateExisting - (boolean) 如何设置为true，已有的图表将会被更新；默认：false； deleteMissing - (boolean) 如果

参数类	说明	
groups	object	关于如何导入主机组的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的主机组将会被创建；默认：false；

参数类	说明	
hosts	object	关于如何导入主机的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的主机将会被创建；默认：false； updateExistin - (boolean) 如果设置为 true，已有的主机将会被更新；默认：false。

参数类	说明	
images	object	关于如何导入图片的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的图片将会被创建；默认：false; updateExistin - (boolean) 如果设置为 true，已有的图片将会被创建；默认：false。

参数类	说明	
items	object	关于如何导入监控项的规则。 支持的参数： createMissing - (boolean) 如果设置为true，新的监控项将会被创建；默认：false; updateExisting - (boolean) 如果设置为true，已有的监控项将会被更新；默认：false; deleteMissing -

参数类	说明	
maps	object	关于如何导入拓扑图的规则 支持的参数： createMissing - (boolean) 如果设置为true，新的拓扑图将会被创建；默认：false; updateExistin - (boolean) 如果设置为true，已有的拓扑图将会被更新；默认：false。

参数类	说明	
screens	object	关于如何导入聚合图形的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的聚合图形将会被创建；默认：false; updateExistin - (boolean) 如果设置为 true，已有的聚合图形将会被更新；默认：

参数类	说明	
templateLinkage	object	关于如何导入模板链接的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的模板和主机之间的链接将会被创建；默认：false。

参数类	说明	
templates	object	关于如何导入模板的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的模板将会被创建；默认：false; updateExistin - (boolean) 如果设置为 true，已有的模板将会被更新；默认：false。

参数类	说明	
templateScreens	object	关于如何导入聚合图形模板的规则。 支持的参数： createMissing - (boolean) 如果设置为 true，新的聚合图形模板将会被创建；默认：false; updateExistin - (boolean) 如果设置为 true，已有的聚合图形模板将

参数类	说明
triggers	object 关于如何导入触发器的规则。 支持的参数： createMissing - (boolean) 如果设置为true，新的触发器将会被创建；默认：false; updateExisting - (boolean) 如果设置为true，已有的触发器将会被更新；默认：false; deleteMissing -

参数类	说明	
valueMaps	object	关于如何导入值映射的规则。 支持的参数： createMissing - (boolean) 如果设置为true，新的值映射将会被创建；默认：false; updateExisting - (boolean) 如果设置为true，已有的值映射将会被更新默认：false。

参数类	说明

返回值

(boolean) 如果导入成功则返回 true。

范例

导入主机和监控项

导入的主机和监控项包含在 XML 字符串中。如果在 XML 中遗漏了任何监控项，这些监控项将会在数据库中被删除，其他的则不改变。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "applications": {
                "createMissing": true,
                "deleteMissing": false
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            },
            "hosts": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><zabbix_export><version>4.0</version><date>20
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

来源

CConfiguration::import() in *frontends/php/include/classes/api/services/CConfiguration.php*.

Description

`boolean configuration.import(object parameters)`

This method allows to import configuration data from a serialized string.

Parameters

(object) Parameters containing the data to import and rules how the data should be handled.

Parameter	Type	Description
format (required)	string	Format of the serialized string. Possible values: `json` - JSON; `xml` - XML.
source (required)	string	Serialized string containing the configuration data.
rules (required)	object	Rules on how new and existing objects should be imported. The `rules` parameter is described in detail in the table below.

Note:
If no rules are given, the configuration will not be updated.

The `rules` object supports the following parameters.

Parameter	Type	Description
applications	object	Rules on how to import applications. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new applications will be created; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, applications not present in the imported data will be deleted from the database; default: `false`.
discoveryRules	object	Rules on how to import LLD rules. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new LLD rules will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing LLD rules will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, LLD rules not present in the imported data will be deleted from the database; default: `false`.
graphs	object	Rules on how to import graphs. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new graphs will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing graphs will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, graphs not present in the imported data will be deleted from the database; default: `false`.
groups	object	Rules on how to import host groups. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new host groups will be created; default: `false`.
hosts	object	Rules on how to import hosts. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new hosts will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing hosts will be updated; default: `false`.

Parameter	Type	Description
images	object	Rules on how to import images. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new images will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing images will be updated; default: `false`.
items	object	Rules on how to import items. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new items will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing items will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, items not present in the imported data will be deleted from the database; default: `false`.
maps	object	Rules on how to import maps. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new maps will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing maps will be updated; default: `false`.
screens	object	Rules on how to import screens. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new screens will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing screens will be updated; default: `false`.
templateLinkage	object	Rules on how to import template links. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new links between templates and host will be created; default: `false`.
templates	object	Rules on how to import templates. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new templates will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing templates will be updated; default: `false`.
templateScreens	object	Rules on how to import template screens. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new template screens will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing template screens will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, template screens not present in the imported data will be deleted from the database; default: `false`.

Parameter	Type	Description
triggers	object	Rules on how to import triggers. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new triggers will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing triggers will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, triggers not present in the imported data will be deleted from the database; default: `false`.
valueMaps	object	Rules on how to import value maps. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new value maps will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing value maps will be updated; default: `false`.

Return values

`(boolean)` Returns `true` if importing has been successful.

Examples

Importing hosts and items

Import the host and items contained in the XML string. If any items in XML are missing, they will be deleted from the database, and everything else will be left unchanged.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "applications": {
                "createMissing": true,
                "deleteMissing": false
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            },
            "hosts": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><zabbix_export><version>4.0</version><date>20
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CConfiguration::import() in *frontends/php/include/classes/api/services/CConfiguration.php*.

Discovery rule 发现规则

发现规则 This class is designed to work with network discovery rules. 该类用于处理网络发现规则。

This API is meant to work with network discovery rules. For the low-level discovery rules see the LLD rule API. ::: <note tip> 此 API 旨在处理网络发现规则。对于低级别发现规则，请参考低级别发现规则 API.

Object references:
对象引用:

- Discovery rule
- 发现规则

Available methods:
可用方法:

- drule.create - create new discovery rules
- drule.delete - delete discovery rules
- drule.get - retrieve discovery rules
- drule.update - update discovery rules
- drule.create - 创建发现规则
- drule.delete - 删除发现规则
- drule.get - 获取发现规则
- drule.update - 更新发现规则

> Discovery rule object

> 发现规则对象

The following objects are directly related to the `drule` API. 以下是和 `drule` API 相关的对象。

Discovery rule

发现规则

The discovery rule object defines a network discovery rule. It has the following properties. 发现规则对象用于定义网络发现规则. 它有如下属性:

Property	Type	Description
druleid	string	*(readonly)* ID of the discovery rule.
iprange (required)	string	One or several IP ranges to check separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.
name (required)	string	Name of the discovery rule.
delay	string	Execution interval of the discovery rule. Accepts seconds, time unit with suffix and user macro. Default: 1h.
nextcheck	timestamp	*(readonly)* Time when the discovery rule will be executed next.
proxy_hostid	string	ID of the proxy used for discovery.

Property	Type	Description
status	integer	Whether the discovery rule is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

属性类	描述	
druleid	string	*(只读)* 发现规则的ID
iprange (必选)	string	一个或多个要检查的IP范围，用逗号进行分隔。更多有关IP范围的支持格式的信息，请参考网络发现规则配置。

属性类	描述	
name (必选)	string	发现规则名称
delay	string	发现规则的执行间隔。支持秒、用户宏以及带后缀的时间单位. 默认: 1h.
nextcheck	timestamp	*(只读)* 发现规则下一次执行的时间。
proxy_hostid	string	用于发现的proxy的ID.

属性类	描述	
status	integer	发现规则是否启用. 可选值: 0 - (默认)启用; 1 - 禁用.

drule.create

Description

描述

`object drule.create(object/array discoveryRules)`对象 `drule.create(object/array discoveryRules)`

This method allows to create new discovery rules. 该方法用于创建新的发现规则。

Parameters

参数

`(object/array)`Discovery rules to create.（对象/数组）要创建的发现规则。

Additionally to the standard discovery rule properties, the method accepts the following parameters. 除了标准的发现规则属性之外，该方法还接受以下参数：

Parameter	Type	Description
dchecks (required)	array	Discovery checks to create for the discovery rule.
参数类	描述	
dchecks (必选)	array	为发现规则创建发现检查

Return values

返回值

`(object)` Returns an object containing the IDs of the created discovery rules under the `druleids` property. The order of the returned IDs matches the order of the passed discovery rules.（对象）在 `druleids` 属性下，返回一个包含已创建的发现规则的 ID 的对象。返回的 ID 的顺序与传递的发现规则的顺序相匹配。

Examples

例子

Create a discovery rule

创建发现规则

Create a discovery rule to find machines running the Zabbix agent in the local network. The rule must use a single Zabbix agent check on port 10050. 创建一个发现规则，用于发现在本地网络中运行 Zabbix Agent 的主机. 此规则必须用在在 10050 端口运行的 Zabbix agent 下【备注 zmhuang: 怀疑原文语法有错误 The rule must use in a single Zabbix agent checked on port 10050】

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.create",
    "params": {
        "name": "Zabbix agent discovery",
        "iprange": "192.168.1.1-255",
        "dchecks": [
            {
                "type": "9",
                "key_": "system.uname",
                "ports": "10050",
                "uniq": "0"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

See also

参考

- Discovery check
- 发现检查

Source

来源

CDRule::create() in *frontends/php/include/classes/api/services/CDRule.php*.

drule.delete

Description

描述

`object drule.delete(array discoveryRuleIds)` 对象 `drule.delete(array discoveryRuleIds)`

This method allows to delete discovery rules. 该方法用于删除发现规则。

Parameters

参数

(`array`) IDs of the discovery rules to delete.（数组）要删除的发现规则的 ID

Return values

返回值

(`object`) Returns an object containing the IDs of the deleted discovery rules under the `druleids` property.（对象）在 `druleids` 属性下，返回包含已删除的发现规则的 ID 的对象.

Examples

例子

Delete multiple discovery rules

删除多个发现规则

Delete two discovery rules. 删除两个发现规则

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.delete",
    "params": [
        "4",
        "6"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "4",
            "6"
        ]
    },
    "id": 1
}
```

Source

来源

CDRule::delete() in *frontends/php/include/classes/api/services/CDRule.php*.

drule.get

Description

描述

`integer/array drule.get(object parameters)` 整数/数组 `drule.get(object parameters)`

The method allows to retrieve discovery rules according to the given parameters. 该方法用于根据给定的参数获取发现规则。

Parameters

参数

`(object)` Parameters defining the desired output.（对象）定义所需输出的参数.

The method supports the following parameters. 该方法支持以下参数。

Parameter	Type	Description
dhostids	string/array	Return only discovery rules that created the given discovered hosts.
druleids	string/array	Return only discovery rules with the given IDs.
dserviceids	string/array	Return only discovery rules that created the given discovered services.
selectDChecks	query	Return discovery checks used by the discovery rule in the `dchecks` property. Supports `count`.
selectDHosts	query	Return the discovered hosts that the discovery rule created in the `dhosts` property. Supports `count`.

Parameter	Type	Description
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectDChecks` - results will be sorted by `dcheckid`; `selectDHosts` - results will be sorted by `dhostsid`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `druleid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

参数类	描述	
dhostids	string/array	仅返回创建给定已发现主机的发现规则
druleids	string/array	仅返回给定ID的发现规则

参数类	描述	
dserviceids	string/array	仅返回创建给定已发现服务的发现规则
selectDChecks	query	在dchecks属性下，返回被发现规则使用的发现检查 支持count.
selectDHosts	query	在dhosts属性下，返回发现规则创建的发现主机 支持count.

参数类	描述	
limitSelects	integer	限制子选项返回的记录数 适用于以下子选项: selectDChecks - 结果按dcheckid排序; selectDHosts - 结果按dhostsid排序
sortfield	string/array	结果按给定属性排序. 可能的值: druleid和name.

参数类	描述	
countOutput	boolean	以下参数为get方法通常参数，在参考注释有详细说明
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

返回值

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

(整数/数组) 返回:

- 对象数据;
- 如果 countOutput 被使用，返回获取对象的计数.

Examples

例子

Retrieve all discovery rules

获取所有发现规则

Retrieve all configured discovery rules and the discovery checks they use. 获取所有已配置的发现规则及使用的发现检查。

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.get",
    "params": {
        "output": "extend",
        "selectDChecks": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "druleid": "2",
            "proxy_hostid": "0",
            "name": "Local network",
            "iprange": "192.168.3.1-255",
            "delay": "5s",
            "nextcheck": "1348754327",
            "status": "0",
            "dchecks": [
                {
                    "dcheckid": "7",
                    "druleid": "2",
                    "type": "3",
                    "key_": "",
                    "snmp_community": "",
                    "ports": "21",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0"
                },
                {
                    "dcheckid": "8",
                    "druleid": "2",
                    "type": "4",
                    "key_": "",
                    "snmp_community": "",
                    "ports": "80",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0"
                }
            ]
        },
        {
            "druleid": "6",
            "proxy_hostid": "0",
            "name": "Zabbix agent discovery",
            "iprange": "192.168.1.1-255",
            "delay": "1h",
            "nextcheck": "0",
            "status": "0",
            "dchecks": [
                {
                    "dcheckid": "10",
                    "druleid": "6",
                    "type": "9",
                    "key_": "system.uname",
```

```
                    "snmp_community": "",
                    "ports": "10050",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0"
                }
            ]
        }
    ],
    "id": 1
}
```


See also

参考

- Discovered host
- Discovery check
- 已发现主机
- 发现检查

Source

来源

CDRule::get() in *frontends/php/include/classes/api/services/CDRule.php*.

drule.update

Description

描述

`object drule.update(object/array discoveryRules)` 对象 `drule.update(object/array discoveryRules)`

This method allows to update existing discovery rules. 该方法用于更新已存在的发现规则。

Parameters

参数

(`object/array`) Discovery rule properties to be updated.（对象/数组）要更新的发现规则属性

The `druleid` property must be defined for each discovery rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每条发现规则定义 `druleid` 属性, 其它属性是可选的. 只有传参进去的属性才会被更新，其它属性不变.

Additionally to the standard discovery rule properties, the method accepts the following parameters. 除了标准的发现规则属性外, 该方法还接受以下参数.

Parameter	Type	Description
dchecks	array	Discovery checks to replace existing checks.

参数类	描述	
dchecks	array	替代已存在的发现检查.

Return values

返回值

(`object`) Returns an object containing the IDs of the updated discovery rules under the `druleids` property.（对象）在 `druleids` 属性下，返回包含已更新的发现规则的 ID 对象.

Examples

例子

Change the IP range of a discovery rule

更改发现规则的 IP 范围

Change the IP range of a discovery rule to ”192.168.2.1-255”. 将发现规则的 IP 范围更改为 192.168.2.1-255

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.update",
    "params": {
        "druleid": "6",
        "iprange": "192.168.2.1-255"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

See also

参考

- Discovery check
- 发现检查

Source

来源

CDRule::update() in *frontends/php/include/classes/api/services/CDRule.php*.

Event 事件

事件　This class is designed to work with events. 这个类用于配合事件使用

Object references:

- Event

对象引用:

- 事件

Available methods:

- event.get - retrieving events
- event.acknowledge - acknowledging events

可用方法:

- event.get - 获取事件

- event.acknowledge - 确认事件

> Event object

The following objects are directly related to the `event` API. 以下对象与 `event`［事件］API 直接相关

Event

事件

Note:
Events are created by the Zabbix server and cannot be modified via the API.

Note:
事件是由 Zabbix server 创建，并且不能通过 API 进行修改。

The event object has the following properties. 事件对象具有以下属性：

Property	Type	Description
eventid	string	ID of the event.
source	integer	Type of the event. Possible values: 0 - event created by a trigger; 1 - event created by a discovery rule; 2 - event created by active agent auto-registration; 3 - internal event.
object	integer	Type of object that is related to the event. Possible values for trigger events: 0 - trigger. Possible values for discovery events: 1 - discovered host; 2 - discovered service. Possible values for auto-registration events: 3 - auto-registered host. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
objectid	string	ID of the related object.
acknowledged	integer	Whether the event has been acknowledged.
clock	timestamp	Time when the event was created.
ns	integer	Nanoseconds when the event was created.
name	string	Resolved event name.

Property	Type	Description
value	integer	State of the related object. Possible values for trigger events: 0 - OK; 1 - problem. Possible values for discovery events: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost. Possible values for internal events: 0 - ”normal” state; 1 - ”unknown” or ”not supported” state. This parameter is not used for active agent auto-registration events.
severity	integer	Event current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
r_eventid	string	Recovery event ID
c_eventid	string	Problem event ID who generated OK event
correlationid	string	Correlation ID
userid	string	User ID if the event was manually closed.

属性类	描述	
eventid	string	事件的ID

属性类	描述	
source	integer	事件的类型 可能的值: 0 - 由触发器创建的事件; 1 - 由发现规则创建的事件; 2 - 活动代理自动注册的事件; 3 - 内部事件.

属性类	描述	
object	integer	与事件相关的对象类型. 触发器事件的可能值: 0 - 触发器. 发现事件的可能值: 1 - 发现主机; 2 - 发现服务. 自动注册事件的可能值: 3 - 自动注册的主机. 内部

属性类	描述	
objectid	string	相关对象的ID.
acknowledged	integer	事件是否被确认.
clock	timestamp	事件的创建时间.
ns	integer	事件的创建时间(纳秒).
name	string	已解决事件的名称.

属性类	描述	
value	integer	相关对象的状态. 触发器事件的可能值: 0 - 正常; 1 - 异常. 发现事件的可能值: 0 - 主机或服务正常; 1 - 主机或服务故障; 2 - 主机或服务已发现; 3 - 主机或服务丢失.

属性类	描述	
severity	integer	当前事件的严重等级. 可能值: 0 - 未分类; 1 - 信息; 2 - 警告; 3 - 一般严重; 4 - 严重; 5 - 灾难.
r_eventid	string	恢复事件的ID
c_eventid	string	生成OK事件的问题事件ID
correlationid	string	关联ID

属性类	描述	
userid	string	手动关闭事件的用户的ID.

event.acknowledge

Description

描述

`object event.acknowledge(object/array parameters)` 对象 `event.acknowledge(object/array parameters)`

This method allows to update events. Following update actions can be performed:

- Close event. If event is already resolved, this action will be skipped.
- Acknowledge event. If event is already acknowledged, this action will be skipped.
- Add message.
- Change event severity. If event already has same severity, this action will be skipped.

此方法用于更新事件，可以执行以下更新操作:

- 关闭事件. 如果事件已经解决，此操作将会被跳过.
- 确认事件. 如果事件已经被确认，此操作将会被跳过.
- 增加消息.
- 更改事件严重等级. 如果事件已经拥有相同的严重等级，此操作将会被跳过.

Only trigger events can be updated.
Only problem events can be updated.
Read/Write rights for trigger are required to close the event or to change event's severity.
To close event, it should be allowed in trigger. ::: <note important> 只有触发器事件可以被更新.
只有问题事件可以被更新.
关闭事件或者更改事件的严重等级需要具有对触发器的读写权限.
为了可以关闭事件，你应该在触发器中配置' 允许手动关闭'.

Parameters

参数

`(object/array)` Parameters containing the IDs of the events and update operations that should be performed.（对象/数组）包含事件 ID 和应执行的更新操作的参数.

Parameter	Type	Description
eventids (required)	string/object	IDs of the events to acknowledge.
action (required)	integer	Event update action(s). This is bitmask field, any combination of values are acceptable. Possible values: 1 - close problem; 2 - acknowledge event; 4 - add message; 8 - change severity.
message	string	Text of the message. **Required**, if action contains 'add message' flag.

Parameter	Type	Description
severity	integer	New severity for events. **Required**, if action contains ’change severity’ flag. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.

参数类	描述	
eventids (必选)	string/object	确认事件的ID.

参数类	描述	
action (必选)	integer	更新事件的操作.这是位掩码字段，可接受以下任何值的组合.可能值:1 - 关闭问题;2 - 确认事件;4 - 增加消息;8 - 更改严重等级.

参数类	描述	
message	string	消息文本. 如果操作包含'增加消息'标志，此选项必选.

参数类	描述	
severity	integer	事件的新的严重等级. 如果操作包含'更改严重等级'标志，此选项必选. 可能值: 0 - 未分类; 1 - 信息; 2 - 警告; 3 - 一般严重; 4 - 严重; 5 - 灾难.

Return values

返回值

（object）Returns an object containing the IDs of the updated events under the `eventids` property.（对象）在 `eventids` 属性下，返回一个包含被更新事件的 ID.

Examples

例子

Acknowledging an event

确认一个事件

Acknowledge a single event and leave a message. 确认一个事件并留下消息

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 6,
        "message": "Problem resolved."
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427"
        ]
    },
    "id": 1
}
```

Changing event's severity

更改事件的严重等级

Change multiple event's severity and leave a message. 更改多个事件的严重等级并留下消息.

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": ["20427", "20428"],
        "action": 12,
        "message": "Maintenance required to fix it.",
        "severity": 4
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427",
            "20428"
        ]
    },
    "id": 1
}
```

Source

来源

CEvent::acknowledge() in *frontends/php/include/classes/api/services/CEvent.php*.

event.get

Description

描述

`integer/array event.get(object parameters)` 整数/数组 `event.get(object parameters)`

The method allows to retrieve events according to the given parameters. 此方法用于根据给定参数来获取事件

Parameters

参数

(`object`) Parameters defining the desired output.（对象）定义所需输出的参数.

The method supports the following parameters. 此方法支持以下参数：

Parameter	Type	Description
eventids	string/array	Return only events with the given IDs.
groupids	string/array	Return only events created by objects that belong to the given host groups.
hostids	string/array	Return only events created by objects that belong to the given hosts.
objectids	string/array	Return only events created by the given objects.
applicationids	string/array	Return only events created by objects that belong to the given applications. Applies only if object is trigger or item.
source	integer	Return only events with the given type. Refer to the event object page for a list of supported event types. Default: 0 - trigger events.
object	integer	Return only events created by objects of the given type. Refer to the event object page for a list of supported object types. Default: 0 - trigger.
acknowledged	boolean	If set to `true` return only acknowledged events.
severities	integer/array	Return only events with given event severities. Applies only if object is trigger.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	object	Return only events with given tags. Exact match by tag and case-insensitive search by value and operator. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all events. Possible operator types: 0 - (default) Like; 1 - Equal.
eventid_from	string	Return only events with IDs greater or equal to the given ID.

Parameter	Type	Description
eventid_till	string	Return only events with IDs less or equal to the given ID.
time_from	timestamp	Return only events that have been created after or at the given time.
time_till	timestamp	Return only events that have been created before or at the given time.
value	integer/array	Return only events with the given values.
selectHosts	query	Return hosts containing the object that created the event in the `hosts` property. Supported only for events generated by triggers, items or LLD rules.
selectRelatedObject	query	Return the object that created the event in the `relatedObject` property. The type of object returned depends on the event type.
select_alerts	query	Return alerts generated by the event in the `alerts` property. Alerts are sorted in reverse chronological order.
select_acknowledges	query	Return event's updates in the `acknowledges` property. Event updates are sorted in reverse chronological order. The event update object has the following properties: `acknowledgeid` - `(string)` acknowledgement's ID; `userid` - `(string)` ID of the user that updated the event; `eventid` - `(string)` ID of the updated event; `clock` - `(timestamp)` time when the event was updated; `message` - `(string)` text of the message; `action` - `(integer)` update action that was performed see event.acknowledge; `old_severity` - `(integer)` event severity before this update action; `new_severity` - `(integer)` event severity after this update action; `alias` - `(string)` alias of the user that updated the event; `name` - `(string)` name of the user that updated the event; `surname` - `(string)` surname of the user that updated the event. Supports `count`.
selectTags	query	Return event tags in `tags` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `eventid`, `objectid` and `clock`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

参数类	描述	
eventids	string/array	仅返回具有给定ID的事件.
groupids	string/array	仅返回由属于给定主机组的对象创建的事件.
hostids	string/array	仅返回由属于给定主机的对象创建的事件。.
objectids	string/array	仅返回由给定对象创建的事件.

参数类	描述	
applicationids	string/array	仅返回属于给定应用程序的对象创建的事件。仅当对象为触发器或监控项时才适用。

参数类	描述	
source	integer	仅返回给定类型的事件. 有关支持的事件类型的列表，请参阅事件对象页面. 默认值: 0 - 触发器事件.

参数类	描述	
object	integer	仅返回由给定类型的对象创建的事件. 有关支持的对象类型的列表，请参阅事件对象页面. 默认值: 0 - 触发器.
acknowledged	boolean	如果设置为“true”，则只返回已被确认的事件.

参数类	描述	
severities	integer/array	仅返回符合给定严重程度的事件。 仅当对象为触发器时才适用.
evaltype	integer	标签搜索的规则. 可能值: 0 - (默认) 与/或; 2 - 或.

参数类	描述	
tags	object	仅返回具有给定标签的事件. 按标签进行完全匹配; 按值搜索时，不区分大小写. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. 一个空数组会返回所有事件. 可能的操作类型: 0 - (默认) 相似

参数类	描述	
eventid_from	string	仅返回ID大于或等于给定ID的事件.
eventid_till	string	仅返回ID小于或等于给定ID的事件.
time_from	timestamp	仅返回在给定时间时或之后创建的事件.

参数类	描述	
time_till	timestamp	仅返回在给定时间时或之前创建的事件.
value	integer/array	仅返回具有给定值的事件.

参数类	描述	
selectHosts	query	在主机属性下，返回包含创建该事件的对象的主机. 仅支持由触发器、监控项、低级别发现规则生成的事件.

参数类	描述	
selectRelatedObject	query	在相关对象(relatedObje属性下，返回创建该事件的对象.返回的对象类型会依赖于该事件的类型.

参数类	描述	
select_alerts	query	在告警属性下，返回由该事件生成的告警.告警是按反向时间顺序进行排序

参数类	描述	
select_acknowledges	query	在确认属性下，返回事件的更新. 事件的更新是按反向时间顺序进行排序. 事件更新对象具有以下属性: acknowledgei - (string) 确认的ID; userid - (string) 更新事件的用户的ID; eventid - (string)

参数类	描述	
selectTags	query	在标签属性下，返回事件的标签.
sortfield	string/array	根据给定属性，对结果进行排序. 可能值: `eventid`, `objectid` 以及 `clock`.
countOutput	boolean	以下参数为get方法通常参数，在参考注释有详细说明.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	

参数类	描述
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	boolean

Return values

返回值

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

(整数/数组) 返回:

- 一个数组对象;
- 如果使用了 countOutput 参数，返回获取的对象的数量.

Examples

例子

Retrieving trigger events

获取触发器事件

Retrieve the latest events from trigger "13926." 从触发器"13926" 中获取最新事件

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
        "select_acknowledges": "extend",
        "selectTags": "extend",
        "objectids": "13926",
        "sortfield": ["clock", "eventid"],
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "acknowledges": [
                {
                    "acknowledgeid": "1",
                    "userid": "1",
                    "eventid": "9695",
                    "clock": "1350640590",
                    "message": "Problem resolved.\n\r----[BULK ACKNOWLEDGE]----",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0",
                    "alias": "Admin",
                    "name": "Zabbix",
                    "surname": "Administrator"
                }
```

```
            ],
            "eventid": "9695",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970410",
            "value": "1",
            "acknowledged": "1",
            "ns": "413316245",
            "name": "MySQL is down",
            "severity": "5",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "tags": [
                {
                    "tag": "service",
                    "value": "mysqld"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        },
        {
            "acknowledges": [],
            "eventid": "9671",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970347",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Unavailable by ICMP ping",
            "severity": "4",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "tags": []
        }
    ],
    "id": 1
}
```


Retrieving events by time period

按时间段获取事件

Retrieve all events that have been created between October 9 and 10, 2012, in reverse chronological order. 在 2012-10-9 至 2012-10-10 时间段内，以逆时间顺序获取所有已被创建的事件。

Request: 请求:


```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
        "time_from": "1349797228",
        "time_till": "1350661228",
        "sortfield": ["clock", "eventid"],
```

```
        "sortorder": "desc"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "20616",
            "source": "0",
            "object": "0",
            "objectid": "14282",
            "clock": "1350477814",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": "Less than 25% free in the history cache",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0"
        },
        {
            "eventid": "20617",
            "source": "0",
            "object": "0",
            "objectid": "14283",
            "clock": "1350477814",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Zabbix trapper processes more than 75% busy",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0"
        },
        {
            "eventid": "20618",
            "source": "0",
            "object": "0",
            "objectid": "14284",
            "clock": "1350477815",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": "High ICMP ping loss",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0"
        }
    ],
    "id": 1
}
```

See also

参考

- Alert
- Item
- Host
- LLD rule
- Trigger
- 告警
- 监控项
- 主机
- 低级别发现规则
- 触发器

Source

来源

CEvent::get() in *frontends/php/include/classes/api/services/CEvent.php*.

Graph item 图表监控项

图表监控项

This class is designed to work with hosts. 这个类用于配合主机使用。

Object references:

- Graph item

对象引用:

- 图表监控项

Available methods:

- graphitem.get - retrieving graph items

可用方法:

- graphitem.get - 获取图表监控项

> Graph item object

> 图表监控项对象

The following objects are directly related to the `graphitem` API. 以下对象与 `graphitem` API 直接相关

Graph item

图表监控项

Note:
Graph items can only be modified via the `graph` API.

Note:
图表监控项只能通过 `graph` API 进行修改.

The graph item object has the following properties. 图表监控项具有以下属性:

Property	Type	Description
gitemid	string	*(readonly)* ID of the graph item.
color (required)	string	Graph item's draw color as a hexadecimal color code.

Property	Type	Description
itemid (required)	string	ID of the item.
calc_fnc	integer	Value of the item that will be displayed. Possible values: 1 - minimum value; 2 - *(default)* average value; 4 - maximum value; 7 - all values; 9 - last value, used only by pie and exploded graphs.
drawtype	integer	Draw style of the graph item. Possible values: 0 - *(default)* line; 1 - filled region; 2 - bold line; 3 - dot; 4 - dashed line; 5 - gradient line.
graphid	string	ID of the graph that the graph item belongs to.
sortorder	integer	Position of the item in the graph. Default: starts with 0 and increases by one with each entry.
type	integer	Type of graph item. Possible values: 0 - *(default)* simple; 2 - graph sum, used only by pie and exploded graphs.
yaxisside	integer	Side of the graph where the graph item’s Y scale will be drawn. Possible values: 0 - *(default)* left side; 1 - right side.

属性类	描述	
gitemid	string	*(必选)* 图表监控项的ID.
color (必选)	string	绘制图形项目的颜色，使用十六进制码表示.
itemid (必选)	string	监控项的ID.

属性类	描述	
calc_fnc	integer	监控项显示的值. 可用值: 1 - 最小值; 2 - (默认) 平均值; 4 - 最大值; 7 - 所有值; 9 - 最新的值，仅适用于饼图以及分散饼图.
drawtype	integer	用于绘制图表监控的线形. 可用值: 0 - (默认) 实线; 1 - 面积图(填满的区域); 2 - 粗实线; 3 - 点; 4 - 虚线; 5 - 梯度线.

属性类	描述	
graphid	string	图表监控项所属的图表的ID.
sortorder	integer	图表中监控项的排序. 默认从 0 开始，每增加一个加 1.
type	integer	图表监控项的类型. 可用值: 0 - *(默认)* 简单图形; 2 - 汇总图形, 仅用于饼图和分散饼图.

属性类	描述	
yaxisside	integer	Side of the graph where the graph item's Y scale will be drawn 图表监控项的 Y 轴画在图表的那一侧: 可用值: 0 - (默认) 左侧; 1 - 右侧.

graphitem.get

Description

描述

`integer/array graphitem.get(object parameters)` 整数/数组 `graphitem.get(object parameters)`

The method allows to retrieve graph items according to the given parameters. 此方法用于根据给定参数来获取图表监控项。

Parameters

参数

(`object`) Parameters defining the desired output.（对象）定义所需输出的参数。

The method supports the following parameters. 此方法支持以下参数：

Parameter	Type	Description
gitemids	string/array	Return only graph items with the given IDs.
graphids	string/array	Return only graph items that belong to the given graphs.
itemids	string/array	Return only graph items with the given item IDs.
type	integer	Return only graph items with the given type. Refer to the graph item object page for a list of supported graph item types.
selectGraphs	query	Return the graph that the item belongs to as an array in the `graphs` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `gitemid`.

Parameter	Type	Description
countOutput	boolean	These parameters being common for all get methods are described in detail in the reference commentary page.
editable	boolean	
limit	integer	
output	query	
preservekeys	boolean	
sortorder	string/array	

参数类	描述	
gitemids	string/array	仅返回给定 ID 的图表监控项
graphids	string/array	仅返回属于给定图表的图表监控项
itemids	string/array	仅返回具有给定监控项 ID 的图表监控项
type	integer	仅返回给定类型的图表监控项 有关支持的图表监控项的类型，请参考图表监控项对象.
selectGraphs	query	在 graphs（图表）属性下，以数组的形式返回监控项所属的图表
sortfield	string/array	根据给定属性对结果进行排序. 可能值: gitemid.
countOutput	boolean	以下参数为 get 方法通常参数，在参考注释有详细说明.
editable	boolean	
limit	integer	
output	query	
preservekeys	boolean	
sortorder	string/array	

Return values

返回值

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

（整数/数组）返回:

- 一个数组对象;
- 如果使用了 countOutput 参数，返回获取的对象的数量.

Examples

例子

Retrieving graph items from a graph

从图表中获取图表监控项

Retrieve all graph items used in a graph with additional information about the item and the host. 获取图表中使用的所有图表监控项以及有关项目和主机的其他信息

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphitem.get",
    "params": {
        "output": "extend",
        "graphids": "387"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "gitemid": "1242",
            "graphid": "387",
            "itemid": "22665",
            "drawtype": "1",
            "sortorder": "1",
            "color": "FF5555",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,steal]",
            "hostid": "10001",
            "flags": "0",
            "host": "Template OS Linux"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
            "color": "55FF55",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,softirq]",
            "hostid": "10001",
            "flags": "0",
            "host": "Template OS Linux"
        },
        {
            "gitemid": "1244",
            "graphid": "387",
            "itemid": "22671",
            "drawtype": "1",
            "sortorder": "3",
            "color": "009999",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,interrupt]",
            "hostid": "10001",
            "flags": "0",
            "host": "Template OS Linux"
        }
    ],
    "id": 1
}
```


See also

参考

- Graph
- 图表

Source

来源

CGraphItem::get() in *frontends/php/include/classes/api/services/CGraphItem.php*.

Graph prototype 图形原型

图形原型

This class is designed to work with graph prototypes. 这个类用于配合图形原型使用。

Object references:

- Graph prototype

对象引用:

- 图形原型

Available methods:

- graphprototype.create - creating new graph prototypes
- graphprototype.delete - deleting graph prototypes
- graphprototype.get - retrieving graph prototypes
- graphprototype.update - updating graph prototypes

可用方法:

- graphprototype.create - 新建一个图形原型
- graphprototype.delete - 删除一个图形原型
- graphprototype.get - 获取一个图形原型
- graphprototype.update - 更新一个图形原型

> Graph prototype object

The following objects are directly related to the `graphprototype` API. 以下对象与 `graphprototype` API 直接相关。

Graph prototype

图形原型

The graph prototype object has the following properties. 图形原型对象具有以下属性:

Property	Type	Description
graphid	string	*(readonly)* ID of the graph prototype.
height (required)	integer	Height of the graph prototype in pixels.
name (required)	string	Name of the graph prototype.
width (required)	integer	Width of the graph prototype in pixels.
graphtype	integer	Graph prototypes’ layout type. Possible values: 0 - *(default)* normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile. Default: 0.
percent_right	float	Right percentile. Default: 0.

Property	Type	Description
show_3d	integer	Whether to show discovered pie and exploded graphs in 3D. Possible values: 0 - *(default)* show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.
show_work_period	integer	Whether to show the working time on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	*(readonly)* ID of the parent template graph prototype.
yaxismax	float	The fixed maximum value for the Y axis.
yaxismin	float	The fixed minimum value for the Y axis.
ymax_itemid	string	ID of the item that is used as the maximum value for the Y axis.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.

属性类	描述	
graphid	string	*(只读)* 图形原型的ID.
height (必选)	integer	图形原型的高度(单位：像素).

属性类	描述	
name (必选)	string	图形原型的名称.
width (必选)	integer	图形原型的宽度(单位：像素)
graphtype	integer	图形原型布局类型. 可能值: 0 - (默认) 常规; 1 - 堆积图; 2 - 饼图; 3 - 分散饼图.
percent_left	float	左侧百分比线. 默认: 0.

属性类	描述	
percent_right	float	右侧百分比线. 默认: 0.
show_3d	integer	否使用3D形式显示被发现的饼图和分散饼图. 可能值: 0 - (默认)以2D形式展示; 1 - 以3D形式展示.

属性类	描述	
show_legend	integer	是否在被发现的图表上显示图例. 可能值: 0 - 隐藏; 1 - (默认) 显示.
show_work_period	integer	是否在发现的图表上显示工作时间. 可能值: 0 - 隐藏; 1 - (默认) 显示.

属性类	描述	
templateid	string	*(只读)* 图形原形的父模板的 ID.
yaxismax	float	Y 轴的固定最大值.
yaxismin	float	Y 轴的固定最小值.
ymax_itemid	string	用于作为 Y 轴最大值的监控项 ID.

属性类	描述	
ymax_type	integer	Y 轴最大值的计算方式. 可能值: 0 - (默认) 计算的; 1 - 固定的; 2 - 监控项.
ymin_itemid	string	用于作为 Y 轴最小值的监控项 ID.

属性类		描述
ymin_type	integer	Y 轴最小值的计算方式. 可能值: 0 - (默认) 计算的; 1 - 固定的; 2 - 监控项.

graphprototype.create

Description

描述

`object graphprototype.create(object/array graphPrototypes)` 对象 `graphprototype.create(object/array graphPrototypes)`

This method allows to create new graph prototypes. 此方法用于创建新的图形原型

Parameters

参数

`(object/array)` Graph prototypes to create.（对象/数组）将要创建的图形原型

Additionally to the standard graph prototype properties, the method accepts the following parameters. 除了标准图形原型参数外，此方法还接受以下参数：

Parameter	Type	Description
gitems (required)	array	Graph items to be created for the graph prototypes. Graph items can reference both items and item prototypes, but at least one item prototype must be present.

参数类	描述	
gitems (必选)	array	创建到图形原型中的图表监控项. 图表监控项能同时被监控项与监控项原型检索到，但必须至少有一个监控项原型。

Return values

返回值

(object) Returns an object containing the IDs of the created graph prototypes under the graphids property. The order of the returned IDs matches the order of the passed graph prototypes.（对象）在 graphids 属性下，返回一个包含已被创建的图形原型 ID 的对象。返回的 ID 的顺序与传递的图形原型的顺序相匹配。

Examples

例子

Creating a graph prototype

创建一个图形原型

Create a graph prototype with two items. 创建一个含有两个监控项的图形原型。

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.create",
    "params": {
        "name": "Disk space usage {#FSNAME}",
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

See also

参考

- Graph item
- 图表监控项

Source

来源

CGraphPrototype::create() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.delete

Description

描述

object graphprototype.delete(array graphPrototypeIds) 对象 graphprototype.delete(array graphPrototypeIds)

This method allows to delete graph prototypes. 此方法用于删除图形原型

Parameters

参数

(array) IDs of the graph prototypes to delete.（数组）需要删除的图形原型的 ID

Return values

返回值

(object) Returns an object containing the IDs of the deleted graph prototypes under the graphids property.（对象）在 graphids 属性下，返回一个包含已经删除的图形原型的 ID 的对象。

Examples

例子

Deleting multiple graph prototypes

删除多个图形原型

Delete two graph prototypes. 删除两个图形原型

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.delete",
    "params": [
        "652",
        "653"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

Source

来源

CGraphPrototype::delete() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.get

Description

描述

integer/array graphprototype.get(object parameters) 整数/数组 graphprototype.get(object parameters)

The method allows to retrieve graph prototypes according to the given parameters. 此方法用于根据给定的参数来获取图形原型

Parameters

参数

(object) Parameters defining the desired output.（对象）定义所需输出的参数.

The method supports the following parameters. 此方法支持以下参数:

Parameter	Type	Description
discoveryids	string/array	Return only graph prototypes that belong to the given discovery rules.
graphids	string/array	Return only graph prototypes with the given IDs.
groupids	string/array	Return only graph prototypes that belong to hosts in the given host groups.
hostids	string/array	Return only graph prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only graph prototypes inherited from a template.
itemids	string/array	Return only graph prototypes that contain the given item prototypes.
templated	boolean	If set to `true` return only graph prototypes that belong to templates.
templateids	string/array	Return only graph prototypes that belong to the given templates.
selectDiscoveryRule	query	Return the LLD rule that the graph prototype belongs to in the `discoveryRule` property.
selectGraphItems	query	Return the graph items used in the graph prototype in the `gitems` property.
selectGroups	query	Return the host groups that the graph prototype belongs to in the `groups` property.
selectHosts	query	Return the hosts that the graph prototype belongs to in the `hosts` property.
selectItems	query	Return the items and item prototypes used in the graph prototype in the `items` property.
selectTemplates	query	Return the templates that the graph prototype belongs to in the `templates` property.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the graph prototype belongs to; `hostid` - ID of the host that the graph prototype belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values are: `graphid`, `name` and `graphtype`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

参数类	描述	
discoveryids	string/array	仅返回属于给定自动发现规则的图形原型.
graphids	string/array	仅返回含有给定ID的图形原型.
groupids	string/array	仅返回属于给定主机组的主机的图形原型.

参数类	描述	
hostids	string/array	仅返回属于给定主机的图形原型.
inherited	boolean	如果设置此参数为 `true`，则仅返回从模板继承的图形原型.
itemids	string/array	仅返回包含给定监控项原型的图形原型.

参数类	描述	
templated	boolean	如果设置此参数为 `true`，则仅返回属于模板的图形原型.
templateids	string/array	仅返回属于给定模板的图形原型.
selectDiscoveryRule	query	在 `discoveryRul` 属性下，返回图形原型所属的低级别发现规则.

参数类	描述	
selectGraphItems	query	在gitems属性下，返回在图形原型中使用的图表监控项.
selectGroups	query	在groups属性下，返回图形原型所属的主机组.
selectHosts	query	在hosts属性下，返回图形原型所属的主机.

参数类	描述	
selectItems	query	在 `items` 属性下，返回在图形原型中使用的监控项以及监控项原型.
selectTemplates	query	在 `templates` 属性下，返回图形原型所属的模板.

参数类	描述
filter	object 仅返回精确匹配给定过滤器的结果. 接受一个数组，其中键是属性名称，值是单个值或要匹配的值的数组. 支持的额外的过滤器: host - 图型原型所属主机的

参数类	描述	
sortfield	string/array	根据给定属性对结果进行排序. 可能值: `graphid`, `name` 以及 `graphtype`.
countOutput	boolean	以下参数为 `get` 方法通常参数，在参考注释有详细说明...
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

返回值

(`integer/array`) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

(整数/数组) 返回:

- 一个数组对象;
- 如果使用了 `countOutput` 参数，返回获取的对象的数量..

Examples

例子

Retrieving graph prototypes from a LLD rule

从低级别发现规则获取图形原型

Retrieve all graph prototypes from an LLD rule. 从低级别发现规则获取所有图形原型。

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "1017",
            "parent_itemid": "27426",
            "name": "Disk space usage {#FSNAME}",
            "width": "600",
            "height": "340",
            "yaxismin": "0.0000",
            "yaxismax": "0.0000",
            "templateid": "442",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0"
        }
    ],
    "id": 1
}
```

See also

参考

- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template

- 发现规则
- 图表监控项

- 监控项
- 主机
- 主机组
- 模板

Source

来源

CGraphPrototype::get() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.update

Description

描述

`object graphprototype.update(object/array graphPrototypes)` 对象 `graphprototype.update(object/array graphPrototypes)`

This method allows to update existing graph prototypes. 此方法用于更新已存在的图形原型。

Parameters

参数

`(object/array)` Graph prototype properties to be updated.（对象/数组）需要更新的图形原型.

The `graphid` property must be defined for each graph prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. `graphid` 属性必须定义，其它属性均为可选。只有被传递的属性会被更新，其它都会保持不变。

Additionally to the standard graph prototype properties, the method accepts the following parameters. 除了标准图形原型属性外，此方法还接受以下参数：

Parameter	Type	Description
gitems	array	Graph items to replace existing graph items. If a graph item has the `gitemid` property defined it will be updated, otherwise a new graph item will be created.

参数类	描述	
gitems	array	用于替换现有图形监控项的图形监控项. 如果图表项定义了 `gitemid` 属性，它将被更新，否则将创建一个新的图形监控项.

Return values

返回值

`(object)` Returns an object containing the IDs of the updated graph prototypes under the `graphids` property. （对象）在 `graphids` 属性下，返回一个已更新的图形原型的对象的 ID。

Examples

例子

Changing the size of a graph prototype

更新图形原型大小

Change the size of a graph prototype to 1100 to 400 pixels. 将图形原型的大小从 1100 更新为 400(单位：像素)

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.update",
    "params": {
        "graphid": "439",
        "width": 1100,
```

```
        "height": 400
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

Source

来源

CGraphPrototype::update() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

Graph 图表

图表　This class is designed to work with items. 这个类用于配合监控项使用

Object references:

- Graph

参考对象:

- 图表

Available methods:

- graph.create - creating new graphs
- graph.delete - deleting graphs
- graph.get - retrieving graphs
- graph.update - updating graphs

可用方法:

- graph.create - 创建新的图表
- graph.delete - 删除图表
- graph.get - 获取图表
- graph.update - 更新图表

> Graph object

> 图表对象

The following objects are directly related to the `graph` API. 以下对象与 图表 API 直接相关。

Graph

图表

The graph object has the following properties. 图表对象具有以下属性:

Property	Type	Description
graphid	string	*(readonly)* ID of the graph.

Property	Type	Description
height (required)	integer	Height of the graph in pixels.
name (required)	string	Name of the graph
width (required)	integer	Width of the graph in pixels.
flags	integer	*(readonly)* Origin of the graph. Possible values are: 0 - *(default)* a plain graph; 4 - a discovered graph.
graphtype	integer	Graph's layout type. Possible values: 0 - *(default)* normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile. Default: 0.
percent_right	float	Right percentile. Default: 0.
show_3d	integer	Whether to show pie and exploded graphs in 3D. Possible values: 0 - *(default)* show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the graph. Possible values: 0 - hide; 1 - *(default)* show.
show_work_period	integer	Whether to show the working time on the graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	*(readonly)* ID of the parent template graph.
yaxismax	float	The fixed maximum value for the Y axis. Default: 100.
yaxismin	float	The fixed minimum value for the Y axis. Default: 0.
ymax_itemid	string	ID of the item that is used as the maximum value for the Y axis.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis.

Property	Type	Description
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.

属性类	描述	
graphid	string	*(只读)* 图表的ID.
height (必选)	integer	图表的高度(单位:像素).
name (必选)	string	图表的名称
width (必选)	integer	图表的宽度(单位:像素).

属性类	描述	
flags	integer	*(readonly)* 图表的来源. 可用值: 0 - *(默认)* 简单的图表; 4 - 发现的图表.
graphtype	integer	图表的类型. 可能值: 0 - *(默认)* 常规; 1 - 堆积图; 2 - 饼图; 3 - 分散饼图.
percent_left	float	百分比线(左). 默认: 0.

属性类	描述	
percent_right	float	百分比线(右). 默认: 0.
show_3d	integer	是否以3D形式展示饼图和分散饼图. 可用值: 0 - (默认)以2D展示; 1 - 以3D展示.
show_legend	integer	是否在图表上显示图例. 可用值: 0 - 隐藏; 1 - (默认)显示.

属性类	描述	
show_work_period	integer	是否在图表上显示工作时间. 可用值: 0 - 隐藏; 1 - *(默认)* 显示.
templateid	string	*(只读)* 父模板图表的ID.
yaxismax	float	Y轴的固定最大值. 默认: 100.
yaxismin	float	Y轴的固定最小值. 默认: 0.

属性类	描述	
ymax_itemid	string	用于作为 Y 轴最大值的监控项 ID.
ymax_type	integer	Y 轴最大值的计算方式. 可用值: 0 - (默认) 可计算的; 1 - 固定的; 2 - 监控项.
ymin_itemid	string	用于作为 Y 轴最小值的监控项 ID.

属性类	描述	
ymin_type	integer	Y 轴最小值的计算方式. 可用值: 0 - (默认) 可计算的; 1 - 固定的; 2 - 监控项.

graph.create

Description

描述

`object graph.create(object/array graphs)` 对象 `graph.create(object/array graphs)`

This method allows to create new graphs. 此方法用于创建新的图表

Parameters

参数

`(object/array)` Graphs to create.（对象/数组）要创建的图表.

Additionally to the standard graph properties, the method accepts the following parameters. 除了标准图表属性 外，此方法还接受以下参数。

Parameter	Type	Description
gitems (required)	array	Graph items to be created for the graph.

参数类	描述	
gitems (必选)	array	创建到图表中的监控项.

Return values

返回值

(object) Returns an object containing the IDs of the created graphs under the graphids property. The order of the returned IDs matches the order of the passed graphs.（对象）在 graphids 属性下，返回一个包含已创建图表 ID 的对象. 返回 ID 的顺序与传递图表的顺序想在匹配.

Examples

例子

Creating a graph

创建一个图表

Create a graph with two items. 创建一个包含两个监控项的图表.

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.create",
    "params": {
        "name": "MySQL bandwidth",
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

See also

参考

- Graph item
- 图表监控项

Source

来源

CGraph::create() in *frontends/php/include/classes/api/services/CGraph.php*.

graph.delete

Description

描述

object graph.delete(array graphIds) 对象 graph.delete(array graphIds)

This method allows to delete graphs. 此方法用于删除图表.

Parameters

参数

(array) IDs of the graphs to delete.（数组）要删除的图表的 ID.

Return values

返回值

(object) Returns an object containing the IDs of the deleted graphs under the graphids property.（对象）在 graphids 属性下，返回一个包含已删除图表的对象.

Examples

例子

Deleting multiple graphs

删除多个图表

Delete two graphs. 删除两个图表

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.delete",
    "params": [
        "652",
        "653"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

Source

来源

CGraph::delete() in *frontends/php/include/classes/api/services/CGraph.php*.

graph.get

Description

描述

integer/array graph.get(object parameters) 整数/数组 graph.get(object parameters)

The method allows to retrieve graphs according to the given parameters. 此方法用于根据给定的参数来获取图表。

Parameters

参数

(object) Parameters defining the desired output.（对象）定义所需输出的参数.

The method supports the following parameters. 此方法支持以下参数

Parameter	Type	Description
graphids	string/array	Return only graphs with the given IDs.
groupids	string/array	Return only graphs that belong to hosts in the given host groups.
templateids	string/array	Return only graph that belong to the given templates.
hostids	string/array	Return only graphs that belong to the given hosts.
itemids	string/array	Return only graphs that contain the given items.
templated	boolean	If set to `true` return only graphs that belong to templates.
inherited	boolean	If set to `true` return only graphs inherited from a template.
expandName	flag	Expand macros in the graph name.
selectGroups	query	Return the host groups that the graph belongs to in the `groups` property.
selectTemplates	query	Return the templates that the graph belongs to in the `templates` property.
selectHosts	query	Return the hosts that the graph belongs to in the `hosts` property.
selectItems	query	Return the items used in the graph in the `items` property.
selectGraphDiscovery	query	Return the graph discovery object in the `graphDiscovery` property. The graph discovery objects links the graph to a graph prototype from which it was created. It has the following properties: `graphid` - (`string`) ID of the graph; `parent_graphid` - (`string`) ID of the graph prototype from which the graph has been created.
selectGraphItems	query	Return the graph items used in the graph in the `gitems` property.
selectDiscoveryRule	query	Return the low-level discovery rule that created the graph in the `discoveryRule` property.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the graph belongs to; `hostid` - ID of the host that the graph belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values are: `graphid`, `name` and `graphtype`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

参数类	描述	
graphids	string/array	仅返回含有给定ID的图表.
groupids	string/array	仅返回属于给定主机组的主机的图表.
templateids	string/array	仅返回属于给定模板的图表.
hostids	string/array	仅返回属于给定主机的图表.

参数类	描述	
itemids	string/array	仅返回包含给定监控项的图表.
templated	boolean	如果设置为真(`true`),仅返回属于模板的图表.
inherited	boolean	如果设置为真(`true`),仅返回从模板继承的图表.
expandName	flag	在图表名称中展开宏.

参数类	描述	
selectGroups	query	在 `groups` 属性下，返回图表所属的主机组.
selectTemplates	query	在 `templates` 属性下，返回图表所属的模板.
selectHosts	query	在 `hosts` 属性下，返回图表所属的主机.
selectItems	query	在 `items` 属性下，返回图表使用的监控项.

参数类	描述	
selectGraphDiscovery	query	在graphDiscover属性下，返回图表发现对象. 图表发现对象将图表链接到创建它的图表原型. 它具有以下参数: graphid - (string) 图表的ID; parent_graph - (string) 已创建图表的图表原型的ID.

参数类	描述	
selectGraphItems	query	在gitems属性下，返回图表所使用的图表监控项.
selectDiscoveryRule	query	在discoveryRul属性下，返回创建此图表的低级别发现规则.

参数类	描述
filter	object 仅返回完全匹配给定过滤规则的结果. 接受一个数组，其中键是属性名称，值是单个值或要匹配的值数组. 支持额外的过滤器: `host` - 图表所属主机的名称; `hostid`

参数类	描述	
sortfield	string/array	按给定属性将结果排序. 可能值: `graphid`, `name` and `graphtype`.
countOutput	boolean	以下参数为get方法通常参数，在参考注释有详细说明..
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

返回值

(`integer/array`) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

(整数/级数) 返回:

- 一个数组对象;
- 如果使用了 `countOutput` 参数，返回获取的对象的数量.

Examples

例子

Retrieving graphs from hosts

从主机中获取图表

Retrieve all graphs from host ”10107” and sort them by name. 从主机”10107” 中获取所有图表，并依据名称进行排序。

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.get",
    "params": {
        "output": "extend",
        "hostids": 10107,
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "612",
            "name": "CPU jumps",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "439",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "613",
            "name": "CPU load",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "433",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "1",
            "ymax_type": "0",
            "ymin_itemid": "0",
```

```
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "614",
            "name": "CPU utilization",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "387",
            "show_work_period": "1",
            "show_triggers": "0",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "1",
            "ymax_type": "1",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "645",
            "name": "Disk space usage /",
            "width": "600",
            "height": "340",
            "yaxismin": "0.0000",
            "yaxismax": "0.0000",
            "templateid": "0",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "4"
        }
    ],
    "id": 1
}
```


See also

参考

- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template
- 发现规则
- 图表监控项
- 监控项
- 主机
- 主机组

- 模板

Source

来源

CGraph::get() in *frontends/php/include/classes/api/services/CGraph.php*.

graph.update

Description

描述

`object graph.update(object/array graphs)` 对象 `graph.update(object/array graphs)`

This method allows to update existing graphs. 此方法用于更新已存在的图表。

Parameters

参数

`(object/array)` Graph properties to be updated. (对象/数组) 要更新的图表属性.

The `graphid` property must be defined for each graph, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 每一个图表都必须定义 `graphid` 属性, 其它属性均为可选项. 只有被传递的属性会被更新, 其他属性将保持不变.

Additionally to the standard graph properties the method accepts the following parameters. 除了标准图表属性 之外，此方法还接受以下参数.

Parameter	Type	Description
gitems	array	Graph items to replace existing graph items. If a graph item has the `gitemid` property defined it will be updated, otherwise a new graph item will be created.

参数类	描述	
gitems	array	替换已存在图表监控项的图表监控项. 如果一个图表监控项的 `gitemid` 属性已经被定义，那么它将会被更新，否则将会创建一个新的图表监控项.

Return values

返回值

(`object`) Returns an object containing the IDs of the updated graphs under the `graphids` property.（对象）在 `graphids` 属性下，返回一个包含已更新图表的 ID 的对象。

Examples

例子

Setting the maximum for the Y scale

设置 Y 刻度的最大值

Set the the maximum of the Y scale to a fixed value of 100. 设置 Y 刻度的最大值为固定值 100.

Request: 请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.update",
    "params": {
        "graphid": "439",
        "ymax_type": 1,
        "yaxismax": 100
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response: 响应:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

Source

来源

CGraph::update() in *frontends/php/include/classes/api/services/CGraph.php*.

Item prototype 监控项原型

Item 原型 This class is designed to work with item prototypes. 此类旨辅助 Item 原型的使用。

Object references:
对象引用

- Item prototype

Available methods:
可用方法：

- itemprototype.create - creating new item prototypes
- itemprototype.create - 创建新监控项原型
- itemprototype.delete - deleting item prototypes
- itemprototype.delete - 删除监控项原型
- itemprototype.get - retrieving item prototypes
- itemprototype.get - 获取监控项原型
- itemprototype.update - updating item prototypes
- itemprototype.update - 更新监控项原型

> Item prototype object

The following objects are directly related to the `itemprototype` API. 如下对象与 `itemprototype` API 直接相关。

Item prototype 监控项原型

The item prototype object has the following properties. 监控项原型有如下属性。

Property	Type	Description	
itemid	string	*(readonly)* ID of the item prototype. 监控项原型的 ID。	
delay (required)	string	Update interval of the item prototype. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros and LLD macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon. 监控项原型的更新时间间隔。接受具有后缀的秒或时间单位，并且具有或不具有由灵活间隔和调度间隔组成的一个或多个自定义间隔作为串行化字符串。还接受用户宏和 LLD 宏。灵活的间隔可以写成两个由正斜杠分隔的宏。间隔用分号分隔。custom intervals	< Optional for Zabbix trapper or Dependent item.

