Documentation 5.0

ZABBIX

01.07.2025

Contents

Zabbix 用户手册 **6**
Copyright notice . . . 6
附录 . . . 6
1 常见问题/疑难解答 . . . 6
2 安装 . . . 7
3 后台进程配置 . . . 38
4 各种协议 . . . 391
5 监控项 . . . 416
6 触发器 . . . 441
7 宏 . . . 446
8 单位符号说明 . . . 505
9 时间段配置 . . . 506
10 命令执行 . . . 507
13 版本兼容性 . . . 507
14 Python library for Zabbix API . . . 508
14 数据库错误处理 . . . 508
15 适用于 Windows 的 Zabbix sender 动态链接库 . . . 508
16 SELINUX 的问题 . . . 509
17 其他问题 . . . 509
18 Agent 与 agent2 对比 . . . 510
1. 简介 . . . 512
1 手册结构 . . . 512
2 Zabbix 介绍 . . . 513
3 Zabbix 功能 . . . 513
4 Zabbix 概述 . . . 514
5 Zabbix 5.0.0 新功能特性 . . . 515
6 Zabbix 5.0.1 新功能特性 . . . 527
7 Zabbix 5.0.2 新功能特性 . . . 527
8 Zabbix 5.0.3 新功能特性 . . . 528
9 Zabbix 5.0.4 新功能特性 . . . 529
10 Zabbix 5.0.5 新功能特性 . . . 529
11 Zabbix 5.0.6 新功能特性 . . . 531
12 Zabbix 5.0.7 新功能特性 . . . 531
13 Zabbix 5.0.8 新功能特性 . . . 532
14 What’s new in Zabbix 5.0.9 . . . 532
15 What’s new in Zabbix 5.0.10 . . . 533
15 What’s new in Zabbix 5.0.11 . . . 534
16 What’s new in Zabbix 5.0.12 . . . 534
17 What’s new in Zabbix 5.0.13 . . . 535
18 What’s new in Zabbix 5.0.14 . . . 536
19 What’s new in Zabbix 5.0.15 . . . 536
20 What’s new in Zabbix 5.0.16 . . . 537
21 What’s new in Zabbix 5.0.17 . . . 537
22 What’s new in Zabbix 5.0.18 . . . 537
23 What’s new in Zabbix 5.0.19 . . . 538
24 What’s new in Zabbix 5.0.20 . . . 538
25 What’s new in Zabbix 5.0.21 . . . 538
26 What’s new in Zabbix 5.0.22 . . . 539
27 What’s new in Zabbix 5.0.23 . . . 539
28 What’s new in Zabbix 5.0.24 . . . 539

29 What's new in Zabbix 5.0.25 . . . 539
30 What's new in Zabbix 5.0.26 . . . 540
31 What's new in Zabbix 5.0.27 . . . 540
32 What's new in Zabbix 5.0.28 . . . 540
33 What's new in Zabbix 5.0.29 . . . 540
34 What's new in Zabbix 5.0.30 . . . 541
35 What's new in Zabbix 5.0.31 . . . 541
36 What's new in Zabbix 5.0.32 . . . 541
37 What's new in Zabbix 5.0.33 . . . 542
38 What's new in Zabbix 5.0.34 . . . 542
39 What's new in Zabbix 5.0.35 . . . 542
40 What's new in Zabbix 5.0.36 . . . 542
41 What's new in Zabbix 5.0.37 . . . 542
42 What's new in Zabbix 5.0.38 . . . 542
43 What's new in Zabbix 5.0.39 . . . 542
44 What's new in Zabbix 5.0.40 . . . 543
45 What's new in Zabbix 5.0.41 . . . 543
46 What's new in Zabbix 5.0.42 . . . 543
47 What's new in Zabbix 5.0.43 . . . 543
2. 定义 . . . 543
3. 进程 . . . 545
1 Server . . . 545
2 Agent . . . 547
3 Agent 2 . . . 551
4 Proxy . . . 559
5 Java 网关 . . . 561
6 Sender . . . 567
7 Get . . . 567
8 JS . . . 568
4. 安装 . . . 568
1 获取 Zabbix . . . 568
2 安装要求 . . . 569
3 从源代码包安装 . . . 588
4 从二进制包安装 . . . 597
5 从容器中安装 . . . 612
6 Web 界面安装 . . . 633
7 升级步骤 . . . 645
8 已知问题 . . . 667
9 模板变更 . . . 671
10 Zabbix 5.0.0 升级说明 . . . 675
11 Zabbix 5.0.1 升级说明 . . . 677
12 Zabbix 5.0.2 升级说明 . . . 677
13 Zabbix 5.0.3 升级说明 . . . 678
14 Zabbix 5.0.4 升级说明 . . . 678
15 Zabbix 5.0.5 升级说明 . . . 678
16 Zabbix 5.0.6 升级说明 . . . 678
17 Zabbix 5.0.7 升级说明 . . . 679
18 Zabbix 5.0.8 升级说明 . . . 679
19 Upgrade notes for 5.0.9 . . . 679
20 Upgrade notes for 5.0.10 . . . 679
21 Upgrade notes for 5.0.11 . . . 679
22 Upgrade notes for 5.0.12 . . . 679
23 Upgrade notes for 5.0.13 . . . 680
24 Upgrade notes for 5.0.14 . . . 680
25 Upgrade notes for 5.0.15 . . . 680
26 Upgrade notes for 5.0.16 . . . 680
27 Upgrade notes for 5.0.17 . . . 680
28 Upgrade notes for 5.0.18 . . . 680
29 Upgrade notes for 5.0.19 . . . 680
30 Upgrade notes for 5.0.20 . . . 680
31 Upgrade notes for 5.0.21 . . . 680
32 Upgrade notes for 5.0.22 . . . 681
33 Upgrade notes for 5.0.23 . . . 681

34 Upgrade notes for 5.0.24 . . . 681
35 Upgrade notes for 5.0.25 . . . 681
36 Upgrade notes for 5.0.26 . . . 681
37 Upgrade notes for 5.0.27 . . . 681
38 Upgrade notes for 5.0.28 . . . 681
39 Upgrade notes for 5.0.29 . . . 681
40 Upgrade notes for 5.0.30 . . . 681
41 Upgrade notes for 5.0.31 . . . 681
42 Upgrade notes for 5.0.32 . . . 682
43 Upgrade notes for 5.0.33 . . . 682
44 Upgrade notes for 5.0.34 . . . 682
45 Upgrade notes for 5.0.35 . . . 682
46 Upgrade notes for 5.0.36 . . . 683
47 Upgrade notes for 5.0.37 . . . 683
48 Upgrade notes for 5.0.38 . . . 683
49 Upgrade notes for 5.0.39 . . . 683
50 Upgrade notes for 5.0.40 . . . 683
51 Upgrade notes for 5.0.41 . . . 683
52 Upgrade notes for 5.0.42 . . . 683
5. 快速入门 . . . 683
1 登陆和配置用户 . . . 683
2 新建主机 . . . 687
3 新建监控项 . . . 688
4 新建触发器 . . . 689
5 获取问题通知 . . . 692
6 新建模版 . . . 696
6. Zabbix 应用 . . . 698
7. 配置 . . . 700
1 主机和主机组 . . . 706
2 监控项 . . . 733
3 触发器 . . . 1065
4 事件 . . . 1088
5 事件关联 . . . 1091
6 Visualization . . . 1102
7 模板 . . . 1147
8 开箱即用的模板 . . . 1148
9 事件通知 . . . 1183
10 宏 . . . 1289
11 用户和用户组 . . . 1298
8. 服务监控 . . . 1313
9. Web 监控 . . . 1322
1 Web 监控项 . . . 1349
2 真实场景监控 . . . 1355
10. 虚拟机监控 . . . 1362
1 虚拟机发现 key 字段 . . . 1366
11. 维护 . . . 1368
12. 正则表达式 . . . 1374
13. 问题确认 . . . 1388
14. 配置导出/导入 . . . 1394
1 主机组 . . . 1395
2 模板 . . . 1395
3 主机 . . . 1453
4 网络拓扑图 . . . 1518
5 聚合图形 . . . 1538
6 媒介类型 . . . 1550
15. 发现 . . . 1558
1 网络发现 . . . 1558
2 Active agent 自动注册 . . . 1571
3 自动发现（LLD） . . . 1574
16. 分布式监控 . . . 1626
1 代理 . . . 1627
17. 加密 . . . 1635
1 使用证书 . . . 1643

2 使用共享密钥 . . . 1654
3 排错 . . . 1658
18. Web 界面 . . . 1660
1 菜单 . . . 1661
2 前端 . . . 1662
3 用户资料 . . . 1830
4 全局搜索 . . . 1834
5 前端维护模式 . . . 1835
6 页面参数 . . . 1836
7 定义 . . . 1837
8 自定义主题 . . . 1838
9 调试模式 . . . 1839
10 Zabbix 使用的 Cookies . . . 1840
19. API . . . 1841
方法参考 . . . 1846
Zabbix API 在 5.0 版本中的变更 . . . 2679
附录 1. 参考说明 . . . 2680
附录 2. 从版本 4.4 到版本 5.0 的一些变更 . . . 2695
20. 组件 . . . 2696

Zabbix 手册页 **2702**
zabbix_agent2 . . . 2702
名称 . . . 2702
概要简介 . . . 2702
描述 . . . 2703
选项 . . . 2703
档案 . . . 2704
另请参阅 . . . 2704
作者 . . . 2704
索引 . . . 2704
zabbix_agentd . . . 2704
名称 . . . 2704
概要 . . . 2704
描述 . . . 2704
选项 . . . 2705
档案 . . . 2706
另请参阅 . . . 2706
作者 . . . 2706
索引 . . . 2706
zabbix_get . . . 2706
名称 . . . 2706
概要 . . . 2706
描述 . . . 2706
选项 . . . 2707
例子 . . . 2707
另请参阅 . . . 2708
作者 . . . 2708
使用 . . . 2708
zabbix_js . . . 2708
名称 . . . 2708
概要 . . . 2708
描述 . . . 2708
选项 . . . 2708
例子 . . . 2709
另请参阅 . . . 2709
索引 . . . 2709
zabbix_proxy . . . 2709
名称 . . . 2709
概要 . . . 2709
描述 . . . 2710
选项 . . . 2710
档案 . . . 2710
另请参阅 . . . 2710

作者 . . . 2711
索引 . . . 2711
zabbix_sender . . . 2711
名称 . . . 2711
概要 . . . 2711
描述 . . . 2712
选项 . . . 2712
退出状态 . . . 2714
例子 . . . 2714
另请参阅 . . . 2714
作者 . . . 2714
索引 . . . 2715
zabbix_server . . . 2715
NAME . . . 2715
概要 . . . 2715
描述 . . . 2715
选项 . . . 2715
档案 . . . 2716
另请参阅 . . . 2716
作者 . . . 2716
索引 . . . 2716

Zabbix 用户手册

欢迎查阅 Zabbix 用户使用手册。

Zabbix 产品手册由原厂 Zabbix 技术团队创建，Zabbix 中国——上海宏时数据系统有限公司组织开源社区志愿者翻译并维护。希望可以帮助用户更好地使用 Zabbix，解决和管理日常 IT 运维监控遇到的各种问题。翻译虽然结束，优化并未停止，如有优化反馈及更多问题，欢迎联系小 Z 17502189550。

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

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

2 安装

2 Installation

1 数据库创建

概述

Zabbix 数据库必须在 Zabbix server 或 proxy 安装的时候创建。

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。

UTF-8 是 Zabbix 支持的唯一编码。它可以正常工作而没有任何安全漏洞。用户应注意，如果使用其他一些编码，则存在已知的安全问题。

> **Note:**
> If installing from Zabbix Git repository, you need to run:
> `$ make dbschema`
> prior to proceeding to the next steps.

MySQL

字符集 utf8 和 utf8_bin 排序规则是 Zabbix Server/Proxy 与 MySQL 数据库一起正常工作所必需的。

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

<note important> 如果要从 Zabbix 软件包安装，请在此处停止，并继续说明RHEL/CentOS或Debian/Ubuntu将数据导入数据库。:::

如果要从源代码安装 Zabbix，请继续将数据导入数据库。对于 Zabbix 代理数据库，应仅导入 `schema.sql`（不是 images.sql 或 data.sql）：

```
shell> cd database/mysql
shell> mysql -uzabbix -p<您的密码> 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<您的密码> zabbix < images.sql
shell> mysql -uzabbix -p<您的密码> 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）：

```
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）：

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

<note important> 如果要从 Zabbix 软件包安装，请在此处停止，并继续说明Debian/Ubuntu 或RHEL/CentOS将初始模式和数据导入数据库。

:::

如果要从源代码安装 Zabbix，请继续导入初始架构和数据（假设您位于 Zabbix 源代码的根目录中）。对于 Zabbix 代理数据库，应仅导入 schema.sql（不是 images.sql 或 data.sql）。

```
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 < 您的账号 >"，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。

TIMESCALEDB

在单独的部分中提供了有关创建和配置 TimescaleDB 的说明

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：

```
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：

```
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：

```
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
```

Note:
请设置初始化参数 CURSOR_SHARING = FORCE 以获得最佳性能。

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"
```

SQLite

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

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。

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

返回安装部分。

Additional patches

In some cases, Zabbix might generate non-optimized queries to the database. This may happen, for example, as a result of using an older database version.
Additional DB patches are available for resolving such issues on the specific database or, sometimes, a specific database version. See [Known issues|/manual/installation/known_issues#slow-mysql-queries] for the list of known problems and available patches.

2 修复 **Zabbix** 数据库字符集与排序规则

MySQL/MariaDB 数据库

1. 检查数据库字符集（character）和排序规则（collation）。

例如:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8mb4                  | utf8mb4_general_ci   |
```

```
+--------------------------+----------------------+
```

如我们所见，此处数据库的字符集不是'utf8'，排序规则不是'utf8_bin'，因此我们需要对其进行修复。

2. 停止 Zabbix 服务。

3. 请务必创建一个数据库备份！

4. 在数据库模式下，修改字符集和排序规则:

```
alter database <您的Zabbix数据库名称> character set utf8 collate utf8_bin;
```

验证修改结果:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8                     | utf8_bin             |
+--------------------------+----------------------+
```

5. 下载脚本 在数据库表和列模式下去修改字符集和排序规格:

```
mysql <您的Zabbix数据库名称> < utf8_convert.sql
```

6. 执行脚本:

```
                SET @ZABBIX_DATABASE = '<您的Zabbix数据库名称>';
If MariaDB →  set innodb_strict_mode = OFF;
                CALL zbx_convert_utf8();
If MariaDB →  set innodb_strict_mode = ON;
                drop procedure zbx_convert_utf8;
```

请注意该操作，数据字符集编码将直接在硬盘上更改。例如，将 Æ, Ñ, Ö 之类的字符从'latin1' 转换成'utf8' 事，他们字节大小，将从 1 字节变成 2 字节。因此，在更改数据库的字符集操作，可能需要比之前更多的空间。

7. 如果没有错误，您可以创建一个新的更改过字符集的数据库备份副本，以备不测。**8.** 启动 Zabbix 服务。

3 数据库的安全连接配置

总览

可以通过以下内容，配置 Mysql 和 PostgreSql 数据库，安全 TLS 加密连接:

- Zabbix 前端
- Zabbix server 或 Zabbix proxy

另请参看: 已知问题 (issues)

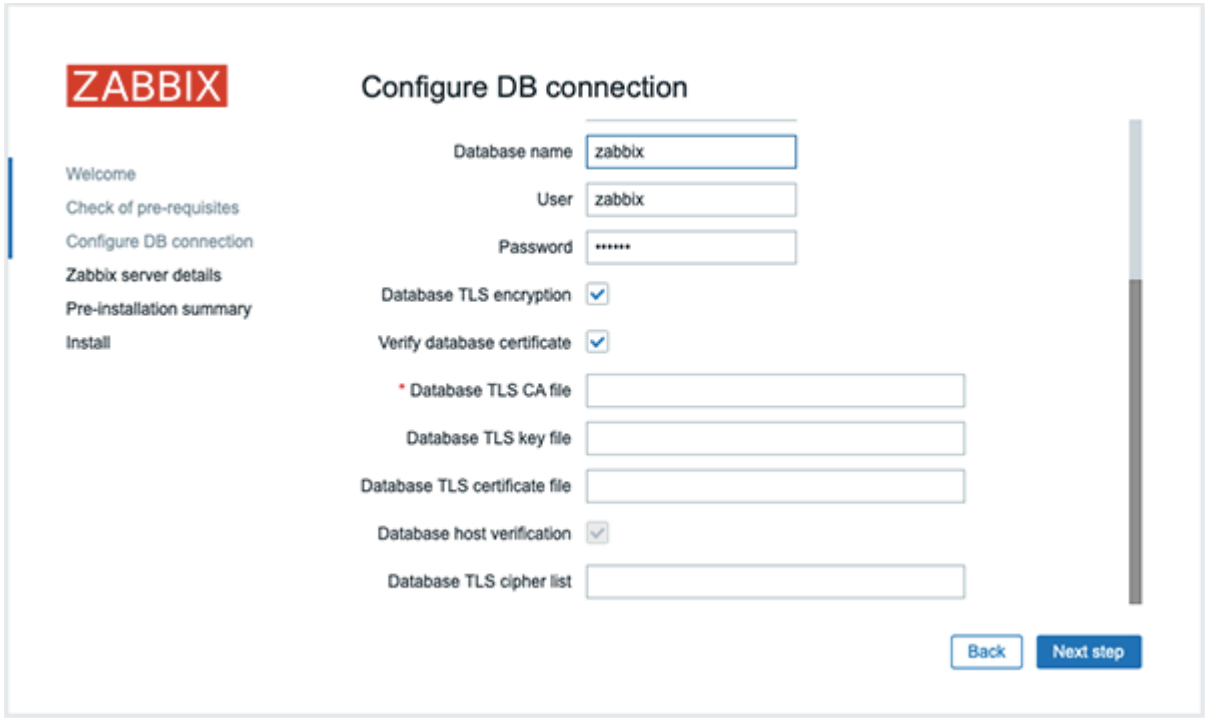
Zabbix 前端配置

> **Note:**
> 自从 Zabbix 5.0.5 版本开始，TLS 加密参数名称已略有更改：为了更方便区分理解，增加了'Database' 数据库前缀。在 Zabbix 5.0.0-5.0.4 版本参数配置名为：*TLS encryption* 加密, *TLS certificate file* 证书文件等。

可以在 Zabbix 初始化安装期间，配置数据库的安全连接参数:

- 勾选 *Database TLS encryption* 数据库 *TLS* 加密复选框Configure DB connection 启用安全传输加密。
- 勾选 *TLS encryption* 加密时，选择勾选 *Verify database certificate* 验证数据库证书复选框，可以检查证书配置是否有效。

> **Note:**
> 从 Zabbix 5.0.5 版本开始: 对于 MySQL 数据库，如果 *Database host* 数据库主机设置成 localhost, 则会禁用 *Database TLS encryption* 数据库 *TLS* 加密复选框，因为使用 socker 套接字文件 (在 Unix 系统下) 或共享内存 (在 Winodws 系统下) 将不能加密。对于 PostgreSql, 如果 *Database host* 数据库主机填写的值开通为斜杠/或空, *TLS encryption* 加密复选项也将不可用。



以下参数在证书配置模式下，TLS 加密选项可用 (从 Zabbix 5.0.5 版本起，仅当两个复选框都被选中时，这些参数才会出现):

参数	描
数据库 *TLS CA* 文件 *(Database TLS CA file)* 指定有效	TLS 证书颁发机构 (CA) 文件的完整路径。
数据库 *TLS* 密钥文件 *(Database TLS key file)* 指定有效的 T	S 密钥文件的完整路径。

参数	描
数据库 *TLS* 证书文件 *(Database TLS certificate file)*	指定有效的 TLS 证书文件的完整路径。
数据库主机验证 *(Database host verification)*	标记此复选框激活主机验证。对于 Mysql 该选项默认禁用，因为 PHP Mysql 类库不充许调过对等证书验证步骤。

参数描	
数据库 *TLS* 密码列表 *(Database TLS cipher list)* 指定有效密码	自定义列表。密码列表的格式必须符合 OpenSSL 标准。仅适用于 MySQL。

<note important>TLS 参数必须指向有效文件。如果它们指向不存在或无效的文件，则将导致认证授权错误。
若果证书文件有写入权限，前端会产生一条系统信息 (System information) 报告，告警内容”TLS 证书文件必须是只读权限。(TLS certificate files must be read-only.)” (仅当 PHP 用户是证书的所有者权限时显示)。

目前不支持受密码保护的证书。. :::

用例

配置结	
无配置 (不标记// 数据库 TLS 加密 (Database TLS encryption)//) 连接到数据库而不进行	密。
1. 仅标记 数据库 *TLS* 加密 *(Database TLS encryption)* 与数据库的安全	LS 连接。

配置结	
1. 标记 数据库 *TLS* 加密 *(Database TLS encryption)* 与数据库的安 2. 指定 TLS 证书颁发机构文件验证数据库服务器证	TLS 连接；，并验证它是否是由受信任的中心签发。

配置结	
1. 标记 数据库 *TLS* 加密 *(Database TLS encryption)* 与数据库的安 2. 指定 TLS 证书颁发机构文件 通过配置将证书中指 3. 标记 验证主机证书 *(With host verification)* 验证证书已由受 4. 指定 TLS 密码列表 (可选)	TLS 连接；的主机名与其所连接的主机名进行比较，来验证数据库服务器证书的有效性；任的机构签发。

配置结	
1. 标记 数据库 *TLS* 加密 *(Database TLS encryption)* 建立到数据库 2. 指定 TLS 秘钥文件 3. 指定 TLS 证书文件 4. 指定 TLS 证书授权文件 5. 标记 //数据库主机验证 (Database host verification) //（在 5.0.5 之前：使用主机验证 *(With host verification)* 6. 指定 TLS 密码列表 (可选)	安全 TLS 连接具有最大的安全性保障。客户端显示证书的要求是在服务器端配置的。

** 另请参见: ** Mysql 配置实例, PostgreSQL 配置实例.

Zabbix server/proxy 配置

数据库安全连接功能，可以通过相关 Zabbix 参数控制配置server 或proxy 配置文件.

配置结	
无	接到数据库而不进行加密。

配置	结
1. 设置 DBTLSConnect=required S	rver/proxy 与数据库建立 TLS 连接。不允许未加密的连接。
1. 设置 DBTLSConnect=verify_ca 验 2. 设置 DBTLSCAFile - 指定 TLS 证书颁发机构文件	数据库证书后，Server/proxy 与数据库建立 TLS 连接。
1. 设置 DBTLSConnect=verify_full 验 2. 设置 DBTLSCAFile - 指定 TLS 证书颁发机构文件	数据库证书和数据库主机身份后，Server/proxy 与数据库建立 TLS 连接。

配置结	
1. 设置 DBTLSCAFile - 指定 TLS 证书颁发机构文件 Server/prox2. 设置 DBTLSCertFile - 指定客户端的公钥证书文件 3. 设置 DBTLSKeyFile - 指定客户端的私钥文件	在连接到数据库时提供客户端证书。
1. 设置 DBTLSCipher - 客户端允许使用直到 TLS 1.2 （MySQL）TLS 或 TLS 1.2 的 TLS 协议进行连接的加密密码的列表-DBSRCipher13-客户端允许使用 TLS 1.3 协议进行连接的加密密码的列表	接是使用提供的列表中的密码进行的。((Post-greSQL）设置此选项将被视为错误。

1 MySQL 加密配置

总览

本节提供了一些关于 CentOS 8.2 和 MySQL 8.0.21 的加密配置示例，并且可以用作数据库加密连接的快速入门指南。加密组合列表不限于此页面上列出的组合。有很多可用的组合。

Attention:
从 Zabbix 5.0.5 版本开始，如果 Mysql 主机设置成 localhost, 则加密选型将不可用。在这种情况下，Zabbix 前端（frontend）与数据库之间的连接使用套接字文件（在 Unix 上）或共享内存（在 Windows 上），并且无法加密。

Note:
在 Zabbix 5.0.5 以前版本，TLS 加密参数名称以来，已略有更改：为了更清楚起见，添加了”Database” 前缀。在 5.0.0-5.0.4 版本中，参数称为 *TLS* 加密 *(TLS encryption)*, *TLS* 证书文件 *(TLS certificate file)* 等。

前提条件

安装 Mysql 数据库，从 官方 Yum 仓库.

有关如何快速使用 Mysql Yum 仓库详细信息，请参考 MySQL 官方文档 。

MySQL 服务器已经准备好使用自签名证书接受安全连接。

若要查看哪些用户正在使用加密的连接，请运行以下查询（性能模式 (Performance Schema) 应打开）：

```
mysql> SELECT sbt.variable_value AS tls_version, t2.variable_value AS cipher, processlist_user AS user, pr
        FROM performance_schema.status_by_thread  AS sbt
        JOIN performance_schema.threads AS t ON t.thread_id = sbt.thread_id
        JOIN performance_schema.status_by_thread AS t2 ON t2.thread_id = t.thread_id
        WHERE sbt.variable_name = 'Ssl_version' and t2.variable_name = 'Ssl_cipher'
        ORDER BY tls_version;
```

必须模式

MySQL 配置

数据库的当前版本可直接使用’ 必须 (required)’ 加密模式。初始设置和启动后，将创建服务器端证书。

为主要组件创建数据库用户和角色:

```
mysql> CREATE USER
 'zbx_srv'@'%' IDENTIFIED WITH mysql_native_password BY '<健壮的复杂密码>',
 'zbx_web'@'%' IDENTIFIED WITH mysql_native_password BY '<健壮的复杂密码>'
 REQUIRE SSL
 PASSWORD HISTORY 5;

mysql> CREATE ROLE 'zbx_srv_role', 'zbx_web_role';

mysql> GRANT SELECT, UPDATE, DELETE, INSERT, CREATE, DROP, ALTER, INDEX, REFERENCES ON zabbix.* TO 'zbx_sr
mysql> GRANT SELECT, UPDATE, DELETE, INSERT ON zabbix.* TO 'zbx_web_role';

mysql> GRANT 'zbx_srv_role' TO 'zbx_srv'@'%';
mysql> GRANT 'zbx_web_role' TO 'zbx_web'@'%';

mysql> SET DEFAULT ROLE 'zbx_srv_role' TO 'zbx_srv'@'%';
mysql> SET DEFAULT ROLE 'zbx_web_role' TO 'zbx_web'@'%';
```

请注意，X.509 协议不用于检查身份，而是将用户配置为仅使用加密连接。有关配置用户的更多详细信息，请参见MySQL 官方文档。

远程连接验证 (socket 套接字连接不能用于测试安全连接):

```
$ mysql -u zbx_srv -p -h 10.211.55.9 --ssl-mode=REQUIRED
```

检查当前状态和可用的密码组合:

```
mysql> status
--------------
mysql Ver 8.0.21 for Linux on x86_64 (MySQL Community Server - GPL)

Connection id: 62
Current database:
Current user: zbx_srv@bfdb.local
SSL: Cipher in use is TLS_AES_256_GCM_SHA384


mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher_list'\G;
*************************** 1. row ***************************
Variable_name: Ssl_cipher_list
Value: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_SHA256:E
1 row in set (0.00 sec)
```

```
ERROR:
No query specified
```

前端

要为 Zabbix frontend 和数据库之间的连接启用仅传输加密：

- 检查 数据库 *TLS* 加密 *(Database TLS encryption)*
- 不选中 验证数据库证书 *(Verify database certificate)*

Server

要为服务器和数据库之间的连接启用仅传输加密，请配置/etc/zabbix/zabbix_server.conf：

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<健 壮 的 复 杂 密 码>
DBTLSConnect=required
...
```

验证 CA 模式

将所需的 MySQL CA 复制到 Zabbix frontend 服务器，分配适当的权限以允许 Web 服务器读取此文件。

前端 (Frontend)

为 Zabbix 前端 (frontend) 和数据库之间的连接启用带有证书验证的加密：

- 检查 数据库 *TLS* 加密 *(Database TLS encryption)* 和 验证数据库证书 *(Verify database certificate)*
- 指定数据库 TLS CA 文件的路径

或者，可以在/etc/zabbix/web/zabbix.conf.php 中进行设置：

```
...
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = false;
$DB['CIPHER_LIST'] = '';
...
```

使用命令行工具对用户进行故障排除，以检查所需用户是否可以建立连接：

```
$ mysql -u zbx_web -p -h 10.211.55.9 --ssl-mode=REQUIRED --ssl-ca=/var/lib/mysql/ca.pem
```

Zabbix Server

要为 Zabbix server 和数据库之间的连接启用带有证书验证的加密，请配置/etc/zabbix/zabbix_server.conf：

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/mysql/ca.pem
...
```

验证完全模式 (Full mode)

MySQL 配置

将 MySQL CE 服务器配置选项（/etc/my.cnf.d/server-tls.cnf）设置为：

```
[mysqld]
...
# 在 此 示 例 中 ， 配 置 位 于MySQL CE datadir目 录 中
ssl_ca=ca.pem
ssl_cert=server-cert.pem
ssl_key=server-key.pem
```

```
require_secure_transport=ON
tls_version=TLSv1.3
...
```

应根据 MySQl CE 文档手动创建 MySQL CE 服务器和客户端的密钥 (Zabbix 前端)使用 MySQL 创建 SSL 和 RSA 证书和密钥文档 or 使用 openssl 创建 SSL 证书和密钥文档

> **Attention:**
> MySQL 服务器证书应包含设置为 FQDN 名称的 Common Name 字段，因为 Zabbix 前端 (frontend) 将使用 DNS 名称与数据库通信或数据库主机的 IP 地址。

创建 MySQL 用户：

```
mysql> CREATE USER
  'zbx_srv'@'%' IDENTIFIED WITH mysql_native_password BY '<健壮的复杂密码>',
  'zbx_web'@'%' IDENTIFIED WITH mysql_native_password BY '<健壮的复杂密码>'
  REQUIRE X509
  PASSWORD HISTORY 5;
```

检查是否可以使用该用户登录：

```
$ mysql -u zbx_web -p -h 10.211.55.9 --ssl-mode=VERIFY_IDENTITY --ssl-ca=/var/lib/mysql/ca.pem --ssl-cert=
```

前端

要对 Zabbix 前端和数据库之间的连接进行完整验证的加密，请执行以下操作：

- 检查数据库 TLS 加密并验证数据库证书
- 指定数据库 TLS 密钥文件的路径
- 指定数据库 TLS CA 文件的路径
- 指定数据库 TLS 证书文件的路径

请注意，数据库主机验证 *(Database host verification)* 已选中并显示为灰色-MySQL 不能跳过此步骤。

> **Warning:**
> 密码列表应该为空，以便前端和服务器可以从两端支持的服务器中协商所需的一个。

或者，可以在/etc/zabbix/web/zabbix.conf.php 中进行设置：

```
...
// 用于严格定义密码列表的TLS连接。
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = 'TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_1
...
// or
...
// 用于未定义密码列表的TLS连接-由MySQL服务器选择
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';
...
```

Zabbix Server

要通过对 Zabbix server 和数据库之间的连接进行全面验证来启用加密，请配置/etc/zabbix/zabbix_server.conf：

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
```

```
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/mysql/ca.pem
DBTLSCertFile=/etc/ssl/mysql/client-cert.pem
DBTLSKeyFile=/etc/ssl/mysql/client-key.pem
...
```

2 PostgreSQL 加密配置

总览

本节提供了一些针对 CentOS 8.2 和 PostgreSQL 13 的加密配置示例。

> **Note:**
> 如果 数据库主机 *(Database host)* 字段的值以/斜杠开头或该字段为空，则无法加密 Zabbix 前端与 PostgreSQL 之间的连接（禁用了 GUI 中的参数）。

> **Note:**
> 从 Zabbix 5.0.5 版本，TLS 加密参数名称，已略有更改：为了更清楚起见，添加了" 数据库 (Database)" 前缀。在 5.0.0-5.0.4 版中，参数名称为 *TLS* 加密 *(TLS encryption)*, *TLS* 证书文件 *(TLS certificate file)* 等。

前提条件

使用官方仓库安装 PostgreSQL 数据库。

若 PostgreSQL 没有配置接受 TLS 连接。请按照文档使用 postgresql.conf 准备配置证书，以及通过 ph_hba.conf 进行用户访问控制配置文档。

默认情况下，PostgreSQL socket 套接字绑定到本机 (localhost)，因为网络远程连接允许监听真实的网络接口。

所有模式的 PostgreSQL 设置如下所示：

/var/lib/pgsql/13/data/postgresql.conf:

```
...
ssl = on
ssl_ca_file = 'root.crt'
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_min_protocol_version = 'TLSv1.3'
...
```

对于用户访问控制，请调整 */var/lib/pgsql/13/data/pg_hba.conf*：

```
...
### require
hostssl all all 0.0.0.0/0 md5

### verify CA
hostssl all all 0.0.0.0/0 md5 clientcert=verify-ca

### verify full
hostssl all all 0.0.0.0/0 md5 clientcert=verify-full
...
```

必须模式（Required mode）

前端

要为 Zabbix 前端和数据库之间的连接启用仅传输加密：

- 检查 数据库 *TLS* 加密 *(Database TLS encryption)*
- 不选择 验证数据库证书 *(Verify database certificate)*

Server

要为 Zabbix server 和数据库之间的连接启用仅传输加密，请配置/etc/zabbix/zabbix_server.conf：

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<健壮的复杂密码>
DBTLSConnect=required
...
```

验证 CA 模式 (Verify CA mode)

Frontend

To enable encryption with certificate authority verification for connections between Zabbix frontend and the database:

- 检查 数据库 *TLS* 加密 *(Database TLS encryption)* 和 验证数据库证书 *(Verify database certificate)*
- 指定 数据库 *TLS* 密钥文件的路径
- 指定 数据库 *TLS CA* 文件的路径
- 指定 数据库 *TLS* 证书文件的路径

或者，可以在 */etc/zabbix/web/zabbix.conf.php:* 中进行设置：

```
...
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt';
$DB['VERIFY_HOST'] = false;
$DB['CIPHER_LIST'] = '';
...
```

Server

要为 Zabbix server 和数据库之间的连接启用带有证书验证的加密，请配置 */etc/zabbix/zabbix_server.conf:*

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<健壮的复杂的密码>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/pgsql/root.crt
...
```

验证完整模式 (full mode)

前端

为 Zabbix 前端和数据库之间的连接启用使用证书和数据库主机身份验证的加密：

- 检查 数据库 *TLS* 加密 *(Database TLS encryption)* 和 验证数据库证书 *(Verify database certificate)*
- 指定 数据库 *TLS* 密钥文件的路径 *(Database TLS key file)*
- 指定 数据库 *TLS CA* 文件的路径 *(Database TLS CA file)*
- 指定 数据库 *TLS* 证书文件的路径 *(Database TLS certificate file)*
- 检查 数据库主机验证 *(Database host verification)*

或者，可以在*/etc/zabbix/web/zabbix.conf.php* 中进行设置：

```
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';
...
```

Server

要为 Zabbix server 和数据库之间的连接启用使用证书和数据库主机身份验证的加密，请配置*/etc/zabbix/zabbix_server.conf*:

```
...
DBHost=10.211.55.9
DBName=zabbix
```

```
DBUser=zbx_srv
DBPassword=<健壮的复杂密码>
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/pgsql/root.crt
DBTLSCertFile=/etc/ssl/pgsql/client.crt
DBTLSKeyFile=/etc/ssl/pgsql/client.key
...
```

4 TimescaleDB 配置

概述

Zabbix 支持时序数据库 TimescaleDB，这是一种基于 PostgreSQL 的数据库解决方案，可将数据自动划分为基于时间的块，以支持更快的大规模性能。

> **Warning:**
> 目前时序数据库不支持 Zabbix proxy。

本页上的说明可用于创建 TimescaleDB 数据库或从现有 PostgreSQL 表迁移到 TimescaleDB。

配置

我们假设 TimescaleDB 扩展项已经安装在数据库服务器上（查看 安装说明）。

还必须通过执行以下命令为特定的数据库启用 TimescaleDB 扩展项：

```
echo "CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;" | sudo -u postgres psql zabbix
```

运行此命令需要数据库管理员权限。

> **Note:**
> 如果你使用的数据库 schema 不是'public' 模式则需要通过以上命令添加 SCHEMA 子句，例如：
> `echo "CREATE EXTENSION IF NOT EXISTS timescaledb SCHEMA yourschema CASCADE;" | sudo -u postgres psql zabbix`

然后运行位于 database/postgresql 中的 `timescaledb.sql` 脚本。对于新的安装，必须在使用初始 schema/data 创建常规 PostgreSQL 数据库之后再运行脚本。(查看数据库创建):

```
cat timescaledb.sql | sudo -u zabbix psql zabbix
```

现有历史记录和趋势数据的迁移可能需要很多时间。在迁移期间。Zabbix Server 和前端必须关闭。

`timescaledb.sql` 脚本设置以下内置数据管理 housekeeping 参数：

- 覆盖监控项趋势周期 Override item history period
- 覆盖监控项历史周期 Override item trend period

为了将用于历史和趋势的内置数据管理进行分区，这两个选项都必须启用。可以将 TimescaleDB 分区仅用于趋势（通过设置覆盖监控项趋势周期 *Override item trend period*）或仅用于历史记录（覆盖监控项历史周期 *Override item history period*）。

对于 PostgreSQL 10.2 版以及 TimescaleDB 1.5 版或更高版本，`timescaledb.sql` 脚本设置了两个附加参数：

- 启用压缩
- 压缩七天以上的记录

所有这些参数都可以于安装之后在 *Administration → General → Housekeeping* 进行修改。

<note tip> 您可能需要运行 TimescaleDB 提供的 timescaledb-tune 工具对 `postgresql.conf` 中的 PostgreSQL 配置参数进行优化。
:::

Compression can be used only if both *Override item history period* and *Override item trend period* options are enabled.

All of these parameters can be changed in *Administration → General → Housekeeping* after the installation.

> **Note:**
> You may want to run the timescaledb-tune tool provided by TimescaleDB to optimize PostgreSQL configuration parameters in your `postgresql.conf`.

时序数据库压缩

从 Zabbix 5.0 开始，原生数据库压缩已在 PostgreSQL 10.2 版以及 TimescaleDB 1.5 版或更高版本的时序数据库所管理的全部 Zabbix 表中得到支持。在升级或迁移到时序数据库的过程中，大型表的初始压缩可能需要很多时间。

> **Note:**
> 推荐用户在使用压缩之前熟悉 TimescaleDB 压缩说明文档。

注意, 压缩是有一定限制的, 确切地说：

- 压缩块的编辑 (插入，删除，更新) 是不支持的
- 压缩表的架构更改是不支持的

压缩设置可以在位于 Zabbix 前端 *Administration* → *General* → *Housekeeping* 中的 *History and trends compression* 项中修改。

参数默	注释
Enable compression	Enabled

选中或取消选中该复选框不会立即激活/禁用压缩。由于压缩是由内置数据管理机制 House-keeper 处理的，因此更改最多将在 2 倍 Housekeeping 时间后生效 (zab-bix_server.conf 中设置)

禁用压缩后，

参数默	注释
Compress records older than	7d 此参数不能少于7天。 由于压缩块的不可变性，所有早于此值的后期数据(例如，代理延迟的数据)将被丢弃。

5 Elasticsearch 配置

Attention:
目前 Zabbix 对 Elasticsearch 的支持，仍在试验阶段！

Zabbix 支持通过 Elasticsearch，而不使用数据库来存储历史数据。用户可以在兼容的数据库和 Elasticsearch 之间来选择历史数据的存储位置。本章中所描述的设置过程适用于 Elasticsearch 7.X 版本。如果使用了较早或更高的版本，某些功能则可能会无法正常工作。

Warning:
如果所有历史数据都存储在 Elasticsearch 上，将不会计算趋势，也不会存储在数据库中。如果没有计算和存储趋势，历史数据保留时长可能需要延长。

配置

为保证涉及的所有元素之间能正常通信，请确保正确配置了 Zabbix server 及其前端配置文件的参数。

Zabbix server 和前端

在 Zabbix server 初始的配置文件中，需要更新如下参数：

```
### 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
```

例如使用以下示例参数值，来设置 Zabbix server 的配置文件：

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

使用此配置，Zabbix server 会将数值类型的历史数据存储在相应的数据库中，将文本历史数据存储在 Elasticsearch 中。

Elasticsearch 支持以下几种监控项类型：

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

支持的监控项类型说明：

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

Zabbix 前端配置文件 (`conf/zabbix.conf.php`) 中，需要更新如下参数：

```
// 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'];
```

例如使用以下示例参数值，来设置 Zabbix 前端的配置文件：

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

使用此配置，文本、字符和日志类型的历史数据将存储到 Elasticsearch 中。

您还需要将 `conf/zabbix.conf.php` 文件的中 `$HISTORY` 配置为全局参数，以确保一切正常（了解如何配置，请参考 `conf/zabbix.conf.php.example`）：

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

Elasticsearch 安装及创建映射

使其正常运行的最后两个步骤是安装 Elasticsearch 和创建映射。

安装 Elasticsearch，请参考 Elasticsearch 安装指南 。

> **Note:**
> 映射是 Elasticsearch 中的一种数据结构（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 warning> 创建映射是强制性的。如果未按照要求创建映射，则某些功能将无法正常使用。:::

创建 `text` 类型的映射，可以发送如下请求到 Elasticsearch：

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

对于创建字符和日志类型的历史数据映射并有相应类型的修改，也需要执行类似的请求。

> **Note:**
> 要使用 Elasticsearch，请参考安装条件页面 以获取更多信息。

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

在多个基于日期的索引中存储历史数据

本节将介绍使用 pipelines 和 ingest 节点所需的其他配置步骤。

首先，您必须为索引创建一个模板。

创建 uint 模板的请求示例如下：

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_template/uint_template \
 -H 'content-type:application/json' \
 -d '{
   "index_patterns": [
      "uint*"
   ],
   "settings": {
      "index": {
         "number_of_replicas": 1,
         "number_of_shards": 5
```

```
      }
   },
   "mappings": {
      "properties": {
         "itemid": {
            "type": "long"
         },
         "clock": {
            "format": "epoch_second",
            "type": "date"
         },
         "value": {
            "type": "long"
         }
      }
   }
}'
```

若要创建其他模板，用户需要修改 URL（最后一部分是模板名称），更改"index_patterns" 字段以匹配索引名称并设置有效的映射，这些映射可以在 database/elasticsearch/elasticsearch.map 中获取。

例如：我们可以使用以下命令，来为文本索引创建一个模板：

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

这是允许 Elasticsearch 为自动创建的索引设置有效映射所必需的。然后需要创建 pipeline 定义。Pipeline 是在将数据放入索引之前，对数据的某种预处理。可以使用以下命令，为 uint 索引创建 pipeline:

```
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 rounding 参数（“date_rounding”）来设置特定的索引循环周期。要创建其他的 pipeline，用户需要修改 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。

请参考 Elasticsearch 文档。

另外，还需要在 Zabbix server 配置中，加入新参数来启用在多个基于日期的索引中存储历史数据。

```
### 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

以下步骤可帮助您解决 Elasticsearch 的配置问题：

1. 检查映射是否正确 (向 URL 发送 GET 请求获取索引信息，例如：`http://localhost:9200/uint`)。
2. 检查分片状态是否处于失败状态 (可以通过重启 Elasticsearch 来解决)。
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。

如果您仍然遇到安装问题，请创建一个 bug 报告，其中需包含该列表中的所有信息（映射、错误日志、配置、版本等信息）。

6 实时导出事件，监控项采集值，趋势数据

概述

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

导出完成后的文件中，每一行都是一个 JSON 对象。值映射不被应用。

如果出现错误（导出文件无法写入数据、无法重命名导出文件或重命名后无法创建新的导出文件），数据项将被删除，并且永远不会写入导出文件。它只写入 Zabbix 数据库中。当写入问题解决后，即可恢复将数据写入导出文件的操作。

有关导出数据的详细信息，请参考导出协议 页面。

注意：如果在收到数据后、服务器导出数据之前，删除了主机/监控项，那么主机/监控项将没有元数据（例如：主机组、主机名、监控项名称等）。

配置

我们通过为导出文件指定目录，来配置实时导出触发器事件、监控项采集值和趋势数据。请参考服务器配置 中 `ExportDir` 参数。

另外两个可用的参数是：

- `ExportFileSize` 可以用来设置单个导出文件的最大允许大小。当一个进程需要写入文件时，它首先会检查文件的大小。如果超出了配置的大小限制，则将在文件名后加上.old 来重命名该文件，并会创建一个具有原文件名的新文件。

Attention:
每个将写入数据的进程都将会创建一个文件 (例如：approximately 4-30 files)。由于每个导出文件的默认大小是 1G，保留较大的导出文件可能会很快耗尽磁盘空间。

- ExportType 允许指定要导出的实体类型（事件、历史数据和趋势数据）。从 Zabbix 5.0.10 开始，支持此参数。

7 关于 **Zabbix** 配置 **Nginx** 的特别说明

SLES 12

在 SUSE Linux Enterprise Server 12 中，需要在安装 Nginx 之前添加 Nginx 源:

```
zypper addrepo -G -t yum -c 'http://nginx.org/packages/sles/12' nginx
```

同时还需要配置 php-fpm:

```
cp /etc/php5/fpm/php-fpm.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php5/fpm/php-fpm.conf
```

SLES 15

在 SUSE Linux Enterprise Server 15 中需要配置 php-fpm:

```
cp /etc/php7/fpm/php-fpm.conf{.default,}
cp /etc/php7/fpm/php-fpm.d/www.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php7/fpm/php-fpm.d/www.conf
```

8 使用 **root** 权限运行 **agent**

从 **5.0.0** 版本开始 官方软件包 中 Zabbix agent 的 systemd 服务文件已更新为明确包含 User and Group 的指令. 两者均设置为 zabbix。

这意味着通过 zabbix_agentd.conf 配置文件中指定用户运行 Zabbix Agent 的功能会被绕过，Zabbix agent 将使用 systemd 服务文件中指定的用户运行服务。

若要修改 Zabbix Agent 服务运行的用户，请创建新的文件/etc/systemd/system/zabbix-agent.service.d/override.conf 并包含以下内容：

```
[Service]
User=root
Group=root
```

重新加载守护程序并重新启动 zabbix-agent 服务：

```
systemctl daemon-reload
systemctl restart zabbix-agent
```

对于 **Zabbix agent2**，这完全取决于它运行的用户角色。

对于旧 **agent**，指定服务运行的用户功能需要在 zabbix_agentd.conf 文件中进行配置。因此要以 root 用户身份运行 zabbix agent，您仍需编辑配置文件并指定 User=root 和 AllowRoot=1 选项。

Zabbix agent

To override the default user and group for Zabbix agent, run:

```
systemctl edit zabbix-agent
```

Then, add the following content:

```
[Service]
User=root
Group=root
```

Reload daemons and restart the zabbix-agent service:

```
systemctl daemon-reload
systemctl restart zabbix-agent
```

For **Zabbix agent** this re-enables the functionality of configuring user in the zabbix_agentd.conf file. Now you need to set User=root and AllowRoot=1 configuration parameters in the agent configuration file.

Zabbix agent 2

To override the default user and group for Zabbix agent 2, run:

```
systemctl edit zabbix-agent2
```

Then, add the following content:

```
[Service]
User=root
Group=root
```

Reload daemons and restart the zabbix-agent service:

```
systemctl daemon-reload
systemctl restart zabbix-agent2
```

For **Zabbix agent2** this completely determines the user that it runs as. No additional modifications are required.