Property	Type	Description
hostid (required)	string	ID of the host that the item prototype belongs to. 监控项原型所属的主机的 ID。 For update operations this field is *readonly*.
ruleid (required)	string	ID of the LLD rule that the item belongs to. 监控项所属的 LLD（低级别发现）的 ID For update operations this field is *readonly*.
interfaceid (required)	string	ID of the item prototype's host interface. Used only for host item prototypes. 监控项原型的主机的接口的 ID。仅用于主机监控项原型。 Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper, Dependent item, Zabbix aggregate, database monitor and calculated item prototypes.
key_ (required)	string	Item prototype key. 监控项原型的键。
name (required)	string	Name of the item prototype. 监控项原型的名称。
type (required)	integer	Type of the item prototype. 监控项原型的类型。 Possible values: 可能的值： 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - calculated; 16 - JMX agent; 17 - SNMP trap; 18 - Dependent item; 19 - HTTP agent;
url (required)	string	URL string required only for HTTP agent item prototypes. Supports LLD macros, user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}. 仅在 HTTP agent 监控项原型有要求的 URL 字符串。支持 LLD macros, user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}。
value_type (required)	integer	Type of information of the item prototype. 监控项原型信息类型。 Possible values: 可能的值： 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.

Property	Type	Description
allow_traps	integer	HTTP agent item prototype field. Allow to populate value as in trapper item type also. HTTP agent 监控项原型字段。允许像 trapper 监控项一样的填充值。 0 - *(default)* Do not allow to accept incoming data. 1 - Allow to accept incoming data.
authtype	integer	Used only by SSH agent item prototypes or HTTP agent item prototypes. 仅用于 SSH agent 监控项原型或者 HTTP agent 监控项原型。 SSH agent authentication method possible values: SSH agent 认证方法可能的值： 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM
description	string	Description of the item prototype. 监控项原型的说明。
follow_redirects	integer	HTTP agent item prototype field. Follow respose redirects while pooling data. HTTP agent 监控项原型字段。当合并数据时跟随重定向。 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent item prototype field. Object with HTTP(S) request headers, where header name is used as key and header value as value. HTTP agent 监控项原型字段。带有 HTTP(S) 的报头，名称是键，报文的值是键的值。 Example: 示例： { ”User-Agent”: ”Zabbix” }
history	string	A time unit of how long the history data should be stored. Also accepts user macro and LLD macro. 历史数据应被保存的时间。接受用户宏和 LLD 宏。 Default: 90d.
http_proxy	string	HTTP agent item prototype field. HTTP(S) proxy connection string. HTTP agent 监控项原型字段。HTTP(S) 代理连接字符串。
ipmi_sensor	string	IPMI sensor. Used only by IPMI item prototypes. IPMI 传感器，仅用于 IPMI 监控项原型。
jmx_endpoint	string	JMX agent custom connection string. JMX agent 自定义的连接字符串。 Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
logtimefmt	string	Format of the time in log entries. Used only by log item prototypes. 日志条目的时间格式。仅用于日志监控项原型。

Property	Type	Description
master_itemid	integer	Master item ID. 主监控项 ID。 递归 3 层 Recursion up to 3 dependent items and item prototypes and maximum count of dependent items and item prototypes equal to 999 are allowed. Required by Dependent items.
output_format	integer	HTTP agent item prototype field. Should response converted to JSON. HTTP agent 监控项原型字段。返回数据应被转换为 JSON 格式。 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	Additional parameters depending on the type of the item prototype: 附加参数依赖于监控项原型的类型： - executed script for SSH and Telnet item prototypes; - SQL query for database monitor item prototypes; - formula for calculated item prototypes.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes. 认证的密码。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent 监控项原型。
port	string	Port monitored by the item prototype. Used only by SNMP items prototype. 监控的监控项原型的端口。仅用于 SNMP 监控项原型。
post_type	integer	HTTP agent item prototype field. Type of post data body stored in posts property. HTTP agent 监控项原型字段。存储在 post 属性的 post 数据体的类型。 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.

赖监控项和监控项原型，最大数目的监控项和监控项原型等999是允许的。

Property	Type	Description
posts	string	HTTP agent item prototype field. HTTP(S) request body data. Used with post_type. HTTP agent 监控项原型字段。HTTP(S) 请求报文数据。用于 post_data。
privatekey	string	Name of the private key file. 私钥文件名。
publickey	string	Name of the public key file. 公钥文件名。
query_fields	array	HTTP agent item prototype field. Query parameters. Array of objects with ’key’:’value’ pairs, where value can be empty string. HTTP agent 监控项原型字段。查询参数。带有键值对的数组对象，值可以为空字符串。
request_method	integer	HTTP agent item prototype field. Type of request method. HTTP agent 监控项原型字段。请求方法类型。 0 - GET 1 - *(default)* POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent item prototype field. What part of response should be stored. HTTP agent 监控项原型字段。指定那一部分的响应应该被存储。 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_community	string	SNMP community. Used only by SNMPv1 and SNMPv2 item prototypes. 仅用于 SNMPv1 和 SNMPv2 监控项原型。
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 item prototypes. SNMPv3 认证密码。仅用于 SNMPv3 监控项原型。
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. SNMPv3 认证协议。仅用于 SNMPv3 监控项原型。 Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 item prototypes. SNMPv3 文本名称。仅用于 SNMPv3 监控项原型。
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 item prototypes. SNMPv3 私有密码。仅用于 SNMPv3 监控项原型。
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. SNMPv3 私有协议。仅用于 SNMPv3 监控项原型。 Possible values: 0 - *(default)* DES; 1 - AES.

Property	Type	Description
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 item prototypes. SNMPv3 安全等级。仅用于 SNMPv3 监控项原型。 Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 item prototypes. SNMPv3 认安全名称。仅用于 SNMPv3 监控项原型。
ssl_cert_file	string	HTTP agent item prototype field. Public SSL Key file path. HTTP agent 监控项原型字段。公共 SSL key 文件路径。
ssl_key_file	string	HTTP agent item prototype field. Private SSL Key file path. HTTP agent 监控项原型字段。私有 SSL key 文件路径。
ssl_key_password	string	HTTP agent item prototype field. Password for SSL Key file. HTTP agent 监控项原型字段。SSL key 文件的密码。
status	integer	Status of the item prototype. 监控项原型的状态。 Possible values: 可能的值。 0 - *(default)* enabled item prototype; 1 - disabled item prototype; 3 - unsupported item prototype.
status_codes	string	HTTP agent item prototype field. Ranges of required HTTP status codes separated by commas. Also supports user macros or LLD macros as part of comma separated list. HTTP agent 监控项原型字段。以逗号分隔的要求的 HTTP 状态码的范围。也接受用户宏和 LLD 宏。 Example: 200,200-{$M},{$M},200-400
templateid	string	(readonly) ID of the parent template item prototype. （只读）父模板的监控项原型的 ID。
timeout	string	HTTP agent item prototype field. Item data polling request timeout. Support user macros and LLD macros. HTTP agent 监控项原型字段。监控项数据合并请求超时时间。支持用户宏和 LLD 宏。 default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper item prototypes or HTTP item prototypes. 允许主机。用于 trapp 监控项原型或者 HTTP 监控项原型。
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro and LLD macro. 趋势数据被保存的时间。也接受用户宏和 LLD 宏。 Default: 365d.
units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes. 认证的用户名。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent 监控项原型。 Required by SSH and Telnet item prototypes. SSH 和 Telnet 监控项原型要求。
valuemapid	string	ID of the associated value map. 相关值映射的 ID。

Property	Type	Description
verify_host	integer	HTTP agent item prototype field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. HTTP agent 监控项原型字段。验证 URL 中的主机名在主机证书中的通用名字段或者备用字段。 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item prototype field. Validate is host certificate authentic. HTTP agent 监控项原型字段。主机合法性认证。 0 - *(default)* Do not validate. 1 - Validate.

itemprototype.create

Description

说明

`object itemprototype.create(object/array itemPrototypes)`

This method allows to create new item prototypes. 此方法用于创建新的 item prototypes（监控项原型）。

Parameters

参数

`(object/array)` Item prototype to create. `(object/array)` 需要创建的 item prototypes。

Additionally to the standard item prototype properties, the method accepts the following parameters. 除标准项原型属性外，该方法还接受以下参数。

Parameter	Type	Description
ruleid (required)	string	ID of the LLD rule that the item belongs to. 该项所属的 LLD 规则的 ID。
applications	array	IDs of applications to be assigned to the discovered items. 要分配给自动发现 item 的应用程序的 ID。
applicationPrototypes	array	Names of application prototypes to be assigned to the item prototype. 要分配给 item prototypes 的应用程序原型的名称。
preprocessing	array	Item prototype preprocessing options. Item prototype 预处理选项。

Return values

返回值

`(object)` Returns an object containing the IDs of the created item prototypes under the `itemids` property. The order of the returned IDs matches the order of the passed item prototypes. `(object)` 返回一个对象，该对象 ID 包含在 “itemid” 属性中。返回的 ID 的顺序与传递的 item prototypes 的顺序相对应。

Examples

示例

Creating an item prototype

Create an item prototype to monitor free disc space on a discovered file system. Discovered items should be numeric Zabbix agent items updated every 30 seconds. 创建一个 item prototype 去监控自动发现的文件系统上的磁盘空间。Discovered items 应该每 30 秒更新数字化的 Zabbix agent 监控项。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Free disk space on $1",
        "key_": "vfs.fs.size[{#FSNAME},free]",
        "hostid": "10197",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "interfaceid": "112",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27666"
        ]
    },
    "id": 1
}
```

Creating an item prototype with preprocessing

创建一个预处理的 item prototype

Create an item using change per second and a custom multiplier as a second step. 创建一个使用每秒变化并带有自定义乘法器作为第二部的监控项。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Incoming network traffic on $1",
        "key_": "net.if.in[{#IFNAME}]",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
        "units": "bps",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": "10",
                "params": ""
            },
            {
                "type": "1",
                "params": "8"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Creating dependent item prototype

创建依赖 item prototype

Create Dependent item prototype for Master item prototype with ID 44211. Only dependencies on same host (template/discovery rule) are allowed, therefore Master and Dependent item should have same hostid and ruleid. 为 ID 为 44211 的主 item prototype 创建一个依赖 item prototype。只有在同一个主机的 (template/discovery rule) 依赖才可以被接受，因此主 item prototype 和依赖 item prototype 应该拥有相同的 hostid 和 ruleid。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
      "hostid": "10001",
      "ruleid": "27665",
      "name": "Dependent test item prototype",
      "key_": "dependent.prototype",
      "type": "18",
      "master_itemid": "44211",
      "value_type": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44212"
        ]
    },
    "id": 1
}
```

Create HTTP agent item prototype

创建 HTTP agent item prototype

Create item prototype with URL using user macro, query fields and custom headers. 创建带有 URL 使用用户宏，查询字段和自定义选项的 item prototype。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "type": "19",
        "hostid": "10254",
        "ruleid":"28256",
        "interfaceid":"2",
        "name": "api item prototype example",
```

```
        "key_": "api_http_item",
        "value_type": "3",
        "url": "{$URL_PROTOTYPE}",
        "query_fields": [
            {
                "min": "10"
            },
            {
                "max": "100"
            }
        ],
        "headers": {
            "X-Source": "api"
        },
        "delay":"35"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

Source

源

CItemPrototype::create() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.delete

Description

说明

`object itemprototype.delete(array itemPrototypeIds)`

This method allows to delete item prototypes. 此方法允许删除 item prototypes。

Parameters

参数

(`array`) IDs of the item prototypes to delete.（`array`）要删除的 item prototypes IDs.

Return values

返回值

(`object`) Returns an object containing the IDs of the deleted item prototypes under the `prototypeids` property.（`object`）`prototypeids` 属性下在返回一个带有被删除的 item prototypes 的 IDs.

Examples

示例

Deleting multiple item prototypes

删除多个 item prototypes

Delete two item prototypes.
Dependent item prototypes are removed automatically if master item or item prototype is deleted. 删除 2 个 item prototypes。如果主 item 或者 item prototype 被删除，，依赖其的 item prototype 也会被删除。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.delete",
    "params": [
        "27352",
        "27356"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "prototypeids": [
            "27352",
            "27356"
        ]
    },
    "id": 1
}
```

Source

源

CItemPrototype::delete() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.get

Description

说明

`integer/array itemprototype.get(object parameters)`

The method allows to retrieve item prototypes according to the given parameters. 此方法可以根据提供的参数获取 item prototypes。

Parameters

`(object)` Parameters defining the desired output. `(object)` 参数定义期望输出

The method supports the following parameters. 此方法提供以下参数。

Parameter	Type	Description
discoveryids	string/array	Return only item prototypes that belong to the given LLD rules. 只返回属于给定 LLD 规则的 item prototypes。
graphids	string/array	Return only item prototypes that are used in the given graph prototypes. 只返回在给定图标原型中使用的 item prototypes。
hostids	string/array	Return only item prototypes that belong to the given hosts. 只返回属于给定 host 的 item prototypes。
inherited	boolean	If set to `true` return only item prototypes inherited from a template. 如果设为”true”，返回继承自某个 template 的 item prototypes。
itemids	string/array	Return only item prototypes with the given IDs. 返回给定 IDS 的 item prototypes。

Parameter	Type	Description
monitored	boolean	If set to `true` return only enabled item prototypes that belong to monitored hosts. 如果设为”true”，只返回已启动的属于已监控主机的 item prototypes。
templated	boolean	If set to `true` return only item prototypes that belong to templates. 如果设为”true”，只发挥属于给定模板的 item prototypes。
templateids	string/array	Return only item prototypes that belong to the given templates. 只返回属于给定模板的 item prototypes。
triggerids	string/array	Return only item prototypes that are used in the given trigger prototypes. 只返回使用在给定 trigger prototypes 的 item prototypes。
selectApplications	query	Return applications that the item prototype belongs to in the `applications` property. 在 `applications` 属性中返回 item prototype 所属的 applications。
selectApplicationPrototypes	query	Return application prototypes linked to item prototype in `applicationPrototypes` property. 只返回被连接到 `applicationPrototypes` 属性中的 item prototype 的 application prototypes。
selectDiscoveryRule	query	Return the low-level discovery rule that the graph prototype belongs to in the `discoveryRule` property. 在 `discoveryRule` 属性中返回图表原型所属的低级发现规则。
selectGraphs	query	Return graph prototypes that the item prototype is used in in the `graphs` property. 在 `graphs` 属性中返回被 item prototype 使用的 graph prototypes。 Supports `count`. 支持 `count`。
selectHosts	query	Returns the host that the item prototype belongs to as an array in the `hosts` property. 在 `hosts` 属性中以数组的形式返回 item prototype 所属的 host。
selectTriggers	query	Return trigger prototypes that the item prototype is used in in the `triggers` property. 在 `triggers` 属性中返回 item prototype 被使用的 trigger prototypes。 Supports `count`. 支持 `count`。
selectPreprocessing	query	Return item preprocessing options in `preprocessing` property. 在 `preprocessing` 属性中返回项 item preprocessing 选项。 It has the following properties: 它有如下属性 `type` - (`string`) The preprocessing option types:preprocessing 选项类型： 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n)character.

Parameter	Type	Description
filter	object	Return only those results that exactly match the given filter. 只返回精确匹配筛选条件的结果。 Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受一个数组，数组键为属性名称，值为单个值或者数组。 Supports additional filters: 支持可选筛选条件： `host` - technical name of the host that the item prototype belongs to. `host` - item prototype 所属的主机的技术名称。
limitSelects	integer	Limits the number of records returned by subselects. 限制子选择返回的记录数。 Applies to the following subselects: 应用于如下子选择： `selectGraphs` - results will be sorted by `name`; `selectTriggers` - results will be sorted by `description`.
sortfield	string/array	Sort the result by the given properties. 根据给定的属性排序 Possible values are: 可能的值有：`itemid`, `name`, `key_`, `delay`, `type` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary. 这些参数对于所有在reference commentary详细描述的”get” 方法都是通用的。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

返回值

`(integer/array)` Returns either:

- an array of objects; 对象数组；
- the count of retrieved objects, if the `countOutput` parameter has been used. 已获取到的对象的数量，如果 `countOutput` 参数被使用。

Examples

示例

Retrieving item prototypes from an LLD rule

获取 item prototypes

Retrieve all item prototypes from an LLD rule. 从 LLD 规则中获取所有 item prototypes

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
```

```
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23077",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10079",
            "name": "Incoming network traffic on $1",
            "key_": "net.if.in[en0]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "bps",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "0",
            "interfaceid": "0",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
```

```
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "10010",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10001",
            "name": "Processor load (1 min average per core)",
            "key_": "system.cpu.load[percpu,avg1]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "0",
            "interfaceid": "0",
            "port": "",
            "description": "The processor load is calculated as system CPU load divided by number of CPU c
            "inventory_link": "0",
            "lifetime": "0",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
```

```
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```


Finding dependent item

查找依赖的 item

Find one Dependent item for item with ID ”25545”. 为 ID 为”25545” 的 item 查找一个赖的 item。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "filter": {
            "type": "18",
            "master_itemid": "25545"
        },
        "limit": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25547",
            "type": "18",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Seconds",
            "key_": "apache.status.uptime.seconds",
            "delay": "0",
```

```
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "0",
            "interfaceid": "0",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
```

```
    "id": 1
}
```

Find HTTP agent item prototype

查找 HTTP agent item prototype

Find HTTP agent item prototype with request method HEAD for specific host id. 为请求方法头定义的 host id 查找 HTTP agent item prototype。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "hostids": "10254",
        "filter": {
            "type": "19",
            "request_method": "3"
        }
    },
    "id": 17,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28257",
            "type": "19",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10254",
            "name": "discovered",
            "key_": "item[{#INAME}]",
            "delay": "{#IUPDATE}",
            "history": "90d",
            "trends": "30d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "28255",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "2",
            "interfaceid": "2",
```

```
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "{#IURL}",
            "query_fields": [],
            "posts": "",
            "status_codes": "",
            "follow_redirects": "0",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 17
}
```


See also

参考其他

- Application
- Host
- Graph prototype
- Trigger prototype

Source

源

CItemPrototype::get() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.update

Description

说明

`object itemprototype.update(object/array itemPrototypes)`

This method allows to update existing item prototypes. 此方法允许更新存在的 item prototypes（监控项原型）。

Parameters

参数

(`object/array`) Item prototype properties to be updated.Item prototype 要更新的属性。

The `itemid` property must be defined for each item prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.Item prototype 的 `itemid` 的属性必须定义，所有其他属性为可选。只用被传递的属性才会被更新，所有其他未被传递的属性保持不变。

Additionally to the standard item prototype properties, the method accepts the following parameters. 此方法接受如下参数。

Parameter	Type	Description 说明
applications	array	IDs of the applications to replace the current applications. 要替换当前应用程序的应用程序的 IDS。
applicationPrototypes	array	Names of the application prototypes to replace the current application prototypes. 要替换当前应用程序原型的应用程序原型名称。
preprocessing	array	Item prototype preprocessing options to replace the current preprocessing options. 要替换当前预处理选项的 item prototype 的预处理选项。

Return values

返回值

(object) Returns an object containing the IDs of the updated item prototypes under the itemids property. (object) 在 itemids 属性中返回一个包含已被更新的 item prototypes 的 IDs 对象。

Examples

示例

Changing the interface of an item prototype

改变 item prototype 的接口

Change the host interface that will be used by discovered items. 改变将被用于发现监控项的主机接口。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "27428",
        "interfaceid": "132"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27428"
        ]
    },
    "id": 1
}
```

Update dependent item prototype

更新依赖的 item prototype

Update Dependent item prototype with new Master item prototype ID. Only dependencies on same host (template/discovery rule) are allowed, therefore Master and Dependent item should have same hostid and ruleid. 以新的主 item prototype 的 ID 来更新依赖的 item prototype。只有在相同 host(template/discovery rule) 的依赖才允许，因此主监控项和依赖监控项应有相同 hostid 和 ruleid。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "master_itemid": "25570",
        "itemid": "189030"
```

```
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189030"
        ]
    },
    "id": 1
}
```

Update HTTP agent item prototype

更新 HTTP agent item prototype

Change query fields and remove all custom headers. 改变查询字段并移除所有自定义请求头。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid":"28305",
        "query_fields": [
            {
                "random": "qwertyuiopasdfghjklzxcvbnm"
            }
        ],
        "headers": []
    }
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

Source

源

CItemPrototype::update() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

Item 监控项

监控项　This class is designed to work with items. 此类用于管理监控项。

Object references:
对象引用：

- Item

- Item

Available methods:
可用的方法：

- item.create - creating new items
- item.create - 创建新监控项
- item.delete - deleting items
- item.delete - 删除监控项
- item.get - retrieving items
- item.get - 检索监控项
- item.update - updating items
- item.update - 更新监控项

> Item object 监控项对象

The following objects are directly related to the `item` API. 以下对象与 “item” API 直接相关。

Item

监控项

Note:
Web items cannot be directly created, updated or deleted via the Zabbix API.

Note:
Web 监控项无法通过 Zabbix API 直接创建，更新或删除。

The item object has the following properties. 监控项对象具有以下属性。

Property 属性	Type 类型	Description 说明
itemid	string	*(readonly)* ID of the item. 监控项 ID。
delay (required)	string	Update interval of the item. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon. 更新监控项的时间间隔。接受具有后缀的秒或时间单位，并且具有或不具有由灵活间隔和调度间隔组成的一个或多个自定义间隔作为串行化字符串。也接受用户宏。灵活的间隔可以写成两个由正斜杠分隔的宏。间隔用分号分隔。 Optional for Zabbix trapper or Dependent item.

hostid (required)	string	ID of the host that the item belongs to. 该监控项所属的主机 ID。 For update operations this field is *readonly*.	
interfaceid (required)	string	ID of the item's host interface. Used only for host items. 监控项主机接口的 ID。仅用于主机项。 Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper, Dependent item, Zabbix aggregate, database monitor and calculated items. 适用于 Zabbix 代理（活动），Zabbix 内部，Zabbix 陷阱，依赖项，Zabbix 聚合，数据库监控和计算项	
	key_ (required)	string	Item key.
name (required)	string	Name of the item.	
type (required)	integer	Type of the item. Possible values: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 9 - web item; 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - calculated; 16 - JMX agent; 17 - SNMP trap; 18 - Dependent item; 19 - HTTP agent;	
url (required)	string	URL string, required only for HTTP agent item type. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}. URL 字符串，仅 HTTP agent 监控项类型需要。支持用户宏，{HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST},{HOST.NAME}, {ITEM.ID}, {ITEM.KEY}。	

value_type (required)	integer	Type of information of the item. 监控项信息的类型。 Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.
allow_traps	integer	HTTP agent item field. Allow to populate value as in trapper item type also. HTTP agent 监控项字段。允许和 trapper 监控项一样的填充值。 0 - *(default)* Do not allow to accept incoming data. 1 - Allow to accept incoming data.
authtype	integer	Used only by SSH agent items or HTTP agent items. 仅在 SSH agent items or HTTP agent items 中使用。 SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM
description	string	Description of the item. 监控项说明。
error	string	*(readonly)* Error text if there are problems updating the item. 当更新监控项出错时的错误文本。
flags	integer	*(readonly)* Origin of the item. Possible values: 0 - a plain item; 4 - a discovered item.
follow_redirects	integer	HTTP agent item field. Follow respose redirects while pooling data. HTTP agent 监控项字段。合并数据时跟随重定向。 0 - Do not follow redirects. 1 - *(default)* Follow redirects.

headers	object	HTTP agent item field. Object with HTTP(S) request headers, where header name is used as key and header value as value. HTTP agent 监控项字段。带有 HTTP(S) 请求报头的对象，报头名为键名，报头值为值。 Example: { "User-Agent": "Zabbix" }
history	string	A time unit of how long the history data should be stored. Also accepts user macro. 一个历史数据被保存的时长的时间单位。接受用户宏。 Default: 90d.
http_proxy	string	HTTP agent item field. HTTP(S) proxy connection string. HTTP agent 监控项字段。HTTP(S) 代理连接字符串。
inventory_link	integer	ID of the host inventory field that is populated by the item. 监控项填充的主机资产的 ID。 Refer to the host inventory page for a list of supported host inventory fields and their IDs. Default: 0.
ipmi_sensor	string	IPMI sensor. Used only by IPMI items. IPMI 传感器。仅用于 IPMI 监控项。
jmx_endpoint	string	JMX agent custom connection string. JMX agent 自定义的连接字符串。 Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
lastclock	timestamp	*(readonly)* Time when the item was last updated. 监控项最后被更新的时间。 This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
lastns	integer	*(readonly)* Nanoseconds when the item was last updated. 监控项最后被更新的纳秒。 This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.

lastvalue	string	*(readonly)* Last value of the item. 监控项最新的值。 This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
logtimefmt	string	Format of the time in log entries. Used only by log items. 日志条目的时间格式。仅用于日志监控项。
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 999 are allowed. 允许多达 3 个依赖监控项的递归和监控项的最大计数等于 999 Required by Dependent items.
mtime	timestamp	Time when the monitored log file was last updated. Used only by log items. 被监控的日志文件最后一次被更新的时间。仅泳衣日志监控项。
output_format	integer	HTTP agent item field. Should response converted to JSON. HTTP agent 监控项字段。返回数据应被转换成 JSON。 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	Additional parameters depending on the type of the item: 取决于监控项类型的附加参数： - executed script for SSH and Telnet items; - SQL query for database monitor items; - formula for calculated items.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. 认证的密码。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX, username should also be specified together with password or both properties should be left blank. 当贝 JMX 使用时，用户名应和密码一起提供，或者同时留空。

port	string	Port monitored by the item. Used only by SNMP items. 监控项监控的端口。仅用于 SMNP 监控项。
post_type	integer	HTTP agent item field. Type of post data body stored in posts property. HTTP agent 字段。存储在 post 属性的 post 的数据类型。 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent item field. HTTP(S) request body data. Used with post_type. HTTP agent 字段。HTTP(S) 请求报文。仅用于 post_type。
prevvalue	string	*(readonly)* Previous value of the item. 监控项的前一个值。 This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
privatekey	string	Name of the private key file. 私钥文件名。
publickey	string	Name of the public key file. 公钥的文件名。
query_fields	array	HTTP agent item field. Query parameters. Array of objects with 'key':'value' pairs, where value can be empty string. HTTP agent 监控项字段。查询参数。带有键值对的数组对象，值可为空。
request_method	integer	HTTP agent item field. Type of request method. HTTP agent 监控项字段。请求方法的类型。 0 - GET 1 - *(default)* POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent item field. What part of response should be stored. HTTP agent 监控项字段。被存储的响应的部分。 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_community	string	SNMP community. Used only by SNMPv1 and SNMPv2 items. SNMP community。仅用于 SNMPv1 and SNMPv2 监控项。

snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 items. SNMPv3 认证密码。仅用于 SNMPv3 监控项。
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. SNMPv3 认证协议。仅用于 SNMPv3 监控项。 Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 items. SNMPv3 文本名称。仅用于 SNMPv3 监控项。
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 items. SNMPv3 私钥。仅用于 SNMPv3 监控项。
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. SNMPv3 文私密协议。仅用于 SNMPv3 监控项。 Possible values: 0 - *(default)* DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 items. SNMPv3 安全等级。仅用于 SNMPv3 监控项。 Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 items. SNMPv3 安全名称。仅用于 SNMPv3 监控项。
ssl_cert_file	string	HTTP agent item field. Public SSL Key file path. HTTP agent 监控项字段。公共 SSL 秘钥的文件路径。
ssl_key_file	string	HTTP agent item field. Private SSL Key file path. HTTP agent 监控项字段。私有 SLL 秘钥的文件路径。
ssl_key_password	string	HTTP agent item field. Password for SSL Key file. HTTP agent 监控项字段。SSL 秘钥的文件密码。
state	integer	*(readonly)* State of the item. Possible values: 0 - *(default)* normal; 1 - not supported.

status	integer	Status of the item. Possible values: 0 - *(default)* enabled item; 1 - disabled item.
status_codes	string	HTTP agent item field. Ranges of required HTTP status codes separated by commas. Also supports user macros as part of comma separated list. HTTP agent 监控项字段。以逗号分隔的 HTTP 状态码的范围。也支持作为逗号分隔的用户宏列表。 Example: 200,200-{$M},{$M},200-400
templateid	string	(readonly) ID of the parent template item. （只读）父模板的 ID。
timeout	string	HTTP agent item field. Item data polling request timeout. Support user macros. HTTP agent 监控项字段。监控项数据轮询超时时间。支持用户宏。 default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper items or HTTP agent items. 接受的主机。仅用于 trapper 监控项或者 HTTP agent 监控项。
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro. 时间单位，数据数据被保存的时间长度。也接受用户宏。 Default: 365d.
units	string	Value units. 值的单位。

username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. 认证的用户名。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent 监控项。 Required by SSH and Telnet items. SSH and Telnet items 要求提供。 When used by JMX, password should also be specified together with username or both properties should be left blank. 当被 JMX 使用时，密码也要和用户名一起被提供或者一起留空。
valuemapid	string	ID of the associated value map. 关联映射值的 ID。
verify_host	integer	HTTP agent item field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. HTTP agent 字段。验证 URL 中的主机名处于通用名称字段或主机证书的主题备用名称字段 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item field. Validate is host certificate authentic. HTTP agent 字段。验证主机的合法性。 0 - *(default)* Do not validate. 1 - Validate.

Item preprocessing 监控项预处理

The item preprocessing object has the following properties. 监控项预处理对象有如下属性。

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second.

Property	Type	Description
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.

item.create

Description 说明

`object item.create(object/array items)`

This method allows to create new items. 此方法允许创建监控项。

> **Note:**
> Web items cannot be created via the Zabbix API. WEB 监控项不能通过 Zabbix API 创建。

Parameters 参数

(`object/array`) Items to create. (`object/array`) 要创建的监控项。

Additionally to the standard item properties, the method accepts the following parameters. 另外见standard item properties，此方法接受如下参数。

Parameter	Type	Description
applications	array	IDs of the applications to add the item to. 要添加到监控项的应用 IDs。
preprocessing	array	Item preprocessing options. 监控项预处理选项。

Return values 返回值

(`object`) Returns an object containing the IDs of the created items under the `itemids` property. The order of the returned IDs matches the order of the passed items. (`object`) 在 `itemids` 属性下返回包含已创建的监控项的对象的 IDs。返回的 IDs 的顺序与传递的监控项的 IDs 的顺序一致。

Examples 示例

Creating an item 创建一个监控项

Create a numeric Zabbix agent item to monitor free disk space on host with ID "30074" and add it to two applications. 创建一个数字类型的 Zabbix agent 监控项监控 ID 为"30074" 的主机的可用磁盘空间并添加到 2 个应用。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on $1",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "30074",
        "type": 0,
        "value_type": 3,
        "interfaceid": "30084",
        "applications": [
            "609",
            "610"
        ],
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24758"
        ]
    },
    "id": 1
}
```

Creating a host inventory item 创建一个主机清单监控项

Create a Zabbix agent item to populate the host's "OS" inventory field. 创建一个 Zabbix agent 监控项填充主机的"OS" 清单字段。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "uname",
        "key_": "system.uname",
        "hostid": "30021",
        "type": 0,
        "interfaceid": "30007",
        "value_type": 1,
        "delay": "10s",
        "inventory_link": 5
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 1
}
```

Creating an item with preprocessing 创建带有预处理的监控项

Create an item using custom multiplier. 使用自定义乘法器创建监控项。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Device uptime",
        "key_": "sysUpTime",
        "hostid": "11312",
        "type": 4,
        "snmp_community": "{$SNMP_COMMUNITY}",
        "snmp_oid": "SNMPv2-MIB::sysUpTime.0",
        "value_type": 1,
        "delay": "60s",
        "units": "uptime",
        "interfaceid": "1156",
        "preprocessing": [
            {
```

```
                "type": "1",
                "params": "0.01"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44210"
        ]
    },
    "id": 1
}
```

Creating dependent item 创建依赖监控项

Create a dependent item for the master item with ID 24759. Only dependencies on the same host are allowed, therefore master and the dependent item should have the same hostid. 为 ID 为 24759 的主监控项创建依赖监控项。仅依同一主机的以来监控项被允许，因此主监控项和依赖监控应有相同的 hostid。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
      "hostid": "30074",
      "name": "Dependent test item",
      "key_": "dependent.item",
      "type": "18",
      "master_itemid": "24759",
      "value_type": "2"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Create HTTP agent item 创建 HTTP agent 监控项

Create POST request method item with JSON response preprocessing. 创建带有 JSON 响应预处理的 POST 请求的方法监控项。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "url":"http://127.0.0.1/http.php",
        "query_fields": [
```

```
            {
                "mode":"json"
            },
            {
                "min": "10"
            },
            {
                "max": "100"
            }
        ],
        "interfaceid": "1",
        "type":"19",
        "hostid":"10254",
        "delay":"5s",
        "key_":"json",
        "name":"http agent example JSON",
        "value_type":"0",
        "output_format":"1",
        "preprocessing": [
            {
                "type": "12",
                "params": "$.random"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

Source

CItem::create() in *frontends/php/include/classes/api/services/CItem.php*.

item.delete

Description 说明

`object item.delete(array itemIds)`

This method allows to delete items. 此方法允许删除监控项。

> **Note:**
> Web items cannot be deleted via the Zabbix API.WEB 监控项不能通过 Zabbix API 删除。

Parameters 参数

(`array`) IDs of the items to delete.（`array`）要删除的监控下的 IDs。

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted items under the `itemids` property.（`object`）在 `itemids` 属性下返回一个包含已被删除的监控项的 IDs 的对象。

Examples 示例

Deleting multiple items 删除多个监控项

Delete two items.
Dependent items and item prototypes are removed automatically if master item is deleted. 删除 2 个监控项。\\如果主监控项被删除，依赖监控项和监控项原型也会被自动删除。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.delete",
    "params": [
        "22982",
        "22986"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22982",
            "22986"
        ]
    },
    "id": 1
}
```

Source

CItem::delete() in *frontends/php/include/classes/api/services/CItem.php*.

item.get

Description 说明

`integer/array item.get(object parameters)`

The method allows to retrieve items according to the given parameters. 此方法允许根据给定的参数获取监控项。

Parameters 参数

`(object)` Parameters defining the desired output. `(object)` 参数定义期望输出。

The method supports the following parameters. 此方法支持如下参数。

Parameter	Type	Description
itemids	string/array	Return only items with the given IDs. 返回给定 IDs 的监控项。
groupids	string/array	Return only items that belong to the hosts from the given groups. 返回属于给定组的主机的监控项。
templateids	string/array	Return only items that belong to the given templates. 返回属于给定模板的监控项。
hostids	string/array	Return only items that belong to the given hosts. 返回属于给定主机的监控项。
proxyids	string/array	Return only items that are monitored by the given proxies. 返回被给定代理监控的监控项。
interfaceids	string/array	Return only items that use the given host interfaces. 返回使用给定主机接口的监控项。
graphids	string/array	Return only items that are used in the given graphs. 返回在给定图表中使用的监控项。
triggerids	string/array	Return only items that are used in the given triggers. 返回给定触发器所使用的监控项。

Parameter	Type	Description
applicationids	string/array	Return only items that belong to the given applications. 返回属于给定应用的监控项。
webitems	flag	Include web items in the result. 返回结果中包含 web 监控项。
inherited	boolean	If set to `true` return only items inherited from a template. 如果设为 `true`，返回继承自某个模板的监控项。
templated	boolean	If set to `true` return only items that belong to templates. 如果设为 `true`，返回属于某个模板的监控项。
monitored	boolean	If set to `true` return only enabled items that belong to monitored hosts. 如果设为 `true`，返回属于已监控主机的已启用的监控项。
group	string	Return only items that belong to a group with the given name. 返回属于给定组名的监控项。
host	string	Return only items that belong to a host with the given name. 返回给定主机名的监控项。
application	string	Return only items that belong to an application with the given name. 返回属于给定应用名的监控项。
with_triggers	boolean	If set to `true` return only items that are used in triggers. 如果设为 `true`，返回在触发器中使用的监控项。
selectHosts	query	Returns the host that the item belongs to as an array in the `hosts` property. 在 `host` 属性中已数组的形式返回监控项所属的主机。
selectInterfaces	query	Returns the host interface used by the item as an array in the `interfaces` property. 在"interfaces" 属性中返回在主机接口中使用的监控项。
selectTriggers	query	Return triggers that the item is used in in the `triggers` property. 在 `triggers` 属性中返回使用该监控项的触发器。 Supports `count`.
selectGraphs	query	Return graphs that contain the item in the `graphs` property. 在 `graphs` 属性中返回包含该监控项的图表。 Supports `count`.
selectApplications	query	Return the applications that the item belongs to in the `applications` property. 在 `application` 属性中返回监控项所属的应用。
selectDiscoveryRule	query	Return the LLD rule that created the item in the `discoveryRule` property. 在 `discoveryRule` 属性中返回创建该监控项的 LLD 规则。
selectItemDiscovery	query	Return the item discovery object in the `itemDiscovery` property. The item discovery object links the item to an item prototype from which it was created. 在 `itemDiscovery` 属性中返回监控项发现对象。该监控项发现对象连接该监控项到监控项原型。 It has the following properties: 它有如下属性： `itemdiscoveryid` - (`string`) ID of the item discovery; `itemid` - (`string`) ID of the discovered item; `parent_itemid` - (`string`) ID of the item prototype from which the item has been created; `key_` - (`string`) key of the item prototype; `lastcheck` - (`timestamp`) time when the item was last discovered; `ts_delete` - (`timestamp`) time when an item that is no longer discovered will be deleted.

Parameter	Type	Description
selectPreprocessing	query	Return item preprocessing options in `preprocessing` property. It has the following properties: `type` - (`string`) The preprocessing option types: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n)character.
filter	object	Return only those results that exactly match the given filter. 返回紧缺匹配给定筛选条件的结果。 Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受数组，数组的键为属性名，值为要匹配的一个值或者值数组。 Supports additional filters: 支持附加筛选条件： `host` - technical name of the host that the item belongs to.
limitSelects	integer	Limits the number of records returned by subselects. 限制子查询所返回的结果的数量。 Applies to the following subselects: 应用到如下子查询： `selectGraphs` - results will be sorted by `name`; `selectTriggers` - results will be sorted by `description`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `itemid`, `name`, `key_`, `delay`, `history`, `trends`, `type` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values 返回值

(`integer/array`) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples 示例

Finding items by key 根据 key 查找监控项

Retrieve all items from host with ID ”10084” that have the word ”system” in the key and sort them by name. 从 ID 为”10084” 的主机获取 key 带有”system” 的监控项，并以名称排序。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10084",
        "search": {
            "key_": "system"
        },
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23298",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Context switches per second",
            "key_": "system.cpu.switches",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "2552",
            "lastclock": "1351090998",
            "prevvalue": "2641",
            "state": "0",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "sps",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "snmpv3_contextname": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "22680",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
```

```
        "privatekey": "",
        "mtime": "0",
        "lastns": "564054253",
        "flags": "0",
        "interfaceid": "1",
        "port": "",
        "description": "",
        "inventory_link": "0",
        "lifetime": "0s",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "1",
        "output_format": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0"
    },
    {
        "itemid": "23299",
        "type": "0",
        "snmp_community": "",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU $2 time",
        "key_": "system.cpu.util[,idle]",
        "delay": "1m",
        "history": "7d",
        "trends": "365d",
        "lastvalue": "86.031879",
        "lastclock": "1351090999",
        "prevvalue": "85.306944",
        "state": "0",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "snmpv3_securityname": "",
        "snmpv3_securitylevel": "0",
        "snmpv3_authpassphrase": "",
        "snmpv3_privpassphrase": "",
        "snmpv3_authprotocol": "0",
        "snmpv3_privprotocol": "0",
        "snmpv3_contextname": "",
        "error": "",
        "lastlogsize": "0",
        "logtimefmt": "",
        "templateid": "17354",
        "valuemapid": "0",
```

```
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "mtime": "0",
        "lastns": "564256864",
        "flags": "0",
        "interfaceid": "1",
        "port": "",
        "description": "The time the CPU has spent doing nothing.",
        "inventory_link": "0",
        "lifetime": "0s",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "1",
        "output_format": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0"
    },
    {
        "itemid": "23300",
        "type": "0",
        "snmp_community": "",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU $2 time",
        "key_": "system.cpu.util[,interrupt]",
        "history": "7d",
        "trends": "365d",
        "lastvalue": "0.008389",
        "lastclock": "1351091000",
        "prevvalue": "0.000000",
        "state": "0",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "snmpv3_securityname": "",
        "snmpv3_securitylevel": "0",
        "snmpv3_authpassphrase": "",
        "snmpv3_privpassphrase": "",
        "snmpv3_authprotocol": "0",
        "snmpv3_privprotocol": "0",
        "snmpv3_contextname": "",
```

```
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "22671",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "lastns": "564661387",
            "flags": "0",
            "interfaceid": "1",
            "port": "",
            "description": "The amount of time the CPU has been servicing hardware interrupts.",
            "inventory_link": "0",
            "lifetime": "0s",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 1
}
```


Finding dependent items by key 根据 key 查找依赖监控项

Retrieve all dependent items from host with ID ”10116” that have the word ”apache” in the key. 从 ID 未”10116” 的主机中获取 key 名包含”apache” 的依赖监控项。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10116",
        "search": {
            "key_": "apache"
        },
        "filter": {
            "type": "18"
```

```
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25550",
            "type": "18",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Days",
            "key_": "apache.status.uptime.days",
            "delay": "",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "0",
            "interfaceid": "0",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
```

```
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "25555",
            "type": "18",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Hours",
            "key_": "apache.status.uptime.hours",
            "delay": "0",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "0",
            "interfaceid": "0",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
            "master_itemid": "25545",
```

```
            "jmx_endpoint": "",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```


Find HTTP agent item 查找 HTTP agent 监控项

Find HTTP agent item with post body type XML for specific host id. 根据定义的主机 id 来查找带有 XML post 报文类型的监控项。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "hostids": "10255",
        "filter": {
            "type": "19",
            "post_type": "3"
        }
    },
    "id": 3,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28252",
            "type": "19",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10255",
            "name": "template item",
            "key_": "ti",
            "delay": "30s",
            "history": "90d",
            "trends": "365d",
            "status": "0",
```

```
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "0",
            "interfaceid": "0",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "localhost",
            "query_fields": [
                {
                    "mode": "xml"
                }
            ],
            "posts": "<body>\r\n<![CDATA[{$MACRO}<foo></bar>]]>\r\n</body>",
            "status_codes": "200",
            "follow_redirects": "0",
            "post_type": "3",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "1",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
```

```
    "id": 3
}
```

See also

- Application
- Discovery rule
- Graph
- Host
- Host interface
- Trigger

Source

CItem::get() in *frontends/php/include/classes/api/services/CItem.php*.

item.update

Description 说明

`object item.update(object/array items)`

This method allows to update existing items. 此方法允许更新已存在的监控项。

> **Note:**
> Web items cannot be updated via the Zabbix API. WEB 监控项不能通过 Zabbix API 更新。

Parameters 参数

(`object/array`) Item properties to be updated. (`object/array`) 要更新的监控项的属性。

The `itemid` property must be defined for each item, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 每个的监控项的 `itemid` 属性必须被定义，其他属性可选。只有被传递的属性才会更新，其他所有属性保持不变。

Additionally to the standard item properties, the method accepts the following parameters. 另外见standard item properties，此方法接受如下参数。

Parameter	Type	Description
applications	array	IDs of the applications to replace the current applications. 要替换当前应用的应用的 ID。
preprocessing	array	Item preprocessing options to replace the current preprocessing options. 要替换的当前监控项预处理选项。

Return values 返回值

(`object`) Returns an object containing the IDs of the updated items under the `itemids` property. (`object`) 在 `itemids` 属性下返回已被更新的监控项的对象的 IDs。

Examples 示例

Enabling an item 启用一个监控项

Enable an item, that is, set its status to "0". 启用一个监控项就是设置他的 status 属性为"0"。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 1
}
```

Update dependent item 更新依赖监控项

Update Dependent item name and Master item ID. Only dependencies on same host are allowed, therefore Master and Dependent item should have same hostid. 更新依赖监控项名称和主监控项的 ID。只有同一个主机上的依赖监控项才允许，因此主监控项和依赖监控项应有相同的 hostid。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "name": "Dependent item updated name",
        "master_itemid": "25562",
        "itemid": "189019"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189019"
        ]
    },
    "id": 1
}
```

Update HTTP agent item 更新 HTTP agent 监控项

Enable item value trapping. 启用监控项的 trapping 值。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "allow_traps": "1"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ]
```

```
    },
    "id": 1
}
```

Source

CItem::update() in *frontends/php/include/classes/api/services/CItem.php*.

LLD rule LLD 规则

LLD 规则 This class is designed to work with low level discovery rules. 此类设计用于低级发现规则。

Object references:
对象参考:

- LLD rule

Available methods:

- discoveryrule.copy - copying LLD rules 复制 LLD 规则
- discoveryrule.create - creating new LLD rules 创建新的 LLD 规则
- discoveryrule.delete - deleting LLD rules 删除 LLD 规则
- discoveryrule.get - retrieving LLD rules
- discoveryrule.update - updating LLD rules 更新 LLD 规则

> LLD rule object LLD 规则对象

> LLD 规则对象

The following objects are directly related to the `discoveryrule` API. 下面的对象直接关联到 `discoveryrule`（发现规则）API。

LLD rule

LLD 规则

The low-level discovery rule object has the following properties. 低级发现规则对象有如下属性。

Property 属性 T	pe 类型 Des	ription 说明
itemid	string	*(readonly)* ID of the LLD rule. LLD 规则的 ID
delay (required)	string	Update interval of the LLD rule. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon.LLD 规则更新间隔。接受 s 或者时间单位，有或没有一个或者多个的灵活间隔和固定计划间隔作为序列化字符串组成的custom intervals
hostid (required)	string	ID of the host that the LLD rule belongs to. LLD 规则所属的 Host 的 ID。

Property 属性 T	pe 类型 Des	ription 说明
interfaceid (required)	string	ID of the LLD rule's host interface. Used only for host LLD rules. LLD 规则所属的 host 的借口的 ID Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper and database monitor LLD rules. Zabbix agent (active), Zabbix internal, Zabbix trapper and 数据库监控 LLD 规则的可选参数。
key_ (required)	string	LLD rule key. LLD 规则键。
name (required)	string	Name of the LLD rule. LLD 规则名称。
type (required)	integer	Type of the LLD rule. LLD 规则类型。 Possible values: 可能的值： 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 16 - JMX agent; 19 - HTTP agent;
url (required)	string	URL string, required for HTTP agent LLD rule. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}. URL 字符串，HTTP agent LLD rule 要求有。支持用户宏，{HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}。
allow_traps	integer	HTTP agent LLD rule field. Allow to populate value as in trapper item type also.HTTP agent LLD 规则字段。在陷阱监控项类型中也允许填充值 0 - *(default)* Do not allow to accept incoming data.0 - *(默认)* 不允许接受输入数据 1 - Allow to accept incoming data.1 - 允许输入数据

Property 属性 T	pe 类型 Des	ription 说明
authtype	integer	Used only by SSH agent or HTTP agent LLD rules. 只能被 SSH agent 或 HTTP agent 使用 SSH agent authentication method possible values: SSH agent 认证方法可能的值： 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values:HTTP agent 认证方法可能的值： 0 - *(default)* none 1 - basic 2 - NTLM
description	string	Description of the LLD rule. LLD 规则说明。
error	string	*(readonly)* Error text if there are problems updating the LLD rule. 如果更新 LLD 规则出问题时的错误文本。
follow_redirects	integer	HTTP agent LLD rule field. Follow respose redirects while pooling data.HTTP agentLLD 规则字段。当合并数据时进行重定向。 0 - Do not follow redirects.0 - 不跟随重定向。 1 - *(default)* Follow redirects. 1 - *(default)* 跟随重定向。
headers	object	HTTP agent LLD rule field. Object with HTTP(S) request headers, where header name is used as key and header value as value. HTTP agent LLD 规则字段。该对象带有 HTTP(S) 已键为名称，包头的值作为值的请求头。 Example: { ”User-Agent”: ”Zabbix” }
http_proxy	string	HTTP agent LLD rule field. HTTP(S) proxy connection string. HTTP agent LLD 规则字段。HTTP(S) proxy 连接字符串。
ipmi_sensor	string	IPMI sensor. Used only by IPMI LLD rules. IPMI sensor。只用于 IPMILLD 规则
jmx_endpoint	string	JMX agent custom connection string. JMX agent 自定义连接字符串。 Default value: 默认值： service:jmx:rmi:///jndi/rmi://{HOST.CONN}

Property 属性 T	pe 类型 Des	ription 说明
lifetime	string	Time period after which items that are no longer discovered will be deleted. Accepts seconds, time unit with suffix and user macro. 不在用于发现的 item 被删除的的时间周期。 Default: 30d.
output_format	integer	HTTP agent LLD rule field. Should response converted to JSON.HTTP agent LLD 规则字段。应返回传递给 JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	Additional parameters depending on the type of the LLD rule: 依赖于 LLD 规则类型的其他参数： - executed script for SSH and Telnet LLD rules;- 为 SSH 何 Telnet LLD 规则执行脚本； - SQL query for database monitor LLD rules;- 数据库监控 LLD 规则的 SQL 查询； - formula for calculated LLD rules. - 计算类的 LLD 规则公式。
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules. 认证密码。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD 规则。
port	string	Port used by the LLD rule. Used only by SNMP LLD rules. LLD 规则使用的端口。仅 SNMP LLD 规则使用
post_type	integer	HTTP agent LLD rule field. Type of post data body stored in posts property.HTTP agent LLD 规则字段。post 数据 body 部分存储在 posts 属性中的类型。 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent LLD rule field. HTTP(S) request body data. Used with post_type. HTTP agent LLD 规则字段。HTTP(S) 请求 body 数据，在 post_type 中使用。
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file. 公共键文件的名称。

Property 属性 T	pe 类型 Des	ription 说明
query_fields	array	HTTP agent LLD rule field. Query parameters. Array of objects with ’key’:’value’ pairs, where value can be empty string. HTTP agent LLD 规则字段。查询参数。带有’key’:’value’ 键值对的数组对象，值可以为空。
request_method	integer	HTTP agent LLD rule field. Type of request method. HTTP agent LLD 规则字段。请求方法类型。 0 - GET 1 - *(default)* POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent LLD rule field. What part of response should be stored. HTTP agent LLD 规则字段。指明哪部分响应应被存储起来。 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_community	string	SNMP community. Required for SNMPv1 and SNMPv2 LLD rules.
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 LLD rules. SNMPv3 认证密码。仅在 SNMPv3 LLD 规则中使用。
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 LLD rules. SNMPv3 认证协议。仅在 SNMPv3 LLD 规则中使用。 Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks. SNMPv3 文本名称。仅在 SNMPv3 检查中使用。
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 LLD rules. SNMPv3 秘钥。仅在 SNMPv3 LLD 规则使用。

Property 属性 T	pe 类型 Des	ription 说明
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 LLD rules. SNMPv3 私密协议。仅在 SNMPv3 LLD 规则使用。 Possible values: 0 - *(default)* DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 LLD rules. SNMPv3 安全等级。仅在 SNMPv3 LLD 规则使用。 Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 LLD rules. SNMPv3 安全名称。仅在 SNMPv3 LLD 规则使用。
ssl_cert_file	string	HTTP agent LLD rule field. Public SSL Key file path. HTTP agent LLD 规则字段。公共 SSL 键文件路径。
ssl_key_file	string	HTTP agent LLD rule field. Private SSL Key file path. HTTP agent LLD 规则字段。私有 SSL 键文件路径。
ssl_key_password	string	HTTP agent LLD rule field. Password for SSL Key file. HTTP agent LLD 规则字段。SSL 键文件密码。
state	integer	*(readonly)* State of the LLD rule. Possible values: 0 - *(default)* normal; 1 - not supported.
status	integer	Status of the LLD rule. Possible values: 0 - *(default)* enabled LLD rule; 1 - disabled LLD rule.
status_codes	string	HTTP agent LLD rule field. Ranges of required HTTP status codes separated by commas. Also supports user macros as part of comma separated list. HTTP agent LLD 规则字段。以逗号分隔的 HTTP 要求的状态码范围。 Example: 200,200-{$M},{$M},200-400
templateid	string	(readonly) ID of the parent template LLD rule. （只读）父模板 LLD 规则的 ID。

Property 属性 T	pe 类型 Des	ription 说明
timeout	string	HTTP agent LLD rule field. Item data polling request timeout. Support user macros. HTTP agent LLD 规则字段。Item 数据轮训请求超时时间。知识用户宏。 default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper LLD rules or HTTP agent LLD rules. 允许的主机。用于 trapper LLD 规则或 HTTP agent LLD 规则。
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules. 认证的用户名。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD 规则 Required by SSH and Telnet LLD rules. SSH 和 Telnet LLD 规则要求。
verify_host	integer	HTTP agent LLD rule field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. HTTP agent LLD 规则字段。URL 中的主机名处于通用名称字段或主机证书的主题备用名称字段的合法性。 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent LLD rule field. Validate is host certificate authentic. HTTP agent LLD 规则字段。主机认证证书合法性。 0 - *(default)* Do not validate. 1 - Validate.

LLD rule filter

LLD 规则过滤器

The LLD rule filter object defines a set of conditions that can be used to filter discovered objects. It has the following properties:
LLD 规则筛选器对象定义一套能被用于过滤器发现对象的条件。它包含如下属性：

Property	Type	Description
conditions (required)	array	Set of filter conditions to use for filtering results.

Property	Type	Description
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(readonly)* Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression. 生成的表达式将用于评估过滤器条件。表达式包含通过其 “ormulaid” 引用特定筛选条件的 ID。`eval_formula` 的值与 `formula` 的值相等
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. 用户定义表达式，用于评估具有自定义表达式的筛选器的条件。表达式必须包含通过其 “公式辅助” 引用特定筛选条件的 ID。表达式中使用的 ID 必须与筛选条件中定义的 ID 完全匹配：没有条件可以保持未使用或省略。 Required for custom expression filters.

LLD rule filter condition

LLD rule 过滤器条件

The LLD rule filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties:
LLD 规则过滤器条件对象定义对 LLD 宏的值执行的单独检查：

Property	Type	Description
macro (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward. 用于从自定义表达式引用条件的任意唯一 ID。只能包含大写字母。在修改过滤条件时，ID 必须由用户定义，但在请求之后，将重新生成 ID。
operator	integer	Condition operator. Possible values: 8 - *(default)* matches regular expression; 9 - does not match regular expression.

Note:
To better understand how to use filters with various types of expressions, see examples on the discoveryrule.get and discoveryrule.create method pages.

discoveryrule.copy

Description

说明

`object discoveryrule.copy(object parameters)`

This method allows to copy LLD rules with all of the prototypes to the given hosts. 此方法允许复制包含所有属性的 LLD 规则到给定的主机。

Parameters 参数

`(object)` Parameters defining the LLD rules to copy and the target hosts.

Parameter	Type	Description
discoveryids	array	IDs of the LLD rules to be copied.
hostids	array	IDs of the hosts to copy the LLD rules to.

Return values

`(boolean)` Returns `true` if the copying was successful.

Examples

Copy an LLD rule to multiple hosts

Copy an LLD rule to two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.copy",
    "params": {
        "discoveryids": [
            "27426"
        ],
        "hostids": [
            "10196",
            "10197"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CDiscoveryrule::copy() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.create

Description 说明

`object discoveryrule.create(object/array lldRules)`

This method allows to create new LLD rules. 此方法允许创建新的 LLD 规则。

Parameters 参数

`(object/array)` LLD rules to create.

Additionally to the standard LLD rule properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	LLD rule filter object for the LLD rule.

Return values

(object) Returns an object containing the IDs of the created LLD rules under the itemids property. The order of the returned IDs matches the order of the passed LLD rules.（object）在 itemids 属性下返回一个包含 IDs 的被创建的 LLD 规则。返回的 IDs 的顺序与传递的 LLD 规则顺序相匹配。

Examples 示例

Creating an LLD rule

Create a Zabbix agent LLD rule to discover mounted file systems. Discovered items will be updated every 30 seconds. 创建 Zabbix agent LLD 规则去发现以已装入的文件系统。发现 items 将被每 30s 被更新一次。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Mounted filesystem discovery",
        "key_": "vfs.fs.discovery",
        "hostid": "10197",
        "type": "0",
        "interfaceid": "112",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Using a filter

Create an LLD rule with a set of conditions to filter the results by. The conditions will be grouped together using the logical "and" operator. 创建有由一套删选条件的得到的 LLD 规则。这些条件将使用逻辑 "和" 运算符将条件组合在一起。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": "0",
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 1,
            "conditions": [
```

```
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex2"
                },
                {
                    "macro": "{#MACRO3}",
                    "value": "@regex3"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Using a custom expression filter 使用自定义表达式的筛选器

Create an LLD rule with a filter that will use a custom expression to evaluate the conditions. The LLD rule must only discover objects the "{#MACRO1}" macro value of which matches both regular expression "regex1" and "regex2", and the value of "{#MACRO2}" matches either "regex3" or "regex4". The formula IDs "A", "B", "C" and "D" have been chosen arbitrarily.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": "0",
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 3,
            "formula": "(A and B) and (C or D)",
            "conditions": [
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1",
                    "formulaid": "A"
                },
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex2",
                    "formulaid": "B"
                },
                {
                    "macro": "{#MACRO2}",
```

```
                    "value": "@regex3",
                    "formulaid": "C"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex4",
                    "formulaid": "D"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Using custom query fields and headers 使用自定义查询字段和报头

Create LLD rule with custom query fields and headers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "hostid": "10257",
        "interfaceid": "5",
        "type": "19",
        "name": "API HTTP agent",
        "key_": "api_discovery_rule",
        "value_type": "3",
        "delay": "5s",
        "url": "http://127.0.0.1?discoverer.php",
        "query_fields": [
            {
                "mode": "json"
            },
            {
                "elements":"2"
            }
        ],
        "headers": {
            "X-Type": "api",
            "Authorization": "Bearer mF_A.B5f-2.1JcM"
        },
        "allow_traps": "1",
        "trapper_hosts": "127.0.0.1",
        "id": 35,
        "auth": "d678e0b85688ce578ff061bd29a20d3b",
    }
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 35
}
```

See also

- LLD rule filter

Source

CDiscoveryRule::create() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.delete

Description 说明

`object discoveryrule.delete(array lldRuleIds)`

This method allows to delete LLD rules. 此方法允许删除 LLD 规则。

Parameters 参数

(`array`) IDs of the LLD rules to delete.（`array`）要删除的 LLD 规则的 IDs。

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted LLD rules under the `itemids` property.（`object`）在 `itemids` 下返回一个包含被删除的 LLD 规则的 IDs。

Examples 示例

Deleting multiple LLD rules 删除多个 LLD 规则

Delete two LLD rules. 删除 2 个 LLD 规则。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.delete",
    "params": [
        "27665",
        "27668"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "ruleids": [
            "27665",
            "27668"
        ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::delete() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.get

Description 说明

`integer/array discoveryrule.get(object parameters)`

The method allows to retrieve LLD rules according to the given parameters. 此方法允许根据给定的参数获取 LLD 规则。

Parameters 参数

(object) Parameters defining the desired output. (object) 参数定义期望输出。

The method supports the following parameters. 此方法支持如下参数。

Parameter	Type	Description
itemids	string/array	Return only LLD rules with the given IDs. 返回给定 IDs 的 LLD 规则。
hostids	string/array	Return only LLD rules that belong to the given hosts. 返回属于给定主机的 LLD 规则。
inherited	boolean	If set to `true` return only LLD rules inherited from a template. 如果设为 `true`，返回自称自某模板的 LLD 规则。
interfaceids	string/array	Return only LLD rules use the given host interfaces. 返回使用给定主机接口的 LLD 规则。
monitored	boolean	If set to `true` return only enabled LLD rules that belong to monitored hosts. 如果设为 `true`，返回已经启用的属于已监控主机的 LLD 规则。
templated	boolean	If set to `true` return only LLD rules that belong to templates. 如果设为 `true`，返回属于（多个）模板的 LLD 规则。
templateids	string/array	Return only LLD rules that belong to the given templates. 返回属于给定给定模板的 LLD 规则。
selectFilter	query	Returns the filter used by the LLD rule in the `filter` property. 在 `filter` 中返回 LLD 使用的筛选器。
selectGraphs	query	Returns graph prototypes that belong to the LLD rule in the `graphs` property. 在 `graphs` 属性中返回属于 LLD 规则的图表原型。 Supports `count`.
selectHostPrototypes	query	Returns host prototypes that belong to the LLD rule in the `hostPrototypes` property. 在 `hostPrototypes` 属性中返回属于该 LLD 规则的主机原型。 Supports `count`.
selectHosts	query	Returns the host that the LLD rule belongs to as an array in the `hosts` property. 在 `hosts` 属性下以数组形式返回属于该 LLD 规则的主机。
selectItems	query	Returns item prototypes that belong to the LLD rule in the `items` property. 在 `items` 下返回属于该 LLD 规则的 item。 Supports `count`.
selectTriggers	query	Returns trigger prototypes that belong to the LLD rule in the `triggers` property. 在 `triggers` 属性下返回属于该触发器原型。 Supports `count`.

Parameter	Type	Description
filter	object	Return only those results that exactly match the given filter. 仅返回紧缺匹配给定筛选条件的结果。 Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受一个数组，这些数组的键为属性名称，值是一个或数组中的值的要匹配的值。 Supports additional filters: `host` - technical name of the host that the LLD rule belongs to.
limitSelects	integer	Limits the number of records returned by subselects. 限制子选择返回的结果的数量。 Applies to the following subselects: `selctItems`; `selectGraphs`; `selectTriggers`.
sortfield	string/array	Sort the result by the given properties. 根据给定的属性把结果进行排序。 Possible values are: `itemid`, `name`, `key_`, `delay`, `type` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values 返回值

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples 示例

Retrieving discovery rules from a host 从一个主机获取多有的发现规则

Retrieve all discovery rules from host "10202". 获取主机 "10202" 所有的发现规则。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": "extend",
        "hostids": "10202"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "27425",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Network interface discovery",
            "key_": "net.if.discovery",
            "delay": "1h",
            "state": "0",
            "status": "0",
            "trapper_hosts": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "error": "",
            "templateid": "22444",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "port": "",
            "description": "Discovery of network interfaces as defined in global regular expression \"Netw
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "snmpv3_contextname": "",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        },
        {
            "itemid": "27426",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Mounted filesystem discovery",
```

```
            "key_": "vfs.fs.discovery",
            "delay": "1h",
            "state": "0",
            "status": "0",
            "trapper_hosts": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "error": "",
            "templateid": "22450",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "port": "",
            "description": "Discovery of file systems of different types as defined in global regular expr
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "snmpv3_contextname": "",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 1
}
```


Retrieving filter conditions

Retrieve the name of the LLD rule "24681" and its filter conditions. The filter uses the "and" evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": [
            "name"
        ],
        "selectFilter": "extend",
```

```
        "itemids": ["24681"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "24681",
            "name": "Filtered LLD rule",
            "filter": {
                "evaltype": "1",
                "formula": "",
                "conditions": [
                    {
                        "macro": "{#MACRO1}",
                        "value": "@regex1",
                        "operator": "8",
                        "formulaid": "A"
                    },
                    {
                        "macro": "{#MACRO2}",
                        "value": "@regex2",
                        "operator": "8",
                        "formulaid": "B"
                    },
                    {
                        "macro": "{#MACRO3}",
                        "value": "@regex3",
                        "operator": "8",
                        "formulaid": "C"
                    }
                ],
                "eval_formula": "A and B and C"
            }
        }
    ],
    "id": 1
}
```

Retrieve LLD rule by URL 根据 URL 获取 LLD 规则

Retrieve LLD rule for host by rule URL field value. Only exact match of URL string defined for LLD rule is supported. 根据主机的规则 URL 字段值获取 LLD 规则。仅返回精确匹配定义的 URL 字符串的规则。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "hostids": "10257",
        "filter": {
            "type": "19",
            "url": "http://127.0.0.1/discoverer.php"
        }
    },
    "id": 39,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28336",
            "type": "19",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10257",
            "name": "API HTTP agent",
            "key_": "api_discovery_rule",
            "delay": "5s",
            "history": "90d",
            "trends": "0",
            "status": "0",
            "value_type": "4",
            "trapper_hosts": "",
            "units": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "flags": "1",
            "interfaceid": "5",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "state": "0",
            "snmpv3_contextname": "",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "http://127.0.0.1/discoverer.php",
            "query_fields": [
                {
                    "mode": "json"
                },
                {
                    "elements": "2"
                }
            ],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
```

```
            "headers": {
                "X-Type": "api",
                "Authorization": "Bearer mF_A.B5f-2.1JcM"
            },
            "retrieve_mode": "0",
            "request_method": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 39
}
```


See also

- Graph prototype
- Host
- Item prototype
- LLD rule filter
- Trigger prototype

Source

CDiscoveryRule::get() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.update

Description 说明

`object discoveryrule.update(object/array lldRules)`

This method allows to update existing LLD rules. 此方法允许更新已存在的 LLD 规则。

Parameters 参数

`(object/array)` LLD rule properties to be updated. `(object/array)` 要更新的”LLD 规则属性。

The `itemid` property must be defined for each LLD rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 每个 LLD 规则的 `itemid` 属性必须被定义，其他属性为可选。值传递要被更新的属性，其他属性保持不变。

Additionally to the standard LLD rule properties, the method accepts the following parameters. 另外见standard LLD rule properties，此方法接受如下参数。

Parameter	Type	Description
filter	object	LLD rule filter object to replace the current filter. LLD 规则要替换当前的筛选对象。

Return values 返回值

`(object)` Returns an object containing the IDs of the updated LLD rules under the `itemids` property. `(object)` 在 `itemids` 属性下返回一个包含被更新的 LLD 规则的 IDs。

Examples 示例

Adding a filter to an LLD rule 为 LLD 规则添加一个筛选器

Add a filter so that the contents of the *{#FSTYPE}* macro would match the *@File systems for discovery* regexp.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
```

```
    "params": {
        "itemid": "22450",
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
                    "macro": "{#FSTYPE}",
                    "value": "@File systems for discovery"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

Disable trapping 禁用 trapping

Disable LLD trapping for discovery rule. 禁用 LLD trapping 发现规则。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "28336",
        "allow_traps": "0"
    },
    "id": 36,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 36
}
```

Source

CDiscoveryRule::update() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

Maintenance 维护模式

This class is designed to work with maintenances. 此类设计用于维护模式。

Object references: 对象引用

- Maintenance
- Time period

Available methods: 可用的方法

- maintenance.create - creating new maintenances 创建新的维护模式
- maintenance.delete - deleting maintenances 删除维护模式
- maintenance.get - retrieving maintenances 获取维护模式
- maintenance.update - updating maintenances 更新维护模式

> Maintenance object 维护模式对象

The following objects are directly related to the `maintenance` API. 如下对象与 `maintenance`API 关联。

Maintenance 维护模式

The maintenance object has the following properties. 维护模式对象有如下属性。

Property	Type	Description
maintenanceid	string	*(readonly)* ID of the maintenance. 维护模式的 ID。
name (required)	string	Name of the maintenance. 维护模式的名称。
active_since (required)	timestamp	Time when the maintenance becomes active. 维护模式生效的时刻。
active_till (required)	timestamp	Time when the maintenance stops being active. 维护模式失效的时刻。
description	string	Description of the maintenance. 维护模式说明。
maintenance_type	integer	Type of maintenance. 维护模式类型。 Possible values: 可能的值： 0 - *(default)* with data collection; 1 - without data collection.

Time period 时间周期

The time period object is used to define periods when the maintenance must come into effect. It has the following properties. 时间周期对象用于定义维护模式生效的时间周期。它有如下属性。

Property	Type	Description
timeperiodid	string	*(readonly)* ID of the maintenance. 维护模式 ID。
day	integer	Day of the month when the maintenance must come into effect. 维护模式生效的月份天次。 Required only for monthly time periods. 月份时间周期要求。
dayofweek	integer	Days of the week when the maintenance must come into effect. 维护模式生效的周次。 Days are stored in binary form with each bit representing the corresponding day. For example, 4 equals 100 in binary and means, that maintenance will be enabled on Wednesday. 日期以二进制形式存储，每个比特代表对应的一天。例如，4 在二进制中等于 100，意味着星期三将启用维护。 Used for weekly and monthly time periods. Required only for weekly time periods. 用于周或月时间周期。仅周时间周期要求。

Property	Type	Description
every	integer	For daily and weekly periods `every` defines day or week intervals at which the maintenance must come into effect. 对于天或者周的周期 `every` 定义维护模式生效的天或者周间隔。 For monthly periods `every` defines the week of the month when the maintenance must come into effect. 对于月周期 `every` 定义该月维护模式生效的周次。 Possible values: 可能的值： 1 - first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week.
month	integer	Months when the maintenance must come into effect. 维护模式必须生效的月份。 Months are stored in binary form with each bit representing the corresponding month. For example, 5 equals 101 in binary and means, that maintenance will be enabled in January and March. 月份以二进制形式存储，每个位代表相应月份。例如，5 在二进制中等于 101，意味着维护将在一月和 3 月启用。 Required only for monthly time periods. 只有月时间周期要求。
period	integer	Time of day when the maintenance starts in seconds. 维护模式周期的时间（秒）。 Default: 3600.
start_date	timestamp	Date when the maintenance period must come into effect. 维护模式必须生效的日期。 Required only for one time periods. Default: current date.
start_time	integer	Time of day when the maintenance starts in seconds. 一天内维护模式开始的时刻。 Required for daily, weekly and monthly periods. 天、周、月周期要求。
timeperiod_type	integer	Type of time period. 时间周期类型。 Possible values: 可能的值： 0 - *(default)* one time only; 2 - daily; 3 - weekly; 4 - monthly.

maintenance.create

Description 说明

`object maintenance.create(object/array maintenances)`

This method allows to create new maintenances. 此方法允许创建新的维护模式。

Parameters 参数

`(object/array)` Maintenances to create.

`(object/array)` 要创建的维护模式。

Additionally to the standard maintenance properties, the method accepts the following parameters. 另外见standard maintenance properties，此方法接受如下参数。

Parameter	Type	Description
groupids (required)	array	IDs of the host groups that will undergo maintenance. 要执行维护模式的主机组 IDs。
hostids (required)	array	IDs of the hosts that will undergo maintenance. 要执行维护模式的主机的 IDs。
timeperiods (required)	array	Maintenance time periods. 维护模式时间周期。

Attention:
At least one host or host group must be defined for each maintenance. 每个维护模式至少一个主机或主机组被定义。

Return values 返回值

(`object`) Returns an object containing the IDs of the created maintenances under the `maintenanceids` property. The order of the returned IDs matches the order of the passed maintenances.（`object`）在 `maintenanceids` 属性中返回一个包含所有已被创建的维护模式的对象的 ID。返回的 IDs 的排序与传递的维护模式的 IDs 顺序一致。

Examples 示例

Creating a maintenance 创建一个维护模式

Create a maintenance with data collection for host group "2". It must be active from 22.01.2013 till 22.01.2014, come in effect each Sunday at 18:00 and last for one hour. 为主机组"2" 以 with data collection(持续收集数据) 模式创建一个维护模式。该维护模式生效于 22.01.2013 到 22.01.2014，每周六的 18:00 生效，并持续 1 个小时。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Sunday maintenance",
        "active_since": 1358844540,
        "active_till": 1390466940,
        "groupids": [
            "2"
        ],
        "timeperiods": [
            {
                "timeperiod_type": 3,
                "every": 1,
                "dayofweek": 64,
                "start_time": 64800,
                "period": 3600
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Time period

Source

CMaintenance::create() in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.delete

Description 说明

`object maintenance.delete(array maintenanceIds)`

This method allows to delete maintenances. 此方法允许删除维护模式。

Parameters 参数

(`array`) IDs of the maintenances to delete.(`array`) 要删除的维护模式的 IDs。

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted maintenances under the `maintenanceids` property.(`object`) 在 `maintenanceids` 属性下返回包含已被删除的维护模式的 ID 对象。

Examples 示例

Deleting multiple maintenances 删除多个维护模式

Delete two maintenanaces. 删除 2 个维护模式。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.delete",
    "params": [
        "3",
        "1"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3",
            "1"
        ]
    },
    "id": 1
}
```

Source

CMaintenance::delete() in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.get

Description 说明

`integer/array maintenance.get(object parameters)`

The method allows to retrieve maintenances according to the given parameters. 此方法用于根据给定参数获取维护模式。

Parameters 参数

(`object`) Parameters defining the desired output.(`object`) 定义期望输出的参数。

The method supports the following parameters. 此方法支持如下参数。

Parameter	Type	Description
groupids	string/array	Return only maintenances that are assigned to the given host groups. 仅返回指定到给定主机组的维护模式。
hostids	string/array	Return only maintenances that are assigned to the given hosts. 仅返回指定到给定主机的维护模式。
maintenanceids	string/array	Return only maintenances with the given IDs. 仅返回给定 IDs 的维护模式。
selectGroups	query	Return host groups assigned to the maintenance in the `groups` property. 在 `group` 属性中返回维护模式所指定的主机组。
selectHosts	query	Return hosts assigned to the maintenance in the `hosts` property. 在 `host` 属性中返回维护模式所指定的主机。
selectTimeperiods	query	Return the maintenance's time periods in the `timeperiods` property. 在 `timeperiods` 属性中返回维护模式的时间周期。
sortfield	string/array	Sort the result by the given properties. 根据给定的属性记性排序。 Possible values are: `maintenanceid`, `name` and `maintenance_type`. 可能的值有：, `name` and `maintenance_type`。
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary. 这些参数在reference commentary中详细描述的所有 `get` 方法是通用的。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples 示例

Retrieving maintenances 获取维护模式

Retrieve all configured maintenances, and the data about the assigned host groups, hosts and defined time periods. 获取所有配置的维护模式，以及关于指定主机组、主机和定义的时间周期数据。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.get",
    "params": {
        "output": "extend",
        "selectGroups": "extend",
        "selectTimeperiods": "extend"
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenanceid": "3",
            "name": "Sunday maintenance",
            "maintenance_type": "0",
            "description": "",
            "active_since": "1358844540",
            "active_till": "1390466940",
            "groups": [
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "internal": "0"
                }
            ],
            "timeperiods": [
                {
                    "timeperiodid": "4",
                    "timeperiod_type": "3",
                    "every": "1",
                    "month": "0",
                    "dayofweek": "1",
                    "day": "0",
                    "start_time": "64800",
                    "period": "3600",
                    "start_date": "2147483647"
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Host
- Host group
- Time period

Source

CMaintenance::get() in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.update

Description 说明

`object maintenance.update(object/array maintenances)`

This method allows to update existing maintenances. 此方法允许更新已存在的维护模式。

Parameters 参数

(`object/array`) Maintenance properties to be updated. (`object/array`) 要更新的维护模式的属性。

The `maintenanceid` property must be defined for each maintenance, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 每一个维护模式的 `maintenanceid` 属性必须被定义，其他所有属性均为可选。只有被传递的属性才会被更新，所有它属性保持不变。

Additionally to the standard maintenance properties, the method accepts the following parameters. 另外见standard maintenance properties, 此方法接受如下参数。

Parameter	Type	Description
groupids	array	IDs of the host groups to replace the current groups. 要替换的当前主机组的主机组 IDs。
hostids	array	IDs of the hosts to replace the current hosts. 要替换当前主机的主机 IDs。
timeperiods	array	Maintenance time periods to replace the current periods. 要替换当前维护模式时间周期的时间周期。

Attention:
At least one host or host group must be defined for each maintenance. 每一个维护模式至少一个主机或者一个主机组被定义。

Return values 返回值

(object) Returns an object containing the IDs of the updated maintenances under the maintenanceids property. (object) 在 maintenanceids 属性中返回一个包含已被更新的维护模式的 IDs 的对象。

Examples 示例

Assigning different hosts 指定不同的主机

Replace the hosts currently assigned to maintenance "3" with two different ones. 用两个不同的主机替换当前分配给维护 "3" 的主机。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.update",
    "params": {
        "maintenanceid": "3",
        "hostids": [
            "10085",
            "10084"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Time period

Source

CMaintenance::update() in *frontends/php/include/classes/api/services/CMaintenance.php*.

Screen item 聚合图形项

This class is designed to work with screen items. 此类设计用来工作于聚合类型项

Object references:

- Screen item

Available methods:

- screenitem.create - creating new screen items
- screenitem.delete - deleting screen items
- screenitem.get - retrieving screen items
- screenitem.update - updating screen items
- screenitem.updatebyposition - updating screen items in a specific screen cell

> 对象

The following objects are directly related to the `screenitem` API. 以下类直接关联到 `screenitem` API.

Screen item 聚合类型项

The screen item object defines an element displayed on a screen. It has the following properties. 聚合类型项定义一个展示元素在聚合图形上，他拥有以下属性

Property	Type	Description
screenitemid	string	*(readonly)* ID of the screen item. *(只读)* 聚合图形项的 ID
resourcetype (required)	integer	Type of screen item. Possible values: 0 - graph; 1 - simple graph; 2 - map; 3 - plain text; 4 - hosts info; 5 - triggers info; 6 - system information; 7 - clock; 8 - screen; 9 - triggers overview 10 - data overview; 11 - URL; 12 - history of actions; 13 - history of events; 14 - latest host group issues; 15 - problems by severity; 16 - latest host issues; 19 - simple graph prototype; 20 - graph prototype. 聚合图形项的类型 可能的值：0 - 图形; 1 - 简图; 2 -拓扑图; 3 -纯文本; 4 -主机信息; 5 -触发信息;6 -系统信息; 7 -时钟; 8 -聚合图形; 9 -触发概览 10 -资料概览; 11 - URL; 12 -动作日志; 13 -历史事件; 14 -最新的主机组问题; 15 -严重程度的问题; 16 -最新的主机问题; 19 -简易图形原型; 20 -图形原型。
screenid (required)	string	ID of the screen that the item belongs to. 监控项所属聚合图形的 id
application	string	Application or part of application name by which data in screen item can be filtered. Applies to resource types: ”Data overview” and ”Triggers overview”. 应用程序或应用程序名称的一部分，通过它可以过滤聚合图形项中的数据。适用于资源类型: ”Data overview” and ”Triggers overview”.
colspan	integer	Number of columns the screen item will span across. Default: 1. 屏幕项将跨越的列数。 \\默认：1

Property	Type	Description
dynamic	integer	Whether the screen item is dynamic. Possible values: 0 - *(default)* not dynamic; 1 - dynamic. 聚合图形项是否是动态的 可能的值： 0 - (默认) 不是动态的 1 - 动态的
elements	integer	Number of lines to display on the screen item. Default: 25. 每行可以展示聚合图像项的个数。 默认：25
halign	integer	Specifies how the screen item must be aligned horizontally in the cell. Possible values: 0 - *(default)* center; 1 - left; 2 - right. 指定聚合图形项必须在单元格中水平对齐的方式。 可能的值：0 - *(default)* 中心；1 - 左边；2 - 右边
height	integer	Height of the screen item in pixels. Default: 200. 聚合图像项的高度，单位是像素 默认：200
max_columns	integer	Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have. Default: 3. 指定图形原型或简单图形原型聚合图形元素可以拥有的最大列数。 默认：200
resourceid	string	ID of the object displayed on the screen item. Depending on the type of a screen item, the `resourceid` property can reference different objects. Required for data overview, graph, map, plain text, screen, simple graph and trigger overview screen items. Unused by local and server time clocks, history of actions, history of events, hosts info, system information, problems by severity and URL screen items. 显示在屏幕项上的对象的 ID。根据屏幕项目的类型，`resourceid` 属性可以引用不同的对象。 关于数据概述、图表、拓扑图、纯文本、屏幕、简单图形和触发器概述屏幕项目的要求。不使用本地和服务器时间时钟、操作历史、事件历史、主机信息、系统信息、严重程度问题和 URL 屏幕项。
rowspan	integer	Number or rows the screen item will span across. Default: 1. 数字或行将跨越聚合图形项。 默认：1

Property	Type	Description
sort_triggers	integer	Order in which actions or triggers must be sorted. Possible values for history of actions screen elements: 3 - time, ascending; 4 - time, descending; 5 - type, ascending; 6 - type, descending; 7 - status, ascending; 8 - status, descending; 9 - retries left, ascending; 10 - retries left, descending; 11 - recipient, ascending; 12 - recipient, descending. Possible values for latest host group issues and latest host issues screen items: 0 - *(default)* last change, descending; 1 - severity, descending; 2 - host, ascending. 操作或触发器必须排序。 \\拓扑图元素历史记录的可能值: 3 - 时间，上升的 4 - 时间，下降的 5 - 类型，上升的 6- 类型，下降的 7 - 状态，升序 8 - 状态，降序 9 -重试，升序 10 - 重试，降序 11-接受者，升序 12-接受者，降序 最新主机组问题和最新主机问题聚合图形项的可能值: 0 - (默认）最后修改，降序 1 - 级别，降序 2 - 主机，升序
style	integer	Screen item display option. Possible values for data overview and triggers overview screen items: 0 - *(default)* display hosts on the left side; 1 - display hosts on the top. Possible values for hosts info and triggers info screen elements: 0 - *(default)* horizontal layout; 1 - vertical layout. Possible values for clock screen items: 0 - *(default)* local time; 1 - server time; 2 - host time. Possible values for plain text screen items: 0 - *(default)* display values as plain text; 1 - display values as HTML. 聚合图形项展示操作 可能的数据概述和触发概述聚合图形项目: 0-*(默认)* 显示主机在左边 1-显示主机在顶部 可能的主机信息和触发器信息聚合图形的值： 0-*(默认)* 水平布局 1-垂直布局 时钟聚合图形项可能的值： 0 - （默认）本地时间 1 - server 时间 2 - 主机时间 纯文本聚合图形项目的可能值: 0-（默认）显示纯文本信息的值 1-显示 HTML 的值
url	string	URL of the webpage to be displayed in the screen item. Used by URL screen items. 将显示在聚合图形项中的网页的 URL。用于 URL 屏幕项。

Property	Type	Description
valign	integer	Specifies how the screen item must be aligned vertically in the cell. Possible values: 0 - *(default)* middle; 1 - top; 2 - bottom. 指定聚合图形项必须在单元格中垂直对齐的方式。 \\可能的值：0-*(默认)* 中间 1-顶部 2-底部
width	integer	Width of the screen item in pixels. Default: 320. 聚合图形项的宽度，单位是像素 默认是：320
x	integer	X-coordinates of the screen item on the screen, from left to right. Default: 0. 屏幕上的屏幕项的 x 坐标，从左到右。 默认：0
y	integer	Y-coordinates of the screen item on the screen, from top to bottom. Default: 0. 屏幕上的屏幕项的 y 坐标，从左到右。 默认：0

screenitem.get

Description

`integer/array screenitem.get(object parameters)`

The method allows to retrieve screen items according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenitemids	string/array	Return only screen items with the given IDs.
screenids	string/array	Return only screen items that belong to the given screen.
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenitemid` and `screenid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	

Parameter	Type	Description
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving screen items from screen

Retrieve all screen items from the given screen.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.get",
    "params": {
        "output": "extend",
        "screenids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenitemid": "20",
            "screenid": "3",
            "resourcetype": "0",
            "resourceid": "433",
            "width": "500",
            "height": "120",
            "x": "0",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "21",
            "screenid": "3",
            "resourcetype": "0",
            "resourceid": "387",
            "width": "500",
            "height": "100",
            "x": "0",
            "y": "1",
            "colspan": "1",
            "rowspan": "1",
```

```
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "22",
            "screenid": "3",
            "resourcetype": "1",
            "resourceid": "10013",
            "width": "500",
            "height": "148",
            "x": "1",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "23",
            "screenid": "3",
            "resourcetype": "1",
            "resourceid": "22181",
            "width": "500",
            "height": "184",
            "x": "1",
            "y": "1",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
        }
    ],
    "id": 1
}
```


Source

CScreenItem::get() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.update

Description

`object screenitem.update(object/array screenItems)`

This method allows to update existing screen items.

Parameters

`(object/array)` Screen item properties to be updated.

The `screenitemid` property must be defined for each screen item, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated screen items under the `screenitemids` property.

Examples

Setting the size of the screen item

Set the width of the screen item to 500px and height to 300px.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.update",
    "params": {
        "screenitemid": "20",
        "width": 500,
        "height": 300
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "20"
        ]
    },
    "id": 1
}
```

See also

- screenitem.updatebyposition

Source

CScreenItem::update() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.updatebyposition

Description

`object screenitem.updatebyposition(array screenItems)`

This method allows to update screen items in the given screen cells. If a cell is empty, a new screen item will be created.

Parameters

`(array)` Screen item properties to be updated.

The `x`, `y` and `screenid` properties must be defined for each screen item, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated and created screen items under the `screenitemids` property.

Examples

Changing a screen items resource ID

Change the resource ID for the screen element located in the upper-left cell of the screen.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.updatebyposition",
    "params": [
        {
            "screenid": "16",
            "x": 0,
            "y": 0,
            "resourceid": "644"
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "66"
        ]
    },
    "id": 1
}
```

See also

- screenitem.update

Source

CScreenItem::update() in *frontends/php/include/classes/api/services/CScreenItem.php*.

创建

Description 描述

`object screenitem.create(object/array screenItems)`

This method allows to create new screen items. 此方法允许创建一个新的聚合类型项

Parameters 参数

(`object/array`) Screen items to create.

(`object/array`) 创建聚合图形项

The method accepts screen items with the standard screen item properties. 此方法接受聚合图形项关于standard screen item properties.

Return values 返回值

(`object`) Returns an object containing the IDs of the created screen items under the `screenitemids` property. The order of the returned IDs matches the order of the passed screen items.（`object`）返回一个对象，该对象包含在 `screenitemids` 属性下创建的聚合图形项的 id。返回 id 的顺序与所传递的聚合图形项的顺序相匹配。

Examples 示例如下

Creating a screen item 创建一个聚合图形项

Create a screen item displaying a graph in the left-upper cell of the screen. 创建一个聚合图像项展示一个图像在左上角的聚合图形

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.create",
    "params": {
        "screenid": 16,
        "resourcetype": 0,
        "resourceid": 612,
        "x": 0,
        "y": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "65"
        ]
    },
    "id": 1
}
```

See also

- screen.update

Source

CScreenItem::create() in *frontends/php/include/classes/api/services/CScreenItem.php*.

删除

Description 描述

`object screenitem.delete(array screenItemIds)`

This method allows to delete screen items. 此方法允许删除一个聚合图形项

Parameters 参数

(array) IDs of the screen items to delete.（array）删除聚合图形项的 IDs

Return values 返回值

(object) Returns an object containing the IDs of the deleted screen items under the `screenitemids` property.（object）返回在 `screenitemids` 属性下包含已删除聚合图形项的 id 的对象。

Examples 示例如下

Deleting multiple screen items 删除多个聚合图像项

Delete two screen items. 删除两个聚合图形项

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.delete",
    "params": [
        "65",
        "63"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "65",
            "63"
        ]
    },
    "id": 1
}
```

See also

- screen.update

Source

CScreenItem::delete() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screen 聚合图形

This class is designed to work with screen. 这个类设计工作于聚合图形

Object references:

- Screen
- Screen user
- Screen user group

Available methods:

- screen.create - creating new screen
- screen.delete - deleting screens
- screen.get - retrieving screens
- screen.update - updating screens

> 对象

The following objects are directly related to the `screen` API. 以下对象直接跟 `screen` API 相关

Screen 聚合图形

The screen object has the following properties. 聚合图形对象拥有以下属性

Property	Type	Description
screenid	string	*(readonly)* ID of the screen. *(readonly)* 聚合图形 ID
name (required)	string	Name of the screen. 聚合图形名称
hsize	integer	Width of the screen. Default: 1 聚合图形宽度 默认：1
vsize	integer	Height of the screen. Default: 1 聚合图形高度 默认：1
userid	string	Screen owner user ID. 聚合图形所属用户 ID

Property	Type	Description
private	integer	Type of screen sharing. Possible values: 0 - public screen; 1 - *(default)* private screen. 聚合图形共享类型 可能的值： 0 - 共有的聚合图形 1 - （默认）私有的聚合图形

Screen user 聚合图形用户

List of screen permissions based on users. It has the following properties: 聚合图形的权限基于用户，它拥有以下属性：

Property	Type	Description
screenuserid	string	*(readonly)* ID of the screen user. *(默认)* 聚合图形用户 ID
userid (required)	string	User ID. 用户 ID
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write; 权限等级类别 可能的值： 2 - 只读 3 - 读写权限

Screen user group 聚合图形用户组

List of screen permissions based on user groups. It has the following properties: 基于用户组的聚合图形权限列表。它拥有以下属性：

Property	Type	Description
screenusrgrpid	string	*(readonly)* ID of the screen user group. 聚合图形用户组 ID
usrgrpid (required)	string	User group ID. 用户组 ID
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write; 权限等级类别 可能的值： 2 - 只读 3 - 读写权限

创建

Description 描述

`object screen.create(object/array screens)`

This method allows to create new screens. 此方法允许创建新的聚合图形

Parameters 参数

`(object/array)` Screens to create. `(object/array)` 创建聚合图形

Additionally to the standard screen properties, the method accepts the following parameters. 此外standard screen properties，此方法接受以下参数：

Parameter	Type	Description
screenitems	array	Screen items to be created for the screen. 为聚合图形创建聚合图形项
users	array	Screen user shares to be created on the screen. 聚合图形用户共享在聚合图形上创建
userGroups	array	Screen user group shares to be created on the screen. 聚合图形用户组共享在聚合图形上创建

Return values 返回值

(`object`) Returns an object containing the IDs of the created screens under the `screenids` property. The order of the returned IDs matches the order of the passed screens.（`object`）返回一个对象，该对象包含在 `screenids` 属性下创建的聚合图形的 id。返回的 id 的顺序与传递的聚合图形的顺序相匹配。

Examples 示例如下

Creating a screen 创建一个聚合图形

Create a screen named "Graphs" with 2 rows and 3 columns and add a graph to the upper-left cell. 创建一个 2 行 3 列名字叫"Graphs" 的聚合图形，并且在表格的左上角添加一个图形。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.create",
    "params": {
        "name": "Graphs",
        "hsize": 3,
        "vsize": 2,
        "screenitems": [
            {
                "resourcetype": 0,
                "resourceid": "612",
                "rowspan": 1,
                "colspan": 1,
                "x": 0,
                "y": 0
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "26"
        ]
    },
    "id": 1
}
```

Screen sharing

Create a screen with two types of sharing (user and user group). 创建一个两种共享类型的聚合图形（用户和用户组）

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.create",
    "params": {
```

```
        "name": "Screen sharing",
        "hsize": 3,
        "vsize": 2,
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "83"
        ]
    },
    "id": 1
}
```

See also

- Screen item
- Screen user
- Screen user group

Source

CScreen::create() in *frontends/php/include/classes/api/services/CScreen.php*.

删除

Description 描述

`object screen.delete(array screenIds)`

This method allows to delete screens. 此方法允许删除聚合图形

Parameters 参数

(array) IDs of the screens to delete.（array）删除聚合图形的 IDs

Return values 返回值

(object) Returns an object containing the IDs of the deleted screens under the screenids property. （object）返回包含 screenids 属性下的已删除屏幕的 id 的对象。

Examples 示例如下

Deleting multiple screens 删除多个聚合图形

Delete two screens. 删除两个聚合图形

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.delete",
```

```
    "params": [
        "25",
        "26"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "25",
            "26"
        ]
    },
    "id": 1
}
```

Source

CScreen::delete() in *frontends/php/include/classes/api/services/CScreen.php*.

更新

Description 描述

`object screen.update(object/array screens)`

This method allows to update existing screens. 此方法允许更新已存在的聚合图形

Parameters 参数

(`object/array`) Screen properties to be updated. (`object/array`) 聚合图形参数将被更新

The `screenid` property must be defined for each screen, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 每个聚合图形必须定义 `screenid` 参数，其他参数是可以选择的。仅传递的参数会被更新，其他的的参数将保持不变。

Additionally to the standard screen properties, the method accepts the following parameters. 此外standard screen properties, 此方法接受以下参数

Parameter	Type	Description
screenitems	array	Screen items to replace existing screen items. Screen items are updated by coordinates, so each screen item must have the `x` and `y` properties defined. 聚合图形项替换已存在的聚合图行项 聚合图形项通过坐标更新，所以每个聚合图形项必须拥定义 `x` and `y` 属性
users	array	Screen user shares to replace the existing elements. 聚合图形用户共享替换已存在的元素
userGroups	array	Screen user group shares to replace the existing elements. 聚合图形用户组共享替换已存在的元素

Return values 返回值

(`object`) Returns an object containing the IDs of the updated screens under the `screenids` property. (`object`) 返回一个对象，该对象包含 `screenids` 属性下更新聚合图形的 id。

Examples 示例如下

Renaming a screen 重命名一个聚合图形

Rename a screen to ”CPU Graphs”. 重命名一个聚合图形为”CPU Graphs”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.update",
    "params": {
        "screenid": "26",
        "name": "CPU Graphs"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "26"
        ]
    },
    "id": 1
}
```

Change screen owner 改变聚合图形属主

Available only for admins and super admins. 仅仅适用于管理员和超级管理员

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.update",
    "params": {
        "screenid": "83",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "83"
        ]
    },
    "id": 2
}
```

See also

- Screen item
- screenitem.create
- screenitem.update
- screenitem.updatebyposition
- Screen user
- Screen user group

Source

CScreen::update() in *frontends/php/include/classes/api/services/CScreen.php*.

获取

Description 描述

`integer/array screen.get(object parameters)`

The method allows to retrieve screens according to the given parameters. 此方法允许搜索符合所给参数的聚合图形

Parameters 参数

`(object)` Parameters defining the desired output. `(object)` 定义所需输出的参数。

The method supports the following parameters. 此方法支持以下参数

Parameter	Type	Description
screenids	string/array	Return only screens with the given IDs. 返回所给 IDs 的聚合图形
userids	string/array	Return only screens that belong to the given user IDs. 返回所给用户 IDs 的聚合图形
screenitemids	string/array	Return only screen that contain the given screen items. 返回所给聚合图形项的的聚合图形
selectUsers	query	Returns users that the screen is shared with in `users` property. 返回 `users` 属性中与聚合图形共享的用户。
selectUserGroups	query	Returns user groups that the screen is shared with in `userGroups` property. 返回 `userGroups` 属性中与聚合图形共享的用户组。
selectScreenItems	query	Return the screen items that are used in the screen. 返回聚合图形上使用的聚合图形项。
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenid` and `name`. 根据所给参数对结果进行排序 可能的值：`screenid` 和 `name`
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page. 这个参数通用与所有的 `get` 方法，详细描述在reference commentary页
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values 返回值

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.
- 一个数组对象
- 查看对象的个数，如果 `countOutput` 参数被使用

Examples 示例如下

Retrieving a screen by ID 通过 ID 查看一个聚合图形

Retrieve all data about screen "26" and its screen items. 搜索所有的数据关于聚合图形 id 是 26 和他的聚合图形项

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.get",
    "params": {
        "output": "extend",
        "selectScreenItems": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "screenids": "26"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenitems": [
                {
                    "screenitemid": "67",
                    "screenid": "26",
                    "resourcetype": "0",
                    "resourceid": "612",
                    "width": "320",
                    "height": "200",
                    "x": "0",
                    "y": "0",
                    "colspan": "0",
                    "rowspan": "0",
                    "elements": "25",
                    "valign": "0",
                    "halign": "0",
                    "style": "0",
                    "url": "",
                    "dynamic": "0",
                    "sort_triggers": "0"
                }
            ],
            "users": [
                {
                    "sysmapuserid": "1",
                    "userid": "2",
                    "permission": "2"
                }
            ],
            "userGroups": [
                {
                    "screenusrgrpid": "1",
                    "usrgrpid": "7",
                    "permission": "3"
                }
            ],
            "screenid": "26",
            "name": "CPU Graphs",
            "hsize": "3",
            "vsize": "2",
            "templateid": "0",
            "userid": "1",
            "private": "1"
        }
```

```
    ],
    "id": 1
}
```

See also

- Screen item
- Screen user
- Screen user group

Source

CScreen::get() in *frontends/php/include/classes/api/services/CScreen.php*.

Script 脚本

This class is designed to work with scripts. 这个类设计工作于脚本。

Object references:

- Script

Available methods:

- script.create - create new scripts
- script.delete - delete scripts
- script.execute - run scripts
- script.get - retrieve scripts
- script.getscriptsbyhosts - retrieve scripts for hosts
- script.update - update scripts

> 对象

The following objects are directly related to the `script` API. 以下对象直接关联到 `script` API.

脚本

The script object has the following properties. 这个脚本对象拥有以下属性

Property	Type	Description
scriptid	string	*(readonly)* ID of the script. 脚本的 ID
command (required)	string	Command to run. 运行命令
name (required)	string	Name of the script. 脚本名称
confirmation	string	Confirmation pop up text. The pop up will appear when trying to run the script from the Zabbix frontend. 确认弹出文本信息，如果尝试在 zabbix 界面运行脚本，将会弹出文本信息。
description	string	Description of the script. 脚本描述
execute_on	integer	Where to run the script. Possible values: 0 - run on Zabbix agent; 1 - run on Zabbix server. 2 - *(default)* run on Zabbix server (proxy). 哪儿可以去运行这个脚本 可能的值： 0 - 运行在 zabbix agent 1 - 运行在 zabbix server 2 - *(默认)* 运行在 zabbix server 或 zabbix proxy

Property	Type	Description
groupid	string	ID of the host group that the script can be run on. If set to 0, the script will be available on all host groups. Default: 0. 可以运行脚本主机组的 ID，如果设置为 0，这个脚本适用于所有的主机组
host_access	integer	Host permissions needed to run the script. Possible values: 2 - *(default)* read; 3 - write. 运行脚本主机的权限 可能的值： 2 - *(默认)* 读 3 - 写
type	integer	Script type. Possible values: 0 - *(default)* script; 1 - IPMI. 脚本类型 可能的值： 0 - *(默认)* 脚本 1 - IPMI
usrgrpid	string	ID of the user group that will be allowed to run the script. If set to 0, the script will be available for all user groups. Default: 0. 允许运行脚本的用户组的 ID，如果设置为 0，这个脚本适用于所有的的用户组

script.getscriptsbyhosts 通过主机获取脚本

Description 描述

`object script.getscriptsbyhosts(array hostIds)`

This method allows to retrieve scripts available on the given hosts. 此方法允许检索适用所给主机的脚本

Parameters 参数

`(string/array)` IDs of hosts to return scripts for.`(string/array)` 主机 IDs

Return values 返回值

`(object)` Returns an object with host IDs as properties and arrays of available scripts as values.`(object)` 返回一个对象，该对象的主机 id 作为属性，而可用脚本的数组作为值。<note tip>The method will automatically expand macros in the `confirmation` text. ::: <note tip> 该方法将在 `confirmation` 文本中自动扩展宏。:::

Examples 示例如下

Retrieve scripts by host IDs 通过主机的 ids 检索脚本

Retrieve all scripts available on hosts ”30079” and ”30073”. 检索所有适用于主机”30079” 和”30073” 的脚本。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyhosts",
    "params": [
        "30079",
        "30073"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "30079": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "hostid": "10001"
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "hostid": "10001"
            },
            {
                "scriptid": "2",
                "name": "Traceroute",
                "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "hostid": "10001"
            }
        ],
        "30073": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "hostid": "10001"
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
```

```
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "hostid": "10001"
            },
            {
                "scriptid": "2",
                "name": "Traceroute",
                "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "hostid": "10001"
            }
        ]
    },
    "id": 1
}
```

Source

CScript::getScriptsByHosts() in *frontends/php/include/classes/api/services/CScript.php*.

创建

Description 描述

`object script.create(object/array scripts)`

This method allows to create new scripts. 此方法允许创建一个新的脚本

Parameters 参数

(`object/array`) Scripts to create.

The method accepts scripts with the standard script properties.

Return values

(`object`) Returns an object containing the IDs of the created scripts under the `scriptids` property. The order of the returned IDs matches the order of the passed scripts.（`object`）返回一个对象，该对象包含在 `scriptids` 属性下创建的脚本的 id。返回的 id 的顺序与通过的脚本的顺序相匹配。

Examples 示例如下

Create a script 创建一个脚本

Create a script that will reboot a server. The script will require write access to the host and will display a configuration message before running in the frontend. 创建一个重启一个 server 的脚本，这个脚本需要对该主机有写的权限，并且在脚本运行在界面之前会提示一个确认信息。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Reboot server",
        "command": "reboot server 1",
        "host_access": 3,
        "confirmation": "Are you sure you would like to reboot the server?"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3"
        ]
    },
    "id": 1
}
```

Source

CScript::create() in *frontends/php/include/classes/api/services/CScript.php*.

删除

Description 描述

`object script.delete(array scriptIds)`

This method allows to delete scripts. 此方法允许去删除脚本

Parameters 参数

(`array`) IDs of the scripts to delete.(`array`) 返回删除脚本的 IDs

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted scripts under the `scriptids` property.(`object`) 返回一个对象包含在 `scriptids` 属性之下删除的脚本

Examples 示例如下

Delete multiple scripts 删除多个脚本

Delete two scripts. 删除两个脚本。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.delete",
    "params": [
        "3",
        "4"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3",
            "4"
        ]
    },
    "id": 1
}
```

Source

CScript::delete() in *frontends/php/include/classes/api/services/CScript.php*.

更新

Description 描述

`object script.update(object/array scripts)`

This method allows to update existing scripts. 此方法更新已存在的脚本

Parameters 参数

`(object/array)` Script properties to be updated.

The `scriptid` property must be defined for each script, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. `scriptid` 属性必须被每个脚本定义，其他属性是可选择的。仅仅传递的参数会被更新，其他参数将保持不变。

Return values 返回值

`(object)` Returns an object containing the IDs of the updated scripts under the `scriptids` property.`(object)` 返回一个对象包含在 `scriptids` 属性下更新脚本的 IDs

Examples 示例如下

Change script command 改变一个脚本的命令

Change the command of the script to "/bin/ping -c 10 {HOST.CONN} 2>&1". 改变一个脚本的命令为"/bin/ping -c 10 {HOST.CONN} 2>&1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.update",
    "params": {
        "scriptid": "1",
        "command": "/bin/ping -c 10 {HOST.CONN} 2>&1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CScript::update() in *frontends/php/include/classes/api/services/CScript.php*.

脚本执行

Description 描述

`object script.execute(object parameters)`

This method allows to run a script on a host. 此方法允许在一个主机上运行一个脚本

Parameters 参数

`(object)` Parameters containing the ID of the script to run and the ID of the host.`(object)` 参数包含要运行脚本的 id 和主机的 id

Parameter	Type	Description
hostid (required)	string	ID of the host to run the script on. 要运行脚本的主机 id
scriptid (required)	string	ID of the script to run. 要运行脚本的脚本 id

Return values 返回值

(object) Returns the result of script execution. (object) 返回脚本执行的结果

Property	Type	Description
response	string	Whether the script was run successfully. Possible values: success or failed. 脚本是否执行成功 可能的值：成功或 失败
value	string	Script output. 脚本的输出结果

Examples 示例如下

Run a script 运行一个脚本

Run a "ping" script on a host. 在一个主机上运行一个"ping" 脚本

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "1",
        "hostid": "30079"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes from 127.0.0.1: icmp_req=1 tt
    },
    "id": 1
}
```

Source

CScript::execute() in *frontends/php/include/classes/api/services/CScript.php*.

获取

Description 描述

integer/array script.get(object parameters)

The method allows to retrieve scripts according to the given parameters. 此方法允许检索符合所给参数的脚本

Parameters 参数

(object) Parameters defining the desired output. (object) 定义所需输出的参数。

The method supports the following parameters. 此方法支持以下参数

Parameter	Type	Description
groupids	string/array	Return only scripts that can be run on the given host groups. 仅能运行在所给主机组的脚本
hostids	string/array	Return only scripts that can be run on the given hosts. 仅能运行在所给主机的脚本
scriptids	string/array	Return only scripts with the given IDs. 仅返回所给 IDs 的脚本
usrgrpids	string/array	Return only scripts that can be run by users in the given user groups. 仅返回所给用户组可以运行的脚本
selectGroups	query	Return host groups that the script can be run on in the `groups` property. 返回可以在 `groups` 属性中运行脚本的主机组。
selectHosts	query	Return hosts that the script can be run on in the `hosts` property. 返回可以在 `hosts` 属性中运行脚本的主机组。
sortfield	string/array	Sort the result by the given properties. Possible values are: `scriptid` and `name`. 根据所给参数对参数进行排序 \\可能的值：`scriptid` 和 `name`
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values 返回值

(`integer/array`) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.
- 一个数组对象
- 检索到对象的数目，如果 `countOutput` 参数被使用

Examples 示例如下

Retrieve all scripts 检索所有脚本

Retrieve all configured scripts. 检索所有的已确认的脚本

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
        {
            "scriptid": "1",
            "name": "Ping",
            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1"
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1"
        },
        {
            "scriptid": "3",
            "name": "Detect operating system",
            "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1"
        }
    ],
    "id": 1
}
```


See also

- Host
- Host group

Source

CScript::get() in *frontends/php/include/classes/api/services/CScript.php*.

User group [用户组]

This class is designed to work with user groups.

Object references:

- User group

Available methods:

- usergroup.create - creating new user groups
- usergroup.delete - deleting user groups
- usergroup.get - retrieving user groups
- usergroup.update - updating user groups

> 用户组对象

以下对象与 `usergroup` 直接相关。

> User group object

The following objects are directly related to the `usergroup` API.

用户组

用户组对象具有以下属性。

属性类	说明	
usrgrpid	string	*(readonly)* 用户组的ID。
name (required)	string	用户组的名称。
debug_mode	integer	是否启用或禁用调试模式。 可能的值: 0 - *(default)* 禁用; 1 - 启用。

属性类	说明	
gui_access	integer	组中用户的前端身份验证方法。 可能的值: 0 - *(default)* 使用系统默认身份验证方法; 1 - 使用内部认证; 2 - 禁止访问前端。

属性类	说明	
users_status	integer	用户组是启用还是禁用。 可能的值: 0 - *(default)* 启用; 1 - 禁用。

User group

The user group object has the following properties.

Property	Type	Description
usrgrpid	string	*(readonly)* ID of the user group.
name (required)	string	Name of the user group.
debug_mode	integer	Whether debug mode is enabled or disabled. Possible values are: 0 - *(default)* disabled; 1 - enabled.
gui_access	integer	Frontend authentication method of the users in the group. Possible values: 0 - *(default)* use the system default authentication method; 1 - use internal authentication; 2 - disable access to the frontend.
users_status	integer	Whether the user group is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.

权限

权限对象具有以下属性。

属性类	说明	
id (required)	string	要添加权限的主机组的ID。
permission (required)	integer	访问到主机组的级别。 \\可能的值： 0 - 拒绝访问; 2 - 只读访问; 3 -读写访问。

Permission

The permission object has the following properties.

Property	Type	Description
id (required)	string	ID of the host group to add permission to.
permission (required)	integer	Access level to the host group. Possible values: 0 - access denied; 2 - read-only access; 3 - read-write access.

基于标签的权限

基于标签的权限对象具有以下属性。

属性类	说明	
groupid (required)	string	要添加权限的主机组的 ID。
tag	string	标签名。
value	string	标签值。

Tag based permission

The tag based permission object has the following properties.

Property	Type	Description
groupid (required)	string	ID of the host group to add permission to.
tag	string	Tag name.
value	string	Tag value.

usergroup.create

说明

`object usergroup.create(object/array userGroups)` 此方法允许创建新的用户组。

Description

`object usergroup.create(object/array userGroups)`

This method allows to create new user groups.

参数

`(object/array)` 要创建的用户组。

除了标准用户组属性之外, 该方法接受以下参数。

属性类	说明	
rights	object/array	分配给组的权限
tag_filters	array	基于标签的权限分配给组
userids	string/array	要添加到用户组的用户的 ID。

Parameters

`(object/array)` User groups to create.

Additionally to the standard user group properties, the method accepts the following parameters.

Parameter	Type	Description
rights	object/array	Permissions to assign to the group
tag_filters	array	Tag based permissions to assign to the group
userids	string/array	IDs of users to add to the user group.

返回值

`(object)` 返回包含 “usrgrpids” 属性下创建的用户组的 ID 的对象。返回的 ID 的顺序与传递的用户组的顺序相匹配。

Return values

`(object)` Returns an object containing the IDs of the created user groups under the `usrgrpids` property. The order of the returned IDs matches the order of the passed user groups.

示例

创建一个用户组

创建一个用户组，拒绝访问主机组 “2”，并向其添加用户。

Examples

Creating a user group

Create a user group, which denies access to host group "2", and add a user to it.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.create",
    "params": {
        "name": "Operation managers",
        "rights": {
            "permission": 0,
            "id": "2"
        },
        "userids": "12"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20"
        ]
    },
    "id": 1
}
```

参见

- Permission

来源

CUserGroup::create() in *frontends/php/include/classes/api/services/CUserGroup.php*.

See also

- Permission

Source

CUserGroup::create() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.delete

说明

`object usergroup.delete(array userGroupIds)`

此方法允许删除用户组。

参数

(array) 要删除的用户组的 ID。

返回值

(object) 返回包含 "usrgrpids" 属性下删除的用户组的 ID 的对象。

Description

`object usergroup.delete(array userGroupIds)`

This method allows to delete user groups.

Parameters

(array) IDs of the user groups to delete.

Return values

(object) Returns an object containing the IDs of the deleted user groups under the usrgrpids property.

示例

删除多个用户组

Examples

Deleting multiple user groups

删除 2 个用户。

Delete two user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.delete",
    "params": [
        "20",
        "21"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20",
            "21"
        ]
    },
    "id": 1
}
```

来源

CUserGroup::delete() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.get

说明

integer/array usergroup.get(object parameters)

该方法允许根据给定的参数检索用户组。

Description

integer/array usergroup.get(object parameters)

The method allows to retrieve user groups according to the given parameters.

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

属性类	说明	
status	integer	只返回具有给定状态的用户组。 请参阅用户组页面以获取支持的状态列表。
userids	string/array	只返回包含给定用户的用户组。
usrgrpids	string/array	只返回具有给定ID的用户组。

属性类	说明	
with_gui_access	integer	只返回具有给定前端身份验证方法的用户组。 有关支持的方法的列表，请参阅用户组页面。

属性类	说明	
selectTagFilters	query	Return user group tag based permissions in the `tag_filters` property. It has the following properties: `groupid` - (string) ID of the host group; `tag` - (string) tag name; `value` - (string) tag value.
selectUsers	query	在“users”属性中返回用户组中的用户。

属性类	说明	
selectRights	query	在“权限”属性中返回用户组权限。 它具有以下属性： 权限 - (整数) 访问级别到主机组; id - (string) 主机组的ID。 有关主机组的访问级别列表，请参阅用户组页面。

属性类	说明	
limitSelects	integer	限制子选择返回的记录数。
sortfield	string/array	按照给定的属性对结果进行排序。 可能的值为：`usrgrpid`，`name`。
countOutput	flag	参考文献中详细描述了所有“获得”方法的常用参数。 .
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	

属性类	说明
preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
status	integer	Return only user groups with the given status. Refer to the user group page for a list of supported statuses.
userids	string/array	Return only user groups that contain the given users.
usrgrpids	string/array	Return only user groups with the given IDs.
with_gui_access	integer	Return only user groups with the given frontend authentication method. Refer to the user group page for a list of supported methods.
selectTagFilters	query	Return user group tag based permissions in the `tag_filters` property. It has the following properties: `groupid` - (string) ID of the host group; `tag` - (string) tag name; `value` - (string) tag value.
selectUsers	query	Return the users from the user group in the `users` property.
selectRights	query	Return user group rights in the `rights` property. It has the following properties: `permission` - (integer) access level to the host group; `id` - (string) ID of the host group. Refer to the user group page for a list of access levels to host groups.
limitSelects	integer	Limits the number of records returned by subselects.
sortfield	string/array	Sort the result by the given properties. Possible values are: `usrgrpid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回：

- 一组对象；
- 如果已经使用 “countOutput” 参数，则检索到的对象的计数。

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

示例

检索已启用的用户组

检索所有已启用的用户组。

Examples

Retrieving enabled user groups

Retrieve all enabled user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.get",
    "params": {
        "output": "extend",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "usrgrpid": "7",
            "name": "Zabbix administrators",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1"
        },
        {
            "usrgrpid": "8",
            "name": "Guests",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0"
        },
        {
            "usrgrpid": "11",
            "name": "Enabled debug mode",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1"
        },
        {
            "usrgrpid": "12",
            "name": "No access to the frontend",
            "gui_access": "2",
            "users_status": "0",
```

```
            "debug_mode": "0"
        },
        {
            "usrgrpid": "14",
            "name": "Read only",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0"
        },
        {
            "usrgrpid": "18",
            "name": "Deny",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0"
        }
    ],
    "id": 1
}
```


参见

- User

来源

CUserGroup::get() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.update

说明

`object usergroup.update(object/array userGroups)`

此方法允许更新现有的用户组。

Description

`object usergroup.update(object/array userGroups)`

This method allows to update existing user groups.

参数

`(object/array)` 要更新的用户组属性。

必须为每个用户组定义 “usrgrpid” 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

除了标准用户组属性之外, 该方法接受以下参数。

属性类	说明	
rights	object/array	更改分配给用户组的当前权限的权限。
tag_filters	array	基于标记的权限以分配给组。
userids	string/array	用户的 ID 替换组中的用户。

Parameters

`(object/array)` User group properties to be updated.

The `usrgrpid` property must be defined for each user group, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard user group properties, the method accepts the following parameters.

Parameter	Type	Description
rights	object/array	Permissions to replace the current permissions assigned to the user group.
tag_filters	array	Tag based permissions to assign to the group

Parameter	Type	Description
userids	string/array	IDs of the users to replace the users in the group.

返回值

(object) 返回包含 “usrgrpids” 属性下更新的用户组的 ID 的对象。

Return values

(object) Returns an object containing the IDs of the updated user groups under the usrgrpids property.

示例

禁用用户组

禁用用户组。

Examples

Disabling a user group

Disable a user group.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.update",
    "params": {
        "usrgrpid": "17",
        "users_status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17"
        ]
    },
    "id": 1
}
```

参考

- Permission

来源

CUserGroup::update() in *frontends/php/include/classes/api/services/CUserGroup.php*.

User macro [用户宏]

此类用于全局宏的使用。

对象引用：

- Global macro
- Host macro

可用的方法：

- usermacro.create - 创建新的主机宏
- usermacro.createglobal - 创建新的全局宏

- usermacro.delete - 删除主机宏
- usermacro.deleteglobal - 删除全局宏
- usermacro.get - 检索主机和全局宏
- usermacro.update - 更新主机宏
- usermacro.updateglobal - 更新全局宏

User macro This class is designed to work with host and global macros.

Object references:

- Global macro
- Host macro

Available methods:

- usermacro.create - creating new host macros
- usermacro.createglobal - creating new global macros
- usermacro.delete - deleting host macros
- usermacro.deleteglobal - deleting global macros
- usermacro.get - retrieving host and global macros
- usermacro.update - updating host macros
- usermacro.updateglobal - updating global macros

> 用户宏对象

以下对象与 “usermacro”API 直接相关。

全局宏

全局宏对象具有以下属性。

属性类	说明	
globalmacroid	string	*(readonly)* 全局宏的 ID。
macro (required)	string	宏字符串。
value (required)	string	宏的价值。

> User macro object

The following objects are directly related to the `usermacro` API.

Global macro

The global macro object has the following properties.

Property	Type	Description
globalmacroid	string	*(readonly)* ID of the global macro.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.

主机宏

主机宏对象定义主机或模板上可用的宏。它具有以下属性。

属性类	说明	
hostmacroid	string	*(readonly)* 主机宏的 ID。
hostid (required)	string	宏所属主机的 ID。

属性类	说明	
macro (required)	string	宏字符串。
value (required)	string	宏的值。

Host macro

The host macro object defines a macro available on a host or template. It has the following properties.

Property	Type	Description
hostmacroid	string	*(readonly)* ID of the host macro.
hostid (required)	string	ID of the host that the macro belongs to.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.

usermacro.create

说明

`object usermacro.create(object/array hostMacros)`

此方法允许创建新的主机宏。

Description

`object usermacro.create(object/array hostMacros)`

This method allows to create new host macros.

参数

`(object/array)` 要创建的主机宏。

该方法接受有标准主机宏属性的主机宏.

Parameters

`(object/array)` Host macros to create.

The method accepts host macros with the standard host macro properties.

返回值

`(object)` 返回包含 “hostMacroids” 属性下创建的主机宏的 ID 的对象。返回的 ID 的顺序与传递的主机宏的顺序相匹配。

Return values

`(object)` Returns an object containing the IDs of the created host macros under the `hostmacroids` property. The order of the returned IDs matches the order of the passed host macros.