9 Microsoft Windows 下的 Zabbix agent

配置 agent

Zabbix agent 以 Windows 服务运行。

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)。

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

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

> **Warning:**
> **Windows Zabbix agent** 不支持 **CPU** 在 **NUMA** 节点上非均匀分布的非标准 **Windows** 配置。如果逻辑 CPU 的分布不均匀，那么某些 CPU 可能无法获得 CPU 性能指标。例如，如果有 72 个逻辑 CPU 和 2 个 NUMA 节点，那么两个节点都必须有 36 个 cpu。

主机名参数

要在主机上执行主动检查 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。

agent 端的主机名可以通过配置文件中的 **Hostname** 或 **HostnameItem** 参数定义 - 如果没有指定这些参数，则使用默认值。

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。

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。

<note important>**system.hostname** 参数始终返回 NetBIOS 主机名，该主机名限制在 15 个符号以内，并且只包含大写字母 - 而不管实际主机名中的长度和大小写字母。:::

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

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

因此，为了简化 zabbix_agentd.conf 文件的配置并使其统一，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 将使用完整的真实的（区分大小写的）Windows 主机名作为主机名。

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 服务安装”`Zabbix Agent [Windows_db_server]`”。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。

Windows 下安装 agent 服务

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

```
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。:::

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

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

应指定配置文件的完整路径。

Zabbix agent 多实例作为服务安装的命令如下:

```
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。

启动 agent

启动 agent 服务，可以使用控制面板或通过命令行方式。

启动使用默认配置文件的单实例 Zabbix agent 命令如下：

```
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：

```
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:

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

停止 agent

停止 agent 服务，可以使用控制面板或通过命令行方式。

停止使用默认配置文件的单实例 Zabbix agent 命令如下：

```
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：

```
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:

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

Windows 下卸载 agent 服务

卸载使用默认配置文件的单实例 Zabbix agent 服务命令如下：

```
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：

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

卸载多实例 Zabbix agent 服务命令如下:

```
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
```

11 Additional frontend languages

Overview

In order to use any other language than English in Zabbix web interface, its locale should be installed on the web server. Additionally, the PHP gettext extension is required for the translations to work.

If a locale is installed, a language becomes available in the language selector in Zabbix web interface. Languages for which locales are not installed are greyed out and cannot be selected.

Installing locales

To list all installed languages, run:

```
locale -a
```

If some languages that are needed are not listed, open the */etc/locale.gen* file and uncomment the required locales. Since Zabbix uses UTF-8 encoding, you need to select locales with UTF-8 charset.

Now, run:

```
locale-gen
```

Restart the web server.

The locales should now be installed. It may be required to reload Zabbix frontend page in browser using Ctrl + F5 for new languages to appear.

Installing Zabbix

If installing Zabbix directly from Zabbix git repository, translation files should be generated manually. To generate translation files, run:

```
make gettext
locale/make_mo.sh
```

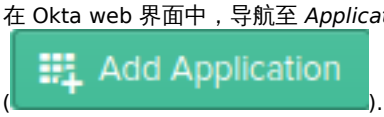
This step is not needed when installing Zabbix from packages or source tar.gz files.

11 使用 **OKTA** 进行 **SAML** 设置

本节介绍如何配置 Okta，为 Zabbix 启用 SAML 2.0 身份验证。

OKTA 配置

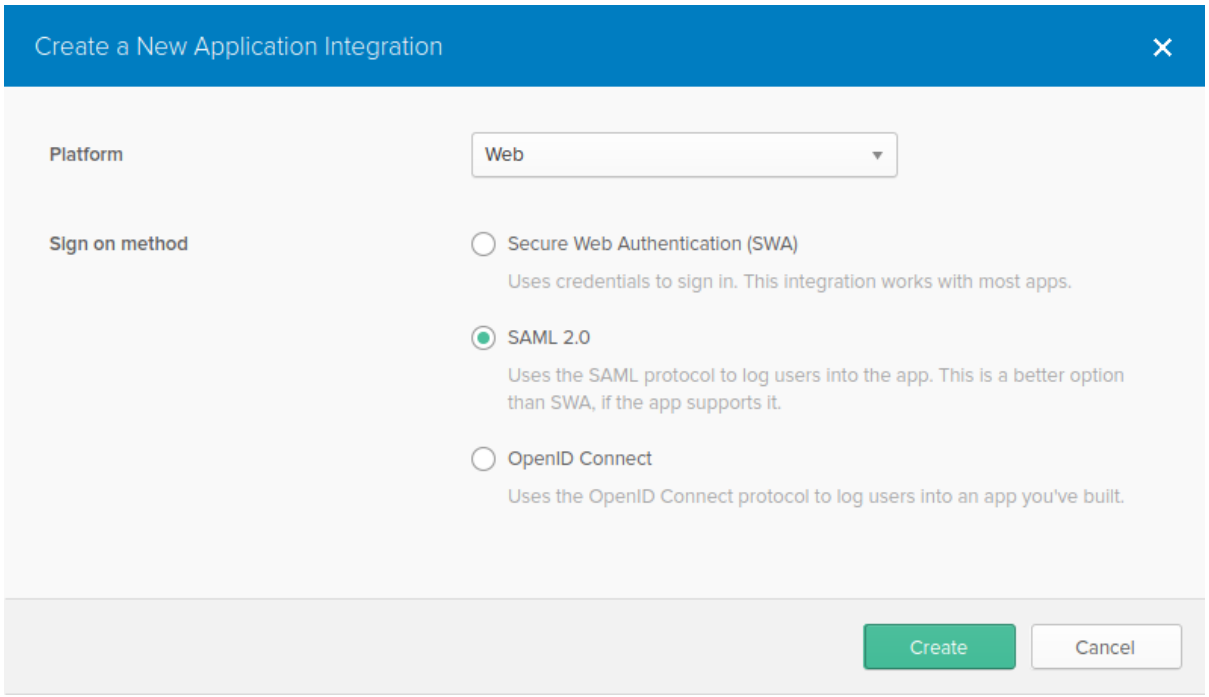
1. 访问 https://okta.com 并注册或登录您的账户。

2. 在 Okta web 界面中，导航至 *Applications*（应用程序）→ *Applications*（应用程序），然后按”Add Application（添加应用程序）” 按钮 (

).

3. 按”Create New App（创建应用集）” 按钮 (

). 在弹出窗口中，选择 *Platform*（平台）*:* Web, *Sign on method*（登录方法）*:* SAML 2.0，然后按”Create（创建）” 按钮。

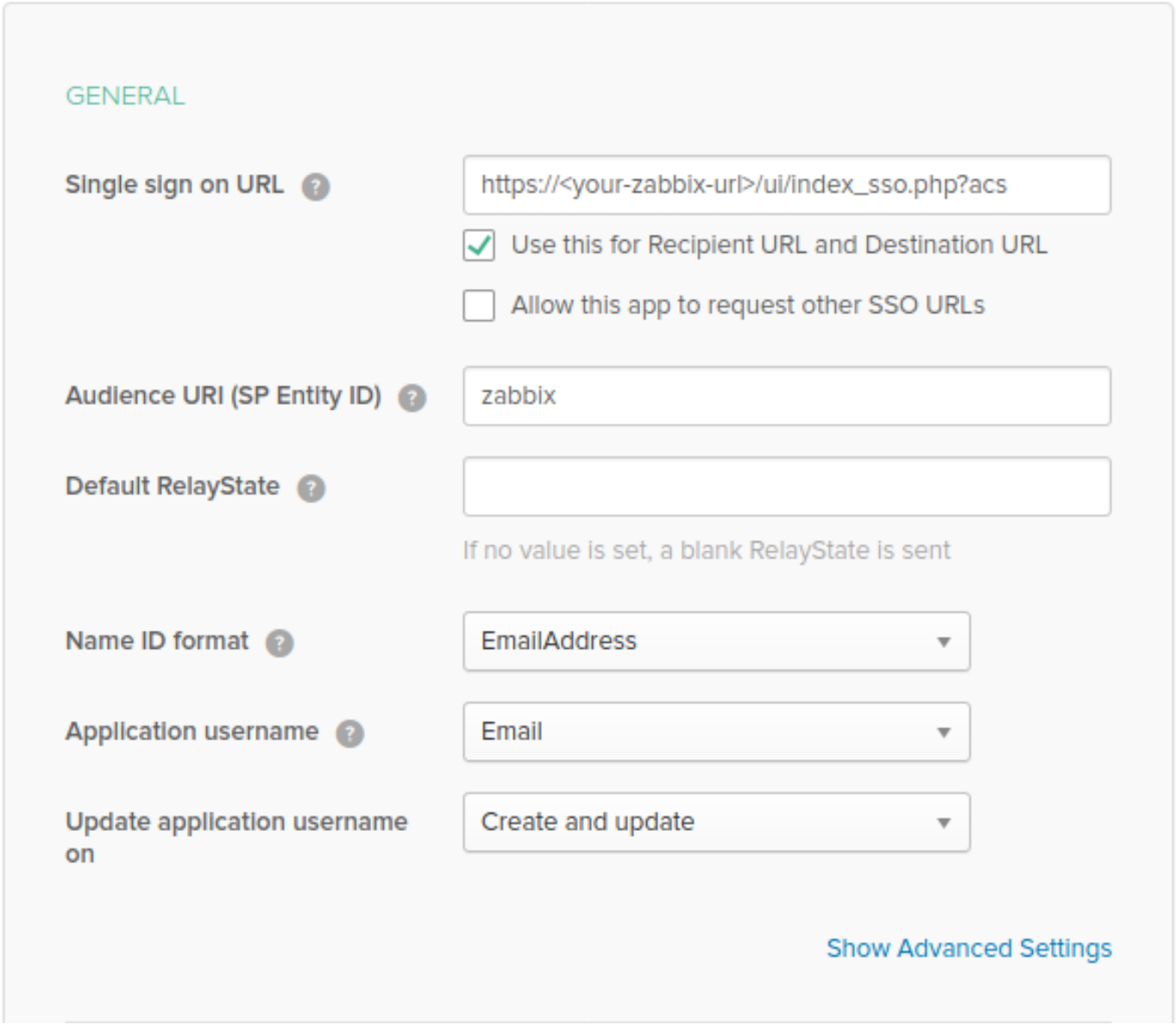


4. 根据您的喜好填写 *General settings*（常规设置）选项卡 (出现第一个选项卡) 中的字段，然后按”Next（下一步）”。

5. 在 *Configure SAML*（配置 *SAML*）选项卡中，输入以下提供的值，然后按”Next（下一步）”.

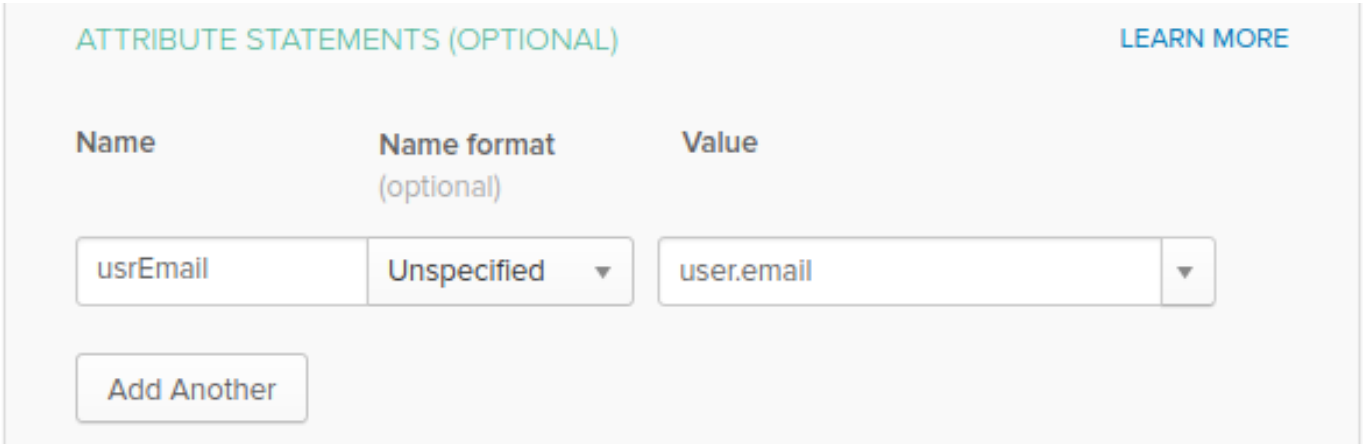
- 在 **GENERAL**（常规）部分中：
  - *Single sign on URL*（登录 *URL*）：https://<your-zabbix-url>/ui/index_sso.php?acs
    选中 *Use this for Recipient URL and Destination URL*（用于接收 *URL* 和目标 *URL*）的复选框。

- *Audience URI (SP Entity ID)*: zabbix
  注意，此值将在 SAML 声明中用作唯一的服务提供者标识符 (如果不匹配，则将拒绝该操作)。可在此字段中指定 URL 或任何数据字符串。
- *Default RelayState*:
  将此字段留空；如果需要自定义重定向，则可以在 Zabbix 的 *Administration*（管理）→ *Users*（用户）设置中添加它。
- 根据您的喜好填写其它字段。



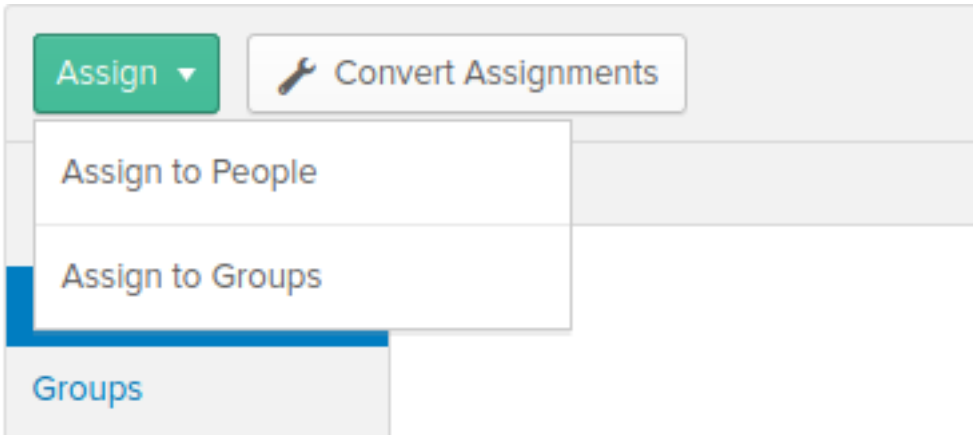
Note:
如果计划使用加密连接，请生成专用和公用加密证书，然后将公用证书上传到 Okta。当 *Assertion Encryption*（断言加密）设置为 “已加密” 时，将显示证书上传表单 (单击 *Show Advanced Settings*（显示高级设置）以找到此参数).

- 在 **ATTRIBUTE STATEMENTS (OPTIONAL)** 部分中，添加带有以下内容的属性语句：
  - *Name*（名称）*：* 输入您的电子邮箱名称
  - *Name format*（名称格式）*：* Unspecified
  - *Value*（值）*：* 输入您的电子邮箱地址



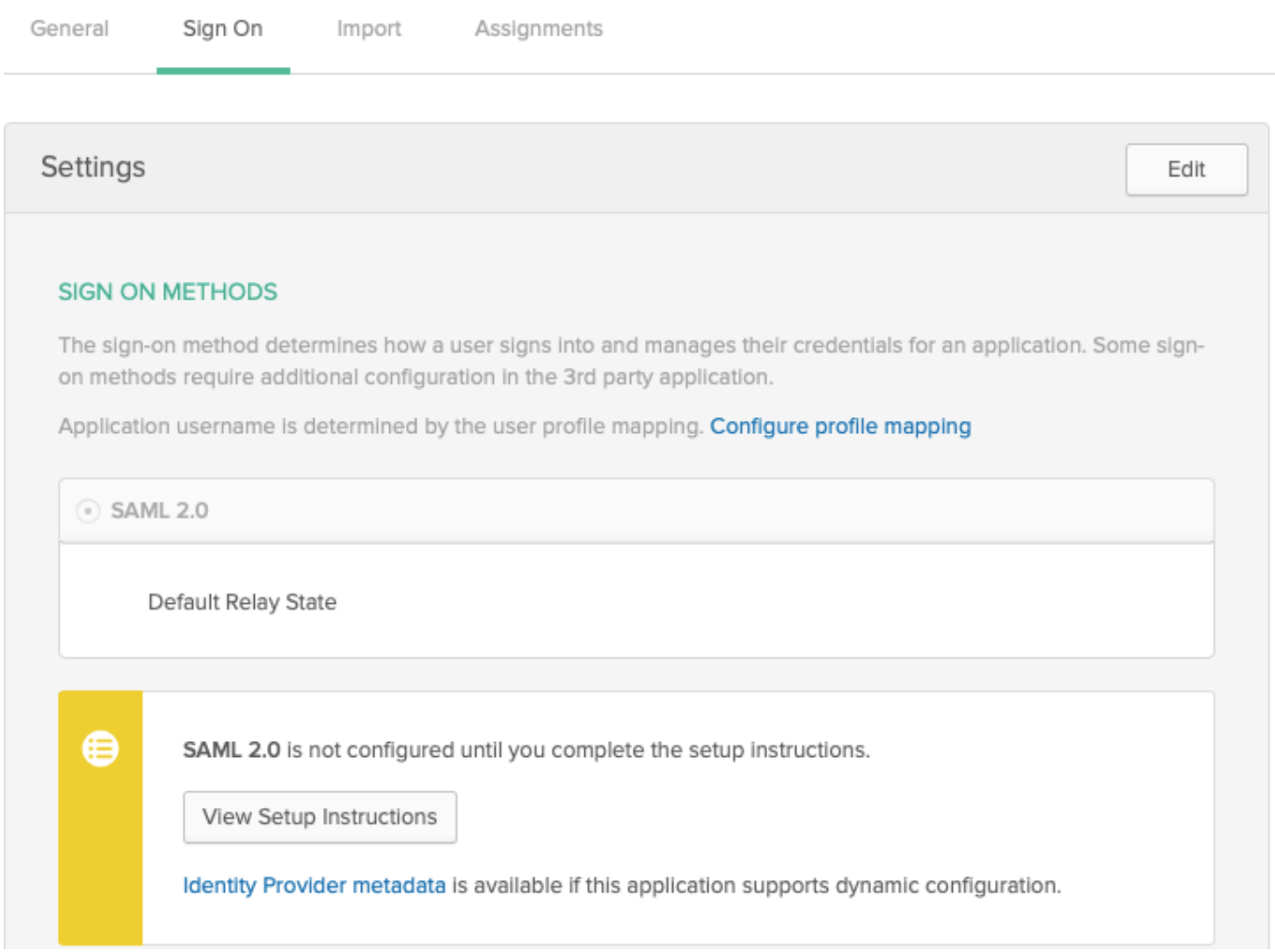
6. 在下一个选项卡中，选择”I’m a software vendor. I’d like to integrate my app with Okta”，然后按”Finish”.

7. 现在，导航到 *Assignments*（分配）选项卡，然后按”Assign（指定）” 按钮，然后从下拉菜单中选择 *Assign to People*。



8. 在弹出窗口中，将创建的应用分配给将使用 SAML 2.0 与 Zabbix 进行身份验证的人员，然后按”Save and go back”.

9. 导航到 *Sign On*（登录）选项卡，然后按”View Setup Instructions” 按钮。设置说明将显示在新选项卡中；在配置 Zabbix 时，请保持此标签打开。



Zabbix 配置

1. 在 Zabbix 中，进入 *Administration* → *Authentication* 部分中的 SAML 设置，然后将信息从 Okta 设置说明复制到相应的字段中：

- 身份提供者单点登录 URL → SSO 服务 URL
- 身份提供商发布者 → IdP entity ID
- 用户名属性 → 属性名（usrEmail）
- SP 实体 ID → 受众 URI

2. 将 Okta 设置说明页面中提供的证书作为 idp.crt 下载到 *ui/conf/certs* 文件夹中，并通过运行以下命令设置权限 644：

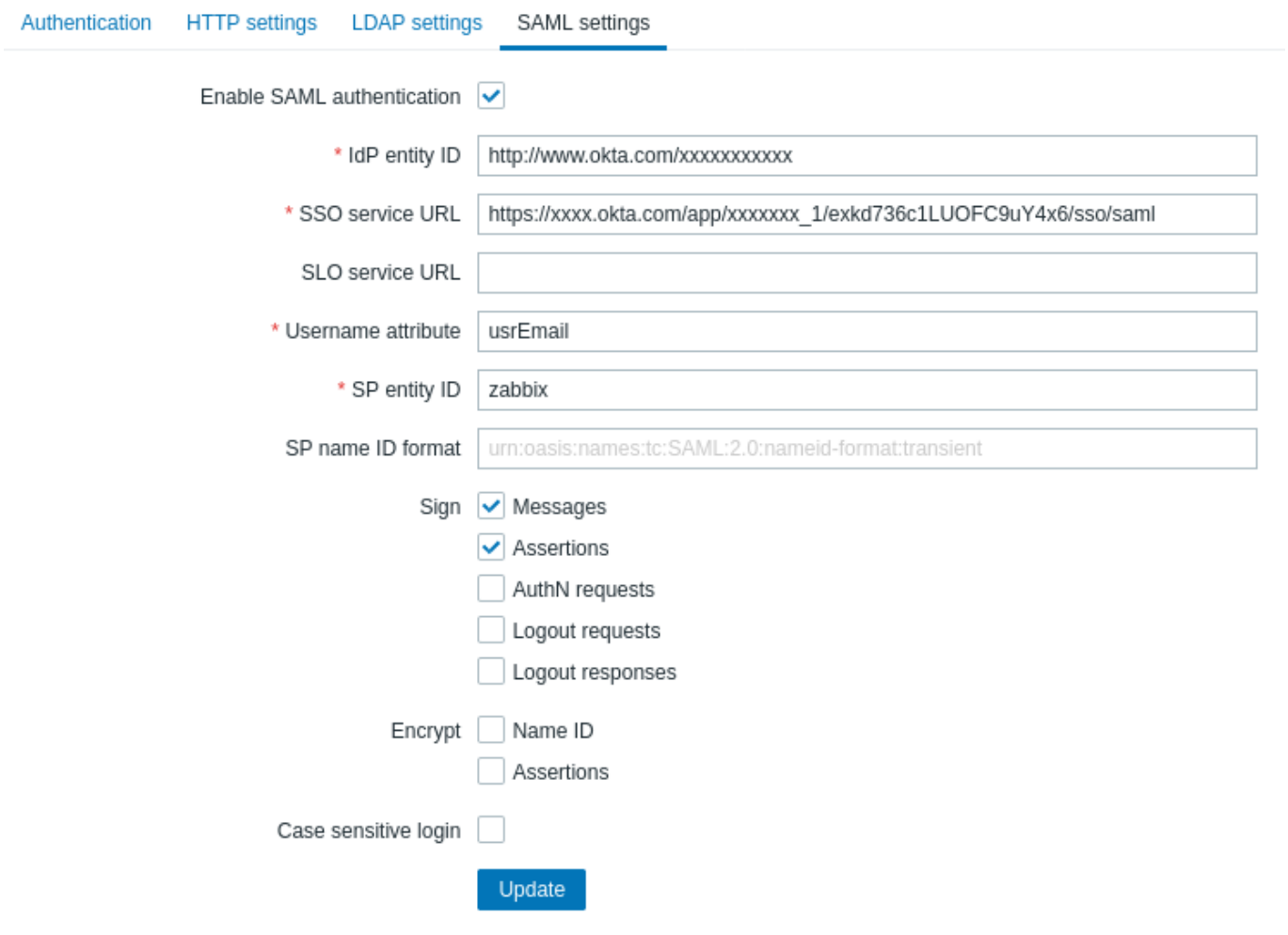
```
chmod 644 idp.crt
```

请注意，如果您已从旧版本升级到 Zabbix 5.0，则还需要手动将这些行添加到 zabbix.conf.php 文件 (位于 //ui/conf/ // 目录):