示例

创建主机宏

在主机”10198” 创建主机宏”{$SNMP_COMMUNITY}” 值为”public”。

Creating a host macro

Creat a host macro ”{$SNMP_COMMUNITY}” with the value ”public” on host ”10198”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.create",
    "params": {
        "hostid": "10198",
```

```
        "macro": "{$SNMP_COMMUNITY}",
        "value": "public"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "11"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::create() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.createglobal

说明

`object usermacro.createglobal(object/array globalMacros)`

此方法允许创建新的全局宏。

Description

`object usermacro.createglobal(object/array globalMacros)`

This method allows to create new global macros.

参数

(`object/array`) 要创建的全局宏。

该方法接受具有标准全局宏属性. 的全局宏。

Parameters

(`object/array`) Global macros to create.

The method accepts global macros with the standard global macro properties.

返回值

(`object`) 返回包含 `globalmacroids` 属性下创建的全局宏的 ID 的对象。返回的 ID 的顺序与传递的全局宏的顺序相匹配。

Return values

(`object`) Returns an object containing the IDs of the created global macros under the `globalmacroids` property. The order of the returned IDs matches the order of the passed global macros.

示例

Examples

创建一个全局宏

创建一个宏"{$SNMP_COMMUNITY}" 值为"public".

Creating a global macro

Create a global macro "{$SNMP_COMMUNITY}" with value "public".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.createglobal",
    "params":  {
        "macro": "{$SNMP_COMMUNITY}",
        "value": "public"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "6"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::createGlobal() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.delete

说明

`object usermacro.delete(array hostMacroIds)`

Description

`object usermacro.delete(array hostMacroIds)`

This method allows to delete host macros.

参数

(`array`) 要删除的主机宏的 ID。

Parameters

(`array`) IDs of the host macros to delete.

返回值

(`object`) 返回一个包含 “hostMacs” 属性下删除的主机宏 ID 的对象。

Return values

(`object`) Returns an object containing the IDs of the deleted host macros under the `hostmacroids` property.

示例

Examples

删除多个主机宏

删除 2 个主机宏

Deleting multiple host macros

Delete two host macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.delete",
    "params": [
```

```
        "32",
        "11"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::delete() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.deleteglobal

说明

`object usermacro.deleteglobal(array globalMacroIds)` 此方法允许删除全局宏。

Description

`object usermacro.deleteglobal(array globalMacroIds)`

This method allows to delete global macros.

参数

(`array`) 要删除的全局宏的 ID。

Parameters

(`array`) IDs of the global macros to delete.

返回值

(`object`) 返回包含 “globalmacroids” 属性下删除的全局宏 ID 的对象。

Return values

(`object`) Returns an object containing the IDs of the deleted global macros under the `globalmacroids` property.

示例

Examples

删除多个全局宏

删除 2 个主机宏。

Deleting multiple global macros

Delete two global macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.deleteglobal",
    "params": [
        "32",
        "11"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::deleteGlobal() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.get

说明

`integer/array usermacro.get(object parameters)` 该方法允许根据给定的参数检索主机和全局宏。

Description

`integer/array usermacro.get(object parameters)`

The method allows to retrieve host and global macros according to the given parameters.

参数

`(object)` 定义所需输出的参数。该方法支持以下参数。

属性类	说明	
globalmacro	flag	返回全局宏而不是主机宏。
globalmacroids	string/array	仅返回具有给定 ID 的全局宏。
groupids	string/array	只返回属于主机的主机宏或来自给定主机组的模板。
hostids	string/array	仅返回属于给定主机的主机宏。
hostmacroids	string/array	只返回具有给定 ID 的主机宏。
templateids	string/array	只返回属于给定模板的主机宏。
selectGroups	query	在 `groups` 属性中返回主机宏所属的主机组。 仅在检索主机宏时使用。
selectHosts	query	在 `hosts` 属性中返回主机宏所属的主机。 仅在检索主机宏时使用。
selectTemplates	query	在 `template` 属性中返回主机宏所属的模板。 仅在检索主机宏时使用。
sortfield	string/array	按照给定的属性对结果进行排序。 可能的值：`macro`。
countOutput	boolean	这些参数对于所有的 “获取” 方法是常见的，在页参考评论 page. 中有详细描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
globalmacro	flag	Return global macros instead of host macros.
globalmacroids	string/array	Return only global macros with the given IDs.
groupids	string/array	Return only host macros that belong to hosts or templates from the given host groups.
hostids	string/array	Return only macros that belong to the given hosts or templates.
hostmacroids	string/array	Return only host macros with the given IDs.
selectGroups	query	Return host groups that the host macro belongs to in the `groups` property. Used only when retrieving host macros.
selectHosts	query	Return hosts that the host macro belongs to in the `hosts` property. Used only when retrieving host macros.
selectTemplates	query	Return templates that the host macro belongs to in the `templates` property. Used only when retrieving host macros.
sortfield	string/array	Sort the result by the given properties. Possible value: `macro`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回：

- 一组对象;
- 如果已经使用 “countOutput” 参数，则检索到的对象的计数。

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

示例

Examples

检索主机的主机宏

检索主机”10198” 定义的所有主机宏。

Retrieving host macros for a host

Retrieve all host macros defined for host ”10198”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "hostids": "10198"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostmacroid": "9",
            "hostid": "10198",
            "macro": "{$INTERFACE}",
            "value": "eth0"
        },
        {
            "hostmacroid": "11",
            "hostid": "10198",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public"
        }
    ],
    "id": 1
}
```

检索全局宏

检索所有全局宏。

Retrieving global macros

Retrieve all global macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "globalmacro": true
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "globalmacroid": "6",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public"
        }
    ],
    "id": 1
}
```

来源

CUserMacro::get() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.update

说明

`object usermacro.update(object/array hostMacros)`

此方法允许更新现有的主机宏。

Description

`object usermacro.update(object/array hostMacros)`

This method allows to update existing host macros.

参数

(`object/array`) 要更新的主机宏属性。

必须为每个主机宏定义 `hostmacroid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Parameters

(`object/array`) Host macro properties to be updated.

The `hostmacroid` property must be defined for each host macro, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

返回值

(`object`) 返回包含 `hostMacroids` 属性下更新的主机宏的 ID 的对象。

Return values

(`object`) Returns an object containing the IDs of the updated host macros under the `hostmacroids` property.

示例

Examples

更改主机宏的值

更改主机宏的值为”public”.

Changing the value of a host macro

Change the value of a host macro to ”public”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.update",
    "params": {
        "hostmacroid": "1",
        "value": "public"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::update() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.updateglobal

说明

`object usermacro.updateglobal(object/array globalMacros)`

此方法允许更新现有的全局宏。

Description

`object usermacro.updateglobal(object/array globalMacros)`

This method allows to update existing global macros.

参数

(`object/array`) 要更新的全局宏属性。

必须为每个全局宏定义 `globalmacroid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Parameters

(`object/array`) Global macro properties to be updated.

The `globalmacroid` property must be defined for each global macro, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

返回值

(`object`) 返回包含 “globalmacroids” 属性下更新的全局宏的 ID 的对象。

Return values

(`object`) Returns an object containing the IDs of the updated global macros under the `globalmacroids` property.

示例

Examples

更改全局宏的值

将全局宏的值更改为 “public”。

Changing the value of a global macro

Change the value of a global macro to ”public”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.updateglobal",
    "params": {
        "globalmacroid": "1",
        "value": "public"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::updateGlobal() in *frontends/php/include/classes/api/services/CUserMacro.php*.

User [用户]

该类用于用户的使用.

对象引用:

- User

可用的方法:

- user.create - 创建新用户
- user.delete - 删除用户
- user.get - 检索用户
- user.login - 登录到 API
- user.logout - 退出 API
- user.update - 更新用户

> 用户对象

以下对象与 `user` API 直接相关.

> User object

The following objects are directly related to the `user` API.

用户

用户对象具有以下属性。

属性类	说明	
userid	string	*(readonly)* 用户的ID。
alias *(required)*	string	用户别名。
attempt_clock	timestamp	*(readonly)* 最近一次登录失败的时间。

属性类	说明	
attempt_failed	integer	*(readonly)* 最近失败的登录尝试次数。
attempt_ip	string	*(readonly)* 最近一次失败的登录来源IP地址。
autologin	integer	允许自动登录。 可能的值: 0 - *(default)* 禁止自动登录; 1 - 允许自动登录。

属性类	说明	
autologout	string	会话过期时间。接受具有后缀的秒或时间单位。如果设置为0s，用户登录会话永远不会过期。 默认：15m.
lang	string	用户默认语言代码 默认：en_GB。
name	string	用户名.

属性类	说明
refresh	string 自动刷新时间间隔. 接受具有后缀的秒或时间单位。 默认: 30s.
rows_per_page	integer 每页显示的对象行数。 Default: 50.
surname	string 姓。
theme	string 用户的主题。 可能的值: `default` - *(default)* system default; `blue-theme` - Blue; `dark-theme` - Dark.

属性类	说明	
type	integer	用户类型。 Possible values: 1 - *(default)* Zabbix user; 2 - Zabbix admin; 3 - Zabbix super admin.
url	string	在登录后将用户重定向到页面的URL。

User

The user object has the following properties.

Property	Type	Description
userid	string	*(readonly)* ID of the user.
alias *(required)*	string	User alias.
attempt_clock	timestamp	*(readonly)* Time of the last unsuccessful login attempt.
attempt_failed	integer	*(readonly)* Recent failed login attempt count.
attempt_ip	string	*(readonly)* IP address from where the last unsuccessful login attempt came from.
autologin	integer	Whether to enable auto-login. Possible values: 0 - *(default)* auto-login disabled; 1 - auto-login enabled.

Property	Type	Description
autologout	string	User session life time. Accepts seconds and time unit with suffix. If set to 0s, the session will never expire. Default: 15m.
lang	string	Language code of the user's language. Default: `en_GB`.
name	string	Name of the user.
refresh	string	Automatic refresh period. Accepts seconds and time unit with suffix. Default: 30s.
rows_per_page	integer	Amount of object rows to show per page. Default: 50.
surname	string	Surname of the user.
theme	string	User's theme. Possible values: `default` - *(default)* system default; `blue-theme` - Blue; `dark-theme` - Dark.
type	integer	Type of the user. Possible values: 1 - *(default)* Zabbix user; 2 - Zabbix admin; 3 - Zabbix super admin.
url	string	URL of the page to redirect the user to after logging in.

媒体

媒体对象具有以下属性。

属性类	说明	
mediatypeid (required)	string	用于媒体的媒体类型 ID

属性类	说明	
sendto (required)	string/array	地址，用户名或者接收方的其他标识符。 如果类型是Media type电子邮件，值被设置为数组。其他类型Media types，值被设置为字符串。

属性类	说明	
active	integer	是否启用媒体。 可能的值: 0 - *(default)* enabled; 1 - disabled.

属性类	说明	
severity	integer	触发发送通知告警级别。 Severities are stored in binary form with each bit representing the corresponding severity. For example, 12 equals 1100 in binary and means, that notifications will be sent from triggers with severities warning and average.

属性类	说明	
period	string	当通知可以作为time pe-riod发送或者用分号隔开用户宏。 Default: 1-7,00:00-24:00

Media

The media object has the following properties.

Property	Type	Description
mediatypeid (required)	string	ID of the media type used by the media.
sendto (required)	string/array	Address, user name or other identifier of the recipient. If type of Media type is e-mail, values are represented as array. For other types of Media types, value is represented as a string.
active	integer	Whether the media is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
severity	integer	Trigger severities to send notifications about. Severities are stored in binary form with each bit representing the corresponding severity. For example, 12 equals 1100 in binary and means, that notifications will be sent from triggers with severities warning and average. Refer to the trigger object page for a list of supported trigger severities. Default: 63
period	string	Time when the notifications can be sent as a time period or user macros separated by a semicolon. Default: 1-7,00:00-24:00

user.create

描述

`object user.create(object/array users)`

此方法允许创建新的用户。

Description

`object user.create(object/array users)`

This method allows to create new users.

Parameters

`(object/array)` 要创建的用户.

该方法接受有标准用户属性的用户。

属性类	说明	
passwd *(required)*	string	用户密码。
usrgrps (required)	array	用户添加到的组。 用户组必须有存在的 `usrgrpid` 属性定义。
user_medias	array	为用户创建媒体。

Parameters

`(object/array)` Users to create.

Additionally to the standard user properties, the method accepts the following parameters.

Parameter	Type	Description
passwd *(required)*	string	User's password.
usrgrps (required)	array	User groups to add the user to. The user groups must have the `usrgrpid` property defined.
user_medias	array	Medias to create for the user.

返回值

`(object)` 返回一个包含创建值的 ID 的对象映射 `userids` 属性。返回的 ID 的顺序与传递的用户的顺序相匹配。

Return values

`(object)` Returns an object containing the IDs of the created users under the `userids` property. The order of the returned IDs matches the order of the passed users.

示例

Examples

创建一个用户

创建一个新用户, 把用户加入用户组同时添加用户媒体。

Creating a user

Create a new user, add him to a user group and create a new media for him.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.create",
    "params": {
        "alias": "John",
```

```
        "passwd": "Doe123",
        "usrgrps": [
            {
                "usrgrpid": "7"
            }
        ],
        "user_medias": [
            {
                "mediatypeid": "1",
                "sendto": [
                    "support@company.com"
                ],
                "active": 0,
                "severity": 63,
                "period": "1-7,00:00-24:00"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
        ]
    },
    "id": 1
}
```

参考

- Media
- User group

来源

CUser::create() in *frontends/php/include/classes/api/services/CUser.php*.

user.delete

说明

`object user.delete(array users)`

此方法允许删除用户。

Description

`object user.delete(array users)`

This method allows to delete users.

Parameters

(`array`) 要删除用户 ID。

Parameters

(`array`) IDs of users to delete.

Return values

(`object`) 返回一个包含 `userids` 属性下删除用户 ID 的对象。

Return values

(object) Returns an object containing the IDs of the deleted users under the userids property.

示例

Examples

删除多个用户

删除 2 个用户。

Deleting multiple users

Delete two users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.delete",
    "params": [
        "1",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

来源

CUser::delete() in *frontends/php/include/classes/api/services/CUser.php*.

user.get

Description

integer/array user.get(object parameters) 此方法允许根据给定的参数检索用户。

Description

integer/array user.get(object parameters)

The method allows to retrieve users according to the given parameters.

Parameters

(object) 定义所需输出的参数。

该方法支持以下参数。

属性类	说明	
mediaids	string/array	只返回用户给定媒体。
mediatypeids	string/array	只返回用户给定媒体类型。
userids	string/array	只返回用户给定ID。
usrgrpids	string/array	只返回用户给定用户组ID。

属性类	说明
getAccess	flag 添加关于用户权限附加信息。 为每个用户添加以下属性: gui_access - *(integer)* 用户的前端认证方法。参考 gui_access 的属性关于用户组对象列出可能的值。 debug_mode - *(integer)* 表明是否

属性类	说明	
selectMedias	query	在 `medias` 属性返回用户使用的媒体。
selectMediatypes	query	在 `mediatypes` 属性返回用户使用的媒体类型。
selectUsrgrps	query	在 `usrgrps` 属性返回用户所属的组

属性类	说明	
sortfield	string/array	根据给定的属性对结果进行排序。 可能的值: `userid` and `alias`.
countOutput	boolean	这些参数对于所有的 `get` 方法是常见的，在reference commentary中有详细描述.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回:

- 一个对象数组;
- 检索对象的计数, 如果 countOutput 参数被使用。

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
mediaids	string/array	Return only users that use the given media.
mediatypeids	string/array	Return only users that use the given media types.
userids	string/array	Return only users with the given IDs.
usrgrpids	string/array	Return only users that belong to the given user groups.
getAccess	flag	Adds additional information about user permissions. Adds the following properties for each user: gui_access - *(integer)* user's frontend authentication method. Refer to the gui_access property of the user group object for a list of possible values. debug_mode - *(integer)* indicates whether debug is enabled for the user. Possible values: 0 - debug disabled, 1 - debug enabled. users_status - *(integer)* indicates whether the user is disabled. Possible values: 0 - user enabled, 1 - user disabled.
selectMedias	query	Return media used by the user in the medias property.
selectMediatypes	query	Return media types used by the user in the mediatypes property.
selectUsrgrps	query	Return user groups that the user belongs to in the usrgrps property.
sortfield	string/array	Sort the result by the given properties. Possible values are: userid and alias.
countOutput	boolean	These parameters being common for all get methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

示例

Examples

检索用户

Retrieving users

检索所有已配置的用户。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "alias": "Admin",
            "name": "Zabbix",
            "surname": "Administrator",
            "url": "",
            "autologin": "1",
            "autologout": "0s",
            "lang": "ru_RU",
            "refresh": "0s",
            "type": "3",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50"
        },
        {
            "userid": "2",
            "alias": "guest",
            "name": "Default2",
            "surname": "User",
            "url": "",
            "autologin": "0",
            "autologout": "15m",
            "lang": "en_GB",
            "refresh": "30s",
            "type": "1",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50"
        }
    ],
    "id": 1
}
```

参考

- 媒体
- 媒体类型
- 用户组

来源

CUser::get() in *frontends/php/include/classes/api/services/CUser.php*.

user.login

说明

`string/object user.login(object parameters)`

这个方法用户登录 API 和获取一个认证票据。

> **Warning:**
> 当使用这个方法的时候, 你必须使用user.logout方法，防止产生大量的开放会话记录。.

Description

`string/object user.login(object parameters)`

This method allows to log in to the API and generate an authentication token.

> **Warning:**
> When using this method, you also need to do user.logout to prevent the generation of a large number of open session records.

参数

<note important> 这种方法对于未经身份验证的用户是可用的，并且必须在 JSON-RPC 请求中没有 `auth` 数调用。:::

`(object)` 包含用户名和密码的参数。

该方法接受以下参数。

属性类	说明	
password (required)	string	用户密码。未使用的 HTTP 身份验证。
user (required)	string	用户名。
userData	flag	R 返回关于已认证用户的信息。

<note important> 当使用 HTTP 认证时，API 请求中的用户名必须与授权头中使用的名称相匹配。密码将不会被验证，并且可以省略。:::

Parameters

> **Attention:**
> This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

`(object)` Parameters containing the user name and password.

The method accepts the following parameters.

Parameter	Type	Description
password (required)	string	User password. Unused for HTTP authentication.
user (required)	string	User name.
userData	flag	Return information about the authenticated user.

> **Attention:**
> When using HTTP authentication, the user name in the API request must match the one used in the `Authorization` header. The password will not be validated and can be omitted.

返回值

`(string/object)` 如果使用 `userDat` 参数，则返回包含关于经过身份验证用户信息的对象。

关于standard user properties, 返回以下信息:

属性类	说明	
debug_mode	boolean	是否为用户启用了调试模式。
gui_access	integer	用户的身份验证方法到前端。 Refer to the gui_access property of the user group object for a list of possible values.
sessionid	string	身份验证令牌，必须在下列 API 请求中使用。
userip	string	用户的 IP 地址。

<note tip> 如果一个用户在一次或多次失败的尝试之后成功地进行了身份验证，该方法将返回 `attempt_clock`、尝试失败和尝试 `ip` 属性的当前值，然后重新设置它们。:::

如果不使用 `userData` 参数，该方法将返回身份验证令牌。

<note tip> 所生成的认证令牌必须存储，并在以下 JSON-RPC 请求的 `auth` 参数中使用。在使用 HTTP 认证时也需要它。:::

Return values

(`string/object`) If the `userData` parameter is used, returns an object containing information about the authenticated user.

Additionally to the standard user properties, the following information is returned:

Property	Type	Description
debug_mode	boolean	Whether debug mode is enabled for the user.
gui_access	integer	User's authentication method to the frontend. Refer to the `gui_access` property of the user group object for a list of possible values.
sessionid	string	Authentication token, which must be used in the following API requests.
userip	string	IP address of the user.

Note:
If a user has been successfully authenticated after one or more failed attempts, the method will return the current values for the `attempt_clock`, `attempt_failed` and `attempt_ip` properties and then reset them.

If the `userData` parameter is not used, the method returns an authentication token.

Note:
The generated authentication token should be remembered and used in the `auth` parameter of the following JSON-RPC requests. It is also required when using HTTP authentication.

示例

Examples

认证一个用户

Authenticating a user

认证一个用户

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

Requesting authenticated user's information

Authenticate and return additional information about the user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix",
        "userData": true
    },
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "alias": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "type": "3",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "debug_mode": true,
        "userip": "127.0.0.1",
        "sessionid": "5b56eee8be445e98f0bd42b435736e42",
        "gui_access": "0"
    },
    "id": 1
}
```


参考

- user.logout

来源

CUser::login() in *frontends/php/include/classes/api/services/CUser.php*.

user.logout

说明

`string/object user.logout(array)`

这个方法用于用户登出 API 与使当前认证令牌失效。

Description

`string/object user.logout(array)`

This method allows to log out of the API and invalidates the current authentication token.

参数

(`array`) 这个方法接受一个空数组。

Parameters

(`array`) The method accepts an empty array.

返回值

(`boolean`) 如果用户已成功注销，则返回 `true`。

Return values

(`boolean`) Returns `true` if the user has been logged out successfully.

示例

Examples

登出

Logging out

从 API 登出。

Log out from the API.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.logout",
    "params": [],
    "id": 1,
    "auth": "16a46baf181ef9602e1687f3110abf8a"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

参考

- user.login

来源

CUser::login() in *frontends/php/include/classes/api/services/CUser.php*.

user.update

Description

`object user.update(object/array users)`

这个方法允许更新存在的用户。

Description

`object user.update(object/array users)`

This method allows to update existing users.

Parameters

`(object/array)` 需要更新的用户属性。

必须为每个用户定义 `userid` 属性，所有其他属性都是可选的。只有传递的属性将被更新，其他所有的属性将保持不变。

此外, standard user properties, 该方法接受以下参数。

属性类	说明	
passwd	string	用户的密码。
usrgrps	array	用户组来替换现有的用户组。 用户组 ID 必须是存在的 `usrgrpid`。
user_medias	array	新的媒体用于替换旧的。

Parameters

`(object/array)` User properties to be updated.

The `userid` property must be defined for each user, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard user properties, the method accepts the following parameters.

Parameter	Type	Description
passwd	string	User's password.

Parameter	Type	Description
usrgrps	array	User groups to replace existing user groups. The user groups must have the `usrgrpid` property defined.
user_medias	array	Medias to replace existing medias.

返回值

（object）在 `userids` 属性下，返回包含更新用户 id 对象。

Return values

（object）Returns an object containing the IDs of the updated users under the `userids` property.

示例

Examples

Renaming a user

把一个用户重命名为 John Doe.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "1",
        "name": "John",
        "surname": "Doe"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CUser::update() in *frontends/php/include/classes/api/services/CUser.php*.

Value map [值映射**]**

此类用于值映射的使用

对象引用：

- Value map

可用的方法：

- valuemap.create - 创建新的 value maps
- valuemap.delete - 删除 value maps
- valuemap.get - 检索 value maps
- valuemap.update - 更新 value maps

*

Value map This class is designed to work with value maps.

Object references:

- Value map

Available methods:

- valuemap.create - creating new value maps
- valuemap.delete - deleting value maps
- valuemap.get - retrieving value maps
- valuemap.update - updating value maps

> 值映射对象

以下对象与 `VALUEAPI` 直接相关。

> Value map object

The following objects are directly related to the `valuemap` API.

值映射

值映射对象具有以下属性。

属性类	说明	
valuemapid	string	*(readonly)* 值映射的 ID
name (required)	string	值映射的名称。
mappings (required)	array	值映射当前映射值。值映射对象object#value_mappings 细节描述如下.

Value map

The value map object has the following properties.

Property	Type	Description
valuemapid	string	*(readonly)* ID of the value map.
name (required)	string	Name of the value map.
mappings (required)	array	Value mappings for current value map. The mapping object is described in detail below.

价值映射

值映射对象定义值映射的映射值。它具有以下属性。

属性类	说明	
value (required)	string	原值。
newvalue (required)	string	原始值映射到的值。

Value mappings

The value mappings object defines value mappings of the value map. It has the following properties.

Property	Type	Description
value (required)	string	Original value.
newvalue (required)	string	Value to which the original value is mapped to.

valuemap.create

说明

`object valuemap.create(object/array valuemaps)`

此方法允许创建新的值映射。

Description

`object valuemap.create(object/array valuemaps)`

This method allows to create new value maps.

参数

`(object/array)` 要创建的值映射。

该方法接受有标准值映射属性的值映射。

Parameters

`(object/array)` Value maps to create.

The method accepts value maps with the standard value map properties.

返回值

`(object)` 返回一个包含创建值的 ID 的对象映射 `valemapids` 属性。返回的 ID 的顺序与传递的值映射的顺序相匹配。

Return values

`(object)` Returns an object containing the IDs of the created value maps the `valuemapids` property. The order of the returned IDs matches the order of the passed value maps.

示例

Examples

创建一个值映射

使用两个映射创建一个值映射。

Creating a value map

Create one value map with two mappings.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.create",
    "params": {
        "name": "Service state",
        "mappings": [
            {
                "value": "0",
                "newvalue": "Down"
            },
            {
                "value": "1",
                "newvalue": "Up"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

Source

CValueMap::create() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.delete

说明

`object valuemap.delete(array valuemapids)`

此方法允许删除值映射。

Description

`object valuemap.delete(array valuemapids)`

This method allows to delete value maps.

参数

(`array`) 要被删除的映射的 ID。

Parameters

(`array`) IDs of the value maps to delete.

返回值

(`object`) 返回一个对象，该对象包含 “VALUE” 属性下的已删除值映射的 ID。

Return values

(`object`) Returns an object containing the IDs of the deleted value maps under the `valuemapids` property.

示例

Examples

删除多个值映射

删除 2 个值映射。

Deleting multiple value maps

Delete two value maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

来源

CValueMap::delete() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.get

说明

`integer/array valuemap.get(object parameters)`

该方法允许根据给定的参数来检索值映射。

Description

`integer/array valuemap.get(object parameters)`

The method allows to retrieve value maps according to the given parameters.

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

属性类	说明	
valuemapids	string/array	只返回具有给定 ID 的值映射。
selectMappings	query	在 “映射” 属性中返回当前值映射的值映射。
sortfield	string/array	按照给定的属性对结果进行排序。 可能的值为：`valuemapid`，`name`。
countOutput	flag	这些参数对于所有的 “get” 方法是常见的，在参考评论中有详细描述.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

*

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
valuemapids	string/array	Return only value maps with the given IDs.
selectMappings	query	Return the value mappings for current value map in the `mappings` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `valuemapid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

示例

Examples

Retrieving value maps

Retrieve all configured value maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "valuemapid": "4",
            "name": "APC Battery Replacement Status"
        },
        {
            "valuemapid": "5",
            "name": "APC Battery Status"
        },
        {
            "valuemapid": "7",
            "name": "Dell Open Manage System Status"
```

```
        }
    ],
    "id": 1
}
```

Retrieve one value map with its mappings.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend",
        "selectMappings": "extend",
        "valuemapids": ["4"]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "valuemapid": "4",
            "name": "APC Battery Replacement Status",
            "mappings": [
                {
                    "value": "1",
                    "newvalue": "unknown"
                },
                {
                    "value": "2",
                    "newvalue": "notInstalled"
                },
                {
                    "value": "3",
                    "newvalue": "ok"
                },
                {
                    "value": "4",
                    "newvalue": "failed"
                },
                {
                    "value": "5",
                    "newvalue": "highTemperature"
                },
                {
                    "value": "6",
                    "newvalue": "replaceImmediately"
                },
                {
                    "value": "7",
                    "newvalue": "lowCapacity"
                }
            ]
        }
    ],
    "id": 1
}
```

来源

Source

CValueMap::get() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.update

说明

`object valuemap.update(object/array valuemaps)`

该方法允许更新现有的值映射。

Description

`object valuemap.update(object/array valuemaps)`

This method allows to update existing value maps.

参数

(`object/array`) 要更新的值映射特性。

必须为每个值映射定义 `valuemapid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Parameters

(`object/array`) Value map properties to be updated.

The `valuemapid` property must be defined for each value map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

返回值

(`object`) 返回一个对象，它包含 `valuemapids` 属性下更新的值映射的 ID。

Return values

(`object`) Returns an object containing the IDs of the updated value maps under the `valuemapids` property.

示例

Examples

更改值映射名称

将值映射名称更改为 “设备状态”

Changing value map name

Change value map name to ”Device status”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "name": "Device status"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "2"
        ]
    },
    "id": 1
}
```

Changing mappings for one value map.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "mappings": [
            {
                "value": "0",
                "newvalue": "Online"
            },
            {
                "value": "1",
                "newvalue": "Offline"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "2"
        ]
    },
    "id": 1
}
```

来源

Source

CValueMap::update() in *frontends/php/include/classes/api/services/CValueMap.php*.

Web scenario [Web 场景]

此类用于 Web 场景的使用。

对象引用：

- Web scenario
- Scenario step

可用的方法：

- httptest.create - 创建新的 Web 场景
- httptest.delete - 删除 Web 场景
- httptest.get - 获取 Web 场景
- httptest.update - 更新 Web 场景

Web scenario This class is designed to work with web scenarios.

Object references:

- Web scenario
- Scenario step

Available methods:

- httptest.create - creating new web scenarios
- httptest.delete - deleting web scenarios
- httptest.get - retrieving web scenarios
- httptest.update - updating web scenarios

> Web 场景对象

以下对象与 `webcheck`API 直接相关。

> Web scenario object

The following objects are directly related to the `webcheck` API.

Web 场景

Web 场景对象具有以下属性。

属性类	说明	
httptestid	string	*(readonly)* Web 场景的 ID
hostid (required)	string	Web 场景所属主机的 ID。
name (required)	string	Web 场景的名称
agent	string	将由 Web 场景使用的用户代理字符串。 默认: Zabbix

属性类	说明	
applicationid	string	Web场景所属应用程序的ID。
authentication	integer	将由Web场景使用的身份验证方法。 可能的值： 0 - (默认) 无; 1 - 基本的HTTP认证; 2 - NTLM身份验证

属性类	说明	
delay	string	Web场景的执行间隔。接受秒，时间单位后缀和用户宏。 默认: 1m.
headers	string	执行请求时将发送的HTTP标题。

属性类	说明	
http_password	string	用于认证的密码。 对于具有基本HTTP或NTLM身份验证的Web场景是必需的.
http_proxy	string	将由Web场景使用的代理 *http://[username*

属性类	说明	
http_user	string	用于认证的用户名 对于具有基本HTTP或NTLM身份验证的Web场景，必需。
nextcheck	timestamp	*(readonly)* 下一个Web场景执行的时间。

属性类	说明	
retries	integer	Web场景在失败之前尝试执行每个步骤的次数。 默认: 1.
ssl_cert_file	string	用于客户端身份验证的SSL证书文件的名称(必须为PEM格式)。

属性类	说明	
ssl_key_file	string	用于客户端认证的SSL私钥文件的名称(必须为PEM格式)。
ssl_key_password	string	SSL私钥密码。
status	integer	是否启用了Web方案。 可能的值： 0 - (默认)启用; 1 - 禁用.
templateid	string	*(readonly)* 父模板Web方案的ID。

属性类	说明	
variables	string	Web场景变量。
verify_host	integer	验证SSL证书中指定的主机名是否与场景中使用的主机名相匹配。 可能的值： 0 - (默认)跳过主机验证; 1 - 验证主机。

属性类	说明	
verify_peer	integer	是否验证Web服务器的SSL证书。\\可能的值：0 - *(默认)* 跳过对等验证; 1 - 验证对等

Web scenario

The web scenario object has the following properties.

Property	Type	Description
httptestid	string	*(readonly)* ID of the web scenario.
hostid (required)	string	ID of the host that the web scenario belongs to.
name (required)	string	Name of the web scenario.
agent	string	User agent string that will be used by the web scenario. Default: Zabbix
applicationid	string	ID of the application that the web scenario belongs to.
authentication	integer	Authentication method that will be used by the web scenario. Possible values: 0 - *(default)* none; 1 - basic HTTP authentication; 2 - NTLM authentication.
delay	string	Execution interval of the web scenario. Accepts seconds, time unit with suffix and user macro. Default: 1m.
headers	string *(deprecated)* array of HTTP fields	HTTP headers that will be sent when performing a request.

Property	Type	Description
http_password	string	Password used for authentication. Required for web scenarios with basic HTTP or NTLM authentication.
http_proxy	string	Proxy that will be used by the web scenario given as *http://[username[:password]@]proxy.example.com[:port]*.
http_user	string	User name used for authentication. Required for web scenarios with basic HTTP or NTLM authentication.
nextcheck	timestamp	*(readonly)* Time of the next web scenario execution.
retries	integer	Number of times a web scenario will try to execute each step before failing. Default: 1.
ssl_cert_file	string	Name of the SSL certificate file used for client authentication (must be in PEM format).
ssl_key_file	string	Name of the SSL private key file used for client authentication (must be in PEM format).
ssl_key_password	string	SSL private key password.
status	integer	Whether the web scenario is enabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.
templateid	string	*(readonly)* ID of the parent template web scenario.
variables	string *(deprecated)* array of HTTP fields	Web scenario variables.
verify_host	integer	Whether to verify that the host name specified in the SSL certificate matches the one used in the scenario. Possible values are: 0 - *(default)* skip host verification; 1 - verify host.
verify_peer	integer	Whether to verify the SSL certificate of the web server. Possible values are: 0 - *(default)* skip peer verification; 1 - verify peer.

场景步骤

场景步骤对象定义特定的 Web 场景检查。它具有以下属性。

属性类	说明	
httpstepid	string	*(readonly)* 情景步骤的ID
name (required)	string	场景步骤的名称。

属性类	说明	
no (required)	integer	Web场景中步骤的序列号。
url (required)	string	要检查的URL。
follow_redirects	integer	是否遵循HTTP重定向 可能的值： 0 - 不要重新导向; 1 - *(default)* 遵循重定向

属性类	说明	
headers	string *(deprecated)* array of HTTP fields	执行请求时将发送的HTTP headers。 场景步骤headers将覆盖Web场景指定的HTTP headers。
httptestid	string	*(readonly)* 该步骤所属的Web方案的ID。

属性类	说明	
posts	string array of HTTP fields	HTTP POST 字符串(原始 POST 数据)或者一个HTTP 字段数组(来自字段数据)。
required	string	必须在响应中存在的文本。

属性类	说明	
retrieve_mode	integer	方案步骤必须检索的HTTP响应的一部分。\\可能的值：0 - *(default)* 仅有文体; 1 - 仅有标题.
status_codes	string	所需HTTP状态代码的范围用逗号分隔。

属性类	说明	
timeout	string	请求超时(秒)。接受秒数，带后缀的时间单位和用户宏。 默认: 15s.
variables	string *(deprecated)* array of HTTP 字段	场景步骤变量。
query_fields	array of HTTP 字段 查	字段 - 在执行请求时将添加到 URL HTTP 字段

<note important> 对于 Web 场景和 Web 场景步骤对象的 `headers` 和 `variables` 字段，都允许使用HTTP 字段类型的字符串和数组。不推荐使用 `headers` 和 `variables` 的字符串数据类型，将来的版本将删除它们。:::

Scenario step

The scenario step object defines a specific web scenario check. It has the following properties.

Property	Type	Description
httpstepid	string	*(readonly)* ID of the scenario step.
name (required)	string	Name of the scenario step.
no (required)	integer	Sequence number of the step in a web scenario.

Property	Type	Description
url (required)	string	URL to be checked.
follow_redirects	integer	Whether to follow HTTP redirects. Possible values are: 0 - don't follow redirects; 1 - *(default)* follow redirects.
headers	string *(deprecated)* array of HTTP fields	HTTP headers that will be sent when performing a request. Scenario step headers will overwrite headers specified for the web scenario.
httptestid	string	*(readonly)* ID of the web scenario that the step belongs to.
posts	string array of HTTP fields	HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
required	string	Text that must be present in the response.
retrieve_mode	integer	Part of the HTTP response that the scenario step must retrieve. Possible values are: 0 - *(default)* only body; 1 - only headers.
status_codes	string	Ranges of required HTTP status codes separated by commas.
timeout	string	Request timeout in seconds. Accepts seconds, time unit with suffix and user macro. Default: 15s.
variables	string *(deprecated)* array of HTTP fields	Scenario step variables.
query_fields	array of HTTP fields	Query fields - array of HTTP fields that will be added to URL when performing a request

Attention:
Both string and array of HTTP fields types are allowed for `headers` and `variables` fields of both web scenario and web scenario step object.
String data type for `headers` and `variables` is deprecated and will be removed in future versions.

HTTP 字段

HTTP 字段对象定义名称和值，用于指定查询字段数据的变量，HTTP 标头，POST 表单字段数据。它具有以下属性。

属性类	说明	
name (required)	string	header / variable / POST 或者 GET 字段的名称。
value (required)	string	header / variable / POST 或者 GET 字段的值。

HTTP field

The HTTP field object defines a name and value that is used to specify variable, HTTP header, POST form field data of query field data. It has the following properties.

Property	Type	Description
name (required)	string	Name of header / variable / POST or GET field.
value (required)	string	Value of header / variable / POST or GET field.

httptest.create

说明

`object httptest.create(object/array webScenarios)`

此方法允许创建新的 Web 场景。

> **Note:**
> 创建 Web 场景将自动创建一组web 监控项.

Description

`object httptest.create(object/array webScenarios)`

This method allows to create new web scenarios.

> **Note:**
> Creating a web scenario will automatically create a set of web monitoring items.

参数

`(object/array)` 要创建的 Web 场景。

除了标准 Web 场景属性之外, 该方法接受以下参数

参数类	说明	
steps (required)	array	Web 方案步骤。

Parameters

`(object/array)` Web scenarios to create.

Additionally to the standard web scenario properties, the method accepts the following parameters.

Parameter	Type	Description
steps (required)	array	Web scenario steps.

返回值

`(object)` 返回一个包含 “httptestids” 属性下创建的 Web 场景的 ID 的对象。返回的 ID 的顺序与传递的 Web 方案的顺序相匹配。

Return values

`(object)` Returns an object containing the IDs of the created web scenarios under the `httptestids` property. The order of the returned IDs matches the order of the passed web scenarios.

示例

Examples

创建 Web 场景

创建一个 Web 场景来监视公司主页。该方案将有两个步骤，以检查主页和 “关于” 页面，并确保它们返回 HTTP 状态代码 200。

Creating a web scenario

Create a web scenario to monitor the company home page. The scenario will have two steps, to check the home page and the "About" page and make sure they return the HTTP status code 200.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.create",
    "params": {
        "name": "Homepage check",
```

```
        "hostid": "10085",
        "steps": [
            {
                "name": "Homepage",
                "url": "http://mycompany.com",
                "status_codes": "200",
                "no": 1
            },
            {
                "name": "Homepage / About",
                "url": "http://mycompany.com/about",
                "status_codes": "200",
                "no": 2
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ]
    },
    "id": 1
}
```

参见

See also

- Scenario step

来源

Source

CHttpTest::create() in *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.delete

说明

`object httptest.delete(array webScenarioIds)`

此方法允许删除 Web 场景。

Description

`object httptest.delete(array webScenarioIds)`

This method allows to delete web scenarios.

参数

(`array`) 要删除的网络场景的 ID。

Parameters

(`array`) IDs of the web scenarios to delete.

返回值

(`object`) 返回包含 `httptestids` 属性下删除的 Web 方案的 ID 的对象。

Return values

(object) Returns an object containing the IDs of the deleted web scenarios under the `httptestids` property.

示例

删除多个 Web 场景

删除 2 个 Web 场景

Examples

Deleting multiple web scenarios

Delete two web scenarios.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "2",
            "3"
        ]
    },
    "id": 1
}
```

来源

Source

CHttpTest::delete() in *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.get

说明

`integer/array httptest.get(object parameters)`

该方法允许根据给定的参数检索 Web 场景。

Description

`integer/array httptest.get(object parameters)`

The method allows to retrieve web scenarios according to the given parameters.

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数类	描述	
applicationids	string/array	仅返回属于给定应用程序的 Web 场景。
groupids	string/array	仅返回属于给定主机组的 Web 方案。
hostids	string/array	仅返回属于给定主机的 Web 场景。
httptestids	string/array	只返回具有给定 ID 的 Web 场景。
inherited	boolean	如果设置为 “true”，只返回从模板继承的 Web 场景。

参数类	描述	
monitored	boolean	如果设置为 “true”，则只返回属于受监视主机的启用的 Web 场景。
templated	boolean	如果设置为 “true”，则只返回属于模板的 Web 场景。
templateids	string/array	仅返回属于给定模板的 Web 场景
expandName	flag	以 Web 方案的名称展开宏。
expandStepName	flag	在方案步骤的名称中展开宏。
selectHosts	query	将网站场景所属的主机作为 “hosts” 属性中的数组返回.
selectSteps	query	在 `steps` 属性中返回 Web 方案步骤。
sortfield	string/array	按照给定的属性对结果进行排序。 可能的值为：`httptestid` 和 `name`。
countOutput	flag	这些参数对于所有的 “get” 方法是常见的，在参考. 中有详细描述
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
applicationids	string/array	Return only web scenarios that belong to the given applications.
groupids	string/array	Return only web scenarios that belong to the given host groups.
hostids	string/array	Return only web scenarios that belong to the given hosts.
httptestids	string/array	Return only web scenarios with the given IDs.
inherited	boolean	If set to `true` return only web scenarios inherited from a template.
monitored	boolean	If set to `true` return only enabled web scenarios that belong to monitored hosts.
templated	boolean	If set to `true` return only web scenarios that belong to templates.
templateids	string/array	Return only web scenarios that belong to the given templates.
expandName	flag	Expand macros in the name of the web scenario.
expandStepName	flag	Expand macros in the names of scenario steps.
selectHosts	query	Return the host that the web scenario belongs to as an array in the `hosts` property.
selectSteps	query	Return web scenario steps in the `steps` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `httptestid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	

Parameter	Type	Description
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回：

- 一组对象;
- 如果已经使用 “countOutput” 参数，则检索到的对象的计数。

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

示例

检索网络场景

Examples

Retrieving a web scenario

Retrieve all data about web scenario ”4”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.get",
    "params": {
        "output": "extend",
        "selectSteps": "extend",
        "httptestids": "9"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "httptestid": "9",
            "name": "Homepage check",
            "applicationid": "0",
            "nextcheck": "0",
            "delay": "1m",
            "status": "0",
            "variables": [],
            "agent": "Zabbix",
            "authentication": "0",
            "http_user": "",
            "http_password": "",
            "hostid": "10084",
            "templateid": "0",
            "http_proxy": "",
            "retries": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
```

```
            "verify_peer": "0",
            "verify_host": "0",
            "headers": [],
            "steps": [
                {
                    "httpstepid": "36",
                    "httptestid": "9",
                    "name": "Homepage",
                    "no": "1",
                    "url": "http://mycompany.com",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": [
                        {
                            "name":"{var}",
                            "value":"12"
                        }
                    ],
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": [],
                    "query_fields": []
                },
                {
                    "httpstepid": "37",
                    "httptestid": "9",
                    "name": "Homepage / About",
                    "no": "2",
                    "url": "http://mycompany.com/about",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": [],
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": [],
                    "query_fields": []
                }
            ]
        }
    ],
    "id": 1
}
```


参考

- Host
- Scenario step

来源

CHttpTest::get() in *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.update

Description

`object httptest.update(object/array webScenarios)` 此方法允许更新现有的 Web 场景。

Description

`object httptest.update(object/array webScenarios)`

This method allows to update existing web scenarios.

参数

(object/array) 要更新的 Web 场景属性。

必须为每个 Web 场景定义 `httptestid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变除了标准 Web 场景属性外, 该方法接受以下参数。

参数类	说明	
steps	array	用来替代现有的步骤的方案步骤。

Parameters

(object/array) Web scenario properties to be updated.

The `httptestid` property must be defined for each web scenario, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard web scenario properties, the method accepts the following parameters.

Parameter	Type	Description
steps	array	Scenario steps to replace existing steps.

返回值

Return values

(object) Returns an object containing the IDs of the updated web scenarios under the `httptestid` property.

Examples

Enabling a web scenario

Enable a web scenario, that is, set its status to "0".

示例

启用 Web 方案

启用 Web 方案，即将其状态设置为 "0"。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.update",
    "params": {
        "httptestid": "5",
        "status": 0
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ]
    },
    "id": 1
}
```

参考

See also

- Scenario step

来源

CHttpTest::update() in *frontends/php/include/classes/api/services/CHttpTest.php*.

主机

这个类是设计用于处理主机

对象引用:

- Host
- Host inventory

相关方法:

- host.create - 创建新的主机
- host.delete - 删除主机
- host.get - 获取主机信息
- host.massadd - 给主机添加相关对象
- host.massremove - 删除主机相关对象
- host.massupdate - 替换或移除主机相关对象
- host.update - 更新主机

> 主机对象

下列是与主机相关的对象

主机

主机对象具有以下属性:

属性类	描述	
hostid	字符串 *(	读)* 主机的ID.
host (必选)	字符串主机	正式名称.

属性类	描述	
available	整数 *	只读）*Zab-bix agent的可用性. 可能的值为:0 - 默认）未知1 - 可用;2 - 不可用.
description	文本主	说明.
disable_until	时间戳 *(	读)*不可用状态Agent的下一次轮训时间.
error	字符串 *(	读)*Agent不可用时的错误信息

属性类	描述	
errors_from	时间戳 *(	读)* Agent 不可用时的时间.
flags	整数 *	只读)* 主机的来源. 可能的值: 0 - 普通主机; 4 - 自动发现的主机.
inventory_mode	整数主	资产清单模式. 可能的值: -1 - 禁用; 0 - (默认) 手动; 1 - 自动.

属性类	描述
ipmi_authtype	整数 I MI 认证算法. 可能的值: -1 - (默认) 默认; 0 - 无; 1 - MD2; 2 - MD5 4 - straight; 5 - OEM; 6 - RMCP+.
ipmi_available	整数 * 只读)* IPMI agent 的可用性. 可能的值: 0 - (默认) 未知; 1 - 可用; 2 - 不可用.

属性类	描述	
ipmi_disable_until	时间戳 *(	读)* 不可用状态 IPMI agent 的下一次轮训时间.
ipmi_error	字符串 *(	读)* IPMI agent 不可用时的错误信息.
ipmi_errors_from	时间戳 *(	读)* IPMI agent 不可用时的时间.
ipmi_password	字符串 IP	I 密码.

属性类	描述	
ipmi_privilege	整数 I	MI 权限等级. 可能的值: 1 - 回调; 2 - (默认) 用户; 3 - 操作员; 4 - 管理员; 5 - OEM 原厂.
ipmi_username	整数 I	MI 用户名.
jmx_available	整数 *	只读)* JMX agent 的可用性. 可能的值: 0 - (默认) 未知; 1 - 可用; 2 - 不可用.

属性类	描述	
jmx_disable_until	时间戳 *(	读)* 不可用状态 JMX agent 的下一次轮训时间.
jmx_error	字符串 *(	读)* JMX agent 不可用时的错误信息.
jmx_errors_from	时间戳 *(	读)* JMX agent 不可用时的时间.
maintenance_from	时间戳 *(	读)* 有效维护的开始时间.

属性类	描述	
maintenance_status	整数 *	只读)*有效维护的状态. 可能的值: 0 - (默认)没有维护; 1 - 有效维护.
maintenance_type	整数 *	只读)*有效维护类型. 可能的值: 0 - (默认)维护期间搜集数据; 1 - 维护期间不搜集数据.

属性类	描述	
maintenanceid	字符串 *(	读)*目前对主机生效的维护的ID.
name	字符串主机	.
proxy_hostid	字符串 ID	默认: host 属性值. of the proxy that is used to mon-itor the host. 用于监控主机的Proxy服务器的hostid

属性类	描述	
snmp_available	整数 *	只读)* SNMP agent 的可用性. 可能的值: 0 - (默认) 未知; 1 - 可用; 2 - 不可用.
snmp_disable_until	时间戳 *(	读)* 不可用状态 SNMP agent 的下一次轮训时间.
snmp_error	字符串 *(	读)* SNMP agent 不可用时的错误信息.

属性类	描述	
snmp_errors_from	时间戳 *(	读)* SNMP agent 不可用时的时间.
status	整数主	的状态. 可能的值: 0 - (默认) 已监控的主机; 1 - 未监控的主机.
tls_connect	整数到	机的连接. // 可能的值: 1 - (默认) 没有加密; 2 - PSK; 4 - 证书.

属性类	描述	
tls_accept	整数来	主机的连接. // 可能的值: 1 - (默认)没有加密; 2 - PSK; 4 - 证书.
tls_issuer	字符串证书	行机构.
tls_subject	字符串证书	主题.
tls_psk_identity	字符串 PS	认证. 如果 `tls_connect` 或 `tls_accept` 启用了 PSK, 那么该选项是必选.

属性类	描述	
tls_psk	字符串 PS	至少需要32位16进制数字构成. 如果 tls_connect 或 tls_accept 启用了PSK, 那么该选项是必选.

主机资产清单

主机资产对象具有以下属性:

<note tip> 每一个属性拥有自己唯一的 ID 编号，用于将主机资产清单字段和事项关联在一起. :::

ID	属性类	描述
4	alias	字符串别名
11	asset_tag	字符串资产 签.
28	chassis	字符串机架
23	contact	字符串联系 .
32	contract_number	字符串联系 码.
47	date_hw_decomm	字符串硬件 汰时间.
46	date_hw_expiry	字符串硬件 保过期时间.
45	date_hw_install	字符串硬件 装时间.
44	date_hw_purchase	字符串硬件 买时间.
34	deployment_status	字符串部署 态.
14	hardware	字符串硬件 备.
15	hardware_full	字符串硬件 备详情.
39	host_netmask	字符串主机 网掩码.
38	host_networks	字符串主机 络.
40	host_router	字符串主机 由.
30	hw_arch	字符串硬件 构.
33	installer_name	字符串安装 员姓名.
24	location	字符串地点
25	location_lat	字符串纬度 置.
26	location_lon	字符串经度 置.
12	macaddress_a	字符串 MA 地址一.
13	macaddress_b	字符串 MA 地址二.
29	model	字符串型号

ID	属性类	描述
3	name	字符串名称
27	notes	字符串注释
41	oob_ip	字符串带外 P 地址.
42	oob_netmask	字符串带外 机子网掩码.
43	oob_router	字符串带外 由.
5	os	字符串操作 统名称.
6	os_full	字符串详尽 作系统名称.
7	os_short	字符串简短 作系统名称.
61	poc_1_cell	字符串主 P C 移动号码.
58	poc_1_email	字符串主 P C 邮箱.
57	poc_1_name	字符串主 P C 名称.
63	poc_1_notes	字符串主 P C 注释.
59	poc_1_phone_a	字符串主 P C 电话一.
60	poc_1_phone_b	字符串主 P C 电话二.
62	poc_1_screen	字符串主 P C 显示名.
68	poc_2_cell	字符串次 P C 移动号码.
65	poc_2_email	字符串次 P C 邮箱.
64	poc_2_name	字符串次 P C 名称.
70	poc_2_notes	字符串次 P C 注释.
66	poc_2_phone_a	字符串次 P C 电话一.
67	poc_2_phone_b	字符串次 P C 电话二.
69	poc_2_screen	字符串次 P C 显示名.
8	serialno_a	字符串序列 一.
9	serialno_b	字符串序列 二.
48	site_address_a	字符串所在 址一.
49	site_address_b	字符串所在 址二.
50	site_address_c	字符串所在 址三.
51	site_city	字符串所在 市.
53	site_country	字符串所在 家.
56	site_notes	字符串所在 注解.
55	site_rack	字符串所在 机柜位置.
52	site_state	字符串所在 说明.
54	site_zip	字符串所在 邮政编码.
16	software	字符串软件
18	software_app_a	字符串应用 件一.
19	software_app_b	字符串应用 件二.
20	software_app_c	字符串应用 件三.
21	software_app_d	字符串应用 件四.
22	software_app_e	字符串应用 件五.
17	software_full	字符串软件 情.
10	tag	字符串标签
1	type	字符串类型
2	type_full	字符串具体 型.
35	url_a	字符串网址 .
36	url_b	字符串网址 .
37	url_c	字符串网址 .
31	vendor	字符串供应 .

创建

描述

对象 `host.create(对象/数组 hosts)`

这个方法可以用来创建主机.

参数

(对象/数组) 要创建的主机.

另外，对于标准的主机属性，该方法接受下列参数

属性类	描述	
groups (必选)	对象/数组添加主	的主机组. 主机组必须已定义 `groupid` 属性.
interfaces (必选)	对象/数组为主机	建的接口.
templates	对象/数组链接到	机的模板. 模板必须已定义过 `templateid` 属性.
macros	对象/数组为主机	建的用户宏.
inventory	对象主	资产清单属性.

返回值

（对象）返回包含已创建主机 ID 的属性 hostid，返回 ID 的顺序与传入主机的顺序一致

示例

创建主机

Create a host called “Linux server” with an IP interface, add it to a group, link a template to it and set the MAC addresses in the host inventory.

创建一个具有 IP 接口的 “Linux Server” 主机，将其添加到主机组中，链接一个模板并且把 MAC 地址设置到主机资产清单里请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "50"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
        "inventory_mode": 0,
        "inventory": {
            "macaddress_a": "01234",
            "macaddress_b": "56768"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "107819"
```

```
        ]
    },
    "id": 1
}
```

参考

- Host group
- Template
- User macro
- Host interface
- Host inventory

来源

CHost::create() in *frontends/php/include/classes/api/services/CHost.php*.

删除

描述

`object host.delete(array hosts)`

该方法允许删除主机

参数

(array) 要删除的主机的 ID.

返回值

(object) 返回值包含已删除主机 ID 的 `hostid` 属性.

示例

删除多个主机

删除两个主机.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.delete",
    "params": [
        "13",
        "32"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "13",
            "32"
        ]
    },
    "id": 1
}
```

来源

CHost::delete() in *frontends/php/include/classes/api/services/CHost.php*.

批量创建

描述

`object host.massadd(object parameters)`

这个方法允许同时向所有给定的主机添加多个相关的对象

参数

`(object)` 参数包含要更新主机的 ID 和添加到所有主机的对象.

此方法接受如下参数：

参数类	描述	
hosts	对象/数组要更新	主机.
(必选)		主机必须已定义过 `hostid` 属性.
groups	对象/数组添加到	定主机的主机组.
		主机组必须已定义过 `groupid` 属性.
interfaces	对象/数组为指定	机创建主机接口.
macros	对象/数组为指定	机创建用户宏.
templates	对象/数组为指定	机关联模板.
		模板必须已定义过 `templateid` 属性.

返回值

`(object)` 在 `hostids` 属性下返回包含已更新主机 ID 的对象.

示例

添加宏

给两个主机添加两个宏

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "10160"
            },
            {
                "hostid": "10167"
            }
        ],
        "macros": [
            {
                "macro": "{$TEST1}",
                "value": "MACROTEST1"
            },
            {
                "macro": "{$TEST2}",
                "value": "MACROTEST2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "hostids": [
            "10160",
            "10167"
        ]
    },
    "id": 1
}
```

参考

- host.update
- Host group
- Template
- User macro
- Host interface

来源

CHost::massAdd() in *frontends/php/include/classes/api/services/CHost.php*.

批量删除

描述

`object host.massremove(object parameters)`

这个方法允许同时向所有给定的主机移除相关的对象

参数

`(object)` 参数包含要更主机的 ID 和应该移除的对象.

参数类	描述	
hostids (必选)	字符串/数组要更新的	机的 ID.
groupids	字符串/数组移除给定	机的主机组.
interfaces	对象/数组移除给	主机的主机接口. 主机接口对象必须已定义 `ip`, `dns` and `port` 属性.
macros	字符串/数组移除给定	机的用户宏.
templateids	字符串/数组移除给定	机的模板关联.
templateids_clear	字符串/数组移除给定	机的模板关联，并清空与该模板关联的数据.

返回值

`(object)` 在 `hostids` 属性中返回包含已更新主机 ID 对象.

示例

删除模板链接

从两个主机中删除一个模板链接并且删除所有模板实体

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.massremove",
    "params": {
        "hostids": ["69665", "69666"],
        "templateids_clear": "325"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "69665",
            "69666"
        ]
    },
    "id": 1
}
```

参考

- host.update
- User macro
- Host interface

来源

CHost::massRemove() in *frontends/php/include/classes/api/services/CHost.php*.

批量更新

描述

`object host.massupdate(object parameters)`

此方法允许同时对多个主机替换或移除相关对象和更新属性

参数

`(object)` 参数包含更新主机的 ID 和需要更新的属性.

另外，对于标准的主机属性，此方法可以接受如下参数:

参数类	描述	
hosts (必选)	对象/数组要更新	主机机必须已定义过 `hostid` 属性.

参数类	描述	
groups	对象/数组替换当	主机所属主机组. 主机组必须已定义过 groupid 属性.
interfaces	对象/数组在指定	机上替换当前主机接口.
inventory	对象主	资产清单属性. 使用参数 inventory 无法更新主机资产清单模式，用参数 inventory_mod 替换.

参数类	描述	
inventory_mode	整数主	资产清单群体模式. 参考host inventory object page获取支持的资产清单模式列表.
macros	对象/数组在指定	机中替换当前用户宏.
templates	对象/数组在指定	机中替换当前链接的模板. 模板必须已定义过templateid属性.

参数类	描述	
templates_clear	对象/数组移除给	主机的模板关联，并清空与该模板关联的数据. 模板必须已定义过 tem-plateid 属性.

返回值

(object) 在 hostids 属性中返回包含已更新主机 ID 对象.

示例

启用多个主机

启用两个主机，将 status 设置为 0

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.massupdate",
    "params": {
        "hosts": [
            {
                "hostid": "69665"
            },
            {
                "hostid": "69666"
            }
        ],
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "69665",
            "69666"
        ]
    },
    "id": 1
}
```

参考

- host.update
- host.massadd
- host.massremove
- Host group
- Template
- User macro
- Host interface

来源

CHost::massUpdate() in *frontends/php/include/classes/api/services/CHost.php*.

更新

描述

`object host.update(object/array hosts)`

该方法用来更新已存在的主机

参数

(`object/array`) 要更新的主机属性.

每个主机的 `hostid` 属性必须已定义过，其他属性为可选项. 只会更新指定的属性，其他属性保持不变. 另外，对于标准主机属性，此方法接受如下参数:

参数类	描述	
groups	对象/数组替换主	当前归属的组. 主机组的 `groupid` 必须已定义.
interfaces	对象/数组替换当	主机接口.
inventory	对象主	资产清单属性.
macros	对象/数组替换当	用户宏.
templates	对象/数组替换当	链接的模板. 模板没有传递仅删除链接. 模板必须已定义过 `templateid` 属性
templates_clear	对象/数组从主机	删除模板链接并清除. 模板必须已定义过 `templateid` 属性.

<note tip> 相对于 Zabbix 前端，当 `name` 和 `host` 一致，更新 `host` 的时候不会自动更新 `name`. 两个属性需要明确的更新. 这两个属性都需要显式地更新. :::

返回值

(`object`) 在 `hostids` 属性中返回包含已更新主机 ID 对象.

示例

启用主机

启用主机，将 status 设置为 0

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

删除模板链接

从主机中删除链接并清除两个模板.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "templates_clear": [
            {
                "templateid": "10124"
            },
            {
                "templateid": "10125"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

更新主机宏

用两个新的宏替换主机所有的宏.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
```

```
    "params": {
        "hostid": "10126",
        "macros": [
            {
                "macro": "{$PASS}",
                "value": "password"
            },
            {
                "macro": "{$DISC}",
                "value": "sda"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

更新主机资产清单

更改资产清单模式并添加地点

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "inventory_mode": 0,
        "inventory": {
            "location": "Latvia, Riga"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 2
}
```

参考

- host.massadd
- host.massupdate
- host.massremove
- Host group

- Template
- User macro
- Host interface
- Host inventory

来源

CHost::update() in *frontends/php/include/classes/api/services/CHost.php*.

获取

描述

`integer/array host.get(object parameters)`

此方法允许根据指定的参数获取主机.

参数

`(object)` 定义期望输出的参数.

该方法支持以下参数.

参数类	描述	
groupids	字符串/数组返回指定	机组的主机.
applicationids	字符串/数组返回指定	用集的主机.
dserviceids	字符串/数组返回与指	自动发现服务相关的主机.
graphids	字符串/数组返回包含	指定图表的主机.
hostids	字符串/数组返回指定	机ID的主机.
httptestids	字符串/数组返回指定	页监测的主机.

参数类	描述	
interfaceids	字符串/数组返回 指定	口 的 主 机.
itemids	字符串/数组返回 指定	控 项 的 主 机.
maintenanceids	字符串/数组返回 指定	护 的 主 机.
monitored_hosts	标识返	被 监 控 的 主 机.
proxy_hosts	标识返	代 理 服 务 器.
proxyids	字符串/数组返回 被代	服 务 器 监 控 的 主 机.
templated_hosts	标识返	主 机 和 模 板.
templateids	字符串/数组返回 使用	定 模 板 的 主 机.
triggerids	字符串/数组返回 指定	发 器 的 主 机.

参数类	描述
with_items	标识返含有监控项的主机. 覆盖with_monitor和with_simple_参数.
with_applications	标识返含有应用集的主机.
with_graphs	标识返含有图表的主机.
with_httptests	标识返含有web监测的主机. 覆盖with_monitor参数.
with_monitored_httptests	标识返含有启动网页监测的主机.

参数类	描述	
with_monitored_items	标识返	启用监控项的主机. 覆盖 with_simple_ 参数.
with_monitored_triggers	标识返	启用触发器的主机.所有在触发器中使用到的监控项必须也要启用.
with_simple_graph_items	标识返	含有数字类信息监控项的主机.

参数类	描述
with_triggers	标识返含有触发器的主机. 覆盖 with_monitor 参数.
withInventory	标识返含有资产清单数据的主机.
selectGroups	查询在 groups 属性中返回主机所属的主机组.| |selectAppli 查询在 |application 属性中返回来自主机的应用集. 支持 count.

参数类	描述
selectDiscoveries	查询在 discoveries 属性中返回来自主机的底层自动发现. 支持 count.||selectDisco 查询在 |discoveryRu 属性中返回创建主机的底层自动发现规则.

参数类	描述
selectGraphs	查询在 graphs 属性中返回来自主机的图表. 支持 count.||selectHostD 查询在 |hostDiscove 属性中返回主机自动发现对象. 主机自动发现对象将一个自动发现的主机和一个原型主机连接起来，

参数类	描述
selectHttpTests	查询在 httpTests 属性中返回主机的 web 场景. 支持 count.||selectInter 查询在 |interfaces 属性中返回主机的接口. 支持 count.

参数类	描述
selectInventory	查询在 inventory 属性中返回主机清单.||selectItems 查询在 |item 属性中返回主机监控项. 支持 count.
selectMacros	查询在 macros 属性中返回主机宏.||selectParent 查询在 |parentTempla 属性中返回主机连接的模板. 支持 count.

参数类	描述
selectScreens	查询在 screens 属性中返回主机的屏幕. 支持 count.||selectTrigg 查询在 |triggers 属性中返回主机的触发器. 支持 count.

参数类	描述
filter	对象仅回完全匹配指定筛选后的结果. 接受数组，键为属性名，值为一个单一值或者一个要匹配的数组. 允许通过接口属性进行过滤.

参数类	描述	
limitSelects	整数限	由子查询返回的记录数量. 适用于以下子查询: selectParent - 结果将按照host排序; selectInterf selectItems - 按name排序; selectDiscov - 按name排序; selectTrigge - 按description排序; selectGraphs - 按name排序; selectApplic - 按name排序; selectScreen - 按name

参数类	描述
search	对象返 与通配符相匹配的结果. 接受数组，键为属性名，值为待匹配搜索的字符串. 如果没有指定的额外选项，将会以 LIKE “%…%” 方式执行搜索. 允许通过接口属性搜索，仅

参数类	描述	
searchInventory	对象仅	回与指定通配符搜索资产清单数据匹配的主机. 这个参数同时受 `search` 参数影响.
sortfield	字符串/数组结果按给	的属性进行排序. 可能值: `hostid`, `host`, `name`, `status`.

参数类	描述	
countOutput	布尔值以下	是与reference commentary中详细描述的get方法相同的参数.
editable	布尔值::	
excludeSearch	布尔值::	
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(integer/array) 返回其中之一:

- 一组对象;
- 如果使用了 countOutput 参数，则返回获取的对象数量.

示例

通过名称获取数据

获取所有关于"Zabbix server" 和"Linux server" 两个主机的数据请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": [
                "Zabbix server",
                "Linux server"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenances": [],
            "hostid": "10160",
```

```
        "proxy_hostid": "0",
        "host": "Zabbix server",
        "status": "0",
        "disable_until": "0",
        "error": "",
        "available": "0",
        "errors_from": "0",
        "lastaccess": "0",
        "ipmi_authtype": "-1",
        "ipmi_privilege": "2",
        "ipmi_username": "",
        "ipmi_password": "",
        "ipmi_disable_until": "0",
        "ipmi_available": "0",
        "snmp_disable_until": "0",
        "snmp_available": "0",
        "maintenanceid": "0",
        "maintenance_status": "0",
        "maintenance_type": "0",
        "maintenance_from": "0",
        "ipmi_errors_from": "0",
        "snmp_errors_from": "0",
        "ipmi_error": "",
        "snmp_error": "",
        "jmx_disable_until": "0",
        "jmx_available": "0",
        "jmx_errors_from": "0",
        "jmx_error": "",
        "name": "Zabbix server",
        "description": "The Zabbix monitoring server.",
        "tls_connect": "1",
        "tls_accept": "1",
        "tls_issuer": "",
        "tls_subject": "",
        "tls_psk_identity": "",
        "tls_psk": ""
    },
    {
        "maintenances": [],
        "hostid": "10167",
        "proxy_hostid": "0",
        "host": "Linux server",
        "status": "0",
        "disable_until": "0",
        "error": "",
        "available": "0",
        "errors_from": "0",
        "lastaccess": "0",
        "ipmi_authtype": "-1",
        "ipmi_privilege": "2",
        "ipmi_username": "",
        "ipmi_password": "",
        "ipmi_disable_until": "0",
        "ipmi_available": "0",
        "snmp_disable_until": "0",
        "snmp_available": "0",
        "maintenanceid": "0",
        "maintenance_status": "0",
        "maintenance_type": "0",
        "maintenance_from": "0",
        "ipmi_errors_from": "0",
        "snmp_errors_from": "0",
```

```
            "ipmi_error": "",
            "snmp_error": "",
            "jmx_disable_until": "0",
            "jmx_available": "0",
            "jmx_errors_from": "0",
            "jmx_error": "",
            "name": "Linux server",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        }
    ],
    "id": 1
}
```

获取主机组

获取主机"Zabbix server"所属的主机组，并不检索主机本身的详细信息

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectGroups": "extend",
        "filter": {
            "host": [
                "Zabbix server"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "groups": [
                {
                    "groupid": "2",
                    "name": "Linux servers",
                    "internal": "0",
                    "flags": "0"
                },
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "internal": "0",
                    "flags": "0"
                }
            ]
        }
    ],
    "id": 2
```

```
}
```

获取关联的模板

获取主机”10084” 关联的模板的 ID 和名称请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectParentTemplates": [
            "templateid",
            "name"
        ],
        "hostids": "10084"
    },
    "id": 1,
    "auth": "70785d2b494a7302309b48afcdb3a401"
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "parentTemplates": [
                {
                    "name": "Template OS Linux",
                    "templateid": "10001"
                },
                {
                    "name": "Template App Zabbix Server",
                    "templateid": "10047"
                }
            ]
        }
    ],
    "id": 1
}
```

根据主机资产清单数据进行检索

获取主机清单中”OS” 字段包含”Linux” 的主机请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "host"
        ],
        "selectInventory": [
            "os"
        ],
        "searchInventory": {
            "os": "Linux"
        }
    },
    "id": 2,
    "auth": "7f9e00124c75e8f25facd5c093f3e9a0"
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "inventory": {
                "os": "Linux Ubuntu"
            }
        },
        {
            "hostid": "10107",
            "host": "Linux server",
            "inventory": {
                "os": "Linux Mint"
            }
        }
    ],
    "id": 1
}
```


参考

- Host group
- Template
- User macro
- Host interface

来源

CHost::get() in *frontends/php/include/classes/api/services/CHost.php*.

主机原型

该类被设计用来处理主机原型.

对象引用:

- Host prototype
- Host prototype inventory
- Group link
- Group prototype

可用方法:

- hostprototype.create - 创建新的主机原型
- hostprototype.delete - 删除主机原型
- hostprototype.get - 获取主机原型
- hostprototype.update - 更新主机原型

> 主机原型对象

以下对象与主机原型 API 直接相关.

主机原型

主机原型对象具有以下属性:

属性类	描述	
hostid	字符串 *(	读)* 主机原型的ID.
host (必选)	字符串主机	型的技术名称.
name	字符串主机	型的可见名称. 默认: host属性的值.
status	整数主	原型的状态. 可能的值: 0 - (默认) 被监控的主机; 1 - 不受监控的主机.

属性类	描述	
templateid	字符串 *(	读)*父模板主机原型的ID.
tls_connect	整数到	机的连接. 可能的值: 1 - (默认)无加密; 2 - PSK; 4 - 证书.
tls_accept	整数来	主机的连接. 可能的值: 1 - (默认)无加密; 2 - PSK; 4 - 证书.
tls_issuer	字符串证书	行者.
tls_subject	字符串证书	体.

属性类	描述	
tls_psk_identity	字符串 PS	身份如果 tls_connect 或 tls_accept 启用了 PSK，则必需.
tls_psk	字符串预共	密钥，至少 32 位十六进制数字。如果 tls_connect 或 tls_accept 启用了 PSK，则必需.

主机原型资产

主机原型资产对象有以下属性:

属性类	描述	
inventory_mode	整数主	原型资产模式. 可能的值: -1 - 禁用; 0 - (默认)手动; 1 - 自动.

组链接

组链接对象将主机原型与主机组链接，并具有以下属性:

属性类	描述	
group_prototypeid	字符串 *(	读)* 组链接的 ID.
groupid (必选)	字符串主机	的 ID.
hostid	字符串 *(	读)* 主机原型的 ID
templateid	字符串 *(	读)* 父模板组链接的 ID.

组原型

组原型对象定义将为已发现的主机创建的组，并具有以下属性:

属性类	描述	
group_prototypeid	字符串 *(	读)* 组原型的 ID.
name (必选)	字符串组原	的名称.
hostid	字符串 *(	读)* 主机原型的 ID
templateid	字符串 *(	读)* 父模板组原型的 ID.

创建

描述

`object hostprototype.create(object/array hostPrototypes)`

此方法允许创建新的主机原型.

参数

(对象/数组) 要创建的主机原型.

除标准主机原型属性之外，该方法接受以下参数.

参数类	描述	
groupLinks (必选)	数组要	主机原型创建的组链接.
ruleid (必选)	字符串主机	型所属的 LLD 规则的 ID.
groupPrototypes	数组将	主机原型创建的组原型.
inventory	对象主	原型资产属性.
templates	对象/数组连接到	机原型的模板. 模板必须已定义 `templateid` 属性.

返回值

(`object`) 在 `hostids` 属性中返回已创建主机原型 ID 的对象, 返回 ID 的顺序与传入主机原型的顺序一致.

示例

创建主机原型

使用组原型 `{# HV.NAME}` 为 LLD 规则 `23542`, 创建主机原型 `{# VM.NAME}`, 连接到主机组 `2`

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.create",
    "params": {
        "host": "{#VM.NAME}",
        "ruleid": "23542",
        "groupLinks": [
            {
                "groupid": "2"
            }
        ],
        "groupPrototypes": [
            {
                "name": "{#HV.NAME}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103"
        ]
    },
    "id": 1
}
```

参考

- Group link
- Group prototype
- Host prototype inventory

来源

CHostPrototype::create() in *frontends/php/include/classes/api/services/CHostPrototype.php*.

删除

描述

`object hostprototype.delete(array hostPrototypeIds)`

该方法允许删除主机原型.

参数

(数组) 要删除主机原型的 ID.

返回值

(对象) 在 `hostids` 属性中返回已删除主机原型 ID 的对象.

示例

删除多个主机原型

删除两个主机原型

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.delete",
    "params": [
        "10103",
        "10105"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103",
            "10105"
        ]
    },
    "id": 1
}
```

来源

CHostPrototype::delete() in *frontends/php/include/classes/api/services/CHostPrototype.php*.

更新

描述

`object hostprototype.update(object/array hostPrototypes)`

此方法允许更新已存在的主机原型.

参数

(对象/数组) 要更新的主机原型属性.

必须为每个主机原型定义 `hostid` 属性，所有其他属性都是可选的。只有过期的属性将被更新，所有其他属性将保持不变. 除标准主机原型属性外，该方法还接受以下参数:

参数类	描述	
groupLinks	数组组	接来替换主机原型上的当前组链接.
groupPrototypes	数组组	型替主机原型中已存在的组原型.
inventory	对象主	原型资产属性.
templates	对象/数组替换当	已连接的模板的模板. 模板必须已定义 `templateid` 属性.

返回值

（对象）在 hostids 属性中放回已更新主机原型 ID 的对象.

示例

禁用主机原型

通过将 status 状态设置为 1，可禁用主机原型.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "status": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

参考

- Group link
- Group prototype
- Host prototype inventory

来源

CHostPrototype::update() in *frontends/php/include/classes/api/services/CHostPrototype.php*.

获取

描述

`integer/array hostprototype.get(object parameters)`

该方法允许根据给定的参数获取主机原型记录.

参数

（对象）定义要输出的参数.

该方法支持如下属性:

参数类	描述	
hostids	字符串/数组返回给定	D 的主机原型.

参数类	描述
discoveryids	字符串/数组返回归属
inherited	布尔值如果
selectDiscoveryRule	查询在

定LLD规则的主机原型.

置为true,只返回模板从模板继承的项目.

discoveryRule属性中返回主机原型归属的LLD规则.|

|selectGroup

查询在|groupLinks属性中返回主机原型的组连接.

参数类	描述
selectGroupPrototypes	查询在 groupPrototypes 属性中返回主机原型的组原型.||selectInven 查询在 |inventory 属性中返回主机原型资产信息.

参数类	描述	
selectParentHost	查询在	parentHost 属性中返回主机原型所属的主机.||selectTempl 查询在 |templates 属性中返回连接到主机原型的模板.
sortfield	字符串/数组按照给定	属性对结果进行排序. 可能的值: hostid, host, name and status.

参数类	描述	
countOutput	布尔值这些	数对于所有get方法都是通用的，详情可参考Generic Zabbix API information.
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

（整数/数组）返回:

- 一组对象;
- 如果设置了 `countOutput` 参数, 则返回对象的数量.

示例

从 LLD 规则中获取主机原型

从 LLD 规则中获取所有主机原型及其组链接和组原型

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.get",
    "params": {
        "output": "extend",
        "selectGroupLinks": "extend",
        "selectGroupPrototypes": "extend",
        "discoveryids": "23554"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10092",
            "host": "{#HV.UUID}",
            "status": "0",
            "name": "{#HV.NAME}",
            "templateid": "0",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "groupLinks": [
                {
                    "group_prototypeid": "4",
                    "hostid": "10092",
                    "groupid": "7",
                    "templateid": "0"
                }
            ],
            "groupPrototypes": [
                {
                    "group_prototypeid": "7",
                    "hostid": "10092",
                    "name": "{#CLUSTER.NAME}",
                    "templateid": "0"
                }
            ]
        }
    ],
    "id": 1
}
```


参考

- Group link
- Group prototype
- Host prototype inventory

来源

CHostPrototype::get() in *frontends/php/include/classes/api/services/CHostPrototype.php*.

主机接口

这个类是设计用于处理主机接口.

对象引用:

- Host interface

可用方法:

- hostinterface.create - 创建新的主机接口
- hostinterface.delete - 删除主机接口
- hostinterface.get - 获取主机接口
- hostinterface.massadd - 批量添加主机接口
- hostinterface.massremove - 批量
- hostinterface.replacehostinterfaces - 替换主机接口
- hostinterface.update - 更新主机接口

以下对象与 `hostinterface`API 直接相关.

主机接口

主机接口对象具有以下属性.

Attention:
请注意，IP 和 DNS 都是必需的。如果您不想使用 DNS，请将其设置为空字符串.

属性类	描述	
interfaceid	字符串 *(	读)* 接口 ID.
dns (必选)	字符接	使用的 DNS 名称. 如果通过 IP 连接，可以设置为空.
hostid (必选)	字符接	归属的主机 ID.
ip (必选)	字符接	使用的 IP 地址. 如果通过 DNS 域名连接，可以设置为空.

属性类	描述	
main (必选)	整数该	口是否在主机上用作默认接口. 主机上只能有一种类型的接口作为默认设置. 可能的值: 0 - 不是默认; 1 - 默认.
port (必选)	字符接	使用的端口号, 可以包含用户宏.
type (必选)	整数接	类型. 可能的值: 1 - agent; 2 - SNMP; 3 - IPMI; 4 - JMX.

属性类	描述	
useip (必选)	整数是	应通过 IP 进行连接. 可能的值: 0 - 使用主机 DNS 名称连接; 1 - 使用该主机接口的主机 IP 地址进行连接.
bulk	整数是	使用批量 SNMP 请求. 可能的值: 0 - 不要使用批量请求; 1 - (默认) 使用批量请求.

创建

描述

`object hostinterface.create(object/array hostInterfaces)`

该方法允许创建新的主机接口.

参数

（对象/数组）创建主机接口, 该方法接受标准主机接口属性的主机接口.

返回值

（对象）在 `interfaceids` 属性中返回已创建主机接口 ID 的对象. 返回的 ID 顺序与传入的主机接口顺序保持一致.

示例

创建主机接口

给 ID 为 30052 主机创建辅助 IP 代理接口

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.create",
    "params": {
        "hostid": "30052",
        "dns": "",
        "ip": "127.0.0.1",
        "main": 0,
        "port": "10050",
        "type": 1,
        "useip": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30062"
        ]
    },
    "id": 1
}
```

参考

- hostinterface.massadd
- host.massadd

来源

CHostInterface::create() in *frontends/php/include/classes/api/services/CHostInterface.php*.

删除

描述

`object hostinterface.delete(array hostInterfaceIds)`

此方法允许删除主机接口

参数

(数组) 要删除主机接口的 ID.

返回值

(对象) 在 `interfaceids` 属性中返回已删除主机接口 ID 的对象.

示例

删除主机接口

删除 ID 为 30062 的主机接口.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.delete",
    "params": [
        "30062"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30062"
        ]
    },
    "id": 1
}
```

参考

- hostinterface.massremove
- host.massremove

来源

CHostInterface::delete() in *frontends/php/include/classes/api/services/CHostInterface.php*.

批量删除

描述

`object hostinterface.massremove(object parameters)`

此方法允许删除给定主机的主机接口.

参数

(对象) 包含要更新的主机的 ID 和要删除的接口的参数.

参数类	描述	
hostids (必选)	对象/数组要更新	主机 ID.
interfaces (必选)	对象/数组从给定	主机中删除主机接口. 主机接口对象必须已定义 ip,dns 和 port 属性

返回值

(对象) 在 `interfaceids` 属性中返回已删除主机接口 ID 的对象.

示例

删除接口

从给定的两台主机中删除"127.0.0.1" SNMP 接口.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massremove",
    "params": {
        "hostids": [
            "30050",
            "30052"
        ],
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "port": "161"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30069",
            "30070"
        ]
    },
    "id": 1
}
```

参考

- hostinterface.delete
- host.massremove

*

来源

CHostInterface::massRemove() in *frontends/php/include/classes/api/services/CHostInterface.php*.

批量添加

描述

`object hostinterface.massadd(object parameters)`

该方法允许同时向多个主机添加主机接口.

参数

(object) 包含要在给定主机上创建的主机接口的参数.

该方法接受以下参数:

参数类	描述	
hosts (必选)	对象/数组要更新的主机. 主机必须已定义 hostid 属性.	
interfaces (必选)	对象/数组在给定主机上创建主机接口.	

返回值

(object) 在 iterfaceids 属性中返回包含已创建主机接口 ID 的对象.

示例

创建接口

给两个主机创建接口.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30052"
            }
        ],
        "interfaces": {
```

```
            "dns": "",
            "ip": "127.0.0.1",
            "main": 0,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30069",
            "30070"
        ]
    },
    "id": 1
}
```

参考

- hostinterface.create
- host.massadd
- Host

来源

CHostInterface::massAdd() in *frontends/php/include/classes/api/services/CHostInterface.php*.

更新

描述

`object hostinterface.update(object/array hostInterfaces)`

此方法允许更新已存在的主机接口

参数

(对象/数组) 要更新的主机接口属性.

必须为每个主机接口定义 `interfaceid` 属性，所有其他属性都是可选的。只有给定的属性将被更新，所有其他属性将保持不变.

返回值

(对象) 在 `interfaceids` 属性中返回已更新主机接口 ID 的对象.

示例

更改主机接口端口

更改主机接口的端口.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.update",
    "params": {
        "interfaceid": "30048",
        "port": "30050"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30048"
        ]
    },
    "id": 1
}
```

来源

CHostInterface::update() in *frontends/php/include/classes/api/services/CHostInterface.php*.

替换

描述

`object hostinterface.replacehostinterfaces(object parameters)`

此方法允许给指定主机替换所有主机接口.

参数

（对象）包含要更新的主机 ID 和新主机接口的参数.

参数类	描述	
hostid (必选)	字符串要更	主机的 ID.
interfaces (必须)	对象/数组替换当	主机接口的主机接口.

返回值

（对象）在 `interfaceids` 属性中返回已创建主机接口 ID 的对象.

示例

更换主机接口

用单个代理接口替换所有主机接口.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.replacehostinterfaces",
    "params": {
        "hostid": "30052",
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "main": 1,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30081"
        ]
    },
    "id": 1
}
```

参考

- host.update
- host.massupdate

来源

CHostInterface::replaceHostInterfaces() in *frontends/php/include/classes/api/services/CHostInterface.php*.

获取

描述

`integer/array hostinterface.get(object parameters)`

此方法允许获取给定参数的主机接口记录

参数

(对象) 定义期望输出的参数.

该方法支持以下参数.

参数类	描述	
hostids	字符串/数组返回给定	机使用的主机接口.
interfaceids	字符串/数组返回给定	D 的主机接口.
itemids	字符串/数组返回给定	目的主机接口.
triggerids	字符串/数组返回给定	发器中项目使用的主机接口.
selectItems	查询返	`items` 属性中使用接口的监控项. 支持 `count`.
selectHosts	查询返	`hosts` 属性中使用接口作为数组的主机.
limitSelects	整数限	子选择返回的记录数. 适用于以下子选项: `selectItems`.
sortfield	字符串/数组按照给定	属性对结果进行排序. 可能的值: `interfaceid`, `dns`, `ip`.
countOutput	布尔值这些	数对于所有 get 方法都是通用的，详情可参考reference commentary.
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
nodeids	字符串/数组:::	
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(整数/数组) 返回:

- 一组对象;
- 如果设置了 `countOutput` 参数, 则返回获取到的对象数量.

示例

获取主机接口

获取 ID 为"30057' 的主机使用的接口的所有数据.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.get",
    "params": {
        "output": "extend",
        "hostids": "30057"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "interfaceid": "30050",
            "hostid": "30057",
            "main": "1",
            "type": "1",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "10050",
            "bulk": "1"
        },
        {
            "interfaceid": "30067",
            "hostid": "30057",
            "main": "0",
            "type": "1",
            "useip": "0",
            "ip": "",
            "dns": "localhost",
            "port": "10050",
            "bulk": "1"
        },
        {
            "interfaceid": "30068",
            "hostid": "30057",
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "161",
            "bulk": "1"
        }
    ],
    "id": 1
}
```

参考

- Host
- Item

来源

CHostInterface::get() in *frontends/php/include/classes/api/services/CHostInterface.php*.

主机组

该类用于管理主机组.

对象引用:

- Host group

可用的方法:

- hostgroup.create - 创建新的主机组
- hostgroup.delete - 删除主机组
- hostgroup.get - 获取主机组
- hostgroup.massadd - 给主机组添加相关的对象
- hostgroup.massremove - 删除主机组相关对象
- hostgroup.massupdate - 替换或删除主机组相关对象
- hostgroup.update - 更新主机组

> 主机组对象

以下对象是和 `hostgroup` 直接相关的 API

主机组

主机组对象有以下属性.

属性类	描述	
groupid	字符串 *(	读)* 主机组的 ID.
name (必选)	字符串主机	的名称.
flags	整数 *	只读)* 主机组的来源. 可能值: 0 - 普通的主机组; 4 - 被发现的主机组.

属性类	描述	
internal	整数 *	只读)* 该组是否由系统内部使用, 内部组无法被删除. 可能值: 0 - (默认) 不是内部; 1 - 内部.

创建

描述

`object hostgroup.create(object/array hostGroups)`

此方法允许创建新的主机组.

参数

`(object/array)` 创建主机组. 该方法接受具有标准主机组属性的主机组.

返回值

`(object)` 在 `groupids` 属性下返回包含已创建主机组 ID 的对象. 返回主机组 ID 的顺序与传入的主机组顺序一致.

示例

创建主机组

创建名为"Linux servers" 的主机组.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.create",
    "params": {
        "name": "Linux servers"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107819"
        ]
    },
```

```
    "id": 1
}
```

来源

CHostGroup::create() in *frontends/php/include/classes/api/services/CHostGroup.php*.

删除

描述

`object hostgroup.delete(array hostGroupIds)`

此方法允许删除主机组.

如果主机组有以下情况，则不能被删除:

- 包含仅属于该主机组的主机;
- 被标记为内部;
- 被主机原型引用;
- 在全局脚本中使用;
- 在相关条件下使用.

参数

(array) 要删除主机组的 ID.

返回值

(object) 在 groupids 属性中返回包含已删主机组 ID 的对象.

示例

删除多个主机组

删除两个主机组.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.delete",
    "params": [
        "107824",
        "107825"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107824",
            "107825"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::delete() in *frontends/php/include/classes/api/services/CHostGroup.php*.

批量删除

描述

`object hostgroup.massremove(object parameters)`

此方法允许从多个主机组中删除相关对象

参数

`(object)` 含要更新的主机组的 ID 和应该删除的对象的参数.

参数类	描述	
groupids (必选)	字符串/数组要更新主	组的 ID.
hostids	字符串/数组要从所有	机组中删除的主机.
templateids	字符串/数组要从所有	机组中删除的模板.

返回值

`(object)` 在 `groupids` 属性中返回已更新主机组 ID 的对象.

示例

从主机组中删除主机

从给定的主机组中删除两个主机

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "hostids": [
            "30050",
            "30001"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::massRemove() in *frontends/php/include/classes/api/services/CHostGroup.php*.

批量更新

描述

`object hostgroup.massupdate(object parameters)`

该方法允许对多个主机组批量替换或删除相关对象

参数

(object) 包含要更新的主机组的 ID 和应更新的对象的参数.

参数类	描述	
groups (必选)	对象/数组要更新主机组.	主机组必须已定义 groupid 属性.
hosts	对象/数组替换给主机组上当前主机的主机.	主机必须已定义 hostid 属性.
templates	对象/数组替换给主机组上当前模板的模板.	模板必须已定义 templateid 属性.

返回值

(object) 在 groupids 属性中返回包含已更新主机组 ID 的对象.

示例

替换主机组中的主机

替换主机组 ID 的所有主机

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massupdate",
    "params": {
        "groups": [
            {
                "groupid": "6"
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "6",
        ]
    },
    "id": 1
}
```

参考

- hostgroup.update
- hostgroup.massadd
- Host
- Template

来源

CHostGroup::massUpdate() in *frontends/php/include/classes/api/services/CHostGroup.php*.

批量添加

描述

`object hostgroup.massadd(object parameters)`

此方法允许给指定的主机组批量添加多个相关对象

参数

`(object)` 包含要更新的主机组的 ID 和要添加到所有主机组的对象的参数.

该方法接受如下参数.

参数类	描述	
groups (必选)	对象/数组要更新主机组必须已定义 `groupid` 属性.	主机组.
hosts	对象/数组添加到有主机组的主机. 主机必须已定义 `hostid` 属性.	
templates	对象/数组添加到有主机组的模板. 模板必须已定义 `templateid` 属性.	

返回值

`(object)` 在 `groupids` 属性中返回已更新主机组 ID 的对象.

示例

给主机组添加主机

给 ID 为 5 和 6 的主机组添加两个主机

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "5"
            },
            {
                "groupid": "6"
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30001"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

参考

- Host
- Template

来源

CHostGroup::massAdd() in *frontends/php/include/classes/api/services/CHostGroup.php*.

更新

描述

`object hostgroup.update(object/array hostGroups)`

此方法允许对已存在的主机组进行更新操作

参数

`(object/array)` 要更新的主机组属性.

必须为每个主机组定义 `groupid` 属性，所有其他属性都是可选的。只有给定的属性将被更新，所有其他属性将保持不变.

返回值

`(object)` 在 `groupids` 属性中返回包含已更新主机组 ID 的对象.

示例

重命名主机组

重命名 `Linux hosts` 主机组

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.update",
    "params": {
        "groupid": "7",
        "name": "Linux hosts"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "7"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::update() in *frontends/php/include/classes/api/services/CHostGroup.php*.

获取

描述

`整数/数组 hostgroup.get(object parameters)`

该方法允许根据指定的参数获取主机组

参数

`(object)` 定义期望输出的参数.

该方法支持以下参数:

参数类	描述	
graphids	字符串/数组返回包含	有给定图表的主机或模板的主机组.
groupids	字符串/数组返回给定	机组ID的主机组.
hostids	字符串/数组返回包含	定主机的主机组.
maintenanceids	字符串/数组返回受指	维护影响的主机组.
monitored_hosts	标识返	包含受监视主机的主机组.
real_hosts	标识返	包含主机的主机组.

参数类	描述	
templated_hosts	标识返	包含模板的主机组.
templateids	字符串/数组返回包含	定模板的主机组.
triggerids	字符串/数组返回包含	定触发器的主机或模板的主机组.
with_applications	标识返	给定应用集包含主机的主机组.
with_graphs	标识返	给定图表包含主机的主机组.

参数类	描述
with_hosts_and_templates	标识返包含主机或模板的主机组.
with_httptests	标识返给定web检查包含主机的主机组. 覆盖with_monitor参数.
with_items	标识返给定监控项包含主机或模板的主机组. 覆盖with_monitor和with_simple_参数.

参数类	描述	
with_monitored_httptests	标识返	启用web检查包含主机的主机组.
with_monitored_items	标识返	给定启动监控项包含主机或模板的主机组. 覆盖with_simple_参数.

参数类	描述	
with_monitored_triggers	标识返	给定启用触发器包含主机的主机组.触发器中使用的监控项必须事先已经启用.
with_simple_graph_items	标识返	给定数字型监控项包含主机的主机组.

参数类	描述	
with_triggers	标识返	给定触发器包含主机的主机组. 覆盖 with_monitor 参数.

参数类	描述	
selectDiscoveryRule	查询在	discoveryRule 属 性 中 返 回 创 建 主 机 组 的 发 现 规 则.| |selectGroup 查 询 在 |groupDiscov 属 性 中 返 回 主 机 组 发 现 对 象. 主 机 组 发 现 对 象 将 发 现 的 主 机 组 链 接 到 主 机 组 原 型， 并 具 有 以 下 属 性: groupid -

参数类	描述
selectHosts	查询在 hosts 属性中返回归属主机组的主机. 支持 count.||selectTempl 查询在 |tempaltes 属性中返回归属主机组的模板. 支持 count.

参数类	描述	
limitSelects	整数限	子选择返回的记录数. 适用于以下子选项: selectHosts - 结果将按host排序; selectTempla - 结果将按host排序.
sortfield	字符串/数组根据给定	属性排序. 可能值: groupid, name.

参数类	描述	
countOutput	布尔值这些	数对于所有get方法都是通用的，详情可参考reference commentary.
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

（整数/数组）返回:

- 一组对象;
- 如果使用了 `countOutput` 参数, 返回对象的数量.

示例

根据名称获取数据

获取所有关于主机组 `Zabbix servers` 和 `Linux servers` 的数据

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.get",
    "params": {
        "output": "extend",
        "filter": {
            "name": [
                "Zabbix servers",
                "Linux servers"
            ]
        }
    },
    "auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "2",
            "name": "Linux servers",
            "internal": "0"
        },
        {
            "groupid": "4",
            "name": "Zabbix servers",
            "internal": "0"
        }
    ],
    "id": 1
}
```


参考

- Host
- Template

来源

CHostGroup::get() in *frontends/php/include/classes/api/services/CHostGroup.php*.

代理

This class is designed to work with proxies. 这个类主要用来设计工作于代理

Object references:

- Proxy
- Proxy interface

Available methods:

- proxy.create - create new proxies
- proxy.delete - delete proxies
- proxy.get - retrieve proxies
- proxy.update - update proxies

> 代理对象

The following objects are directly related to the `proxy` API. 以下对象直接关系到 `proxy`API

代理

The proxy object has the following properties. 代理对象拥有以下属性

Property	Type	Description
proxyid	string	*(readonly)* ID of the proxy. 代理的 id
host (required)	string	Name of the proxy. 代理的名称
status (required)	integer	Type of proxy. Possible values: 5 - active proxy; 6 - passive proxy. 代理的类型 可能的值： 5 - 主动代理 6 - 被动代理
description	text	Description of the proxy. 代理的描述

Property	Type	Description
lastaccess	timestamp	*(readonly)* Time when the proxy last connected to the server. 上一次代理连接 zabbix server 的时间
tls_connect	integer	Connections to host. Possible values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate. 连接主机 可能的值： 1 - *(default)* 非加密 2 - 共享密钥（PSK）4 - 证书
tls_accept	integer	Connections from host. Possible bitmap values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate. 从代理连接 可能的值： 1 - *(default)* 非加密 2 - 共享密钥（PSK）4 - 证书
tls_issuer	string	Certificate issuer. 证书发行者
tls_subject	string	Certificate subject. 证书问题
tls_psk_identity	string	PSK identity. Required if either `tls_connect` or `tls_accept` has PSK enabled. 共享密钥（PSK）的身份，如果 `tls_connect` 或 `tls_accept` 都启用了 PSK，则需要使用。
tls_psk	string	The preshared key, at least 32 hex digits. Required if either `tls_connect` or `tls_accept` has PSK enabled. 预共享密钥，至少 32 位十六进制数字。如果 `tls_connect` 或 `tls_accept` 都启用了 PSK，则需要使用。
auto_compress	integer	*(readonly)* Indicates if communication between Zabbix server and proxy is compressed. Possible values are: 0 - No compression; 1 - Compression enabled; 只读指示 Zabbix 服务器和代理之间的通信是否被压缩。 \\可能的值： 0 - 不压缩 1 - 压缩

Proxy interface 代理接口

The proxy interface object defines the interface used to connect to a passive proxy. It has the following properties. 代理接口对象默认接口用于连接被动代理。他有一下属性

Property	Type	Description
interfaceid	string	*(readonly)* ID of the interface. 接口的 ID
dns (required)	string	DNS name to connect to. Can be empty if connections are made via IP address. 连接的 DNS 名称 \\如果通过 IP 地址进行连接，可以为空
ip (required)	string	IP address to connect to. Can be empty if connections are made via DNS names. 连接到 IP 地址 如果通过 DNS 名称连接可以为空

Property	Type	Description
port (required)	string	Port number to connect to.
useip (required)	integer	Whether the connection should be made via IP address. Possible values are: 0 - connect using DNS name; 1 - connect using IP address. 是否应该通过 IP 地址进行连接。 可能的值： 0 - 用 DNS 名称链接 1 - 用 IP 地址连接
hostid	string	*(readonly)* ID of the proxy the interface belongs to. // (只读) // 接口所属的代理的 ID。

创建

Description 描述

`object proxy.create(object/array proxies)`

This method allows to create new proxies. 此方法用于创建新的代理

Parameters 参数

`(object/array)` Proxies to create. `(object/array)` 创建代理

Additionally to the standard proxy properties, the method accepts the following parameters. 此外standard proxy properties，此方法接受以下参数。

Parameter	Type	Description
hosts	array	Hosts to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the `hostid` property defined. 由代理监视的主机。如果一个主机已经被另一个代理监视，那么它将被重新分配给当前代理。 此主机必须拥有 `hostid` 属性
interface	object	Host interface to be created for the passive proxy. Required for passive proxies. 创建主机接口用于被动代理 被动代理的需求

Return values 返回值

`(object)` Returns an object containing the IDs of the created proxies under the `proxyids` property. The order of the returned IDs matches the order of the passed proxies.

`(object)` 返回一个对象，该对象包含在 `proxyids` 属性下创建的代理的 id。返回的 id 的顺序与所传递的代理的顺序相匹配。

Examples 示例如下

Create an active proxy 创建一个主动的代理

Create an action proxy "Active proxy" and assign a host to be monitored by it. 创一个动作代理"Active proxy"，并分配一个由其监控的主机

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
```

```
    "params": {
        "host": "Active proxy",
        "status": "5",
        "hosts": [
            {
                "hostid": "10279"
            }
        ]
    },
    "auth": "ab9638041ec6922cb14b07982b268f47",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10280"
        ]
    },
    "id": 1
}
```

Create a passive proxy 创建一个反向代理

Create a passive proxy "Passive proxy" and assign two hosts to be monitored by it. 创建一个反向代理"Passive proxy"，并分配 2 个由其监控的主机。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "host": "Passive proxy",
        "status": "6",
        "interface": {
            "ip": "127.0.0.1",
            "dns": "",
            "useip": "1",
            "port": "10051"
        },
        "hosts": [
            {
                "hostid": "10192"
            },
            {
                "hostid": "10139"
            }
        ]
    },
    "auth": "ab9638041ec6922cb14b07982b268f47",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10284"
        ]
    },
```

```
    "id": 1
}
```

See also

- Host
- Proxy interface

Source

CProxy::create() in *frontends/php/include/classes/api/services/CProxy.php*.

删除

Description 描述

`object proxy.delete(array proxies)`

This method allows to delete proxies. 此方法允许删除代理

Parameters 参数

(`array`) IDs of proxies to delete.(`array`) 删除代理的 IDs

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted proxies under the `proxyids` property.(`object`) 返回在 `proxyids` 属性下包含已删除代理的 id 的对象。

Examples 示例如下

Delete multiple proxies 删除多个代理

Delete two proxies. 删除两个代理

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.delete",
    "params": [
        "10286",
        "10285"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10286",
            "10285"
        ]
    },
    "id": 1
}
```

Source

CProxy::delete() in *frontends/php/include/classes/api/services/CProxy.php*.

更新

Description 描述

`object proxy.update(object/array proxies)`

This method allows to update existing proxies. 此方法允许更新已存在的代理

Parameters 参数

(object/array) Proxy properties to be updated. (object/array) 代理参数被更新

The proxyid property must be defined for each proxy, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 每个主机必须定义 proxyid 参数，其他参数是可选的。仅仅传递的参数会被更新，其他的参数将保持不变。

Additionally to the standard proxy properties, the method accepts the following parameters. 此外standard proxy properties，此方法接受以下参数

Parameter	Type	Description
hosts	array	Hosts to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the hostid property defined. 代理监视的主机。如果一个主机已经被一个不同的代理监控，他将会重新分配到当前的代理 \\主机必须拥有 hostid 属性
interface	object	Host interface to replace the existing interface for the passive proxy. 主机接口将会替换已存在的主机接口用于被动代理

Return values 返回值

(object) Returns an object containing the IDs of the updated proxies under the proxyids property. (object) 返回一个对象，该对象包含 proxyids 属性下更新的代理的 id。

Examples 示例如下

Change hosts monitored by a proxy 改变一个主机的代理

Update the proxy to monitor the two given hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.update",
    "params": {
        "proxyid": "10293",
        "hosts": [
            "10294",
            "10295"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10293"
        ]
    },
    "id": 1
}
```

Change proxy status 改变代理的状态

Change the proxy to an active proxy and rename it to ”Active proxy”. 改变代理的模式是主动模式，并且重命名为”Active proxy”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.update",
    "params": {
        "proxyid": "10293",
        "host": "Active proxy",
        "status": "5"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10293"
        ]
    },
    "id": 1
}
```

See also

- Host
- Proxy interface

Source

CProxy::update() in *frontends/php/include/classes/api/services/CProxy.php*.

获取

Description 描述

`integer/array proxy.get(object parameters)`

The method allows to retrieve proxies according to the given parameters. 该方法允许根据给定的参数查询代理。

Parameters 参数

(`object`) Parameters defining the desired output. (`object`) 定义所需输出的参数。

The method supports the following parameters. 此方法支持一下参数

Parameter	Type	Description
proxyids	string/array	Return only proxies with the given IDs. 仅返回所给 IDs 的代理
selectHosts	query	Return hosts monitored by the proxy in the `hosts` property. 返回在 `hosts` 属性中代理监控的主机
selectInterface	query	Return the proxy interface used by a passive proxy in the `interface` property. 返回在 `interface` 属性中被动代理使用代理接口
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host` and `status`. 根据所给的属性进行排序 \\可能的值：`hostid`, `host` 和 `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary. 改参数适用于所有的 `get` 方法，详细描述是在reference commentary
editable	boolean	

Parameter	Type	Description
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values 返回值

(`integer/array`) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.
- 一个对象数组
- 搜索到对象的数量，如果 `countOutput` 对象被使用

Examples 示例如下

Retrieve all proxies 检索所有的代理

Retrieve all configured proxies and their interfaces. 检索所有配置的代理和他们的接口

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.get",
    "params": {
        "output": "extend",
        "selectInterface": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "interface": [],
            "host": "Active proxy",
            "status": "5",
            "lastaccess": "0",
            "proxyid": "30091",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        },
        {
            "interface": {
                "interfaceid": "30109",
                "hostid": "30092",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
```

```
                "port": "10051"
            ],
            "host": "Passive proxy",
            "status": "6",
            "lastaccess": "0",
            "proxyid": "30092",
            "description": ""
        }
    ],
    "id": 1
}
```


See also

- Host
- Proxy interface

Source

CProxy::get() in *frontends/php/include/classes/api/services/CProxy.php*.

任务

This class is designed to work with tasks. 此类用于管理任务。

Available methods: 可用方法：

- task.create - creating new tasks 创建新的任务。

创建

Description 说明

`object task.create(object task)`

This method allows to create new task. 该方法允许创建新的任务。

Parameters 参数

(object) A task to create. (object) 需要创建的任务。The method accepts the following parameters. 此方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
type (required 必须)	integer 整数型 Ta	k type. 任务类型。 Possible values: 许可值： 6 - Check now. 正在核实。

Parameter 参数 T	pe 类型 Des	ription 说明
itemids (required 必须)	string/array 字符串/数组 IDs	f items and low-level discovery rules. 监控项和低级别发现规则的 ID。 Item or discovery rule must of the following type: 监控项或自动发现规则必须是以下类型： 0 - Zabbix agent; Zabbix agent； 1 - SNMPv1 agent; SNMPv1 客户端； 3 - simple check; 简单检查； 4 - SNMPv2 agent; SNMPv2 客户端； 5 - Zabbix internal; Zabbix 内部； 6 - SNMPv3 agent; SNMPv3 客户端； 8 - Zabbix aggregate; Zabbix 整合； 10 - external check; 外部检查； 11 - database monitor; 数据库监控； 12 - IPMI agent; IPMI 客户端； 13 - SSH agent; SSH 客户端； 14 - TELNET agent; TELNET 客户端； 15 - calculated; 计算项； 16 - JMX agent. JMX 客户端。

Return values 返回值

(`object`) Returns an object containing the IDs of the created tasks under the `taskids` property. One task is created for each item and low-level discovery rule. The order of the returned IDs matches the order of the passed `itemids`. (`object`) 返回一个对象，该对象包含在 `taskids` 属性中已创建任务的 ID。为每个监控项和低级别发现规则创建的任务，返回 ID 的顺序与传递 `itemids` 的顺序相匹配。

Examples 范例

Creating a task 创建任务

Create a task `check now` for two items. One is an item, the other is a low-level discovery rule. 为两个项目，其中一个是监控项，另外一个低级别发现规则，创建一个 `check now` 任务。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": {
        "type": "6",
        "itemids": ["10092", "10093"],
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Source 源码

CTask::create() in *frontends/php/include/classes/api/services/CTask.php*. CTask::create() 方法可在 *frontends/php/include/classes/api/services/C* 中参考。

历史

这个类是设计用于处理历史数据

对象引用:

- History

可用方法:

- history.get - 获取历史数据.

> 历史对象

下列是与 `history` API 相关的对象。

Note:
历史对象会因为 item 的数据类型而有所不同. 它们都是 Zabbix Server 创建的，不能通过 API 进行修改

浮点类型

浮点类型的历史对象具有以下属性：

属性类	描述	
clock	时间戳获取	的时间
itemid	字符串相关	tem 的 ID
ns	整数获	值时的纳秒
value	浮点获	到的值

整数类型

整数类型的历史对象具有以下属性：

属性类	描述	
clock	时间戳获取	的时间
itemid	字符串相关	tem 的 ID
ns	整数获	值时的纳秒
value	整数获	到的值

字符串类型

字符串类型的历史对象具有以下属性：

属性类	描述	
clock	时间戳获取	的时间
itemid	字符串相关	tem 的 ID
ns	整数获	值时的纳秒
value	字符串获取	的值

文本类型

文本类型的历史对象具有以下属性：

属性类	描述	
id	字符串历史	录条目的 ID
clock	时间戳获取	的时间
itemid	字符串相关	tem 的 ID
ns	整数获	值时的纳秒
value	文本获	到的值

日志类型

日志类型的历史对象具有以下属性：

属性类	描述	
id	字符串历史	录条目的 ID
clock	时间戳获取	的时间
itemid	字符串相关	tem 的 ID
logeventid	整数 W	ndows 事件日志条目 ID
ns	整数获	值时的纳秒
severity	整数 W	ndows 事件日志条目级别
source	字符串 Wi	dows 事件日志条目源
timestamp	时间戳 Wi	dows 事件日志输入时间
value	文本获	到的值

获取

描述

整数/数组 `history.get(object parameters)`

该方法允许根据给定的参数检索历史数据参考: known issues

参数

`(object)` 定义期望输出的参数.

该方法支持以下参数：

参数类	描述	
history	整数要	回的历史对象类型. 可能的值: 0 - 数字浮点; 1 - 字符串; 2 - 日志; 3 - 无符号数字; 4 - 文本. 默认值: 3.
hostids	字符串/数组只返回给	主机的历史记录
itemids	字符串/数组只返回给	监控项的历史记录

参数类	描述	
time_from	时间戳仅返	在给定时间时或之后收到的值
time_till	时间戳仅返	在给定时间时或之前收到的值
sortfield	字符串/数组按照给定	属性对结果进行排序 可能的值: `itemid` 或 `clock`

参数类	描述	
countOutput	布尔值这些	数对于所有get方法都是通用的，详细描述可参考：reference commentary
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

（整数/数组）返回:

- 一组对象
- 如果使用了 `countOutput` 参数，返回检索对象的数量

示例

获取监控项历史数据

从数字 (浮点) 监控项中获取最近 10 条数据

请求:

```
{
    "jsonrpc": "2.0",
    "method": "history.get",
    "params": {
        "output": "extend",
        "history": 0,
        "itemids": "23296",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 10
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23296",
            "clock": "1351090996",
            "value": "0.0850",
            "ns": "563157632"
        },
        {
            "itemid": "23296",
            "clock": "1351090936",
            "value": "0.1600",
            "ns": "549216402"
        },
        {
            "itemid": "23296",
            "clock": "1351090876",
            "value": "0.1800",
            "ns": "537418114"
        },
        {
            "itemid": "23296",
            "clock": "1351090816",
            "value": "0.2100",
            "ns": "522659528"
        },
        {
            "itemid": "23296",
            "clock": "1351090756",
            "value": "0.2150",
            "ns": "507809457"
        },
        {
            "itemid": "23296",
            "clock": "1351090696",
            "value": "0.2550",
            "ns": "495509699"
        },
        {
            "itemid": "23296",
            "clock": "1351090636",
            "value": "0.3600",
            "ns": "477708209"
        },
        {
            "itemid": "23296",
            "clock": "1351090576",
            "value": "0.3750",
            "ns": "463251343"
        },
        {
            "itemid": "23296",
            "clock": "1351090516",
            "value": "0.3150",
            "ns": "447947017"
        },
        {
```

```
            "itemid": "23296",
            "clock": "1351090456",
            "value": "0.2750",
            "ns": "435307141"
        }
    ],
    "id": 1
}
```

来源

CHistory::get() in *frontends/php/include/classes/api/services/CHistory.php*.

图像

此类被设计用于管理图像.

对象引用:

- Image

可用的方法:

- image.create - 创建新图像
- image.delete - 删除图像
- image.get - 获取图像
- image.update - 更新图像

> 图像对象

以下对象是和 `image` API 直接相关.

图像

图像对象具有以下属性:

属性类	描述	
imageid	字符串 *(	读)* 图像的 ID.
name (必选)	字符串图像	名称.
imagetype	整型图	类型. 可能的值: 1 - (默认) 图标; 2 - 背景图.

创建

描述

`object image.create(object/array images)`

该方法允许创建新的图像.

参数

(对象/数组) 要创建的图像.

除标准图像属性了外，该方法接受以下参数:

属性类	说明	
image (必选)	字符串 Ba	e64 编码图像, 编码图像的最大大小为 1 MB.

返回值

（对象）返回一个包含 “imageid” 属性下创建的图像 ID 的对象。返回的 ID 的顺序与传递的图像的顺序相匹配.

示例

创建图像

创建一个云图标.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "image.create",
    "params": {
        "imagetype": 1,
        "name": "Cloud_(24)",
        "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAAApgB
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "188"
        ]
    },
    "id": 1
}
```

来源

CImage::create() in *frontends/php/include/classes/api/services/CImage.php*.

删除

描述

`object image.delete(array imageIds)`

此方法允许删除图像.

参数

（数组）要删除的图像 ID.

返回值

（对象）在 `imageids` 属性中返回已删除图像 ID 的对象.

示例

删除多个图像

删除两个图像.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "image.delete",
    "params": [
        "188",
        "192"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "188",
            "192"
        ]
    },
    "id": 1
}
```

来源

CImage::delete() in *frontends/php/include/classes/api/services/CImage.php*.

更新

描述

`object image.update(object/array images)`

该方法允许对已存在的图片进行更新.

参数

(对象/数组) 要更新的图像属性

必须为每个图像定义 `imageid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变

除了标准图像属性外，该方法接受以下参数：

参数类	描述
image	字符串 Ba e64 编码图像。编码图像的最大大小为 1 MB.

返回值

(对象) 在 `imageids` 属性中返回已更新图像 ID 的对象.

示例

重命名图像

将图像重命名为"Cloud icon".

请求:

```
{
    "jsonrpc": "2.0",
    "method": "image.update",
    "params": {
        "imageid": "2",
        "name": "Cloud icon"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "2"
```

```
        ]
    },
    "id": 1
}
```

来源

CImage::update() in *frontends/php/include/classes/api/services/CImage.php*.

获取

描述

`integer/array image.get(object parameters)`

该方法允许根据给定的参数获取图像记录.

参数

(对象) 定义要输出的参数.

该方法支持如下参数:

属性类	描述	
imageids	字符串/数组返回具有	定 ID 的图像.
sysmapids	字符串/数组返回给定	扑上使用的图像.
select_image	标识返	“image” 属性中的 Base64 编码图像.
sortfield	字符串/数组按照给定	属性对结果进行排序. 可能的值: `imageid` 和 `name`.
countOutput	布尔值这些	数对于所有 get 方法都是通用的，详情可参考reference commentary.
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(整数/数组) 返回:

- 一组对象;
- 如果设置了参数 `countOutput`, 则返回对象的数量.

示例

获取图像

获取 ID 为 2 的图像的所有数据。.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "image.get",
    "params": {
        "output": "extend",
        "select_image": true,
        "imageids": "2"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "imageid": "2",
            "imagetype": "1",
            "name": "Cloud_(24)",
            "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAA
        }
    ],
    "id": 1
}
```


来源

CImage::get() in *frontends/php/include/classes/api/services/CImage.php*.

图标映射

这个类被设计用来处理图标映射

对象引用:

- Icon map
- Icon mapping

可用的方法:

- iconmap.create - 创建新的图标映射
- iconmap.delete - 删除图标映射图
- iconmap.get - 获取图标映射
- iconmap.update - 更新图标映射

> 图标映射对象

以下是和 `iconmap` API 相关的方法

图标拓扑

图标映射对象有以下属性：

属性类	描述	
iconmapid	字符串 *(	读)* 图标映射的 ID.
default_iconid (必选)	字符串默认	标的 ID.
name (必选)	字符串图标	射的名称.

图标映射

图标映射对象定义了一个具体的图标，给具有特定资产清单字段值的主机使用. 图标映射有以下属性:

属性类	描述	
iconmappingid	字符串 *(	读)* 图标拓扑图 ID.

属性类	描述	
iconid (必选)	字符串被图	映射使用到的图标ID.
expression (必选)	字符串使资	清单字段匹配的表达式.
inventory_link (必选)	整数主	资产清单字段ID. 参考host inventory object支持的资产清单字段列表.
iconmapid	字符串 *(	读)* 图标映射所属的图标拓扑图ID.

属性类	描述	
sortorder	整数 *	只读)*在图标拓扑图中图标映射的位置.

创建

描述

`object iconmap.create(object/array iconMaps)`

此方法允许创建新的图标拓扑

参数

（对象/数组）要创建的图标拓扑.

另外，对于标准图标拓扑图属性，此方法接受以下参数:

参数类	描述	
mappings (必选)	数组为	标拓扑创建图标映射.

返回值

（对象）返回一个对象其中包含在 iconmapids 属性下已创建图标拓扑图的 ID。返回 ID 的命令与传递图标拓扑图的命令匹配.

示例

创建一个图标拓扑图

创建一个图标拓扑图来显示不同类型的主机.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.create",
    "params": {
        "name": "Type icons",
        "default_iconid": "2",
        "mappings": [
            {
                "inventory_link": 1,
                "expression": "server",
                "iconid": "3"
            },
            {
                "inventory_link": 1,
                "expression": "switch",
                "iconid": "4"
            }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2"
        ]
    },
    "id": 1
}
```

参考

- Icon mapping

来源

CIconMap::create() in *frontends/php/include/classes/api/services/CIconMap.php*.

删除

描述

`object iconmap.delete(array iconMapIds)`

此方法允许删除图标拓扑图.

参数

（数组）需要删除的图标拓扑图 ID.

返回值

（对象）返回一个对象其中包含在 iconmapids 属性下的已删除图标拓扑图 ID.

示例

删除多个图标拓扑图

删除两个图标拓扑图.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.delete",
    "params": [
        "2",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2",
            "5"
        ]
    },
```

```
    "id": 1
}
```

来源

CIconMap::delete() in *frontends/php/include/classes/api/services/CIconMap.php*.

更新

描述

`object iconmap.update(object/array iconMaps)`

此方法允许更新已存在的图标拓扑.

参数

(对象/数组) 要更新的图标拓扑的属性.

每一个图标拓扑图的 iconmapid 属性必须已定义过，其他属性为可选项。仅被传递的属性会被更新，其他属性保持不变.

参数类	描述	
mappings	数组替	当前图标映射.

返回值

(object) 返回一个对象其中包含在 iconmapids 属性下已更新图标拓扑图的 ID.

示例

重命名图标拓扑图

将图标拓扑图重命名为"OS icons".

请求:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.update",
    "params": {
        "iconmapid": "1",
        "name": "OS icons"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "1"
        ]
    },
    "id": 1
}
```

参考

- Icon mapping

来源

CIconMap::update() in *frontends/php/include/classes/api/services/CIconMap.php*.

获取

描述

`integer/array iconmap.get(object parameters)`

此方法允许根据指定的参数获取图标拓扑图.

参数

(对象) 定义要输出的参数.

该方法支持以下参数:

参数类	描述	
iconmapids	字符串/数组返回指定	D 的图标拓扑图.
sysmapids	字符串/数组返回在指	拓扑图中使用的图标拓扑图.
selectMappings	查询返	在 mappings 属性中使用的图标映射.
sortfield	字符串/数组根据指定	属性将结果排序. 可能的值: `iconmapid` 和 `name`.
countOutput	布尔值这些	数对于所有 get 方法都是通用的，详情可参考reference commentary.
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	integer	
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(整数/数组) 返回:

- 一组对象;
- 如果设置了 `countOutput` 参数, 则返回对象数量.