```
// 用于SAML身份验证
$SSO['SP_KEY'] = 'conf/certs/sp.key'; // 您的私钥路径
$SSO['SP_CERT'] = 'conf/certs/sp.crt'; // 您的公钥路径
$SSO['IDP_CERT'] = 'conf/certs/idp.crt'; // idp公钥路径
$SSO['SETTINGS'] = []; // 其它设置
```

有关更多详细信息，请参见通用的SAML 身份验证 说明。

3. 如果在 Okta 中将 *Assertion Encryption* 设置为 Encrypted, 则还应在 Zabbix 中标记 *Encrypt* 参数的复选框”Assertions”。



4. 按”Update” 按钮保存这些设置。

Note:
要使用 SAML 登录, Zabbix 中的用户别名应与他的 Okta 电子邮件匹配。可以在 Zabbix Web 界面的 *Administration* → *Users* 部分中更改此设置。

3 后台进程配置

3 Daemon configuration

1 Zabbix server

概述

本节列出了 Zabbix 服务器配置文件 (zabbix_server.conf) 中支持的参数. 请注意:

- 默认值反映守护程序的默认值, 不是附带的配置文件中的值;
- Zabbix 只支持 UTF-8 编码的配置文件，不支持 BOM;
- 以 “#” 开头的注释只支持在行首.

以下参数可以在 Zabbix server 配置文件中配置：

参数名称必须配	范围	默认值	描述信息
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	默认值	描述信息
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

参数名称必须配	范围	默认值	描述信息	
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。

参数名称必须配	范围	默认值	描述信息	
CacheUpdateFrequency	否	-3600	0	abbix 配置缓存更新频率,单位为秒.另外参考runtime con-trol 选项。

参数名称必须配	范围	默认值	描述信息	
DBHost	否		ocalhost	据库主机名。如果是MySQL lo-cal-host或空字符串会导致使用套接字。如果是Post-greSQL只有空字符串会使用套接字。
DBName	是			据库名称。

参数名称必须配	范围	默认值	描述信息	
DBPassword	否			据库登录密码。如果数据库没有密码，请注释掉此参数。
DBPort	否	024-65535		使用本地套接字时的数据库端口。
DBSchema	否			据库Schema名字。仅IBM DB2和Post-greSQL使用。
DBSocket	否			ySQL套接字文件的路径。

参数名称	必须	配	范围	默认值	描述信息
DBUser			否		据库用户名。

参数名称必须配	范围	默认值	描述信息
DBTLSConnect	否		置此选项将强制使用TLS连接到数据库: *required* - 使用TLS连接 *verify_ca* - 使用校验证书的TLS连接 *verify_full* - 使用校验证书的TLS连接，并验证DB-Host指定的数据库标识是否与其证

参数名称必须配	范围	默认值	描述信息	
DBTLSCAFile	否 (是, 如果 DBTLSCon-nect 设置为如下选项时: verify_ca, ver-ify_full)		< 从 Zabbix 5.	含用于数据库证书验证的顶级CA证书文件的完整路径名。.0 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
DBTLSCertFile	否		于对数据库进行身份验证的Zabbix服务器证书文件的完整路径名。从Zabbix 5.0.0开始支持此参数。

参数名称必须配	范围	默认值	描述信息
DBTLSKeyFile	否		于对数据库进行身份验证的私钥文件的完整路径名。从 Zab-bix 5.0.0 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
DBTLSCipher	否		abbix 服务器允许 TLS 协议通过 TLSv1.2 的加密密码列表。仅支持 MySQL. 从 Zab-bix 5.0.0 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
DBTLSCipher13	否		abbix服务器允许TLSv1.3协议使用的加密密码套件列表。仅支持MySQL 8.0.16以上版本。从Zab-bix 5.0.0开始支持此参数。

参数名称必须配	范围	默认值	描述信息
DebugLevel	否	-5	定调试等级:0 - Zab-bix进程的起停基本信息1 - 严重(Crit-i-cal)信息2 - 错误(Er-ror)信息3 - 警告(Warn-ing)信息4 - 调试(De-bug)信息(产生大量信息)5 - 扩展调试(产生

参数名称必须配	范围	默认值	描述信息
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。

参数名称必须配	范围	默认值	描述信息	
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，用于轮转生成导出的文件。此参数从 Zabbix 4.0.0 开始支持。

参数名称必须配	范围	默认值	描述信息
ExportType	no		List of comma-delimited entity types (events, history, trends) for real-time export (all types by default). Valid only if ExportDir is set. *Note* that if ExportType is specified, but ExportDir is not, then this is a configuration error and the server will not start. e.g.: ExportType=his-

参数名称必须配	范围	默认值	描述信息
ExternalScripts	否		usr/local/share/zabbix/externalscripts 部脚本位置(依赖编译安装时的环境变量 *datadir*)。

参数名称必须配	范围	默认值	描述信息	
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fping程序能够处理IPv6地址，就置空(”Fping6Locatio参数。

参数名称必须配	范围	默认值	描述信息	
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。
HistoryCacheSize	否	28K-2G	6M	史缓存数据大小,单位为字节。

参数名称必须配	范围	默认值	描述信息	
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Zab-bix 3.0.0开始支持。

参数名称必须配	范围	默认值	描述信息
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
HistoryStorageURL	否		史数据存储 HTTP[S] URL，用于把历史数据存储到 ElasticSearch。这个参数参考Elasticsearch进行配置。

参数名称必须配	范围	默认值	描述信息	
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Elas-tic-search。这个参数参考Elasticsearch进行配置。

参数名称必须配	范围	默认值	描述信息
HousekeepingFrequency	否	-24	abbix 执行 house-keep-ing 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 house-keeper 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

参数名称必须配	范围	默认值	描述信息
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/pa Zab- bix 2.4.0 以

参数名称必须配	范围	默认值	描述信息
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.

参数名称必须配	范围	默认值	描述信息	
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。

参数名称必须配	范围	默认值	描述信息	
ListenIP	否		.0.0.0	rapper 监听的 Ip 地址，使用逗号进行分割。如果没有设置该参数，会监听所有网络接口。从 Zab-bix 1.8.3 开始支持多 Ip 地址。
ListenPort	否	024-32767	0051	rapper 监听端口。

参数名称必须配	范围	默认值	描述信息
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.s 这些模块必须在LoadModulePath参数指定的路径中。允许多个LoadModule参数。

参数名称必须配	范围	默认值	描述信息	
LoadModulePath	否			erver模块的绝对路径。默认值在编译时指定。
LogFile	是, 如果LogType设置为*file*, 否则为否		日志文件名称。	

参数名称必须配	范围	默认值	描述信息
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。

参数名称必须配	范围	默认值	描述信息	
LogType	否		ile	志输出类型: *file* - 写入 Log-File 参数指定的日志文件中, *system* - 写入 sys-log, *console* - 控制台输出. 从 Zab-bix 3.0.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息
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 Zab-bix 1.8.2 开始支持该参数

参数名称必须配	范围	默认值	描述信息	
MaxHousekeeperDelete	否	-1000000	000	个 house-keep-ing 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 Zab-bix 1.8.2 开始支持该

参数名称必须配	范围	默认值	描述信息
PidFile	否		tmp/zabbix_server.pid ID 文件名称。
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。

参数名称必须配	范围	默认值	描述信息
ProxyDataFrequency	否	-3600	abbix server 多少秒向 Zab-bix proxy 请求一次历史数据，用于被动模式的 proxy。\\从 Zab-bix 1.8.3 开始支持该参数。

参数名称必须配	范围	默认值	描述信息
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 zab-bix_trap_receiv 或 SN-MPTT 配置文件中的配置保持一致。从 Zab-bix 2.0.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
SocketDir	否		tmp	abbix内部服务使用的，用于存储IPC sock-ets的目录。从Zab-bix 3.4.0开始支持该参数。
SourceIP	否			外连接的源IP地址。
SSHKeyLocation	no			SSH检查和操作的公钥和私钥的位置。

参数名称必须配	范围	默认值	描述信息
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Zab-bix 2.4开始支持该参数。

参数名称必须配	范围	默认值	描述信息
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Zab-bix 2.4开始支持该参数。

参数名称必须配	范围	默认值	描述信息
SSLCALocation	否		盖为SSL服务器证书验证，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CURLOPT_CAPATH，在7.42.0之前的libcurl版本中，

参数名称必须配	范围	默认值	描述信息
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 Zab-bix 1.8.3 开始得到了支持。
StartAlerters	否	-100	警进程的初始实例数量。从 Zab-bix 3.4.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StartDiscoverers	否	-250	现进程的初始实例数量。在 Zab-bix 1.8.5 版本之前，最大能设置为 255。
StartEscalators	否	-100	scalators 进程的初始实例数量。从 Zab-bix 3.0.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StartHTTPPollers	否	-1000	TTP 轮询进程的初始实例数量[1]. 在 Zabbix 1.8.5 版本之前，最大能设置为 255。
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。

参数名称必须配	范围	默认值	描述信息	
StartJavaPollers	否	-1000		ava轮询子进程的初始实例数量。[1]. 从Zab-bix 2.0.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StartLLDProcessors	no	1-100	2 Number of pre-forked instances of low-level discovery (LLD) workers[1]. The LLD manager process is automatically started when an LLD worker is started. This parameter is supported since Zabbix 4.2.0.

参数名称必须配	范围	默认值	描述信息
StartPingers	否	-1000	CMP pingers进程的初始实例数量[1]. 在Zabbix 1.8.5版本之前，最大能设置为255。

参数名称必须配	范围	默认值	描述信息
StartPollersUnreachable	否	-1000	可达主机(包括IPMI和Java)的轮询进程的初始实例数量。[1]. 从Zab-bix 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Zab-bix

参数名称必须配	范围	默认值	描述信息
StartPollers	否	-1000	询进程的初始实例数量。[1].\\注意如果要内部，聚合，计算的监控项能正常工作，这个参数值必须非0。

参数名称必须配	范围	默认值	描述信息
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。[1]. 从 Zabbix 3.4.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StartProxyPollers	否	-250	动proxy的轮询进程初始实例数量。[1].\\在Zab-bix 1.8.5版本之前，最大能设置为255。从Zab-bix 1.8.3开始支持该参数。

参数名称必须配	范围	默认值	描述信息
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。

参数名称必须配	范围	默认值	描述信息
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 Zab-bix 2.2.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StartTrappers	否	-1000	rapper进程的初始实例数量。[1]. Trapper接收来自Zab-bix发送者、主动agent和主动prox-ies的数据。至少要运行一个trap-per进程用于在web前端展示服务器可用性和队列视图。在Zab-bix

参数名称必须配	范围	默认值	描述信息	
StartVMwareCollectors	否	-250		mware采集器进程的初始实例数量。\\从Zab-bix 2.2.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StatsAllowedIP	否		号分隔的IP地址列表,可选CIDR表示法,或外部Zab-bix实例的DNS名称.只接受来自此处列出的地址的Stats请求。如果未设置此参数，则不接受stats请求。如果启用IPv6支持，则'127.0.0.1',

参数名称必须配	范围	默认值	描述信息	
Timeout	否	-30		gent, SNMP 设备或外部检查的超时时长(单位为秒)。

参数名称必须配	范围	默认值	描述信息
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Zab-bix组件之间的加密通信。从Zab-bix 3.0.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息
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。

参数名称必须配	范围	默认值	描述信息
TLSCipherAll	否		nuTLS 优先级字符串或 OpenSSL (TLS 1.2) 密码字符串。覆盖基于证书和 PSK 的加密的默认密码套件选择标准。例如: TLS_AES_256_C 从 Zab-bix 4.4.7 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
TLSCipherAll13	否		LS 1.3 中 OpenSSL 1.1.1 或更新版本的密码字符串。覆盖基于证书和 PSK 的加密的默认密码套件选择标准。GnuTLS 示例: NONE:+VERS-TLS1.2:+ECDHE RSA:+RSA:+EC PSK:+PSK:+AES 128-GCM:+AES-128-CBC:+AEAD:+S ALL:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 OpenSSL 示例: EECDH+aRSA+ 从 Zab-bix 4.4.7 开始支持

参数名称必须配	范围	默认值	描述信息
TLSCipherCert	否		nuTLS 优先级字符串或 OpenSSL (TLS 1.2) 密码字符串。覆盖基于证书加密的默认密码套件选择条件。GnuTLS 示例: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SIGN-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509 OpenSSL 示例: EECDH+aRSA+Zab-bix 4.4.7 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
TLSCipherCert13	否		LS 1.3 中 OpenSSL 1.1.1 或更新版本的密码字符串。覆盖基于证书的加密的默认密码套件选择标准。从 Zab-bix 4.4.7 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
TLSCipherPSK	否		nuTLS 优先级字符串或 OpenSSL (TLS 1.2) 密码字符串。覆盖基于 PSK 的加密的默认密码套件选择标准。GnuTLS 示例: NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA384:+ALL:+COMP-NULL:+SIGN-ALL OpenSSL 示例: kECDHEPSK+AES128 从 Zabbix 4.4.7 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
TLSCipherPSK13	否		LS 1.3 中 OpenSSL 1.1.1 或更新版本的密码字符串。覆盖基于 PSK 的加密的默认密码套件选择标准。示例: TLS_CHACHA20 从 Zab-bix 4.4.7 开始支持此参数。

参数名称必须配	范围	默认值	描述信息
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。

参数名称必须配	范围	默认值	描述信息	
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。
TmpDir	否		tmp	时目录。
TrapperTimeout	否	-300	00	义trapper处理数据的超时时间。

参数名称必须配	范围	默认值	描述信息	
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。
UnavailableDelay	否	-3600	0	资源不可用期间，Zab-bix多少秒检查一次资源是否可用。

参数名称必须配	范围	默认值	描述信息	
UnreachableDelay	否	-3600	5	资源不可达期间，Zab-bix多少秒检查一次资源是否可达。
UnreachablePeriod	否	-3600	5	主机不可用多少秒后，即视为主机不可用。

参数名称必须配	范围	默认值	描述信息	
User	否		abbix	低系统某普通用户的权限。仅当以’root’身份运行且 Al-low-Root 参数设置为禁止时，该参数才起作用。从 Zab-bix 2.4.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
ValueCacheSize	否	,128K-64G	M	史数据缓存大小,单位为字节。缓存item历史数据请求的共享内存大小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

参数名称必须配	范围	默认值	描述信息	
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

参数名称必须配	范围	默认值	描述信息	
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Zab-bix 2.2.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息
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。

参数名称必须配	范围	默认值	描述信息	
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。

注脚

[1] 请注意，太多的数据采集进程 (pollers, unreachable pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers) 与 IPMI manager , SNMP trapper 和预处理工作进程 (preprocessing workers) 一起会耗尽预处理管理器的每进程文件描述符限制。

<note warning> 这将导致 Zabbix 服务器停止（通常在启动后不久，但有时可能需要更多时间），应修改配置文件或提高限制以避免这种情况。:::

[2] 当大量监控项被删除时，会增加数据库的负载, 因为 `housekeeper` 需要删除这些监控项的所有历史数据。例如, 如果我们只需要删除一个监控项原型, 但是这个原型链接到 50 个主机，每个主机的原型扩展到 100 个真实的监控项，总共需要删除 5000 个监控项 (1*50*100)。如果 MaxHousekeeperDelete 设置了 500（MaxHousekeeperDelete=500），则 `housekeeper` 进程必须在一个周期内从 history 和 trends 表中删除多达 2500000 个值（5000*500）。

Footnotes

[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.

Warning:
This will cause Zabbix server to stop (usually shortly after the start, but sometimes it can take more time). The configuration file should be revised or the limit should be raised to avoid this situation.

[2] When a lot of items are deleted it increases the load to the database, because the housekeeper will need to remove all the history data that these items had. For example, if we only have to remove 1 item prototype, but this prototype is linked to 50 hosts

and for every host the prototype is expanded to 100 real items, 5000 items in total have to be removed (1*50*100). If 500 is set for MaxHousekeeperDelete (MaxHousekeeperDelete=500), the housekeeper process will have to remove up to 2500000 values (5000*500) for the deleted items from history and trends tables in one cycle.

2 Zabbix proxy

概述

本节列出了 Zabbix proxy 配置文件（zabbix_proxy.conf）中支持的参数, 请注意:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- 默认值反映了守护程序启动的默认值，而不是配置文件中的值;
- Zabbix 只支持不带BOM的 UTF-8 编码的配置文件
- 在配置文件中行首使用”#” 可以注释此行配置。

参数

参数名称必须配	范围	默认值	描述信息	
AllowRoot	no		0	允许服务以’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 -

参数名称必须配	范围	默认值	描述信息	
CacheSize	no	128K-64G	8M	缓存大小,单位为字节。用于存储主机、监控项数据的共享内存大小。在 Zabbix 5.0.1 版本这参数的最大值可以从 8G 增加到 64G。

参数名称必须配	范围	默认值	描述信息	
ConfigFrequency	no	1-604800	3600	每隔多少秒proxy从Zab-bix 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Prox-y-Mode决定。

参数名称必须配	范围	默认值	描述信息	
DataSenderFrequency	no	1-3600	1	Proxy 将采集到的数据以一定的时间间隔(单位为秒)发送给 Zab-bix server。该参数只有主动 proxy 才会使用，proxy 工作模式由参数 Prox-y-Mode 决定。

参数名称必须配	范围	默认值	描述信息	
DBHost	no		localhost	数据库主机名。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。

参数名称必须配	范围	默认值	描述信息
DBName	yes		数据库名称。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

参数名称必须配	范围	默认值	描述信息	
DBPassword	no			数据库登录密码。此参数SQLite不使用。如果数据库没有密码，请注释掉此参数。
DBSchema	no			数据库Schema名字。仅IBM DB2和Post-greSQL使用。

参数名称必须配	范围	默认值	描述信息	
DBSocket	no		3306	MySQL 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。
DBUser				数据库用户名。此参数 SQLite 不使用。

参数名称必须配	范围	默认值	描述信息
DBTLSConnect	no		设置此选项将强制使用到 TLS 连接数据库。 *required* - 使用 TLS 连接 *verify_ca* - 使用 TLS 并且验证 *verify_full* - 使用 TLS 连接加证书，并且验证 DB-Host 指定的数据库标识是否与其证书匹配。

参数名称必须配	范围	默认值	描述信息	
DBTLSCAFile	no (yes, 如果数据库连接为其中一个: verify_ca, verify_full)		< 从 Zabbix 5.0	包含用于数据库证书验证的顶级CA证书的文件的完整路径名。0开始支持此参数。

参数名称必须配	范围	默认值	描述信息	
DBTLSCertFile	no			数据库进行身份验证时对 Zabbix 服务器证书文件进行完整路径名验证。从 Zabbix 5.0.0 开始支持此参数。

参数名称必须配	范围	默认值	描述信息	
DBTLSKeyFile	no			数据库进行身份验证时对私钥文件进行完整路径名验证。从 Zabbix 5.0.0 开始支持此参数。

参数名称必须配	范围	默认值	描述信息	
DBTLSCipher	no			Zabbix服务器允许TLS协议通过TLSv1.2的加密密码列表。只支持MySQL.从Zab-bix 5.0.0开始支持此参数。

参数名称必须配	范围	默认值	描述信息	
DBTLSCipher13	no			Zabbix服务器允许TLS协议通过TLSv1.3的加密密码列表。只支持MySQL8.0.16之后的版本。从Zab-bix5.0.0开始支持此参数。

参数名称必须配	范围	默认值	描述信息	
DebugLevel	no	0-5	3	指定调试等级: 0 - Zabbix进程的起停基本信息 1 - 重要(Critical)信息 2 - 错误(Error)信息 3 - 警告(Warning)信息 4 - 调试(Debug)信息(产生大量信息) 5 - 扩展调试(产

参数名称必须配	范围	默认值	描述信息
EnableRemoteCommands	no	0	是否允许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。
ExternalScripts	no	/usr/local/share/zabbix/externalscripts	外部脚本位置(依赖编译安装时的环境变量*datadir*)。

参数名称必须配	范围	默认值	描述信息	
Fping6Location	no		/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。

参数名称必须配	范围	默认值	描述信息	
FpingLocation	no		/usr/sbin/fping	fping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

参数名称必须配	范围	默认值	描述信息	
HeartbeatFrequency	no	0-3600	60	心跳信息发送频率，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Prox-y-Mode决定。

参数名称必须配	范围	默认值	描述信息	
HistoryCacheSize	no	128K-2G	16M	历史缓存数据大小,单位为字节。存储历史数据使用共享内存.

参数名称必须配	范围	默认值	描述信息	
HistoryIndexCacheSize	no	128K-2G	4M	历史索引缓存大小,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Zab-bix 3.0.0开始支持。

参数名称必须配	范围	默认值	描述信息	
Hostname	no		由参数 HostnameItem 设置唯一的大	写敏感的 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: 128

参数名称必须配	范围	默认值	描述信息	
HostnameItem	no		system.hostname	当参数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[]。如果Hostname设置了，该参数可以忽略。从Zabbix 1.8.6开

参数名称必须配	范围	默认值	描述信息	
HousekeepingFrequency	no	0-24	1	Zabbix 执行 house-keep-ing 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 house-keeper 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

参数名称必须配	范围	默认值	描述信息	
Include	no			可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/Zab-bix 2.4.0

参数名称必须配	范围	默认值	描述信息	
JavaGateway	no			Zabbix 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.

参数名称必须配	范围	默认值	描述信息	
JavaGatewayPort	no	1024-32767	10052	Zabbix Java 网关监听端口。Zabbix 2.0.0 以后的所有版本都支持该参数。

参数名称必须配	范围	默认值	描述信息	
ListenIP	no		0.0.0.0	trapper 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 Zab-bix 1.8.3 开始支持多 Ip 地址。
ListenPort	no	1024-32767	10051	trapper 监听端口。

参数名称必须配	范围	默认值	描述信息	
LoadModule	no			server端启动时加载的模块，这些模块用来扩展server的功能。格式：LoadModule=<mod LoadModule=<path LoadModule=</abs 这些模块必须位于Load-Mod-ulePath指定的目录中，或者路径必须位于模块名称之前。如果前面的路径

参数名称必须配	范围	默认值	描述信息	
LoadModulePath	no			proxy模块本地完整路径.默认值在编译时指定。
LogFile	如果日志类型选择 yes, 其他选择 no		日志文件名字	

参数名称必须配	范围	默认值	描述信息	
LogFileSize	no	0-1024	1	日志文件大小，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。

参数名称必须配	范围	默认值	描述信息	
LogRemoteCommands	no		0	当执行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。

参数名称必须配	范围	默认值	描述信息	
LogType	no		file	日志输出类型: *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。

参数名称必须配	范围	默认值	描述信息	
LogSlowQueries	no	0-3600000	0	数据库查询消耗时间，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。
PidFile	no		/tmp/zabbix_proxy.pid	PID 文件名字。

参数名称必须配	范围	默认值	描述信息	
ProxyLocalBuffer	no	0-720	0	Proxy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。

参数名称	必须	配	范围	默认值	描述信息
ProxyMode	no		0-1	0	Proxy 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

参数名称必须配	范围	默认值	描述信息	
ProxyOfflineBuffer	no	1-720	1	如果无法连接Zab-bix server，proxy将保留数据N小时。\\旧数据将丢失。

参数名称必须配	范围	默认值	描述信息	
ServerPort	no	1024-32767	10051	当 Prox- y- Mode 参 数 设 置 为 主 动 模 式:\\可 以 通 过 Zab- bix server 的 IP 地 址 或 DNS 名 称 获 取 配 置 数 据 并 将 数 据 发 送 给 Zab- bix server。 当 Prox- y- Mode 参 数 设 置 为 被 动 模 式:\\逗 号 分 隔 的 IP 地 址 列

参数名称必须配	范围	默认值	描述信息
SNMPTrapperFile	no	/tmp/zabbix_traps.tmp	临时文件，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。

参数名称必须配	范围	默认值	描述信息	
SocketDir	no		/tmp	Zabbix内部服务使用的用于存储IPC sock-ets的目录。从Zab-bix 3.4.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
SourceIP	no			对外连接的源IP地址。如: - 连接到Zab-bix server; - 无代理连接(VMware, SSH, JMX, SNMP, Tel-net and sim-ple checks); - HTTP agent 连接; - 预处理 JavaScript HTTP 请求
SSHKeyLocation	no			SSH检查和操作的公钥和私钥的位置。

参数名称必须配	范围	默认值	描述信息	
SSLCertLocation	no			用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Zab-bix 2.4开始支持该参数

参数名称必须配	范围	默认值	描述信息	
SSLKeyLocation	no			用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。

参数名称必须配	范围	默认值	描述信息
SSLCALocation	no		为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curlopt-capath，在7.42.0之前的

参数名称必须配	范围	默认值	描述信息	
StartDBSyncers	no	1-100	4	数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Zab-bix 1.8.3开始得到了支持。

参数名称必须配	范围	默认值	描述信息	
StartDiscoverers	no	0-250	1	发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。
StartHTTPPollers	no	0-1000	1	HTTP 轮询进程的初始实例数量。

参数名称必须配	范围	默认值	描述信息	
StartIPMIPollers	no	0-1000	0	IPMI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。
StartJavaPollers	no	0-1000	0	Java轮询子进程的初始实例数量。从Zabbix 2.0.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
StartPingers	no	0-1000	1	ICMP 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。

参数名称必须配	范围	默认值	描述信息	
StartPollersUnreachable	no	0-1000	1	不可达主机(包括IPMI和Java)的轮询进程的初始实例数量。从Zab-bix 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Zab-bix

参数名称必须配	范围	默认值	描述信息	
StartPollers	no	0-1000	5	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。

参数名称必须配	范围	默认值	描述信息	
StartPreprocessors	no	1-1000	3	pre-forked 实例的预处理线程数量[1]. 预处理器工作进程启动时，预处理管理器进程将自动启动。从 Zab-bix 4.2.0 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
StartSNMPTrapper	no	0-1	0	设置为1，SNMP trapper进程将启动。从Zabbix 2.0.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
StartTrappers	no	0-1000	5	trapper进程的初始实例数量。Trapper接收来自Zab-bix发送者、主动agent的数据。至少要运行一个trap-per进程用于在web前端展示服务器可用性和队列视图。在Zab-bix 1.8.5版本之前，最

参数名称必须配	范围	默认值	描述信息	
StartVMwareCollectors	no	0-250	0	vmware采集器进程的初始实例数量。\\从Zab-bix 2.2.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息
StatsAllowedIP	no		ip 地址列表以”，”分割，也可以使用 CIDR、或者 DNS. 只接受 ip 里表中的请求. 如果未设置此参数，则不接受请求. 如果启用 ipv6, 则’127.0.0.1’, ’::127.0.0.1’, ’::ffff:127.0.0.1’ ipv4 和 ipv6 都支持，’::/0’ 支持 IPv4 或者 IPv6 地址.

参数名称必须配	范围	默认值	描述信息	
Timeout	no	1-30	3	agent、SNMP设备或外部检查的超时时长(单位为秒)。

参数名称必须配	范围	默认值	描述信息	
TLSAccept	如果 TLS certifi-cate 或 PSK 参数都进行了定义 (即使是未加密的连接) 选择 yes, 否则为否		Zabbix server 能接受哪些连接方式。此	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

参数名称必须配	范围	默认值	描述信息	
TLSCAFile	no			包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。

参数名称必须配	范围	默认值	描述信息	
TLSCertFile	no			包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。

参数名称必须配	范围	默认值	描述信息	
TLSCipherAll	no			GnuTLS 或者 OpenSSL (TLS1.2) 加密.默认的密码套件为 PSK-based. 如 GnuTLS 例如: TLS_AES_256_GCM_ 从 Zabbix 4.4.7 开始支持该参数。从 Zabbix

参数名称必须配	范围	默认值	描述信息	
TLSCipherCert	no			GnuTLS 加密或者 OpenSSL (TLS 1.2) 加密. 基于证书的加密的默认 certificate-based. 例如 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA2 ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509 例如 OpenSSL: EECDH+aRSA+AES1 从 Zab-bix 4.4.7 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
TLSCipherCert13	no			OpenSSL 1.1.1 或者 TLS 1.3，基于证书的加密的默认 certificate-based. 从 Zab-bix 4.4.7 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
TLSCipherPSK	no			GnuTLS 或者 OpenSSL (TLS 1.2) 加密。默认的密码套件为 PSK-based. 例如 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA2 ALL:+COMP-NULL:+SIGN-ALL 例如 OpenSSL: kECD-HEPSK+AES128:kPS 从 Zab-bix 4.4.7 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
TLSCipherPSK13	no			OpenSSL 1.1.1 或者 TLS 1.3. 默认的密码套件为 PSK-based.. 例如: TLS_CHACHA20_POL 从 Zab-bix 4.4.7 开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
TLSConnect	如果 TLS certifi-cate 或 PSK 参数进行了定义 (即使是未加密的连接) 选是, 否则为否		该参数仅用于主动 proxy 连接 Zabbix se	ver。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 Zab-bix 3.0.0 开始支持该参

参数名称必须配	范围	默认值	描述信息	
TLSKeyFile	no			包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。

参数名称必须配	范围	默认值	描述信息	
TLSPSKFile	no			包含proxy与Zab-bix server加密通信所使用的预共享密钥文件的完整路径名。从Zab-bix 3.0.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
TLSPSKIdentity	no			预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。
TLSServerCertIssuer	no			允许server证书颁发者。从Zabbix 3.0.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
TLSServerCertSubject	no			允许的server证书主题。从Zab-bix 3.0.0开始支持该参数。
TmpDir	no		/tmp	临时目录
TrapperTimeout	no	1-300	300	定义trap-per处理数据的超时时间。

参数名称必须配	范围	默认值	描述信息	
User	no		zabbix	为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。

参数名称必须配	范围	默认值	描述信息	
UnavailableDelay	no	1-3600	60	在资源不可达期间，Zabbix多少秒检查一次资源是否可用。
UnreachableDelay	no	1-3600	15	在资源不可期间，Zabbix多少秒检查一次资源是否可达.

参数名称必须配	范围	默认值	描述信息	
UnreachablePeriod	no	1-3600	45	在资源不可期间，Zabbix多少秒检查一次资源是否可达。

参数名称必须配	范围	默认值	描述信息	
VMwareCacheSize	no	256K-2G	8M	存储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。

参数名称必须配	范围	默认值	描述信息	
VMwareFrequency	no	10-86400	60	间隔多少秒从单个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Zab-bix 2.2.0开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
VMwarePerfFrequency	no	10-86400	60	间隔多少秒从单个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Zab-bix 2.2.9, 2.4.4开始支持该参数。

参数名称必须配	范围	默认值	描述信息	
VMwareTimeout	no	1-300	10	vmware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。

3 Zabbix agent (UNIX)

概述

本节列出了 Zabbix agent 配置文件（zabbix_agentd.conf）中支持的参数。注意:

- 默认值反映的是进程默认值，而不是出厂配置文件中的值;
- Zabbix 仅支持 UTF-8 编码的配置文件，而没有 BOM;
- 仅在行首支持以“#”开头的注释。

参数

参数必	项范围	默认值	描述
Alias	否		置监控项key的别名。它可以用一个更小更简单代替长而复杂的监控项key。可能存在多个*Alias*参数。多个参数允许使用相同的*Alias*。不同*Alias*可能引用相同的监控项key。

参数必	项范围	默认值	描述
AllowKey	否		许执行与模式匹配的那些监控项key。值模式是通配符表达式，支持“*”字符以匹配任意数量的任何字符。可以结合DenyKey定义多个值的匹配规则。根据其出现顺序

参数必	项范围	默认值	描述
AllowRoot	否		许代理以“root”身份运行。如果禁用并且代理由“root”启动，则代理将尝试切换到用户“zab-bix”。如果在普通用户下启动，则无效。 0 - 不允许 1 - 允许

参数必	项范围	默认值	描述
BufferSend	否	-3600	冲区中的数据不要保留超过N秒。
BufferSize	否	-65535	00 存缓冲区中的最大值数。如果缓冲区已满，则代理会将所有收集的数据发送到Zab-bix server或代理。

参数必	项范围	默认值	描述
DebugLevel	否	-5	定调试级别: 0 - 有关启动和停止 Zab-bix 进程的基本信息 1 - 关键信息 2 - 错误信息 3 - 警告 4 - 用于调试(产生大量信息) 5 - 扩展调试(产生更多信息)

参数必	项范围	默认值	描述
DenyKey	否		绝执行与模式匹配的那些监控项key。值模式是通配符表达式，支持“*”字符以匹配任意数量的任何字符。可以结合AllowKey定义多个值的匹配规则。根据其出现顺

参数必	项范围	默认值	描述
EnableRemoteCommands	否		否允许来自 Zabbix server 的远程命令。从 Zabbix 5.0.2 开始，此参数已弃用，请使用 AllowKey=system 或 DenyKey=syste 来替代它是 AllowKey/DenyKe 参数的内部别名，具体取决于值: 0 - DenyKey=syste 1 - AllowKey=system

参数必	项范围	默认值	描述
HostInterface	否	-255 个字符	定义主接口的可选参数。主机接口用于主机自动注册过程。如果值超过255个字符的限制，代理将发出错误并且不会启动。如果未定义，将从HostInterface获取值。自Zabbix 4.4.0起支

参数必	项范围	默认值	描述
HostInterfaceItem	否		选参数，用于定义用于获取主机接口的监控项。主机接口用于主机自动注册过程。在自动注册请求期间，如果指定项返回的值超过255个字符的限制，则代理将记

参数必	项范围	默认值	描述
HostMetadata	否	-255 个字符	定义主元数据的可选参数。主机元数据仅在主机自动注册过程(主动代理)中使用。如果未定义，将从 HostMetadataItem 获取该值。如果指定的值超出限制或非 UTF-8 字符串，则代理将

参数必	项范围	默认值	描述
HostMetadataItem	否		选参数，用于定义用于获取主机元数据的 *Zabbix agent* 监控项。仅当未定义 HostMetadata 时才使用此选项。支持 UserParameters 和 aliases。不管 allowKey/DenyKe 值如何，都支持 *system.run[]* HostMetadataIt 值在每次自

参数必	项范围	默认值	描述
Hostname	否		Hostname设置唯一的区分大小写的主机名。主动检查所必需，并且必须与服务器上配置的主机名匹配。允许的字符：字母数字，'.'，' '，','，'_'and'-'。最大长度：128。

参数必	项范围	默认值	描述
HostnameItem	否	ystem.hostname	选参数，用于定义用于获取主机名的 *Zabbix agent* 监控项。仅当未定义主机名时才使用此选项。不支持 UserParameters 或 aliases，但不管 allowKey/DenyKey 值如何，都支持 *system.run[]* 1.8.6 及更高版

参数必	项范围	默认值	描述
Include	否		可以在配置文件的目录中包括单个文件或所有文件。要仅在指定目录中包含相关文件，模式匹配支持使用星号通配符。例如:/absolute/从Zab-bix 2.4.0开始支持模式匹配。请参

参数必	项范围	默认值	描述
ListenIP	否	.0.0.0	理应侦听使用逗号分隔的IP地址列表。1.8.3及更高版本支持多个IP地址。
ListenPort	否	024-32767	0050	理将在此端口上侦听来自服务器的连接。

参数必	项范围	默认值	描述
LoadModule	否		代理启动时加载的模块。模块用于扩展代理的功能。格式: LoadModule=< LoadModule=< LoadModule=< 模块必须位于 Load-Mod-ulePath 指定的目录中，或者路径必须在模块名称之前。如果前面的路径是绝对路

参数必	项范围	默认值	描述
LoadModulePath	否		理模块位置的完整路径。默认值取决于编译选项。
LogFile	是，如果日志类型设置为 *file*, 否则为否		日志文件名。

参数必	项范围	默认值	描述
LogFileSize	否	-1024	志文件的最大大小，以MB为单位。 0 - 禁用自动日志轮换。 注意:如果达到了日志文件大小限制并且文件轮换失败，则无论出于何种原因，现有的日志文件都会被

参数必	项范围	默认值	描述
LogType	否	ile	志输出类型：*file* - 将日志写入 LogFile 参数指定的文件，*system* - 将日志写入 syslog，*console* - 将日志写入标准输出。从 Zabbix 3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
LogRemoteCommands	否		否启用执行Shell命令记录为警告。 0 - 禁用 1 - 启用仅当远程执行命令时，才会记录命令。如果通过HostMetadataItem，HostInterfaceItem或HostnameItem参数在本地启动system.run[]，则不会创建日志

参数必	项范围	默认值	描述
MaxLinesPerSecond	否	-1000	0 处理“log”和“event-log”主动检查时，代理每秒发送给Zab-bix server或代理的最大新行数。所提供的值将被“log”或“event-log”监控项key中提供的参数“max-lines”所覆盖。注意:Zabbix将处理比*Max-Lines-Per-*

参数必	项范围	默认值	描述
PidFile	否		tmp/zabbix_agentd.pid PID文件的名称。
RefreshActiveChecks	否	0-3600	20 动检查刷新列表的频率(以秒为单位)。请注意，主动检查刷新失败后，将在60秒后尝试进行下一次刷新。

参数必	项范围	默认值	描述
Server	是, 如果未将 StartAgents 显式设置为 0		逗号分隔的 IP 地址表，可以选择使用 CIDR 表示法，或者 Zabbix servers 和 Zabbix proxies 的主机名。仅从此处列出的主机接受传入的连接。如果启用了 IPv6 支持，则将’127.0.0.1’, ’::ffff:127.0.0.1’ 同等对待，并且’::/0’ 将允许任何 IPv4 或

参数必	项范围	默认值	描述
ServerActive	否		动检查 Zab-bix servers 和 Zab-bix prox-ies 以逗号分隔 IP：端口对(或主机名：端口对)列表。可以提供多个地址来并行使用多个独立的 Zab-bix servers，允许有空格。如果未指定端口，则使用

参数必	项范围	默认值	描述
SourceIP	否		于以下目的的源IP地址: - 与Zab-bix server或Zab-bix proxy的传出连接; - 在执行某些监控项(web.page.get, net.tcp.port, 等等)时建立连接。

参数必	项范围	默认值	描述
StartAgents	否	-100	理被动检查的zab-bix_agentd的预分支实例数。如果设置为0，则禁用被动检查，并且代理将不侦听任何TCP端口。在版本1.8.5之前，上限为16。

参数必	项范围	默认值	描述
Timeout	否	-30	处理上花费的时间不超过该值(秒)。

参数必	项范围	默认值	描述
TLSAccept	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否		接受什么传入连接。用于被动检查。可以指定多个值，逗号分隔: *unencrypted* - 接受不加密的连接 (默认) *psk* - 接受带 TLS 和预共享密钥 (PSK) *cert* - 接受带 TLS 和证书的连接 从 Zabbix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSCAFile	否		含用于对等证书验证的顶级CA证书的文件的完整路径名，用于Zab-bix组件之间的加密通信。从Zab-bix3.0.0开始支持此参数。

参数必	项范围	默认值	描述
TLSCertFile	否		含代理证书或证书链的文件的完整路径名，用于与 Zabbix 组件进行加密通信。从 Zabbix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSCipherAll	否		nuTLS 优先级字符串或 OpenSSL (TLS 1.2) 密码字符串。覆盖用于基于证书和 PSK 的加密的默认密码套件选择标准。示例: TLS_AES_256_C 从 Zab-bix 4.4.7 开始支持此参数。

参数必	项范围	默认值	描述
TLSCipherAll13	否		LS 1.3 中 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖用于基于证书和 PSK 加密的默认密码套件选择条件。 GnuTLS 示例: NONE:+VERS-TLS1.2:+ECDHE RSA:+RSA:+EC PSK:+PSK:+AES 128-GCM:+AES-128-CBC:+AEAD:+S ALL:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 OpenSSL 示例: EECDH+aRSA+ 从 Zab-bix4.4.7 开始支持

参数必	项范围	默认值	描述
TLSCipherCert	否		nuTLS 优先级字符串或 OpenSSL (TLS 1.2) 密码字符串。覆盖基于证书加密的默认密码套件选择条件。GnuTLS 示例: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA384:+ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509 OpenSSL 示例: EECDH+aRSA+AES128 从 Zabbix 4.4.7 开始支持此参数。

参数必	项范围	默认值	描述
TLSCipherCert13	否		LS 1.3 中 OpenSSL1.1.1 或更高版本的密码字符串。覆盖基于证书加密的默认密码套件选择条件。从 Zab-bix4.4.7 开始支持此参数。

参数必	项范围	默认值	描述
TLSCipherPSK	否		nuTLS 优先级字符串或 OpenSSL (TLS1.2) 密码字符串。覆盖基于 PSK 加密的默认密码套件选择条件。GnuTLS 示例: NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA384:+ECDHE-PSK:+ALL:+COMP-NULL:+SIGN-ALL OpenSSL 示例: kECDHEPSK+AES128 从 Zabbix4.4.7 开始支持此参数。

参数必	项范围	默认值	描述
TLSCipherPSK13	否		LS 1.3 中 OpenSSL1.1.1 或更高版本的密码字符串。覆盖基于 PSK 加密的默认密码套件选择条件。示例: TLS_CHACHA20 从 Zab-bix4.4.7 开始支持此参数。

参数必	项范围	默认值	描述
TLSConnect	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否		代理应如何连接到 Zab-bix server 或 pr xy。用于主动检查。只能指定一个值： *unencrypted* - 接受不加密的连接 (默认) *psk* - 接受带 TLS 和预共享密钥 (PSK) *cert* - 接受带 TLS 和证书的连接 从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSCRLFile	否		含已撤销证书文件的完整路径名。此参数用于与 Zab-bix 组件的加密通信。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSKeyFile	否		含用于与 Zab-bix 组件进行加密通信的代理私钥文件的完整路径名。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSPSKFile	否		含用于与 Zab-bix 组件进行加密通信的代理预共享密钥文件的完整路径名。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSPSKIdentity	否		共享密钥标识字符串，用于与 Zabbix server 进行加密通信。从 Zabbix3.0.0 开始支持此参数。
TLSServerCertIssuer	否		许的服务器(代理)证书颁发者。从 Zabbix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSServerCertSubject	否		许的服务器(代理)证书主题。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
UnsafeUserParameters	否	,1	许将所有字符都通过参数传递给用户定义的参数。从 Zab-bix1.8.2 开始支持。不允许使用以下字符: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @另外，不允许使用

参数必	项范围	默认值	描述
User	否	abbix	特权授予系统上特定的现有用户。仅当以“root”身份运行且AllowRoot被禁用时才有效。从Zabbix 2.4.0开始支持此参数。

参数必	项范围	默认值	描述
UserParameter	否		户定义的参数进行监控。可以有多个用户自定义参数。格式: User-Pa-ram-e-ter=<key>,<sh com-mand> 注意 shell 命令一定不能返回空字符串或仅返回 EOL。示例: User-Pa-ram-e-ter=system.tes -l

另请参阅

1. 从版本 2.0.0 开始，主动和被动检查的 Zabbix agent 配置差异

4 Zabbix agent 2 (UNIX)

概述

Zabbix agent 2 是新一代的 Zabbix agent ，可用于替代 Zabbix agent 。

本节列出了 Zabbix agent 2 配置文件（zabbix_agent2.conf）中支持的参数。注意:

- 默认值反映的是进程默认值，而不是出厂配置文件中的值;
- Zabbix 仅支持无 BOM 的 UTF-8 编码配置文件;
- 支持注释需以 “ # ” 作为行首。

参数

参数	必	项范围	默认值	描述
Alias		否		置监控项键值的别名。它可以用一个更小更简单代替长而复杂的监控项键值。可能存在多个 *Alias* 参数。多个参数允许使用相同的 *Alias*。不同 *Alias* 可能引用相同的监控

参数	必	项范围	默认值	描述
AllowKey	否			许执行与模式匹配的那些监控项键值。键值匹配模式是支持通配字符“*”用于匹配任意数量的任何字符。可以结合DenyKey定义多个键值的匹配规则。根据其出现顺

参数	必	项范围	默认值	描述
BufferSend	否		-3600	间间隔(以秒为单位)，用于确定从缓冲区向 Zab-bix server 发送值的频率。请注意，如果缓冲区已满，则数据将尽快发送。

参数	必	项范围	默认值	描述
BufferSize	否		-65535	00 存缓冲区中的最大容量。如果缓冲区已满，则代理会将所有收集的数据发送到 Zabbix server 或 proxy。仅当禁用持久缓冲区时才应使用此参数 (*EnablePersistentBuffer=0*)。

参数	必	项范围	默认值	描述
ControlSocket	否		tmp/agent.sock	控制套接字，用于发送带有“-R”选项的运行时命令。

参数	必	项范围	默认值	描述
DebugLevel	否		-5	定调试级别: 0 - 有关启动和停止 Zab-bix 进程的基本信息 1 - 关键信息 2 - 错误信息 3 - 警告 4 - 用于调试(产生大量信息) 5 - 扩展调试(产生更多信息)

参数	必	项范围	默认值	描述
DenyKey	否			绝执行与模式匹配的那些监控项键值。键值匹配模式是支持通配字符“*”用于匹配任意数量的任何字符。可以结合AI-lowKey定义多个键值的匹配规则。根据其出现

参数	必	项范围	默认值	描述
EnablePersistentBuffer	否		-1	主动监控项启用本地永久性存储。 0 - 禁用 1 - 启用 如果禁用持久性存储，则将使用内存缓冲区。

参数	必	项范围	默认值	描述
HostInterface	否	-255 个字符	定义主接口的可选参数。主机接口用于主机自动注册过程。如果值超过 255 个字符的限制，agent 将发出错误并且不会启动。如果未定义，将从 HostInterfaceItem 获取值。自 Zabbix 4.4.0 起支持。	

参数	必	项范围	默认值	描述
HostInterfaceItem	否			选参数，用于定义用于获取主机接口的监控项。主机接口用于主机自动注册过程。在自动注册请求期间，如果指定项返回的值超过255个字符的限制，则agent将记录

参数	必	项范围	默认值	描述
HostMetadata	否		-255 个字符	定义主机元数据的可选参数。主机元数据用于主机自动注册过程。如果指定的值超出限制或非 UTF-8 字符串，则 agent 将发出错误，并且不会启动。如果未定义，将从 HostMetadataItem 获取

参数	必	项范围	默认值	描述
HostMetadataItem	否			选参数，用于定义用于获取主机元数据的项目。每次自动注册尝试时都会检索主机元数据项值，以进行主机自动注册过程。在自动注册请求期间，如果指定项返

参数	必	项范围	默认值	描述
Hostname	否			Hostname设置唯一的区分大小写的主机名。主动检查所必需，并且必须与服务器上配置的主机名匹配。允许的字符：字母数字, '.', ' ', '_' and '-'。最大长度: 128。

参数	必	项范围	默认值	描述
HostnameItem	否		ystem.hostname	未定义用于生成主机名的监控项。 如果定义了主机名，则忽略。 不支持 UserParameters 或 aliases，不管 AllowKey/DenyKey 值如何，都支持 *system.run[]*。

参数	必	项范围	默认值	描述
Include		否		可以在配置文件的目录中包括单个文件或所有文件。在安装过程中，除非在编译期间进行了修改，否则 Zabbix 将在/usr/local/etc 中创建 include 目录。要仅在指定目录中包含相

参数	必	项范围	默认值	描述
ListenIP		否	.0.0.0	gent 应监听使用逗号分隔的 IP 地址列表。第一个 IP 地址被发送到 Zab-bix server，如果已连接，以检索主动检查的列表。
ListenPort		否	024-32767	0050 gent 将监听此端口上来自服务器的连接。

参数	必	项范围	默认值	描述
LogFile		是, 如果 LogType 设置为 *file*, 则否	/tmp/za	bix_agent2.log 如果 LogTy 为'file'，则记录文件名。

参数	必	项范围	默认值	描述
LogFileSize	否		-1024	志文件的最大大小，以MB为单位。0 - 禁用自动日志轮换。注意:如果达到了日志文件大小限制并且文件轮换失败，则无论出于何种原因，现有的日志文件都会被

参数	必	项范围	默认值	描述
LogType		否	ile	定将日志消息写入的位置: *system* - sys-log, *file* - Log-File 参数指定的文件, *console* - 标准输出。

参数	必	项范围	默认值	描述
PersistentBufferFile	否			abbix Agent2 应该在其中保存 SQLite 数据库的文件。必须是完整的文件名。仅当启用了持久缓冲区 (*EnablePersistentBuffer=1*) 时，才使用此参数。

参数	必	项范围	默认值	描述
PersistentBufferPeriod	否		分钟-365 天 1 小时	没有与服务或 proxy 的连接时，应该存储数据的时间段。较旧的数据将丢失。日志数据将被保留。仅当启用了持久缓冲区 (*EnablePersistentBuffer=1*) 时，才使用此参数。
PidFile	否			tmp/zabbix_agent2.pid 程 ID 文件位置。

参数	必	项范围	默认值	描述
Plugins		否		个插件可以具有一个或多个特定于插件的配置参数，格式为: Plugins.<Plugin Plugins.<Plugin

参数	必	项范围	默认值	描述
Plugins否PluginName>.KeepAlive		0-900	00	闭未使用的插件连接之前的最长时间(以秒为单位)。支持以下插件: *Ceph*, *Memcached*, *MySQL*, *Oracle*, *Redis*, *PostgreSQL*. **<PluginName>** - 插件的名称。示例: Plugins.Memc

参数	必	项范围	默认值	描述
	否	Plugins.<PluginName>.Timeout	-30	局超时请求执行时(关闭请求之前等待一个请求完成的时间)。支持以下插件: *Ceph*, *Memcached*, *MySQL*, *Redis*, *Docker*, *PostgreSQL*. **<PluginName>** - 插件的名称。

参数	必	项范围	默认值	描述
Plugins.Ceph.InsecureSkipVerify	否	alse / true	alse	定 http 客户端是否应验证服务器的证书链和主机名。如果为 *true*，则 TLS 接受服务器提供的任何证书以及该证书中的任何主机名。在这种模式下，TLS 容易受到中间人

参数	必项	范围	默认值	描述
Plugins.Ceph.Uri	否		ttps://localhost:8003	连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(仅https受支持)。可以省略端口(默认为8003)。示例:https://127.https://loca

参数	必	项范围	默认值	描述
Plugins.Docker.Endpoint	否		nix:///var/run/docker.sock	Docker 守护程序 unix 套接字位置。必须包含一个方案(仅 unix:// 支持)。

参数	必	项范围	默认值	描述
Plugins.Log.MaxLinesPerSecond	否		-1000	0 处理“日志”和“事件日志”主动检查时，agent每秒发送给Zabbix server或proxy的最大新行数。所提供的值将被“log”或“eventlog”监控项中提供的参数“maxlines”所覆盖。注意：Zabbix处理的新行

参数	必	项范围	默认值	描述
Plugins.Memcached.Uri	否		tcp://localhost:11211	连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(支持:tcp,unix)。可以省略端口(默认=11211)。示例:tcp://localh tcp://localh unix:/var/ru

参数	必	项范围	默认值	描述
Plugins.Mysql.Uri	否		tcp://localhost:3306	MySQL 连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与 URI 格式匹配。必须包含一个方案(支持: tcp, unix)。可以省略端口(默认为 3306)。示例: tcp://localh tcp://localh unix:/var/ru

参数	必	项范围	默认值	描述
Plugins.Oracle.CallTimeout	否		-30	局超时完成请求最大等待时间(以秒为单位)。
Plugins.Oracle.ConnectTimeout	否		-30	局超时建立连接最大等待时间(以秒为单位)。
Plugins.Oracle.CustomQueriesPath	否			含带有自定义查询的.sql文件的目录的完整路径名。默认禁用。示例:/etc/zabbix/

参数	必	项范围	默认值	描述
Plugins.Oracle.Service	否			E于连接的服务名称(不支持SID)。

参数	必	项范围	默认值	描述
Plugins.Oracle.Uri	否		tcp://localhost:1521	连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(支持:tcp)。可以省略端口(默认=1521)。示例:tcp://localh tcp://localh

参数	必	项范围	默认值	描述
Plugins.Postgres.Database	否		ostgres	ostgreSQL 使用的数据库名称。
Plugins.Postgres.Host	否		ocalhost	ostgreSQL 使用的主机的 IP 地址或 DNS 名称。示例: `localhost`, `192.168.1.1`
Plugins.Postgres.Port	否		432	于 Post-greSQL 的端口。

参数	必项	范围	默认值	描述
Plugins.Redis.Uri	否		tcp://localhost:6379	连接字符串。可以省略端口(默认为6379)。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(支持: tcp, unix)。示例: tcp://localh tcp://localh unix:/var/ru

参数	必	项范围	默认值	描述
Plugins.SystemRun.EnableRemoteCommands	否			否允许来自 Zabbix server 的远程命令。 0 - 不允许 1 - 允许 从 5.0.2 开始不支持此参数，请改用 AllowKey/DenyKey 参数。

参数	必	项范围	默认值	描述
Plugins.	否	SystemRun.LogRemoteCommands		用将执行的Shell命令记录为警告。 0 - 禁用 1 - 启用仅当远程执行命令时，才会记录命令。如果通过Host-Meta-dataItem，HostIn-ter-faceItem或Host-nameItem参数在本地启动sys-tem.run[]，则不会创建日

参数	必	项范围	默认值	描述
Plugins’ named ses-sions		否		果使用 Zab-bix agent 监视相同种类的多个实例，则可以为每个实例创建具有自己的一组授权参数的命名会话。命名的会话参数格式: Plugins.<Plugin Plugins.<Plugin

参数	必	项范围	默认值	描述
	否	Plugins.<PluginName>.Sessions.<SessionName>.Password		名会话密码。支持: *Memcached*, *MySQL*, *Oracle*, *PostgreSQL*, *Redis*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项 key 的会话名称。

参数	必	项范围	默认值	描述
Plugins否<PluginName>.Sessions.<SessionName>.Uri				名会话连接字符串。支持: *Ceph*, *Memcached*, *MySQL*, *Oracle*, *Redis*, *PostgreSQL*. **<PluginName>** - 插件的名称。**<SessionNam** - 用于监控项 key 的会话名称。有关特定于插件的描述和支持的模式，请参见 Plugins.<PluginNam

参数	必	项范围	默认值	描述
		Plugins否PluginName>.Sessions.<SessionName>.User		名的会话用户名。支持: *Ceph*, *Memcached*, *MySQL*, *Oracle*, *PostgreSQL*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项 key 的会话名称。

参数	必	项范围	默认值	描述
Plugins.Ceph.Sessions.<sessionName>.ApiKey	否			名会话API密钥。支持: *Ceph*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项key的会话名称。

参数	必	项范围	默认值	描述
Plugins.Oracle.Sessions.<SessionName>.Service	否			于连接的命名会话服务名称(不支持SID)。支持: *Oracle*. **<PluginName>** - 插件的名称。 **<SessionName>** - 用于监控项key的会话名称。

参数	必	项范围	默认值	描述
RefreshActiveChecks	否	0-3600	20	新主动检查列表的频率(以秒为单位)。请注意，刷新主动检查失败后，将在60秒后尝试进行下一次刷新。

参数	必	项范围	默认值	描述
Server		是		号分隔的IP地址列表，可以选择使用CIDR表示法，或者Zabbix servers和Zabbix proxies的DNS名称。仅接受从此处列出的主机传入的连接。如果启用了IPv6支持，则将'127.0.0.1', '::ffff:127.0.0.1'同等对待，并

参数	必	项范围	默认值	描述
ServerActive	否			动检查Zab-bix servers和Zab-bix prox-ies以逗号分隔IP：端口对(或DNS名称：端口对)列表。可以提供多个地址来并行使用多个独立的Zab-bix servers，允许有空格。如果未指定端口，则使用

参数	必	项范围	默认值	描述
SourceIP		否		出连接使用的源IP。
StatusPort		否	024-32767	果设置，proxy将在此端口上监听HTTP状态请求(http://localhost
Timeout		否	-30	处理上花费的时间不超过该(秒)。

参数	必	项范围	默认值	描述
TLSAccept		是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否		制定接受哪些传入连接。用于被动检查。可以指定多个，以逗号分隔: *unencrypted* - 接受不加密的连接 (默认) *psk* - 接受带 TLS 和预共享密钥 (PSK) *cert* - 接受带 TLS 和证书的连接

参数	必	项范围	默认值	描述
TLSCAFile	否			含用于对等证书验证的顶级CA证书的文件的完整路径名，用于Zab-bix组件之间的加密通信。

参数	必	项范围	默认值	描述
TLSCertFile	否			含agent证书或证书链的文件的完整路径名，用于与Zabbix组件进行加密通信。

参数	必	项范围	默认值	描述
TLSConnect	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否			agent 应如何连接到 Zabbix server proxy。用于主动检查。只能指定一个值： *unencrypted* - 不加密连接 (默认) *psk* - 使用 TLS 和预共享密钥 (PSK) *cert* - 使用 TLS 和证书进行连接

参数	必	项范围	默认值	描述
TLSCRLFile	否			含已撤销证书的文件的完整路径名。此参数用于与Zabbix组件的加密通信。
TLSKeyFile	否			含用于与Zabbix组件进行加密通信的agent专用密钥的文件的完整路径名。

参数	必	项范围	默认值	描述
TLSPSKFile	否			含用于与Zab-bix组件进行加密通信的agent预共享密钥的文件的完整路径名。
TLSPSKIdentity	否			共享密钥标识字符串，用于与Zab-bix server进行加密通信。

参数	必项	范围	默认值	描述
TLSServerCertIssuer	否			许的服务器(proxy)证书颁发者。
TLSServerCertSubject	否			许的服务器(proxy)证书主题。

参数	必	项范围	默认值	描述
UnsafeUserParameters	否		,1	许将所有字符都通过参数传递给用户定义的参数。不允许使用以下字符：\ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @另外，不允许使用换行符。

参数	必	项范围	默认值	描述
UserParameter	否			户自定义的监控参数。可以有几个用户定义的参数。格式: User-Pa-ram-e-ter=<key>,<sh com-mand> 请注意，shell 命令不得返回空字符串或仅返回 EOL。示例: User-Pa-ram-e-ter=system.tes -l

5 Zabbix agent (Windows)

概述

本节列出了 Zabbix agent（Windows）配置文件（zabbix_agent.conf）中支持的参数。注意:

- 默认值反映的是进程默认值，而不是出厂配置文件中的值;
- Zabbix 仅支持无 BOM标识的 UTF-8 编码配置文件;
- 支持注释需以 “ # ” 作为行首。

参数

参数必	项范围	默认值	描述
Alias	否		置监控项键值的别名。它可以用一个更小更简单代替长而复杂的监控项键值。可能存在多个*Alias*参数。多个参数允许使用相同的*Alias*。不同*Alias*可能引用相同的监控

参数必	项范围	默认值	描述
AllowKey	否		许执行与模式匹配的那些监控项键值。键值匹配模式是支持通配字符“*”用于匹配任意数量的任何字符。可以结合DenyKey定义多个值的匹配规则。根据其出现顺序

参数必	项范围	默认值	描述
BufferSend	否	-3600	冲区中的数据不要保留超过N秒。
BufferSize	否	-65535	00 存缓冲区中的最大容量。如果缓冲区已满，则agent会将所有收集的数据发送到Zab-bix server或proxy。

参数必	项范围	默认值	描述
DebugLevel	否	-5	定调试级别: 0 - 有关启动和停止 Zab-bix 进程的基本信息 1 - 关键信息 2 - 错误信息 3 - 警告 4 - 用于调试(产生大量信息) 5 - 扩展调试(产生更多信息)

参数必	项范围	默认值	描述
DenyKey	否		绝执行与模式匹配的那些监控项键值。键值匹配模式是支持通配字符“*”用于匹配任意数量的任何字符。可以结合AllowKey定义多个key的匹配规则。根据其出现顺

参数必	项范围	默认值	描述
EnableRemoteCommands	否		否允许来自 Zabbix server 的远程命令。从 Zabbix 5.0.2 开始，此参数已弃用，请使用 AllowKey=system 或 DenyKey=syste 来替代。它是 AllowKey/DenyKe 参数的内部别名，具体取决于值: 0 - DenyKey=syste 1 - AllowKey=system

参数必	项范围	默认值	描述
HostInterface	否	-255 个字符	定义主接口的可选参数。主机接口用于主机自动注册过程。如果值超过 255 个字符的限制，agent 将发出错误并且不会启动。如果未定义，将从 HostInterface 获取值。自 Zabbix 4.4.0 起支持。

参数必	项范围	默认值	描述
HostInterfaceItem	否		于定义用于获取主机接口监控项的可选参数。主机接口用于主机自动注册过程。在自动注册请求期间，如果指定项返回的值超过255个字符的限制，则agent将记录

参数必	项范围	默认值	描述
HostMetadata	否	-255 个字符	定义主元数据的可选参数。主机元数据仅在主机自动注册过程(主动模式 agent)中使用。如果未定义，将从 HostMetadataItem 获取该值。如果指定的值超出限制或非 UTF-8 字符串，则 agent 将

参数必	项范围	默认值	描述
HostMetadataItem	否		于定义用于获取主机元数据的 *Zabbix agent* 监控项可选参数。仅当未定义 HostMetadata 时才使用此选项。支持用户参数，性能计数器和别名。不管 *EnableRemoteCommands* 值如何，都支持 *sys-*

参数必	项范围	默认值	描述	
Hostname	否		Hostnameitem 设置唯一的	区分大小写的主机名。对于主动检查是必需的，并且必须与服务器上配置的主机名匹配。允许的字符:字母,’.’,’,’,’_和’-’。最大长度：128

参数必	项范围	默认值	描述
HostnameItem	否	ystem.hostname	定义用于获取主机名的 *Zabbix agent* 监控项可选参数。仅当未定义主机名时才使用此选项。不支持用户参数，性能计数器或别名。不管 *EnableRemoteCommands* 值如何，都支持 *system.run[]*。

参数必	项范围	默认值	描述
Include	否		可以在配置文件的目录中包括单个文件或所有文件。要仅在指定目录中包含相关文件，模式匹配支持使用 * 通配符。示例: /absolute/pa 从 Zab-bix 2.4.0 开始支持此模式匹配。请

参数必	项范围	默认值	描述	
ListenIP	否		.0.0.0	gent 应监听使用逗号分隔的 IP 地址列表。从 Zab-bix 1.8.3 开始支持多个 IP 地址。
ListenPort	否	024-32767	0050	gent 监听来自服务器的连接所使用的端口。
LogFile	是，如果 LogType 设置为 *file*，否则为否	C:\zabbi	_agentd.log agent 日志文件	名称。

参数必	项范围	默认值	描述
LogFileSize	否	-1024	志文件的最大大小，以MB为单位。 0 - 禁用自动日志轮换。 注意:如果达到了日志文件大小限制并且文件轮换失败，则无论出于何种原因，现有的日志文件都会被

参数必	项范围	默认值	描述
LogType	否	ile	志输出类型: *file* - 将日志写入由 LogFile 参数指定的文件， *system* - 写入日志 Windows 事件日志， *console* - 将日志写入标准输出。从 Zabbix 3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
LogRemoteCommands	否		否启用执行Shell命令记录为警告。 0 - 禁用 1 - 启用

参数必	项范围	默认值	描述
MaxLinesPerSecond	否	-1000	0 处理“log”，“logrt”和“eventlog”主动检查时，agent每秒发送给Zabbix server或proxy的最大新行数。所提供的值将被“log”，“logrt”或“eventlog”监控项key中提供的参数“maxlines”所覆盖。注意:Zabbix将处理

参数必	项范围	默认值	描述
PerfCounter	否		义一个新参数<pa-ram-e-ter_name>，其是系统性能计数器<perf_counter_在指定时间段<pe-riod>(以秒为单位)的平均值。语法:<pa-ram-e-ter_name>,”<p例如,如果希望在最后一分钟接收平均每秒的处理

参数必	项范围	默认值	描述
PerfCounterEn	否		义一个新参数«pa-ram-e-ter_name>，其是系统性能计数器<perf_counter_在指定时间段<pe-riod>(以秒为单位)的平均值。语法:<pa-ram-e-ter_name>,"<p与Per-f-Counter相比，per-f-counter路径必须为英文。仅在**Win-dows Server**

参数必	项范围	默认值	描述
RefreshActiveChecks	否	0-3600	20 动检查刷新列表的频率(以秒为单位)。请注意，主动检查刷新失败后，将在60秒后尝试进行下一次刷新。

参数必	项范围	默认值	描述
Server	是, 如果未将 StartAgents 显式设置为 0		逗号分隔的 IP 地址表，可以选择使用 CIDR 表示法，或者 Zabbix servers 的主机名。仅从此处列出的主机接受传入的连接。如果启用了 IPv6 支持，则将’127.0.0.1’,’::ffff:127.0.0.1’同等对待，并且’::/0’将允许任何 IPv4 或 IPv6 地址。’0.0.0.0/0’可

参数必	项范围	默认值	描述
ServerActive	否	*)	动检查 Zab-bix server 或 Zab-bix proxy 的 IP 地址：端口 (或主机名：端口)。可以提供多个以逗号分隔的 IP 地址，以并行使用多个独立的 Zab-bix servers。允许有空格。如果未指定端口，则使用

参数必	项范围	默认值	描述
SourceIP	否		于以下情形的源IP: - 与Zab-bix server或Zab-bix proxy的传出连接; - 在执行某些监控项(web.page.get, net.tcp.port, 等等)时建立连接。

参数必	项范围	默认值	描述
StartAgents	否	-63 (*)	理被动检查的 zab-bix_agentd 的预分支实例数。如果设置为 0，则禁用被动检查，agent 将不监听任何 TCP 端口。在版本 1.8.5 之前，上限为 16。

参数必	项范围	默认值	描述
Timeout	否	-30	处理上花费的时间不超过该值(秒)。

参数必	项范围	默认值	描述
TLSAccept	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否		接受什么传入连接。用于被动检查。可以指定多个值，逗号分隔: *unencrypted* - 接受不加密的连接 (默认) *psk* - 接受带 TLS 和预共享密钥 (PSK) *cert* - 接受带 TLS 和证书的连接 从 Zabbix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSCAFile	否		含用于对等证书验证的顶级CA证书的文件的完整路径名，用于Zab-bix组件之间的加密通信。从Zab-bix3.0.0开始支持此参数。

参数必	项范围	默认值	描述
TLSCertFile	否		含agent证书或证书链的文件的完整路径名，用于与Zab-bix组件进行加密通信。从Zab-bix3.0.0开始支持此参数。

参数必	项范围	默认值	描述
TLSConnect	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否		agent 应如何连接到 Zabbix server proxy。用于主动检查。只能指定一个值： *unencrypted* - 接受不加密的连接 (默认) *psk* - 接受带 TLS 和预共享密钥 (PSK) *cert* - 接受带 TLS 和证书的连接 从 Zabbix 3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSCRLFile	否		含已撤销证书文件的完整路径名。此参数用于与 Zab-bix 组件的加密通信。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSKeyFile	否		含用于与 Zab-bix 组件进行加密通信的 agent 私钥文件的完整路径名。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSPSKFile	否		含用于与 Zab-bix 组件进行加密通信的 agent 预共享密钥文件的完整路径名。从 Zab-bix3.0.0 开始支持此参数。

参数必	项范围	默认值	描述
TLSPSKIdentity	否		共享密钥标识字符串，用于与Zabbix server进行加密通信。从Zabbix3.0.0开始支持此参数。
TLSServerCertIssuer	否		许的服务器(proxy)证书颁发者。从Zabbix3.0.0开始支持此参数。

参数必	项范围	默认值	描述
TLSServerCertSubject	否		许的服务器(proxy)证书主题。从Zab-bix3.0.0开始支持此参数。

参数必	项范围	默认值	描述
UnsafeUserParameters	否	-1	许将所有字符都通过参数传递给用户定义的参数。 0 - 不允许 1 - 允许不允许使用以下字符: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @另外，不允许

参数必	项范围	默认值	描述
UserParameter			用户自定义的监控参数。可以有多个用户自定义参数。格式: User-Parameter=<key>,<shell command> shell 命令不得返回空字符串或仅返回 EOL。示例: UserParameter=system.test,1

Note:
(*) ServerActive 中列出的活动服务器数加上 StartAgent 中指定的用于被动检查的预分支实例数必须少于 64。

另请参阅

1. 从版本 2.0.0 开始，主动和被动检查的 Zabbix agent 配置差异。

6 Zabbix agent 2 (Windows)

总览

Zabbix agent 2 是新一代的 Zabbix agent ，可以代替 Zabbix agent 使用.

本节列出了 Zabbix agent 2 配置文件（zabbix_agent2.win.conf）中支持的参数。注意:

- 默认值反映的是进程默认值，而不是出厂配置文件中的值;
- Zabbix 仅支持 UTF-8 编码的配置文件，而没有 BOM;
- 仅在行首支持以 “ # ” 开头的注释。

参数

参数	必	项范围	默认值	描述
Alias		否		置监控项key的别名。它可以用一个更小更简单代替长而复杂的监控项key。可能存在多个*Alias*参数。多个参数允许使用相同的*Alias*。不同*Alias*可能引用相同的监控项key。

参数	必	项范围	默认值	描述
AllowKey	否			许执行与模式匹配的那些监控项key。key模式是通配符表达式，支持“*”字符以匹配任意数量的任何字符。可以结合DenyKey定义多个key的匹配规则。根据其出现顺序

参数	必	项范围	默认值	描述
BufferSend	否		-3600	间间隔(以秒为单位)，用于确定从缓冲区向 Zab-bix server 发送值的频率。请注意，如果缓冲区已满，则数据将尽快发送。

参数	必	项范围	默认值	描述
BufferSize	否		-65535	00 存缓冲区中的最大值数。如果缓冲区已满，则代理会将所有收集的数据发送到 Zabbix server 或代理。仅当禁用持久缓冲区时才应使用此参数 (*EnablePersistentBuffer=0*)。

参数	必	项范围	默认值	描述
ControlSocket	否		\.\pipe\agent.sock	的套接字，用于发送带有“-R”选项的运行时命令。

参数	必	项范围	默认值	描述
DebugLevel	否		-5	定调试级别: 0 - 有关启动和停止 Zab-bix 进程的基本信息 1 - 关键信息 2 - 错误信息 3 - 警告 4 - 用于调试(产生大量信息) 5 - 扩展调试(产生更多信息)

参数	必	项范围	默认值	描述
DenyKey	否			绝执行与模式匹配的那些监控项key。key模式是通配符表达式，支持“*”字符以匹配任意数量的任何字符。可以结合AI-lowKey定义多个key的匹配规则。根据其出现顺

参数	必	项范围	默认值	描述
EnablePersistentBuffer	否		-1	主动监控项启用本地永久性存储。 0 - 禁用 1 - 启用 如果禁用持久性存储，则将使用内存缓冲区。

参数	必	项范围	默认值	描述
HostInterface	否	-255 个字符	定义主接口的可选参数。主机接口用于主机自动注册过程。如果值超过 255 个字符的限制，代理将发出错误并且不会启动。如果未定义，将从 HostInterface 获取值。自 Zabbix 4.4.0 起支

参数	必	项范围	默认值	描述
HostInterfaceItem	否			选参数，用于定义用于获取主机接口的监控项。主机接口用于主机自动注册过程。在自动注册请求期间，如果指定项返回的值超过255个字符的限制，则代理将记

参数	必	项范围	默认值	描述
HostMetadata	否		-255 个字符	定义主机元数据的可选参数。主机元数据用于主机自动注册过程。如果指定的值超出限制或非 UTF-8 字符串，则代理将发出错误，并且不会启动。如果未定义，将从 HostMetadataItem 获

参数	必	项范围	默认值	描述
HostMetadataItem	否			选参数，用于定义用于获取主机元数据的项目。每次自动注册尝试时都会检索主机元数据项值，以进行主机自动注册过程。在自动注册请求期间，如果指定项返

参数	必	项范围	默认值	描述
Hostname	否			Hostname设置唯一的Item区分大小写的主机名。对于主动检查是必需的，并且必须与服务器上配置的主机名匹配。允许的字符：字母数字，'.'，' '，'_'和'-'。最大长度：128。

参数	必	项范围	默认值	描述
HostnameItem	否		ystem.hostname	未定义用于生成主机名的监控项。如果定义了主机名，则忽略。不支持 UserParameters 或 aliases，但不管 *EnableRemoteCommands* 值如何，都支持 *system.run[]*。

参数	必	项范围	默认值	描述
Include		否		可以在配置文件的目录中包括单个文件或所有文件。在安装过程中，除非在编译期间进行了修改，否则 Zabbix 将在/usr/local/etc 中创建 include 目录。要仅在指定目录中包含相关

参数	必	项范围	默认值	描述
ListenIP		否	.0.0.0	理应侦听使用逗号分隔的IP地址列表。第一个IP地址被发送到Zab-bix server,如果已连接，以检索主动检查的列表。

参数	必	项范围	默认值	描述
ListenPort		否	024-32767	0050 理侦听来自服务器的连接所使用的端口。
LogFile		是, 如果 LogType 设置为 *file*, 否则为否	c:\zabbi	_agent2.log 如果 Log-Type file’，则记录文件名。

参数	必	项范围	默认值	描述
LogFileSize	否		-1024	志文件的最大大小，以MB为单位。 0 - 禁用自动日志轮换。 注意:如果达到了日志文件大小限制并且文件轮换失败，则无论出于何种原因，现有的日志文件都会被

参数	必	项范围	默认值	描述
LogType		否	ile	定将日志消息写入的位置: *system* - sys-log, *file* - Log-File 参数指定的文件, *console* - 标准输出。

参数	必	项范围	默认值	描述
PersistentBufferFile	否			abbix Agent2 应该在其中保存 SQLite 数据库的文件。必须是完整的文件名。仅当启用了持久缓冲区 (*EnablePersistentBuffer=1*) 时，才使用此参数。

参数	必	项范围	默认值	描述
PersistentBufferPeriod	否		分钟-365 天 1 小时	没有与服务或代理的连接时，应该存储数据的时间段。较旧的数据将丢失。日志数据将被保留。仅当启用了持久缓冲区(*EnablePersistentBuffer=1*)时，才使用此参数。

参数	必	项范围	默认值	描述
Plugins		否		个插件可以具有一个或多个特定于插件的配置参数，格式为: Plugins.<Plugin Plugins.<Plugin

参数	必	项范围	默认值	描述
Plugins否PluginName>.KeepAlive		0-900	00	闭未使用的插件连接之前的最长时间(以秒为单位)。支持以下插件: *Ceph*, *Memcached*, *MySQL*, *Oracle*, *Redis*, *PostgreSQL*. **<PluginName>** - 插件的名称。示例: Plugins.Memc

参数	必	项范围	默认值	描述
Plugins否<PluginName>.Timeout			-30	局超时请求执行时(关闭请求之前等待一个请求完成的时间)。支持以下插件: *Ceph*, *Memcached*, *MySQL*, *Redis*, *Docker*, *PostgreSQL*. **<PluginName>** - 插件的名称。

参数	必	项范围	默认值	描述
Plugins.Ceph.InsecureSkipVerify	否	alse / true	alse	定 http 客户端是否应验证服务器的证书链和主机名。如果为 *true*，则 TLS 接受服务器提供的任何证书以及该证书中的任何主机名。在这种模式下，TLS 容易受到中间人

参数	必项	范围	默认值	描述
Plugins.Ceph.Uri	否		ttps://localhost:8003	连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(仅https受支持)。可以省略端口(默认为8003)。示例:https://127.https://loca

参数	必	项范围	默认值	描述
Plugins.Docker.Endpoint	否		unix:///var/run/docker.sock	Docker 守护程序 unix 套接字位置。必须包含一个方案(仅 unix:// 支持)。

参数	必	项范围	默认值	描述
Plugins.Log.MaxLinesPerSecond	否		-1000	0 处理“日志”和“事件日志”主动检查时，代理每秒发送给 Zabbix server 或代理的最大新行数。所提供的值将被“log”或“eventlog”监控项中提供的参数“maxlines”所覆盖。注意：Zabbix 处

参数	必	项范围	默认值	描述
Plugins.Memcached.Uri	否		tcp://localhost:11211	连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(支持: tcp, unix)。可以省略端口(默认=11211)。示例: tcp://localh tcp://localh unix:/var/ru

参数	必	项范围	默认值	描述
Plugins.Mysql.Uri	否		tcp://localhost:3306	MySQL 连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与 URI 格式匹配。必须包含一个方案(支持: tcp, unix)。可以省略端口(默认为 3306)。示例: tcp://localh tcp://localh unix:/var/ru

参数	必	项范围	默认值	描述
Plugins.Oracle.CallTimeout	否		-30	局超时完成请求最大等待时间(以秒为单位)。
Plugins.Oracle.ConnectTimeout	否		-30	局超时建立连接最大等待时间(以秒为单位)。
Plugins.Oracle.CustomQueriesPath	否			含带有自定义查询的.sql文件的目录的完整路径名。默认禁用。示例:/etc/zabbix/

参数	必项	范围	默认值	描述
Plugins.Oracle.Service	否			E于连接的服务名称(不支持SID)。

参数	必	项范围	默认值	描述
Plugins.Oracle.Uri	否		tcp://localhost:1521	连接字符串。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(支持:tcp)。可以省略端口(默认=1521)。示例:tcp://localh tcp://localh

参数	必	项范围	默认值	描述
Plugins.Postgres.Database	否		ostgres	ostgreSQL 使用的数据库名称。
Plugins.Postgres.Host	否		ocalhost	ostgreSQL 使用的主机的 IP 地址或 DNS 名称。示例: `localhost`, `192.168.1.1`
Plugins.Postgres.Port	否		432	于 Post-greSQL 的端口。

参数	必	项范围	默认值	描述
Plugins.Redis.Uri	否		tcp://localhost:6379	连接字符串。可以省略端口(默认为6379)。不应包含嵌入式凭据(它们将被忽略)。必须与URI格式匹配。必须包含一个方案(支持: tcp, unix)。示例: tcp://localh tcp://localh unix:/var/ru

参数	必	项范围	默认值	描述
Plugins.SystemRun.EnableRemoteCommands	否			否允许来自Zabbix server的远程命令。 0 - 不允许 1 - 允许 从5.0.2开始不支持此参数,请改用AllowKey/DenyKey参数。

参数	必	项范围	默认值	描述
Plugins.	否	SystemRun.LogRemoteCommands		用将执行的Shell命令记录为警告。 0 - 禁用 1 - 启用仅当远程执行命令时，才会记录命令。如果通过Host-Meta-dataItem，HostIn-ter-faceItem或Host-nameItem参数在本地启动sys-tem.run[]，则不会创建日

参数	必	项范围	默认值	描述
Plugins.Windows.Eventlog.MaxLinesPerSecond	否		-1000	0 理每秒发送给 Zab-bix server 或代理处理“事件日志”检查的最大新行数。所提供的值将被“event-log”监控项 key 中提供的参数“max-lines”所覆盖。

参数	必	项范围	默认值	描述
Plugins’ named ses-sions		否		果使用 Zab-bix agent 监视相同种类的多个实例，则可以为每个实例创建具有自己的一组授权参数的命名会话。命名的会话参数格式: Plugins.<Plugin Plugins.<Plugin

参数	必	项范围	默认值	描述
	否	Plugins.<PluginName>.Sessions.<SessionName>.Password		名会话密码。支持: *Memcached*, *MySQL*, *Oracle*, *PostgreSQL*, *Redis*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项 key 的会话名称。

参数	必	项范围	默认值	描述
Plugins否<PluginName>.Sessions.<SessionName>.Uri				名会话连接字符串。支持: *Ceph*, *Memcached*, *MySQL*, *Oracle*, *Redis*, *PostgreSQL*. **<PluginName>** - 插件的名称。**<SessionNam** - 用于监控项 key 的会话名称。有关特定于插件的描述和支持的模式，请参见 Plugins.<PluginNam

参数	必	项范围	默认值	描述
		Plugins否PluginName>.Sessions.<SessionName>.User		名的会话用户名。支持: *Ceph*, *Memcached*, *MySQL*, *Oracle*, *PostgreSQL*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项 key 的会话名称。

参数	必	项范围	默认值	描述
Plugins.Ceph.Sessions.<sessionName>.ApiKey	否			名会话API密钥。支持: *Ceph*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项key的会话名称。

参数	必	项范围	默认值	描述
Plugins.Oracle.Sessions.<SessionName>.Service	否			于连接的命名会话服务名称(不支持SID)。支持: *Oracle*. **<PluginName>** - 插件的名称。 **<SessionNam** - 用于监控项key的会话名称。

参数	必	项范围	默认值	描述
RefreshActiveChecks	否	0-3600	20	新主动检查列表的频率(以秒为单位)。请注意，刷新主动检查失败后，将在60秒后尝试进行下一次刷新。

参数	必	项范围	默认值	描述
Server		是		号分隔的IP地址列表，可以选择使用CIDR表示法，或者Zabbix servers和Zabbix proxies的DNS名称。仅从此处列出的主机接受传入的连接。如果启用了IPv6支持，则将'127.0.0.1', '::ffff:127.0.0.1'同等对待，并

参数	必	项范围	默认值	描述
ServerActive	否			动检查Zab-bix servers和Zab-bix prox-ies以逗号分隔IP：端口对(或DNS名称：端口对)列表。可以提供多个地址来并行使用多个独立的Zab-bix servers，允许有空格。如果未指定端口，则使用

参数	必	项范围	默认值	描述
SourceIP	否			出连接的源IP地址。
StatusPort	否		024-32767	果设置，代理将在此端口上侦听HTTP状态请求(http://localhost
Timeout	否		-30	处理上花费的时间不超过超时秒。

参数	必	项范围	默认值	描述
TLSAccept	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否			接受什么传入连接。用于被动检查。可以指定多个值，逗号分隔: *unencrypted* - 接受不加密的连接 (默认) *psk* - 接受带 TLS 和预共享密钥 (PSK) *cert* - 接受带 TLS 和证书的连接

参数	必	项范围	默认值	描述
TLSCAFile	否			含用于对等证书验证的顶级CA证书的文件的完整路径名，用于Zab-bix组件之间的加密通信。

参数	必	项范围	默认值	描述
TLSCertFile	否			含代理证书或证书链的文件的完整路径名，用于与 Zabbix 组件进行加密通信。

参数	必	项范围	默认值	描述
TLSConnect	是, 如果定义了 TLS 证书或 PSK 参数 (甚至用于未加密的连接), 否则为否			代理应如何连接到 Zabbix server 或代理用于主动检查。只能指定一个值： *unencrypted* - 不加密连接 (默认) *psk* - 使用 TLS 和预共享密钥 (PSK) *cert* - 使用 TLS 和证书进行连接

参数	必	项范围	默认值	描述
TLSCRLFile	否			含已撤销证书的文件的完整路径名。此参数用于与Zabbix组件的加密通信。
TLSKeyFile	否			含用于与Zabbix组件进行加密通信的代理专用密钥的文件的完整路径名。

参数	必	项范围	默认值	描述
TLSPSKFile	否			含用于与 Zab-bix 组件进行加密通信的代理预共享密钥的文件的完整路径名。
TLSPSKIdentity	否			共享密钥标识字符串，用于与 Zab-bix server 进行加密通信。

参数	必项	范围	默认值	描述
TLSServerCertIssuer	否			许的服务器(代理)证书颁发者。
TLSServerCertSubject	否			许的服务器(代理)证书主题。

参数	必项	范围	默认值	描述
UnsafeUserParameters	否		,1	许将所有字符都通过参数传递给用户定义的参数。不允许使用以下字符：\ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @另外，不允许使用换行符。

参数	必	项范围	默认值	描述
UserParameter	否			户定义的参数进行监视。可以有几个用户定义的参数。格式: User-Pa-ram-e-ter=<key>,<sh com-mand> 请注意，shell 命令不得返回空字符串或仅返回 EOL。示例: User-Pa-ram-e-ter=system.tes -l

7 Zabbix Java gateway

如果使用 `startup.sh` 和 `shutdown.sh` 脚本启动和停止Zabbix Java gateway, 则可以在 `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)。

如果通过手动运行 `java` 命令来起动 Zabbix Java gateway, 可以通过命令行方式来指定 Java 属性.

变量参	必须配	范围	默认值	描述信息	
LISTEN_IP	zabbix.listenIP	否		.0.0.0	听IP。
LISTEN_PORT	zabbix.listenPort	否	024-32767	0052	听端口。
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。

变量参	必须配	范围	默认值	描述信息
PROPERTIES_FILE	zabbix.propertiesFile	否		性文件的名称。可用于使用键值格式设置其他属性，以使其在命令行上不可见或覆盖现有属性。例如：“javax.net.ssl.trus=<密码>”
START_POLLERS	zabbix.startPollers	否	-1000	动多少个轮询线程。

变量参	必须配	范围	默认值	描述信息
TIMEOUT	zabbix.timeout	否	-30	络超时时间。从 Zab-bix 2.0.15, 2.2.10 和 2.4.5 开始支持该参数。

Warning:
端口 10052 没有IANA 注册.

8 概述

概述

可以使用 Include 参数将其他文件或目录包含在服务器/代理/代理配置中。

包含注意事项

如果 Include 参数用于包含文件，则该文件必须可读。

如果 Include 参数用于包含目录：

- 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

如果 Include 参数使用模式来匹配包含的文件：

- 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(

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": "5.0.0"
}
```

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
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*

name	value type	description
	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
	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

name	value type	description
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:

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
        }
    ],
    "version":"5.0.0"
}
```


server→proxy:


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

```
}
```

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": "5.0.0"
}
```

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'
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')

例如:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version":"5.0.0"
}
```

server→proxy:

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

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

proxy→server:

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

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'

name	value type	description
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	*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
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",
  "session": "12345678901234567890123456789012",
  "host availability":[
        {
            "hostid":10106,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        },
        {
```

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

```
        }
    ],
    "tasks":[
        {
            "type": 2,
            "clock":1478608371,
            "ttl": 600,
            "commandtype": 2,
            "command": "restart_service1.sh",
            "execute_on": 2,
            "port": 80,
            "authtype": 0,
            "username": "userA",
            "password": "password1",
            "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
            "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5uO9lHNsj6tQ5QCqGKukO1De7zhd",
            "parent_taskid": 10,
            "hostid": 10070
        },
        {
            "type": 2,
            "clock":1478608381,
            "ttl": 600,
            "commandtype": 1,
            "command": "restart_service2.sh",
            "execute_on": 0,
            "authtype": 0,
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "parent_taskid": 20,
            "hostid": 10084
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "version":"5.0.0"
}
```

server→proxy:

```
{
  "response": "success",
  "tasks":[
      {
         "type": 1,
         "clock": 1478608371,
         "ttl": 600,
         "commandtype": 2,
         "command": "restart_service1.sh",
         "execute_on": 2,
```

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

2 Zabbix Agent 协议

有关详细信息，请参阅被动和主动代理检查页面。

3 Zabbix agent 2 protocol

Overview

This section provides information on:

- Agent2 -> Server : active checks request
- Server -> Agent2 : active checks response
- Agent2 -> Server : agent data request
- Server -> Agent2 : agent data response

Active checks request

The active checks request is used to obtain the active checks to be processed by agent. This request is sent by the agent upon start and then with *RefreshActiveChecks* inervals.