示例

获取一个图标拓扑图

获取所有关于 ID 为 3 的图标拓扑图数据.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.get",
    "params": {
        "iconmapids": "3",
        "output": "extend",
        "selectMappings": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mappings": [
                {
```

```
                    "iconmappingid": "3",
                    "iconmapid": "3",
                    "iconid": "6",
                    "inventory_link": "1",
                    "expression": "server",
                    "sortorder": "0"
                },
                {
                    "iconmappingid": "4",
                    "iconmapid": "3",
                    "iconid": "10",
                    "inventory_link": "1",
                    "expression": "switch",
                    "sortorder": "1"
                }
            ],
            "iconmapid": "3",
            "name": "Host type icons",
            "default_iconid": "2"
        }
    ],
    "id": 1
}
```


参考

- Icon mapping

来源

CIconMap::get() in *frontends/php/include/classes/api/services/CIconMap.php*.

媒介类型

This class is designed to work with media types. 这个类设计用来处理媒介类型。

Object references:

- Media type

Available methods:

- mediatype.create - creating new media types
- mediatype.delete - deleting media types
- mediatype.get - retrieving media types
- mediatype.update - updating media types

> 对象

The following objects are directly related to the `mediatype` API. 以下对象是直接关联到 `mediatype` 接口

Media type 媒介类型

The media type object has the following properties. 媒介类型参数拥有以下参数

Property	Type	Description
mediatypeid	string	*(readonly)* ID of the media type. 媒介类型 ID
description (required)	string	Name of the media type. 媒介类型名称

Property	Type	Description
type (required)	integer	Transport used by the media type. Possible values: 0 - e-mail; 1 - script; 2 - SMS; 3 - Jabber; 100 - Ez Texting. 媒介类型的传输方式 可能的值： 0-电子邮件 1-脚本 2-SMS 3-Jabber 100-Ez Texting。
exec_path	string	For script media types `exec_path` contains the name of the executed script. For Ez Texting `exec_path` contains the message text limit. Possible text limit values: 0 - USA (160 characters); 1 - Canada (136 characters). Required for script and Ez Texting media types. 对于脚本媒体类型，“exec_path” 包含已执行脚本的名称。 对于 z Texting `exec_path` 包含了消息文本的限制 可能的文本限定值：0- USA (160 characters) 1 - Canada (136 characters). 用于脚本和 Ez 短信媒体类型。
gsm_modem	string	Serial device name of the GSM modem. Required for SMS media types. GSM 调制解调器的串行设备名称。 用于 SMS 媒介类型
passwd	string	Authentication password. Required for Jabber and Ez Texting media types. 认证的密码 \\用于 Jabber 和 Ez Texting 媒介类型
smtp_email	string	Email address from which notifications will be sent. Required for email media types. 发送通知的电子邮件地址。 用于电子邮件媒介类型
smtp_helo	string	SMTP HELO. Required for email media types. SMTP HELO 用于电子邮件媒介类型
smtp_server	string	SMTP server. Required for email media types. SMTP server. \\用于电子邮件媒介类型
status	integer	Whether the media type is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled. 媒介类型是否是启用的 \\可能的值： 0-启用（默认）1-禁用

Property	Type	Description
username	string	Username or Jabber identifier. Required for Jabber and Ez Texting media types. 用户名或 Jabber 标识符 用于 Jabber and Ez Texting 媒介类型
exec_params	string	Script parameters. Each parameter ends with a new line feed. 脚本参数。 每个参数以新的行提要结束
maxsessions	integer	The maximum number of alerts that can be processed in parallel. Possible values for SMS: 1 - *(default)* Possible values for other media types: 0-100 可以并行处理的警报的最大数量。 SMS 可能的值：1（默认的）\\其他媒介类型可能的值： 0-100
maxattempts	integer	The maximum number of attempts to send an alert. Possible values: 1-10 Default value: 3 发送警报的最大尝试次数。 \\可能的值： 1-10，默认是 3
attempt_interval	string	The interval between retry attempts. Accepts seconds and time unit with suffix. Possible values: 0-60s Default value: 10s 重试尝试之间的间隔。接收带后缀的秒和时间单位。 可能的值：0~60s 默认是：10s

创建

Description 描述

`object mediatype.create(object/array mediaTypes)`

This method allows to create new media types. 此方法允许创建新的媒介类型

Parameters 参数

(`object/array`) Media types to create. (`object/array`) 创建媒介类型

The method accepts media types with the standard media type properties. 该方法接受媒介类型关于standard media type properties.

Return values 返回值

(`object`) Returns an object containing the IDs of the created media types under the `mediatypeids` property. The order of the returned IDs matches the order of the passed media types. (`object`) 返回一个包含在 “mediatypeids” 属性下创建的媒体类型的 ids 的对象，返回 id 的顺序与传递的媒介类型的顺序匹配。

Examples 示例如下

Creating a media type 创建一个媒介类型

Create a new e-mail media type. 创建一个新的邮件媒介类型

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "description": "E-mail",
        "type": 0,
        "smtp_server": "rootmail@company.com",
        "smtp_helo": "company.com",
        "smtp_email": "zabbix@company.com"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "7"
        ]
    },
    "id": 1
}
```

Creating a media type with custom options 创建具有自定义选项的媒体类型

Create a new script media type with custom value for number of attempts and interval between them. 创建一个具有自定义值的新脚本媒体类型，用于尝试次数和尝试间隔。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": 1,
        "description": "Push notifications",
        "exec_path": "push-notification.sh",
        "exec_params": "{ALERT.SENDTO}\n{ALERT.SUBJECT}\n{ALERT.MESSAGE}\n",
        "maxattempts": "5",
        "attempt_interval": "11s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "8"
        ]
    },
    "id": 1
}
```

Source

CMediaType::create() in *frontends/php/include/classes/api/services/CMediaType.php*.

删除

Description 描述

`object mediatype.delete(array mediaTypeIds)`

This method allows to delete media types. 此方法适合删除媒介类型

Parameters 参数

(array) IDs of the media types to delete. (array) 要删除媒介类型的 IDS

Return values 返回值

(object) Returns an object containing the IDs of the deleted media types under the mediatypeids property. (object) 返回一个对象，该对象包含 mediatypeids 属性下已删除的媒体类型的 id。

Examples 示例如下

Deleting multiple media types 删除多个媒介类型

Delete two media types. 删除 2 个媒介类型

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.delete",
    "params": [
        "3",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "3",
            "5"
        ]
    },
    "id": 1
}
```

Source

CMediaType::delete() in *frontends/php/include/classes/api/services/CMediaType.php*.

更新

Description 描述

`object mediatype.update(object/array mediaTypes)`

This method allows to update existing media types. 此方法允许更新已存在的媒介类型。

Parameters 参数

(object/array) Media type properties to be updated.

The mediatypeid property must be defined for each media type, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. mediatypeid 参数需要被每个每个类型所定义，其他的属性都是可选的。仅仅传递的属性会被更新，其他的属性将会保持不变

Return values 返回值

(object) Returns an object containing the IDs of the updated media types under the mediatypeids property. (object) 返回包含 mediatypeids 属性下所更新 IDs 的对象。

Examples 示例如下

Enabling a media type 启用一个媒介类型

Enable a media type, that is, set its status to 0. 启用一个媒介类型，就是设置他的 status 属性是 0.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.update",
    "params": {
        "mediatypeid": "6",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "6"
        ]
    },
    "id": 1
}
```

Source

CMediaType::update() in *frontends/php/include/classes/api/services/CMediaType.php*.

获取

Description 描述

`integer/array mediatype.get(object parameters)`

The method allows to retrieve media types according to the given parameters. 此方法用于检索给定参数和符合条件的媒介类型

Parameters 参数

(`object`) Parameters defining the desired output. (`object`) 定义所需输出的参数。

The method supports the following parameters. 此方法支持一下参数。

Parameter	Type	Description
mediatypeids	string/array	Return only media types with the given IDs. 仅返回所给 IDs 的媒介类型
mediaids	string/array	Return only media types used by the given media. 只返回给定媒体使用的媒介类型。
userids	string/array	Return only media types used by the given users. 只返回给定用户使用的媒介类型。
selectUsers	query	Return the users that use the media type in the `users` property. 返回 `users` 属性中使用媒介类型的用户。
sortfield	string/array	Sort the result by the given properties. Possible values are: `mediatypeid`. 根据给定的属性对结果进行排序。 可能的值是: `mediatypeid`
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary. 这些参数对于所有的 “get” 方法都是通用的reference commentary

Parameter	Type	Description
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(`integer/array`) Returns either: 返回如下：

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.
- 一个对象数组；
- 如果使用了 “countOutput” 参数，则检索对象的计数。

Examples 示例如下

Retrieving media types 检索媒介类型

Retrieve all configured media types. 检索所有配置的媒介类型

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "description": "Email",
            "smtp_server": "mail.company.com",
            "smtp_helo": "company.com",
            "smtp_email": "zabbix@company.com",
            "exec_path": "",
            "gsm_modem": "",
            "username": "",
            "passwd": "",
            "status": "0",
            "maxsessions": "1",
            "maxattempts": "7",
            "attempt_interval": "10s"
        },
        {
            "mediatypeid": "2",
            "type": "3",
            "description": "Jabber",
            "smtp_server": "",
```

```
            "smtp_helo": "",
            "smtp_email": "",
            "exec_path": "",
            "gsm_modem": "",
            "username": "jabber@company.com",
            "passwd": "zabbix",
            "status": "0",
            "maxsessions": "1",
            "maxattempts": "7",
            "attempt_interval": "10s"
        },
        {
            "mediatypeid": "3",
            "type": "2",
            "description": "SMS",
            "smtp_server": "",
            "smtp_helo": "",
            "smtp_email": "",
            "exec_path": "",
            "gsm_modem": "/dev/ttyS0",
            "username": "",
            "passwd": "",
            "status": "0",
            "maxsessions": "1",
            "maxattempts": "7",
            "attempt_interval": "10s"
        }
    ],
    "id": 1
}
```


See also

- User

Source

CMediaType::get() in *frontends/php/include/classes/api/services/CMediaType.php*.

拓扑图

This class is designed to work with maps. 这个类设计用来处理拓扑图

Object references:

- Map
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shape
- Map line

Available methods:

- map.create - create new maps
- map.delete - delete maps
- map.get - retrieve maps
- map.update - update maps

> 对象

The following objects are directly related to the map API. 以下内容是关于拓扑图接口。

Map 拓扑图

拓扑图对象具有以下属性 The map object has the following properties.

Property	Type	Description
sysmapid	string	*(readonly)* ID of the map. 拓扑图 id
height (required)	integer	Height of the map in pixels. 拓扑图画布高度
name (required)	string	Name of the map. 拓扑图名称
width (required)	integer	Width of the map in pixels. 拓扑图宽度
backgroundid	string	ID of the image used as the background for the map. 拓扑图背景图像 id
expand_macros	integer	Whether to expand macros in labels when configuring the map. Possible values: 0 - *(default)* do not expand macros; 1 - expand macros. 配置拓扑图时是否展开标签中的宏 可能的值: 0 - *(默认)* 不展开; 1 - 展开.
expandproblem	integer	Whether the the problem trigger will be displayed for elements with a single problem. Possible values: 0 - always display the number of problems; 1 - *(default)* display the problem trigger if there's only one problem. 如果只有一个触发器告警是否显示详情， \\可能的值： 0 是只显示数目，\\1 是显示触发器详情
grid_align	integer	Whether to enable grid aligning. Possible values: 0 - disable grid aligning; 1 - *(default)* enable grid aligning. 是否启用网格对齐 \\可能的值： 0 是不用 1 是使用
grid_show	integer	Whether to show the grid on the map. Possible values: 0 - do not show the grid; 1 - *(default)* show the grid. 是否显示拓扑图网格 \\可能的值： 0 是不显示 1 是显示
grid_size	integer	Size of the map grid in pixels. Supported values: 20, 40, 50, 75 and 100. Default: 50. 拓扑图网格的大小（以像素为单位） 支持 20, 40, 50, 75，100 像素， \\默认是 50

Property	Type	Description
highlight	integer	Whether icon highlighting is enabled. Possible values: 0 - highlighting disabled; 1 - *(default)* highlighting enabled. 是否启用图标高亮显示，\\可能的值： 0 是不用 1 是使用
iconmapid	string	ID of the icon map used on the map. 拓扑图使用图表的 ID
label_format	integer	Whether to enable advanced labels. Possible values: 0 - *(default)* disable advanced labels; 1 - enable advanced labels. 是否启用高级标签 \\可能的值： 0 是不用 1 是使用
label_location	integer	Location of the map element label. Possible values: 0 - *(default)* bottom; 1 - left; 2 - right; 3 - top. 拓扑图标签的位置 \\可能的值： 0 是底部 1 是左边 2 是右边 3 是顶部
label_string_host	string	Custom label for host elements. Required for maps with custom host label type. 主机元素自定义标签；需要拓扑图中的主机自定义标签类型
label_string_hostgroup	string	Custom label for host group elements. Required for maps with custom host group label type. 主机组元素自定义标签；需要拓扑图中的主机组自定义标签类型
label_string_image	string	Custom label for image elements. Required for maps with custom image label type. 图像元素的自定义标签。
label_string_map	string	Custom label for map elements. Required for maps with custom map label type. 拓扑图元素自定义标签
label_string_trigger	string	Custom label for trigger elements. Required for maps with custom trigger label type. 触发器元素自定义标签
label_type	integer	Map element label type. Possible values: 0 - label; 1 - IP address; 2 - *(default)* element name; 3 - status only; 4 - nothing. 拓扑图元素的标签类型可能的类型：0：标签 1：IP 地址 2：元素名称（默认）3：状态 4：没有

Property	Type	Description
label_type_host	integer	Label type for host elements. Possible values: 0 - label; 1 - IP address; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom. 主机元素的标签类型可能的值：0：标签 1：ip 地址 2：元素名称（默认）3：状态 4：没有 5：自定义
label_type_hostgroup	integer	Label type for host group elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom. 主机组元素的标签类型可能的值：0：标签 1：ip 地址 2：元素名称（默认）3：状态 4：没有 5：自定义
label_type_image	integer	Label type for host group elements. Possible values: 0 - label; 2 - *(default)* element name; 4 - nothing; 5 - custom. 图像元素的标签类型可能的值：0：标签 1：ip 地址 2：元素名称（默认）3：状态 4：没有 5：自定义
label_type_map	integer	Label type for map elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom. 拓扑图元素的标签类型可能的值：0：标签 1：ip 地址 2：元素名称（默认）3：状态 4：没有 5：自定义
label_type_trigger	integer	Label type for trigger elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom. 触发器元素的标签类型可能的值：0：标签 1：ip 地址 2：元素名称（默认）3：状态 4：没有 5：自定义
markelements	integer	Whether to highlight map elements that have recently changed their status. Possible values: 0 - *(default)* do not highlight elements; 1 - highlight elements. 是否突出显示最近更改其状态的拓扑图元素可能的值：0：不高亮 1：显示高亮
severity_min	integer	Minimum severity of the triggers that will be displayed on the map. Refer to the trigger ”severity” property for a list of supported trigger severities. 显示在拓扑图上的严重程度最小触发器。参考trigger ”severity” property，获取支持的触发器严重程度列表。

Property	Type	Description
show_unack	integer	How problems should be displayed. Possible values: 0 - *(default)* display the count of all problems; 1 - display only the count of unacknowledged problems; 2 - display the count of acknowledged and unacknowledged problems separately. 如何显示问题，可能的值：0：显示所有问题的总数 1：仅显示未确认问题的总数 2：分别显示已确认和未确认的数目
userid	string	Map owner user ID. 拓扑图所有用户的 id
private	integer	Type of map sharing. Possible values: 0 - public map; 1 - *(default)* private map. 拓扑图的共享类型可能的值：0：公共的拓扑图 1：私有的拓扑图

Map element 拓扑图元素

The map element object defines an object displayed on a map. It has the following properties. 拓扑图元素对象定义显示在拓扑图上的对象。它具有以下属性。

Property	Type	Description
selementid	string	*(readonly)* ID of the map element. 拓扑图元素的 id
elements (required)	array	Element data object. Required for host, host group, trigger and map type elements. 元素数据对象。需要主机、主机组、触发器和拓扑图类型元素。
elementtype (required)	integer	Type of map element. Possible values: 0 - host; 1 - map; 2 - trigger; 3 - host group; 4 - image. 拓扑图元素类型 可能的值：\\0-主机 1-拓扑图 2-触发器 3-主机组 4-图像
iconid_off (required)	string	ID of the image used to display the element in default state. 用于在默认状态下显示元素的图像的 ID。
areatype	integer	How separate host group hosts should be displayed. Possible values: 0 - *(default)* the host group element will take up the whole map; 1 - the host group element will have a fixed size. 应该如何显示独立的主机组主机。\\可能的值 0-主机组元素占用整个拓扑图 1-主机组元素的大小是固定的
application	string	Name of the application to display problems from. Used only for host and host group map elements. 显示问题的应用程序的名称。只用于主机和主机组映射元素。
elementsubtype	integer	How a host group element should be displayed on a map. Possible values: 0 - *(default)* display the host group as a single element; 1 - display each host in the group separately. 一个主机组元素如何显示在拓扑图上 可能的值： 0- 显示主机组作为一个单独的元素 1-分别显示组中的每个主机

Property	Type	Description
height	integer	Height of the fixed size host group element in pixels. Default: 200. 固定大小的主机组元素的高度 (以像素为单位)。 默认是：200
iconid_disabled	string	ID of the image used to display disabled map elements. Unused for image elements. 用于显示禁用映射元素的图像的 ID。未使用的图像元素。
iconid_maintenance	string	ID of the image used to display map elements in maintenance. Unused for image elements. 用于显示维护中的拓扑图元素的图像的 ID。未使用的图像元素。
iconid_on	string	ID of the image used to display map elements with problems. Unused for image elements. 用于显示有问题的拓扑图元素的图像的 ID。未使用的图像元素。
label	string	Label of the element. 元素的标签
label_location	integer	Location of the map element label. Possible values: -1 - *(default)* default location; 0 - bottom; 1 - left; 2 - right; 3 - top. 映射元素标签的位置 \\可能的值： -1: 默认的位置 0-底部 1-左边 2-右边 3-上边
permission	integer	Type of permission level. Possible values: -1 - none; 2 - read only; 3 - read-write. 类型的权限级别 可能的值：-1:-没有权限 2-只读权限 3-读写权限
sysmapid	string	*(readonly)* ID of the map that the element belongs to. 元素所述拓扑图的 ID
urls	array	Map element URLs. The map element URL object is described in detail below. 拓扑图元素的 URLS \
use_iconmap	integer	Whether icon mapping must be used for host elements. Possible values: 0 - do not use icon mapping; 1 - *(default)* use icon mapping. 是否必须为主机元素使用图标映射。 \\可能的值： 0-不使用图标映射 1-使用图表映射（默认的）
viewtype	integer	Host group element placing algorithm. Possible values: 0 - *(default)* grid. 主机组元素放置算法 \\可能的值： 0-网格
width	integer	Width of the fixed size host group element in pixels. Default: 200. 主机组元素固定的像素宽度。 默认是：200

Property	Type	Description
x	integer	X-coordinates of the element in pixels. Default: 0. 元素的 x 坐标，单位为像素。 默认是：0
y	integer	Y-coordinates of the element in pixels. Default: 0. 元素的 y 坐标，单位为像素。 默认是：0

Map element Host 拓扑图元素的主机

The map element Host object defines one host element. 拓扑图元素中的主机对象定义是一个主机元素

Property	Type	Description
hostid	string	Host ID

Map element Host group 拓扑图元素中的主机组

The map element Host group object defines one host group element. 拓扑图元素中的主机组对象定义是一个主机组元素。

Property	Type	Description
groupid	string	Host group ID

Map element Map 拓扑图元素中的拓扑图

The map element Map object defines one map element. 拓扑图元素中的拓扑图对象默认是一个拓扑图元素

Property	Type	Description
sysmapid	string	Map ID

Map element Trigger 拓扑图元素中的触发器

The map element Trigger object defines one or more trigger elements. 拓扑图元素中的触发器对象定义的是一个或者多个触发器元素

Property	Type	Description
triggerid	string	Trigger ID

Map element URL 拓扑图元素中的 URL

The map element URL object defines a clickable link that will be available for a specific map element. It has the following properties:
拓扑图元素 URL 对象定义了一个可单击的链接，该链接将对特定的 map 元素可用。它具有以下特性:

Property	Type	Description
sysmapelementurlid	string	*(readonly)* ID of the map element URL.
name (required)	string	Link caption.
url (required)	string	Link URL.
selementid	string	ID of the map element that the URL belongs to.

Map link 拓扑图关联

The map link object defines a link between two map elements. It has the following properties. 拓扑图链接对象定义两个映射元素之间的链接。它具有以下属性。

Property	Type	Description
linkid	string	*(readonly)* ID of the map link.
selementid1 (required)	string	ID of the first map element linked on one end. 在一端连接的第一个拓扑图元素的 ID。
selementid2 (required)	string	ID of the first map element linked on the other end. 另一端连接的第一个拓扑图元素的 ID。
color	string	Line color as a hexadecimal color code. Default: 000000. 行颜色作为十六进制颜色代码。 默认是：“000000”
drawtype	integer	Link line draw style. Possible values: 0 - *(default)* line; 2 - bold line; 3 - dotted line; 4 - dashed line. 链接线画的风格。 可能的值：0-线（默认）2-粗线 3-点线 4-虚线
label	string	Link label. 行标签
linktriggers	array	Map link triggers to use as link status indicators. The map link trigger object is described in detail below. 拓扑图链接触发器用作链接状态指示器。
permission	integer	Type of permission level. Possible values: -1 - none; 2 - read only; 3 - read-write. 权限等级类型 \\可能的值：-1-没有 2-只读 3-可读可写
sysmapid	string	ID of the map the link belongs to. 该关联所属拓扑图 ID

Map link trigger 拓扑图关联触发器

The map link trigger object defines a map link status indicator based on the state of a trigger. It has the following properties: 拓扑图链接触发器对象根据触发器的状态定义一个拓扑图链接状态指示器。它具有以下特性:

Property	Type	Description
linktriggerid	string	*(readonly)* ID of the map link trigger.
triggerid (reqiuired)	string	ID of the trigger used as a link indicator.
color	string	Indicator color as a hexadecimal color code. Default: DD0000.
drawtype	integer	Indicator draw style. Possible values: 0 - *(default)* line; 2 - bold line; 3 - dotted line; 4 - dashed line. 指标画的风格 \\可能的值：0-线（默认）2-粗线 3-点线 4-虚线
linkid	string	ID of the map link that the link trigger belongs to. 关联触发器所属拓扑图 ID

Map URL 拓扑图 URL

The map URL object defines a clickable link that will be available for all elements of a specific type on the map. It has the following properties: 拓扑图 URL 对象定义了一个可单击的链接，该链接可用于映射上特定类型的所有元素。它具有以下特性:

Property	Type	Description
sysmapurlid	string	*(readonly)* ID of the map URL. 拓扑图 URL ID
name (required)	string	Link caption. 链接标题。
url (required)	string	Link URL. 链接 URL
elementtype	integer	Type of map element for which the URL will be available. Refer to the map element ”type” property for a list of supported types. Default: 0. 拓扑图元素可用 URL 类型 默认：0
sysmapid	string	ID of the map that the URL belongs to. 所属 URL 的拓扑图 ID

Map user 拓扑图用户

List of map permissions based on users. It has the following properties: 基于用户的拓扑图权限列表。它具有以下特性:

Property	Type	Description
sysmapuserid	string	*(readonly)* ID of the map user. 拓扑图用户 ID
userid (required)	string	User ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write; 权限等级类型 \\可能的值：-1-没有 2-只读 3-可读可写

Map user group 拓扑图用户组

List of map permissions based on user groups. It has the following properties: 基于用户组的拓扑图权限列表。它具有以下特性:

Property	Type	Description
sysmapusrgrpid	string	*(readonly)* ID of the map user group. 拓扑图用户组的 ID
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write; 权限等级类型 \\可能的值：-1-没有 2-只读 3-可读可写

Map shapes 地图形状

The map shape object defines an geometric shape (with or without text) displayed on a map. It has the following properties: 拓扑图形状对象定义了显示在拓扑图上的几何形状 (包含或不包含文本)。它具有以下特性:

Property	Type	Description
sysmap_shapeid	string	*(readonly)* ID of the map shape element. 拓扑图形状元素的 ID

Property	Type	Description
type (required)	integer	Type of map shape element. Possible values: 0 - rectangle; 1 - ellipse. Property is required when new shapes are created. 拓扑图形状元素的类型 可能的值： 0-矩形 1-椭圆 创建新形状时需要属性。
x	integer	X-coordinates of the shape in pixels. Default: 0. 元素的 x 坐标，单位为像素。 默认是：0
y	integer	Y-coordinates of the shape in pixels. Default: 0. 元素的 y 坐标，单位为像素。 默认是：0
width	integer	Width of the shape in pixels. Default: 200. 以像素为单位的形状宽度。 \\默认是：200
height	integer	Height of the shape in pixels. Default: 200. 以像素为单位的形状高度。 \\默认是：200
text	string	Text of the shape. 文本的形状。

Property	Type	Description
font	integer	Font of the text within shape. Possible values: 0 - Georgia, serif 1 - “Palatino Linotype”, “Book Antiqua”, Palatino, serif 2 - “Times New Roman”, Times, serif 3 - Arial, Helvetica, sans-serif 4 - “Arial Black”, Gadget, sans-serif 5 - “Comic Sans MS”, cursive, sans-serif 6 - Impact, Charcoal, sans-serif 7 - “Lucida Sans Unicode”, “Lucida Grande”, sans-serif 8 - Tahoma, Geneva, sans-serif 9 - “Trebuchet MS”, Helvetica, sans-serif 10 - Verdana, Geneva, sans-serif 11 - “Courier New”, Courier, monospace 12 - “Lucida Console”, Monaco, monospace Default: 9. Font of the text within shape. 可能的值： 0 - Georgia, serif 1 - “Palatino Linotype”, “Book Antiqua”, Palatino, serif 2 - “Times New Roman”, Times, serif 3 - Arial, Helvetica, sans-serif 4 - “Arial Black”, Gadget, sans-serif 5 - “Comic Sans MS”, cursive, sans-serif 6 - Impact, Charcoal, sans-serif 7 - “Lucida Sans Unicode”, “Lucida Grande”, sans-serif 8 - Tahoma, Geneva, sans-serif 9 - “Trebuchet MS”, Helvetica, sans-serif 10 - Verdana, Geneva, sans-serif 11 - “Courier New”, Courier, monospace 12 - “Lucida Console”, Monaco, monospace 默认是：9
font_size	integer	Font size in pixels. Default: 11. 字体大小，单位是像素 默认：11
font_color	string	Font color. Default: ’000000’. 字体颜色 默认是：“000000”
text_halign	integer	Horizontal alignment of text. Possible values: 0 - center; 1 - left; 2 - right. Default: 0. 水平对齐的文本 \\可能的值： 0-中间（默认）1-左边 2-右边

Property	Type	Description
text_valign	integer	Vertical alignment of text. Possible values: 0 - middle; 1 - top; 2 - bottom. Default: 0. 垂直对齐文本 \\可能的值： 0-中间（默认）1-顶部 2-底部
border_type	integer	Type of the border. Possible values: 0 - none; 1 - ──────; 2 - ┅; 3 - ╌ ╌ ╌. Default: 0. 边界类型 可能的值： 0-没有（默认）1 - ────── 2 - ┅ 3 - ╌ ╌ ╌
border_width	integer	Width of the border in pixels. Default: 0. 边框的宽度，以像素为单位 \\默认：0
border_color	string	Border color. Default: '000000'. 边界的颜色 \\默认：'000000'
background_color	string	Background color (fill color). Default: (empty). 背景颜色（填充颜色） 默认是：无
zindex	integer	Value used to order shapes (z-index). Default: 0. 用于定制形状的值 (z-index)。 默认是：0

Map lines 拓扑图线

The map line object defines an line displayed on a map. It has the following properties: 拓扑图线对象定义显示在拓扑图上的行。它具有以下特性:

Property	Type	Description
sysmap_shapeid	string	*(readonly)* ID of the map shape element. 拓扑图形状元素的 ID
x1	integer	X-coordinates of the line point 1 in pixels. Default: 0. 以像素为单位的直线点 1 的 x 坐标。 \\默认是：0
y1	integer	Y-coordinates of the line point 1 in pixels. Default: 0. 以像素为单位的直线点 1 的 y 坐标。 \\默认是：0
x2	integer	X-coordinates of the line point 2 in pixels. Default: 200. 以像素为单位的直线点 2 的 x 坐标。 \\默认是：200

Property	Type	Description
y2	integer	Y-coordinates of the line point 2 in pixels. Default: 200. 以像素为单位的直线点 2 的 x 坐标。 \\默认是：200
line_type	integer	Type of the border. Possible values: 0 - none; 1 - ──────; 2 - ┅; 3 - – – –. Default: 0. 边界类型 可能的值： 0-没有（默认）1 - ────── 2 - ┅ 3 - – – –
line_width	integer	Width of the border in pixels. Default: 0. 边框的宽度，以像素为单位 \\默认：0
line_color	string	Border color. Default: '000000'. 边界的颜色 \\默认：'000000'
zindex	integer	Value used to order shapes (z-index). Default: 0. 用于定制形状的值 (z-index)。 默认是：0

创建

Description 描述

`object map.create(object/array maps)`

This method allows to create new maps. 这个方法允许创建一个新的拓扑图

Parameters 参数

`(object/array)` Maps to create.

Additionally to the standard map properties, the method accepts the following parameters. 除了standard map properties之外，该方法还接受以下参数。

Parameter	Type	Description
links	array	Map links to be created on the map. 拓扑图上创建拓扑图链接
selements	array	Map elements to be created on the map. 拓扑图上创建拓扑图元素
urls	array	Map URLs to be created on the map. 拓扑图上创建拓扑图 URL
users	array	Map user shares to be created on the map. 拓扑图共享用户
userGroups	array	Map user group shares to be created on the map. 拓扑图共享用户组
shapes	array	Map shapes to be created on the map. 拓扑图上创建拓扑图图形
lines	array	Map lines to be created on the map. 拓扑图上创建拓扑图线

Note:
To create map links you'll need to set a map elements `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See example. 要创建映射链接，您需要将映射元素设置为任意值，然后使用该值在链接 `selectd1` 或 `selemand2` 属性中引用该元素。在创建元素时，将用 Zabbix 生成的正确 ID 替换该值。

Return values 返回值

(object) Returns an object containing the IDs of the created maps under the sysmapids property. The order of the returned IDs matches the order of the passed maps.（对象）返回一个对象，该对象包含在 “sysmapid” 属性下创建的拓扑图的 id。返回 id 的顺序与传递的拓扑图的顺序相匹配。

Examples 例子

Create an empty map 创建一个空的拓扑图

Create a map with no elements. 创建一个拓扑图没有任何元素

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map",
        "width": 600,
        "height": 600
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

Create a host map 创建一个主机拓扑图

Create a map with two host elements and a link between them. Note the use of temporary ”selementid1” and ”selementid2” values in the map link object to refer to map elements. 创建一个关于两个主机的拓扑图，并且关联他们，需要注意的是在地图上临时使用”selementid1” 和”selementid2” 的值来引用地图元素。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "selementid": "1",
                "elements": [
                    {"hostid": "1033"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
            },

            {
                "selementid": "2",
                "elements": [
                    {"hostid": "1037"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
```

```
            }
        ],
        "links": [
            {
                "selementid1": "1",
                "selementid2": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

Create a trigger map 创建一个触发器拓扑图

Create a map with trigger element, which contains two triggers. 创建一个关于触发器元素的拓扑图，包含两个触发器

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Trigger map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "elements": [
                    {"triggerid": "12345"},
                    {"triggerid": "67890"}
                ],
                "elementtype": 2,
                "iconid_off": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

Map sharing 拓扑图共享

Create a map with two types of sharing (user and user group). 创建一个关于两种共享类项（用户和用户组）的拓扑图。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map sharing",
        "width": 600,
        "height": 600,
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

Map shapes 拓扑图形状

Create a map with map name title. 创建一个带有主题的拓扑图

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "shapes": [
            {
                "type": 0,
                "x": 0,
                "y": 0,
                "width": 600,
                "height": 11,
                "text": "{MAP.NAME}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

Map lines

Create a map line.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map API lines",
        "width": 500,
        "height": 500,
        "lines": [
            {
                "x1": 30,
                "y1": 10,
                "x2": 100,
                "y2": 50,
                "line_type": 1,
                "line_width": 10,
                "line_color": "009900"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "11"
        ]
    },
    "id": 1
}
```

See also

- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shape
- Map line

Source

CMap::create() in *frontends/php/include/classes/api/services/CMap.php*.

删除

Description 描述

`object map.delete(array mapIds)`

This method allows to delete maps. 这个方法允许删除拓扑图

Parameters 参数

(array) IDs of the maps to delete. 需要删除拓扑图的 IDS

Return values 返回值

(object) Returns an object containing the IDs of the deleted maps under the sysmapids property. 返回包含 “sysmapid” 属性下的已删除拓扑图的 IDS 的对象。

Examples 示例如下

Delete multiple maps 删除多个拓扑图

Delete two maps. 删除 2 个

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.delete",
    "params": [
        "12",
        "34"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "12",
            "34"
        ]
    },
    "id": 1
}
```

Source

CMap::delete() in *frontends/php/include/classes/api/services/CMap.php*.

更新

描述

`object map.update(object/array maps)`

This method allows to update existing maps. 此方法可以用来更新已存在的拓扑图

Parameters 参数

(object/array) Map properties to be updated. (object/array) 更新拓扑图参数

The mapid property must be defined for each map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. mapid 属性必须在每个拓扑图中定义，其他的属性是可选择的。只有传递的参数会被更新，其他的参数将会保持不变。

Additionally to the standard map properties, the method accepts the following parameters. 除了standard map properties, 此方法还接受以下参数。

Parameter	Type	Description
links	array	Map links to replace the existing links. 拓扑图链接以替换现有的链接。
selements	array	Map elements to replace the existing elements. 拓扑图元素替换成已存在的拓扑图元素
urls	array	Map URLs to replace the existing URLs. 拓扑图 URLs 替换成已存在的 URLs
users	array	Map user shares to replace the existing elements. 拓扑图的共享用户替换成已存在的共享用户
userGroups	array	Map user group shares to replace the existing elements. 拓扑图共享用户组替换成已存在的共享用户组
shapes	array	Map shapes to replace the existing shapes. 拓扑图图形替换成已存在的图形
lines	array	Map lines to replace the existing lines. 图谱图的连线替换成已存在的连线

To create map links between new map elements you'll need to set an elements `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See example for map.create. ::: <note tip> 要在新的拓扑图元素之间创建映射链接，您需要将一个元素设置为一个任意的值，然后使用这个值在链接 `selemand1` 或 `selemand2` 属性中引用这个元素。在创建元素时，将用 Zabbix 生成的正确 ID 替换该值。See example for map.create.

Return values 返回值

（`object`）Returns an object containing the IDs of the updated maps under the `sysmapids` property.（`object`）返回一个对象，该对象包含 "sysmapid" 属性下更新的映射的 id。

Examples 示例如下

Resize a map 调整拓扑图的大小

Change the size of the map to 1200x1200 pixels. 改变拓扑图的大小为 1200*1200，单位是像素。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "8",
        "width": 1200,
        "height": 1200
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

Change map owner 改变拓扑图的属组

Available only for admins and super admins. 仅适用于管理员和超级管理员

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
```

```
    "params": {
        "sysmapid": "9",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 2
}
```

See also

- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shapes
- Map lines

Source

CMap::update() in *frontends/php/include/classes/api/services/CMap.php*.

获取

Description 描述

`integer/array map.get(object parameters)`

The method allows to retrieve maps according to the given parameters. 这个方法允许根据给定参数检索出符合条件的拓扑图。

Parameters 参数

`(object)` Parameters defining the desired output. `(object)` 定义所需输出的参数。

The method supports the following parameters. 此方法支持一下参数。

Parameter	Type	Description
sysmapids	string/array	Return only maps with the given IDs. 仅返回给出 IDS 的拓扑图
userids	string/array	Return only maps that belong to the given user IDs. 仅返回所给用户 IDS 所属的拓扑图
expandUrls	flag	Adds global map URLs to the corresponding map elements and expands macros in all map element URLs. 将全局拓扑图 url 添加到相应的拓扑图元素，并扩展所有拓扑图元素 url 中的宏。
selectIconMap	query	Returns the icon map used on the map in the `iconmap` property. 返回 “iconmap” 属性中拓扑图上使用的图标映射。
selectLinks	query	Returns map links between elements in the `links` property. 返回 “links” 属性中元素之间的映射链接。
selectSelements	query	Returns the map elements from the map in the `selements` property.
selectUrls	query	Returns the map URLs in the `urls` property.
selectUsers	query	Returns users that the map is shared with in `users` property.

Parameter	Type	Description
selectUserGroups	query	Returns user groups that the map is shared with in `userGroups` property.
selectShapes	query	Returns the map shapes from the map in the `shapes` property.
selectLines	query	Returns the map lines from the map in the `lines` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `name`, `width` and `height`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:（整数/数组）回报:

- an array of objects 一个数组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果使用了 `countOutput` 参数，则检索对象的计数。

Examples 举例

Retrieve a map 检索一个拓扑图

Retrieve all data about map "3". 检索关于拓扑图 id 为 3 的所有数据。

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.get",
    "params": {
        "output": "extend",
        "selectSelements": "extend",
        "selectLinks": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "selectShapes": "extend",
        "selectLines": "extend",
        "sysmapids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "selements": [
                {
                    "selementid": "10",
```

```
                    "sysmapid": "3",
                    "elementtype": "4",
                    "iconid_off": "1",
                    "iconid_on": "0",
                    "label": "Zabbix server",
                    "label_location": "3",
                    "x": "11",
                    "y": "141",
                    "iconid_disabled": "0",
                    "iconid_maintenance": "0",
                    "elementsubtype": "0",
                    "areatype": "0",
                    "width": "200",
                    "height": "200",
                    "viewtype": "0",
                    "use_iconmap": "1",
                    "application": "",
                    "urls": [],
                    "elements": []
                },
                {
                    "selementid": "11",
                    "sysmapid": "3",
                    "elementtype": "4",
                    "iconid_off": "1",
                    "iconid_on": "0",
                    "label": "Web server",
                    "label_location": "3",
                    "x": "211",
                    "y": "191",
                    "iconid_disabled": "0",
                    "iconid_maintenance": "0",
                    "elementsubtype": "0",
                    "areatype": "0",
                    "width": "200",
                    "height": "200",
                    "viewtype": "0",
                    "use_iconmap": "1",
                    "application": "",
                    "urls": [],
                    "elements": []
                },
{
                    "selementid": "12",
                    "sysmapid": "3",
                    "elementtype": "0",
                    "iconid_off": "185",
                    "iconid_on": "0",
                    "label": "{HOST.NAME}\r\n{HOST.CONN}",
                    "label_location": "0",
                    "x": "111",
                    "y": "61",
                    "iconid_disabled": "0",
                    "iconid_maintenance": "0",
                    "elementsubtype": "0",
                    "areatype": "0",
                    "width": "200",
                    "height": "200",
                    "viewtype": "0",
                    "use_iconmap": "0",
                    "application": "",
                    "urls": [],
```

```
                "elements": [
                    {
                        "hostid": "10084"
                    }
                ]
            }
        ],
        "links": [
            {
                "linkid": "23",
                "sysmapid": "3",
                "selementid1": "10",
                "selementid2": "11",
                "drawtype": "0",
                "color": "00CC00",
                "label": "",
                "linktriggers": []
            }
        ],
        "users": [
            {
                "sysmapuserid": "1",
                "userid": "2",
                "permission": "2"
            }
        ],
        "userGroups": [
            {
                "sysmapusrgrpid": "1",
                "usrgrpid": "7",
                "permission": "2"
            }
        ],
        "shapes":[
            {
                "sysmap_shapeid":"1",
                "type":"0",
                "x":"0",
                "y":"0",
                "width":"680",
                "height":"15",
                "text":"{MAP.NAME}",
                "font":"9",
                "font_size":"11",
                "font_color":"000000",
                "text_halign":"0",
                "text_valign":"0",
                "border_type":"0",
                "border_width":"0",
                "border_color":"000000",
                "background_color":"",
                "zindex":"0"
            }
        ],
        "lines":[
            {
                "sysmap_shapeid":"2",
                "x1": 30,
                "y1": 10,
                "x2": 100,
                "y2": 50,
                "line_type": 1,
```

```
                    "line_width": 10,
                    "line_color": "009900",
                    "zindex":"1"
                }
            ],
            "sysmapid": "3",
            "name": "Local nerwork",
            "width": "400",
            "height": "400",
            "backgroundid": "0",
            "label_type": "2",
            "label_location": "3",
            "highlight": "1",
            "expandproblem": "1",
            "markelements": "0",
            "show_unack": "0",
            "grid_size": "50",
            "grid_show": "1",
            "grid_align": "1",
            "label_format": "0",
            "label_type_host": "2",
            "label_type_hostgroup": "2",
            "label_type_trigger": "2",
            "label_type_map": "2",
            "label_type_image": "2",
            "label_string_host": "",
            "label_string_hostgroup": "",
            "label_string_trigger": "",
            "label_string_map": "",
            "label_string_image": "",
            "iconmapid": "0",
            "expand_macros": "0",
            "severity_min": "0",
            "userid": "1",
            "private": "1"
        }
    ],
    "id": 1
}
```


See also

- Icon map
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shapes
- Map lines

Source

CMap::get() in *frontends/php/include/classes/api/services/CMap.php*.

服务

This class is designed to work with services. 该类用于配合服务使用。

Object references 对象引用:

- Service
- Service time
- Service dependency

- Service alarm

Available methods 可用方法:

- service.adddependencies - adding dependencies between IT services 增加 IT 服务之间的依赖关系
- service.addtimes - adding service times 增加服务时间
- service.create - creating new IT services 创建新的 IT 服务
- service.delete - deleting IT services 删除 IT 服务
- service.deletedependencies - deleting dependencies between IT services 删除 IT 服务之间的依赖关系
- service.deletetimes - deleting service times 删除服务时间
- service.get - retrieving IT services 检索 IT 服务
- service.getsla - retrieving availability information about IT services 检索有关 IT 服务的可用性信息
- service.update - updating IT services 更新 IT 服务

> 对象

The following objects are directly related to the `service` API. 以下对象与 `service`API 直接相关。

Service 服务

The service object has the following properties. 服务对象具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
serviceid	string 字符串 *(	eadonly 只读)* ID of the service. 服务的 ID。
algorithm (required 必须)	integer 整数型 Al	orithm used to calculate the state of the service. 用于计算服务状态的算法。 Possible values: 许可值： 0 - do not calculate; 不计算； 1 - problem 问题, if at least one child has a problem; 如果至少有一个子项有问题。 2 - problem 问题, if all children have problems. 如果所有子项都有问题。
name (required 必须)	string 字符串 Na	e of the service. 服务的名称。
showsla (required 必须)	integer 整数型 Wh	ther SLA should be calculated. 是否应计算 SLA。 Possible values: 许可值： 0 - do not calculate; 不计算； 1 - calculate. 计算。
sortorder (required 必须)	integer 整数型 Po	ition of the service used for sorting. 用于排序服务的位置。
goodsla	float 浮点数 Mi	imum acceptable SLA value. If the SLA drops lower, the service is considered to be in problem state. 最低可接受的 SLA 值，如果 SLA 降低，则该服务被认为处于有问题状态。 Default: 99.9. 默认：99.9。

Property 属性 T	pe 类型 Des	ription 说明
status	integer 整数型 *(	eadonly 只读)* Whether the service is in OK or problem state. 服务是否处于正常或问题状态。 If the service is in problem state, `status` is equal either to: 如果服务处于问题状态，`status` 相当于以下情况之一： - the priority of the linked trigger if it is set to 2, ”Warning” or higher (priorities 0, ”Not classified” and 1, ”Information” are ignored); 优先级设置为 2，”Warning” 或更高级别所链接的触发器告警（忽略优先级 0，”Not classified” 和 1，”Information”） - the highest status of a child service in problem state. 其中一个最高级别状态的子服务处于问题中。 If the service is in OK state, `status` is equal to 0. 如果服务是正常状态，那么 `status` 等于 0。
triggerid	string 字符串 Tr	gger associated with the service. Can only be set for services that don’t have children. 与服务相关联的触发器只能设置在没有子项的服务上。 Default: 0 默认：0

Service time 服务时间

The service time object defines periods, when an service is scheduled to be up or down. It has the following properties. 当一个服务按照计划上线或下线时，服务时间对象可定义周期。服务时间对象具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
timeid	string 字符串 *(	eadonly 只读)* ID of the service time. 服务时间的 ID。
serviceid (required 必须)	string 字符串 ID	of the service. 服务的 ID。 Cannot be updated. 不可更新。

Property 属性 T	pe 类型 Des	ription 说明
ts_from (required 必须)	integer 整数型 Ti	e when the service time comes into effect. 服务时间生效的时间。 For onetime downtimes `ts_from` must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Tue, 2:00 AM. 对于一次性停机时间，`ts_from` 必须设置为 Unix 时间戳，对于其他类型的事件 —— 设置为一周中的特定时间，以秒为单位，例如，90000 代表星期二，凌晨 2:00。
ts_to (required 必须)	integer 整数型 Ti	e when the service time ends. 服务时间关闭的时间。 For onetime uptimes `ts_to` must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Tue, 2:00 AM. 对于一次性开机时间，`ts_to` 必须设置为 Unix 时间戳，对于其他类型的事件 —— 设置为一周中的特定时间，以秒为单位，例如，90000 代表星期二，凌晨 2:00。
type (required 必须)	integer 整数型 Se	vice time type. 服务时间类型 Possible values: 许可值： 0 - planned uptime, repeated every week; 计划开机，每周重复； 1 - planned downtime, repeated every week; 计划停机，每周重复； 2 - one-time downtime. 一次性停机。
note	string 字符串 Ad	itional information about the service time. 有关服务时间的附加信息。

Service dependency 服务依赖

The service dependency object represents a dependency between services. It has the following properties. 服务依赖对象表示服务之间的依赖关系，它具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
linkid	string 字符串 *(	eadonly 只读)* ID of the service dependency. 服务依赖的 ID。
servicedownid (required 必须)	string 字符串 ID	of the service, that a service depends on, that is, the child service. An service can have multiple children. 被子服务依赖的服务 ID，一个服务可以有多个子服务。

Property 属性 T	pe 类型 Des	ription 说明
serviceupid (required 必须)	string 字符串 ID	of the service, that is dependent on a service, that is, the parent service. An service can have multiple parents forming a directed graph. 依赖于父服务的服务 ID，一个服务可以有多个父服务，从而形成一张定向图表。
soft (required 必须)	integer 整数型 Ty	e of dependency between services. 服务之间的依赖关系类型。 Possible values: 许可值： 0 - hard dependency; 硬依赖； 1 - soft dependency. 软依赖。 An service can have only one hard-dependent parent. This attribute has no effect on status or SLA calculation and is only used to create a core service tree. Additional parents can be added as soft dependencies forming a graph. 一个服务只能有一个强依赖的父服务。该属性对状态或 SLA 计算没有影响，仅用于创建核心服务树。新增的父服务可以作为形成图形的软依赖添加。 An service can not be deleted if it has hard-dependent children. 如果服务有硬依赖子服务，则无法删除该服务。

Service alarm 服务告警

Note:
Service alrams cannot be directly created, updated or deleted via the Zabbix API. 不能通过 Zabbix API 直接创建，更新或删除服务告警。

The service alarm objects represents an service's state change. It has the following properties. 服务告警对象代表服务的状态变化，它具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
servicealarmid	string 字符串 ID	of the service alarm. 服务告警的 ID。
serviceid	string 字符串 ID	of the service. 服务的 ID。
clock	timestamp 时间戳 Ti	e when the service state change has happened. 服务状态发生变化的时间。

Property 属性 T	pe 类型 Des	ription 说明
value	integer 整数型 St	tus of the service. 服务的状态。 Refer the the service status property for a list of possible values. 请参阅service status property以获取许可值列表。

创建

Description 说明

`object service.create(object/array services)`

This method allows to create new services. 此方法允许创建新的服务。

Parameters 参数

`(object/array)` services to create. `(object/array)` 创建服务。

Additionally to the standard service properties, the method accepts the following parameters. 除standard service properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
dependencies	array 数组 S	rvice dependencies. 服务依赖。 Each service dependency has the following parameters: 每个服务依赖项具有以下参数: - `dependsOnServiceid` - *(string 字符串)* ID of an service the service depends on, that is, the child service. 被子服务依赖的服务 ID。 - `soft` - *(integer 整数型)* type of service dependency; refer to the service dependency object page for more information on dependency types. 有关依赖关系类型的更多信息，请参阅service dependency object page
parentid	string 字符串 ID	of a hard-linked parent service. 硬链接的父服务的 ID。
times	array 数组 S	rvice times to be created for the service. 为服务创建的服务时间。

Return values 返回值

`(object)` Returns an object containing the IDs of the created services under the `serviceids` property. The order of the returned IDs matches the order of the passed services. `(object)` 返回一个对象，该对象包含在 `serviceids` 属性中已创建服务的 ID。返回 ID 的顺序与传递服务的顺序相匹配。

Examples 范例

Creating an service 创建服务

Create an service that will be switched to problem state, if at least one child has a problem. SLA calculation will be on and the minimum acceptable SLA is 99.99%. 创建一个至少有一个子服务有问题，将被切换到问题状态的服务。SLA 计算将打开并且 SLA 最低可接受 99.99%。Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.create",
    "params": {
        "name": "Server 1",
        "algorithm": 1,
        "showsla": 1,
        "goodsla": 99.99,
        "sortorder": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

Source 源码

CService::create() in *frontends/php/include/classes/api/services/CService.php*. CService::create() 方法可在 *frontends/php/include/classes/api/s* 中参考。

删除

Description 说明

`object service.delete(array serviceIds)`

This method allows to delete services. 此方法允许删除服务。

Services with hard-dependent child services cannot be deleted. 与子级服务有硬依赖关系的服务无法被删除。

Parameters 参数

(array) IDs of the services to delete.（array）要删除的服务 ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted services under the serviceids property.（object）返回一个对象，该对象包含在 serviceids 属性中被删除服务的 ID。

Examples

Deleting multiple services 删除多个服务

Delete two services. 删除两个服务。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.delete",
    "params": [
        "4",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source 源码

CService::delete() in *frontends/php/include/classes/api/services/CService.php*. CService::delete() 方法可在 *frontends/php/include/classes/api/s* 中参考。

删除依赖

Description 说明

`object service.deletedependencies(string/array serviceIds)`

This method allows to delete all dependencies from services. 此方法允许从服务中删除所有依赖关系。

Parameters 参数

(`string/array`) IDs of the services to delete all dependencies from. (`string/array`) 删除所有依赖关系的服务 ID。

Return values 返回值

(`object`) Returns an object containing the IDs of the affected services under the `serviceids` property. (`object`) 返回一个对象，该对象包含在 `serviceids` 属性中受影响服务的 ID。

Examples 范例

Deleting dependencies from an service 从服务中删除依赖关系

Delete all dependencies from service "2". 从服务"2" 中删除所有依赖项。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.deletedependencies",
    "params": [
        "2"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "2"
        ]
    },
    "id": 1
}
```

See also 参考

- service.update

Source 源码

CService::delete() in *frontends/php/include/classes/api/services/CService.php*. CService::delete() 方法可在 *frontends/php/include/classes/api/s* 中参考。

删除服务时间

Description 说明

`object service.deletetimes(string/array serviceIds)`

This method allows to delete all service times from services. 此方法允许从服务中删除所有服务时间。

Parameters 参数

(`string/array`) IDs of the services to delete all service times from. (`string/array`) 删除所有服务时间的服务 ID。

Return values 返回值

(`object`) Returns an object containing the IDs of the affected services under the `serviceids` property. (`object`) 返回一个对象，该对象包含在 `serviceids` 属性中受影响服务的 ID。

Examples 范例

Deleting service times from an service 从服务中删除服务时间

Delete all service times from service "2". 从服务"2" 中删除所有服务时间。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.deletetimes",
    "params": [
        "2"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "2"
        ]
    },
    "id": 1
}
```

See also 参考

- service.update

Source 源码

CService::delete() in *frontends/php/include/classes/api/services/CService.php*. CService::delete() 方法可在 *frontends/php/include/classes/api/s* 中参考。

更新

Description 说明

`object service.update(object/array services)`

This method allows to update existing services. 此方法允许更新现有服务。

Parameters 参数

(`object/array`) service properties to be updated. (`object/array`) 需要更新的服务属性。The `serviceid` property must be defined for each service, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个服务定义 `serviceid` 属性，所有其他属性为可选项。只有通过的属性会被更新，所有其他属性将保持不变。Additionally to the standard service properties, the method accepts the following parameters. 除standard service properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
dependencies	array 数组 S	rvice dependencies to replace the current service dependencies. 用来替换当前内容的服务依赖关系。 Each service dependency has the following parameters: 每个服务依赖项具有以下参数： - `dependsOnServiceid` - *(string 字符串)* ID of an service the service depends on, that is, the child service. 依赖服务的服务，即子服务的 ID。 - `soft` - *(integer 整数型)* type of service dependency; 服务依赖类型；refer to the service dependency object page for more information on dependency types. 有关依赖关系类型的更多信息，请参阅service dependency object page。
parentid	string 字符串 ID	of a hard-linked parent service. 硬链接的父服务 ID。
times	array 数组 S	rvice times to replace the current service times. 用来替换当前内容的服务时间。

Return values 返回值

（object）Returns an object containing the IDs of the updated services under the `serviceids` property.（object）返回一个对象，该对象包含在 `serviceids` 属性中已更新服务的 ID。

Examples 范例

Setting the parent of an service 设置父服务

Make service ”3” the hard-linked parent of service ”5”. 使服务”3” 硬链接于父服务”5”。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "5",
        "parentid": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

See also 参考

- service.adddependencies
- service.addtimes
- service.deletedependencies
- service.deletetimes

Source 源码

CService::update() in *frontends/php/include/classes/api/services/CService.php*. CService::update() 方法可在 *frontends/php/include/classes/api* 中参考。

添加依赖

Description 说明

`object service.adddependencies(object/array serviceDependencies)`

This method allows to create dependencies between services. 此方法允许创建服务之间的依赖关系。

Parameters 参数

(`object/array`) Service dependencies to create. (`object/array`) 创建服务依赖关系。

Each service dependency has the following parameters. 每个服务依赖项具有以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
serviceid	string 字符串 ID	of the service that depends on a service, that is, the parent service. 依赖父服务的服务 ID。
dependsOnServiceid	string 字符串 ID	of the service that a service depends on, that is, the child service. 被子服务依赖的服务 ID。
soft	string 字符串 Ty	e of dependency. 依赖类型。 Refer to the service dependency object page for more information on dependency types. 有关依赖关系类型的更多信息，请参阅service dependency object page。

Return values 返回值

(`object`) Returns an object containing the IDs of the affected parent services under the `serviceids` property. (`object`) 返回一个对象，该对象包含在 `serviceids` 属性中受影响父服务的 ID。

Examples 范例

Creating a hard dependency 创建一个硬依赖

Make service "2" a hard-dependent child of service "3". 使服务"2" 成为服务"3" 强依赖的子服务。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.adddependencies",
    "params": {
        "serviceid": "3",
        "dependsOnServiceid": "2",
        "soft": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also 参考

- service.update

Source 源码

CService::addDependencies() in *frontends/php/include/classes/api/services/CService.php*. CService::addDependencies() 方法可在 *frontends/php/include/classes/api/services/CService.php* 中参考。

添加服务时间

Description 说明

`object service.addtimes(object/array serviceTimes)`

This method allows to create new service times. 此方法允许创建新的服务时间。

Parameters 参数

`(object/array)` Service times to create. `(object/array)` 创建服务时间。

The method accepts service times with the standard service time properties. 该方法接受带有standard service time properties的服务时间。

Return values 返回值

`(object)` Returns an object containing the IDs of the affected services under the `serviceids` property. `(object)` 返回一个对象，该对象包含在 `serviceids` 属性中受影响服务的 ID。

Examples 范例

Adding a scheduled downtime 添加一个计划停机时间

Add a downtime for service "2" scheduled weekly from Monday 22:00 till Tuesday 10:00. 为服务"2" 添加一个从周一 22 点到周二 10 点的每周停机计划。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.addtimes",
    "params": {
        "serviceid": "4",
        "type": 1,
        "ts_from": 165600,
        "ts_to": 201600
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4"
        ]
```

```
    },
    "id": 1
}
```

See also 参考

- service.update

Source 源码

CService::addTimes() in *frontends/php/include/classes/api/services/CService.php*. CService::addTimes() 方法可在 *frontends/php/include/classe* 中参考。

获取

Description 说明

`integer/array service.get(object parameters)`

The method allows to retrieve services according to the given parameters. 此方法允许根据给定的参数检索服务。

Parameters 参数

`(object)` Parameters defining the desired output. `(object)` 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
serviceids	string/array 字符串/数组 Retu	n only services with the given IDs. 仅返回拥有指定 ID 的服务。
parentids	string/array 字符串/数组 Retu	n only services with the given hard-dependent parent services. 仅返回拥有指定硬依赖父服务的服务。
childids	string/array 字符串/数组 Retu	n only services that are hard-dependent on the given child services. 仅返回在指定子服务上有硬依赖的服务。
selectParent	query 查询 R	turn the hard-dependent parent service in the `parent` property. 返回 `parent` 属性中的硬依赖父服务。
selectDependencies	query 查询 R	turn child service dependencies in the `dependencies` property. 返回在 `dependencies` 属性中有依赖的子服务。
selectParentDependencies	query 查询 R	turn parent service dependencies in the `parentDependencies` property. 返回在 `parentDependencies` 属性中有依赖的父服务。
selectTimes	query 查询 R	turn service times in the `times` property. 返回在 `times` 属性中的服务时间。

Parameter 参数 T	pe 类型 Des	ription 说明
selectAlarms	query 查询 R	turn service alarms in the `alarms` property. 返回在 `alarms` 属性中的服务告警。
selectTrigger	query 查询 R	turn the associated trigger in the `trigger` property. 返回在 `trigger` 属性中的关联触发器。
sortfield	string/array 字符串/数组 Sort	the result by the given properties. 按指定的属性对结果分类。 Possible values are: `name` and `sortorder`. 许可值是：`name` 和 `sortorder`。
countOutput	boolean 布尔值 Th	se parameters being common for all `get` methods are described in detail in the reference commentary. 这些参数非常普遍，适用于所有 `get` 方法，具体描述详见于reference commentary。
editable	boolean 布尔值::	
excludeSearch	boolean 布尔值::	
filter	object 对象:	:
limit	integer 整数型::	
output	query 查询:	:
preservekeys	boolean 布尔值::	
search	object 对象:	:
searchByAny	boolean 布尔值::	
searchWildcardsEnabled	boolean 布尔值::	
sortorder	string/array 字符串/数组:::	
startSearch	boolean 布尔值::	

Return values 返回值

(`integer/array`) Returns either: 返回两者其中任一：

- an array of objects; 一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用了 `countOutput` 参数，则检索对象的计数。

Examples 范例

Retrieving all services 检索所有服务

Retrieve all data about all services and their dependencies. 检索有关所有服务及其依赖关系的所有数据。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.get",
    "params": {
        "output": "extend",
        "selectDependencies": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "serviceid": "2",
            "name": "Server 1",
            "status": "0",
            "algorithm": "1",
            "triggerid": "0",
            "showsla": "1",
            "goodsla": "99.9000",
            "sortorder": "0",
            "dependencies": []
        },
        {
            "serviceid": "3",
            "name": "Data center 1",
            "status": "0",
            "algorithm": "1",
            "triggerid": "0",
            "showsla": "1",
            "goodsla": "99.9000",
            "sortorder": "0",
            "dependencies": [
                {
                    "linkid": "11",
                    "serviceupid": "3",
                    "servicedownid": "2",
                    "soft": "0",
                    "sortorder": "0",
                    "serviceid": "2"
                },
                {
                    "linkid": "10",
                    "serviceupid": "3",
                    "servicedownid": "5",
                    "soft": "0",
                    "sortorder": "1",
                    "serviceid": "5"
                }
            ]
        },
        {
            "serviceid": "5",
            "name": "Server 2",
            "status": "0",
            "algorithm": "1",
            "triggerid": "0",
            "showsla": "1",
            "goodsla": "99.9900",
            "sortorder": "1",
            "dependencies": []
        }
    ],
    "id": 1
}
```

Source 源码

CService::get() in *frontends/php/include/classes/api/services/CService.php*. CService::get() 方法可在 *frontends/php/include/classes/api/service* 中参考。

获取 **SLA**

Description 说明

`object service.getsla(object parameters)`

This method allows to calculate availability information about services. 此方法允许计算有关服务的可用性信息。

Parameters 参数

`(object)` Parameters containing the IDs of the services and time intervals to calculate SLA. `(object)` 参数包含服务 ID 以及计算 SLA 的时间间隔。

Parameter 参数 T	pe 类型 Des	ription 说明
serviceids	string/array 字符串/数组 IDs	f services to return availability information for. 提供可用性信息的服务 ID。
intervals	array 数组 T	me intervals to return service layer availability information about. 返回服务层可用性信息的时间间隔。 Each time interval must have the following parameters: 每个时间间隔必须具有以下参数： - `from` - *(timestamp* 时间戳*)* interval start time; 间隔开始时间； - `to` - *(timestamp* 时间戳*)* interval end time. 间隔结束时间。

Return values 返回值

`(object)` Returns the following availability information about each service under the corresponding service ID. `(object)` 返回关于相应服务 ID 下每个服务的可用性信息。

Property 属性 T	pe 类型 Des	ription 说明
status	integer 整数型 Cu	rent status of the service. 当前服务的状态。 Refer to the service object page for more information on service statuses. 有关服务状态的更多信息，请参阅service object page。
problems	array 数组 T	iggers that are currently in problem state and are linked either to the service or one of its descendants. 当前处于故障状态并且与服务或服务的子项所关联的触发器。

Property 属性 T	pe 类型 Des	ription 说明
sla	array 数组 S	A data about each time period. 每个时间段的 SLA 数据。 Each SLA object has the following properties: 每个 SLA 对象具有以下属性： - `from` - *(timestamp 时间戳)* interval start time; 间隔开始时间； - `to` - *(timestamp 时间戳)* interval end time; 间隔结束时间； - `sla` - *(float 浮点数)* SLA for the given time interval; 指定时间间隔的 SLA； - `okTime` - *(integer 整数型)* time the service was in OK state, in seconds; 服务处于正常状态的时间，单位秒； - `problemTime` - *(integer 整数型)* time the service was in problem state, in seconds; 服务处于故障状态的时间，单位秒； - `downtimeTime` - *(integer 整数型)* time the service was in scheduled downtime, in seconds. 服务处于计划停机的时间，单位秒。

Examples 范例

Retrieving availability information for an service 检索服务的可用性信息

Retrieve availability information about a service during a week. 检索有关服务在一周内的可用性信息。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.getsla",
    "params": {
        "serviceids": "2",
        "intervals": [
            {
                "from": 1352452201,
                "to": 1353057001
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "2": {
            "status": "3",
            "problems": {
                "13904": {
```

```
                    "triggerid": "13904",
                    "expression": "{13359}=0",
                    "description": "Service unavailable",
                    "url": "",
                    "status": "0",
                    "value": "1",
                    "priority": "3",
                    "lastchange": "1352967420",
                    "comments": "",
                    "error": "",
                    "templateid": "0",
                    "type": "0",
                    "value_flags": "0",
                    "flags": "0"
                }
            },
            "sla": [
                {
                    "from": 1352452201,
                    "to": 1353057001,
                    "sla": 97.046296296296,
                    "okTime": 586936,
                    "problemTime": 17864,
                    "downtimeTime": 0
                }
            ]
        }
    },
    "id": 1
}
```


See also 参考

- Trigger

Source 源码

CService::getSla() in *frontends/php/include/classes/api/services/CService.php*. CService::getSla() 方法可在 *frontends/php/include/classes/api/s* 中参考。

模板

This class is designed to work with templates. 此类用于管理模板。Object references: 对象引用:

- 模板

Available methods: 可用方法:

- template.create - creating new templates 创建新模板
- template.delete - deleting templates 删除模板
- template.get - retrieving templates 检索模板
- template.massadd - adding related objects to templates 添加相关对象到模板中
- template.massremove - removing related objects from templates 从模板中删除相关对象
- template.massupdate - replacing or removing related objects from templates 从模板中替换或删除相关对象
- template.update - updating templates 更新模板

> 对象

The following objects are directly related to the `template` API. 以下对象与 API 模板直接相关。

Template 模板

The template object has the following properties. 模板对象具有以下属性。

Property 参数 T	pe 类型 Des	ription 说明
templateid	string 字符串 *(	eadonly 只读)* ID of the template. 模板 ID。
host (required 必须)	string 字符串 Te	hnical name of the template. 模板的正式名称。
description	text 文本 D	scription of the template. 模板说明。
name	string 字符串 Vi	ible name of the host. 主机的可见名称。 Default: `host` property value. 默认：主机的属性值。

创建

Description 说明

`object template.create(object/array templates)`

This method allows to create new templates. 此方法允许创建新模板。

Parameters 参数

(`object/array`) Templates to create. 创建模板。

Additionally to the standard template properties, the method accepts the following parameters. 除了标准模板属性之外，该方法接受以下属性。

Parameter 参数 T	pe 类型 Des	ription 说明
groups (required 必须)	object/array 对象/数组 Hos	groups to add the template to. 模板添加到主机组。 The host groups must have the `groupid` property defined. 主机组必须定义 `groupid` 属性。
templates	object/array 对象/数组 Tem	lates to be linked to the template. 被链接到模板的模板。 The templates must have the `templateid` property defined. 模板必须定义 `templateid` 属性。
macros	object/array 对象/数组 Use	macros to be created for the template. 为模板创建的用户宏。
hosts	object/array 对象/数组 Hos	s to link the template to. 链接到模板的主机。 The hosts must have the `hostid` property defined. 主机必须定义 `hostid` 属性。

Return values 返回值

(`object`) Returns an object containing the IDs of the created templates under the `templateids` property. The order of the returned IDs matches the order of the passed templates.(`object`) 返回一个对象，包含 `templateids` 属性中创建的模板 ID，返回 ID 的顺序与传递模板的顺序一致。

Examples 范例

Creating a template 创建模板

Create a template and link it to two hosts. 创建一个模板并将其链接到两台主机上。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.create",
    "params": {
        "host": "Linux template",
        "groups": {
            "groupid": 1
        },
        "hosts": [
            {
                "hostid": "10084"
            },
            {
                "hostid": "10090"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Source 源码

CTemplate::create() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::create() 方法可在 *frontends/php/include/classes* 中参考。

删除

Description 说明

`object template.delete(array templateIds)`

This method allows to delete templates. 此方法允许删除模板。

Parameters 参数

(`array`) IDs of the templates to delete. (`array`) 需要删除的模板 ID。

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted templates under the `templateids` property. (`object`) 返回一个对象，包含 `templateids` 属性中被删除模板的 ID。

Examples 范例

Deleting multiple templates 删除多个模板

Delete two templates. 删除两个模板。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.delete",
    "params": [
        "13",
        "32"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "13",
            "32"
        ]
    },
    "id": 1
}
```

Source 源码

CTemplate::delete() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::delete() 方法可在 *frontends/php/include/classes* 中参考。

批量删除

Description 说明

`object template.massremove(object parameters)`

This method allows to remove related objects from multiple templates. 方法允许从多个模板中删除相关对象。

Parameters 参数

(`object`) Parameters containing the IDs of the templates to update and the objects that should be removed. (`object`) 参数包含需要更新的模板 ID 以及需要删除的对象。

Parameter 参数 T	pe 类型 Des	ription 说明
templateids (required 必须)	string/array 字符串/数组 IDs	f the templates to be updated. 将要更新的模板 ID。
groupids	string/array 字符串/数组 Host	groups to remove the given templates from. 从指定的模板中删除主机组。
hostids	string/array 字符串/数组 Host	or templates to unlink the given templates from (downstream). 从主机或模板中取消指定模板（下游）的链接。
macros	string/array 字符串/数组 User	macros to delete from the given templates. 删除指定模板的用户宏。
templateids_clear	string/array 字符串/数组 Temp	ates to unlink and clear from the given templates (upstream). 从指定模板（上游）中取消模板链接并清除数据。

Parameter 参数 T	pe 类型 Des	ription 说明
templateids_link	string/array 字符串/数组 Temp	ates to unlink from the given templates (upstream). 从指定模板（上游）中取消模板链接。

Return values 返回值

（object）Returns an object containing the IDs of the updated templates under the templateids property.（object）返回一个对象，此对象包含在 templateids 中已更新模板的 ID。

Examples 范例

Removing templates from a group 从组中删除模板

Remove two templates from group ”2”. 从 ID 为”2” 的组中删除两个模板。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": [
            "10085",
            "10086"
        ],
        "groupids": "2"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Unlinking templates from a host 主机中取消模板链接

Unlink template ”10085” from two hosts. 从两台主机中取消 ID 为”10085” 的模板链接。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "hostids": [
            "10106",
            "10104"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085"
        ]
    },
    "id": 1
}
```

See also 参考

- template.update
- User macro

Source 源码

CTemplate::massRemove() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::massRemove() 方法可在 *frontends/php/include/classes/api/services/CTemplate.php* 中参考。

批量更新

Description 说明

`object template.massupdate(object parameters)`

This method allows to simultaneously replace or remove related objects and update properties on multiple templates. 此方法允许同时替换或删除相关对象并更新多个模板上的属性。

Parameters 参数

`(object)` Parameters containing the IDs of the templates to update and the properties that should be updated. `(object)` 参数包含需要更新的模板 ID 以及需要更新的属性。Additionally to the standard template properties, the method accepts the following parameters. 除standard template properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
templates (required 必须)	object/array 对象/数组 Tem	lates to be updated. 需要更新的模板。 The templates must have the `templateid` property defined. 模板必须已定义 `templateid` 属性。
groups	object/array 对象/数组 Hos	groups to replace the current host groups the templates belong to. 替换所属模板的当前主机组。 The host groups must have the `groupid` property defined. 主机组必须已定义 `groupid` 属性。
hosts	object/array 对象/数组 Hos	s and templates to replace the ones the templates are currently linked to. 替换当前链接模板的主机和模板。 Both hosts and templates must use the `hostid` property to pass an ID. 主机和模板都必须使用 `hostid` 属性传递唯一 ID。

Parameter 参数 T	pe 类型 Des	ription 说明
macros	object/array 对象/数组 Use	macros to replace the current user macros on the given templates. 替换指定模板上的当前用户宏。
templates_clear	object/array 对象/数组 Tem	lates to unlink and clear from the given templates. 从指定模板中取消链接并清除数据。 The templates must have the `templateid` property defined. 模板必须已定义 `templateid` 属性。
templates_link	object/array 对象/数组 Tem	lates to replace the currently linked templates. 替换当前链接的模板。 The templates must have the `templateid` property defined. 模板必须已定义 `templateid` 属性。

Return values 返回值

（`object`）Returns an object containing the IDs of the updated templates under the `templateids` property.（`object`）返回一个对象，此对象包含在 `templateids` 中已更新模板的 ID。

Examples 范例

Replacing host groups 替换主机组

Unlink and clear template ”10091” from the given templates. 从指定的模板中取消链接并清除 ID 为”10091” 的模板。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massupdate",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "templates_clear": [
            {
                "templateid": "10091"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

See also 参考

- template.update
- template.massadd
- Host group
- User macro

Source 源码

CTemplate::massUpdate() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::massUpdate() 方法可在 *frontends/php/include/classes/api/services/CTemplate.php* 中参考。

批量添加

Description 说明

`object template.massadd(object parameters)`

This method allows to simultaneously add multiple related objects to the given templates. 此方法允许同时替换或删除相关对象并更新多个模板上的属性。

Parameters 参数

(`object`) Parameters containing the IDs of the templates to update and the objects to add to the templates. (`object`) 参数包含需要更新的模板 ID 以及添加到模板的对象。The method accepts the following parameters. 该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
templates (required 必须)	object/array 对象/数组 Tem	lates to be updated. 需要更新的模板。 The templates must have the `templateid` property defined. 模板必须定义 `templateid` 属性。
groups	object/array 对象/数组 Hos	groups to add the given templates to. 主机组添加指定的模板。 The host groups must have the `groupid` property defined. 主机组必须定义 `groupid` 属性。
hosts	object/array 对象/数组 Hos	s and templates to link the given templates to. 将主机和模板链接到指定的模板中。 The hosts must have the `hostid` property defined. 主机必须定义 `hostid` 属性。
macros	object/array 对象/数组 Use	macros to be created for the given templates. 为指定的模板创建用户宏。

Parameter 参数 T	pe 类型 Des	ription 说明
templates_link	object/array 对象/数组 Tem	lates to link to the given templates. 将模板链接到指定模板。 The templates must have the `templateid` property defined. 模板必须定义 `templateid` 属性。

Return values 返回值

(`object`) Returns an object containing the IDs of the updated templates under the `templateids` property. (`object`) 返回一个对象，此对象包含在 `templateids` 属性中已更新模板的 ID。

Examples 范例

Adding templates to a group 添加模板到组

Add two templates to the host group "2". 添加两个模板到 ID 为"2" 的主机组中。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Linking a template to hosts 链接模板到主机

Link template "10073" to two hosts. 链接 ID 为"10073" 的模板到两台主机。

Request 请求:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10073"
            }
        ],
        "hosts": [
            {
                "hostid": "10106"
            },
            {
                "hostid": "10104"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10073"
        ]
    },
    "id": 1
}
```

See also 参考

- template.update
- Host
- Host group
- User macro

Source 源码

CTemplate::massAdd() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::massAdd() 方法可在//frontends/php/include/
中参考。

更新

Description 说明

`object template.update(object/array templates)`

This method allows to update existing templates. 此方法允许更新现有模板。

Parameters 参数

(`object/array`) Template properties to be updated. (`object/array`) 需要被更新的模板属性。

The `templateid` property must be defined for each template, all other properties are optional. Only the given properties will be updated, all others will remain unchanged. 必须为每个模板定义 `templateid` 属性，所有其他属性都是可选的。只有给定的属性将被更新，所有其他属性将保持不变。Additionally to the standard template properties, the method accepts the following parameters. 除standard template properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
groups	object/array 对象/数组 Hos	groups to replace the current host groups the templates belong to. 替换所属模板的当前主机组。 The host groups must have the `groupid` property defined. 主机组必须已定义 `groupid` 属性。
hosts	object/array 对象/数组 Hos	s and templates to replace the ones the templates are currently linked to. 替换当前链接模板的主机和模板。 Both hosts and templates must use the `hostid` property to pass an ID. 主机和模板都必须使用 `hostid` 属性传递唯一 ID。
macros	object/array 对象/数组 Use	macros to replace the current user macros on the given templates. 替换指定模板上的当前用户宏。
templates	object/array 对象/数组 Tem	lates to replace the currently linked templates. Templates that are not passed are only unlinked. 用于替换当前链接的模板，未通过的模板只是被取消链接。 The templates must have the `templateid` property defined. 模板必须已定义 `templateid` 属性。
templates_clear	object/array 对象/数组 Tem	lates to unlink and clear from the given templates. 从指定模板中取消链接并清除数据。 The templates must have the `templateid` property defined. 模板必须已定义 `templateid` 属性。

Return values 返回值

（`object`）Returns an object containing the IDs of the updated templates under the `templateids` property.（`object`）返回一个对象，此对象包含在 `templateids` 属性中已更新模板的 ID。

Examples 范例

Renaming a template 重命名模板

Rename the template to ”Template OS Linux”. 将模板重命名为”Template OS Linux”。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "name": "Template OS Linux"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Source 源码

CTemplate::update() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::update() 方法可在 *frontends/php/include/classe* 中参考。

获取

Description 说明

`integer/array template.get(object parameters)`

The method allows to retrieve templates according to the given parameters. 此方法允许根据指定的参数检索模板。

Parameters 参数

(`object`) Parameters defining the desired output. (`object`) 定义需要输出的参数。

The method supports the following parameters. 此方法支持以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
templateids	string/array 字符串/数组 Retu	n only templates with the given template IDs. 只返回指定模板 ID 的模板。
groupids	string/array 字符串/数组 Retu	n only templates that belong to the given host groups. 只返回指定主机组所属的模板。
parentTemplateids	string/array 字符串/数组 Retu	n only templates that are children of the given templates. 只返回指定子类模板的模板。
hostids	string/array 字符串/数组 Retu	n only templates that are linked to the given hosts. 只返回被链接到指定主机的模板。
graphids	string/array 字符串/数组 Retu	n only templates that contain the given graphs. 只返回包含指定图形的模板。

Parameter 参数 T	pe 类型 Des	ription 说明
itemids	string/array 字符串/数组 Retu	n only templates that contain the given items. 只返回包含指定监控项的模板。
triggerids	string/array 字符串/数组 Retu	n only templates that contain the given triggers. 只返回包含指定触发器的模板。
with_items	flag 标记 R	turn only templates that have items. 只返回具有监控项的模板。
with_triggers	flag 标记 R	turn only templates that have triggers. 只返回具有触发器的模板。
with_graphs	flag 标记 R	turn only templates that have graphs. 只返回具有图形的模板。
with_httptests	flag 标记 R	turn only templates that have web scenarios. 只返回具有 web 场景的模板。
selectGroups	query 查询 R	turn the host groups that the template belongs to in the `groups` property. 从 `groups` 属性中返回所属模板的主机组。
selectHosts	query 查询 R	turn the hosts that are linked to the template in the `hosts` property. 从 `hosts` 属性中返回被链接模板的主机。 Supports `count`. 支持 `count`。
selectTemplates	query 查询 R	turn the child templates in the `templates` property. 从 `templates` 属性中返回子类模板。 Supports `count`. 支持 `count`。
selectParentTemplates	query 查询 R	turn the parent templates in the `parentTemplates` property. 从 `parentTemplates` 属性中返回父类模板。 Supports `count`. 支持 `count`。
selectHttpTests	query 查询 R	turn the web scenarios from the template in the `httpTests` property. 从 `httpTests` 属性中返回来自模板的 web 场景。 Supports `count`. 支持 `count`。

Parameter 参数 T	pe 类型 Des	ription 说明
selectItems	query 查询 R	turn items from the template in the `items` property. 从 `items` 属性中返回来自模板的监控项。 Supports `count`. 支持 `count`。
selectDiscoveries	query 查询 R	turn low-level discoveries from the template in the `discoveries` property. 从 `discoveries` 属性中返回来自模板的低级别发现。 Supports `count`. 支持 `count`。
selectTriggers	query 查询 R	turn triggers from the template in the `triggers` property. 从 `triggers` 属性中返回来自模板的触发器。 Supports `count`. 支持 `count`。
selectGraphs	query 查询 R	turn graphs from the template in the `graphs` property. 从 `graphs` 属性中返回来自模板的图表。 Supports `count`. 支持 `count`。
selectApplications	query 查询 R	turn applications from the template in the `applications` property. 从 `applications` 属性中返回来自模板的应用。 Supports `count`. 支持 `count`。
selectMacros	query 查询 R	turn the macros from the template in the `macros` property. 从 `macros` 属性中返回来自模板的宏。
selectScreens	query 查询 R	turn screens from the template in the `screens` property. 从 `screens` 属性中返回来自模板的聚合图形。 Supports `count`. 支持 `count`。

Parameter 参数 T	pe 类型 Des	ription 说明
limitSelects	integer 整数型 Li	its the number of records returned by subselects. 限制子查询返回的记录数。 Applies to the following subselects: 应用于以下子查询： `selectTemplates` - results will be sorted by `name`; 结果将以 `name` 排序； `selectHosts` - sorted by `host`; 以 `host` 排序； `selectParentTemplates` - sorted by `host`; 以 `host` 排序； `selectItems` - sorted by `name`; 以 `name` 排序； `selectDiscoveries` - sorted by `name`; 以 `name` 排序； `selectTriggers` - sorted by `description`; 以 `description` 排序； `selectGraphs` - sorted by `name`; 以 `name` 排序； `selectApplications` - sorted by `name`; 以 `name` 排序； `selectScreens` - sorted by `name`. 以 `name` 排序；
sortfield	string/array 字符串/数组 Sort	the result by the given properties. 根据给定的属性为结果排序。 Possible values are: `hostid`, `host`, `name`, `status`. 许可值为：`hostid`,`host`,`name`,`status`。
countOutput	boolean 布尔值 Th	se parameters being common for all `get` methods are described in detail in the reference commentary. 这些参数十分普遍，适用所有 `get` 方法，详情参见reference commentary。
editable	boolean 布尔值::	
excludeSearch	boolean 布尔值::	
filter	object 对象:	:
limit	integer 整数型::	
output	query 查询:	:

Parameter 参数 T	pe 类型 Des	ription 说明
preservekeys	boolean 布尔值::	
search	object 对象:	:
searchByAny	boolean 布尔值::	
searchWildcardsEnabled	boolean 布尔值::	
sortorder	string/array 字符串/数组:::	
startSearch	boolean 布尔值::	

Return values 返回值

(integer/array) Returns either: (integer/array) 返回两者其中任一：

- an array of objects; 一组对象；
- the count of retrieved objects, if the countOutput parameter has been used. 如果已经使用了 countOutput 参数，则检索对象的计数。

Examples 范例

Retrieving templates by name 按名称检索模板

Retrieve all data about two templates named "Template OS Linux" and "Template OS Windows". 检索名称为"Template OS Linux"和"Template OS Windows" 这两个模板的所有数据。Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "Template OS Linux",
                "Template OS Windows"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "proxy_hostid": "0",
            "host": "Template OS Linux",
            "status": "3",
            "disable_until": "0",
            "error": "",
            "available": "0",
            "errors_from": "0",
            "lastaccess": "0",
            "ipmi_authtype": "0",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "ipmi_disable_until": "0",
            "ipmi_available": "0",
            "snmp_disable_until": "0",
            "snmp_available": "0",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
```

```
            "ipmi_errors_from": "0",
            "snmp_errors_from": "0",
            "ipmi_error": "",
            "snmp_error": "",
            "jmx_disable_until": "0",
            "jmx_available": "0",
            "jmx_errors_from": "0",
            "jmx_error": "",
            "name": "Template OS Linux",
            "flags": "0",
            "templateid": "10001",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        },
        {
            "proxy_hostid": "0",
            "host": "Template OS Windows",
            "status": "3",
            "disable_until": "0",
            "error": "",
            "available": "0",
            "errors_from": "0",
            "lastaccess": "0",
            "ipmi_authtype": "0",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "ipmi_disable_until": "0",
            "ipmi_available": "0",
            "snmp_disable_until": "0",
            "snmp_available": "0",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "ipmi_errors_from": "0",
            "snmp_errors_from": "0",
            "ipmi_error": "",
            "snmp_error": "",
            "jmx_disable_until": "0",
            "jmx_available": "0",
            "jmx_errors_from": "0",
            "jmx_error": "",
            "name": "Template OS Windows",
            "flags": "0",
            "templateid": "10081",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        }
    ],
    "id": 1
}
```

See also 参考

- Host group
- Template
- User macro
- Host interface

Source 源码

CTemplate::get() in *frontends/php/include/classes/api/services/CTemplate.php*. CTemplate::get() 方法可在 frontends/php/include/classes/api/s
中参考。

触发器

This class is designed to work with triggers. 此类用于管理触发器。

Object references: 对象引用：

- Trigger

Available methods: 可用方法：

- trigger.adddependencies - adding new trigger dependencies 添加新的触发器依赖
- trigger.create - creating new triggers 创建新的触发器
- trigger.delete - deleting triggers 删除触发器
- trigger.deletedependencies - deleting trigger dependencies 删除触发器依赖
- trigger.get - retrieving triggers 检索触发器
- trigger.update - updating triggers 更新触发器

> 对象

The following objects are directly related to the `trigger` API. 以下对象与 `trigger`API 直接相关。

Trigger 触发器

The trigger object has the following properties. 触发器对象具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
triggerid	string 字符串 *(	eadonly 只读)* ID of the trigger. 触发器的 ID。
description (required 必须)	string 字符串 Na	e of the trigger. 触发器的名称。
expression (required 必须)	string 字符串 Re	uced trigger expression. 生成的触发表达式。
comments	string 字符串 Ad	itional description of the trigger. 触发器的附加说明。
error	string 字符串 *(	eadonly 只读)* Error text if there have been any problems when updating the state of the trigger. 错误概述，如果在更新触发器的状态时出现任何问题。
flags	integer 整数型 *(	eadonly 只读)* Origin of the trigger. 原始触发器。 Possible values are: 许可值为： 0 - *(default 默认)* a plain trigger; 普通触发器； 4 - a discovered trigger. 自动发现的触发器。

Property 属性 T	pe 类型 Des	ription 说明
lastchange	timestamp 时间戳 *(	eadonly 只读)* Time when the trigger last changed its state. 触发器最后更改其状态的时间。
priority	integer 整数型 Se	erity of the trigger. 触发器的严重性级别。 Possible values are: 许可值为： 0 - *(default* 默认*)* not classified; 未分类； 1 - information; 信息； 2 - warning; 警告； 3 - average; 一般严重； 4 - high; 严重； 5 - disaster. 灾难。
state	integer 整数型 *(	eadonly 只读)* State of the trigger. 触发器的状态。 Possible values: 许可值： 0 - *(default* 默认*)* trigger state is up to date; 触发器状态是最新的； 1 - current trigger state is unknown. 当前的触发器状态是未知的。
status	integer 整数型 Wh	ther the trigger is enabled or disabled. 触发器是否处于启用状态或禁用状态。 Possible values are: 许可值为： 0 - *(default* 默认*)* enabled; 启用； 1 - disabled. 禁用。
templateid	string 字符串 *(	eadonly 只读)* ID of the parent template trigger. 父触发器模板 ID。
type	integer 整数型 Wh	ther the trigger can generate multiple problem events. 触发器是否能够生成多个故障事件。 Possible values are: 许可值为： 0 - *(default* 默认*)* do not generate multiple events; 不生成多个事件。 1 - generate multiple events. 生成多个事件。
url	string 字符串 UR	associated with the trigger. 与触发器相关联的 URL。
value	integer 整数型 *(	eadonly 只读)* Whether the trigger is in OK or problem state. 触发器是否处于正常或故障状态。 Possible values are: 许可值为： 0 - *(default* 默认*)* OK; 正常； 1 - problem. 故障。

Property 属性 T	pe 类型 Des	ription 说明
recovery_mode	integer 整数型 OK	event generation mode. 事件恢复生成模式。 Possible values are: 许可值为： 0 - *(default* 默认*)* Expression; 表达式； 1 - Recovery expression; 恢复表达式； 2 - None. 无。
recovery_expression	string 字符串 Re	uced trigger recovery expression. 生成的触发恢复表达式。
correlation_mode	integer 整数型 OK	event closes. 事件恢复关闭。 Possible values are: 许可值为： 0 - *(default* 默认*)* All problems; 所有故障； 1 - All problems if tag values match. 与标签值匹配的所有故障。
correlation_tag	string 字符串 Ta	for matching. 用于匹配的标签。
manual_close	integer 整数型 Al	ow manual close. 允许手动关闭。 Possible values are: 许可值为： 0 - *(default* 默认*)* No; 不允许； 1 - Yes. 允许。

创建

Description 说明

`object trigger.create(object/array triggers)`

This method allows to create new triggers. 此方法允许创建新的触发器。

Parameters 参数

`(object/array)` Triggers to create. `(object/array)` 需要创建的触发器。Additionally to the standard trigger properties the method accepts the following parameters. 除standard trigger properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
dependencies	array 数组 T	iggers that the trigger is dependent on. 依赖触发的触发器。 The triggers must have the `triggerid` property defined. 触发器必须已定义 `triggerid` 属性。
tags	array 数组 T	igger tags. 触发器标签。

Attention:
The trigger expression has to be given in its expanded form. 指定的触发器表达式必须为展开式。

Return values 返回值

(object) Returns an object containing the IDs of the created triggers under the triggerids property. The order of the returned IDs matches the order of the passed triggers.（object）返回一个对象，该对象包含在 triggerids 属性中已创建触发器的 ID，返回 ID 的顺序与传递触发器的顺序相匹配。

Examples 范例

Creating a trigger 创建触发器

Create a trigger with a single trigger dependency. 创建具有单个触发依赖关系的触发器。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": [
        {
            "description": "Processor load is too high on {HOST.NAME}",
            "expression": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
            "dependencies": [
                {
                    "triggerid": "17367"
                }
            ]
        },
        {
            "description": "Service status",
            "expression": "{Linux server:log[/var/log/system,Service .* has stopped].strlen()}<>0",
            "dependencies": [
                {
                    "triggerid": "17368"
                }
            ],
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 1
}
```

Source 源码

CTrigger::create() in *frontends/php/include/classes/api/services/CTrigger.php*. CTrigger::create() 方法可在 *frontends/php/include/classes/api/se* 中参考。

删除

Description 说明

`object trigger.delete(array triggerIds)`

This method allows to delete triggers. 此方法允许删除触发器。

Parameters 参数

(`array`) IDs of the triggers to delete. (`array`) 需要删除的触发器 ID。

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted triggers under the `triggerids` property. (`object`) 返回一个对象，该对象包含在 `triggerids` 属性中已删除触发器的 ID。

Examples 范例

Delete multiple triggers 删除多个触发器

Delete two triggers. 删除两个触发器。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.delete",
    "params": [
        "12002",
        "12003"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

Source 源码

CTrigger::delete() in *frontends/php/include/classes/api/services/CTrigger.php*. CTrigger::delete() 方法可在 *frontends/php/include/classes/api/se* 中参考。

删除依赖

Description 说明

`object trigger.deletedependencies(string/array triggers)`

This method allows to delete all trigger dependencies from the given triggers. 此方法允许从指定的触发器中删除所有的触发依赖关系。

Parameters 参数

(`string/array`) Triggers to delete the trigger dependencies from. (`string/array`) 需要从触发依赖中删除的触发器。

Return values 返回值

(`object`) Returns an object containing the IDs of the affected triggers under the `triggerids` property. (`object`) 返回一个对象，该对象包含在 `triggerids` 属性中已受影响触发器的 ID。

Examples 范例

Deleting dependencies from multiple triggers 从多个触发器中删除依赖关系

Delete all dependencies from two triggers. 从两个触发器中删除所有依赖关系。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.deleteDependencies",
    "params": [
        {
            "triggerid": "14544"
        },
        {
            "triggerid": "14545"
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14544",
            "14545"
        ]
    },
    "id": 1
}
```

See also 参考

- trigger.update

Source 源码

CTrigger::deleteDependencies() in *frontends/php/include/classes/api/services/CTrigger.php*. CTrigger::deleteDependencies() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

更新

Description 说明

`object trigger.update(object/array triggers)`

This method allows to update existing triggers. 此方法用于更新目前的触发器。

Parameters 参数

(`object/array`) Trigger properties to be updated. (`object/array`) 需要更新的触发器属性。The `triggerid` property must be defined for each trigger, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. `triggerid` 属性必须在每个应用集中已定义，其他所有属性为可选项。只有传递过去的属性会被更新，其他所有属性仍然保持不变。Additionally to the standard trigger properties the method accepts the following parameters. 除standard trigger properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
dependencies	array 数组 T	iggers that the trigger is dependent on. 依赖触发的触发器。 The triggers must have the `triggerid` property defined. 触发器必须已定义 `triggerid` 属性。

Parameter 参数 T	pe 类型 Des	ription 说明
tags	array 数组 T	igger tags. 触发器标签。

Attention:
The trigger expression has to be given in its expanded form. 指定的触发器表达式必须为展开式。

Return values 返回值

（object）Returns an object containing the IDs of the updated triggers under the `triggerids` property.（object）返回一个对象，该对象包含在 `triggerids` 属性中已更新触发器的 ID。

Examples 范例

Enabling a trigger 启用触发器

Enable a trigger, that is, set its status to 0. 启用触发器，即将其状态设置为 0。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing triggers tags 替换触发器标签

Replace tags for trigger. 为触发器替换标签。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "tags": [
            {
                "tag": "service",
                "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
            },
            {
                "tag": "error",
                "value": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

See also 参考

- trigger.adddependencies
- trigger.deletedependencies

Source 源码

CTrigger::update() in *frontends/php/include/classes/api/services/CTrigger.php*. CTrigger::update() 方法可在 *frontends/php/include/classes/api/s* 中参考。

添加依赖

Description 说明

`object trigger.adddependencies(object/array triggerDependencies)`

This method allows to create new trigger dependencies. 此方法允许创建新的触发器依赖关系。

Parameters 参数

(`object/array`) Trigger dependencies to create. (`object/array`) 需要创建的触发器依赖。Each trigger dependency has the following parameters: 每一个触发器依赖具有以下参数：

Parameter 参数 T	pe 类型 Des	ription 说明
triggerid (required 必须)	string 字符串 ID	of the dependent trigger. 依赖触发器的 ID。
dependsOnTriggerid (required 必须)	string 字符串 ID	of the trigger that the trigger depends on. 依赖触发的触发器 ID。

Return values 返回值

(`object`) Returns an object containing the IDs of the dependent triggers under the `triggerids` property. (`object`) 返回一个对象，该对象包含在 `triggerids` 属性中依赖触发器的 ID。

Examples 范例

Add a trigger dependency 添加触发器依赖

Make trigger ”14092” dependent on trigger ”13565.” 触发器”14092” 依赖于触发器”13565”。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.adddependencies",
    "params": {
        "triggerid": "14092",
        "dependsOnTriggerid": "13565"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14092"
        ]
    },
    "id": 1
}
```

See also 参考

- trigger.update

Source 源码

CTrigger::addDependencies() in *frontends/php/include/classes/api/services/CTrigger.php*. CTrigger::addDependencies() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

获取

Description 说明

`integer/array trigger.get(object parameters)`

The method allows to retrieve triggers according to the given parameters. 此方法允许根据指定的参数检索触发器。

Parameters 参数

(`object`) Parameters defining the desired output. (`object`) 定义需要输出的参数。The method supports the following parameters. 该方法支持以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
triggerids	string/array 字符串/数组 Retu	n only triggers with the given IDs. 仅返回指定 ID 的触发器。
groupids	string/array 字符串/数组 Retu	n only triggers that belong to hosts from the given host groups. 仅返回来自指定主机组中所属主机的触发器。
templateids	string/array 字符串/数组 Retu	n only triggers that belong to the given templates. 仅返回指定模板所属的触发器。
hostids	string/array 字符串/数组 Retu	n only triggers that belong to the given hosts. 仅返回指定主机所属的触发器。
itemids	string/array 字符串/数组 Retu	n only triggers that contain the given items. 仅返回包含指定监控项的触发器。
applicationids	string/array 字符串/数组 Retu	n only triggers that contain items from the given applications. 仅返回来自指定应用集中包含监控项的触发器。

Parameter 参数 T	pe 类型 Des	ription 说明
functions	string/array 字符串/数组 Retu	n only triggers that use the given functions. 仅返回使用指定函数的触发器。 Refer to the supported trigger functions page for a list of supported functions. 有关支持的功能列表，请参阅supported trigger functions页面。
group	string 字符串 Re	urn only triggers that belong to hosts from the host group with the given name. 仅返回来自指定名称的主机组中所属主机的触发器。
host	string 字符串 Re	urn only triggers that belong to host with the given name. 仅返回指定名称的所属主机的触发器。
inherited	boolean 布尔值 If	set to `true` return only triggers inherited from a template. 仅返回从模板继承的触发器，如果设置为 `true`。
templated	boolean 布尔值 If	set to `true` return only triggers that belong to templates. 仅返回所属模板的触发器，如果设置为 `true`。
monitored	flag 标记 R	turn only enabled triggers that belong to monitored hosts and contain only enabled items. 仅返回所属被监控主机的已启用触发器，并包含已启用的监控项。
active	flag 标记 R	turn only enabled triggers that belong to monitored hosts. 仅返回所属被监控主机的已启用触发器。
maintenance	boolean 布尔值 If	set to `true` return only enabled triggers that belong to hosts in maintenance. 仅返回在维护中所属主机的已启用触发器，如果设置为 `true`。
withUnacknowledgedEvents	flag 标记 R	turn only triggers that have unacknowledged events. 仅返回事件未确认的触发器。

Parameter 参数 T	pe 类型 Des	ription 说明
withAcknowledgedEvents	flag 标记 R	turn only triggers with all events acknowledged. 仅返回所有事件已确认的触发器。
withLastEventUnacknowledged	flag 标记 R	turn only triggers with the last event unacknowledged. 仅返回最后一个未确认事件的触发器。
skipDependent	flag 标记 S	ip triggers in a problem state that are dependent on other triggers. 依赖其他触发器的触发器处在故障状态时就跳过。Note that the other triggers are ignored if disabled, have disabled items or disabled item hosts. 请注意，如果依赖触发器被禁用，或监控项被禁用，或监控项主机被禁用，那么触发将被忽略。
lastChangeSince	timestamp 时间戳 Re	urn only triggers that have changed their state after the given time. 仅返回指定时间之后变更状态的触发器。
lastChangeTill	timestamp 时间戳 Re	urn only triggers that have changed their state before the given time. 仅返回指定时间之前变更状态的触发器。
only_true	flag 标记 R	turn only triggers that have recently been in a problem state. 仅返回最近处于故障状态的触发器。
min_severity	integer 整数型 Re	urn only triggers with severity greater or equal than the given severity. 仅返回严重级别大于或等于指定严重级别的触发器。
expandComment	flag 标记 E	pand macros in the trigger description. 展开触发器描述中的宏。
expandDescription	flag 标记 E	pand macros in the name of the trigger. 展开触发器名称中的宏。
expandExpression	flag 标记 E	pand functions and macros in the trigger expression. 展开在触发器表达式中的函数和宏。

Parameter 参数 T	pe 类型 Des	ription 说明
selectGroups	query 查询 R	turn the host groups that the trigger belongs to in the `groups` property. 返回在 `groups` 属性中触发器所属的主机组。
selectHosts	query 查询 R	turn the hosts that the trigger belongs to in the `hosts` property. 返回在 `hosts` 属性中触发器所属的主机。
selectItems	query 查询 R	turn items contained by the trigger in the `items` property. 返回在 `items` 属性中触发器所包含的监控项。
selectFunctions	query 查询 R	turn functions used in the trigger in the `functions` property. 返回在 `functions` 属性中在触发器中使用的函数。 The function objects represents the functions used in the trigger expression and has the following properties: 函数对象代表使用在触发器表达式中的函数，并具有以下属性： `functionid` - *(string 字符串)* ID of the function; 函数的 ID； `itemid` - *(string 字符串)* ID of the item used in the function; 使用在函数中的监控项 ID。 `function` - *(string 字符串)* name of the function; 函数的名称； `parameter` - *(string 字符串)* parameter passed to the function. 传递给函数的参数。
selectDependencies	query 查询 R	turn triggers that the trigger depends on in the `dependencies` property. 返回在 `dependencies` 属性中依赖触发的触发器。
selectDiscoveryRule	query 查询 R	turn the low-level discovery rule that created the trigger. 返回创建了触发器的低级别发现规则。

Parameter 参数 T	pe 类型 Des	ription 说明
selectLastEvent	query 查询 R	turn the last significant trigger event in the `lastEvent` property. 返回在 `lastEvent` 属性中最后一个重要触发事件。
selectTags	query 查询 R	turn the trigger tags in `tags` property. 返回在 `tags` 属性中触发器标签。
selectTriggerDiscovery	query 查询 R	turn the trigger discovery object in the `triggerDiscovery` property. 返回在 `triggerDiscovery` 属性中触发器发现对象。The trigger discovery objects links the trigger to a trigger prototype from which it was created. 触发器发现对象将触发器链接到创建它的触发器原型上。 It has the following properties: 触发器发现对象具有以下属性： `parent_triggerid` - (`string` 字符串) ID of the trigger prototype from which the trigger has been created. 创建触发器的触发器原型 ID。

Parameter 参数 T	pe 类型 Des	ription 说明
filter	object 对象 R	turn only those results that exactly match the given filter. 仅返回与指定筛选完全匹配的结果。 Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受一个数组，其中键为属性名称，值为单个值或要匹配值的数组。 Supports additional filters: 支持额外的筛选： `host` - technical name of the host that the trigger belongs to; 触发器所属主机的正式名称。 `hostid` - ID of the host that the trigger belongs to. 触发器所属主机的 ID。
limitSelects	integer 整数型 Li	its the number of records returned by subselects. 限制子查询返回的记录数量。 Applies to the following subselects: 适用于以下子查询： `selectHosts` - results will be sorted by `host`. 以 `host` 分类结果。
sortfield	string/array 字符串/数组 [Sor	](/manual/api/reference_commenta the result by the given properties. 由指定属性分类结果。 Possible values are: `triggerid`, `description`, `status`, `priority`, `lastchange` and `hostname`. 许可值为：`triggerid`, `description`, `status`, `priority`, `lastchange` 和 `hostname`。

Parameter 参数 T	pe 类型 Des	ription 说明
countOutput	boolean 布尔值 Th	se parameters being common for all get methods are described in detail in the reference commentary page. 这些参数十分普遍，适用于所有 get 方法，详情可参考reference commentary。
editable	boolean 布尔值::	
excludeSearch	boolean 布尔值::	
limit	integer 整数型::	
output	query 查询:	:
preservekeys	boolean 布尔值::	
search	object 对象:	:
searchByAny	boolean 布尔值::	
searchWildcardsEnabled	boolean 布尔值::	
sortorder	string/array 字符串/数组:::	
startSearch	boolean 布尔值::	

Return values 返回值

(integer/array) Returns either: 返回两者其中任一：

- an array of objects; 一组对象；
- the count of retrieved objects, if the countOutput parameter has been used. 如果已经使用了 countOutput 参数，则检索对象的计数。

Examples 范例

Retrieving data by trigger ID 根据触发器 ID 检索数据

Retrieve all data and the functions used in trigger "14062". 检索触发器"14062" 中使用的所有数据和功能。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "triggerids": "14062",
        "output": "extend",
        "selectFunctions": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "functions": [
                {
                    "functionid": "13513",
                    "itemid": "24350",
                    "function": "diff",
                    "parameter": "0"
                }
            ],
            "triggerid": "14062",
            "expression": "{13513}>0",
```

```
            "description": "/etc/passwd has been changed on {HOST.NAME}",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "",
            "templateid": "10016",
            "type": "0",
            "state": "0",
            "flags": "0",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0"
        }
    ],
    "id": 1
}
```

Retrieving triggers in problem state 检索在故障状态的触发器

Retrieve the ID, name and severity of all triggers in problem state and sort them by severity in descending order. 检索在问题状态下的所有触发器的 ID，名称和严重性，并按严重性级别按降序分类。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description",
            "priority"
        ],
        "filter": {
            "value": 1
        },
        "sortfield": "priority",
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13907",
            "description": "Zabbix self-monitoring processes < 100% busy",
            "priority": "4"
        },
        {
            "triggerid": "13824",
            "description": "Zabbix discoverer processes more than 75% busy",
            "priority": "3"
        }
    ],
    "id": 1
```

```
}
```

Retrieving a specific trigger with tags 使用标签检索特定的触发器

Retrieve a specific trigger with tags. 使用标签检索特定的触发器。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17578"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17370",
            "description": "Service status",
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

See also 参考

- Discovery rule
- Item
- Host
- Host group

Source 源码

CTrigger::get() in *frontends/php/include/classes/api/services/CTrigger.php*. CTrigger::get() 方法可在 *frontends/php/include/classes/api/services* 中参考。

触发器原型

This class is designed to work with trigger prototypes. 此类用于管理触发器原型。

Object references: 对象引用：

- Trigger prototype

Available methods: 可用方法：

- triggerprototype.create - creating new trigger prototypes 创建新的触发器原型
- triggerprototype.delete - deleting trigger prototypes 删除触发器原型
- triggerprototype.get - retrieving trigger prototypes 检索触发器原型
- triggerprototype.update - updating trigger prototypes 更新触发器原型

> 对象

The following objects are directly related to the `triggerprototype` API. 以下对象与 `triggerprototype`API 直接相关。

Trigger 触发器

The trigger prototype object has the following properties. 触发器原型对象包含以下属性。

Property 属性 T	pe 类型 Des	ription 说明
triggerid	string 字符串 *(	eadonly 只读)* ID of the trigger prototype. 触发器原型的 ID。
description (required 必须)	string 字符串 Na	e of the trigger prototype. 触发器原型的名称。
expression (required 必须)	string 字符串 Re	uced trigger expression. 生成的触发器表达式。
comments	string 字符串 Ad	itional comments to the trigger prototype. 触发器原型的附加注释。
priority	integer 整数型 Se	erity of the trigger prototype. 触发器原型的严重级别。 Possible values: 许可值： 0 - *(default* 默认*)* not classified; 未分类； 1 - information; 信息； 2 - warning; 警告； 3 - average; 一般严重； 4 - high; 严重； 5 - disaster. 灾难。
status	integer 整数型 Wh	ther the trigger prototype is enabled or disabled. 触发器原型是否在启用状态或禁用状态。 Possible values: 许可值： 0 - *(default* 默认*)* enabled; 已启用； 1 - disabled. 已禁用。
templateid	string 字符串 *(	eadonly 只读)* ID of the parent template trigger prototype. 触发器原型父模板的 ID。

Property 属性 T	pe 类型 Des	ription 说明
type	integer 整数型 Wh	ther the trigger prototype can generate multiple problem events. 触发器原型是否可以生成多个异常事件。 Possible values: 许可值： 0 - *(default* 默认*)* do not generate multiple events; 不生成多个事件； 1 - generate multiple events. 生成多个事件。
url	string 字符串 UR	associated with the trigger prototype. 关联到触发器原型的 URL。
recovery_mode	integer 整数型 OK	event generation mode. 正常事件生成模式。 Possible values are: 许可值为： 0 - *(default* 默认*)* Expression; 表达式； 1 - Recovery expression; 恢复表达式； 2 - None. 无。
recovery_expression	string 字符串 Re	uced trigger recovery expression. 生成的触发器恢复表达式。
correlation_mode	integer 整数型 OK	event closes. 正常事件关闭。 Possible values are: 许可值为： 0 - *(default* 默认*)* All problems; 所有异常； 1 - All problems if tag values match. 匹配标签值的所有异常。
correlation_tag	string 字符串 Ta	for matching. 匹配的标签。
manual_close	integer 整数型 Al	ow manual close. 允许手动关闭。 Possible values are: 许可值为： 0 - *(default* 默认*)* No; 不允许； 1 - Yes. 允许。

创建

Description

`object triggerprototype.create(object/array triggerPrototypes)`

This method allows to create new trigger prototypes.

Parameters 参数

`(object/array)` Trigger prototypes to create. `(object/array)` 需要创建的触发器原型。Additionally to the standard trigger prototype properties the method accepts the following parameters. 除standard trigger prototype properties之外，此方法还接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
dependencies	array 数组 T	iggers and trigger prototypes that the trigger prototype is dependent on. 依赖触发器原型的触发器和触发器原型。 The triggers must have the `triggerid` property defined. 触发器必须已定义 `triggerid` 属性。
tags	array 数组 T	igger prototype tags. 触发器原型标签。

Attention:
The trigger expression has to be given in its expanded form and must contain at least one item prototype. 指定的触发器表达式必须为展开式，并且必须包含至少一个监控项原型。

Return values 返回值

(`object`) Returns an object containing the IDs of the created trigger prototypes under the `triggerids` property. The order of the returned IDs matches the order of the passed trigger prototypes. (`object`) 返回一个对象，该对象包含在 `triggerids` 属性中已创建触发器原型的 ID，返回 ID 的顺序与传递触发器原型的顺序相匹配。

Examples 范例

Creating a trigger prototype 创建触发器原型

Create a trigger prototype to detect when a file system has less than 20% free disk space. 创建一个触发器原型来检测磁盘剩余空间是否小于 20%。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.create",
    "params": {
        "description": "Free disk space is less than 20% on volume {#FSNAME}",
        "expression": "{Zabbix server:vfs.fs.size[{#FSNAME},pfree].last()}<20",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17372"
        ]
    },
    "id": 1
}
```

Source 源码

CTriggerPrototype::create() in *frontends/php/include/classes/api/services/CTriggerPrototype.php*. CTriggerPrototype::create() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

删除

Description 说明

`object triggerprototype.delete(array triggerPrototypeIds)`

This method allows to delete trigger prototypes. 此方法允许删除触发器原型。

Parameters 参数

(`array`) IDs of the trigger prototypes to delete. (`array`) 需要删除的触发器原型 ID。

Return values 返回值

(`object`) Returns an object containing the IDs of the deleted trigger prototypes under the `triggerids` property. (`object`) 返回一个对象，该对象包含在 `triggerids` 属性中已删除触发器原型的 ID。

Examples 范例

Deleting multiple trigger prototypes 删除多个触发器原型

Delete two trigger prototypes. 删除两个触发器原型。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.delete",
    "params": [
        "12002",
        "12003"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

Source 源码

CTriggerPrototype::delete() in *frontends/php/include/classes/api/services/CTriggerPrototype.php*. CTriggerPrototype::delete() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

更新

Description 说明

`object triggerprototype.update(object/array triggerPrototypes)`

This method allows to update existing trigger prototypes. 此方法允许更新已有的触发器原型。

Parameters 参数

(`object/array`) Trigger prototype properties to be updated. (`object/array`) 需要更新的触发器原型Trigger prototype properties。The `triggerid` property must be defined for each trigger prototype, all other properties are optional. Only the passed

properties will be updated, all others will remain unchanged. `triggerid` 属性必须在每个触发器原型中已定义，其他所有属性为可选项。只有传递过去的属性会被更新，其他所有属性仍然保持不变。Additionally to the standard trigger prototype properties the method accepts the following parameters. 除standard trigger prototype properties之外，该方法接受以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
dependencies	array 数组 T	iggers and trigger prototypes that the trigger prototype is dependent on. 依赖触发器原型的触发器和触发器原型。 The triggers must have the `triggerid` property defined. 触发器必须已定义 `triggerid` 属性。
tags	array 数组 T	igger prototype tags. 触发器标签。

Attention:
The trigger expression has to be given in its expanded form and must contain at least one item prototype. 指定的触发器表达式必须为展开式，并且必须包含至少一个监控项原型。

Return values 返回值

（`object`）Returns an object containing the IDs of the updated trigger prototypes under the `triggerids` property.（`object`）返回一个对象，该对象包含在 `triggerids` 属性中已更新触发器原型的 ID。

Examples 范例

Enabling a trigger prototype 启用触发器原型

Enable a trigger prototype, that is, set its status to 0. 启用一个触发器原型，即将其状态设置为 0。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing trigger prototype tags 替换触发器原型标签

Replace tags for one trigger prototype. 为触发器原型替换标签。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
```

```
        "triggerid": "17373",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17373"
        ]
    },
    "id": 1
}
```

Source 源码

CTriggerPrototype::update() in *frontends/php/include/classes/api/services/CTriggerPrototype.php*. CTriggerPrototype::update() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

获取

Description 说明

`integer/array triggerprototype.get(object parameters)`

The method allows to retrieve trigger prototypes according to the given parameters. 此方法允许根据指定的参数检索触发器原型。

Parameters 参数

(`object`) Parameters defining the desired output. (`object`) 定义需要输出的参数。The method supports the following parameters. 该方法支持以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
active	flag 标记 R	turn only enabled trigger prototypes that belong to monitored hosts. 仅返回所属被监控主机的已启用触发器原型。
applicationids	string/array 字符串/数组 Retu	n only trigger prototypes that contain items from the given applications. 仅返回来自指定应用集中包含监控项的触发器原型。
discoveryids	string/array 字符串/数组 Retu	n only trigger prototypes that belong to the given LLD rules. 仅返回所属指定低级别发现规则的触发器原型。

Parameter 参数 T	pe 类型 Des	ription 说明
functions	string/array 字符串/数组 Retu	n only triggers that use the given functions. 仅返回使用指定函数的触发器。 Refer to the supported trigger functions page for a list of supported functions. 有关支持的功能列表，请参阅supported trigger functions 页面。
group	string 字符串 Re	urn only trigger prototypes that belong to hosts from the host groups with the given name. 仅返回来自指定名称的主机组中所属主机的触发器原型。
groupids	string/array 字符串/数组 Retu	n only trigger prototypes that belong to hosts from the given host groups. 仅返回来自指定主机组中所属主机的触发器原型。
host	string 字符串 Re	urn only trigger prototypes that belong to hosts with the given name. 仅返回指定名称的所属主机的触发器原型。
hostids	string/array 字符串/数组 Retu	n only trigger prototypes that belong to the given hosts. 仅返回指定主机所属的触发器原型。
inherited	boolean 布尔值 If	set to `true` return only trigger prototypes inherited from a template. 仅返回从模板继承的触发器原型，如果设置为 `true`。
maintenance	boolean 布尔值 If	set to `true` return only enabled trigger prototypes that belong to hosts in maintenance. 仅返回在维护中所属主机的已启用触发器原型，如果设置为 `true`。
min_severity	integer 整数型 Re	urn only trigger prototypes with severity greater or equal than the given severity. 仅返回严重级别大于或等于指定严重级别的触发器原型。

Parameter 参数 T	pe 类型 Des	ription 说明
monitored	flag 标记 R	turn only enabled trigger prototypes that belong to monitored hosts and contain only enabled items. 仅返回所属被监控主机的已启用触发器原型，并包含已启用的监控项。
templated	boolean 布尔值 If	set to `true` return only trigger prototypes that belong to templates. 仅返回所属模板的触发器原型，如果设置为 `true`。
templateids	string/array 字符串/数组 Retu	n only trigger prototypes that belong to the given templates. 仅返回指定模板所属的触发器原型。
triggerids	string/array 字符串/数组 Retu	n only trigger prototypes with the given IDs. 仅返回指定 ID 的触发器原型。
expandExpression	flag 标记 E	pand functions and macros in the trigger expression. 展开在触发器原型表达式中的函数和宏。
selectDiscoveryRule	query 查询 R	turn the LLD rule that the trigger prototype belongs to. 返回触发器原型所属的低级别发现规则。

Parameter 参数 T	pe 类型 Des	ription 说明
selectFunctions	query 查询 R	turn functions used in the trigger prototype in the `functions` property. 返回在 `functions` 属性中在触发器中使用的函数。 The function objects represents the functions used in the trigger expression and has the following properties: 函数对象代表使用在触发器表达式中的函数，并具有以下属性： `functionid` - *(string 字符串)* ID of the function; 函数的 ID； `itemid` - *(string 字符串)* ID of the item used in the function; 使用在函数中的监控项 ID； `function` - *(string 字符串)* name of the function; 函数的名称； `parameter` - *(string 字符串)* parameter passed to the function. 传递给函数的参数。
selectGroups	query 查询 R	turn the host groups that the trigger prototype belongs to in the `groups` property. 返回在 `groups` 属性中触发器原型所属的主机组。
selectHosts	query 查询 R	turn the hosts that the trigger prototype belongs to in the `hosts` property. 返回在 `hosts` 属性中触发器所属的主机。
selectItems	query 查询 R	turn items and item prototypes used the trigger prototype in the `items` property. 返回在 `items` 属性中触发器所包含的监控项。
selectDependencies	query 查询 R	turn trigger prototypes and triggers that the trigger prototype depends on in the `dependencies` property. 返回在 `dependencies` 属性中依赖触发器原型的触发器原型和触发器。
selectTags	query 查询 R	turn the trigger prototype tags in `tags` property. 返回在 `tags` 属性中触发器原型标签。

Parameter 参数 T	pe 类型 Des	ription 说明
filter	object 对象 R	turn only those results that exactly match the given filter. 仅返回与指定筛选完全匹配的结果。 Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受一个数组，其中键为属性名称，值为单个值或要匹配值的数组。 Supports additional filters: 支持额外的筛选： `host` - technical name of the host that the trigger prototype belongs to; 触发器原型所属主机的正式名称。 `hostid` - ID of the host that the trigger prototype belongs to. 触发器原型所属主机的 ID。
limitSelects	integer 整数型 Li	its the number of records returned by subselects. 限制子查询返回的记录数量。 Applies to the following subselects: 适用于以下子查询： `selectHosts` - results will be sorted by `host`. 以 `host` 分类结果。
sortfield	string/array 字符串/数组 Sort	the result by the given properties. 由指定属性分类结果。 Possible values are: `triggerid`, `description`, `status` and `priority`. 许可值为：`triggerid`, `description`, `status` 和 `priority`。

Parameter 参数 T	pe 类型 Des	ription 说明
countOutput	boolean 布尔值 Th	se parameters being common for all get methods are described in detail in the reference commentary. 这些参数十分普遍，适用于所有 get 方法，详情可参考reference commentary。
editable	boolean 布尔值::	
excludeSearch	boolean 布尔值::	
limit	integer 整数型::	
output	query 查询:	:
preservekeys	boolean 布尔值::	
search	object 对象:	:
searchByAny	boolean 布尔值::	
searchWildcardsEnabled	boolean 布尔值::	
sortorder	string/array 字符串/数组:::	
startSearch	boolean 布尔值::	

Return values 返回值

(integer/array) Returns either: 返回两者其中任一：

- an array of objects; 一组对象；
- the count of retrieved objects, if the countOutput parameter has been used. 如果已经使用了 countOutput 参数，则检索对象的计数。

Examples 范例

Retrieve trigger prototypes from an LLD rule 从低级别发现规则中检索触发器原型

Retrieve all trigger prototypes and their functions from an LLD rule. 从低级别发现规则中检索所有的触发器原型和相关函数。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": "extend",
        "selectFunctions": "extend",
        "discoveryids": "22450"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "functions": [
                {
                    "functionid": "12598",
                    "itemid": "22454",
                    "function": "last",
                    "parameter": "0"
                }
            ],
            "triggerid": "13272",
            "expression": "{12598}<20",
```

```
            "description": "Free inodes is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "priority": "2",
            "comments": "",
            "templateid": "0",
            "type": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0"
        },
        {
            "functions": [
                {
                    "functionid": "13500",
                    "itemid": "22686",
                    "function": "last",
                    "parameter": "0"
                }
            ],
            "triggerid": "13266",
            "expression": "{13500}<201",
            "description": "Free disk space is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "priority": "2",
            "comments": "",
            "templateid": "0",
            "type": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0"
        }
    ],
    "id": 1
}
```


Retrieving a specific trigger prototype with tags 根据标签检索特定的触发器原型

Request 请求:


```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ]
        "selectTags": "extend",
        "triggerids": [
            "17373"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17373",
            "description": "Free disk space is less than 20% on volume {#FSNAME}",
            "tags": [
                {
                    "tag": "volume",
                    "value": "{#FSNAME}"
                },
                {
                    "tag": "type",
                    "value": "{#FSTYPE}"
                }
            ]
        }
    ],
    "id": 1
}
```


See also 参考

- Discovery rule
- Item
- Host
- Host group

Source 源码

CTriggerPrototype::get() in *frontends/php/include/classes/api/services/CTriggerPrototype.php*. CTriggerPrototype::get() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

趋势

This class is designed to work with trend data. 此类用于处理趋势数据。

Object references: 对象引用：

- Trend

Available methods: 可用方法：

- trend.get - retrieving trends 检索趋势数据。

> 对象

The following objects are directly related to the `trend` API. 以下对象与 `trend` API 直接相关。

Note:
Trend objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the API. 趋势对象根据监控项类型的信息而有所不同，它们由 Zabbix server 创建，不能通过 API 进行修改。

Float trend 浮点型趋势

The float trend object has the following properties. 浮点型趋势对象具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
clock	timestamp 时间戳 Ti	e when that value was received. 收取该值的时间。
itemid	string 字符串 ID	of the related item. 相关监控项 ID。
num	integer 整数型 Nu	ber of values within this hour. 在该小时内值的数量。
value_min	float 浮点型 Ho	rly minimum value. 每小时最小值。

Property 属性 T	pe 类型 Des	ription 说明
value_avg	float 浮点型 Ho	rly average value. 每小时平均值。
value_max	float 浮点型 Ho	rly maximum value. 每小时最大值。

Integer trend 整数型趋势

The integer trend object has the following properties. 整数型趋势对象具有以下属性。

Property 属性 T	pe 类型 Des	ription 说明
clock	timestamp 时间戳 Ti	e when that value was received. 收取该值的时间。
itemid	string 字符串 ID	of the related item. 相关监控项 ID。
num	integer 整数型 Nu	ber of values within this hour. 在该小时内值的数量。
value_min	integer 整数型 Ho	rly minimum value. 每小时最小值。
value_avg	integer 整数型 Ho	rly average value. 每小时平均值。
value_max	integer 整数型 Ho	rly maximum value. 每小时最大值。

获取

Description 说明

`integer/array trend.get(object parameters)`

The method allows to retrieve trend data according to the given parameters. 该方法用于根据指定的参数检索趋势数据。

Parameters 参数

(`object`) Parameters defining the desired output. (`object`) 定义所需输出的参数。The method supports the following parameters. 该方法提供以下参数。

Parameter 参数 T	pe 类型 Des	ription 说明
itemids	string/array 字符串/数组 Retu	n only trends with the given item IDs. 仅返回指定监控项 ID 的趋势。
time_from	timestamp 时间戳 Re	urn only values that have been collected after or at the given time. 仅返回指定时间（包含）之后已采集的值。
time_till	timestamp 时间戳 Re	urn only values that have been collected before or at the given time. 仅返回指定时间（包含）之前已采集的值。
countOutput	boolean 布尔值 Co	nt the number of retrieved objects. 计算检索对象的数量。
limit	integer 整数型 Li	it the amount of retrieved objects. 限制检索对象的数量。
output	query 查询 S	t fields to output. 输出设置的字段。

Return values 返回值

(`integer/array`) Returns either: 返回两者其中任一：

- an array of objects; 一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用了 `countOutput` 参数，则检索对象的计数。

Examples 范例

Retrieving item trend data 检索监控项趋势数据

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "trend.get",
    "params": {
        "output": [
            "itemid",
            "clock",
            "num",
            "value_min",
            "value_avg",
            "value_max",
        ],
        "itemids": [
            "23715"
        ],
        "limit": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23715",
            "clock": "1446199200",
            "num": "60",
            "value_min": "0.1650",
            "value_avg": "0.2168",
            "value_max": "0.3500",
        }
    ],
    "id": 1
}
```

Source 源码

CTrend::get() in *frontends/php/include/classes/api/services/CTrend.php*. CTrend::get() 方法可在 *frontends/php/include/classes/api/services/CTr* 中参考。

问题

This class is designed to work with problems. 这个类设计用于描述问题。

Object references:

- Problem

Available methods:

- problem.get - retrieving problems

> 对象

Note:
problems are created by the Zabbix server and cannot be modified via the API.