Field	Type	Value
request	*string*	`active checks`
host	*string*	Host name.
version	*string*	The agent version: <major>.<minor>.
host_metadata	*string*	The configuration parameter HostMetadata or HostMetadataItem metric value (optional).
interface	*string*	The configuration parameter HostInterface or HostInterfaceItem metric value (optional).
ip	*string*	The configuration parameter ListenIP first IP if set (optional).
port	*number*	The configuration parameter ListenPort value if set and not default agent listening port (optional).

Example:

```
{
  "request": "active checks",
  "host": "Zabbix server",
  "version": "6.0",
  "host_metadata": "mysql,nginx",
  "hostinterface": "zabbix.server.lan",
  "ip": "159.168.1.1",
  "port": 12050
}
```

Active checks response

The active checks response is sent by the server back to agent after processing active checks request.

Field	Type	Value
response	*string*	`success` | `failed`
info	*string*	Error information in the case of failure (optional).
data	*array of objects*	Active check items (optional).
key	*string*	Item key with expanded macros.
itemid	*number*	Item identifier.
delay	*string*	Item update interval.
lastlogsize	*number*	Item lastlogsize.
mtime	*number*	Item mtime.
refresh_unsupported	*number*	Unsupported item refresh interval (agent version 5.4).
regexp	*array of objects*	Global regular expressions (optional).
name	*string*	Global regular expression name.
expression	*string*	Global regular expression.
expression_type	*number*	Global regular expression type.
exp_delimiter	*string*	Global regular expression delimiter.
case_sensitive	*number*	Global regular expression case sensitiviness setting.

Example:

```
{
  "response": "success",
  "data": [
    {
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "itemid": 1234,
      "delay": "30s",
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "agent.version",
      "itemid": 5678,
      "delay": "10m",
      "lastlogsize": 0,
      "mtime": 0
    }
  ]
}
```

Agent data request

The agent data request contains the gathered item values.

Field	Type	Value
request	*string*	`agent data`
host	*string*	Host name.
version	*string*	The agent version: <major>.<minor>.

Field	Type	Value
session	*string*	Unique session identifier generated each time when agent is started.
data	*array of objects*	Item values.
id	*number*	The value identifier (incremental counter used for checking duplicated values in the case of network problems).
itemid	*number*	Item identifier.
value	*string*	The item value (optional).
lastlogsize	*number*	The item lastlogsize (optional).
mtime	*number*	The item mtime (optional).
state	*number*	The item state (optional).
source	*string*	The value event log source (optional).
eventid	*number*	The value event log eventid (optional).
severity	*number*	The value event log severity (optional).
timestamp	*number*	The value event log timestamp (optional).
clock	*number*	The value timestamp (seconds since Epoch).
ns	*number*	The value timestamp nanoseconds.

Example:

```
{
  "request": "agent data",
  "data": [
    {
      "id": 1,
      "itemid": 5678,
      "value": "2.4.0",
      "clock": 1400675595,
      "ns": 76808644
    },
    {
      "id": 2,
      "itemid": 1234,
      "lastlogsize": 112,
      "value": " 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 5000
      "clock": 1400675595,
      "ns": 77053975
    }
  ],
  "host": "Zabbix server",
  "version": "6.0",
  "sessionid": "1234456akdsjhfoui"
}
```

Agent data response

The agent data response is sent by the server back to agent after processing the agent data request.

Field	Type	Value
response	*string*	`success` | `failed`
info	*string*	Item processing results.

Example:

```
{
  "response": "success",
  "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.003534"
}
```

3 Zabbix sender 协议

有关详细信息，请参阅Trapper items页面。

4 Zabbix agent 2 plugin protocol

Zabbix agent 2 protocol is based on code, size and data model.

Code

Type	Size	Comments
Byte	4	Payload type, currently only JSON is supported.

Size

Type	Size	Comments
Byte	4	Size of the current payload in bytes.

Payload data

Type	Size	Comments
Byte	Defined by the *Size* field	JSON formatted data.

Payload data definition

Common data

These parameters are present in all requests/responses:

Name	Type	Comments
id	uint32	For requests - the incrementing identifier used to link requests with responses. Unique within a request direction (i.e. from agent to plugin or from plugin to agent). For responses - ID of the corresponding request.
type	uint32	The request type.

Log request

A request sent by a plugin to write a log message into the agent log file.

direction	plugin → agent
response	no

Parameters specific to log requests:

Name	Type	Comments
severity	uint32	The message severity (log level).
message	string	The message to log.

Example:

```
{"id":0,"type":1,"severity":3,"message":"message"}
```

Register request

A request sent by the agent during the agent startup phase to obtain provided metrics to register a plugin.

direction	agent → plugin
response	yes

Parameters specific to register requests:

Name	Type	Comments
version	string	The protocol version <major>.<minor>

Example:

```
{"id":1,"type":2,"version":"1.0"}
```

Register response

Plugin's response to the register request.

direction	plugin → agent
response	n/a

Parameters specific to register responses:

Name	Type	Comments
name	string	The plugin name.
metrics	array of strings (optional)	The metrics with descriptions as used in the plugin. Returns RegisterMetrics(). Absent if error is returned.
interfaces	uint32 (optional)	The bit mask of plugin's supported interfaces. Absent if error is returned.
error	string (optional)	An error message returned if a plugin cannot be started. Absent, if metrics are returned.

Examples:

```
{"id":2,"type":3,"metrics":["external.test", "External exporter Test."], "interfaces": 4}
```

or

```
{"id":2,"type":3,"error":"error message"}
```

Start request

A request to execute the Start function of the Runner interface.

direction	agent → plugin
response	no

The request doesn't have specific parameters, it only contains common data parameters.

Example:

```
{"id":3,"type":4}
```

Terminate request

A request sent by the agent to shutdown a plugin.

direction	agent → plugin
response	no

The request doesn't have specific parameters, it only contains common data parameters.

Example:

```
{"id":3,"type":5}
```

Export request

A request to execute the Export function of the Exporter interface.

direction	agent → plugin
response	no

Parameters specific to export requests:

Name	Type	Comments
key	string	The plugin key.
parameters	array of strings (optional)	The parameters for Export function.

Example:

```
{"id":4,"type":6,"key":"test.key","parameters":["foo","bar"]}
```

Export response

Response from the Export function of the Exporter interface.

direction	plugin → agent
response	n/a

Parameters specific to export responses:

Name	Type	Comments
value	string (optional)	Response value from the Export function. Absent, if error is returned.
error	string (optional)	Error message if the Export function has not been executed successfully. Absent, if value is returned.

Examples:

```
{"id":5,"type":7,"value":"response"}
```

or

```
{"id":5,"type":7,"error":"error message"}
```

Configure request

A request to execute the *Configure* function of the *Configurator* interface.

direction	agent → plugin
response	n/a

Parameters specific to *Configure* requests:

Name	Type	Comments
global_options	JSON object	JSON object containing global agent configuration options.
private_options	JSON object (optional)	JSON object containing private plugin configuration options, if provided.

Example:

```
{"id":6,"type":8,"global_options":{...},"private_options":{...}}
```

Validate request

A request to execute *Validate* function of the *Configurator* interface.

direction	agent → plugin
response	yes

Parameters specific to *Validate* requests:

Name	Type	Comments
private_options	JSON object (optional)	JSON object containing private plugin configuration options, if provided.

Example:

```
{"id":7,"type":9,"private_options":{...}}
```

Validate response

Response from *Validate* function of *Configurator* interface.

direction	plugin → agent
response	n/a

Parameters specific to *Validate* responses:

Name	Type	Comments
error	string (optional)	An error message returned if the Validate function is not executed successfully. Absent if executed successfully.

Example:

```
{"id":8,"type":10}
```

or

```
{"id":8,"type":10,"error":"error message"}
```

4 标头

Overview

The header is present in response and request messages between Zabbix components. It is required to determine the length of message, if it is compressed or not and the format of message length fields. The header consists of:

```
<PROTOCOL> - "ZBXD" (4 bytes).
<FLAGS> - the protocol flags, (1 byte). 0x01 - Zabbix communications protocol, 0x02 - compression, 0x04 -
<DATALEN> - data length (4 bytes or 8 bytes for large packet). 1 will be formatted as 01/00/00/00 (four by
<RESERVED> - uncompressed data length (4 bytes or 8 bytes for large packet). 1 will be formatted as 01/00/
```

When compression is enabled (0x02 flag) the <RESERVED> bytes contains uncompressed data size. When compression is not enabled then <RESERVED> should be zeroes.

Zabbix protocol has 1GB packet size limit per connection. The limit of 1GB is applied for received packet data length and for uncompressed data length, however, when large packet is enabled (0x04 flag) it is possible for Zabbix proxy to receive configuration with size up to 16GB; note that large packet can only be used for Zabbix proxy configuration, and Zabbix server will automatically set (0x04 flag) and send length fields as 8 bytes each when data length before compression exceeds 4GB.

Structure

The header consists of four fields. All numbers in the header are formatted as little-endian.

Field	Size	Size (large packet)	Description
<PROTOCOL>	4	4	"ZBXD" or 5A 42 58 44
<FLAGS>	1	1	Protocol flags: 0x01 - Zabbix communications protocol 0x02 - compression 0x04 - large packet
<DATALEN>	4	8	Data length.
<RESERVED>	4	8	When compression is used (0x02 flag) - the length of uncompressed data When compression is not used - 00 00 00 00

Implementation

这些代码片段展示了如何将 Zabbix 协议标头添加到你想要发送的数据中，从而使 zabbix 正确地解析数据包。

语言代	
bash	printf -v LENGTH
Java	byte[] header =
PHP	$packet = "ZBXD\
Perl	my $packet = "ZB
Python	packet = "ZBXD\1

5 实时导出协议

本节以换行符分隔的 JSON 格式显示实时导出协议的详细信息，用于：

- 触发事件
- 监控项值
- 趋势

所有文件均具有.ndjson 扩展名。导出文件的每一行都是一个 JSON 对象。

触发事件

针对问题事件导出以下信息：

字段	类	描述	
hosts		数组触	器表达式中涉及的主机列表；数组中至少应包含一个元素。
	-	对象	
	host	字符串主机	称。
	name	字符串可见	主机名称。
groups		数组触	器表达式中涉及的所有主机的主机组列表；数组中至少应包含一个元素。
	-	字符串主机	名称。
tags		数组问	标签列表（可以为空）。
	-	对象	
	tag	字符串标签	称。
	value	字符串标签	(可以为空)。
name		字符串问题	件名。
clock		数字从	期开始到检测到问题的时间（整数部分）的秒数。
ns		数字将	秒添加到时钟以获取精确的问题检测时间。
eventid		数字问	事件 ID。
value		数字 1	通常)。

导出以下信息以进行恢复事件：

Field	类型描	
clock	数字从	期开始到问题解决为止的秒数（整数部分）。
ns	数字将	秒添加到添加到 “时钟” 以得到精确的问题解决时间。

Field	类型描	
eventid	数字恢	事件 ID。
p_eventid	数字问	事件 ID。
value	数字 0	通常）。

示例

问题：

```
{"hosts":[{"host":"Host B", "name":"Host B visible"},{"host":"Zabbix Server","name":"Zabbix Server visible
```

恢复：

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

问题（生成多个问题事件）：

```
{"hosts":[{"host":"Host B", "name":"Host B visible"},{"host":"Zabbix Server","name":"Zabbix Server visible
```

```
{"hosts":[{"host":"Host B", "name":"Host B visible"},{"host":"Zabbix Server","name":"Zabbix Server visible
```

恢复：

```
{"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

导出以下信息以收集项目值：

字段	类	描述	
host		对象监	项主机的主机名称。
	host	字符串主机	称。
	name	字符串可见	主机名称。
groups		数组监	项主机的主机组列表；数组中至少应包含一个元素。
	-	字符串主机	名称。
applications		数组项	应用列表；如果没有，则为空。
	-	字符串应用	称。
itemid		数字监	项 ID。
name		字符串可见	监控项名称。
clock		数字从	期开始到值被收集为止的秒数（整数部分）。
ns		数字将	秒添加到 “时钟” 以获取精确的值收集时间。
timestamp (仅日志)		数字 0	如果不可用）。
source (仅日志)		字符串空字	串（如果不可用）。
severity (仅日志)		数字 0	如果不可用）。
eventid (仅日志)		数字 0	如果不可用）。
value		数字（对于数字）或\\字符串（对于文本）收集监控项的值。	
type		数字收	的值类型: 0 - 浮点数, 1 - 字符, 2 - 日志, 3 - 无符号数字, 4 -文本

示例

数值（无符号）：

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

数值（浮点数）：

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

字符, 文本：

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

日志：

{"host":{"host":"Host A","name":"Host A visible"},"groups":["Group X","Group Y","Group Z"],"applications":

趋势

导出以下信息以获取计算出的趋势值：

字段	类	描述	
host		对象监	项主机的主机名称。
	host	字符串主机	称。
	name	字符串可见	主机名称。
groups		数组监	项主机的主机组列表；数组中至少应包含一个元素。
	-	字符串主机	名称。
applications		数组项	应用列表；如果没有，则为空。
	-	字符串应用	称。
itemid		数字监	项 ID。
name		字符串可见	监控项名称。
clock		数字从	期开始到值被收集为止的秒数（整数部分）。
count		数字给	小时内收集的值的数量。
min		数字给	小时内监控项的最小值。
avg		数字给	小时内监控项的平均值。
max		数字给	小时内监控项的最大值。
type		数字值	型： 0-浮点数，3-无符号数字

示例

数值（无符号）：

{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":

数值（浮点数）：

{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":

5 监控项

5 监控项

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 监控项目:

- 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:**
> Windows 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 / 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>,<maxdelay>]					X[4]	X	X	X	X	X	X	X	X	X	X
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]					X[4]	X	X	X	X	X	X	X	X	X	X
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]					X[4]	X	X	X	X	X	X	X	X	X	X
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]					X[4]	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[2]	X	X	-	X	X	r
	(default)														
	packets				X	X	X	X	X	X	X	-	X	X	r
	errors				X	X	X	X	X[2]	X	X	-	X	X	r
	dropped				X	X	X	X	-	X	-	-	X	X	r
	overruns				-	X	X	-	-	-	-	-	-	-	-
	frame				-	X	X	-	-	-	-	-	-	-	-
	compressed				-	X	X	-	-	-	-	-	-	-	-
	multicast				-	X	X	-	-	-	-	-	-	-	-
net.if.out[if,<mode>]					X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes			▲	X	X	X	X	X[2]	X	X	-	X	X	r
	(default)														
	packets				X	X	X	X	X	X	X	-	X	X	r
	errors				X	X	X	X	X[2]	X	X	-	X	X	r
	dropped				X	X	X	-	-	X	-	-	-	-	-
	overruns				-	X	X	-	-	-	-	-	-	-	-
	collision				-	X	X	-	-	-	-	-	-	-	-
	carrier				-	X	X	-	-	-	-	-	-	-	-
	compressed				-	X	X	-	-	-	-	-	-	-	-
net.if.total[if,<mode>]					X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes			▲	X	X	X	X	X[2]	X	X	-	X	X	r
	(default)														
	packets				X	X	X	X	X	X	X	-	X	X	r
	errors				X	X	X	X	X[2]	X	X	-	X	X	r
	dropped				X	X	X	-	-	X	-	-	-	-	-
	overruns				-	X	X	-	-	-	-	-	-	-	-
	compressed				-	X	X	-	-	-	-	-	-	-	-
net.tcp.listen[port]					X	X	X	X	X	-	-	-	X	-	-
net.tcp.port[<ip>,port]					X	X	X	X	X	X	X	X	X	X	X
net.tcp.service[service,<ip>,<port>]					X	X	X	X	X	X	X	X	X	X	X
net.tcp.service.perf[service,<ip>,<port>]					X	X	X	X	X	X	X	X	X	X	X
net.udp.listen[port]					X	X	X	X	X	-	-	-	X	-	-
net.udp.service[service,<ip>,<port>]					X	X	X	X	X	X	X	X	X	X	X
net.udp.service.perf[service,<ip>,<port>]					X	X	X	X	X	X	X	X	X	X	X

		1	2	3	4	5	6	7	8	9	10	11
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]						X [3]			-	-	-	-
type ▲	total *(default)*	-	X	X	-	X	-	-	-	-	-	-
	user	-	X	X	-	X	-	-	-	-	-	-
	system	-	X	X	-	X	-	-	-	-	-	-
mode ▲	avg1 *(default)*	-	X	X	-	X	-	-	-	-	-	-
	avg5	-	X	X	-	X	-	-	-	-	-	-
	avg15	-	X	X	-	X	-	-	-	-	-	-
zone ▲	current *(default)*	-	-	-	-	X	-	-	-	-	-	-
	all	-	-	-	-	X	-	-	-	-	-	-
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]			X	X	X	X [3]		X	X	-	X	X
mode ▲	sum *(default)*	-	X	X	X	X	-	X	X	-	X	X
	avg	-	X	X	X	X	-	X	X	-	X	X
	max	-	X	X	X	X	-	X	X	-	X	X
	min	-	X	X	X	X	-	X	X	-	X	X
memtype ▲		-	X	X	X	X	-	X	-	-	-	-
proc.num[<name>,<user>,<state>,<cmdline>,<zone>]			X	X	X	X [3]	X	X	X	-	X	X
state ▲	all *(default)*	-	X	X	X	X	X	X	X	-	X	X
	disk	-	X	X	X	-	-	-	-	-	X	X
	sleep	-	X	X	X	X	X	X	X	-	X	X
	zomb	-	X	X	X	X	X	X	X	-	X	X
	run	-	X	X	X	X	X	X	X	-	X	X
	trace	-	X	X	X	-	-	-	-	-	X	X
cmdline ▲		-	X	X	X	X	X	X	X	-	X	X
zone ▲	current *(default)*	-	-	-	-	X	-	-	-	-	-	-
	all	-	-	-	-	X	-	-	-	-	-	-
sensor[device,sensor,<mode>]			X	X	-	-	-	-	-	-	X	-
system.boottime		-	X	X	X	X	-	-	-	X	X	X
system.cpu.discovery		X	X	X	X	X	X	X	X	X	X	X
system.cpu.intr		-	X	X	X	X	-	X	-	-	X	X
system.cpu.load[<cpu>,<mode>]		X	X	X	X	X	X	X	X	X	X	X
cpu ▲	all *(default)*	X	X	X	X	X	X	X	X	X	X	X
	percpu	X	X	X	X	X	X	X	-	X	X	X
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	X	X	X
	avg5	X	X	X	X	X	X	X	X	X	X	X
	avg15	X	X	X	X	X	X	X	X	X	X	X
system.cpu.num[<type>]		X	X	X	X	X	X	X	-	X	X	X
type ▲	online *(default)*	X	X	X	X	X	X	X	-	X	X	X
	max	-	X	X	X	X	-	-	-	X	-	-
system.cpu.switches			X	X	X	X	-	X	-	-	X	X
system.cpu.util[<cpu>,<type>,<mode>,<logical_or_physical>]			X	X	X	X	X	X	X	-	X	X

type ▲	user *(default)*	-	X	X	X	X	X	X	X	-	X	X
	nice	-	X	X	X	-	X	-	X	-	X	X
	idle	-	X	X	X	X	X	X	X	-	X	X
	system *(default for Windows)*	X	X	X	X	X	X	X	X	-	X	X
	iowait	-	-	X	-	X	-	X	-	-	-	-
	interrupt	-	-	X	X	-	-	-	-	-	X	-
	softirq	-	-	X	-	-	-	-	-	-	-	-
	steal	-	-	X	-	-	-	-	-	-	-	-
	guest	-	-	X	-	-	-	-	-	-	-	-
	guest_nice	-	-	X	-	-	-	-	-	-	-	-
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	-	X	X
	avg5	X	X	X	X	X	X	X	-	-	X	X
	avg15	X	X	X	X	X	X	X	-	-	X	X
logical_or_physical ▲ *(since Zabbix 5.0.3)*	logical *(default)*	-	-	-	-	-	-	X	-	-	-	-
	physical	-	-	-	-	-	-	X	-	-	-	-
		1	2	3	4	5	6	7	8	9	10	11
system.hostname[<type>]		X	X	X	X	X	X	X	X	X	X	X
system.hw.chassis[<info>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.cpu[<cpu>,<info>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.devices[<type>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.macaddr[<interface>,<format>]		X	X	X	-	-	-	-	-	-	-	-
system.localtime[<type>]		X	X	X	X	X	X	X	X	X	X	X
type ▲	utc *(default)*	X	X	X	X	X	X	X	X	X	X	X
	local	X	X	X	X	X	X	X	X	X	X	X
system.run[command,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	wait *(default)*	X	X	X	X	X	X	X	X	X	X	X
	nowait	X	X	X	X	X	X	X	X	X	X	X
system.stat[resource,<type>]		-	-	-	-	-	-	X	-	-	-	-
system.sw.arch		X	X	X	X	X	X	X	X	X	X	X
system.sw.os[<info>]		X	X	X	-	-	-	-	-	-	-	-
system.sw.packages[<package>,<manager>,<format>]		-	X	X	-	-	-	-	-	-	-	-
system.swap.in[<device>,<type>] *(specifying a device is only supported under Linux)*		X	X	X	-	X	-	-	-	-	X	-

type ▲ (pages will only work if device was not specified)	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-
system.swap.out[<device>,<type>] *(specifying a device is only supported under Linux)*		-	X	X	-	X	-	-	-	-	X	-
type ▲ (pages will only work if device was not specified)	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-
system.swap.size[<device>,<type>] *(specifying a device is only supported under FreeBSD, for other platforms must be empty or "all")*		X	X	X	-	X	-	X	X	-	X	-

type ▲	free *(default)*	X	X	X	X	X	-	X	X	-	X	-
	total	X	X	X	X	X	-	X	X	-	X	-
	used	X	X	X	X	X	-	X	X	-	X	-
	pfree	X	X	X	X	X	-	X	X	-	X	-
	pused	X [6]	X	X	X	X	-	X	X	-	X	-
system.uname		X	X	X	X	X	X	X	X	X	X	X
system.uptime		X	X	X	X	X	-	X	?	X	X	X
system.users.num		-	X	X	X	X	X	X	X	X	X	X
systemd.unit.discovery		X	X	-	-	-	-	-	-	-	-	-
systemd.unit.get		-	X	X	-	-	-	-	-	-	-	-
systemd.unit.info		-	X	X	-	-	-	-	-	-	-	-
		1	2	3	4	5	6	7	8	9	10	11
vfs.dev.discovery		-	X	X	-	-	-	-	-	-	-	-
vfs.dev.read[<device>,<type>,<mode>]		X	X	X	X	X	-	X	-	-	X	-
type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
	operations *(default for OpenBSD, AIX)*	-	X	X	X	X	-	X	-	-	X	-
	bytes *(default for Solaris)*	-	-	-	X	X	-	X	-	-	X	-
	sps *(default for Linux)*	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps *(default for FreeBSD)*	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dev.write[<device>,<type>,<mode>]		X	X	X	X	X	-	X	-	-	X	-
type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
	operations *(default for OpenBSD, AIX)*	-	X	X	X	X	-	X	-	-	X	-

	bytes *(default for Solaris)*	-	-	-	X	X	-	X	-	-	X	-
	sps *(default for Linux)*	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps *(default for FreeBSD)*	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age		X	X	X	X	X	X	X	X	X	X	X
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]		?	?	?	?	?	?	?	?	?	?	?
vfs.file.cksum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.contents[file,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.exists[file,<types_incl>,<types_excl>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.md5sum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regexp[file,regexp,<encoding>,<output>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regmatch[file,regexp,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.size[file]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
vfs.file.time[file,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	modify *(default)*	X	X	X	X	X	X	X	X	X	X	X
	access	X	X	X	X	X	X	X	X	X	X	X
	change	X[5]	X	X	X	X	X	X	X	X	X	X
vfs.fs.discovery		X	X	X	X	X	X	X	-	X	X	X
vfs.fs.get		X	X	X	X	X	X	X	-	X	X	X
vfs.fs.inode[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(default)*	-	X	X	X	X	X	X	X	X	X	X
	free	-	X	X	X	X	X	X	X	X	X	X
	used	-	X	X	X	X	X	X	X	X	X	X
	pfree	-	X	X	X	X	X	X	X	X	X	X
	pused	-	X	X	X	X	X	X	X	X	X	X
vfs.fs.size[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(default)*	X	X	X	X	X	X	X	X	X	X	X
	free	X	X	X	X	X	X	X	X	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pfree	X	X	X	X	X	X	X	X	X	X	X
	pused	X	X	X	X	X	X	X	X	X	X	X
vm.memory.size[<mode>]		X	X	X	X	X	X	X	X	X	X	X

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

<note 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。

这些值不能当做准确值. 它们是临时的，Zabbix agent 不做任何的猜测工作, 但是 net-snmp 将返回这些字段的值。

[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。

[4] 不支持 Windows 事件日志。

[5] 在 Windows XP vfs.file.time[file,change] 等于 vfs.file.time[file,access].

[6] Zabbix 5.0.5 开始只支持 Zabbix agent 2 不支持 Zabbix agent。

2 参数 **vm.memory.size**

概述

本节提供有关agent 监控项的 `vm.memory.size[<mode>]` 参数更多详细信息和特定于平台的信息.

参数

- **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。

Warning:
其中一些参数是特定于平台的，可能在您的平台上不可用。参考关于平台支持监控项详细信息。

特定平台的 **available** 和 **used** 的计算:

平台 *	"available"** *	"used"**
AIX	free + cached	real memory in use
FreeBSD	inactive + cached + free	active + wired + cached
HP UX	free	total - free
Linux<3.14	free + buffers + cached	total - free
Linux 3.14+ (也支持 RHEL 7 的 3.10 内核) Note	/proc/meminfo, 参考"MemAvailable" 在 Linux kernel 详细文档。 total -hat free + buffers + cached 不等于'available' 由于不是所有页面缓存都可以释放，计算中使用的水位很低。	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:
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，监控项 *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。

另见

1. 关于不同操作系统内存计算的详细信息

3 zabbix agent 的被动和主动检查

概述

本节提供关于 Zabbix agent 被动和主动执行检查的详细信息。

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

这里有一些 Zabbix 使用的协议细节中的使用到的定义:

<HEADER> - "ZBXD\x01" (5 bytes)
<DATALEN> - data length (8 bytes). 1 will be formatted as 01/00/00/00/00/00/00/00 (eight bytes in HEX, 64

为了避免耗尽内存，当 Zabbix server 使用 Zabbix protocol 协议时一次连接只接受 128M。

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

<HEADER><DATALEN><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 **<HEADER><DATALEN>agent.ping**
3. Agent 读取请求并响应 **<HEADER><DATALEN>1**
4. Server 处理数据以获取值, 例如'1'
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 端检索独立处理监控项的列表。

The servers 主动检查的列表在 agent 配置文件中的'ServerActive' 参数中列出，请求这些检查的频率是由相同配置文件中的'RefreshActiveChecks' 参数设置的。然而，如果刷新主动检查失败，则在 60 秒后重试。

agent 然后定期向服务器发送新值。

获取监控项列表

Agent 请求

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

Server 响应

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

```
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        }
    ]
}
```

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，对于可以访问 Zabbix 服务器 trapper 端口的配置数据是可得到的。这是可能的，因为任何一个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

```
<HEADER><DATALEN>{
    "request":"agent data",
    "data":[
        {
            "host":"<hostname>",
            "key":"agent.version",
            "value":"2.4.0",
            "clock":1400675595,
            "ns":76808644
        },
        {
            "host":"<hostname>",
            "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
            "lastlogsize":112,
            "value":" 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision
            "clock":1400675595,
            "ns":77053975
        },
        {
            "host":"<hostname>",
            "key":"vfs.fs.size[/nono]",
            "state":1,
            "value":"Cannot obtain filesystem information: [2] No such file or directory",
            "clock":1400675595,
            "ns":78154128
        }
    ],
    "clock": 1400675595,
    "ns": 78211329
}
```

Server 响应

```
<HEADER><DATALEN>{
    "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，上面例子中不支持 vfs.fs.size[/nono] 的状态由”state” 值为 1 和”value” 中的错误消息表示。

<note important> 在服务器端，错误消息将被处理到 2048 个符号。:::

老的 XML 协议

> **Note:**
> Zabbix 将占用 16 MB 的 XML base64 编码的数据, 单个解码值不应该超过 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，在捕捉器的监控项的帮助下接收到 Zabbix 发送的数据。

请求和响应消息必须以header 和 data length开头。

Zabbix 发送请求

```
{
    "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 发送者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 Windows 代理监控项的最小权限级别

总览

当使用代理监控系统时，一种最佳实践是从安装代理的主机上获取指标。要使用最小特权原则，那么必须确定从代理获得了哪些指标。

本文档中的表格允许您选择最小权限，以确保 Zabbix agent 正确运行。

如果选择了其他用户才能使代理工作，而不是选择 “ LocalSystem”，则要使代理作为 Windows 服务运行，新用户必须具有 “本地策略 → 用户权限” 中 “作为服务登录” 的权限，以及创建，写入和删除 Zabbix agent 日志文件的权限。

Note:
当基于 “技术上可接受的最低要求” 组使用代理的权限时，需要事先为监控对象提供权限。

Windows 支持的常见代理监控项

监控项键用户组		
	推荐的最低	术要求（功能有限）
agent.hostname	来宾来	
agent.ping	来宾来	
agent.version	来宾来	
log	管理员来宾	
log.count	管理员来宾	
logrt	管理员来宾	
logrt.count	管理员来宾	
net.dns	来宾来	
net.dns.record	来宾来	
net.if.discovery	来宾来	
net.if.in	来宾来	
net.if.out	来宾来	
net.if.total	来宾来	
net.tcp.listen	来宾来	
net.tcp.port	来宾来	
net.tcp.service	来宾来	
net.tcp.service.perf	来宾来	
net.udp.service	来宾来	
net.udp.service.perf	来宾来	
proc.num	管理员来宾	
system.cpu.discovery	性能监视器用户性能监视器用	
system.cpu.load	性能监视器用户性能监视器用	
system.cpu.num	来宾来	
system.cpu.util	性能监视器用户性能监视器用	
system.hostname	来宾来	
system.localtime	来宾来	
system.run	管理员来宾	
system.sw.arch	来宾来	
system.swap.size	来宾来	
system.uname	来宾来	
system.uptime	性能监视器用户性能监视器用	
vfs.dir.count	管理员来宾	
vfs.dir.size	管理员来宾	
vfs.file.cksum	管理员来宾	
vfs.file.contents	管理员来宾	
vfs.file.exists	管理员来宾	
vfs.file.md5sum	管理员来宾	
vfs.file.regexp	管理员来宾	
vfs.file.regmatch	管理员来宾	
vfs.file.size	管理员来宾	
vfs.file.time	管理员来宾	
vfs.fs.discovery	管理员来宾	
vfs.fs.size	管理员来宾	
vm.memory.size	来宾来	
web.page.get	来宾来	

监控项键用户组	
web.page.perf	来宾来
web.page.regexp	来宾来
zabbix.stats	来宾来

Windows 特定监控项键

监控项键用户组		
	推荐的最低	术要求（功能有限）
eventlog	事件日志阅读器来宾	
net.if.list	来宾来	
perf_counter	性能监视器用户性能监视器用	
proc_info	管理员来宾	
service.discovery	来宾来	
service.info	来宾来	
services	来宾来	
wmi.get	管理员来宾	
vm.vmemory.size	来宾来	

6 返回值的编码

Zabbix server 期望每个返回的文本值都是 UTF8 编码的，这涉及所有的检查类型: 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 都包含一个额外的监控项参数 **<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*.
MySQL 5.1.61 版本及相关版本。

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.
PostgreSQL 9.1.3 版本及相关版本。

7 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。从 Zabbix 2.0 开始支持大文件。该变动会影响日志文件的监控 和所有vfs.file.* 监控项. 大文件的支持依赖于 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。

8 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。

9 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	

Notes for AIX:

1. When choosing parameters for proc.mem[] item key on AIX, try to specify narrow process selection criteria. Otherwise there is a risk of getting unwanted processes counted into proc.mem[] result.

Example:

```
\$ zabbix_agentd -t proc.mem[,,,NonExistingProcess,rss]
proc.mem[,,,NonExistingProcess,rss]           [u|2879488]
```

This example shows how specifying only command line (regular expression to match) parameter results in Zabbix agent self-accounting - probably not what you want.

2. Do not use "ps -ef" to browse processes - it shows only non-kernel processes. Use "ps -Af" to see all processes which will be seen by Zabbix agent.
3. Let's go through example of 'topasrec' how Zabbix agent proc.mem[] selects processes.

```
\$ ps -Af | grep topasrec
root 10747984        1   0   Mar 16      -  0:00 /usr/bin/topasrec  -L -s 300 -R 1 -r 6 -o /var/perf daily
```

proc.mem[] has arguments:

```
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]
```

The 1st criterion is a process name (argument <name>). In our example Zabbix agent will see it as 'topasrec'. In order to match, you need to either specify 'topasrec' or to leave it empty. The 2nd criterion is a user name (argument <user>). To match, you need to either specify 'root' or to leave it empty. The 3rd criterion used in process selection is an argument <cmd-line>. Zabbix agent will see its value as '/usr/bin/topasrec -L -s 300 -R 1 -r 6 -o /var/perf/daily/ -ypersistent=1 -O type=bin -ystart_time=04:08:54,Mar16,2023'. To match, you need to either specify a regular expression which matches this string or to leave it empty.

[1] 当 HttpOnly 设定为'true' 时，将只能通过 HTTP 协议访问 cookie。这意味着无法通过脚本语言（例如 JavaScript）访问 Cookie。此设置可以有效地帮助减少通过 XSS 攻击进行的身份盗用（尽管并非所有浏览器都支持此功能）。^ Secure((Secure 表示应仅通过从来自客户端的安全的 HTTPS 连接传输 cookie。设置为'true' 时，仅当存在安全连接时才设置 cookie。

Arguments <mode> and <memtype> are applied after using the three criteria mentioned above.

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

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

Footnotes

[1] Default value.

10 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。

Linux kernel threads

11 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。

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
```

Unavailable host

After the UnreachablePeriod ends and the host has not reappeared, the host is treated as unavailable.

In the server log it is indicated by messages like these:

```
temporarily disabling Zabbix agent checks on host "New host": host unavailable
```

and in the frontend the host availability icon for the respective interface goes from green (or gray) to red (note that on mouseover a tooltip with the error description is displayed):

The **UnavailableDelay** parameter defines how often a host is checked during host unavailability.

By default it is 60 seconds (so in this case "temporarily disabling", from the log message above, will mean disabling checks for one minute).

When the connection to the host is restored, the monitoring returns to normal automatically, too:

```
enabling Zabbix agent checks on host "New host": host became available
```

13 远程监控 **Zabbix** 状态

概述

可以通过另一个 Zabbix 实例或第三方工具远程访问 Zabbix 服务器和代理的一些内部指标。这可能很有用，以便支持者/服务提供商可以远程监控其客户端 Zabbix 服务器/代理，或者在 Zabbix 不是主要监控工具的组织中，Zabbix 内部指标可以由第三方系统在设置监控。

Zabbix 内部统计信息暴露于新的 "StatsAllowedIP" 中列出的一组可配置地址server/proxy 参数。只接受来自这些地址的请求。

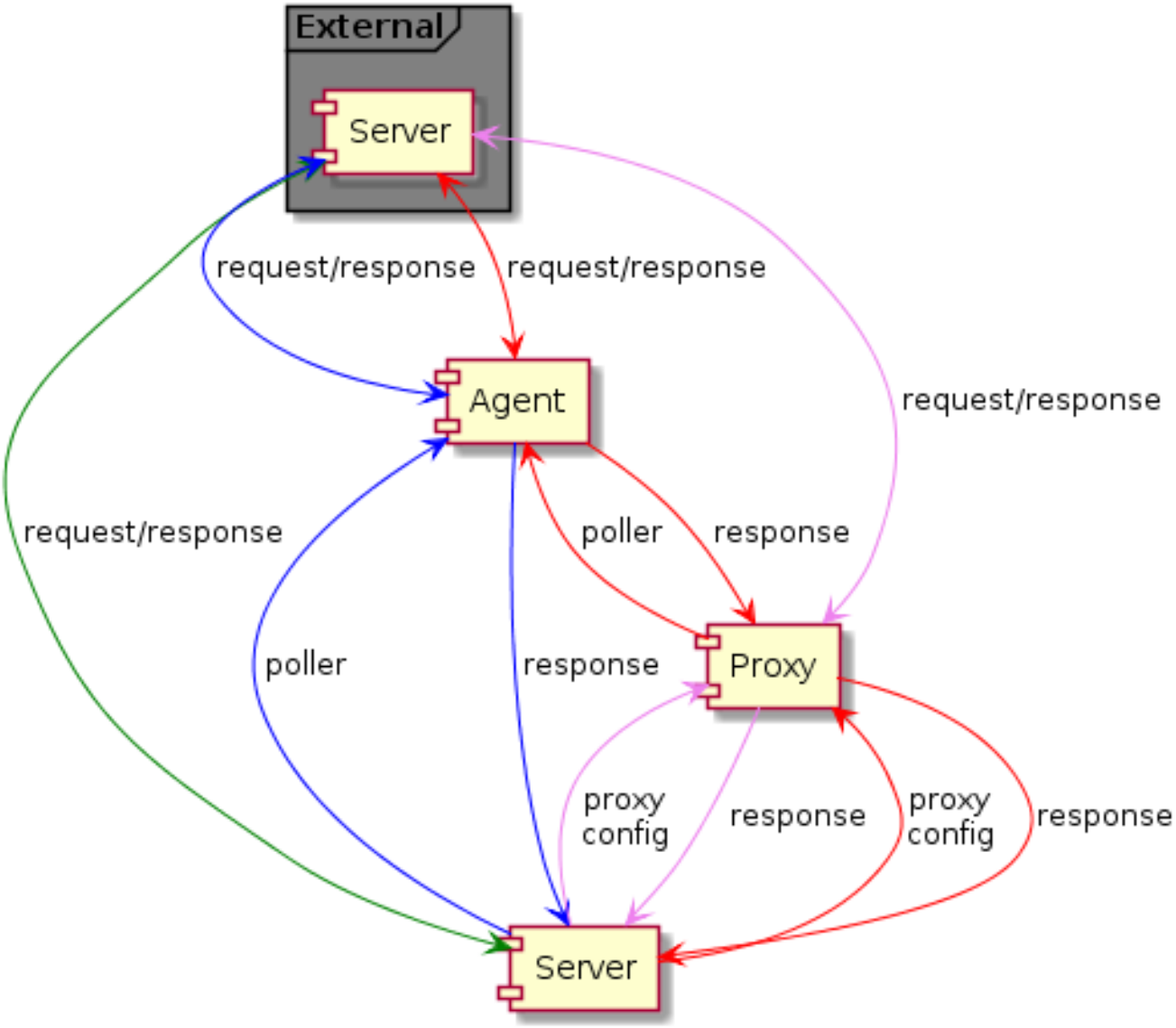
监控项

要在另一个 Zabbix 实例上配置内部统计信息的查询，可以使用两项：

- `zabbix[stats,<ip>,<port>]` 内部监控项 - 用于直接远程查询 Zabbix 服务器/代理。<ip> 和 <port> 用于标识目标实例。
- `zabbix.stats[<ip>,<port>]` Agent 监控项 - 用于基于代理的 Zabbix 服务器/代理的远程查询。<ip> 和 <port> 用于标识目标实例。

另请参见: Internal items, Zabbix agent items

下图根据上下文说明了这两个项的用法。



- - Server → 外部 zabbix 实例 (`zabbix[stats,<ip>,<port>]`)
- - Server → proxy → 外部 zabbix 实例 (`zabbix[stats,<ip>,<port>]`)
- - Server → agent → 外部 zabbix 实例 (`zabbix.stats[<ip>,<port>]`)
- - Server → proxy → agent → 外部 zabbix 实例 (`zabbix.stats[<ip>,<port>]`)

要确保目标实例允许外部实例查询它，请在目标实例的 “StatsAllowedIP” 参数中列出外部实例的地址。

内部指标

状态监控项收集统计信息后返回一个 JSON，这是其他依赖监控项从中获取数据的基础。以下内部指标的用法：

- `zabbix[boottime]`
- `zabbix[hosts]`
- `zabbix[items]`
- `zabbix[items_unsupported]`
- `zabbix[preprocessing_queue]` (server only)
- `zabbix[process,<type>,<mode>,<state>]` (only process type based statistics)
- `zabbix[rcache,<cache>,<mode>]`
- `zabbix[requiredperformance]`
- `zabbix[triggers]` (server only)
- `zabbix[uptime]`
- `zabbix[vcache,buffer,<mode>]` (server only)
- `zabbix[vcache,cache,<parameter>]`
- `zabbix[version]`
- `zabbix[vmware,buffer,<mode>]`

- zabbix[wcache,<cache>,<mode>] ('trends' cache type server only)

模板

Zabbix server 和 Zabbix proxy 远程监控模板:

- Template App Remote Zabbix server
- Template App Remote Zabbix proxy

请注意，为了使用模板远程监视多个外部实例，每个外部实例监视都需要一个单独的主机。

捕捉器执行过程

Zabbix 实例接收内部指标请求由 trapper 进程处理，trapper 进程验证请求、收集、创建 JSON 数据缓冲区并将准备好的 JSON 发回，例如从服务器:

```
{
  "response": "success",
  "data": {
    "boottime": N,
    "uptime": N,
    "hosts": N,
    "items": N,
    "items_unsupported": N,
    "preprocessing_queue": N,
    "process": {
      "alert manager": {
        "busy": {
          "avg": N,
          "max": N,
          "min": N
        },
        "idle": {
          "avg": N,
          "max": N,
          "min": N
        },
        "count": N
      },
...
    },
    "queue": N,
    "rcache": {
      "total": N,
      "free": N,
      "pfree": N,
      "used": N,
      "pused": N
    },
    "requiredperformance": N,
    "triggers": N,
    "uptime": N,
    "vcache": {
      "buffer": {
        "total": N,
        "free": N,
        "pfree": N,
        "used": N,
        "pused": N
      },
      "cache": {
        "requests": N,
        "hits": N,
        "misses": N,
        "mode": N
      }
```

```
    },
    "vmware": {
      "total": N,
      "free": N,
      "pfree": N,
      "used": N,
      "pused": N
    },
    "version": "N",
    "wcache": {
      "values": {
        "all": N,
        "float": N,
        "uint": N,
        "str": N,
        "log": N,
        "text": N,
        "not supported": N
      },
      "history": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      },
      "index": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      },
      "trend": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      }
    }
  }
}
```

内部监控项

另外还有两专门允许个监控项可以远程查询另一个 Zabbix 实例上的内部队列统计信息:

- `zabbix[stats,<ip>,<port>,queue,<from>,<to>]` 内部监控项 - 用于将内部队列查询直接发送到 Zabbix server/proxy
- `zabbix.stats[<ip>,<port>,queue,<from>,<to>]` agent 监控项 - 用于将内部队列查询直接发送到 Zabbix server/proxy

参考: 内部监控项, Zabbix agent 监控项

14 使用 **Zabbix** 配置 **Kerberos** 监控

概述

zabbix 4.4.0 之后版本，Kerberos 身份验证可用于 Zabbix 中的 web 监视和 HTTP 项.