Note:
问题是由 Zabbix 服务器创建的，不能通过 API 进行修改。

The problem object has the following properties. 问题对象拥有以下属性

Property	Type	Description
eventid	string	ID of the problem event. 问题事件的 ID
source	integer	Type of the problem event. Possible values: 0 - event created by a trigger; 3 - internal event. 问题事件类型。 \\可能的值：\\0-触发器创建的时间 3-内部事件
object	integer	Type of object that is related to the problem event. Possible values for trigger events: 0 - trigger. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule. 与问题事件相关的对象类型。 触发器时间可能的值： 0-触发器 内部事件可能的值： 0-触发器 4-监控项 5-LLD 规则
objectid	string	ID of the related object. 关联对象的 ID
clock	timestamp	Time when the problem event was created. 问题事件创建的时间
ns	integer	Nanoseconds when the problem event was created. 问题事件创建的纳秒时间
r_eventid	string	Recovery event ID. 恢复时间的 ID
r_clock	timestamp	Time when the recovery event was created. 恢复事件创建的时间
r_ns	integer	Nanoseconds when the recovery event was created. 恢复事件创建的纳秒时间
correlationid	string	Correlation rule ID if this event was recovered by global correlation rule. 事件被全局的关联规则恢复，关联规则的 ID
userid	string	User ID if the problem was manually closed. 手动关闭问题的用户 ID
name	string	Resolved problem name. 解决问题名称
acknowledged	integer	Acknowledge state for problem. Possible values: 0 - not acknowledged; 1 - acknowledged. 问题知晓状态 \\可能的值： 0-不知道 1-知道

Property	Type	Description
severity	integer	Problem current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster. 问题当前级别 \\可能的值： 0-未定义 1-信息 2-警告 3-一般严重 4-严重 5-灾难

获取

Description 描述

`integer/array problem.get(object parameters)`

The method allows to retrieve problems according to the given parameters. 此方法允许根据给定参数检索符合条件的问题

Parameters 参数

`(object)` Parameters defining the desired output. `(object)` 定义所需输出的参数

The method supports the following parameters. 此方法支持一下参数

Parameter	Type	Description
eventids	string/array	Return only problems with the given IDs. 仅返回所给IDs 的问题
groupids	string/array	Return only problems created by objects that belong to the given host groups. 仅返回所属给定主机组对象的问题
hostids	string/array	Return only problems created by objects that belong to the given hosts. 仅返回所给定主机对象的问题
objectids	string/array	Return only problems created by the given objects. 仅返回所给对象创建的问题
applicationids	string/array	Return only problems created by objects that belong to the given applications. Applies only if object is trigger or item. 只返回属于给定应用程序的对象创建的问题。仅当对象是触发器或监控项时才应用。
source	integer	Return only problems with the given type. Refer to the problem event object page for a list of supported event types. Default: 0 - problem created by a trigger. 只返回给定类型的问题 跳转到problem event object page 用于支持事件类型的列表。 \\默认：0 - 触发器创建的问题
object	integer	Return only problems created by objects of the given type. Refer to the problem event object page for a list of supported object types. Default: 0 - trigger. 只返回由给定类型的对象创建的问题 \\跳转到problem event object page 用于支持事件类型的列表 \\默认：0-触发器

Parameter	Type	Description
acknowledged	boolean	`true` - return acknowledged problems only; `false` - unacknowledged only. `true`-返回已知晓的问题 返回未知晓的问题
severities	integer/array	Return only problems with given event severities. Applies only if object is trigger. 只返回给定事件严重程度的问题。仅当对象是触发器时才应用。
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or. 规则标签搜索。 \\可能的值： 0 - (默认) 与/或；2 - 或
tags	array of objects	Return only problems with given tags. Exact match by tag and case-insensitive search by value and operator. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all problems. Possible operator types: 0 - (default) Like; 1 - Equal. 只返回给定标签的问题。按标记精确匹配，按值和运算符不区分大小写搜索。 格式：`[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. 空数组返回所有问题 可能的分隔类型： 0 - (默认) 相似 1 - 相等
recent	string	true - return PROBLEM and recently RESOLVED problems (depends on Display OK triggers for N seconds) Default: false - UNRESOLVED problems only true - 返回问题和最近已解决的问题（依赖于最近 N 秒显示 OK 的触发器） 默认：false - 仅真正未处理的问题
eventid_from	string	Return only problems with IDs greater or equal to the given ID. 只返回 ID 大于或等于给定 ID 的问题。
eventid_till	string	Return only problems with IDs less or equal to the given ID. 只返回 ID 小于或等于给定 ID 的问题。
time_from	timestamp	Return only problems that have been created after or at the given time. 仅返回问题创建时间在所给时间之后的问题
time_till	timestamp	Return only problems that have been created before or at the given time. 仅返回问题创建时间在所给时间之前的问题

Parameter	Type	Description
selectAcknowledges	query	Return problem's updates in the `acknowledges` property. Problem updates are sorted in reverse chronological order. The problem update object has the following properties: `acknowledgeid` - (`string`) update's ID; `userid` - (`string`) ID of the user that updated the event; `eventid` - (`string`) ID of the updated event; `clock` - (`timestamp`) time when the event was updated; `message` - (`string`) text of the message; `action` - (`integer`) update action that was performed see event.acknowledge; `old_severity` - (`integer`) event severity before this update action; `new_severity` - (`integer`) event severity after this update action; Supports `count`. 在 `acknowledges` 属性中返回问题的更新。问题更新按时间倒序排序。 问题更新对象拥有以下属性：`acknowledgeid` - (`string`) 更新的 ID 'userid" - (`string`) 更新事件的用户的 ID; `eventid` - (`string`) 更新时间的 ID `clock` - (`timestamp`) 事件更新的时间 `message` - (`string`) 文本信息 `action` - (`integer`) 更新动作请参照event.acknowledge; `old_severity` - (`integer`) 更新动作前事件的级别 `new_severity` - (`integer`) 更新动作后事件级别
selectTags	query	Return problem's tags. Output format: `[{"tag": "<tag>", "value": "<value>"}, ...]`. 返回问题标签，输出格式：`[{"tag": "<tag>", "value": "<value>"}, ...]`
sortfield	string/array	Sort the result by the given properties. Possible values are: `eventid`. 根据给定的属性对结果进行排序。 可能的的值：`eventid`
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page. 这个属性使用所有的 `get` 方法，详细定义在reference commentary页
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values 返回值

(integer/array) Returns either:

- an array of objects;
- 一个数组对象
- the count of retrieved objects, if the countOutput parameter has been used.
- 返回检索到对象的数量，如果 countOutput 参数被引用

Examples 示例如下

Retrieving trigger problem events 返回触发器问题事件

Retrieve recent events from trigger "15112." 返回最近触发器 id 是 15112 的事件

Request:

```
{
    "jsonrpc": "2.0",
    "method": "problem.get",
    "params": {
        "output": "extend",
        "selectAcknowledges": "extend",
        "selectTags": "extend",
        "objectids": "15112",
        "recent": "true",
        "sortfield": ["eventid"],
        "sortorder": "DESC"
    },
    "auth": "67f45d3eb1173338e1b1647c4bdc1916",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "1245463",
            "source": "0",
            "object": "0",
            "objectid": "15112",
            "clock": "1472457242",
            "ns": "209442442",
            "r_eventid": "1245468",
            "r_clock": "1472457285",
            "r_ns": "125644870",
            "correlationid": "0",
            "userid": "1",
            "name": "Zabbix agent on localhost is unreachable for 5 minutes",
            "acknowledged": "1",
            "severity": "3",
            "acknowledges": [
                {
                    "acknowledgeid": "14443",
                    "userid": "1",
                    "eventid": "1245463",
                    "clock": "1472457281",
                    "message": "problem solved",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0"
                }
            ],
            "tags": [
                {
                    "tag": "test tag",
                    "value": "test value"
```

```
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Alert
- Item
- Host
- LLD rule
- Trigger

Source

CEvent::get() in *frontends/php/include/classes/api/services/CProblem.php*.

9. 关联

这个类是设计用于关联。

对象引用：

- 关联

可用的方法：

- correlation.create - 创建新的关联
- correlation.delete - 删除关联
- correlation.get - 获取关联
- correlation.update - 更新关联

> 关联对象

下列对象与关联 API 直接相关。

关联

关联对象具有以下属性。

属性类	描述	
correlationid	字符串 *（	读）* 关联的 ID。
name （必填）	字符串 关 丨 的名称。	
description	字符串 关 丨 的描述。	
status	整数 丨 联是启用还是禁用。 可能的值有： 0 - （默认）启用的； 1 - 禁用的。	

关联操作

关联操作对象定义了当一个关联被执行时，该操作的行为表现。它具有如下属性。

属性类	描述	
type （必填）	整数操	类型。 可能的值： 0 - 关闭旧事件。 1 - 关闭新事件。

关联过滤

关联过滤对象定义了配置关联操作时，必须满足的一组条件。它具有如下属性。

属性类	描述	
evaltype (必填)	整数过	条件评价方法。 可能的值： 0 - 与/或； 1 - 与； 2 - 或； 3 - 自定义表达式。
conditions (必填)	数组用	过滤结果的一组过滤条件。

属性类	描述
eval_formula	字符串 * （读）* 生成的表达式将用于评估过滤条件。该表达式包含通过“formulaid”引用特定筛选条件的ID。对于具有自定义表达式的筛选，`eval_formula`的值等于`formula`的值。

属性类	描述
formula	字符串用户义的表达式，用于具有自定义表达式的过滤评估条件。该表达式必须包含通过“formulaid”引用特定筛选条件的ID。表达式中使用的ID必须与过滤条件中定义的ID

属性类	描述

关联过滤条件

关联过滤条件对象定义了在运行关联操作前必须检查的特定条件。

属性类	描述	
type (必填)	整数条	类型。 可能的值： 0 - 旧事件标签； 1 - 新事件标签； 2 - 新事件主机组； 3 - 事件标签对； 4 - 旧事件标签值； 5 - 新事件标签值。
tag	字符串事件	签(旧或新)。 条件类型是：0, 1, 4, 5 时需要。
groupid	字符串主机	ID。 条件类型是：2 时需要。
oldtag	字符串旧事	标签。 条件类型是：3 时需要。

属性类	描述	
newtag	字符串新事	标签。 条件类型是：3 时需要。
value	字符串事件	签(旧或新)值。 条件类型是：4, 5 时需要。
formulaid	字符串任意	唯一 ID，用于引用一个自定义表达式中的条件。只能包含大写字母。 当修改过滤条件时，该 ID 必须由用户定义，但以后请求它们时会重新生成。
operator	整数条	运算符。 条件类型是：2, 4, 5 时需要。

Note:
为了更好地了解如何使用具有各种类型的表达式的过滤，请参阅correlation.get 方法和correlation.create 方法页面上的示例。

以下运算符和值都支持每种条件类型。

条件条	名称支持的运算	期望的值	
2	主机组 =,	<> 主机	ID。
4	旧事件标签值 =, &I	;>, like, not like 字符串	
5	新事件标签值 =, &I	;>, like, not like 字符串	

创建

描述

`object correlation.create(object/array correlations)`

这种方法允许创建新的关联。

参数

(object/array) 要创建的关联。

另外，对于标准关联属性，该方法还接受以下参数。

参数类	描述	
operations（必填）	数组与	建关联相关的操作。
filter（必填）	对象与	关联 '相关的过滤对象。

返回值

(object) 返回一个对象，该对象包含 "correlationids" 属性下创建的关联的 ID。返回的 ID 的顺序与所传递的关联的顺序相匹配。

示例

创建一个新的事件标签关联

使用具有一个条件和一个操作的评估方法 AND/OR 创建一个关联。默认情况下，这个关联将被启用。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new event tag correlation",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "type": 1,
                    "tag": "ok"
                }
            ]
        },
        "operations": [
            {
                "type": 0
            }
        ]
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

使用一个自定义表达式过滤

创建使用自定义筛选条件的关联。公式 id A 或 B 是任意选择的。条件类型为 “主机组”，操作符为 “<>”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new host group correlation",
        "description": "a custom description",
        "status": 0,
        "filter": {
            "evaltype": 3,
            "formula": "A or B",
            "conditions": [
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "A"
                },
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "B"
                }
            ]
        },
        "operations": [
            {
                "type": 1
            }
        ]
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "2"
        ]
    },
    "id": 1
}
```

参见

- 关联过滤
- 关联操作

来源

CCorrelation::create() in *frontends/php/include/classes/api/services/CCorrelation.php*.

删除

描述

`object correlation.delete(array correlationids)`

这个方法允许删除关联。

参数

(array) 要删除的关联的 ID。

返回值

(object) 返回一个对象，该对象包含 “correlationids” 属性下删除的关联的 ID。

示例

删除多个关联

删除 2 个关联。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlaionids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

来源

CCorrelation::delete() in *frontends/php/include/classes/api/services/CCorrelation.php*.

更新

描述

`object correlation.update(object/array correlations)`

这个方法允许更新已存在的关联。

参数

(object/array) 要更新的关联的属性。

必须为每个关联定义 correlationid 属性，其它的属性都是可选的。只有传递的属性会被更新，其它属性都将保持不变。

另外，对于标准关联属性，该方法接受以下参数。

参数类	描述	
filter	对象替	当前筛选的关联筛选对象。
operations	数组替	已存在的操作的关联操作。

返回值

（object）返回一个对象，该对象包含 “correlationids” 属性下更新的关联的 ID。

示例

禁用关联

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "status": "1"
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

替代条件，但评估方法不变

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "filter": {
            "conditions": [
                {
                    "type": 3,
                    "oldtag": "error",
                    "newtag": "ok"
                }
            ]
        }
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
```

```
            ]
        },
        "id": 1
}
```

参见

- 关联过滤
- 关联操作

来源

CCorrelation::update() in *frontends/php/include/classes/api/services/CCorrelation.php*.

获取

描述

`integer/array correlation.get(object parameters)`

这个方法允许根据给定的参数检索关联。

参数

`(object)` 定义需要输出的参数。

这个方法支持以下参数。

参数类	描述	
correlationids	字符串/数组只返回拥	给定ID的关联。
selectFilter	查询返	`filter`属性中的关联过滤。
selectOperations	查询返	`operations`属性中的关联操作。

参数类	描述	
sortfield	字符串/数组根据给定	属性对结果进行排序。 可能的值有：correlationid、name 和 status。
countOutput	布尔值在 [	eference_comm 中详细描述了所有 get 方法的常见参数。
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(integer/array) 返回：

- 一个对象数组；
- 如果使用了 countOutput 参数，被检索的对象的数量。

示例

检索关联

检索所有具有相关条件和操作的已配置过的关联。过滤使用 “AND/OR” 的评估类型，因此 formula 属性为空，且 eval_formula 将自动生成。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectFilter": "extend"
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "correlationid": "1",
            "name": "Correlation 1",
            "description": "",
            "status": "0",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "type": "3",
                        "oldtag": "error",
                        "newtag": "ok",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A"
            },
            "operations": [
                {
                    "type": "0"
                }
            ]
        }
    ],
    "id": 1
}
```

参见

- 关联过滤
- 关联操作

来源

CCorrelation::get() in *frontends/php/include/classes/api/services/CCorrelation.php*.

10. 仪表板

这个类被设计用于仪表板。

对象引用：

- 仪表板
- 仪表板小组件
- 仪表板小组件字段
- 仪表板用户组

- 仪表板用户

可用的方法：

- dashboard.create - 创建新的仪表板
- dashboard.delete - 删除仪表板
- dashboard.get - 检索仪表板
- dashboard.update - 更新仪表板

> 仪表板对象

下列对象与仪表板 API 直接相关。

仪表板

仪表板对象具有以下属性：

属性类	描述	
dashboardid	字符串 *（	读）* 仪表板 ID。
name	字符串仪表	名称。
（必填）		
userid	字符串仪表	所属的用户 ID。
private	整数仪	板类型。 可能的值： 0 - 公用仪表板； 1 - （默认的）私有仪表板。

仪表板小部件

仪表板小部件对象具有以下属性：

属性类	描述	
widgetid	字符串 *（	读）* 仪表板小部件的 ID。

属性类	描述	
type (必填)	字符串仪表	小部件的类型。 可能的值： actionlog - 动作日志； clock - 时钟； dataover - 数据预览； discovery - 发现状态； favgraphs - 常用的图形； favmaps - 常用的拓扑图； favscreens - 常用的聚合图形； graph - 图形； problemhosts - 有问题的主机； map - 拓扑图； navtree - 拓扑图导航树； plaintext - 纯文本； problems - 问题； systeminfo - 系统信息；

属性类	描述	
name	字符串自定	的小部件名称。
x	整数仪	板左侧的水平位置。 有效值范围从 0 到 23。
y	整数仪	板顶部的垂直位置。 有效值范围从 0 到 62。
width	整数小	件的宽度。 有效值范围从 1 到 24。
height	整数小	件的高度； 有效值范围从 1 到 32。
view_mode	整数视	模式 可能的值: 0 - (默认) 默认视图; 1 - 隐藏 header 视图
fields	数组 [	表板小组件字段](object#dashbo 对象的数组。

仪表板小部件字段

仪表板小部件字段对象具有以下属性：

属性类	描述	
type (必填)	整数小	件字段的类型。 可能的值： 0 - 整数； 1 - 字符串； 2 - 主机组； 3 - 主机； 4 - 监控项； 6 - 图形； 8 - 拓扑图；
name	字符串小部	字段的名称。
value (必填)	混合型取决	小部件字段的类型。

仪表板用户组

基于用户组的仪表板权限列表。其具有以下属性：

属性类	描述	
usrgrpid （必填）	字符串用户	ID。
permission （必填）	整数权	类型。 可能的值： 2 - 只读； 3 - 读-写。

仪表板用户

基于用户的仪表板权限列表。其具有以下属性：

属性类	描述	
userid （必填）	字符串用户	ID。

属性类	描述	
permission \\（必填）整数	权限类型。	 可能的值： 2 - 只读； 3 - 读-写。

创建

描述

`object dashboard.create(object/array dashboards)`

这个方法允许创建新的仪表板。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

`(object/array)` 要创建的仪表板。

另外，对于标准仪表板属性，该方法还接受以下参数。

参数类	描述
widgets	数组仪板创建的小组件信息。
users	数组共仪表板的用户信息
userGroups	数组共仪表板的用户组信息。

返回值

`(object)` 返回一个对象，该对象包含 `dashboardids` 属性下创建的仪表板的 ID。返回的 ID 的顺序与所传递的仪表板的顺序一致。

示例

创建一个仪表板

创建一个名为 “My dashboard” 的仪表板，其中有一个带有标签的问题小部件，并使用了两种类型的共享（用户组和用户）。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "widgets": [
            {
                "type": "problems",
                "x": 0,
                "y": 0,
                "width": 6,
                "height": 5,
                "fields": [
                    {
                        "type": 1,
                        "name": "tags.tag.0",
                        "value": "service"
                    },
                    {
                        "type": 1,
                        "name": "tags.value.0",
                        "value": "zabbix_server"
                    }
                ]
```

```
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ],
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

参见

- 仪表板小组件
- 仪表板小组件字段
- 仪表板用户
- 仪表板用户组

来源

CDashboard::create() in *frontends/php/include/classes/api/services/CDashboard.php*.

删除

描述

`object dashboard.delete(array dashboardids)`

这个方法允许删除仪表板。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

(array) 要删除的仪表板的 ID 数组。

返回值

(object) 返回一个对象，该对象包含 dashboardids 属性下删除的仪表板的 ID。

示例

删除多个仪表板

删除 2 个仪表板

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2",
            "3"
        ]
    },
    "id": 1
}
```

来源

CDashboard::delete() in *frontends/php/include/classes/api/services/CDashboard.php*.

更新

描述

`object dashboard.update(object/array dashboards)`

这个方法允许更新已存在的仪表板。

Note:
任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

`(object/array)` 要更新的仪表板的属性。

必须为每个仪表板定义 `dashboardid` 属性，其它的属性都是可选的。只有传递的属性会被更新，其它属性都将保持不变。

另外，对于标准仪表板属性，该方法接受以下参数。

参数类	描述	
widgets	数组替	已存在的仪表板小组件信息。 仪表板小部件由 widgetid 属性更新。将创建没有 widgetid 属性的小部件。
users	数组替	已存在小部件的共享用户信息。
userGroups	数组替	已存在小部件的共享用户组信息。

返回值

（object）返回一个对象，该对象包含 dashboardids 属性下更新的仪表板的 ID。

示例

重命名一个仪表板

将一个仪表板重命名为 “SQL server 状态”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "name": "SQL server status"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

改变仪表板的属主

仅供管理员和超级管理员使用。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 2
}
```

参见

- 仪表板小组件
- 仪表板小组件字段
- 仪表板用户
- 仪表板用户组

来源

CDashboard::update() in *frontends/php/include/classes/api/services/CDashboard.php*.

获取

描述

`integer/array dashboard.get(object parameters)`

这个方法允许根据给定的参数检索仪表板。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

(object) 定义需要输出的参数。

这个方法支持以下参数。

参数类	描述	
dashboardids	字符串/数组只返回给	ID的仪表板。
selectWidgets	查询返	有widgets属性，包含仪表板中使用的小部件信息。
selectUsers	查询返	有users属性，包含共享仪表板的用户信息。
selectUserGroups	查询返	有userGroups属性，包含共享仪表板的用户组信息。

参数类	描述	
sortfield	字符串/数组根据给定属性对结果进行排序。 可能的值有：dashboardid。	
countOutput	布尔值在 [	eference_comm 中详细描述了所有 get 方法的常见参数。
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(integer/array) 返回：

- 一个对象数组；
- 如果使用了 countOutput 参数，被检索的对象的数量。

示例

通过 ID 检索一个仪表板

检索仪表板"1" 和"2" 的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
```

```
    "method": "dashboard.get",
    "params": {
        "output": "extend",
        "selectWidgets": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "dashboardids": [
            "1",
            "2"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "1",
            "name": "Dashboard",
            "userid": "1",
            "private": "0",
            "users": [],
            "userGroups": [],
            "widgets": [
                {
                    "widgetid": "9",
                    "type": "systeminfo",
                    "name": "",
                    "x": "6",
                    "y": "8",
                    "width": "6",
                    "height": "5",
                    "fields": []
                },
                {
                    "widgetid": "8",
                    "type": "problemsbysv",
                    "name": "",
                    "x": "6",
                    "y": "4",
                    "width": "6",
                    "height": "4",
                    "fields": []
                },
                {
                    "widgetid": "7",
                    "type": "problemhosts",
                    "name": "",
                    "x": "6",
                    "y": "0",
                    "width": "6",
                    "height": "4",
                    "fields": []
                },
                {
                    "widgetid": "6",
                    "type": "discovery",
                    "name": "",
                    "x": "3",
```

```
                "y": "9",
                "width": "3",
                "height": "4",
                "fields": []
            },
            {
                "widgetid": "5",
                "type": "web",
                "name": "",
                "x": "0",
                "y": "9",
                "width": "3",
                "height": "4",
                "fields": []
            },
            {
                "widgetid": "4",
                "type": "problems",
                "name": "",
                "x": "0",
                "y": "3",
                "width": "6",
                "height": "6",
                "fields": []
            },
            {
                "widgetid": "3",
                "type": "favmaps",
                "name": "",
                "x": "4",
                "y": "0",
                "width": "2",
                "height": "3",
                "fields": []
            },
            {
                "widgetid": "2",
                "type": "favscreens",
                "name": "",
                "x": "2",
                "y": "0",
                "width": "2",
                "height": "3",
                "fields": []
            },
            {
                "widgetid": "1",
                "type": "favgraphs",
                "name": "",
                "x": "0",
                "y": "0",
                "width": "2",
                "height": "3",
                "fields": []
            }
        ]
    },
    {
        "dashboardid": "2",
        "name": "My dashboard",
        "userid": "1",
        "private": "1",
```

```
            "users": [
                {
                    "userid": "4",
                    "permission": "3"
                }
            ],
            "userGroups": [
                {
                    "usrgrpid": "7",
                    "permission": "2"
                }
            ],
            "widgets": [
                {
                    "widgetid": "10",
                    "type": "problems",
                    "name": "",
                    "x": "0",
                    "y": "0",
                    "width": "6",
                    "height": "5",
                    "fields": [
                        {
                            "type": "2",
                            "name": "groupids",
                            "value": "4"
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


参见

- 仪表板小组件
- 仪表板小组件字段
- 仪表板用户
- 仪表板用户组

来源

CDashboard::get() in *frontends/php/include/classes/api/services/CDashboard.php*.

11. 发现主机

这个类是设计用于发现主机。

对象引用：

- 发现主机

可用的方法：

- dhost.get - 获取已发现的主机。

> 对象

下列对象与 dhost API 直接相关。

发现主机

Note:
发现的主机是由 Zabbix 服务器创建的，不能通过 API 进行修改。

发现的主机对象包含一个被网络发现规则发现的主机信息。其具有以下属性。

属性类	描述	
dhostid	字符串已发	的主机 ID。
druleid	字符串主机	现规则的 ID。
lastdown	时间戳已发	主机异常的最近一次发生时间。
lastup	时间戳已发	主机正常的最近一次发生时间。
status	整数已	现主机是正常还是异常。如果一个主机至少还有一个活动的已发现服务，那么它就是正常的。 可能的值：0 - 主机正常；1 - 主机异常。

获取

描述

`integer/array dhost.get(object parameters)`

这个方法允许根据给定的参数检索已发现的主机。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

`(object)` 定义需要输出的参数。

这个方法支持以下参数。

参数类	描述	
dhostids	字符串/数组只返回给	ID 的已发现主机。
druleids	字符串/数组只返回给	发现规则创建的已发现主机。
dserviceids	字符串/数组只返回运	指定服务的已发现主机。

参数类	描述	
selectDRules	查询在	drules 属性中以数组形式返回主机发现规则信息。
selectDServices	查询在	dservices 属性中的返回运行于主机上的服务信息。 支持 count。

参数类	描述	
limitSelects	整数限	子选择返回的记录数量。 适用于下列子选择：selectDServi - 结果将按dserviceid排序。
sortfield	字符串/数组根据给定	属性对结果进行排序。 可能的值有：dhostid和druleid。

参数类	描述	
countOutput	布尔值在 [	eference_comm 中详细描述了所有 get 方法的常见参数。
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(integer/array) 返回：

- 一个对象数组；
- 如果使用了 countOutput 参数，返回被检索对象的数量。

示例

通过发现规则检索发现的主机

检索通过发现规则"4" 发现的所有正在运行的主机和发现的服务。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dhost.get",
    "params": {
        "output": "extend",
        "selectDServices": "extend",
        "druleids": "4"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dservices": [
                {
```

```
                "dserviceid": "1",
                "dhostid": "1",
                "type": "4",
                "key_": "",
                "value": "",
                "port": "80",
                "status": "0",
                "lastup": "1337697227",
                "lastdown": "0",
                "dcheckid": "5",
                "ip": "192.168.1.1",
                "dns": "station.company.lan"
            }
        ],
        "dhostid": "1",
        "druleid": "4",
        "status": "0",
        "lastup": "1337697227",
        "lastdown": "0"
    },
    {
        "dservices": [
            {
                "dserviceid": "2",
                "dhostid": "2",
                "type": "4",
                "key_": "",
                "value": "",
                "port": "80",
                "status": "0",
                "lastup": "1337697234",
                "lastdown": "0",
                "dcheckid": "5",
                "ip": "192.168.1.4",
                "dns": "john.company.lan"
            }
        ],
        "dhostid": "2",
        "druleid": "4",
        "status": "0",
        "lastup": "1337697234",
        "lastdown": "0"
    },
    {
        "dservices": [
            {
                "dserviceid": "3",
                "dhostid": "3",
                "type": "4",
                "key_": "",
                "value": "",
                "port": "80",
                "status": "0",
                "lastup": "1337697234",
                "lastdown": "0",
                "dcheckid": "5",
                "ip": "192.168.1.26",
                "dns": "printer.company.lan"
            }
        ],
        "dhostid": "3",
        "druleid": "4",
```

```
            "status": "0",
            "lastup": "1337697234",
            "lastdown": "0"
        },
        {
            "dservices": [
                {
                    "dserviceid": "4",
                    "dhostid": "4",
                    "type": "4",
                    "key_": "",
                    "value": "",
                    "port": "80",
                    "status": "0",
                    "lastup": "1337697234",
                    "lastdown": "0",
                    "dcheckid": "5",
                    "ip": "192.168.1.7",
                    "dns": "mail.company.lan"
                }
            ],
            "dhostid": "4",
            "druleid": "4",
            "status": "0",
            "lastup": "1337697234",
            "lastdown": "0"
        }
    ],
    "id": 1
}
```


参见

- 发现服务
- 发现规则

来源

CDHost::get() in *frontends/php/include/classes/api/services/CDHost.php*.

12. 发现服务

这个类被设计用于发现服务。

对象引用：

- 发现服务

可用的方法：

- dservice.get - 获取已发现的服务。

> 对象

下列对象与 dservice API 直接相关。

发现服务

> **Note:**
> 发现的服务是由 Zabbix 服务器创建的，不能通过 API 进行修改。

被发现的服务对象包含由一个主机上的网络发现规则发现的服务信息。其具有以下属性。

属性类	描述	
dserviceid	字符串发现	服务的 ID。
dcheckid	字符串用于	测服务的发现规则的 ID。
dhostid	字符串运行	服务的已发现的主机的 ID。
dns	字符串运行	服务的主机的 DNS。
ip	字符串运行	服务的主机的 IP 地址。
lastdown	时间戳发现	服务最后异常的时间。
lastup	时间戳发现	服务最后正常的时间。
port	整数服	端口号。
status	整数服	的状态。 可能的值： 0 - 服务正常； 1 - 服务异常。
value	字符串当执	Zabbix 客户端、SNMPv1、SNMPv2 或 SNMPv3 等发现检查时，服务返回的值。

获取

描述

`integer/array dservice.get(object parameters)`

这个方法允许根据给定的参数检索发现的服务。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

`(object)` 定义需要输出的参数。

这个方法支持以下参数。

参数类	描述	
dserviceids	字符串/数组只返回具	给定 ID 的已发现服务。
dhostids	字符串/数组只返回给	已发现主机上运行的已发现服务。

参数类	描述	
dcheckids	字符串/数组只返回给	的发现检查检测到的已发现服务。
druleids	字符串/数组只返回给	的发现规则检测到的已发现服务。

参数类	描述
selectDRules	查询在 drules 属性中以数组形式返回服务的发现规则信息。 | |selectDHosts 查询在 |dhosts 属性中以数组形式返回服务运行的主机信息。 | |selectHosts 查询在 |hosts 属性中返回具有与该服务相同的

参数类	描述	
limitSelects	整数限	子选择返回的记录数量。 适用于下列子选择： selectHosts - 结果将按 hostid 排序。
sortfield	字符串/数组根据给定	属性对结果进行排序。 可能的值有：dserviceid，dhostid 和 ip。

参数类	描述	
countOutput	布尔值在 [	eference_comm 中 详 细 描 述 了 所 有 get 方 法 的 常 见 参 数。
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(integer/array) 返回：

- 一个对象数组；
- 如果使用了 countOutput 参数，被检索的对象的数量。

示例

检索在主机上发现的服务

检索在主机"11" 上发现的所有服务。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dservice.get",
    "params": {
        "output": "extend",
        "dhostids": "11"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dserviceid": "12",
            "dhostid": "11",
            "value": "",
```

```
            "port": "80",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650607",
            "dcheckid": "5",
            "ip": "192.168.1.134",
            "dns": "john.local"
        },
        {
            "dserviceid": "13",
            "dhostid": "11",
            "value": "",
            "port": "21",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650610",
            "dcheckid": "6",
            "ip": "192.168.1.134",
            "dns": "john.local"
        }
    ],
    "id": 1
}
```


参见

- 发现主机
- 检查发现
- 主机

来源

CDService::get() in *frontends/php/include/classes/api/services/CDService.php*.

13. 检查发现

这个类是设计用于检查发现。

对象引用：

- 检查发现

可用的方法：

- dcheck.get - 获取检查发现。

> 对象

下列对象与 dcheck API 直接相关。

检查发现

检查发现对象定义了由一个网络发现规则执行的一个特定检查。其具有以下属性。

属性类	描述	
dcheckid	字符串 * (	读) * 检查发现的ID。
druleid	字符串 * (	读) * 该检查属于的发现规则的ID。

属性类	描述
key_	字符串此属性的值根据检查的类型而不同： - 查询 Zabbix 客户端检查的键值，需要的； - SNMPv1，SNMPv2 和 SNMPv3 检查所需的 SNMP OID，需要的。

属性类	描述	
ports	字符串用逗	分隔的一个或几个端口范围。用于除 ICMP 之外的所有检查。 默认：0。
snmp_community	字符串 SN	P 团体串。 SNMPv1 和 SN-MPv2 客户端检查需要使用。

属性类	描述	
snmpv3_authpassphrase	字符串用于	SNMPv3客户端检查的身份验证密码，安全级别可设置为 *authNoPriv* 或 *authPriv*。

属性类	描述	
snmpv3_authprotocol	整数用	SNMPv3 客户端检查的身份验证协议，安全级别可设置为 *authNoPriv* 或 *authPriv*。 可能的值：0 - (默认) MD5；1 - SHA。
snmpv3_contextname	字符串 SN	Pv3 上下文名称。只用于 SNMPv3 检查。

属性类	描述	
snmpv3_privpassphrase	字符串用于	SNMPv3客户端检查的隐私验证密码，安全级别可设置为*auth-Priv*。
snmpv3_privprotocol	整数用	SNMPv3客户端检查的隐私验证协议，安全级别可设置为*auth-Priv*。 可能的值： 0 - (默认) DES； 1 - AES。

属性类	描述	
snmpv3_securitylevel	字符串用于	SNMPv3客户端检查的安全级别。 可能的值：0 - noAuthNoPriv；1 - authNoPriv；2 - authPriv。
snmpv3_securityname	字符串用于	SNMPv3客户端检查的安全名称。

属性类	描述
type (必填)	整数检的类型。可能的值：0 - SSH；1 - LDAP；2 - SMTP；3 - FTP；4 - HTTP；5 - POP；6 - NNTP；7 - IMAP；8 - TCP；9 - Zabbix 客户端；10 - SNMPv1 客户端；11 - SNMPv2 客户端；12 - ICMP ping；13 - SNMPv3 客户端；14 - HTTPS；15 - Telnet。

属性类	描述	
uniq	整数是	将此检查作为设备惟一性条件。对于发现规则，只能配置一个唯一的检查。用于Zabbix客户端、SNMPv1、SNMPv2和SNMPv3等客户端检查。 可能的值： 0 - (默认) 不使用该检查作

属性类	描述	
host_source	整数主	名称来源 可能的值：1 - (默认) DNS；2 - IP；3 - 此检查发现值
name_source	整数可	名称来源 可能的值：0 - (默认) 无；1 - DNS；2 - IP；3 - 此检查发现值

获取

描述

`integer/array dcheck.get(object parameters)`

这个方法允许根据给定的参数检索检查发现信息。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤消调用该方法的权限。有关更多信息，请参见用户角色。

参数

`(object)` 定义需要输出的参数。

这个方法支持以下参数。

参数类	描述	
dcheckids	字符串/数组只返回给	ID的检查发现信息。
druleids	字符串/数组只返回属	给定发现规则的检查发现信息。
dserviceids	字符串/数组只返回给	已发现服务的检查发现信息。
sortfield	字符串/数组根据给定	属性对结果进行排序。 可能的值有：`dcheckid`和`druleid`。

参数类	描述	
countOutput	布尔值在 [	eference_comm 中 详 细 描 述 了 所 有 get 方 法 的 常 见 参 数。
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	整数:	:
output	查询:	:
preservekeys	布尔值::	
search	对象:	:
searchByAny	布尔值::	
searchWildcardsEnabled	布尔值::	
sortorder	字符串/数组:::	
startSearch	布尔值::	

返回值

(integer/array) 返回：

- 一个对象数组；
- 如果使用了 countOutput 参数，被检索的对象的数量。

示例

为一个发现规则检索检查发现信息

检索发现规则"6" 使用的所有检查发现信息。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dcheck.get",
    "params": {
        "output": "extend",
        "dcheckids": "6"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dcheckid": "6",
            "druleid": "4",
            "type": "3",
```

```
            "key_": "",
            "snmp_community": "",
            "ports": "21",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "uniq": "0",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0"
        }
    ],
    "id": 1
}
```


来源

CDCheck::get() in *frontends/php/include/classes/api/services/CDCheck.php*.

Appendix 1. Reference commentary

Notation Data types

The Zabbix API supports the following data types:

Type	Description
boolean	A boolean value, accepts either `true` or `false`.
flag	The value is considered to be `true` if it is passed and not equal to `null` and `false` otherwise.
integer	A whole number.
float	A floating point number.
string	A text string.
text	A longer text string.
timestamp	A Unix timestamp.
array	An ordered sequence of values, that is, a plain array.
object	An associative array.
query	A value which defines, what data should be returned. Can be defined as an array of property names to return only specific properties, or as one of the predefined values: `extend` - returns all object properties; `count` - returns the number of retrieved records, supported only by certain subselects.

Property labels

Some of the objects properties are marked with short labels to describe their behavior. The following labels are used:

- *readonly* - the value of the property is set automatically and cannot be defined or changed by the client;
- *constant* - the value of the property can be set when creating an object, but cannot be changed after.

Reserved ID value "0" Reserved ID value "0" can be used to filter elements and to remove referenced objects. For example, to remove a referenced proxy from a host, proxy_hostid should be set to 0 ("proxy_hostid": "0") or to filter hosts monitored by server option proxyids should be set to 0 ("proxyids": "0").

Common "get" method parameters The following parameters are supported by all `get` methods:

Parameter	Type	Description
countOutput	boolean	Return the number of records in the result instead of the actual data.

Parameter	Type	Description
editable	boolean	If set to `true` return only objects that the user has write permissions to. Default: `false`.
excludeSearch	boolean	Return results that do not match the criteria given in the `search` parameter.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Doesn't work for `text` fields.
limit	integer	Limit the number of records returned.
output	query	Object properties to be returned. Default: `extend`.
preservekeys	boolean	Use IDs as keys in the resulting array.
search	object	Return results that match the given wildcard search (case-insensitive). Accepts an array, where the keys are property names, and the values are strings to search for. If no additional options are given, this will perform a `LIKE "%…%"` search. Works only for `string` and `text` fields.
searchByAny	boolean	If set to `true` return results that match any of the criteria given in the `filter` or `search` parameter instead of all of them. Default: `false`.
searchWildcardsEnabled	boolean	If set to `true` enables the use of "*" as a wildcard character in the `search` parameter. Default: `false`.
sortfield	string/array	Sort the result by the given properties. Refer to a specific API get method description for a list of properties that can be used for sorting. Macros are not expanded before sorting.
sortorder	string/array	Order of sorting. If an array is passed, each value will be matched to the corresponding property given in the `sortfield` parameter. Possible values are: `ASC` - ascending; `DESC` - descending.
startSearch	boolean	The `search` parameter will compare the beginning of fields, that is, perform a `LIKE "…%"` search instead. Ignored if `searchWildcardsEnabled` is set to `true`.

Zabbix manpages

These are Zabbix manpages for Zabbix processes.

zabbix_agentd

Section: Maintenance Commands (8)
Updated: 2016-01-13
Index Return to Main Contents

NAME

zabbix_agentd - Zabbix agent daemon

SYNOPSIS

zabbix_agentd [**-c** *config-file*]
zabbix_agentd [**-c** *config-file*] **-p**
zabbix_agentd [**-c** *config-file*] **-t** *item-key*
zabbix_agentd [**-c** *config-file*] **-R** *runtime-option*
zabbix_agentd -h
zabbix_agentd -V

DESCRIPTION

zabbix_agentd is a daemon for monitoring of various server parameters.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one. Path to the file should be specified.

-f, **--foreground**
Run Zabbix agent in foreground.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified

Log level control targets

pid
Process identifier

process-type
All processes of specified type (e.g., listener)

process-type,N
Process type and number (e.g., listener,3)

-p, **--print**
Print known items and exit. For each item either generic defaults are used, or specific defaults for testing are supplied. These defaults are listed in square brackets as item key parameters. Returned values are enclosed in square brackets and prefixed with the type of the returned value, separated by a pipe character. For user parameters type is always **t**, as the agent can not determine all possible return values. Items, displayed as working, are not guaranteed to work from the Zabbix server or zabbix_get when querying a running agent daemon as permissions or environment may be different. Returned value types are:

d
Number with a decimal part.

m
Not supported. This could be caused by querying an item that only works in the active mode like a log monitoring item or an item that requires multiple collected values. Permission issues or incorrect user parameters could also result in the not supported state.

s
Text. Maximum length not limited.

t
Text. Same as **s**.

u
Unsigned integer.

-t, **--test** *item-key*
Test single item and exit. See **--print** for output description.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_agentd.conf
Default location of Zabbix agent configuration file (if not modified during compile time).

SEE ALSO

zabbix_get(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 08:31:40 GMT, January 19, 2016

zabbix_get

Section: User Commands (1)
Updated: 2015-08-06
Index Return to Main Contents

NAME

zabbix_get - Zabbix get utility

SYNOPSIS

zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] **-k** *item-key*
zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-agent-cert-issuer** *cert-issuer*] [**--tls-agent-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* **-k** *item-key*
zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* **-k** *item-key*
zabbix_get -h
zabbix_get -V

DESCRIPTION

zabbix_get is a command line utility for getting data from Zabbix agent.

OPTIONS

-s, **--host** *host-name-or-IP*
Specify host name or IP address of a host.

-p, **--port** *port-number*
Specify port number of agent running on the host. Default is 10050.

-I, **--source-address** *IP-address*
Specify source IP address.

-k, **--key** *item-key*
Specify key of item to retrieve value for.

--tls-connect *value*
How to connect to agent. Values:

unencrypted
connect without encryption

psk
connect using TLS and a pre-shared key

cert
connect using TLS and a certificate

--tls-ca-file *CA-file*
Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification.

--tls-crl-file *CRL-file*
Full pathname of a file containing revoked certificates.

--tls-agent-cert-issuer *cert-issuer*
Allowed agent certificate issuer.

--tls-agent-cert-subject *cert-subject*
Allowed agent certificate subject.

--tls-cert-file *cert-file*
Full pathname of a file containing the certificate or certificate chain.

--tls-key-file *key-file*
Full pathname of a file containing the private key.

--tls-psk-identity *PSK-identity*
PSK-identity string.

--tls-psk-file *PSK-file*
Full pathname of a file containing the pre-shared key.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

EXAMPLES

zabbix_get -s 127.0.0.1 -p 10050 -k ”system.cpu.load[all,avg1]”
zabbix_get -s 127.0.0.1 -p 10050 -k ”system.cpu.load[all,avg1]” --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-agent-cert-issuer ”CN=Signing CA,OU=IT operations,O=Example Corp,DC=example,DC=com” --tls-agent-cert-subject ”CN=server1,OU=IT operations,O=Example Corp,DC=example,DC=com” --tls-cert-file /home/zabbix/zabbix_get.crt --tls-key-file /home/zabbix/zabbix_get.key
zabbix_get -s 127.0.0.1 -p 10050 -k ”system.cpu.load[all,avg1]” --tls-connect psk --tls-psk-identity ”PSK ID Zabbix agentd” --tls-psk-file /home/zabbix/zabbix_agentd.psk

SEE ALSO

zabbix_agentd(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION
Options

EXAMPLES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 09:21:04 GMT, January 08, 2016

zabbix_proxy

Section: Maintenance Commands (8)
Updated: 2016-01-13
Index Return to Main Contents

NAME

zabbix_proxy - Zabbix proxy daemon

SYNOPSIS

zabbix_proxy [**-c** *config-file*]
zabbix_proxy [**-c** *config-file*] **-R** *runtime-option*
zabbix_proxy -h
zabbix_proxy -V

DESCRIPTION

zabbix_proxy is a daemon that collects monitoring data from devices and sends it to Zabbix server.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one. Path to the file should be specified.

-f, **--foreground**
Run Zabbix proxy in foreground.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Active Zabbix proxy will connect to the Zabbix server and request configuration data. Default configuration file (unless **-c** option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

housekeeper_execute
Execute the housekeeper. Ignored if housekeeper is being currently executed.

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified

Log level control targets

pid
Process identifier

process-type
All processes of specified type (e.g., poller)

process-type,N
Process type and number (e.g., poller,3)

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_proxy.conf
Default location of Zabbix proxy configuration file (if not modified during compile time).

SEE ALSO

zabbix_agentd(8), **zabbix_get**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 09:10:13 GMT, January 19, 2016

zabbix_sender

Section: User Commands (1)
Updated: 2015-10-16
Index Return to Main Contents

NAME

zabbix_sender - Zabbix sender utility

SYNOPSIS

zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] **-s** *host* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] [**-T**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] [**-T**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] **-s** *host* **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**-T**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**-T**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] **-s** *host* **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**-T**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect psk** **--tls-psk-identity** *PSK-identity*

--tls-psk-file *PSK-file* [**-T**] [**-r**] **-i** *input-file*
zabbix_sender -h
zabbix_sender -V

DESCRIPTION

zabbix_sender is a command line utility for sending monitoring data to Zabbix server or proxy. On the Zabbix server an item of type **Zabbix trapper** should be created with corresponding key. Note that incoming values will only be accepted from hosts specified in **Allowed hosts** field for this item.

OPTIONS

-c, **--config** *config-file*
Use *config-file*. **Zabbix sender** reads server details from the agentd configuration file. By default **Zabbix sender** does not read any configuration file. Path to the file should be specified. Only parameters **Hostname**, **ServerActive** and **SourceIP** are supported. First entry from the **ServerActive** parameter is used.

-z, **--zabbix-server** *server*
Hostname or IP address of Zabbix server. If a host is monitored by a proxy, proxy hostname or IP address should be used instead. When used together with **--config**, overrides the first entry of **ServerActive** parameter specified in agentd configuration file.

-p, **--port** *port*
Specify port number of Zabbix server trapper running on the server. Default is 10051. When used together with **--config**, overrides the port of first entry of **ServerActive** parameter specified in agentd configuration file.

-I, **--source-address** *IP-address*
Specify source IP address. When used together with **--config**, overrides **SourceIP** parameter specified in agentd configuration file.

-s, **--host** *host*
Specify host name the item belongs to (as registered in Zabbix frontend). Host IP address and DNS name will not work. When used together with **--config**, overrides **Hostname** parameter specified in agentd configuration file.

-k, **--key** *key*
Specify item key to send value to.

-o, **--value** *value*
Specify item value.

-i, **--input-file** *input-file*
Load values from input file. Specify **-** as **<input-file>** to read values from standard input. Each line of file contains whitespace delimited: **<hostname> <key> <value>**. Each value must be specified on its own line. Each line must contain 3 whitespace delimited entries: **<hostname> <key> <value>**, where ”hostname” is the name of monitored host as registered in Zabbix frontend, ”key” is target item key and ”value” - the value to send. Specify **-** as **<hostname>** to use hostname from agent configuration file or from **--host** argument.

An example of a line of an input file:

”Linux DB3” db.connections 43

The value type must be correctly set in item configuration of Zabbix frontend. Zabbix sender will send up to 250 values in one connection. Contents of the input file must be in the UTF-8 encoding. All values from the input file are sent in a sequential order top-down. Entries must be formatted using the following rules:

•
Quoted and non-quoted entries are supported.

•
Double-quote is the quoting character.

•
Entries with whitespace must be quoted.

•
Double-quote and backslash characters inside quoted entry must be escaped with a backslash.

•
Escaping is not supported in non-quoted entries.

•
Linefeed escape sequences (\n) are supported in quoted strings.

•
Linefeed escape sequences are trimmed from the end of an entry.

-T, **--with-timestamps**
This option can be only used with **--input-file** option.

Each line of the input file must contain 4 whitespace delimited entries: **<hostname> <key> <timestamp> <value>**. Timestamp should be specified in Unix timestamp format. If target item has triggers referencing it, all timestamps must be in an increasing order, otherwise event calculation will not be correct.

An example of a line of the input file:

"Linux DB3" db.connections 1429533600 43

For more details please see option **--input-file**.

If a timestamped value is sent for a host that is in a "no data" maintenance type then this value will be dropped however it is possible to send a timestamped value in for an expired maintenance period and it will be accepted.

-r, **--real-time**
Send values one by one as soon as they are received. This can be used when reading from standard input.

--tls-connect *value*
How to connect to server or proxy. Values:

unencrypted
connect without encryption

psk
connect using TLS and a pre-shared key

cert
connect using TLS and a certificate

--tls-ca-file *CA-file*
Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification.

--tls-crl-file *CRL-file*
Full pathname of a file containing revoked certificates.

--tls-server-cert-issuer *cert-issuer*
Allowed server certificate issuer.

--tls-server-cert-subject *cert-subject*
Allowed server certificate subject.

--tls-cert-file *cert-file*
Full pathname of a file containing the certificate or certificate chain.

--tls-key-file *key-file*
Full pathname of a file containing the private key.

--tls-psk-identity *PSK-identity*
PSK-identity string.

--tls-psk-file *PSK-file*
Full pathname of a file containing the pre-shared key.

-v, **--verbose**
Verbose mode, **-vv** for more details.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

EXIT STATUS

The exit status is 0 if the values were sent and all of them were successfully processed by server. If data was sent, but processing of at least one of the values failed, the exit status is 2. If data sending failed, the exit status is 1.

EXAMPLES

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -k mysql.queries -o 342.45

Send **342.45** as the value for **mysql.queries** item of monitored host. Use monitored host and Zabbix server defined in agent configuration file.

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -s ”Monitored Host” -k mysql.queries -o 342.45

Send **342.45** as the value for **mysql.queries** item of **Monitored Host** host using Zabbix server defined in agent configuration file.

zabbix_sender -z 192.168.1.113 -i data_values.txt

Send values from file **data_values.txt** to Zabbix server with IP **192.168.1.113**. Host names and keys are defined in the file.

echo ”- hw.serial.number 1287872261 SQ4321ASDF” | zabbix_sender -c /usr/local/etc/zabbix_agentd.conf -T -i -

Send a timestamped value from the commandline to Zabbix server, specified in the agent configuration file. Dash in the input data indicates that hostname also should be used from the same configuration file.

echo ’”Zabbix server” trapper.item ””’ | zabbix_sender -z 192.168.1.113 -p 10000 -i -

Send empty value of an item to the Zabbix server with IP address **192.168.1.113** on port **10000** from the commandline. Empty values must be indicated by empty double quotes.

zabbix_sender -z 192.168.1.113 -s ”Monitored Host” -k mysql.queries -o 342.45 --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-cert-file /home/zabbix/zabbix_agentd.crt --tls-key-file /home/zabbix/zabbix_agentd.key

Send **342.45** as the value for **mysql.queries** item in **Monitored Host** host to server with IP **192.168.1.113** using TLS with certificate.

zabbix_sender -z 192.168.1.113 -s ”Monitored Host” -k mysql.queries -o 342.45 --tls-connect psk --tls-psk-identity ”PSK ID Zabbix agentd” --tls-psk-file /home/zabbix/zabbix_agentd.psk

Send **342.45** as the value for **mysql.queries** item in **Monitored Host** host to server with IP **192.168.1.113** using TLS with pre-shared key (PSK).

SEE ALSO

zabbix_agentd(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

EXIT STATUS

EXAMPLES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 09:21:17 GMT, January 08, 2016

zabbix_server

Section: Maintenance Commands (8)
Updated: 2016-01-13
Index Return to Main Contents

NAME

zabbix_server - Zabbix server daemon

SYNOPSIS

zabbix_server [**-c** *config-file*]
zabbix_server [**-c** *config-file*] **-R** *runtime-option*
zabbix_server -h
zabbix_server -V

DESCRIPTION

zabbix_server is the core daemon of Zabbix software.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one. Path to the file should be specified.

-f, **--foreground**
Run Zabbix server in foreground.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Default configuration file (unless **-c** option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

housekeeper_execute
Execute the housekeeper. Ignored if housekeeper is being currently executed.

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified

Log level control targets

pid
Process identifier

process-type
All processes of specified type (e.g., poller)

process-type,N
Process type and number (e.g., poller,3)

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_server.conf
Default location of Zabbix server configuration file (if not modified during compile time).

SEE ALSO

zabbix_agentd(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_sender**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 09:11:11 GMT, January 19, 2016