这部分 Kerberos 配置描述 Zabbix server 使用 zabbix 用户对 `www.example.com` 进行 web 监控

步骤

第 1 步

安装 Kerberos 包。

For Debian/Ubuntu:

```
apt install krb5-user
```

For RHEL/CentOS:

```
yum install krb5-workstation
```

第 2 步

C 设置 Kerberos 配置文件 (看 MIT 详细文档)

```
cat /etc/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM

#### The following krb5.conf variables are only for MIT Kerberos.
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

[realms]
    EXAMPLE.COM = {
    }

[domain_realm]
    .example.com=EXAMPLE.COM
    example.com=EXAMPLE.COM
```

第 3 步

创建 *zabbix* 用户的 Kerberos ticket。使用 *zabbix* 执行命令:

```
kinit zabbix
```

<note important> 要使用 *zabbix* 用户执行以上命令. 如果使用 *root* 用户将不会通过认证。:::

第 4 步

创建具有 Kerberos 身份验证类型的 web 方案或 HTTP 代理项。

可以选择使用以下 curl 命令进行测试:

```
curl -v --negotiate -u : http://example.com
```

请注意，对于冗长的 web 监视，有必要更新 Kerberos ticket。Kerberos ticket 到期时间默认为 10 小时。

6 触发器

6 Triggers

1 触发器函数

所有触发器表达式支持的函数都在下表中。

函数		
	描述信息 ** 参	** ** 备注 *
abschange		

函数			
	后一个值与前一个值变动的绝对值。	支持的值类型: float,	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
avg (sec|#num,<time_shift>)	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>)			

函数			
		将 item 值与 mask 进行按位与操作。 **sec** (可	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。
change			

函数				
			最近获取值与之前获取值的差。	支持值类型: float,	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
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) <*br*> 支持的操作符*:* <*br*>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* 仅	<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; forecast(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 => forecast(1h,,2h,polynomial3,m → 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 prev() 函数更加精确, 特别是 item 很少更新的时候, 比如说, 一小时一次） 从 Zabbix 3.0.0 开始支

函数

Warning:
重要事项:
1) 部分函数不能用于非数值类型数据。
2) 字符型参数都应该使用双引号。否则，可能会被错误解析。
3) 所有 trigger 函数中的 **sec** 和 **time_shift** 参数都必须是带有可选时间单位后缀的整数。时间单位后缀与 item 的数据类型完全无关。

注脚

[1] 从第一个接收值开始计算函数 (除非使用 `time_shift` 参数).

函数与不被支持的监控项

Attention:
从 Zabbix 3.2 开始，**nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **now()** 和 **time()** 函数都支持用于不被支持的监控项。但其他函数都要求用于可支持的监控项。

7 宏

7 Macros

1 宏使用场景

概述

下表包含 Zabbix 支持宏的完整列表。

Note:
要查看某个位置所支持的所有宏 (例如，在"map URL") 中，您可以将位置名称粘贴到浏览器窗口底部的搜索框中 (通过按 CTRL+F 进行访问)，然后搜索 *next*.

宏	持场景描述信息	
{ACTION.ID}	基于 Trigger 的通知和命令 *action→ 发现通知 从 2.→ 自动注册通知 → 自动注册通知 → 故障更新通知	数字标识。*.0 开始支持。
{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。

宏	持场景描述信息	
{ALERT.MESSAGE}	→ 报警脚本参数 * 默认值由	ction配置。* 从3.0.0开始支持。
{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.SUBJECT}	→ 报警脚本参数 * 默认值由	ction配置。* 从3.0.0开始支持。
{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。
{DISCOVERY.DEVICE.IPADDRESS}	→ 发现通知 * 设备	设备的IP地址。* 不依赖于是否添加设备。

宏	持场景描述信息
{DISCOVERY.DEVICE.DNS}	发现通知中被发现设备的DNS名称。*不依赖于是否添加设备。
{DISCOVERY.DEVICE.STATUS}	发现通知中被发现设备的状态。*:可能是UP或DOWN.

宏	持场景描述信息
{DISCOVERY.RULE.NAME}	发现通知

定设备最近一次发现状态改变的时间。
例如:1h 29m.\\对于状态为DOWN的设备，这是其停机时间。|{DISCOVERY.RULE.NAME}→发现通知|发现设备或服务是否存在的发现规则名称。//

宏	持场景描述信息
{DISCOVERY.SERVICE.NAME}	服务的名称。*\\例如: HTTP。
{DISCOVERY.SERVICE.PORT}	服务的端口。*例如: 80。
{DISCOVERY.SERVICE.STATUS}	服务的状态。*可能是UP或DOWN。

宏	持场景描述信息
{DISCOVERY.SERVICE.UPTIME}	发现通知

定服务最近一次发现状态改变的时间。
例如:1h 29m.\\对于状态为DOWN的服务，这是其停服时间。|{ESC.HISTORY|→基于Trigger的通知和命令
→内部通知
→故障更新通知|记录以

宏	场景	描述	信息
{EVENT.ACK.STATUS}	基于 Trigger 的通知和命令	* 事件的确认状 → 故障更新通知	。
{EVENT.ACK.YES}	基于 Trigger 的通知和命令	* 触发动作的事 → 发现通知 → 内部通知 → 故障更新通知	(Yes/No)*. 持续时间。
{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	* 级的消息非常有用。 发生日期。
{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.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。

宏	持场景描述信息	
{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.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.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。

宏	持场景描述信息				
{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.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.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.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。

宏	持场景描述信息
{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.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.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.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。 *

宏	持场景描述信息
{EVENT.UPDATE.ACTION}	一.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。

宏	持场景描述信息
{EVENT.UPDATE}	→ 故障更新通知 * 故障更新
{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
{EVENT.UPDATE.MESSAGE}	→ 故障更新通知 * 故障更新
{EVENT.UPDATE.TIME}	→ 故障更新通知 * 故障更新
{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

间。(确认,等)。*取代以前的宏:{ACK.DATE}志。(确认,等).*:{EVENT.ACK.HIS
息。*取代以前的宏:{ACK.MESSAGE
间。(确认,等).*取代以前的宏:{ACK.TIME}类型(1为故障,0为恢复)。*.0开始支持。

宏	场景描述信息	
{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	S
	→ 全局脚本 (包括确认文本)	名
	→ 地图中的 Icon 标签[1]	称,
	→ Item key 值[2]	依
	→ 设备接口 IP/DNS	赖
	→ 数据库监控附加字段[5]	于
	→ SSH 和 Telnet 脚本[5]	设
	→ JMX item endpoint 字段	备
	→ Web 监控[6]	配
	→ Low-level 发现规则过滤正则表达式[8]	置。
	→ 动态 URL 仪表板小部件/屏幕元素的 URL 字段[8]	*[3].
	→ Trigger 名字和描述	.0
	→ Trigger URLs[10]	开
	→ 事件 tag 的名称和值	始
	→ HTTP agent 的 item 类型, item 原型和发现规则字段:	支
	URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	持。
{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	<.0
	→ 地图中的 Icon 标签[1]	开
		始
		支
		持。
{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	*[3].
	→ 全局脚本 (包括确认文本)	ger
	→ 地图中的 Icon 标签[1]	名
	→ Item key 值[2]	字
	→ 设备接口 IP/DNS	从
	→ 数据库监控附加字段[5]	2.0.0
	→ SSH 和 Telnet 脚本[5]	开
	→ JMX item endpoint 字段	始
	→ Web 监控[6]	支
	→ Low-level 发现规则过滤正则表达式[8]	持。
	→ 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.	
{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.HOST1<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]	ID。
	→ Trigger URLs[10]	*
	→ 事件 tag 的名称和值	

宏	持场景描述信息	
{HOST.IP<1-9>}	基于 Trigger 的通知和命令 → 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.	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	*[3]. 开始支持。 宏 {IPADDRESS< 已经不被支持。
{HOST.IP<1-9>}	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.NAME<1-9>}	基于 Trigger 的通知和命令 → 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.	* 用于显示的设 → 自动注册通知 从 2.0.→ 故障更新通知	名称 * 开始支持。

宏	持场景描述信息	
{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 trig-ger 名称和描述，内部通知，事件 tag 的名称和值。
{HOST.CONN<1-9>}	获取 Trigger URLs *	备组标识。
{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.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。

宏	持场景描述信息	
{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.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.CONTRACTNUMBER<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.DEPLOYMENTSTATUS<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.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.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.HOSTNETMASK<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.HOSTNETWORKS<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。*

宏	持场景描述信息	
{INVENTORY.HOSTROUTER<1 9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的路由字段。*
{INVENTORY.HWARCH<1 9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的硬件架构字段。*
{INVENTORY.HWDATEDOWN<1 9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 主机清 → 内部通知 中的硬件下线日期字段。*
{INVENTORY.HWDATEJUSTRY<1 9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的保修期字段。*
{INVENTORY.HWDATEJUSTALL<1 9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的硬件上线日期字段。*
{INVENTORY.HWDATEJUSTCHASE<1 9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中硬件购买时间字段。*

宏	持场景描述信息	
{INVENTORY.TRIGGER.ERROR.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.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.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.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.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.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.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。*

宏	持场景描述信息	
{INVENTORY.TAGGER.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.NA 已经不被支持。
{INVENTORY.TOTES<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.NO 已经不被支持。
{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.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.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.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。	

宏	持场景描述信息	
{INVENTORY.TS.FULL.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.TS.SHORT.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.POC.PRIMARY.CELL.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.EMAIL.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.NAME.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.NOTE.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。*

宏	持场景	描述信息
{INVENTORY.POC.PRIMARY.PHONE.A<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	*设备清单中的主要 POC 联系电话字段。*
{INVENTORY.POC.PRIMARY.PHONE.B<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	*设备清单中的主要 POC 联系电话字段。*
{INVENTORY.POC.PRIMARY.SCREEN<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	*设备清单中的主要 POC screen 名称字段。*
{INVENTORY.POC.SECONDARY.CELL<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	*设备清单中的辅助 POC cell 字段。*
{INVENTORY.POC.SECONDARY.EMAIL<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	*设备清单中的辅助 POC 电子邮件字段。*

宏	持场景描述信息	
{INVENTORY.TOGGLE.NOTE.A<1-9>}	→ 基于 Toggle 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.TOGGLE.NOTE.B<1-9>}	→ 基于 Toggle 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.TOGGLE.NOTE.C<1-9>}	→ 基于 Toggle 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.TOGGLE.NOTE.D<1-9>}	→ 基于 Toggle 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.TOGGLE.NOTE.N<1-9>}	→ 基于 Toggle 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.SERIALNO.A<1-9>}	→ 基于 Toggle 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。	

宏	持场景	描述信息
{INVENTORY.SERIALNO.B<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	*设备清中的序列号字段。*
{INVENTORY.SITE_ADDRESS_A<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	*设备清中的站点地址字段。*
{INVENTORY.SITE_ADDRESS_B<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	*设备清中的站点地址字段。*
{INVENTORY.SITE_ADDRESS_C<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	*设备清中的站点地址字段。*
{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_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_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。*

宏	持场景	描述信息
{INVENTORY.SITE.RACK<1-9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的站点机架位置字段。*
{INVENTORY.SITE.STATE<1-9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中站点所属州/省字段。*
{INVENTORY.SITE.ZIP<1-9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的站点邮编字段。*
{INVENTORY.SOFTWARE<1-9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的软件描述字段。* 宏 {PROFILE.SO 已经不被支持。
{INVENTORY.SOFTWARE.APP.A<1-9>}	→ 基于 Trigger 的通知 → 故障更新通知 → 事件 tag 的名称和值	* 设备清 → 内部通知 中的应用软件字段。*

宏	持场景描述信息	
{INVENTORY.SOFTWARE.APP.B 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 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 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 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.FULL 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.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已经不被支持。

宏	持场景描述信息	
{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.DE 已经不被支持。
{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.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.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.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.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。*

宏	持场景描述信息	
{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.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.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。	

宏	持场景描述信息	
{ITEM.KEY<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。

宏	持场景描述信息	
{ITEM.LASTVALUE<1 9>}	基于Trigger1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。 * 历史值采集时间已经超过参数ZBX_HISTORY_定义的历史数据保存时间，那么在前端会显示值为*UN-KNOWN*。(参数ZBX_HISTORY_定义于defines.inc.p3

宏	持场景描述信息	
{ITEM.LOG.AGE 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.DATE 9>}	基于 Trigger 的通知 * 日志 i→ 故障更新通知	em 事 件 的 发 生 时 间。 *
{ITEM.LOG.EVENT.ID 9>}	基于 Windows 事件日志监控的通知 * 日志事 → 故障更新通知仅用于 Wi	的 标 识。 * dows 事 件 日 志 监 控。
{ITEM.LOG.NSEVERITY 9>}	基于 Windows 事件日志监控的通知 * 日志事 → 故障更新通知仅用于 Wi	的 级 别。 * dows 事 件 日 志 监 控。
{ITEM.LOG.SEVERITY 9>}	基于 Windows 事件日志监控的通知 * 日志事 → 故障更新通知仅用于 Wi	的 级 别。 * dows 事 件 日 志 监 控。

宏	持场景描述信息	
{ITEM.LOG.SOURCE<1-9>}	基于 Trigger 的通知 * 日志事 → 故障更新通知仅用于 Wi	的来源。
{ITEM.LOG.TIME<1-9>}	基于 Trigger 的通知 * 日志 i→ 故障更新通知	* dows 事件日志监控。 em 事件的发生时间。
{ITEM.NAME<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 的名称。 *

宏	持场景描述信息	
{ITEM.NAME.ORIGIN<1 9>}	基于 Trigger 的通知 * 触发器 → 内部通知 从 2.→ 故障更新通知	达式中导致发送通知的第 N 个 item 的原始名称。* .6 开始支持。
{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。

宏	持场景描述信息
{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
{ITEM.VALUE}	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

宏	持场景描述信息
{LLDRULE.DESCRIPTION}	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.ID}	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.KEY}	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。

宏	持场景描述信息
{LLDRULE.KEY.ORIG}	ed 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.NAME}	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。

宏	持场景描述信息
{LLDRULE.NAME.ORIG}	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.STATE}	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。
{MAP.ID}	地图 URLs *	络地图标识。*

宏	持场景描述信息	
{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。

宏	持场景描述信息
{PROXY.DESCRIPTION<1-9>}	基于Trigger的通知和命令 *proxy 描 → 故障更新通知 1) 触发器 → 发现通知 2)→ 自动注册通知 3) 主动 → 内部通知 从 2.

信息。
*
可
能
的
值:
达
式
中
第
N
个
项
的
proxy
的
信
息
(基
于
Trig-
ger
的
通
知)。
可
以
使
用宏
索
引。
行
发
现
的
proxy
信
息
(发
现
通
知)。
可
以
使
用
宏
{PROXY.DESCRI
而
不
带
宏
索
引。
gent
注
册
的
proxy
信
息。
(自
动
注

宏	持场景描述信息	
{PROXY.NAME<1-9>}	基于 Trigger 的通知和命令 *proxy 的 → 故障更新通知 1) 触发 → 发现通知 2)→ 自动注册通知 3) 主动 → 内部通知从 1.	称 *。可能的值: 表达式中第 N 个项的 proxy 的名称 (基于 Trig- ger 的通知)。可以使用宏索引。 行发现的 proxy 名称 (发现通知)。可以使用宏 {PROXY.NAME}，而不带宏索引。 gent 注册的 proxy 名称。 (自

宏	持场景描述信息	
{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.*
{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.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。

宏	持场景	描述信息
{TRIGGER.EVENTS.PROBLEM.ACK}	→ 基于 trigger 的通知 → 故障更新通知 → 地图中的 Icon 标签[1]	* 忽略状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.UNACK}	→ 基于 trigger 的通知 → 故障更新通知 → 地图中的 Icon 标签[1]	* 忽略状的所有触发器的未确认故障事件数。 * 从 1.8.3 开始使用。

宏	持场景描述信息	
{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。

宏	持场景描述信息	
{TRIGGER.HOSTGROUP.NAME}	→ 基于 Trigger 的通知 → 基于 Trigger 内部通知	L 查询排序，逗号-空格分隔的 trig-ger 所属的设备组列表。 * 从 2.0.6 开始支持。
{TRIGGER.PROBLEM.ACK}	→ 基于 S→ 故障更新通知 → 基于 Trigger 内部通知	状态为问题的已确认问题事件数。 * 从 1.8.3 开始支持。

宏	持场景描述信息	
{TRIGGER.PROBLEM.UNACK}	→ 地图 Cell 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.EXPRESSION}	→ 基于 Trigger 内通知 *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。

宏	持场景描述信息	
{TRIGGER.EXPRESSION.RECOVERY}	基于 Trigger 的通知 * 触发动 → Trigger-based 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.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。

宏	持场景描述信息	
{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.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。

宏	持场景描述信息
{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。

宏	持场景描述信息	
{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.STATUS}	→基于 Trigger 的通知 *trig	er 的最新状态。* 可能的值: **Unknown** and **Normal**. 从 2.2.0 开始支持。
{TRIGGER.VALUE}	→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。	

宏	持场景描述信息
{TRIGGER.TEMPLATE.NAME}	* 排序（→ 基于 Trigger 内部通知 → 故障更新通知
{TRIGGER.TRIGGER.NAME}	* 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*UN-KNOWN**从2.0.6开始支持。

ger URL.*

宏	持场景描述信息	
{TRIGGER.VALUE}	基于 Trigger 的通知 → 故障更新通知	* 触发器 → Trigger 表达式	当前值。*: 0 - trigger 状态为 OK, 1 - trigger 状态为 PROBLEM。

宏	持场景描述信息	
{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。

宏	持场景描述信息
{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。

宏	持场景描述信息
{TRIGGER.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。

宏	持场景描述信息
{TRIGGER.PROBLEM.ACK}	地图PROBLEM[1] * 地图
{USER.FULLNAME}	故障更新通知 * 事件确认

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。

作的用户全名。* 从3.4.0开始支持。

宏	持场景描述信息	
{USER.FULLNAME}	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/ma 宏。 *
{#MACRO}	参考: Low-level 发现宏 *Low	level 发现宏 * 从 2.0.0 开始支持。

脚注

[1] 从 1.8 开始地图标签支持宏。

[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。

[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。

[4] 地图标签中的宏仅仅支持 **avg**, **last**, **max** and **min** 函数, 以秒为单位。

[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。

[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。

[10] 从 3.0.0 开始支持。

宏索引

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。

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。

Warning:
有些场景可以使用不带索引的宏。(例如：{HOST.HOST}, {HOST.IP}, 等)

2 用户自定义宏使用场景

概述

用户自定义宏可以用于以下场景。

动作

在动作中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
基于触发器的通知和命令	yes	
基于触发器的内部通知	yes	
问题更新通知	yes	
时间段条件	no	
操作		
	默认操作步骤持续时间 no	
	步骤持续时间 no	

主机/主机原型

在主机和主机原型配置中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
接口 IP/DNS	只	许 DNS
端口	n	
SNMP v1, v2		
	SNMP 团体名 ye	
SNMP v3		
	Context name	yes
	Security name	yes
	Authentication passphrase	yes
	Privacy passphrase	yes
IPMI		
	用户名 ye	
	密码 y	s
//标签 //		
	标签名字 yes	

位置	多	宏/与文本混合[1]
	标签值 ye	

监控项/监控项原型

在监控项或者监控项原型 配置中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
名字 (已弃用)	yes	
监控项键值参数	yes	
更新间隔	no	
自定义间隔	no	
历史存储周期	no	
趋势存储周期	no	
描述	y	s
//计算监控项 //		
	公式 y	s
数据库监控		
	用户名 ye	
	密码 y	s
	SQL 语句 y	s
//HTTP 客户端 //		
	URL[2]	yes
	查询字段 yes	
	超时时间 no	
	请求体 ye	
	请求头部 (名字和值) yes	
	请求状态码 yes	
	HTTP 代理 y	s
	HTTP 认证用户名 yes	
	HTTP 认证密码 yes	
	SSI 证书文件 yes	
	SSI key 文件 y	s
	SSI key 密码 y	s
	允许请求的主机 yes	
JMX 客户端		
	JMX endpoint	yes
//Script 监控项 //		
	参数的名字和值 yes	
//SNMP 客户端 //		
	SNMP OID	yes
//SSH 客户端 //		
	用户名 ye	
	公钥文件 yes	
	私钥文件 yes	
	密码 y	s
	脚本 y	s
//TELNET 客户端 //		
	用户名 ye	
	密码 y	s
	脚本 y	s
//Zabbix 采集器 //		
	允许的主机 yes	
预处理		
	预处理步骤 (包含自定义脚本) yes	

低级别发现

在低级别发现规则中中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
名字	y	s
键值参数	yes	
更新间隔	no	
自定义间隔	no	
保留丢失的资源期限	no	
描述	y	s
SNMP 客户端		
	SNMP OID	yes
SSH 客户端		
	用户名 ye	
	公钥文件 yes	
	私钥文件 yes	
	密码 y	s
	脚本 y	s
TELNET 客户端		
	用户名 ye	
	密码 y	s
	脚本 y	s
Zabbix 采集器		
	允许的主机 yes	
数据库监控		
	附加参数 yes	
JMX 客户端		
	JMX endpoint	yes
HTTP 客户端		
	URL[2]	yes
	查询字段 yes	
	超时时间 no	
	请求体 ye	
	请求头部 (名字和值) yes	
	请求状态码 yes	
	HTTP 认证用户名 yes	
	HTTP 认证密码 yes	
//过滤器 //		
	正则表达式 yes	
//覆盖 //		
	过滤器: 正则表达式 yes	
	操作: 更新间隔 (对于监控项原型) no	
	操作: 历史存储周期 (对于监控项原型) no	
	操作: 趋势存储周期 (对于监控项原型) no	

网络自动发现

在网络自动发现规则中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
更新间隔 l	no	
SNMP v1, v2		
	SNMP 团体名 ye	
	SNMP OID	yes
SNMP v3		
	Context 名称 y	s
	Security 名称 y	s
	Authentication passphrase	yes
	Privacy passphrase	yes
	SNMP OID	yes

代理

在代理配置中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
接口的端口 (被动代理)	no	

模板

在模板配置中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
//标签 //		
	标签名字	yes
	标签值	ye

触发器

在触发器配置中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
名称	y	s
操作数据	yes	
表达式 (仅在常量和函数参数中; 不支持加密的宏).	yes	
描述	y	s
URL[2]		yes
匹配的标签	yes	
//标签 //		
	标签名字	yes
	标签值	ye

web 场景

在web 场景配置中, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
名字	y	s
更新间隔	no	
客户端	ye	
HTTP 代理	y	s
变量 (只允许值)	yes	
请求头部 (名字和值)	yes	
//步骤 //		
	名字	y s
	URL[2]	yes
	变量 (只允许值)	yes
	请求头部 (名字和值)	yes
	超时时间	no
	请求字符串	yes
	请求状态码	no
//安全认证 //		
	用户名	ye
	密码	y s
	SSL 证书	y s
	SSL key 文件	y s
	SSL key 密码	y s

其它

这里是附加的清单, 用户宏可用于以下字段:

位置	多	宏/与文本混合[1]
全局脚本 (包含配置文件中的文本)	yes	
监控 → 宏		
	URL[2] field of *dynamic URL* screen element	yes
管理 → 用户 → 媒体		
	动作 n	
管理 → 一般 → 图形		
	工作时间 no	
管理 → 媒体类型 → 信息模板		
	主题 y	s
	信息 y	s

查看 Zabbix 中支持的所有宏的完整列表, 参考支持宏。

注

[1] 如果该位置不支持字段中的多个宏或与文本混合的宏，则必须用单个宏填充整个字段。

[2] URLs 支持内部宏将不起作用, 因为它们中的宏将被解析为"******".

8 单位符号说明

概述

在使用一些大数字，例如'86400' 来表示一天中的秒数时，是非常容易出错的。这时就可以使用一些合适的单位符号（或后缀）来简化 Zabbix trigger 表达式和 item key。

你可以直接输入'1d'，而不是一天的秒数'86400'。后缀 d 用作乘数。

时间后缀

可使用的时间后缀:

- **s** - 秒（使用时，与原始值相同）
- **m** - 分
- **h** - 小时
- **d** - 天
- **w** - 周

以下支持时间后缀：

- 触发器expression 常量和函数参数
- 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）
- 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）
- 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）
- 网络发现规则配置（' 更新间隔' 字段）
- web scenario 配置（' 更新间隔'，' 超时' 字段）
- 动作操作配置（' 默认操作步骤持续时间'，' 步骤持续时间' 字段）
- 幻灯片展示配置（' 默认延迟' 字段）
- 用户基本资料配置（' 自动登录'，' 刷新'，' 消息超时' 字段）
- 管理 → 一般 → 管家 (' 存储期' 字段)
- 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)
- 管理 → 一般 → 其他 (' 刷新不支持的项目' 字段)
- 参数 **zabbix[queue,<from>,<to>]** internal item
- aggregate checks最后一个参数

内存后缀

触发器expression 常量和函数参数支持内存大小后缀。

可使用的内存大小后缀:

- **K** - 千字节
- **M** - 兆字节
- **G** - 十亿字节
- **T** - 兆兆字节

其他用法

单位符号还用于前端数据。

Zabbix server 和前端都支持这些符号：

- **K** - kilo
- **M** - mega
- **G** - giga
- **T** - tera

当监控项值 B，Bps 显示在前端时，应用基数 2（1K = 1024），或使用基数 10（1K = 1000）。

此外，前端还支持以下显示：

- **P** - peta
- **E** - exa
- **Z** - zetta
- **Y** - yotta

用法示例

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
```

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

概述

若要设定一个时间段，你会用到下面的格式：

```
d-d,hh:mm-hh:mm
```

符号用法：

符号描	
d	星期几: 1 - 星期一, 2 - 星期二,... , 7 - 星期天
hh	几时: 00-24
mm	几分: 00-59

可使用分隔符即分号（;）指定多个时间段：

```
d-d,hh:mm-hh:mm;d-d,hh:mm-hh:mm...
```

如果时间段的参数为空，系统将默认为 01-07,00:00-24:00

<note important> 不含时间段的上限是开区间的情况。当指定时间段为 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。:::

示例

工作日。星期一到星期五的 9:00 到 18:00:

```
1-5,09:00-18:00
```

工作日加周末。星期一到星期五的 9:00 到 18:00，以及周末的 10:00 到 16:00：

```
1-5,09:00-18:00;6-7,10:00-16:00
```

10 命令执行

Zabbix 常用功能包含外部检查、用户参数、system.run 监控项、自定义告警脚本、远程命令和用户命令。

执行步骤

命令/脚本在 Unix 和 Windows 系统平台上的执行方式相近:

1. Zabbix (父进程) 创建了一个交流通道。
2. Zabbix 将通道设置为要创建的子进程的输出接口。
3. Zabbix 创建子进程（运行命令/脚本）。
4. 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 从通道读取，直到超时或另一端没有其他写入（所有处理/文件描述符都已关闭）。请注意，子进程可创建更多进程并在退出或关闭处理/文件描述符之前退出。
6. 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. 如果初始子进程已退出且尚未超时，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 上执行的自定义告警脚本，远程命令和用户脚本）。
8. 此时，假设一切都已完成，整个过程 tree（即过程组或作业）终止。

<note important>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。:::

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。

退出代码的检查

使用以下条件检查退出代码：

- 仅适用于在 Zabbix server 和 Zabbix proxy 上执行的自定义告警脚本，远程命令和用户脚本。
- 任何不同于 0 的退出代码都被视为执行失败。
- 标准错误的内容和执行失败的标准输出会被收集并展示在前端（显示执行结果）。
- 为 Zabbix server 上的远程命令创建附加日志条目以保存脚本执行输出，可使用 LogRemoteCommands 代理parameter 。

前端可能出现的失败命令/脚本信息和日志条目：

- 执行失败的标准错误和标准输出的内容（如果有的话）。
- ” 进程退出代码：N.”（对于空输出，退出代码不等于 0）。
- ” 进程被信号终止：N.”（对于由信号终止的进程，仅在 Linux 上）。
- ” 进程意外终止。”（由于未知原因进程终止）。

了解更多:

- External checks
- User parameters
- system.run 监控项
- Custom alert scripts
- Remote commands
- Global scripts

See also

- External checks
- User parameters
- system.run items
- Custom alert scripts
- Remote commands
- Global scripts

13 版本兼容性

支持的 agents

从 1.4 版开始的 Zabbix agents 与 Zabbix 4.0 兼容。但是，你可能需要检查旧代理的配置，因为某些参数已更改。例如，3.0 之前的版本，与 logging 相关的参数：logging。

要充分利用新的和改进的项目，提高性能和减少内存使用，请使用最新的 4.0 agent。

支持的 Zabbix proxies

Zabbix 4.0 proxies 和 Zabbix 4.0 server 只分别支持 Zabbix 4.0 server 和 Zabbix 4.0 proxies 一起工作.

<note important> 众所周知，可以启动升级后的 server，使用尚未升级的 proxies 给新的 server 报告数据（proxies 无法刷新其配置）。但是，不推荐使用这种方法，Zabbix 不支持这种方法，选择它完全由你自己承担风险。更多详细说明，请查看升级步骤upgrade procedure. :::

支持的 XML 文件

在 Zabbix 4.0 中支持导入 1.8, 2.0, 2.2, 2.4, 3.0, 3.2 和 3.4 的 XML 文件.

Attention:
在 Zabbix 1.8 XML 导出格式中, 触发依赖仅由名称存储。如果有几个具有相同名称的 triggers（例如，具有不同的严重性和表达式），它们之间定义了依赖关系，则无法导入它们。必须从 XML 文件中手动删除这些依赖关系，并在导入后重新添加。

14 Python library for Zabbix API

Overview

[zabbix_utils](https://github.com/zabbix/python-zabbix-utils/blob/main/README.md) is a Python library for working with Zabbix API as well as with Zabbix sender and Zabbix get protocols.

It is supported for Zabbix 5.0, 6.0, 6.4 and later.

14 数据库错误处理

如果 Zabbix 检测到后端数据库不可访问，它将发送通知消息，并继续尝试连接到数据库。对于某些数据库引擎，会识别出特定的错误代码。

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 适用于 **Windows** 的 **Zabbix sender** 动态链接库

在 Windows 环境中，应用程序可以使用 Zabbix sender 动态链接库（zabbix_sender.dll）直接将数据发送到 Zabbix server/proxy，而不必启动外部进程（zabbix_sender.exe）。

带有开发文件的动态链接库位于 bin\winXX\dev 文件夹中。要使用它，请包含 zabbix_sender.h 头文件并链接到 zabbix_sender.lib 库。可以在 build\win32\examples\zabbix_sender 文件夹中找到具有 Zabbix 发送器 API 用法的示例文件。

Zabbix sender 动态链接库提供以下功能:

```
int zabbix_sender_send_values(const char *address, unsigned short port,const char *source, const zabbix_
char **result);‘{.c}
```

Zabbix sender 动态链接库使用以下数据结构:

```
typedef struct
{
    /* 主机名，必须与 Zabbix 中目标主机的名称匹配 */
    char    *host;
    /* item key */
    char    *key;
    /* item value */
    char    *value;
}
zabbix_sender_value_t;

typedef struct
{
    /* 处理数值的总数量 */
    int total;
    /* 失败值的数量 */
    int failed;
    /* server 处理发送值花的时间（以秒为单位） */
    double  time_spent;
}
zabbix_sender_info_t;
```

16 SELINUX 的问题

从 Zabbix 3.4 版本之后，加入了基于套接字的进程间通信。在启用了 SELinux 的系统上，可能需要添加 SELinux 规则，以允许 Zabbix 在 “SocketDir” 目录中创建或使用 Unix domain socket。当前套接字文件由 Server（报警器、预处理、IPMI）和 proxy（IPMI）使用。套接字文件是持续存在的，也就是说随进程的运行而存在。

17 其他问题

登录及系统守护进程

我们建议创建 zabbix 用户作为系统用户，也就是说，该用户不能登录到系统。一些用户忽略了这个建议，使用相同的帐户登录 (例如使用 SSH)，来管理运行 Zabbix。这可能会使 Zabbix 的守护进程在注销时崩溃。这种情况下，在 Zabbix server 的日志中会出现如下内容：

```
zabbix_server [27730]: [file:'selfmon.c',line:375] lock failed: [22] Invalid argument
zabbix_server [27716]: [file:'dbconfig.c',line:5266] lock failed: [22] Invalid argument
zabbix_server [27706]: [file:'log.c',line:238] lock failed: [22] Invalid argument
```

在 Zabbix agent 的日志中会出现：

```
zabbix_agentd [27796]: [file:'log.c',line:238] lock failed: [22] Invalid argument
```

这是由于在/etc/systemd/logind.conf 配置文件中默设置 RemoveIPC=yes 所致。当您退出系统时，Zabbix 先前创建的信号量将被删除，这将导致崩溃。以下内容摘自 systemd 文档：

```
RemoveIPC=

Controls whether System V and POSIX IPC objects belonging to the user shall be removed when the
user fully logs out. Takes a boolean argument. If enabled, the user may not consume IPC resources
after the last of the user's sessions terminated. This covers System V semaphores, shared memory
and message queues, as well as POSIX shared memory and message queues. Note that IPC objects of the
root user and other system users are excluded from the effect of this setting. Defaults to "yes".
```

此问题有两种解决方案：

1. (推荐) 停止使用 *zabbix* 帐户处理除 zabbix 进程以外的任何事情, 创建专有账户来处理其他事情。
2. (不推荐) 在/etc/systemd/logind.conf 配置文件中，设置 RemoveIPC=no ，并重启系统。需要注意的是，RemoveIPC 是系统范围的参数，其值更改后会影响整个系统。

在代理后面部署 zabbix 前端

如果 Zabbix 前端服务在代理服务器后面运行，则需要代理配置文件中的重定向 cookie 路径以匹配反向代理路径。请看下面的例子。如果不重定向 cookie 路径，用户在尝试登录 Zabbix 前端时可能会遇到授权问题。

NGINX 配置示例

```
# ..
location / {
# ..
proxy_cookie_path /zabbix /;
proxy_pass http://192.168.0.94/zabbix/;
# ..
```

APACHE 配置示例

```
# ..
ProxyPass "/" http://host/zabbix/
ProxyPassReverse "/" http://host/zabbix/
ProxyPassReverseCookiePath /zabbix /
ProxyPassReverseCookieDomain host zabbix.example.com
# ..
```

18 Agent 与 **agent2** 对比

这部分是 agent 与 agne2 对比的描述。

参数 Z	bbix agent Z	bbix agent 2
程序设计语言 C	一部分使用	，其他用 go
守护进程 yes	no	Windows 5.0.4 之后版本支持)
扩展支持自定义	的可加载模块。自定义 GO 的 [插件](/zh	manual/config/items/
请求支持平台 Lin	x, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, Mac OS X, Solaris: 9, 10, 11, Windows: 从 xp 开始所有的桌面和服务端版本。Linux, Windows: 从 x	开始所有的桌面和服务端版本。
支持的加密库 GnuTL	3.1.18 and newer LinuxOpenSSL 1.0.1, 1.0.2, 1.1.0, 1.1.1 SSL 库 - tested with versions 2.7.4, 2.8.2 (某些限制的使用, 查看加密详情页). OpenSSL 库必须开启 PS	OpenSSL 1.0.1 和最新版本在 Zabbix 4.4.8 之后支持。MS Windows: OpenSSL 1.1.1 或者最新版. ，否则 LibreSSL 不支持.
监控进程		

参数 Z	bbix agent Z	bbix agent 2
进程每	server/proxy 都有独立的进程。单个进程多线程。	< 这最大的线程数由 GO-MAX-PROCS 环境变量决定。
指标 *	UNIX**:** 查看支持的**items.** UNIX **Windows**: 查看指定 Windows 版本的监控项.	*: Zabbix agent 支持所有指标。其他的, agent2 提供 Docker, Mem-cached, MySQL, Post-greSQL, Redis, sys-temd (查看 agent2 的监控项) 的 Zabbix-native 监控方案 **Windows**: Zabbix agent 支持所有指标和 HTTPS 检查 net.tcp.service*, LDAP. 其他的, agent2 提供关于 Post-greSQL, Redis 的 Zabbix-native 监控方案。
并发单	程按监控项顺序进行检查来自不同插件的检查或一个插件	的多个检查可以同时执行。
计划/灵活间隔仅支持被动	查。支持主动检查。	

参数 Z	bbix agent Z	bbix agent 2
第三方 traps no Additional features	ye	
永久存储 no	yes	
超时设置只能定	agent 级别。超时插件可以覆盖在 a	ent 上的级别超时设置。
删除用户权限	nix-like systems only) no	
yes (		
用户可配置密码	no	
套件 yes		

参考**:**

- *Zabbix processes description*: Zabbix agent, Zabbix agent 2
- *Configuration parameters*: Zabbix agent UNIX / Windows, Zabbix agent 2 UNIX / Windows

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 5.0 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。

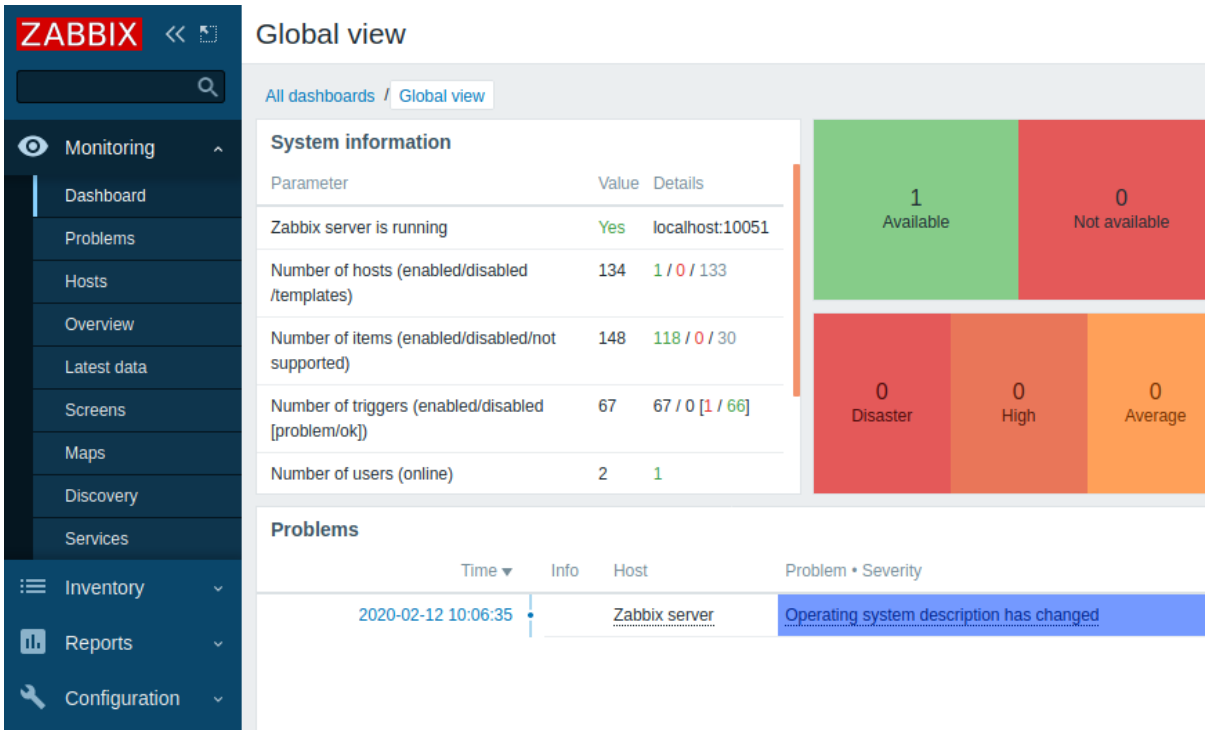
数据流

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。

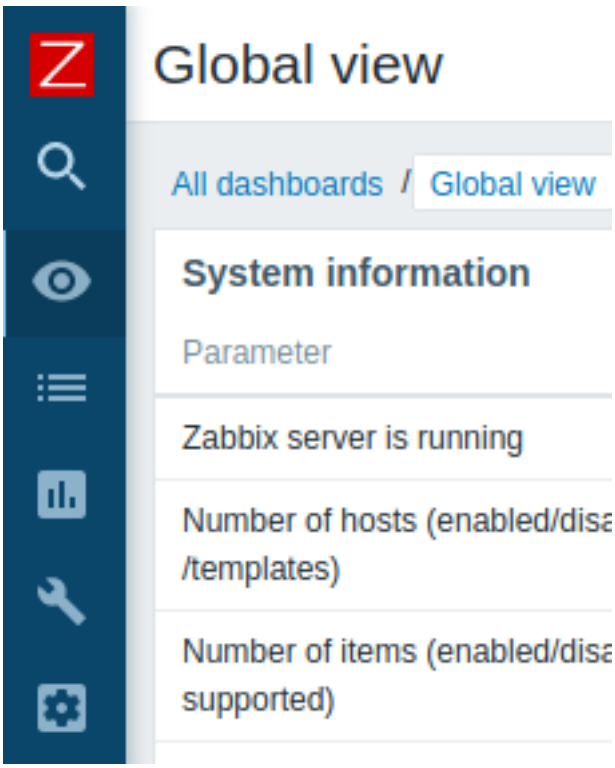
5 Zabbix 5.0.0 新功能特性

垂直菜单

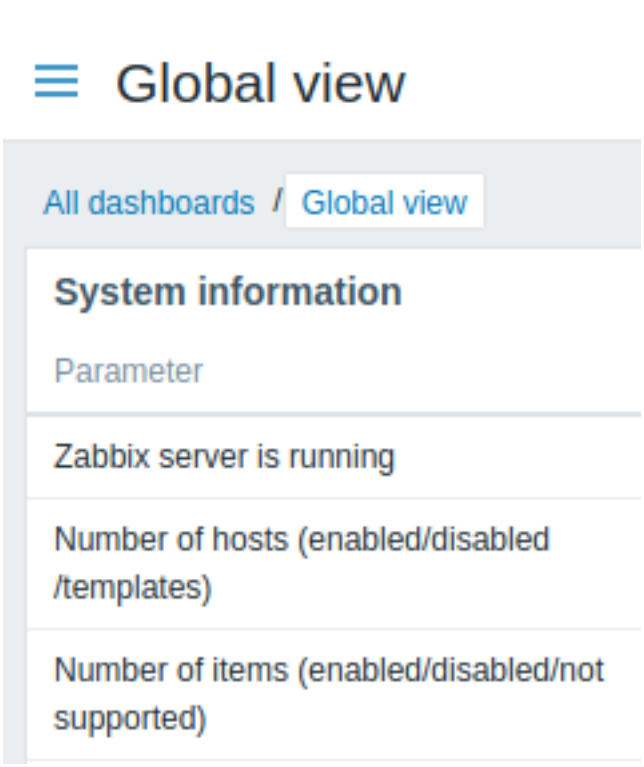
在新版本中，侧边栏中的垂直菜单取代了水平菜单。



菜单可以完全折叠或隐藏：



Collapsed menu with icons only.



Hidden menu.

菜单在折叠状态下，光标放在侧栏处即可完整展示。即使菜单细节被完全隐藏，点击鼠标即可查看完整的zh/manual/web_interface/menu#菜单。

可以进行字符串比较

可以在触发器表达式中使用 **=** (相等) 和 **<>** (不相等) 操作符对字符串进行比较.

因此，举例来说，就算两个监控项返回的字符串不同，在新版本中也是可以定义触发器表达式来创建告警的：

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

在可计算类型的监控项中也可以进行字符串比较。

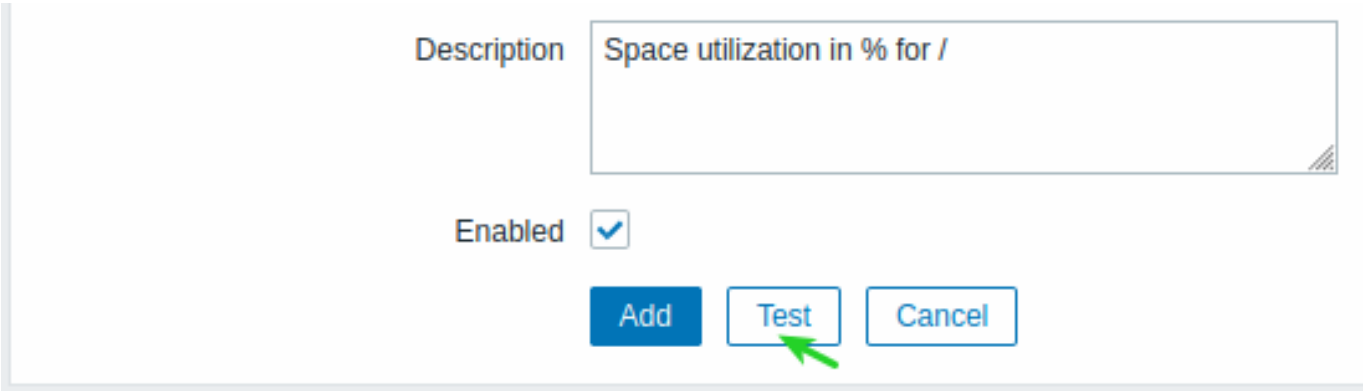
在配置页进行监控项测试

在老版本中，除非你等到监控项获取到数据，否则很难去判断新的监控项 是否配置正确。

新版本可以在保存之前从用户界面测试监控项 (模板监控项，监控项原型，自动发现规则)，如果配置正确，则返回一个正确值。

监控项测试不支持主动式监控项和一些简单检查类型的监控项（例如 `icmpping*`, `vmware.*`）。

测试监控项，单机监控项配置表单底部的 *Test* 按钮。

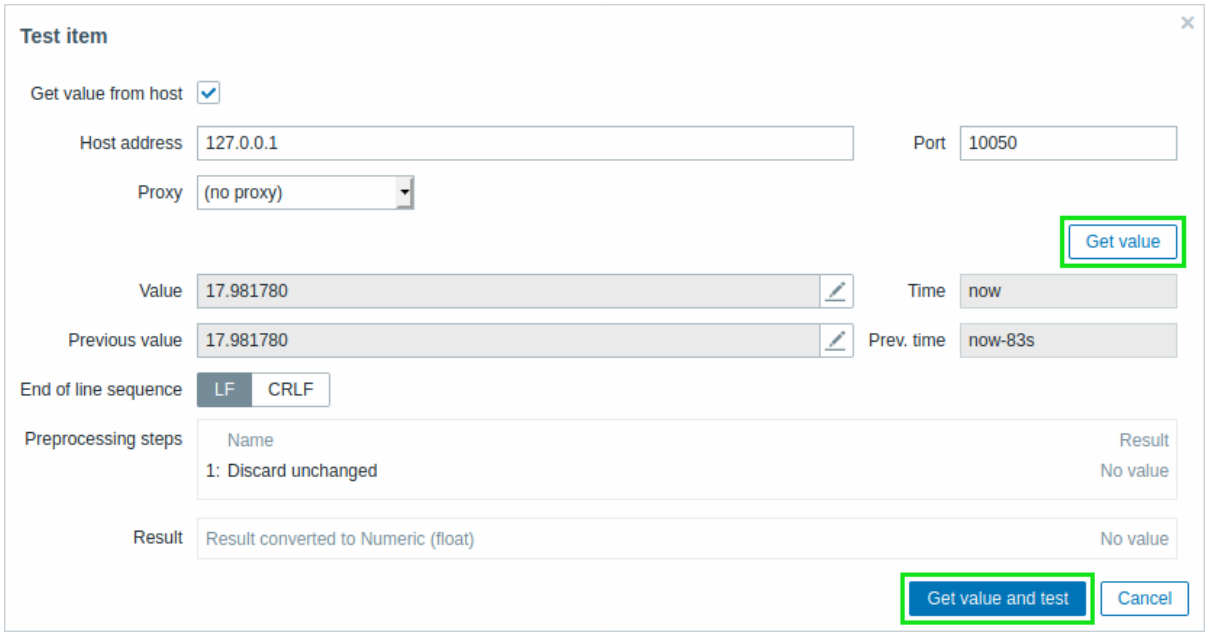


监控项测试功能需要主机参数字段（主机 ip 地址，端口，proxy 名称/没有 proxy）。这些字段如下:

- 可以填充的值会自动填充，例如，对于需要 agent 的项目，通过从主机选定的 agent 接口自动获取信息
- 模板的监控项必须手动填充

- 如果某些字段对于该监控项类型不是必须的，则该字段会被禁用（例如在可计算类型和 zabbix 整合类型的监控项中主机 ip 地址字段不可用，在可计算类型的监控项中 proxy 字段不可用）

对于测试的监控项，点击 *Get value*。如果监控项的值接收成功，则会在 *Value* 字段显示。



也可以使用从 host 成功检索到的值来测试预处理。

实际上，监控项测试表单是 Zabbix 最新版本中已知的预处理测试表单的扩展。因此，如果你以前只能针对假设的输入值测试预处理步骤，那么现在可以根据刚刚收到的实际测试值测试预处理。

根据实际值测试预处理步骤，点击 *Get value and test*。

另请参见：

- 测试监控项
- 测试预处理

现在检查

在最新版中 *Check now*（现在检查*)* 选项 改为 *Execute now*（现在执行*)*，以避免将其与监控项测试功能混淆。

'nodata' 触发器对 proxy 可用性的敏感度

默认情况下，'nodata' 触发器对代理可用性的敏感度—'nodata' 触发器不会在连接恢复后立即触发，但会在延迟期间跳过数据。

告警压制被打开：

- 对于被动 proxy—如果连接恢复超过 15 秒且不少于 2 秒（或 ProxyUpdateFrequency 参数的秒数）
- 对于主动 proxy—超过 15 秒后恢复连接

你还可以利用第二个参数关闭对代理可用性的敏感度，例如：nodata(5m,strict)。这种情况下，函数在没有数据的评估期 (通常情况下是 5 分钟) 一结束就会继续以前的工作。

也可以用新的键值 `zabbix[proxy,<proxy name>,delay]` 监控 proxy 的延迟内部检查.

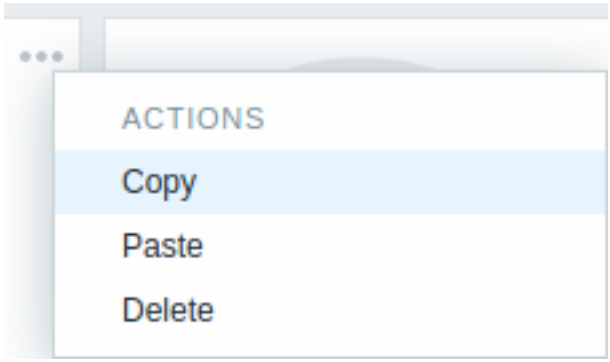
自定义前端模块

可以通过添加第三方模块或开发自己的模块来增强 Zabbix 前端的功能，而不需要更改 Zabbix 的源代码。点击模块查看更多信息。

复制粘贴小部件

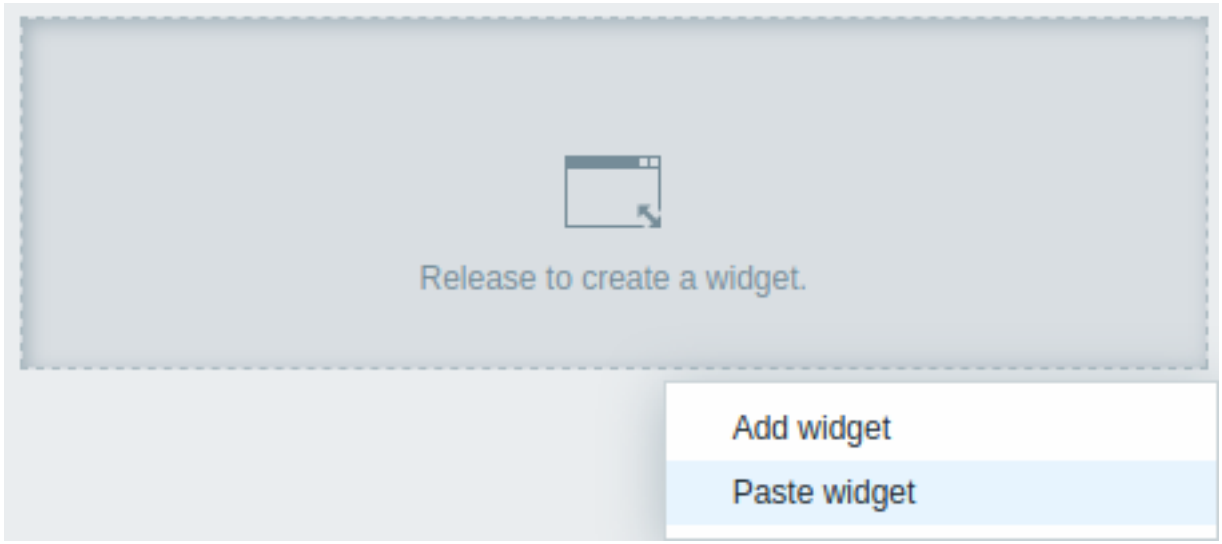
仪表盘的小部件可以复制和粘贴。它们可以复制粘贴到同一个或者不同的仪表板。

可以用小构件菜单复制小部件：



另外，可以使用复制的小部件创建具有相同属性的新小部件。粘贴小部件：

- 编辑仪表盘时点击 *Paste widget* 选项
- 在仪表盘某个区域添加新的小部件时点击 *Paste widget* 选项 (必须首先复制小部件，以便粘贴选项可用)



复制的小部件还可以使用小部件菜单中的 *Paste*（粘贴）选项将其粘贴到现有的小部件上。

管理大量主机

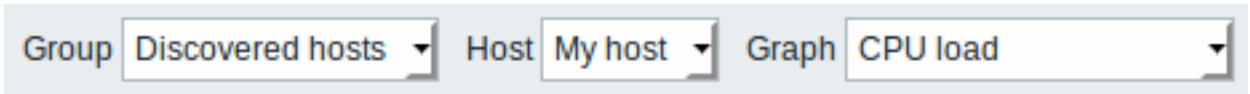
为了更易于使用大量主机和其他元素而做了一些改进。

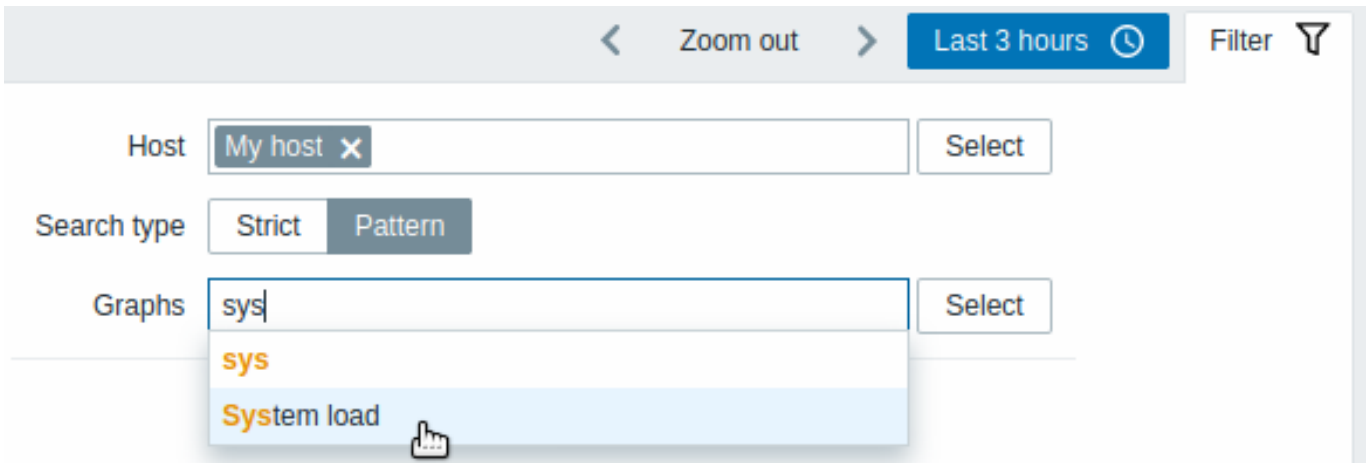
在以前的各版本中，Zabbix 的特性是存在许多用于选择主机和主机组的下拉框，有时也用于选择其他元素（如图形）。这些下拉菜单的位置包括页面顶部和弹出窗口。在新版本中，很多地方都做了修改 (见下面的位置列表)：

- multi-select（多选字段）取代了弹出框中的下拉框
- multi-select（多选字段）取代了许多位于页面顶部的下拉菜单；其中一些字段中被移到了过滤器中

注意:

- 两个主机组和主机下拉菜单更改为单个主机多选字段用于主机的组选择
- 还有一个用于搜索图形名称模式的新选项：





新版本中

- 按主机过滤被添加到触发器/数据页面小部件
- 每页行数设置从用户配置文件应用到 web 监控页面、主机清单概述和可用性报告页面
- 硬编码的 50 条记录的限制没有分页应用于触发器/数据概览页面和小部件

有关更改的主机/主机组/图形/等选择的详细信息，请参见各个页面：

- 监控：
  - 仪表盘（带有动态小构件的主机）
  - 图形
  - Web 场景
  - 触发器概述
  - 数据概述
  - 屏幕/幻灯片（带有查看聚合图形）的主机
- 资产记录：
  - 资产记录概述
  - 主机
- 报表：
  - 主机可用性报告
- 配置列表：
  - 主机
  - 模板
  - 应用集
  - 图形
  - web 场景

重新定义 LLD 规则

可以根据 LLD 对象和原型名称在自动发现过程中过滤掉监控项、触发器、主机和图形或覆盖其属性。

IPMI 传感器自动发现

添加了新的 IPMI 监控项 `ipmi.get`，该监控项返回带有 IPMI 传感器相关信息的 JSON。可用于IPMI 传感器自动发现。

监控项键值限制提高

监控项键值的最大长度从 256 个字符增加到 2048 个字符。

浮点数类型的值的范围得到扩展

浮点数类型支持约 15 位精度，范围从约-1.79E + 308 到 1.79E + 308(PostgreSQL 11 和更早版本除外)。仅对于新部署的环境生效。对于老版本的升级安装，必须用手动补丁 。

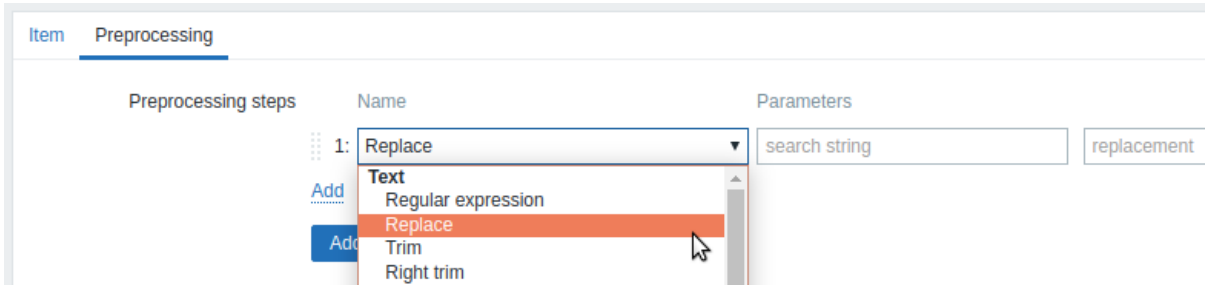
不需要 DNS 的 ODBC 监控

监控项 `db.odbc.*` 中增加了一个新参数 `connection string`。可以通过两种方式配置*ODBC 监控* ：

- 使用/etc/odbc.ini 中设置的数据源名称
- 使用连接字符串

查找并替换预处理步骤

监控项预处理添加了一个选项，允许用另一个字符串来查找和替换指定的字符串：



此步骤有两个参数：

- *search string* - the string to search for
- *replacement* - 用于替换搜索字符串的字符串。替换字符串也可能为空，从而可以在匹配到选定的字符串时进行删除。

Zabbix sender 输入文件支持纳秒级

新的 Zabbix sender 选项

`-N, --with-ns`

Zabbix sender 输入文件时支持纳秒。该选项只能与`--with-timestamps` 选项一起使用，例如：

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

该选项指定输入文件的每一行包含以下以空格分隔的内容，例如 `<host> <key> <timestamp> <ns> <value>`，等

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

与 Zabbix 数据库的安全连接

可以从以下位置配置与 MySQL 和 PostgreSQL 数据库的安全 TLS 连接：

- zabbix_ 前端配置
- zabbix_server/proxy_ 配置

限制代理检查

可以通过创建监控项键值的白名单或黑名单来限制代理端的检查。

白名单/黑名单是通过配置文件中的两个参数实现的：

- `AllowKey=<pattern>` - 允许哪些检查；<pattern> 使用通配符（*）表达式指定
- `DenyKey=<pattern>` - 拒绝哪些检查；<pattern> 使用通配符（*）表达式指定

另请参阅：限制 agent 检查

更强大的加密功能

更强大的 bcrypt 加密技术取代了 MD5 用于对用户密码进行 hash。升级后自动更改为更强的密码，无需在用户端进行任何操作。请注意，超过 72 个字符的密码将被截断。

在 WEBHOOKS 中使用 HTTP 代理

可以在配置webhook时指定 HTTP 代理。新的 HTTPProxy 参数在 webhook 参数列表中列出，默认值为空。

指定代理值时，支持与配置项HTTP 代理配置相同的功能。

发现规则过滤

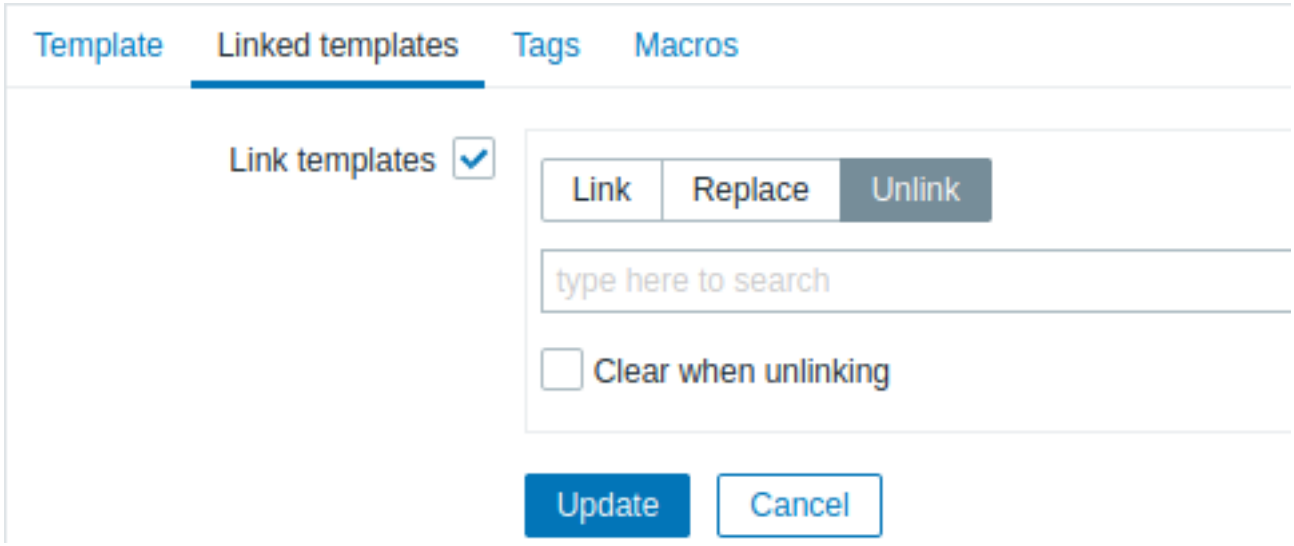
以前低级发现规则的列表总是链接到单个主机，因此无法在一处查看所有发现规则，也无法过滤特定主机组或有错误的主机规则。

新版本中，自动发现规则包含一个过滤器，允许按主机组、主机、发现的监控项类型、发现规则状态和其他参数进行过滤。此外，列表中添加的第一列始终显示发现规则的主机。

新的批量更新选项

现在可以：

- 批量更新主机或模板层面定义的用户宏
- 批量使用主机或更新模板时取消模板链接：

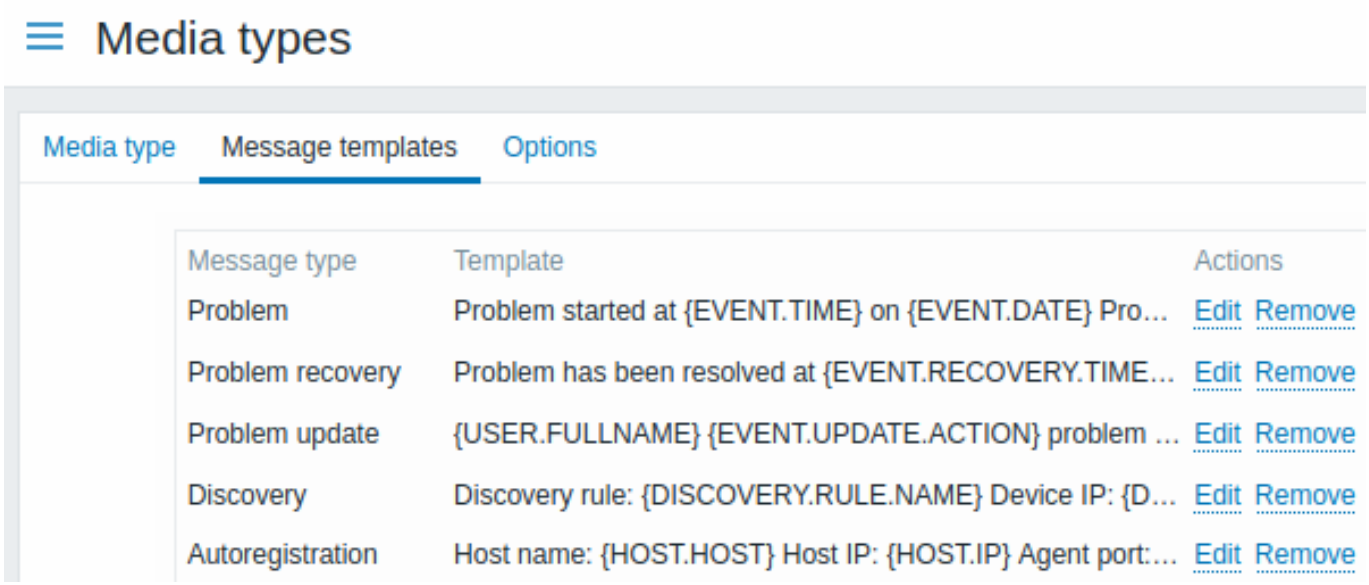


可以参考：

- 主机批量更新
- 模板批量更新

每种媒体类型的默认消息

可以在定义媒体类型时为每种事件类型指定默认消息模板。



因此在配置操作时，默认不再进行信息编辑。

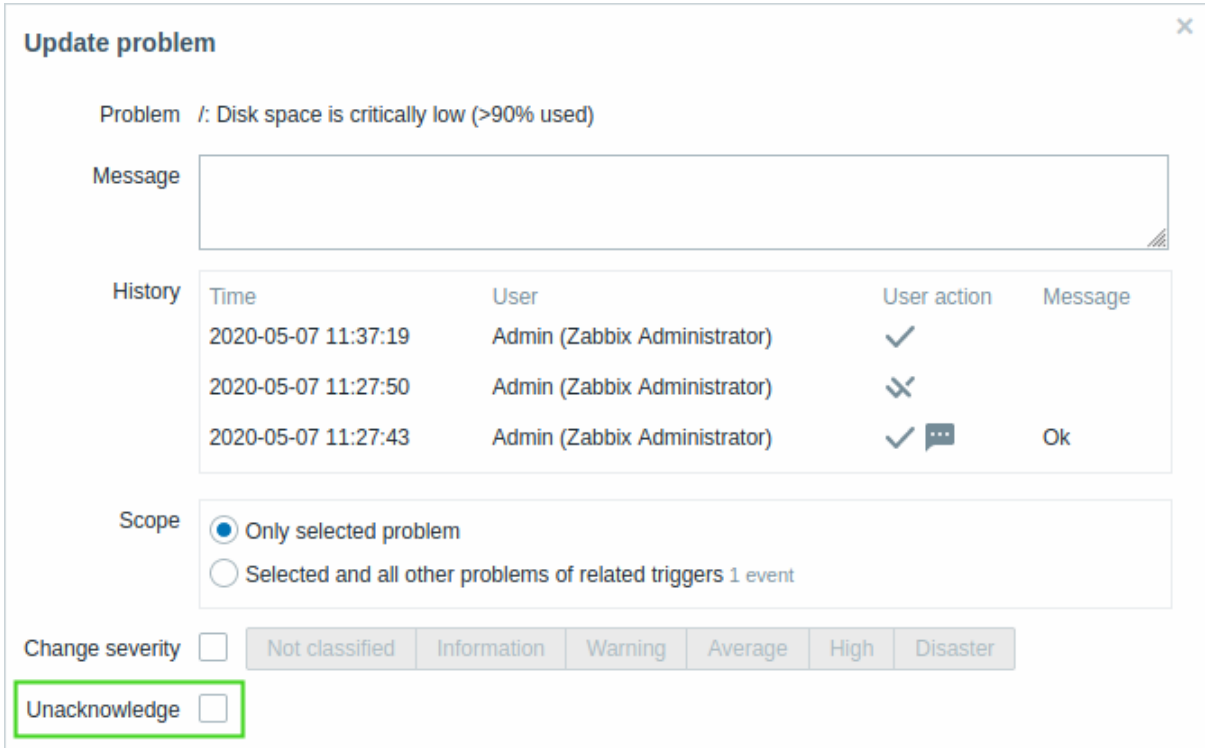
问题的确认

对用于问题确认和更新操作的问题更新页面进行了一些改进：

- 显示问题名称（如果存在多个问题，则选择 N 个问题）
- 问题更新信息的大小从 256 个字符增加到 2048 个字符
- 可以取消问题（见下文）

取消确认选项

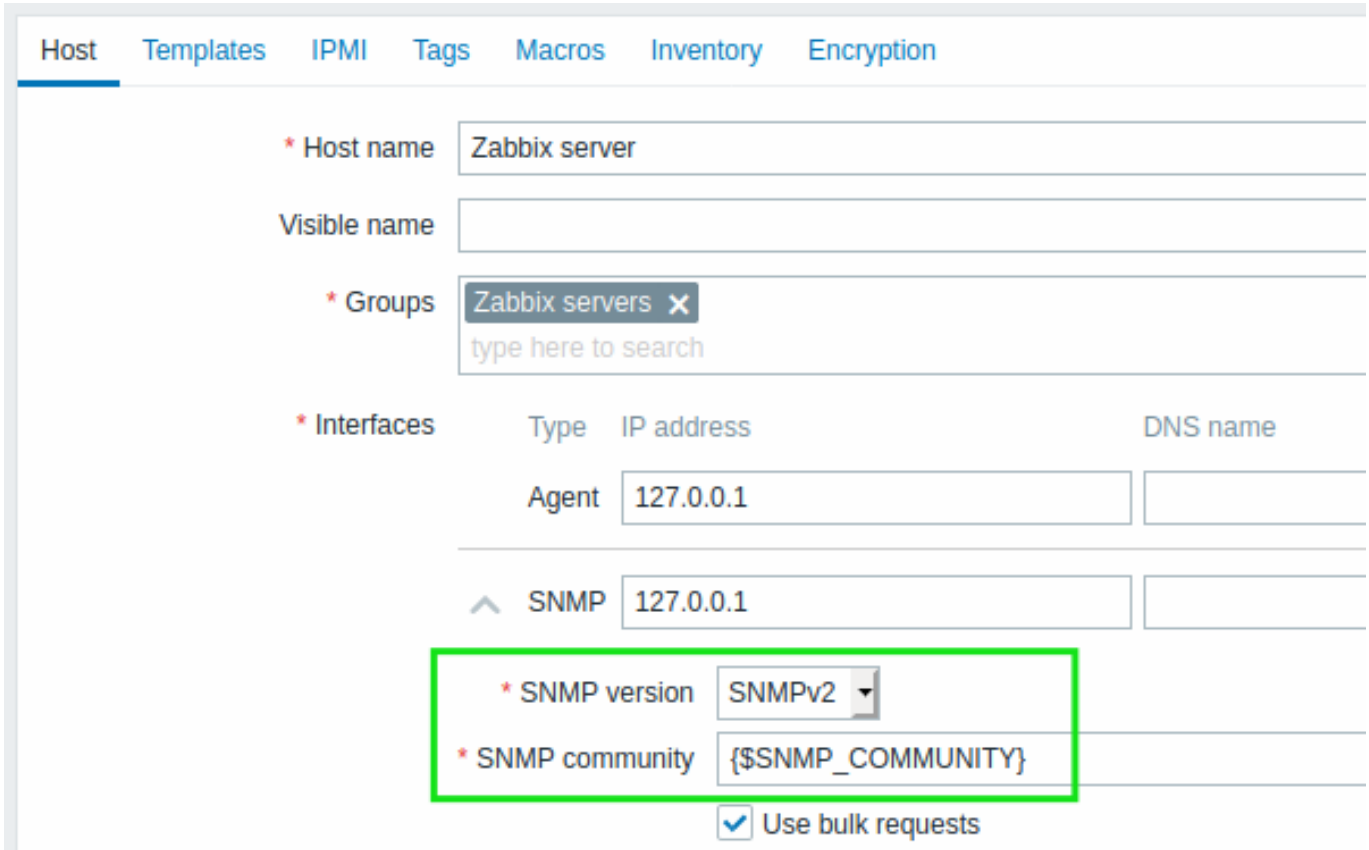
有时可能会误确认问题，新版本可以进行 “不确认” 更正。在更新问题页面问题可能取消确认。



在问题历史记录列表中，不确认会显示一个特殊图标：

主机接口级别的 SNMP 凭据

老版本中，SNMP 版本和凭证设置在监控项级别。在新版本中，这些都可以在主机接口级别设置：



另请参阅：配置 SNMP 监控

创建监控项时，监控项类型下拉列表不再包含 SNMP v1、v2 和 v3 代理的三个条目。只有 *SNMP agent* 类型，并且可以根据需要选择 SNMP 接口。

手动清除 SNMP 缓存

Zabbix server 和 Zabbix proxy 支持使用`-R snmp_cache_reload`选项进行运行时控制，该选项可以重载所有主机的 SNMP 缓存并清

除所有 SNMP 属性 (启动时间、启动装置、ID、凭据等)。Net-SNMP 需要 5.3.0 或更高版本。

电子邮件线程

与同一事件相关的电子邮件通知被划分为一个线程。

支持 Elasticsearch 7

不再支持旧的 Elasticsearch 7.X 之前的版本。

SAML 身份验证

登录到 Zabbix 支持 SAML 2.0 身份认证。

Webhook 集成

新的集成使用webhook媒体类型将 Zabbix 通知推送至：

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

Zabbix agent 2 Zabbix agent 2 首次在 Zabbix 4.4 版本中测试使用，现已得到正式支持且功能得以扩展：

Windows 支持

可以从 Windows 平台上的源代码编译 Agent 2。

Docker 监控插件

Zabbix agent 2 的 Docker 插件现已作为 Docker 容器的现成可用监控的一部分（详情参阅监控项键值 列表）。

Memcached 监控插件

Zabbix agent 2 的 Memcached 插件现已作为 Memcached 实例现成可用监控的一部分（详情参阅说明）。

MySQL 监控插件

Zabbix agent 2 的 MySQL 插件现已作为 MySQL 实例现成可用监控的一部分（详情查看 说明）。

Agent 2 插件更新

现在只支持指定会话通过插件配置参数传递 URI、用户名和密码。因此将不再支持 Plugins.<PluginName>.Uri、Plugins.<PluginName>.User、Plugins.<PluginName>.Password 等格式的参数。可以支持 Plugins.<PluginName>.Sessions.<SessionName>.Uri、Plugins.<PluginName>.Sessio
Plugins.<PluginName>.Sessions.<SessionName>.User 等命名会话参数。

或者，可以在监控项键值参数中直接提供 URI，用户名和密码。

可以参考：

- Zabbix agent 2
- 插件
- 在 Windows 操作系统部署 Zabbix agent 2

宏 能够在前端屏蔽宏内容

宏已具备加密文本模式。如果启用，将用星号屏蔽宏的内容，以保护敏感信息，例如密码或共享密钥。

主机原型中支持的宏

可以为主机原型定义用户宏，并且在宏的值字段中使用 LLD（从原型创建主机时，将解析 LLD 宏）。

IPMI 凭据中支持的宏

IPMI主机配置用户名和密码中支持宏。

新的宏

以下为新支持的宏：

- {EVENT.DURATION} 返回事件的持续时间。
- {EVENT.TAGSJSON} 和 {EVENT.RECOVERY.TAGSJSON} 宏将被解析为包含事件标记对象或恢复事件标记对象的 JSON 数组。

有关更多详细信息，请参见宏的使用场景。

更新的宏

- 基于触发器的通知和命令，问题更新通知和内部通知现在支持 {HOST.ID}。

数据库　不再支持 IBM DB2 数据库

IBM 的 DB2 数据库不能再用作 Zabbix 的存储数据库。

更新最低要求版本

受支持的数据库 的最低要求版本为：

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

TimescaleDB 时序数据库支持本地压缩

在部署 Zabbix server 时如果用了 PostgreSQL 10.2 及以上版本数据库或者 TimescaleDB 1.5 及以上版本数据库，TimescaleDB 时序数据库会支持本地压缩。

新模板　新的官方模板可用于对以下内容进行监控：

Elasticsearch

- *Template App Elasticsearch Cluster by HTTP* - 本地模板monitor Elasticsearch.

ClickHouse

- *Template DB ClickHouse* - 使用 HTTP 代理从 ClickHouse HTTP 接口采集节点指标。（查看 说明）。

Memcached

- *Template App Memcached* - 通过 Zabbix agent 2 监控 Memcached 服务。

MySQL

- *Template DB MySQL by Zabbix agent 2* - 通过 Zabbix agent 2 监控 DBMS MySQL 及其分支。

Docker

- *Template App Docker* - 通过 Zabbix agent 2 监控 Docker 容器。

*

Server

- *Template Server Chassis by IPMI* - 使用 BMC 通过 IPMI 监控服务器机箱。

您可以获取以下模板：

- 在新版本的 *Configuration* → *Templates*（配置 → 模板）页面；
- 从以前的版本升级时，可以从Zabbix Git 存储库下载最新的模板，然后 zabbix web 页面的 *Configuration* → *Templates*（配置 → 模板）手动导入。如果提示同名模板已经存在，请在导入前选择 *Delete missing*（删除缺失）选项以实现干净导入。这样，从更新的模板中排除的监控项将被删除（请注意，已删除项目的历史记录将丢失）。

监控项

- 监控项`zabbix[stats,<ip>,<port>]` 也可以返回 zabbix server 或 proxy 的版本。
- 添加了新的内置监控项 `zabbix[version]` 用于返回 zabbix server 或 proxy 的版本。

前端　PHP 版本的最低要求

PHP 版本最低要求已从 5.4.0 升级到 7.2.0。

不再支持 Internet Explorer 11

Zabbix 不再支持 Microsoft Internet Explorer 11。

页面选择下拉列表集成到标题中

Zabbix 的前端可能会根据用户的选择展示不同的页面效果。例如，*Administration* → *General*（管理 → 一般）可能显示 12 个不同的页面。

以前，页面选择是在页面右上角的一个非常小、容易被忽视的下拉菜单中进行的。现在这个选项已经被整合到左侧的标题栏中。

|<| |<| |<| |<| |-| |<|

此更改影响以下部分：

- *Monitoring* → *Overview*（监测 → 概览）
- *Monitoring* → *Screens*（监测 → 聚合图形）
- *Configuration* → *Actions*（配置 → 动作）
- *Administration* → *General*（管理 → 一般）
- *Administration* → *Queue*（管理 → 队列）

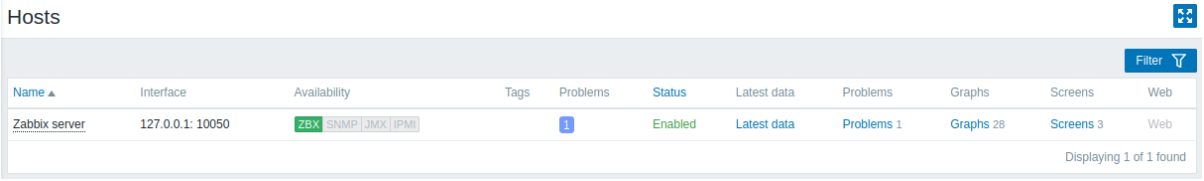
新增监控所有主机的内容

前端新的页面 *Monitoring* → *Hosts*（监测 → 主机）提供了单个位置中所有受监视设备的详细视图。为了简化导航栏，*Monitoring*（监测）选项卡已经删除了 *Web*（Web）和 *Graphs*（图形）菜单。现在，可以通过点击 *Monitoring* → *Hosts* （监测 → 主机）选项中的相关链接来访问这两部分内容。

可以从 *Monitoring* → *Hosts*（监测 → 主机）获取以下信息:

- Hostname 主机名
- Main interface 主要接口
- Availability 可用性
- Tags 标签
- Problems 问题（指示当前未解决问题的图标）
- Status 状态
- Latest data 最新数据（连接到 *Latest data* 最新数据部分）
- Problems 问题（未解决问题的数量以及 *Problems* 问题部分的链接）
- Graphs 图形（图形数量和 *Graphs* 图形部分的链接）
- Screens 聚合图形（聚合图形数量和 *Screens* 聚合图形部分的链接）
- Web scenarios Web 检测（聚合图形数量和 *Web* 部分的链接）

上面列表中的链接提供了一种方便的方式来查看相应的页面，其中包含有关给定主机的更多细节。具有管理员和超级管理员权限的用户还可以从该部分快速导航到主机的配置页面。有关更多详细信息，请参见详细信息。

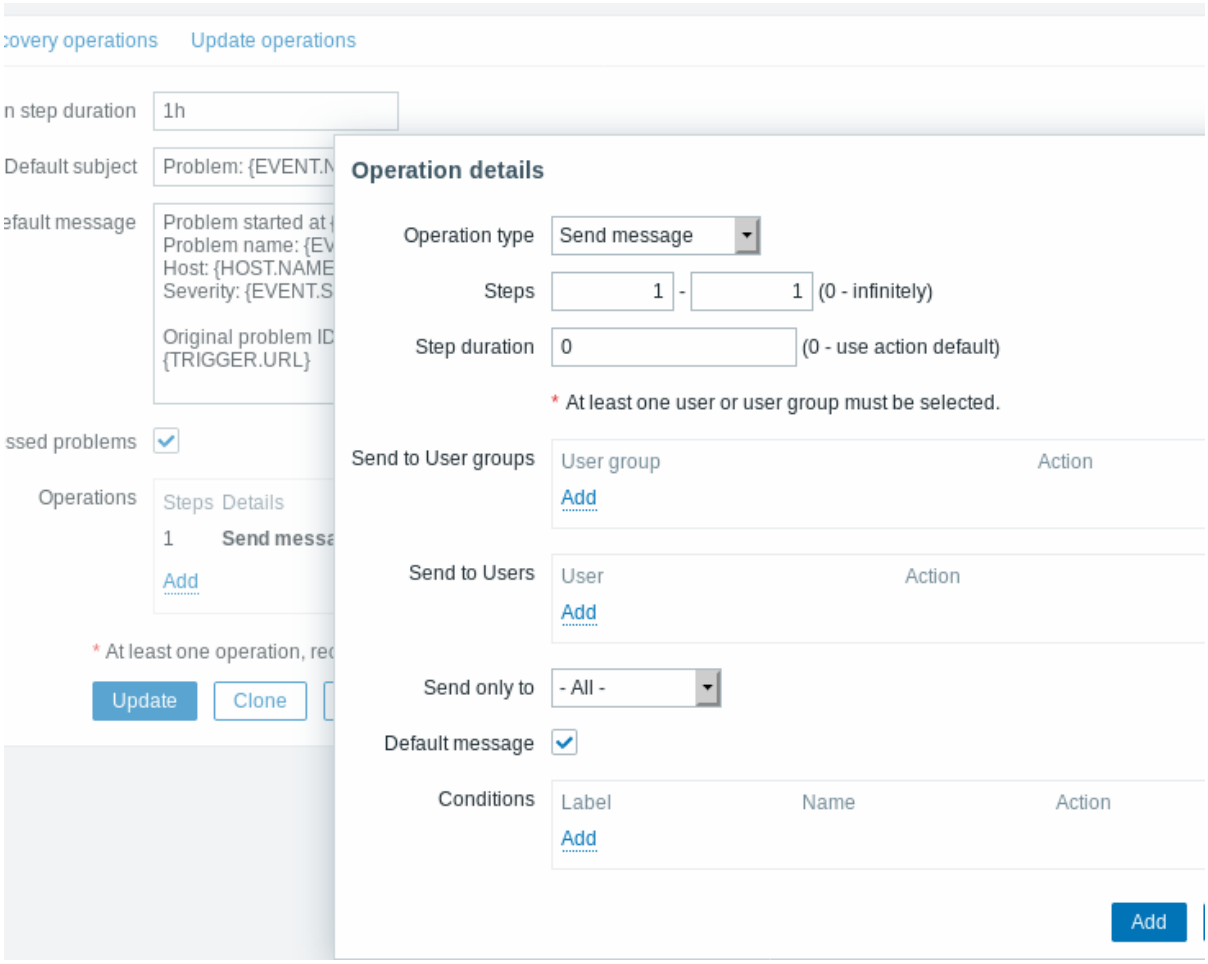


将细节编辑为弹出窗口

在 Zabbix 前端的几个配置项中，细节编辑作为一个弹出窗口打开。这是为了:

- Action conditions 动作条件
- Global correlation conditions 全局相关条件
- Problem update screen 问题更新页面
- Action operation details 动作操作细节
  - 参见配置操作、恢复操作 和更新操作 选项卡。
- 维护期细节
  - 参见维护期配置选项卡
- 发现规则详细信息
  - 查看发现规则属性

在很多情况下，这个改变可以避免在一个用户界面屏幕上配置太多的选项。例如，操作操作的详细信息现在会在单独的弹出窗口中打开。



新增仪表盘小部件过滤条件

仪表盘小部件支持按照按问题严重性 和按问题主机标签过滤问题。

能够将图形小部件下载为图像

图形小部件和图形 (经典) 小部件可以被保存为.png 格式的图片。

在 Monitoring→Problems（监测 → 问题）中按严重性过滤

在 *Monitoring→ Problems*（监测 → 问题）中显示的问题可以通过一个或几个单独选择的严重性来过滤。以前只能根据可用的最低严重级别进行过滤。

Webhook 媒体类型测试可用性得到改善

可以在 webhook 媒体类型测试 期间查看日志条目。

其它

- 首次打开时，最新数据页面不再显示任何内容。
- Web 场景 HTTP 用户代理列表已更新。

守护进程　远程命令登录 agent

远程命令登录日志在 Zabbix agent/agent 2 (LogRemoteCommands=1) 上启用。如果它是由 HostMetadataItem、HostInterfaceItem 或 HostnameItem 参数在本地启动的，则不会为 system.run[] 创建日志条目，仅当远程执行时，才会记录 system.run [] 命令。

agent2 的永久存储

添加以下配置参数，Zabbix agent2 就能将收集到的用于主动检查的数据存储在持久缓冲区中（默认禁用）。

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

加密库

不再支持 mbedTLS（PolarSSL）加密库。

使用表格数据监视 JMX 属性

添加了对列表数据 的支持。支持 JMX 代理数据收集和 LLD 自动发现。

6 Zabbix 5.0.1 新功能特性

Spiceworks 集成

详见集成指南 Spiceworks 。

OTRS 集成

webhook 集成可以使用 webhook 媒体类型来推送 Zabbix 通知到OTRS。

Windows 性能计数器实例

可以使用新的键值 `perf_instance.discovery[]` 和 `perf_instance_en.discovery[]` 发现 Windows 性能计数器的对象实例。这对于发现多实例性能计数器和自动生成 perf_counter 和 perf_counter_en 监控项非常有用 (参见更多信息)。

通过 agent2 监控 PostgreSQL 数据库

可以通过 Zabbix agent2 快速部署 PostgreSQLPostgreSQL 数据库插件 监控。

缓存配置参数

Zabbix server/proxy 配置文件中 CacheSize 参数的最大值已经从 8GB 增加到 64GB。

7 Zabbix 5.0.2 新功能特性

vfs.file.exists[] 监控项有更多的文件类型

老版本的 vfs.file.exists[] 监控项只支持常规文件和链接。现在增加了对目录、套接字、块设备、字符设备等更多文件类型的支持。

要包含的文件类型可以在第二个参数中以逗号分隔的带引号列表形式指定，而要排除的文件类型也可以在第三个形参中指定:

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

查看更多细节，点击agent 监控项 中的 vfs.file.exists[] 。

全局脚本执行日志

全局脚本执行信息记录在 审计日志，详细信息如下

- Timestamp 时间戳
- User that started a script 执行脚本的用户
- IP of the target host or Zabbix server/proxy 目标主机或 Zabbix server/proxy 的 IP 地址
- Script after macros substitution (secret macros will be shown as *****) 宏替换后的脚本 (加密宏将显示为 *****)
- Script results 脚本执行的结果

此外，现在可以通过在脚本 中插入与用户相关的宏来传递用户信息。支持的宏:

- {USER.ALIAS} - Zabbix 用户
- {USER.FULLNAME} - Zabbix(username) 中指定的当前用户的姓和名
- {USER.NAME} - Zabbix 中指定的当前用户的名字
- {USER.SURNAME} - Zabbix 中指定的当前用户的姓氏

如果脚本在动作配置的操作下自动执行，这些宏将不会被解析。

Webhooks

MS Teams webhook 配置在消息卡中支持自定义字段和自定义按钮。

用户宏上下文支持的正则表达式

除了静态字符串外，用户宏上下文也支持正则表达式，使用以下语法:

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

使用正则表达式可以显著减少需要定义的用户宏上下文的数量。

点击带有上下文的用户宏 查看更多信息.

新的模板

Etcd 模板

- *Template App Etcd by HTTP* - 通过 HTTP 代理从 Etcd's `/metrics` 中收集监控指标数据 (点击 查看说明).

Microsoft SQL Server 模板

- *Template DB MSSQL by ODBC* - 通过 ODBC 从 DBMS Microsoft SQL Server 收集指标 (点击 查看说明).

IIS 模板

- *Template App IIS by Zabbix agent*, *Template App IIS by Zabbix agent active* - 通过 Zabbix agent 从 2012R2 版本的 Windows Server 及互联网上的信息服务采集监控指标。

你可以获取以下模板：

- 在新版本的 *Configuration*（配置）→ *Templates*（模板）中；
- 如果你时老版本升级来的，可以从 Zabbix 的 Git 存储库 下载这些模板或者在下载的最新 Zabbix 版本的 "Templates"（模板）目录中找到它们。另外，在 zabbix 页面的 *Configuration* → *Templates* 你可以手动导入模板。

agents 取消 EnableRemoteCommands 参数

目前 agent 的 EnableRemoteCommands 参数 的情况：

- Zabbix agent 已弃用（它直接作为相应的 AllowKey/DenyKey 参数的别名）
- Zabbix agent2 不支持

使用 AllowKey/DenyKey 参数替代。

系统的 "主机 host" 栏中不再包含的模板

在系统 "配置——主机" 中显示的主机信息不再包括模板。模板的信息现在显示在单独的一行中。

log.count 监控项中日志修改时间被忽略

在 log.count 监控项 中日志修改时间被忽略。

增强页面小部件的安全性

现在 URL 仪表盘小部件和 URL 屏幕元素将检索到的 URL 内容放入沙箱中。默认所有沙盒均已启用。可以在 *defines.inc.php* 配置文件中修改 `sandbox` 参数，但是出于安全原因，不建议关闭沙箱。点击sandbox 了解 sandbox 的更多信息及元素说明。

8 Zabbix 5.0.3 新功能特性

VMware 数据中心发现

新的 `vmware.dc.discovery[url]` item 返回 JSON 数据格式含有 {#DATACENTER} 和 {#DATACENTERID} 属性.

在 VMware event log 包含主机信息

在 `vmware.eventlog[<url>,<mode>]` item 返回的信息中心，包含了源主机的信息包含有关源主机的信息（如果在日志中能够检测到此类信息）.

agent 2 新增支持的服务检测

Windows 上的 Zabbix agent 2 items `net.tcp.service[]` and `net.tcp.service.perf[]` 新支持 HTTPS and LDAP 服务的检测.

链接到模板的主机列表

配置 → 模板部分能够显示通过模板进行过滤的主机信息。因此，一次单击就可以看到连接到模板的所有主机。

新模板

针对 Oracle 监控，提供新的开箱即用的模板 *Template DB Oracle by ODBC*. 获取如何使用这个模板, 请参见使用指引.

您能够通过以下方式获取此模板:

- 在新安装的环境下，通过 配置 → 模板获取;
- 如果您是从低版本 Zabbix 升级到高版本 Zabbix, 你可以从 Zabbix Git 仓库下载新的模板 Git 存储库或者在下载的最新 Zabbix 版本的 `templates` 目录中找到它。然后, 在配置 → 模板中，您可以人工导入相关的模板.

最新数据

- 重新实现了扩展/折叠应用程序（5.0.2 中没有）;
- 一个' 显示从 N 到 N 从 N 结果中' (例如：显示从 *1* to 到 *50* 从 *146* 结果中) 字符串已添加到页面中，允许估计总共显示了多少项，以及是否在页面中显示了部分项。
- 优化了页面显示性能, 为了获得更好的页面性能，如果在筛选器中未选择主机，则自动选中并禁用 *Show items without data* 选项。

在 IBM AIX 系统中监控物理主机的 CPU 利用率

针对 AIX 的 Zabbix 代理增强了监视物理 CPU 利用率的能力。`system.cpu.util[]` item key 增加了第四个参数:

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

<logical_or_physical> 键参数允许指定 items 是否应报告逻辑或物理 CPU 利用率 (所支持的值是: `logical`, `physical`). 查看所有 item 描述.

9 Zabbix 5.0.4 新功能特性

Webhooks 新的集成方式

新的集成方式允许使用 webhook 方式把 Zabbix 通知与第三方平台进行集成:

- iLert
- SolarWinds
- SysAid
- TOPdesk

查看所有可用的 webhooks 方式可用的 webhooks.

非触发器事件支持

在现有基于触发器通知的基础上，webhooks 方式能够把自动发现，主动 Agent 自动发现和内部事件向第三方系统进行通知。.

获取 HTTP 响应（response）头部信息（headers）

使用 webhooks 方式，支持从CurlHttpRequest 对象获取 HTTP 响应的头部信息.

前端 web monitoring 页面中的主机信息

在web monitoring页面中点击主机信息，可以打开主机菜单

Aggregate item 帮助器

对于aggregate items，页面中的 item key 选择器，可以了列举所有可用的 aggregate keys (grpavg,grpmax,grpmin,grpsum) 和它们相关的描述.

Oracle 数据库监控 Zabbix agent 2 提供了新的 Oracle 监控插件 *Oracle monitoring plugin*，可以对 Oracle 数据库进行监控. 更多的信息, 请参见:

- 插件配置
- 对于支持的item keys的描述

提供一个可以快速部署的新的官方模板:

- *Template DB Oracle by Zabbix agent 2* - 使用 Zabbix agent 2 对 Oracle 进行监控.

Agent 2 可作为 **Windows** 的服务 Zabbix agent 2 可作为 Windows 的服务被运行Windows service.

age/duration 宏精确到秒级 多个内置的宏 用于返回 age/duration 的信息，现返回精确到秒级的值:

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

此前，这些宏仅仅返回精确到分钟的值。

10 Zabbix 5.0.5 新功能特性

设置数据库加密连接

用于加密前端和数据库之间连接的参数（在 Zabbix 前端安装期间可设置）已被修改. Zabbix web 界面的 *Configure DB connection* 步骤现在提供了一个新的复选框 *Verify certificate*，并在选择复选框时显示了其他选项. 在特定配置中不可用的参数将被禁用. 例如, 当使用

MySQL 数据库和 *Database TLS encryption* 并且 *Database host* 设置为 localhost 时，复选框被禁用. 请参见数据库安全连接 的详细描述.

Ceph 监控

针对 Ceph 监控，Zabbix agent 2 提供了新的监控插件 *Ceph plugin*. 获取更多信息, 请参见:

- Plugin 配置
- 对于支持的item keys描述

新的官方模板 *Ceph by Zabbix Agent 2* 提供可以快速部署的方式。

新模板

提供以下新的开箱即用的模板:

- *Template App Ceph by Zabbix agent 2* - 请参见 Zabbix agent 2 模板关于设置指引.
- *Template App PHP-FPM by Zabbix agent* - 请参见 Zabbix agent 2 模板关于设置指引 .
- *Template App PHP-FPM by HTTP* - 请参见 HTTP 模板设置指引.
- *Template App Squid SNMP* - 请参见 描述.
- *Template Tel Asterisk by HTTP* - 请参见 HTTP 模板设置指引.

您能够通过如下方式获取模板:

- 在新安装的环境下，通过配置 → 模板获取;
- 如果您是从低版本 Zabbix 升级到高版本 Zabbix, 你可以从 Zabbix Git 仓库下载新的模板 Git 存储库或者在下载的最新 Zabbix 版本的 `templates` 目录中找到它。然后, 在配置 → 模板中，您可以人工导入相关的模板.

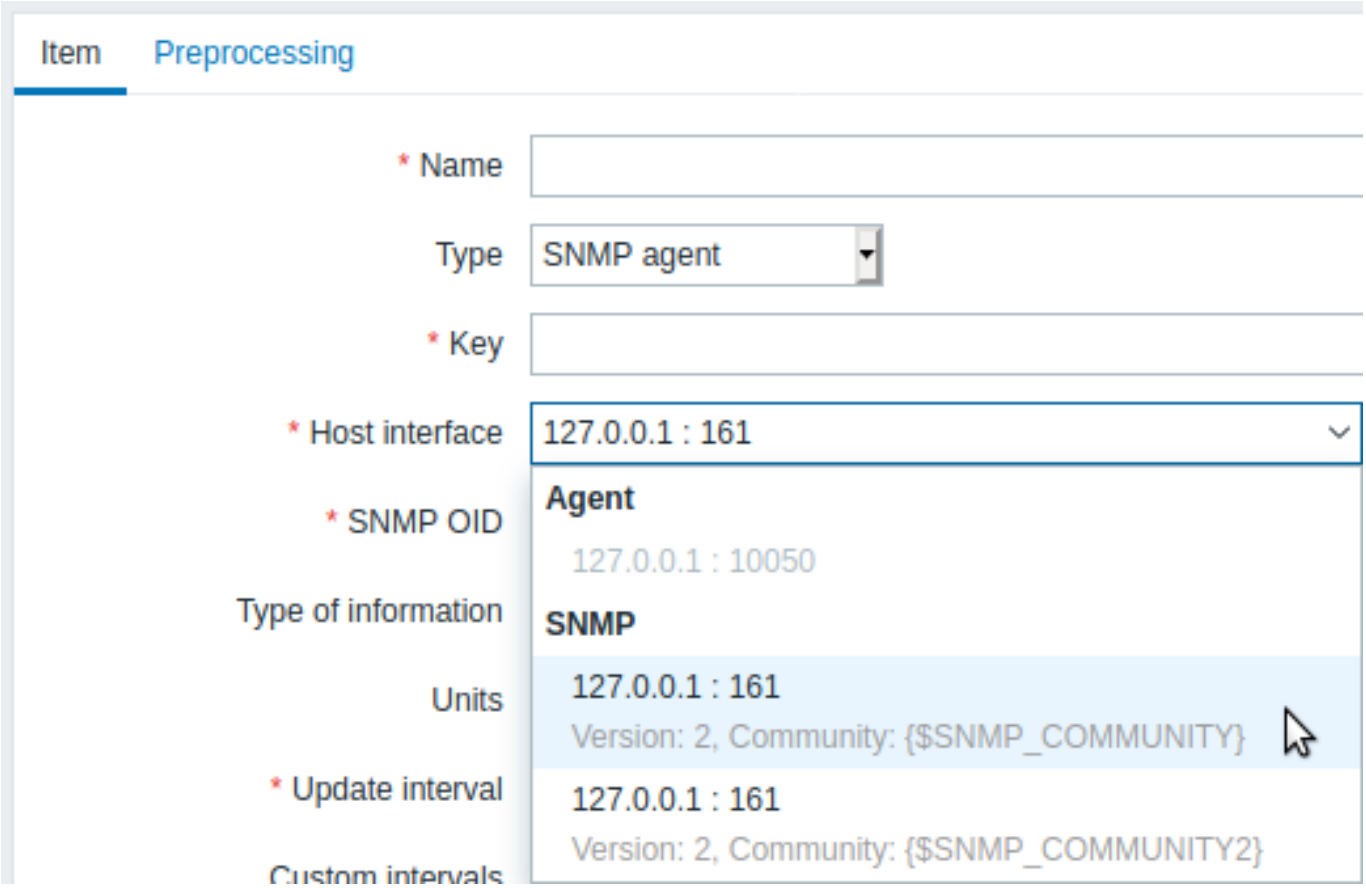
Items

Zabbix agent 2 加入 **system.swap.size[<device>,<type>]** 本地支持.

请参见: Zabbix agent 2 插件

SNMP item 接口选择

item interface selection 的下拉列表经过了可视化的重新设计，现在它包含了 SNMP 接口的详细信息，例如 SNMP 版本和社区字符串，允许查看其他相同接口之间的差异.



类似地，现在在批量更新项目时也会显示此接口详细信息。

将互斥部分添加到诊断信息中

关于server/proxy控制选项'diaginfo'，返回 Zabbix 互斥信息. 您也可以运行'diaginfo' 仅返回互斥部分:

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

11 Zabbix 5.0.6 新功能特性

禁用敏感字段的自动完成属性

为了避免潜在的数据暴露风险，现对许多包含敏感信息的字段关闭自动完成属性，例如用户登录 zabbix 的密码、预共享键 (PSK)、用于各种监控项及主机数据采集的用户名和密码、SNMPv3 身份验证和隐私密码、媒介密码字段；web 场景和 HTTP 监控项中使用的 SSL 密钥密码、HTTP 代理字段；远程命令中的用户名、密码和密钥密码字段。此设置将阻止大多数的浏览器在受影响的字段中使用自动完成。

基于单元状态的系统发现

Zabbix agent 2 的监控项 **systemd.unit.discovery** 通过低级别发现宏{#UNIT.UNITFILESTATE} 返回当前系统单元的启用状态。这些系统单元的状态数据怎么用呢？举一个用例示例：这些系统单元的状态数据可以在自动发现规则过滤器中使用，以过滤掉所有禁用的系统单元，并只发现启用的系统单元。

iTop webhook 集成

这种新的集成方式允许使用webhook 媒介推送 Zabbix 通知到 iTop.

新的模板

以下的监控模板开箱即用:

Apache 项目

- 通过 *JMX* 监控 *Apache Cassandra* - 请查看 JMX 模板的操作指引;
- 通过 *JMX* 监控 *Apache Kafka* - 请查看 JMX 模板的操作指引;
- 通过 *HTTP* 监控 *Hadoop* - 请查看 HTTP 模板的操作指引;
- 通过 *HTTP* 监控 *ZooKeeper* - 请查看 HTTP 模板的操作指引.

Morningstar

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

这些模板是专门用来监控 Morningstar 设备的; 可通过访问链接 用户指引去获取在 zabbix 上面配置 Morningstar 产品监控的详细说明。

你可以通过如下方式获取新的模板:

- 若是新安装的 zabbix，可以在 *Configuration* → *Templates* 获取；
- 若是从旧版本升级的 zabbix, 你可以在 Zabbix 的 Git 仓库 下载新的模板，或者下载最新版本的 zabbix 安装包，在安装包的 templates 目录也可获取到这些新模板。然后在 *Configuration* → *Templates* 页面，手动把这些新模板导入到 zabbix。

防止用户枚举攻击

连续失败的登录尝试后的临时性帐户阻塞仅针对已存在的用户，为了确保攻击者无法猜测出有效的用户名，现使用不存在的用户名尝试登录，帐户阻塞也被强制执行。

为了进一步降低这种攻击的可能性，现在对与不正确登录有关的所有问题都显示一个统一的通用消息:

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

12 Zabbix 5.0.7 新功能特性

监控项

新的 `systemd.unit.get[<unit name>,<interface>]` 监控项 可以允许获取一个系统单元的所有属性。这个监控项仅被 Linux 平台的 Zabbix agent 2 支持。

自定义预处理步骤的乘数

自定义预处理步骤乘数现可接受宏变量 (宏必须解析为整数或浮点数)。

Java 网关配置文件

一个新的 `zabbix.propertiesFile` 配置参数 允许指定一个属性配置文件, 该属性配置文件可用于设置附加属性，附加属性在命令行上不可见或覆盖现有属性。

Miscellaneous

- 现在，Windows 的 Zabbix agent 将事件日志消息中的帐户名和域名替换为 SID。

Miscellaneous

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

13 Zabbix 5.0.8 新功能特性

新的模板

以下模板现在可用于开箱即用的监控:

- *Apache ActiveMQ by JMX* - 请查看 JMX 模板的设置说明;
- *Microsoft Exchange Server 2016 by Zabbix agent*, *Microsoft Exchange Server 2016 by Zabbix agent active* - 请查看 Zabbix agent 模板的设置说明;
- *NetApp FAS3220 SNMP* - 通过 SNMP 监控 NetApp FAS3220.

你可以通过如下方式获取到这些模板:

- 若是新安装的 zabbix，可以在 *Configuration* → *Templates* 获取;
- 若是从旧版本升级的 zabbix, 你可以在 zabbix 的Git 仓库 下载新的模板，或者下载最新版本的 zabbix 安装包，在安装包的 templates 目录也可获取到这些新模板。然后在 Configuration → Templates 页面，手动把这些新模板导入到 zabbix。

Webhook 集成

- 新的集成方式允许使用webhook 媒介推送 zabbix 通知到Rocket.Chat。
- 在微软的 webhook, 已删除硬编码 teams_endpoint 检查，以防止 URL 语法错误。

PostgreSQL 插件

- Zabbix agent 2 的 PostgreSQL 插件 现支撑自定义查询；
- Unix 套接字支持已修复。

RHEL/CentOS packages with PHP 7.3

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

14 What's new in Zabbix 5.0.9

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

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

VictorOps webhook integration

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

HTTP authentication in webhooks

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

JavaScript functions for MD5 and SHA256 calculation

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

JavaScript memory limit

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

New templates

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

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

You can get this template:

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

15 What's new in Zabbix 5.0.10

MongoDB monitoring

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

- Plugin configuration
- Description of supported item keys

brevis.one webhook integration

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

New templates

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

APC UPS

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

Cisco Catalyst 3750 Series

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

See also: Standardized templates for network devices.

Huawei Oceanstor

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

NetApp

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

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

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

See setup instructions for Zabbix agent 2 templates.

MongoDB

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

See setup instructions for Zabbix agent 2 templates.

You can get these templates:

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

Real-time export

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

Server protected against proxy data overload

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

Timescale DB 2.X

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

15 What's new in Zabbix 5.0.11

Improved Slack webhook

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

New templates

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

APC UPS

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

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

You can get these templates:

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

Value cache performance

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

16 What's new in Zabbix 5.0.12

WildFly monitoring

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

You can get these templates:

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

Webhook integrations

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

- Express.ms messenger
- ManageEngine ServiceDesk

Real-time export

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

17 What's new in Zabbix 5.0.13

New templates

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

Cisco

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

DELL PowerEdge

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

HP ProLiant

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

Nginx Plus

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

Zyxel

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

Updated templates

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

You can get these templates:

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

Diagnostic information

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

Default dashboards

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

Miscellaneous

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

18 What's new in Zabbix 5.0.14

Templates

New templates are available:

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

You can get these templates:

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

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

19 What's new in Zabbix 5.0.15

Plugins

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

See also:

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

New templates

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

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

You can get these templates:

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

Configurable TCP queue maximum size

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

20 What's new in Zabbix 5.0.16

Databases

Support for MariaDB 10.6.X has been added.

Handling large proxy configuration

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

See also protocol header information.

Items

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

Frontend

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

21 What's new in Zabbix 5.0.17

Items

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

22 What's new in Zabbix 5.0.18

Items

Agent variant check

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

VMware hypervisor maintenance monitoring

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

VMware system health monitoring

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

Docker container statistics

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

Hostname

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

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

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

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

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

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

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

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

Web monitoring

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

23 What's new in Zabbix 5.0.19

Items

VMware hypervisor discovery

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

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

24 What's new in Zabbix 5.0.20

Sleep method for JavaScript engine

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

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

See also: Additional JavaScript objects

pfSense monitoring template

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

You can get this template:

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

Item changes

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

25 What's new in Zabbix 5.0.21

S.M.A.R.T. plugin Zabbix agent 2 item *smart.disk.discovery*, supported for S.M.A.R.T. plugin, has been updated and now returns {#DISKTYPE} macro value in the lower case.

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

26 What's new in Zabbix 5.0.22

PostgreSQL metrics

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

JMX monitoring The template *Template App Generic Java JMX* now contains discovery rules for low-level discovery of memory pools and garbage collectors.

You can get these templates:

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

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

27 What's new in Zabbix 5.0.23

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

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

You can get these templates:

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

28 What's new in Zabbix 5.0.24

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

29 What's new in Zabbix 5.0.25

This minor version does not have any functional changes.

30 What's new in Zabbix 5.0.26

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

Updated templates PostgreSQL Agent 2 template updated.

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

You can get this template:

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

31 What's new in Zabbix 5.0.27

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

New templates

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

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

You can get this template:

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

32 What's new in Zabbix 5.0.28

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

33 What's new in Zabbix 5.0.29

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

Frontend Miscellaneous

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

34 What's new in Zabbix 5.0.30

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

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

35 What's new in Zabbix 5.0.31

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

Optimized API queries API database queries, created when searching through names in *hosts* and *items* tables, have been optimized and will now be processed more efficiently.

Updated templates *Template DB Oracle by Zabbix agent 2* has been updated according to the changes made to multiple Zabbix agent 2 items.

For more information about the updates, see Template changes.

You can get this template:

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

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

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

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

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

TimescaleDB 2.9.0 support The maximum supported version for TimescaleDB is now 2.9.0.

36 What's new in Zabbix 5.0.32

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

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

37 What's new in Zabbix 5.0.33

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

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

38 What's new in Zabbix 5.0.34

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

39 What's new in Zabbix 5.0.35

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

- Configuring plugins
- Plugin configuration file parameters

40 What's new in Zabbix 5.0.36

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

41 What's new in Zabbix 5.0.37

This minor version does not have any functional changes.

42 What's new in Zabbix 5.0.38

This minor version does not have any functional changes.

43 What's new in Zabbix 5.0.39

This minor version does not have any functional changes.

44 What's new in Zabbix 5.0.40

TimescaleDB 2.12 support Support for TimescaleDB version 2.12 is now available.

45 What's new in Zabbix 5.0.41

This minor version does not have any functional changes.

TimescaleDB 2.13 support

Support for TimescaleDB version 2.13 is now available.

MySQL 8.2 support

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

46 What's new in Zabbix 5.0.42

MySQL 8.3 support

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

MariaDB 11.2 support

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

TimescaleDB 2.14 support

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

Zabbix agent 2 support on Windows

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

47 What's new in Zabbix 5.0.43

MariaDB 11.3 support

The maximum supported version for MariaDB is now 11.3.X.

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。	
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。

Locale

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

2 Agent

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）	

参数 *	描述 **
-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)

选项描	目标
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。

Exit code

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

3 Agent 2

概述

Zabbix agent 2 为新一代 zabbix agent，未来可能会替代原 Zabbix agent。Zabbix agent 2 可以实现：

- 降低 TCP 连接数
- 具有更大的检查并发性
- 易于通过插件进行扩展. 插件可以是:
  - 仅由几行简单代码实现的简单检查
  - 由长时间运行的脚本及数据周期回传的独立数据采集的复杂检查
- 可以替代原有的 Zabbix agent（可以兼容原 Zabbix agent 的所有功能）

Agent 2 是用 Go 语言开发的 (复用了原 Zabbix Agent 的部分 C 代码)。Zabbix agent 2 需要在 1.13+ 版本的Go环境编译。

Agent 2 不支持 Linux 上的守护进程; 而且从 Zabbix 5.0.4 开始，它可以作为Windows service服务运行。

被动检查的工作原理与 Zabbix agent 类似. 主动检查支持 scheduled/flexible 间隔和并行检查。

** 并行检查 **

不同的插件的检查可以并行执行。每个插件的并行检查数量取决于对应插件的能力设置。每个插件可能有一个硬编码的能力设置值 (缺省比如是 100)，该值可在 插件配置参数. 通过 `Plugins.<Plugin name>.Capacity=N` 的命令行进行配置。

支持的平台

Agent 2 支持 Linux 和 Windows 平台。

<note:> 目前, Agent 2 在 Windows 平台上支持的监控项数量是受限的.

:::

Agent 2 安装包支持如下平台：

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

安装

Zabbix agent 2 可使用预先编译好的安装包。若用源码编译 Zabbix agent 2 需要在编译时指定`--enable-agent2` 配置选项。

Agent 选项

如下的命令行参数可以在 Zabbix agent 2 中使用:

参数 *	描述 **
-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* or *--prefix* 变量来确定。
-f --foreground	在前台运行 Zabbix agent (缺省: true)。
-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	显示帮助信息并退出。
-v --verbose	显示 debugging 信息，使用 -p 和 -t 选项。

参数 *	描述 **
-V --version	显示 agent 版本号并退出。
-R --runtime-control <option>	执行管理功能。请参阅运行时控制。

使用命令行 参数的具体示例：:

- 显示 agent 全部的内建监控项和对应的值
- 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。

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

运行时控制

运行时控制可以提供一些远程控制的选项。

选项	描
log_level_increase	增加日志级别。在 Zabbix 5.0.0-5.0.3 版本使用”loglevel increase” 替代 (quoted)。
log_level_decrease	降低日志级别。在 Zabbix 5.0.0-5.0.3 版本使用”loglevel decrease” 替代 (quoted)。

选项	描
metrics	显示可用的指标项。
version	显示agent版本。
help	在运行时控制显示帮助信息。

例:

- 增加 agent 2 日志级别
- 打印运行时控制的选项

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

配置文件

Agent 2 的配置参数除了如下几个有差异外，其余与 Agent 都是兼容的。

新参数	描述
ControlSocket	运行时控制的socket路径。Agent 2的运行时命令行使用控制socket。

新参数	描述
EnablePersistentBuffer, *PersistentBufferFile*, *PersistentBufferPeriod*	agent 2 的这些参数用于配置已激活监控项的持久化存储。

新参数描述	
Plugins	插件可以有自己的参数，以 Plugins.<Plugin name>.<Parameter> 的格式进行设置。插件的常用参数 *Capacity* 的各检查项限制可同时设置。

新参数描述	
StatusPort	agent 2 监听 HTTP 状态请求的端口及显示配置插件列表和一些内部参数。

删除的参数描述

AllowRoot, *User*	不支持，因不支持守护进程。
LoadModule, *LoadModulePath*	可加载模块不支持。

新参数描述
StartAgents	Zabbix agent 中用于使能或关闭增加并行被动检查数。Agent 2 中，因并行检查数是插件层面的配置，且可以通过插件能力配置进行限制，故不支持关闭被动检查。

新参数描述	
HostInterface, *HostInterfaceItem*	目前暂不支持。

更多详细的配置文件选项请参照zabbix_agent2。

退出码

自 4.4.8 版本起，Zabbix agent 2 也可以与较老的 OpenSSL 版本 (1.0.1, 1.0.2) 一起编译。

这样 Zabbix 就可以提供 OpenSSL 中用的互斥锁。如果互斥锁锁定或者解锁失败时就会向标准错误输出 (STDERR) 打印一条错误消息，Agent2 会返回错误码 2 或者 3 并退出。

4 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。
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)

选项描	目标
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。

Supported platforms

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

Locale

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

5 Java 网关

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。

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

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

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

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

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

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。

1 使用源码安装

概述

如果您是使用源码安装, 那么下列信息会帮助你配置Zabbix Java 网关。

文件概述

如果你从源码中获得 Java 网关，那么您最终会得到 $PREFIX/sbin/zabbix_java 下的 Shell 脚本、JAR 和配置文件的集合。这些文件的作用总结如下：

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

Java 网关 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 网关依赖于: Logback, SLF4J和 Android JSON库。

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

Logback 的配置文件。

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

用于启动和停止 Java 网关的便捷脚本。

`settings.sh`

用于控制启动和停止脚本的配置文件。

配置和运行 Java 网关

Java 网关默认监听 10052 端口。如果你想运行 Java 网关在其他端口，你可以在 settings.sh 的脚本中指定其他端口号。详情可见Java 网关配置文件中描述的如何更改端口等其他信息。

> **Warning:**
> 10052 端口不是由IANA 注册的.

当配置好，你可以使用启动脚本来运行 Java 网关：

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

同样，如果你不再使用 Java 网关了，可以运行关闭脚本将其停止：

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

请注意，Java 网关是轻量应用，不需要数据库，这一点与 Server 和 Proxy 不同。

配置 Zabbix Server 使用 Java 网关

当 Java 网关启动并运行后，如何告诉 Zabbix server 去哪里找到 Zabbix Java 网关呢？通过在server 配置文件中制定 JavaGateway 和 JavaGatewayPort 来完成这个操作。如果运行 JMX 应用程序的主机是由 Zabbix 代理监控的，则可以在proxy 配置文件中指定连接参数。

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

默认情况下，server 不会启动任何与 JMX 监控相关的进程。但是，如果你想用到它，则必须制定 Java pollers 的数量。此操作与指定常规 pollers 和 trappers 相同。

```
StartJavaPollers=5
```

配置完 server 或 proxy 后，一定不要忘记重启 server 或 proxy。

调试 Java 网关

为了防止在 Java gateway 出现任何问题或在 Zabbix 前端看不到详细的报错信息的情况下，您可以通过 Java gateway 日志文件来查看。

默认情况下，Java gateway 将其活动日志记录到日志级别为"info" 的 /tmp/zabbix_java.log 文件中。有时候，该日志信息可能不够详细，需要在日志级别为"debug" 中获取。为了提升日志级别，需要修改 lib/logback.xml 文件，并将 <root> 标记的日志等级属性更改为"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 文件并不需要重启 Zabbix Java gateway，它会自动提交。当完成调试后，可以将日志级别修改回 "info"。

如果希望将日志记录到其他文件或完全不同的介质，如数据库，那么只需要调整 logback.xml 文件。详见Logback 手册 获取更多信息。

有时为了调试，将 Java 网关用作控制台应用而不是守护进程来启动是很有必要的。为此，可以在 settings.sh 中注释掉 PID_FILE 变量。如果省略掉 PID_FILE，则 startup.sh 脚本启动 Java gateway 时会作为控制台应用来启动，并将 Logback 使用 lib/logback-console.xml 文件，这不仅会记录到控制台，还会启用日志级别 "debug"。

最后，请注意，由于 Java gateway 使用 SLF4J 来记录，您可以通过在 lib 目录放置合适的 JAR 文件来将 Logback 替换为您选中的框架。详见 SLF4J 手册 以获取更多信息。

JMX 监控

详见JMX 监控 页面以获取更多信息。

2 从 **RHEL/CentOS** 包安装

概述

如果是通过 RHEL/CentOS 的包进行的安装，那么以下的内容将会帮助您设置Zabbix Java 网关。

配置并运行 JAVA 网关

Zabbix Java 网关的配置参数可以通过如下文件进行调整：

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

关于更多信息，详见 Zabbix Java 网关配置参数。

通过以下命令来启动 Zabbix Java 网关：

```
# service zabbix-java-gateway restart
```

通过以下命令来配置 Zabbix Java 网关的开机自启动：

RHEL 7 和 RHEL 7 之后的系统：

```
# systemctl enable zabbix-java-gateway
```

RHEL 7 之前的系统：

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

配置 ZABBIX SERVER 使用 JAVA 网关

当 Java 网关启动并运行后，如何告诉 Zabbix server 去哪里找到 Zabbix Java 网关呢？通过在server 配置文件中制定 JavaGateway 和 JavaGatewayPort 来完成这个操作。如果运行 JMX 应用程序的主机是由 Zabbix 代理监控的，则可以在proxy 配置文件中指定连接参数。

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

默认情况下，server 不会启动任何与 JMX 监控相关的进程。但是，如果你想用到它，则必须制定 Java pollers 的数量。此操作与指定常规 pollers 和 trappers 相同。

```
StartJavaPollers=5
```

配置完 server 或 proxy 后，一定不要忘记重启 server 或 proxy。

调试 JAVA 网关

Zabbix Java 网关日志路径:

```
/var/log/zabbix/zabbix_java_gateway.log
```

如果要增加日志记录，编辑以下文件：

```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

并将 level="info" 更改为”debug” 或”trace” （为了深度排错）：

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

</configuration>
```

JMX 监控

详见JMX 监控 页面以获取更多信息。

3 从 **Debian/Ubuntu** 包安装

概述

如果是通过 Debian/Ubuntu 的包进行的安装，那么以下的内容将会帮助您设置Zabbix Java 网关。

配置并运行 JAVA 网关

Zabbix Java 网关的配置参数可以通过如下文件进行调整：

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

关于更多信息，详见 Zabbix Java geteway 配置参数。

通过以下命令来启动 Zabbix Java 网关：

```
# service zabbix-java-gateway restart
```

通过以下命令来配置 Zabbix Java 网关的开机自启动：

```
# systemctl enable zabbix-java-gateway
```

配置 ZABBIX SERVER 使用 JAVA 网关

当 Java 网关启动并运行后，如何告诉 Zabbix server 去哪里找到 Zabbix Java 网关呢？通过在server 配置文件中制定 JavaGateway 和 JavaGatewayPort 来完成这个操作。如果运行 JMX 应用程序的主机是由 Zabbix 代理监控的，则可以在proxy 配置文件中指定连接参数。

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

默认情况下，server 不会启动任何与 JMX 监控相关的进程。但是，如果你想用到它，则必须制定 Java pollers 的数量。此操作与指定常规 pollers 和 trappers 相同。

```
StartJavaPollers=5
```

配置完 server 或 proxy 后，一定不要忘记重启 server 或 proxy。

调试 JAVA 网关

Zabbix Java 网关的日志文件为：

```
/var/log/zabbix/zabbix_java_gateway.log
```

如果要增加日志记录，编辑以下文件：

```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

并将 level="info" 更改为”debug” 或”trace” （为了深度排错）：

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

</configuration>
```

JMX 监控

详见JMX 监控 页面以获取更多的信息。

6 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。

7 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]
```

8 JS

概述

zabbix_js 是一个命令行实用程序，可用于嵌入脚本测试。

该程序可执行带有字符串参数的用户自定义脚本并打印结果。脚本的执行是由内嵌的 Zabbix 脚本引擎来完成的。

在编译或执行错误的情况下，zabbix_js 将在 stderr 中打印错误并以代码 1 退出。

用法

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

zabbix_js 可接收如下命令行参数：

```
  -s, --script script-file          指定待执行脚本的文件名。若 '-' 作为文件名时，脚本名由stdin输入。
  -i, --input input-file            指定输入参数的文件名。若 '-' 作为文件名时，脚本名由stdin输入。
  -p, --param input-param           指定输入参数。
  -l, --loglevel log-level          指定日志级别。
  -t, --timeout timeout             指定超时时间（单位：秒）。
  -h, --help                        显示帮助信息。
  -V, --version                     显示版本号。
```

例如:

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

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。

获取 ZABBIX 源代码

有几种获取 Zabbix 源代码的方法：

- 您可以从 Zabbix 官方网站下载发布的稳定版本
- 您可以从 Zabbix 官方网站开发人员页面下载每晚构建
- 您可以从 Git 源代码存储库系统获取最新的开发版本：

```
* 完整存储库的主要位置位于
  [[https://git.zabbix.com/scm/zbx/zabbix.git]]
* 主版本和受支持的版本也映射到Github，网址为
  [[https://github.com/zabbix/zabbix]]
```

必须安装 Git 客户端才能克隆存储库。官方命令行 Git 客户端软件包在发行版中通常称为 git。例如，要在 Debian / Ubuntu 上安装，请运行：

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

要获取所有 Zabbix 源，请转到要放置代码的目录并执行：

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

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。

数据库版本	备注	
Oracle	10g or later	使用 Oracle 作为 Zabbix 后端数据库。
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Zabbix proxy支持SQLite，可以使用SQLite作为Zabbix 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。

服务端

始终需要强制性要求。支持特定功能需要可选要求

要求状	描述	
libpcre	强制性 [P	rl 兼容正则表达式](https://en.wik (PCRE) 支持需要 PCRE 库。命名可能因 GNU / Linux 发行版而异，例如’libpcre3’或’libpcre1’。请注意，您确实需要 PCRE (v8.x)；不使用 PCRE2 (v10.x) 库

要求状	描述
libevent	对于批量指标支持和IPMI监视是必需的。1.4版或更高版本。请注意，对于Zabbix代理，此要求是可选的。
libpthread	IPMI监视支持需要它互斥锁和读写锁支持需要
zlib	压缩支持

要求状	描述
OpenIPMI	可选 I
libssh2	MI 支持时需要 支持 SSH 时需要。1.0 或更高版本支持
fping	ICMP ping items时需要
libcurl	用于 web 监控、VMware 监控、SMTP 认证。对于 SMTP 身份验证，需要 7.20.0 或更高版本。
libiksemel	Elas-tic-search 也需要 Jabber 支持时需要

要求状	描述	
libxml2		VMware 监控时需要支持
net-snmp		SNMP 时需要

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。

Default port numbers

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

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

Database size

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

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

- Number of processed values per second

This is the average number of new values Zabbix server receives every second. For example, if we have 3000 items for monitoring with refresh rate of 60 seconds, the number of values per second is calculated as 3000/60 = **50**.

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

- Housekeeper settings for history

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

So, if we would like to keep 30 days of history and we receive 50 values per second, total number of values will be around (**30***24*3600)* **50** = 129.600.000, or about 130M of values.

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

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

- Housekeeper setting for trends

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

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

- Housekeeper settings for events

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

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

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

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

Note:
[1] More when having non-ASCII event names, tags and values.
[2] The size approximations are based on MySQL and might be different for other databases.

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

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

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

Time synchronization

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

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。

Access control

Principle of least privilege

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

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

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

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

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。

utf-8 编码

UTF-8 是 Zabbix 支持的唯一编码。它可以正常工作而没有任何安全漏洞。用户应注意，如果使用其他一些编码，则存在已知的安全问题

Windows installer paths

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

Zabbix Security Advisories and CVE database

See Zabbix Security Advisories and CVE database.

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
```

Web server hardening

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
```

在 Web 服务器上启用 HTTP 严格传输安全性（HSTS）

为了保护 Zabbix 前端不受协议降级攻击，我们建议在 Web 服务器上启用HSTS策略

例如，要在 Apache 配置中为您的 Zabbix 前端启用 HSTS 策略：

```
/etc/httpd/conf/httpd.conf
```

将以下指令添加到虚拟主机的配置中：

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

重新启动 Apache 服务以应用更改：

```
systemctl restart httpd.service
```

在 Web 服务器上启用内容安全策略（CSP）

为了保护 Zabbix 前端免受跨站点脚本（XSS），数据注入和其他类似类型的攻击的影响，我们建议在 Web 服务器上启用内容安全策略。为此，您需要配置 Web 服务器以返回Content-Security-Policy HTTP 标头

要在 Apache 配置中为您的 Zabbix 前端启用 CSP，请编辑：

```
/etc/httpd/conf/httpd.conf
```

例如，如果计划所有内容都来自站点的来源（不包括子域），则可以将以下指令添加到虚拟主机的配置中：

```
<VirtualHost *:*>
 Header set Content-Security-Policy "default-src 'self';"
```

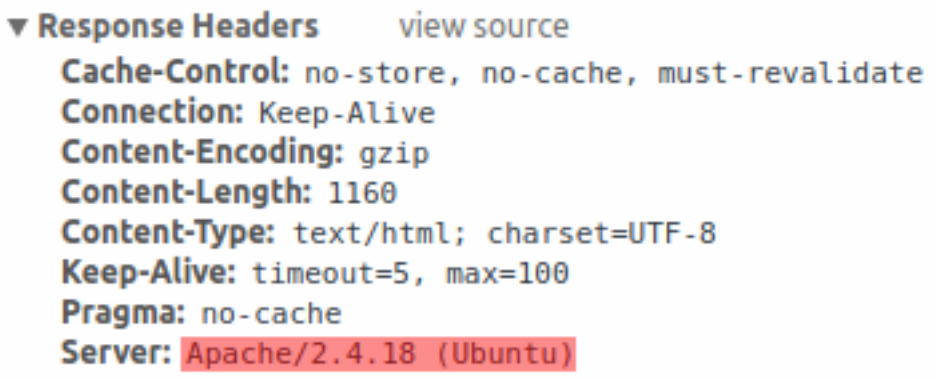
```
</VirtualHost>
```

重新启动 Apache 服务以应用更改：

```
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
```

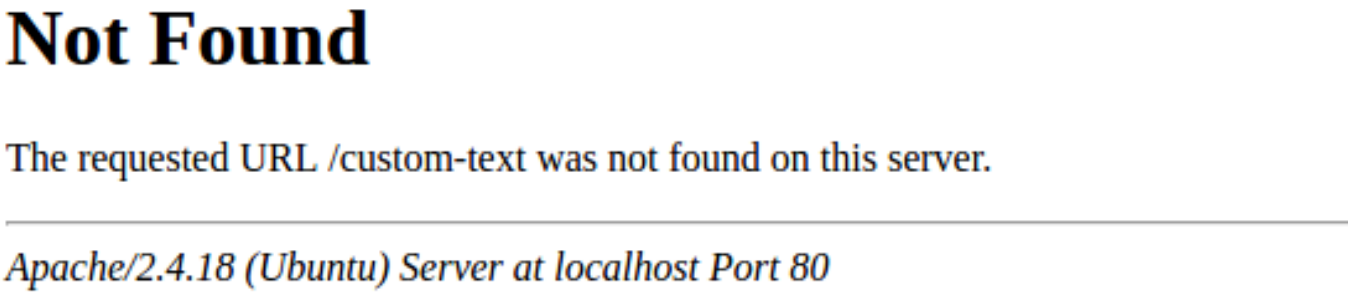
若要应用配置文件更改，需要重新启动 Web 服务器。

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。

禁用默认 Web 服务器错误页面

建议禁用默认错误页面以避免信息泄露。Web 服务器默认使用内置错误页面：



在 Web 服务器强化过程中，应替换/删除默认错误页面。“ ErrorDocument” 指令可用于为 Apache Web 服务器定义自定义错误页面/文本（用作示例）

请参考 Web 服务器的文档以找到有关如何替换/删除默认错误页面的帮助

Set X-Frame-Options HTTP response header

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

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

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

Hiding the file with list of common passwords

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

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

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

On Apache - by using `.htacess` file:

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

Displaying URL content in the sandbox

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

在 sandbox 中显示 URL 内容

从 5.0.2 版开始，一些 Zabbix 前端元素（例如URL 小部件）已预先配置为从 URL 检索的 sandbox 内容。建议保持所有 sandbox 限制处于启用状态，以确保免受 XSS 攻击。

带有 OPENSSL 的在 Windows 上的 zabbix agent

使用 OpenSSL 编译的 Zabbix agent 将尝试在 c：\ openssl-64bit 中访问 SSL 配置文件。磁盘 C：上的 “ openssl-64bit” 目录可以由非特权用户创建

因此，为了加强安全性，需要手动创建此目录并撤消非管理员用户的写访问权限

请注意，在 Windows 的 32 位和 64 位版本上，目录名称将有所不同

Cryptography

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-5.0.0.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”）

在基于 RedHat 的系统上，运行：

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

在基于 Debian 的系统上，运行：

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

<note important>Zabbix 进程不需要主目录，这就是为什么我们不建议创建它的原因。但是，如果您使用某些需要它的功能（例如，将 MySQL 凭据存储在 $ HOME / .my.cnf 中），则可以使用以下命令自由创建它。在基于 RedHat 的系统上，运行：

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

在基于 Debian 的系统上，运行：

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

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
```

或者，Zabbix agent 2：

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

有关编译选项的注意事项：

- 如果使用--enable-agent 选项，则会编译命令行实用程序 zabbix_get 和 zabbix_sender
- 虚拟机监视需要--with-libcurl 和--with-libxml2 配置选项；SMTP 验证和 web.page 也需要--with-libcurl。* Zabbix 代理项目。请注意，使用--with-libcurl 配置选项需要cURL 7.20.0 或更高版本
- Zabbix 始终使用 PCRE 库进行编译（从 3.4.0 版开始）；安装它不是可选的。--with-libpcre = [DIR] 仅允许指向特定的基本安装目录，而不是在多个常用位置中搜索 libpcre 文件
- 您可以使用--enable-static 标志来静态链接库。如果计划在不同的服务器之间分发编译的二进制文件，则必须使用此标志来使这些二进制文件在没有必需的库的情况下工作。请注意，-enable-static 在Solaris中不起作用

- 在构建服务器时，不建议使用--enable-static 选项。为了静态构建服务器，您必须具有所需的每个外部库的静态版本。在配置脚本中没有对此进行严格检查
- 当需要使用不在默认位置的库时，在 MySQL 配置文件--with-mysql = / <path_to_the_file> / mysql_config 中添加可选路径，以选择所需的 MySQL 客户端库。当在同一系统上安装了多个版本的 MySQL 或与 MySQL 一起安装了 MariaDB 时，此功能很有用
- 使用--with-oracle 标志可以指定 OCI API 的位置
- 编译 Zabbix 代理 2 需要 Go 版本 1.13 或更高版本。有关安装说明，请参阅golang.org

<note important> 如果./configure 由于缺少库或其他某些情况而失败，请参阅 config.log 文件以获取有关该错误的更多详细信息。例如，如果缺少 libssl，则立即错误消息可能会引起误解：

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

尽管 config.log 有更详细的描述：

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

- 使用加密支持编译 Zabbix 以支持加密
- 在 HP-UX 上编译 Zabbix agent 的已知问题

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

请参阅Web 界面安装页面以获取有关 Zabbix web 安装向导的信息

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

3 安装 JAVA GATEWAY

仅当您要监视 JMX 应用程序时才需要安装 Java gateway。Java gateway 是轻量级的，不需要数据库

要从源代码安装，请首先下载并解压缩源归档文件

要编译 Java gateway，请使用--enable-java 选项运行./configure 脚本。建议您指定--prefix 选项来请求默认的/ usr / local 以外的安装路径，因为安装 Java gateway 会创建整个目录树，而不仅仅是一个可执行文件

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

要将 Java gateway 编译并打包到 JAR 文件中，请运行 make。请注意，对于此步骤，您将在路径中需要 javac 和 jar 可执行文件

```
$ make
```

现在，您在 src / zabbix_java / bin 中有一个 zabbix-java-gateway- $ VERSION.jar 文件。如果您可以从分发目录中的 src / zabbix_java 运行 Java gateway，那么可以继续阅读有关配置和运行Java gateway的说明。否则，请确保您具有足够的特权并运行 make install

```
$ make install
```

继续安装，了解关于配置和运行 Java gateway 的更多细节

1 在 **Windows** 上安装 **Zabbix agent 2**

概述

本节演示如何从源代码安装 Zabbix agent 2 (Windows)

安装 MinGW 编译器

1. 下载带有 SJLJ(set jump/long jump) 异常处理和 Windows 线程的 MinGW-w64 (例如 *x86_64-8.1.0-release-win32-sjlj-rt_v6-rev0.7z*)
2. 提取并移动到 *c:\mingw*
3. 设置环境变量

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

编译时使用 Windows 提示符而不是 MinGW 提供的 MSYS 终端

编译 PCRE 开发库

下面的说明将编译和安装 64 位的 PCRE 库 *c:\dev\pcre* 和 32 位库 *c:\dev\pcre32*：

1. 从 pcre.org 下载 PCRE 库版本 8.XX (ftp://ftp.pcre.org/pub/pcre/) 并提取
2. 打开 *cmd* 并导航到提取的源

安装 64 位 PCRE

1. 删除旧的配置/缓存（如果存在）：

```
del CMakeCache.txt
rmdir /q /s CMakeFiles
```

2. 运行 cmake (可从 https://cmake.org/download/安装 CMake):

```
cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-O2 -g" -DCMAKE_CXX_FLAGS="-O2 -g" -DCM
```

3. 接下来执行:

```
mingw32-make clean
mingw32-make install
```

安装 32 位 PCRE

1. 执行:

```
mingw32-make clean
```

2. 删除 *CMakeCache.txt*:

```
del CMakeCache.txt
rmdir /q /s CMakeFiles
```

3. 执行 cmake:

```
cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-m32 -O2 -g" -DCMAKE_CXX_FLAGS="-m32 -O
```

4. 接下来执行:

```
mingw32-make install
```

安装 OpenSSL 开发库

1. 从 https://bintray.com/vszakats/generic/openssl/1.1.1d 下载 32 和 64 位版本
2. 分别将文件解压缩到 *c:\dev\openssl32* 和 *c:\dev\openssl* 目录中
3. 之后，删除提取的 **.dll.a* (dll 调用包装库)，因为 MinGW 优先于静态库

编译 Zabbix agent 2

32 位

打开 MinGW 环境（Windows 命令提示符），然后导航到 Zabbix 源代码树中的 *build/mingw* 目录

执行:

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

64 位

打开 MinGW 环境（Windows 命令提示符），然后导航到 Zabbix 源代码树中的 build/mingw 目录

执行:

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

> **Note:**
> 32 位和 64 位版本都可以在 64 位平台上构建，但是只能在 32 位平台上构建 32 位版本。在 32 位平台上工作时，请遵循与 64 位平台上 64 位版本相同的步骤

2 在 macOS 上安装 zabbix agent

总览

本节演示如何从包含或不包含 TLS 的源代码安装 Zabbix agent 二进制文件 (macOS)

先决条件

您将需要命令行开发人员工具（不需要 Xcode），Automake，pkg-config 和 PCRE（v8.x）如果要使用 TLS 构建 agent 二进制文件，则还需要 OpenSSL 或 GnuTLS

要安装 Automake 和 pkg-config，您将需要来自 https://brew.sh/的 Homebrew 软件包管理器。要安装它，请打开终端并运行以下命令：

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

然后安装 Automake 和 pkg-config：

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

准备 PCRE，OpenSSL 和 GnuTLS 库取决于如何将它们链接到 agent

如果打算在已经具有这些库的 macOS 计算机上运行 agent 二进制文件，则可以使用 Homebrew 提供的预编译库。这些通常是使用 Homebrew 来构建 Zabbix agent 二进制文件或用于其他目的的 macOS 计算机

如果 agent 二进制文件将在没有共享库版本的 macOS 计算机上使用，则应从源代码编译静态库并将 Zabbix agent 与它们链接

使用共享库构建 agent 二进制文件

安装 PCRE:

```
$ brew install pcre
```

使用 TLS 构建时，请安装 OpenSSL 和/或 GnuTLS:

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

下载 Zabbix 源码:

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

不使用 TLS 构建 agent:

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

使用 OpenSSL 构建 agent:

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

使用 GnuTLS 构建 agent:

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

使用不带 TLS 的静态库构建 agent 二进制文件

让我们假设 PCRE 静态库将安装在 “$HOME/static-libs” 中。我们将使用 PCRE 8.42

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

下载并构建具有 Unicode 属性支持的 PCRE:

```
$ mkdir static-libs-source
$ cd static-libs-source
$ curl --remote-name https://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
$ tar xf pcre-8.42.tar.gz
$ cd pcre-8.42
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
$ make
$ make check
$ make install
```

下载 Zabbix 源代码并构建 agent：

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$PCRE_PREFIX
$ make
$ make install
```

使用 OpenSSL 构建带有静态库的 agent 二进制文件

构建 OpenSSL 时，建议在成功构建后运行 “make test”。即使构建成功，测试有时也会失败。在这种情况下，应进行研究并解决问题，然后再继续

让我们假设 PCRE 和 OpenSSL 静态库将安装在 “$HOME/static-libs” 中。我们将使用 PCRE 8.42 和 OpenSSL 1.1.1a

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
$ OPENSSL_PREFIX="$HOME/static-libs/openssl-1.1.1a"
```

让我们在 “static-libs-source” 中构建静态库:

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

下载并构建具有 Unicode 属性支持的 PCRE：

```
$ curl --remote-name https://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
$ tar xf pcre-8.42.tar.gz
$ cd pcre-8.42
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
$ make
$ make check
$ make install
$ cd ..
```

下载并构建 OpenSSL：

```
$ curl --remote-name https://www.openssl.org/source/openssl-1.1.1a.tar.gz
$ tar xf openssl-1.1.1a.tar.gz
$ cd openssl-1.1.1a
$ ./Configure --prefix="$OPENSSL_PREFIX" --openssldir="$OPENSSL_PREFIX" --api=1.1.0 no-shared no-capieng n
$ make
$ make test
$ make install_sw
$ cd ..
```

下载 Zabbix 源代码并构建 agent：

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$PCRE_PREFIX
$ make
$ make install
```

使用带有 GnuTLS 的静态库构建 agent 二进制文件

GnuTLS 取决于 Nettle 加密后端和 GMP 算术库。本指南将使用 Nettle 中包含的 mini-gmp，而不是使用完整的 GMP 库

构建 GnuTLS 和 Nettle 时，建议在成功构建后运行 “make check”。即使构建成功，测试有时也会失败。在这种情况下，应进行研究并解决问题，然后再继续

假设 PCRE，Nettle 和 GnuTLS 静态库将安装在 “ $ HOME / static-libs” 中。我们将使用 PCRE 8.42，Nettle 3.4.1 和 GnuTLS 3.6.5

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
$ NETTLE_PREFIX="$HOME/static-libs/nettle-3.4.1"
$ GNUTLS_PREFIX="$HOME/static-libs/gnutls-3.6.5"
```

让我们在 “static-libs-source” 中构建静态库:

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

下载并构建 Nettle：

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

下载并构建 GnuTLS：

```
$ curl --remote-name https://www.gnupg.org/ftp/gcrypt/gnutls/v3.6/gnutls-3.6.5.tar.xz
$ tar xf gnutls-3.6.5.tar.xz
$ cd gnutls-3.6.5
$ PKG_CONFIG_PATH="$NETTLE_PREFIX/lib/pkgconfig" ./configure --prefix="$GNUTLS_PREFIX" --enable-static --d
$ make
$ make check
$ make install
$ cd ..
```

下载 Zabbix 源代码和构建 agent：

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ CFLAGS="-Wno-unused-command-line-argument -framework Foundation -framework Security" \
> LIBS="-lgnutls -lhogweed -lnettle" \
> LDFLAGS="-L$GNUTLS_PREFIX/lib -L$NETTLE_PREFIX/lib" \
> ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$PCRE_PREFIX
$ make
$ make install
```

3 在 **Windows** 上安装 **Zabbix agent**

概述

本节演示如何从包含或不包含 TLS 的源代码安装 Zabbix agent 二进制文件 (Windows)

安装 OpenSSL

以下步骤将帮助您从 MS Windows 10（64 位）上的源代码编译 OpenSSL

1. 要安装 OpenSSL，您将需要在 Windows 计算机上：
    1. C compiler (e.g. VS 2017 RC),
    2. NASM (https://www.nasm.us/),
    3. Perl (e.g. Strawberry Perl from http://strawberryperl.com/),
    4. Perl module Text::Template (cpan Text::Template).
2. 从 https://www.openssl.org/获取 OpenSSL 源。这里使用 OpenSSL 1.1.1
3. 解压缩 OpenSSL 源，例如在 E:\openssl-1.1.1
4. 打开命令行窗口，例如 VS 2017 RC 的 x64 本机工具命令提示符
5. 转到 OpenSSL 源目录，例如 E:\openssl-1.1.1
    1. 验证是否可以找到 NASM:`e:\openssl-1.1.1> nasm --version  NASM version 2.13.01 compiled on May  1 2017`
6. 例如，配置 OpenSSL:`e:\openssl-1.1.1> perl E:\openssl-1.1.1\Configure VC-WIN64A no-shared no-capieng no-srp no-gost no-dgram no-dtls1-method no-dtls1_2-method  --api=1.1.0 --prefix=C:\OpenSSL-Win64-111- --openssldir=C:\OpenSSL-Win64-111-static`
    - 注意选项 “ no-shared”：如果使用 “ no-shared”，则 OpenSSL 静态库 libcrypto.lib 和 libssl.lib 将是 “self-sufficient”，并且所产生的 Zabbix 二进制文件本身将包括 OpenSSL，而无需外部 OpenSSL DLLs。优点：Zabbix 二进制文件无需 OpenSSL 库即可复制到其他 Windows 计算机。缺点：发布新的 OpenSSL 错误修正版本时，Zabbix agent 需要重新编译并重新安装
    - 如果不使用 “ no-shared”，则静态库 libcrypto.lib 和 libssl.lib 将在运行时使用 OpenSSL DLLs。优点：发布新的 OpenSSL 错误修正版本时，可能无需升级 Zabbix agent 即可仅升级 OpenSSL DLLs。缺点：将 Zabbix agent 复制到另一台计算机也需要复制 OpenSSL DLLs
7. 编译 OpenSSL，运行安装，测试:`e:\openssl-1.1.1> nmake  e:\openssl-1.1.1> nmake test  ... All tests successful.  Files=152, Tests=1152, 501 wallclock secs ( 0.67 usr +  0.61 sys = 1.28 CPU)  Result: PASS  e:\openssl-1.1.1> nmake install_sw`’install_sw’ 仅安装软件组件（即库，头文件，但没有文档）。如果需要所有内容，请使用 “ nmake install”

编译 PCRE

1. 从 pcre.org 8.XX 版下载 PCRE 库（自 Zabbix 4.0 起为强制性库）；不是 pcre2（ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.41.zip）
2. 提取到目录 *E:\pcre-8.41*
3. 从 https://cmake.org/download/安装 CMake，在安装过程中选择：并确保 cmake\bin 在您的路径上（经过测试的版本 3.9.4）
4. 创建一个新的空构建目录，最好是源目录的子目录。例如，*E:\pcre-8.41\build*
5. 打开命令行窗口，例如 VS 2017 的 x64 本机工具命令提示符，并从该 Shell 环境运行 cmake-gui。不要尝试从 Windows“开始” 菜单启动 Cmake，因为这可能会导致错误
6. 输入 *E:\pcre-8.41* 和 *E:\pcre-8.41\build* 作为源目录
7. 点击 “Configure” 按钮
8. 为该项目指定生成器时，选择 “ NMake Makefiles”
9. 创建一个新的空安装目录。例如，*E:\pcre-8.41-install*
10. 然后，GUI 将列出几个配置选项。确保选择以下选项：
    - **PCRE_SUPPORT_UNICODE_PROPERTIES** ON
    - **PCRE_SUPPORT_UTF** ON
    - **CMAKE_INSTALL_PREFIX** *E:\pcre-8.41-install*
11. 再次点击 “Configure”。相邻的 “Generate” 按钮现在应该处于 active 状态。
12. 点击 “Generate”
13. 如果发生错误，建议您在尝试重复 CMake 构建过程之前删除 CMake 缓存。在 CMake GUI 中，可以通过选择 “File > Delete Cache” 来删除缓存
14. 现在，构建目录应该包含一个可用的构建系统-Makefile
15. 打开命令行窗口，例如 VS 2017 的 x64 本机工具命令提示符，并导航到上面提到的 Makefile
16. 运行 NMake 命令: `E:\pcre-8.41\build> nmake install`

编译 Zabbix

以下步骤将帮助您从 MS Windows 10（64 位）上的源代码编译 Zabbix。当使用/不支持 TLS 编译 Zabbix 时，唯一的不同是在步骤 4 中

1. 在 Linux 机器上，检查 GIT 的来源:`$ git clone https://git.zabbix.com/scm/zbx/zabbix.git  $ cd zabbix/  $ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available $ ./bootstrap.sh  $ ./configure --enable-agent --enable-ipv6 --prefix=`pwd`  $ make dbschema  $ make dist`
2. 复制并解压缩存档, 例如 Windows 机器上为 zabbix-5.0.0.tar.gz
3. 假设源位于 e:\zabbix-5.0.0 中。打开命令行窗口，例如 VS 2017 RC 的 x64 本机工具命令提示符。转到 E:\zabbix-5.0.0\build\win32\project
4. 编译 zabbix_get，zabbix_sender 和 zabbix_agent
    - 不使用 TLS: `E:\zabbix-5.0.0\build\win32\project> nmake /K PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`

- 使用TLS: E:\zabbix-5.0.0\build\win32\project> nmake /K -f Makefile_get TLS=openssl TLSINCDIR=C:\Ope TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\p E:\zabbix-5.0.0\build\win32\project> nmake /K -f Makefile_sender TLS=openssl TLSINCDIR="C:\OpenSS TLSLIBDIR="C:\OpenSSL-Win64-111-static\lib" PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib E:\zabbix-5.0.0\build\win32\project> nmake /K -f Makefile_agent TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib

5. 新的二进制文件位于 e:\zabbix-5.0.0\bin\win64 中。由于 OpenSSL 是使用 “ no-shared” 选项编译的，因此 Zabbix 二进制文件本身包含 OpenSSL，并且可以将其复制到其他没有 OpenSSL 的计算机上

使用 LibreSSL 编译 Zabbix

该过程类似于使用 OpenSSL 进行编译，但是您需要对 build\win32\project 目录中的文件进行一些小的更改:

* 在 ''Makefile_tls'' 中删除 ''/DHAVE_OPENSSL_WITH_PSK''. 即找到 <code>

CFLAGS = $(CFLAGS) /DHAVE_OPENSSL /DHAVE_OPENSSL_WITH_PSK</code> 并替换为 CFLAGS = $(CFLAGS) /DHAVE_OPENSSL

* 在 ''Makefile_common.inc'' 中添加 ''/NODEFAULTLIB:LIBCMT''. 即找到 <code>

/MANIFESTUAC:”level=’asInvoker’ uiAccess=’false’” /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME).pdb</code> 并替换为 /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME).pdb /NODEFAULTLIB:LIBCMT

4 从二进制包安装

使用 ZABBIX 官方存储库

Zabbix SIA 提供官方 RPM 和 DEB 软件包:

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

yum/dnf, apt 和 zypper 各种 OS 发行版的软件包文件可以在repo.zabbix.com上找到

注意，尽管一些 OS 发行版 (特别是基于 debian 的发行版) 提供了它们自己的 Zabbix 包，但 Zabbix 不支持这些包。第三方提供的 Zabbix 包可能已经过时，可能缺乏最新的特性和 bug 修复。建议只使用repo.zabbix.com上的官方软件包。如果您以前使用过非官方的 Zabbix 包，请参阅关于从操作系统存储库升级 Zabbix 包的说明

1 Red Hat Enterprise Linux/CentOS

概述

Zabbix 官方包可用于: RHEL 8, CentOS 8 and Oracle Linux 8 下载 RHEL 7, CentOS 7 and Oracle Linux 7 下载

软件包可以使用 MySQL/PostgreSQL 数据库和 Apache/Nginx webserver 支持 RHEL 6和RHEL 5也可以使用 Zabbix agent 包和实用程序 Zabbix get 和 Zabbix sender

Zabbix 官方存储库还提供 fping，iksemel，libssh2 软件包。这些软件包位于不支持的目录中

安装注意事项

请参阅下载页面中每个平台的安装说明，以了解：

- 安装存储库
- 安装 server/agent/frontend
- 创建初始数据库，导入初始数据
- 为 Zabbix server 配置数据库
- 为 Zabbix frontend 配置 PHP
- 启动 server/agent 进程
- 配置 Zabbix frontend

如果要以 root 用户身份运行 Zabbix agent，请参阅以 root 用户身份运行 agent

使用 TIMESCALE DB 导入数据

使用 TimescaleDB，除了 PostgreSQL 的 import 命令外，还运行：

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

Warning:
仅 Zabbix server 支持 TimescaleDB

frontend 安装要求

Zabbix frontend 需要基本安装中不提供的其他软件包。您需要在将运行 Zabbix frontend 的系统中启用可选 rpm 的存储库：RHEL 7:

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

Note:
请注意，Nginx for RHEL 在 Red Hat Software Collections 和 EPEL 中可用。如果使用 Red Hat Software Collections，只需安装 zabbix-nginx-conf-scl 软件包

SELINUX 配置

在强制模式下启用 SELinux 状态后，您需要执行以下命令以启用 Zabbix frontend 与 server 之间的通信：RHEL 7 及更高版本：

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

如果可以通过网络访问数据库（在 PostgreSQL 中包括 “ localhost”），则还需要允许 Zabbix frontend 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
```

完成 frontend 和 SELinux 配置后，重新启动 Apache Web 服务器：

```
# service httpd restart
```

代理安装

添加所需的存储库后，可以通过运行以下命令来安装 Zabbix agent：

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

在命令中用 “ pgsql” 代替 “ mysql” 以使用 PostgreSQL，或用 “ sqlite3” 代替以使用 SQLite3（仅代理）

创建数据库

为 Zabbix proxy创建一个单独的数据库

Zabbix server 和 Zabbix proxy 不能使用相同的数据库。如果它们安装在同一主机上，则 proxy 数据库必须具有不同的名称

导入数据

导入初始架构：

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

对于 PostgreSQL（或 SQLite）代理：

```
# zcat /usr/share/doc/zabbix-proxy-pgsql*/schema.sql.gz | sudo -u zabbix psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3*/schema.sql.gz | sqlite3 zabbix.db
```

为 ZABBIX PROXY 配置数据库

编辑 zabbix_proxy.conf：

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

在 DBName for Zabbix proxy 中，请使用与 Zabbix server 不同的数据库在 DBPassword 中，为 MySQL 使用 Zabbix 数据库密码；PostgreSQL 的 PostgreSQL 用户密码在 PostgreSQL 中使用 DBHost =。您可能想要保留默认设置 DBHost = localhost（或 IP 地址），但这将使 PostgreSQL 使用网络套接字连接到 Zabbix。有关说明，请参见 SELinux 配置

启动 ZABBIX PROXY 过程

要启动 Zabbix proxy 进程并使其在系统启动时启动：

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

前端配置

Zabbix proxy 没有前端。它仅与 Zabbix server 通信

JAVA 网关安装

仅当您要监视 JMX 应用程序时才需要安装Java 网关。Java 网关是轻量级的，不需要数据库添加所需的存储库后，您可以通过运行以下命令来安装 Zabbix Java 网关：

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

继续进行设置，以获取有关配置和运行 Java 网关的更多详细信息

安装调试信息包

> **Note:**
> Debuginfo 软件包当前可用于 RHEL / CentOS 版本 7、6 和 5

要启用 debuginfo 存储库，请编辑/etc/yum.repos.d/zabbix.repo 文件。对于 zabbix-debuginfo 存储库，将 enabled = 0 更改为 enabled = 1

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo - $basearch
baseurl=http://repo.zabbix.com/zabbix/5.0/rhel/7/$basearch/debuginfo/
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
gpgcheck=1
```

这将允许您安装 zabbix-debuginfo 软件包

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

该单个软件包包含所有 Zabbix 二进制组件的调试信息

Installing debuginfo packages

> **Note:**
> Debuginfo packages are currently available for RHEL/CentOS versions 7, 6 and 5.

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

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo - $basearch
baseurl=http://repo.zabbix.com/zabbix/5.0/rhel/7/$basearch/debuginfo/
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
gpgcheck=1
```

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

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

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

2 DEBIAN/UBUNTU/RASPBIAN

概述

官方 Zabbix 发行包适用于：

Debian 10 (Buster)	Download
Debian 9 (Stretch)	Download
Debian 8 (Jessie)	Download
Ubuntu 20.04 (Focal Fossa) LTS	Download
Ubuntu 18.04 (Bionic Beaver) LTS	Download
Ubuntu 16.04 (Xenial Xerus) LTS	Download

Ubuntu 14.04 (Trusty Tahr) LTS	Download
Raspbian (Buster)	Download
Raspbian (Stretch)	Download

软件包提供了 MySQL/PostgreSQL 数据库和 Apache/Nginx webserver 支持。

安装注意事项

请参阅下载页中每个平台的安装说明：

- 安装存储库
- 安装 server/agent/前端
- 创建初始数据库，导入初始数据
- 为 Zabbix server 配置数据库
- 为 Zabbix 前端配置 PHP
- 启动 server/agent 进程
- 配置 Zabbix 前端

仅 Debian9/10 和 Ubuntu 18.04/20.04 支持 Zabbix agent 2（zabbix-agent2）。

如果要以 root 用户运行 Zabbix agent，请参阅以root 用户运行 agent。

基于 Debian 的发行版通常在其存储库中提供自己的 Zabbix 包。Zabbix 不支持这些包, 仅支持 Zabbix官方存储库的包。

使用 TIMESCALE DB 导入数据

使用 TimescaleDB，除了 PostgreSQL 的导入命令外，还需运行：

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

<note Warning>TimescaleDB 仅支持 Zabbix server。

PHP 7.2

从 Zabbix 5.0 开始，Zabbix 前端需要 PHP7.2 或更高版本。

请参阅有关在 7.2 以下 PHP 版本上安装 Zabbix 前端的说明。

SELINUX 配置

请参阅 RHEL/CentOS 的SELinux 配置。

完成前端和 SELinux 配置后，重新启动 Apache Web 服务器：

```
# service apache2 restart
```

Proxy 安装

添加所需的存储库后，可以通过运行以下命令来安装 Zabbix proxy ：

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

使用 PostgreSQL，将命令中的 “mysql” 替换为 “pgsql”，使用 sqlite3，将命令中的 “mysql” 替换为 “sqlite3”。

创建数据库

为 Zabbix Proxy创建一个单独的数据库。

Zabbix server 和 Zabbix proxy 不能使用相同的数据库。如果它们安装在同一主机，proxy 数据库必须有一个不同的名字。

导入数据

导入初始 schema

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

使用 PostgreSQL (或者 SQLite) 的 proxy:

```
# zcat /usr/share/doc/zabbix-proxy-pgsql/schema.sql.gz | sudo -u zabbix psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3/schema.sql.gz | sqlite3 zabbix.db
```

为 ZABBIX PROXY 配置数据库

编辑 zabbix_proxy.conf:

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

Zabbix proxy 的 DBName 使用与 Zabbix server 不同的数据库。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。请参阅 RHEL/CentOS 的相应部分有关说明。

启动 ZABBIX PROXY 进程

运行以下命令启动 Zabbix proxy 进程，并使其开机自启:

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

前端配置

Zabbix proxy 没有前端; 它仅与 Zabbix server 通信。

JAVA GATEWAY 安装

仅当您要监视 JMX 应用程序时才需要安装Java 网关。Java 网关是轻量级的，不需要数据库。

添加所需的存储库后，您可以通过运行以下命令来安装 Zabbix Java 网关：

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

继续进行设置，以获取有关配置和运行 Java 网关的更多详细信息。

3 SUSE Linux Enterprise Server

概述

官方 Zabbix 安装包适用于:

SUSE Linux Enterprise Server 15	Download
SUSE Linux Enterprise Server 12	Download

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。

SLES 15:

```
# rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/5.0/sles/15/x86_64/zabbix-release-5.0-1.el15.noarc
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

SLES 12:

```
# rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/5.0/sles/12/x86_64/zabbix-release-5.0-1.el12.noarc
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

Server/前端/agent 安装

要安装 Zabbix 前端，必须激活 `web-scripting` 模块。它包含必要的 PHP 依赖项。

SLES 15:

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

SLES 12:

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

安装支持 MySQL 的 Zabbix server/前端/agent：

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

如果使用 nginx web server，则将命令中的 “apache” 替换为 “nginx”。另请参见：SLES 12/15 Zabbix nginx 设置。

如果使用 zabbix agent 2（仅支持 SLES 15 SP1+），则将命令中的 “zabbix-agent” 替换为 “zabbix-agent2”。

安装支持 MySQL 的 Zabbix proxy ：

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

使用 PostgreSQL, 将命令中的 “mysql” 替换为 “pgsql”。

创建数据库

Zabbix server和proxy守护进程需要数据库。Zabbix agent不需要。

> **Warning:**
> Zabbix server 和 Zabbix proxy 需要单独的数据库，他们不能使用相同的数据库。因此如果它们安装在同一主机上，则必须创建不同名称的数据库！

使用MySQL或PostgreSQL提供的说明创建数据库。

导入数据

现在使用 MySQL 导入 ** server ** 的初始 schema 和数据：

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

系统将提示您输入新创建数据库的密码。

使用 PostgreSQL：

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

使用 TimescaleDB，除了前面的命令外，还需运行：

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

> **Warning:**
> TimescaleDB 仅支持 Zabbix server。

导入初始 **proxy** schema:

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

proxy 使用 PostgreSQL:

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

为 ZABBIX SERVER/PROXY 配置数据库

编辑/etc/zabbix/zabbix_server.conf（和 zabbix_proxy.conf）使用各自的数据库。例如：

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

DBPassword 处 MySQL 使用 Zabbix 数据库密码；PosgreSQL 使用 PosgreSQL 用户密码。

PostgreSQL 使用 `DBHost=`。您可能希望保留默认设置 “DBHost=localhost”（或 IP 地址），但这将使 PostgreSQL 使用网络套接字连接到 Zabbix。

Zabbix 前端配置

根据使用的 web server（Apache/Nginx），编辑 Zabbix 前端相应的配置文件：

- Apache 配置文件位于`/etc/apache2/conf.d/zabbix.conf`。已经配置了一些 PHP 设置。但是有必要取消” date.timezone ” 设置的注释，并为您设置正确的时区。

```
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 max_input_vars 10000
```

```
php_value always_populate_raw_post_data -1
# php_value date.timezone Europe/Riga
```

- zabbix-nginx-conf 软件包为 Zabbix 前端安装了单独的 Nginx server 。其配置文件位于/etc/nginx/conf.d/zabbix.conf。为了使 Zabbix 前端正常工作，必须取消注释并设置 “ listen” 和 “server_name” 指令。

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

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

它的配置文件位于/etc/php7/fpm/php-fpm.d/zabbix.conf。已经配置了一些 PHP 设置。但是有必要为您配置正确的时区。

```
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[max_input_vars] = 10000
; php_value[date.timezone] = Europe/Riga
```

现在您可以继续进行前端安装步骤，这将允许您访问新安装的 Zabbix。

请注意，Zabbix proxy 没有前端；它只与 Zabbix server 通信。

启动 ZABBIX SERVER/AGENT 进程

启动 Zabbix server 和 agent 进程，并使其在系统启动时启动。

使用 Apache web server:

```
# systemctl restart zabbix-server zabbix-agent apache2 php-fpm
# systemctl enable zabbix-server zabbix-agent apache2 php-fpm
```

使用 Nginx web server, 用’nginx’ 代替’apache2’。

安装 debuginfo 包

编辑*/etc/zypp/repos.d/zabbix.repo* 文件启用 debuginfo 存储库

将 zabbix debuginfo 存储库的 enabled=0 更改为 enabled=1。

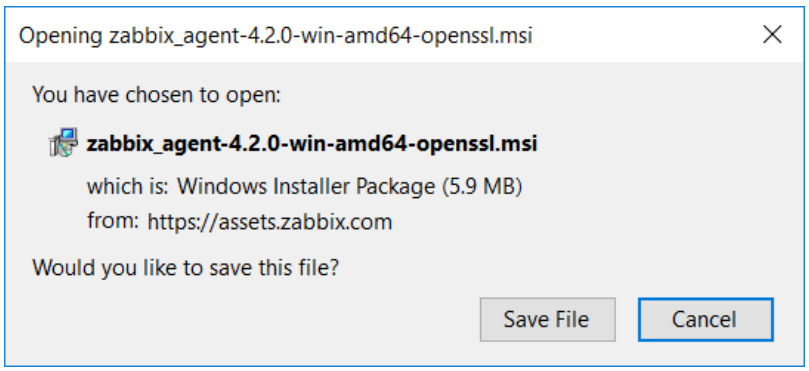
```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo
type=rpm-md
baseurl=http://repo.zabbix.com/zabbix/5.0/sles/15/x86_64/debuginfo/
gpgcheck=1
gpgkey=http://repo.zabbix.com/zabbix/5.0/sles/15/x86_64/debuginfo/repodata/repomd.xml.key
enabled=0
update=1
```

这将允许您安装 zabbix-***<component>***-debuginfo 包。

4 通过 MSI 安装 Windows agent

概述

Zabbix Windows agent 可通过下载的 Windows MSI 安装包 (32 位或者 64 位) 安装。



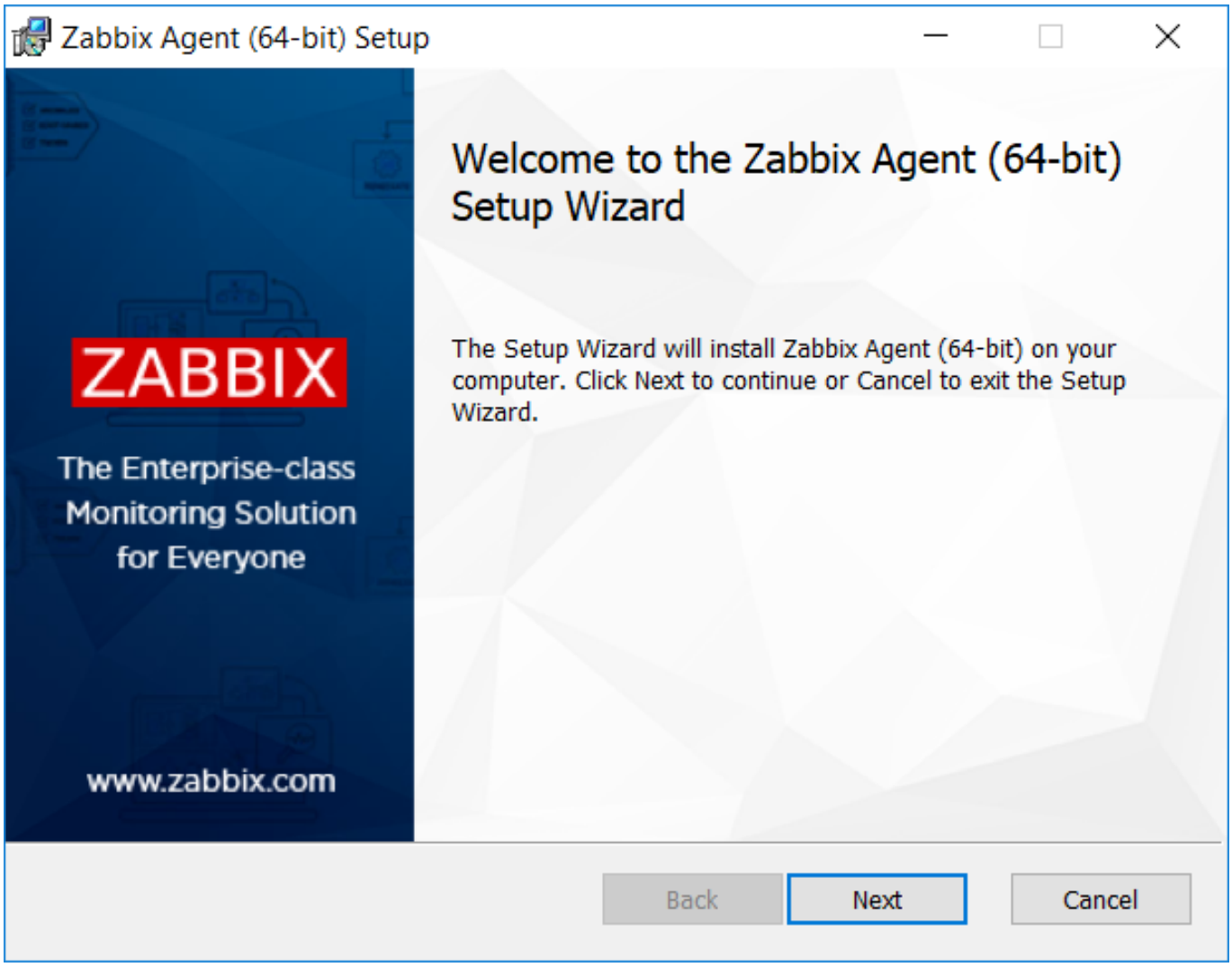
32 位软件包不能安装在 64 位 Windows 上。

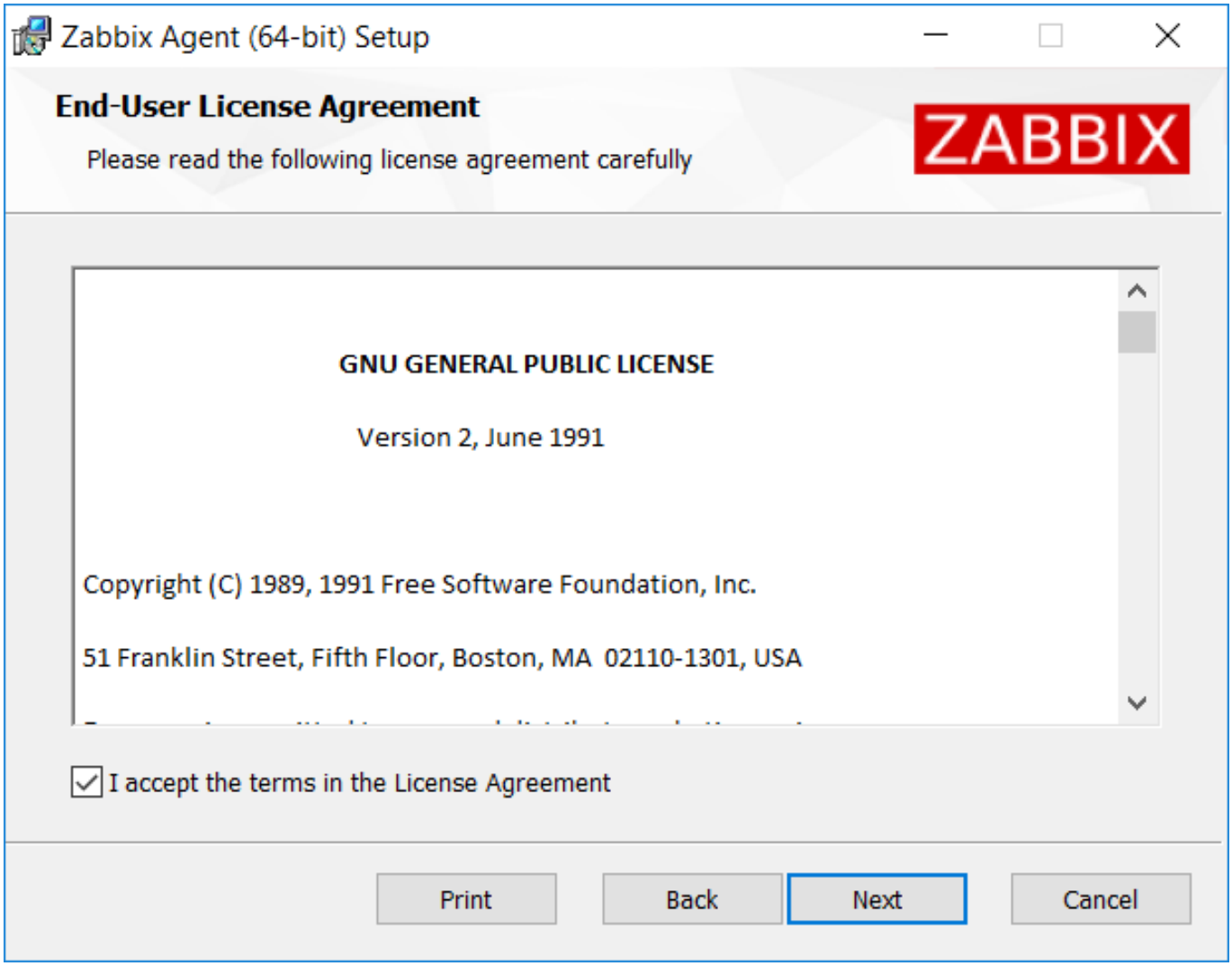
所有软件包都支持 TLS，但是配置 TLS 是可选的。

支持基于 UI 和命令行的安装。

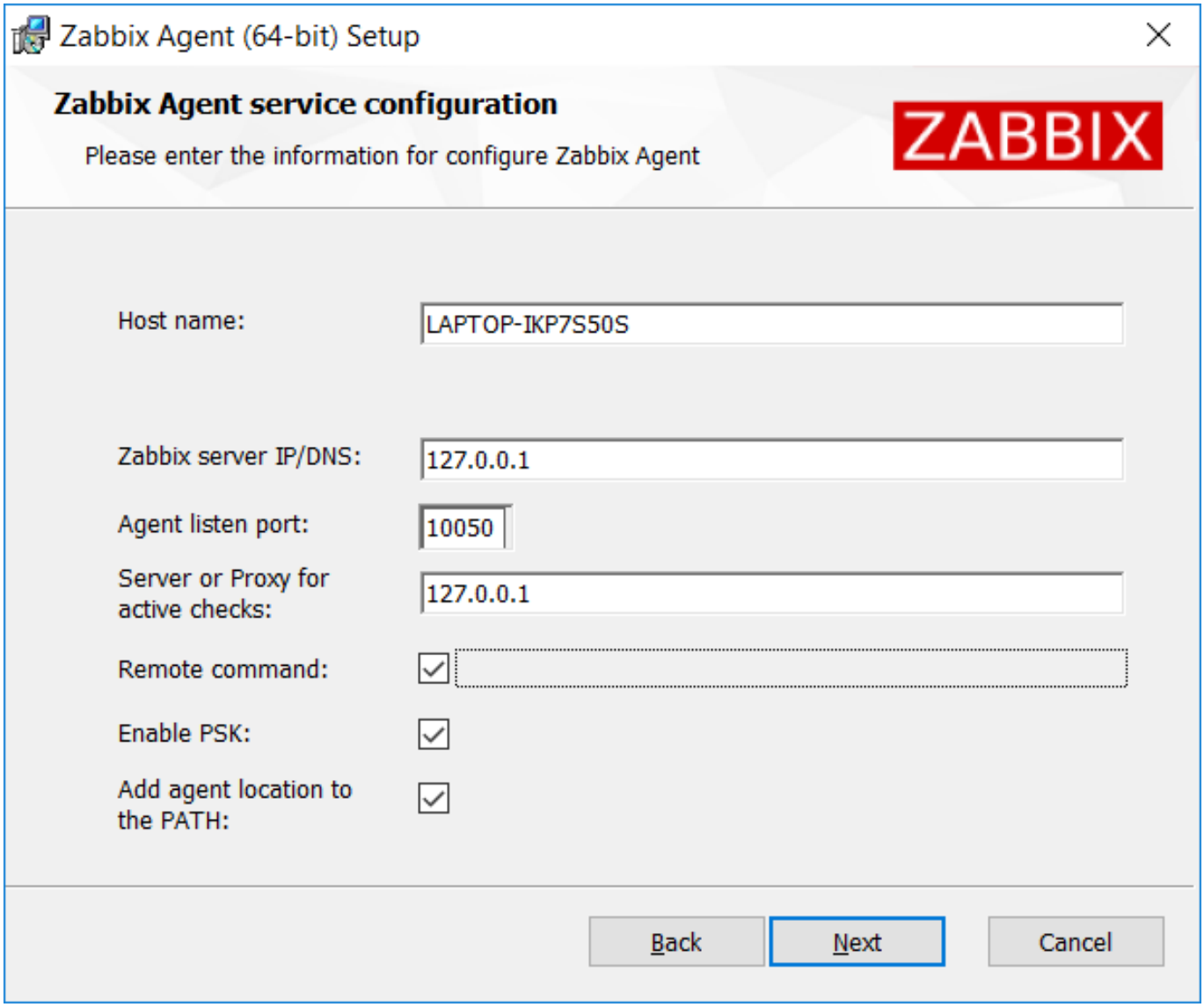
安装步骤

双击下载的 MSI 文件进行安装。



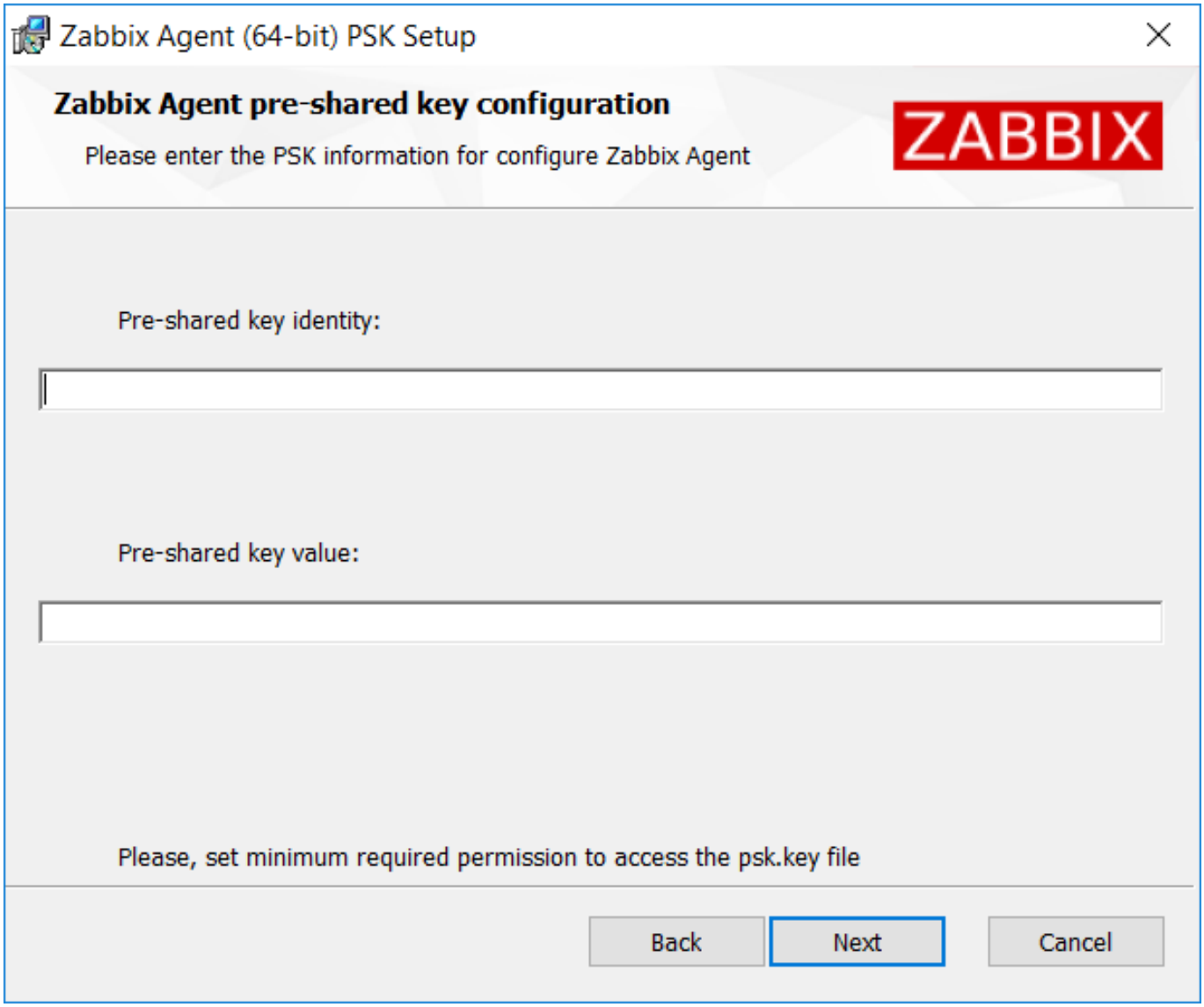

Zabbix Agent (64-bit) Setup
End-User License Agreement
Please read the following license agreement carefully
ZABBIX
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA
I accept the terms in the License Agreement
Print
Back
Next
Cancel

接受许可证点击下一步。

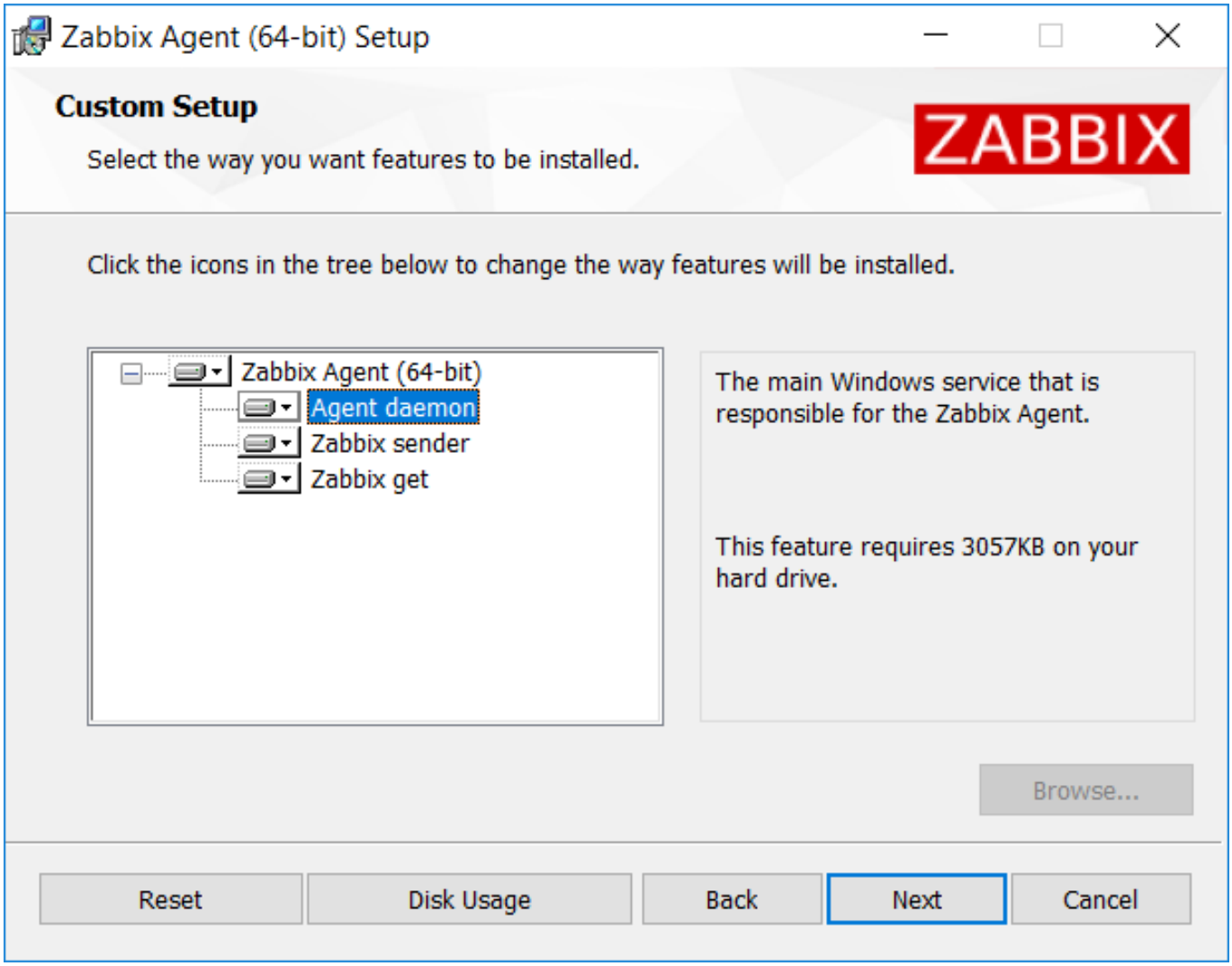


指定以下参数。

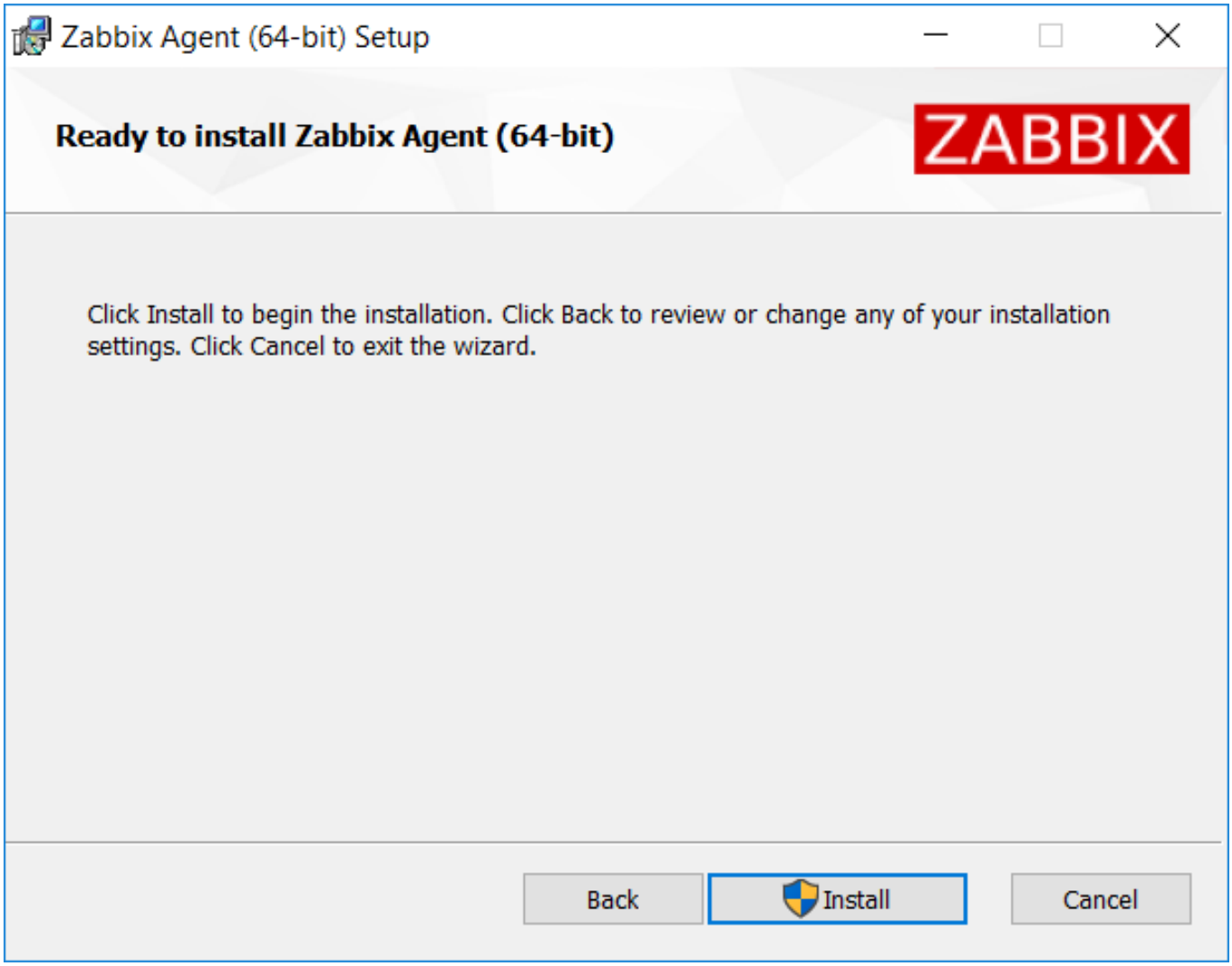
参数	描
Host name	指定主机名。
Zabbix server IP/DNS	指定 Zabbix server 的 IP/DNS。
Agent listen port	指定 Agent 侦听端口（默认为 10050）。
Server or Proxy for active checks	为主动式 agent 指定 Zabbix server/proxy 的 IP/DNS
Remote commands	选中复选框启用远程命令。
Enable PSK	选中复选框通过预共享密钥启用 TLS 支持。
Add agent location to the PATH	将 agent 位置添加到 PATH 变量。



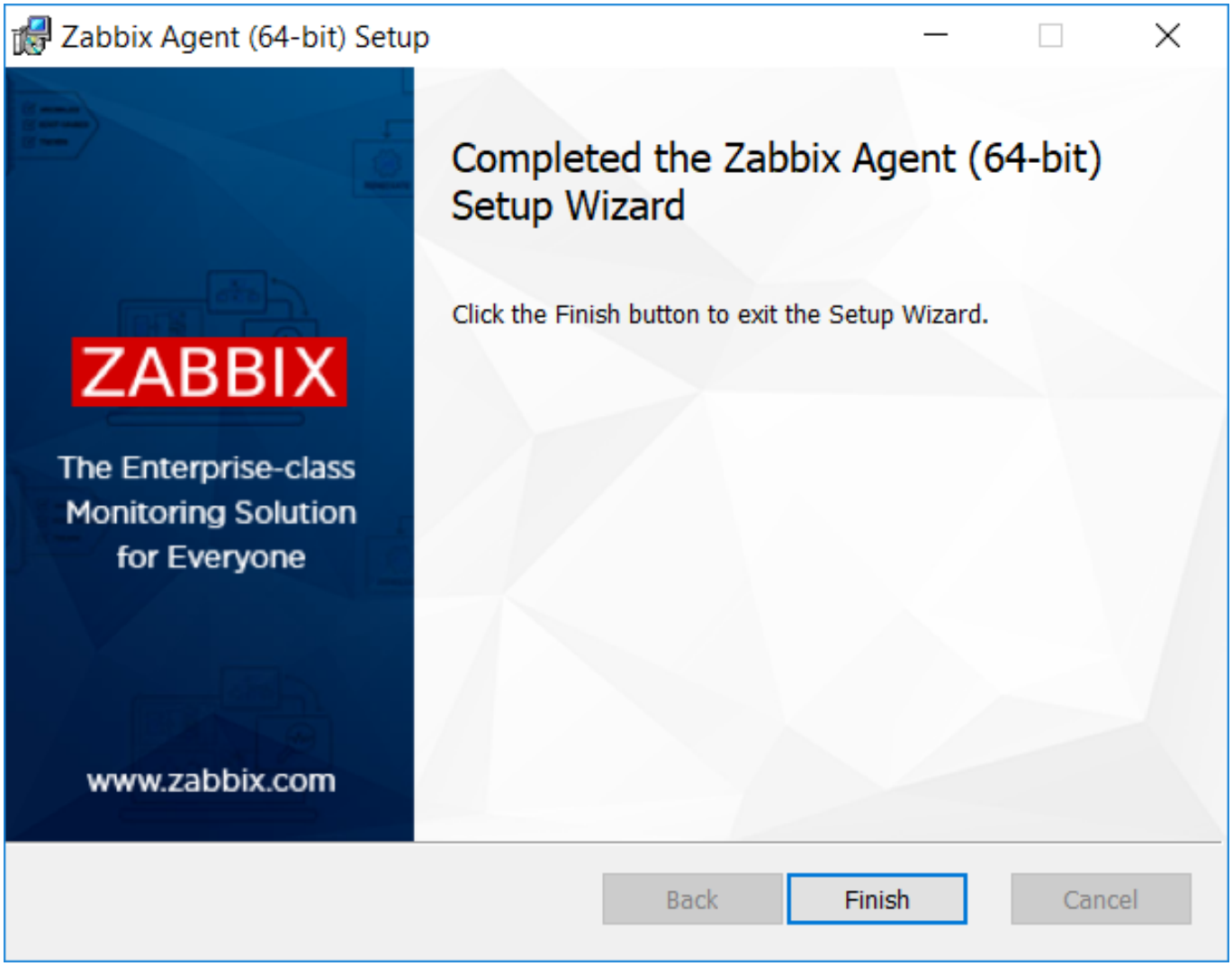
输入预共享密钥标识和值。只有在上一步中选中启用 *PSK* 时，此步骤才可用。



选择要安装的 Zabbix 组件 - Zabbix agent daemon, Zabbix sender, Zabbix get.



Zabbix 组件和配置文件安装在 Program Files 下 Zabbix Agent 文件夹中。zabbix_agentd.exe 将设置为自动启动的 Windows 服务。



基于命令行安装

支持的参数

MSI 安装支持以下参数集：

Number	参数描	
1	LOGTYPE	
2	LOGFILE	
3	ENABLEREMOTECOMMANDS	
4	SERVER	
5	LISTENPORT	
6	SERVERACTIVE	
7	HOSTNAME	
8	TIMEOUT	
9	TLSCONNECT	
10	TLSACCEPT	
11	TLSPSKIDENTITY	
12	TLSPSKFILE	
13	TLSPSKVALUE	
14	TLSCAFILE	
15	TLSCRLFILE	
16	TLSSERVERCERTISSUER	
17	TLSSERVERCERTSUBJECT	
18	TLSCERTFILE	
19	TLSKEYFILE	
20	INSTALLFOLDER	
21	ENABLEPATH	
22	SKIP	SKIP=fw - 不安装防火墙例外规则

运行如下命令安装:

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

msiexec /l*v log.txt /i zabbix_agent-4.0.6-x86.msi /qn^
 LOGTYPE=file^
 LOGFILE="%INSTALLFOLDER%\za.log"^
 ENABLEREMOTECOMMANDS=1^
 SERVER=192.168.6.76^
 LISTENPORT=12345^
 SERVERACTIVE=::1^
 HOSTNAME=myHost^
 TLSCONNECT=psk^
 TLSACCEPT=psk^
 TLSPSKIDENTITY=MyPSKID^
 TLSPSKFILE="%INSTALLFOLDER%\mykey.psk"^
 TLSCAFILE="c:\temp\f.txt1"^
 TLSCRLFILE="c:\temp\f.txt2"^
 TLSSERVERCERTISSUER="My CA"^
 TLSSERVERCERTSUBJECT="My Cert"^
 TLSCERTFILE="c:\temp\f.txt5"^
 TLSKEYFILE="c:\temp\f.txt6"^
 ENABLEPATH=1^
 INSTALLFOLDER="%INSTALLFOLDER%"
 SKIP=fw
```

或者

```
msiexec /l*v log.txt /i zabbix_agent-4.4.0-x86.msi /qn^
 SERVER=192.168.6.76^
 TLSCONNECT=psk^
 TLSACCEPT=psk^
 TLSPSKIDENTITY=MyPSKID^
 TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

5 从 **PKG** 安装 **MAC OS** 代理

概述

Zabbix Mac OS 代理可以使用 PKG 包进行安装，PKG 包可以从如下地址下载 下载. 加密版本和不加密版本均可以下载.

代理安装

代理可以使用图形用户界面方式或者命令行方式，例如:

```
sudo installer -pkg zabbix_agent-4.4.1-macos-amd64-openssl.pkg -target /
```

请保证在命令行中使用正确版本的 Zabbix 安装包版本。在命令行中，pkg 包的名字务必匹配所下载的安装包的名字。

代理运行

在安装完成或者系统重启后，代理会自动启动。

有需要的情况下，您可以编辑相关的配置文件`/usr/local/etc/zabbix/zabbix_agentd.conf` 。

如果需要人工启动代理，执行如下命令:

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

如果需要人工停止代理，执行如下命令:

```
sudo launchctl stop com.zabbix.zabbix_agentd
```

在升级过程中，现有的配置文件不会被覆盖，系统会生成一个新的配置文件，新的配置文件用于检查和更新现有的配置文件。在对配置文件作出任何修改后，必须重启代理才能够生效。

故障排除和删除代理

以下部分列出了许多非常有用的命令，这些命令可以用于故障排除和删除 Zabbix 代理。

查看 Zabbix 代理是否在运行:

```
ps aux | grep zabbix_agentd
```

查看 Zabbix 代理是否使用 PKG 包方式进行安装:

```
$ pkgutil --pkgs | grep zabbix
com.zabbix.pkg.ZabbixAgent
```

查看安装 Zabbix 代理后，有哪些文件被安装在系统中 (注意：每行开头的/并没在以下示例中进行显示):

```
$ pkgutil --only-files --files com.zabbix.pkg.ZabbixAgent
Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
usr/local/bin/zabbix_get
usr/local/bin/zabbix_sender
usr/local/etc/zabbix/zabbix_agentd/userparameter_examples.conf.NEW
usr/local/etc/zabbix/zabbix_agentd/userparameter_mysql.conf.NEW
usr/local/etc/zabbix/zabbix_agentd.conf.NEW
usr/local/sbin/zabbix_agentd
```

如果 Zabbix 的代理使用 `launchctl` 方式启动，您可以使用如下命令停止 Zabbix 代理:

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

删除已安装文件 (包括配置文件和相关日志文件)，使用以下命令:

```
sudo rm -f /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
sudo rm -f /usr/local/sbin/zabbix_agentd
sudo rm -f /usr/local/bin/zabbix_get
sudo rm -f /usr/local/bin/zabbix_sender
sudo rm -rf /usr/local/etc/zabbix
sudo rm -rf /var/logs/zabbix
```

忘记已安装 Zabbix 代理，使用命令:

```
sudo pkgutil --forget com.zabbix.pkg.ZabbixAgent
```

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。

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。

组件

所有 Zabbix 组件都可在以下 Docker 镜像仓库中使用：

- 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 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 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 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。

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；
- `ubuntu-latest` - 基于 Ubuntu 镜像的最新稳定版的 Zabbix 组件；
- `alpine-5.0-latest` - 基于 Alpine Linux 镜像的最新次要版本的 Zabbix 5.0 组件；
- `ubuntu-5.0-latest` - 基于 Unbuntu 镜像的最新次要版本的 Zabbix 5.0 组件；
- `alpine-5.0.*` - 基于 Alpine Linux 镜像的不同次要版本的 Zabbix 5.0 组件，其中 * 代表 Zabbix 组件的次要版本；
- `ubuntu-5.0.*` - 基于 Ubuntu 镜像的不同次要版本的 Zabbix 5.0 组件，其中 * 代表 Zabbix 组件的次要版本

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。

Attention:
一些配置选项是不允许更改的。例如，PIDFile 和 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。

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：

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。

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。1. 创建专用于 Zabbix 组件容器的网络：

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

2. 启动空的 MySQL 服务器实例

```
# docker run --name mysql-server -t \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      --network=zabbix-net \
      -d mysql:8.0 \
      --restart unless-stopped \
      --character-set-server=utf8 --collation-server=utf8_bin \
      --default-authentication-plugin=mysql_native_password
```

3. 启动 Zabbix Java gateway 实例

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

4. 启动 Zabbix server 实例并将该实例与创建的 MySQL 服务器实例链接

```
# docker run --name zabbix-server-mysql -t \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
      --network=zabbix-net \
```

```
      -p 10051:10051 \
      --restart unless-stopped \
      -d zabbix/zabbix-server-mysql:alpine-5.0-latest
```

> **Note:**
> Zabbix 服务器实例向主机公开 10051 / TCP 端口（Zabbix trapper）

5. 启动 Zabbix Web 界面，并将实例与创建的 MySQL 服务器和 Zabbix server 实例链接

```
# docker run --name zabbix-web-nginx-mysql -t \
      -e ZBX_SERVER_HOST="zabbix-server-mysql" \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      --network=zabbix-net \
      -p 80:8080 \
      --restart unless-stopped \
      -d zabbix/zabbix-web-nginx-mysql:alpine-5.0-latest
```

> **Note:**
> Zabbix Web 界面实例向主机公开 80 / TCP 端口（HTTP）

** 示例 2 **

该示例演示了如何在具有 PostgreSQL 数据库支持，基于 Nginx Web 服务器的 Zabbix Web 界面和 SNMP 陷阱功能的情况下运行 Zabbix 服务器。

1. 创建专用于 Zabbix 组件容器的网络：

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

2. 启动空的 PostgreSQL 服务器实例

```
# docker run --name postgres-server -t \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix_pwd" \
      -e POSTGRES_DB="zabbix" \
      --network=zabbix-net \
      --restart unless-stopped \
      -d postgres:latest
```

3. 启动 Zabbix snmptraps 实例

```
# docker run --name zabbix-snmptraps -t \
      -v /zbx_instance/snmptraps:/var/lib/zabbix/snmptraps:rw \
      -v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
      --network=zabbix-net \
      -p 162:1162/udp \
      --restart unless-stopped \
      -d zabbix/zabbix-snmptraps:alpine-5.0-latest
```

> **Note:**
> Zabbix snmptrap 实例向主机公开 162 / UDP 端口（SNMP traps）

4. 启动 Zabbix 服务器实例并将该实例与创建的 PostgreSQL 服务器实例链接

```
# docker run --name zabbix-server-pgsql -t \
      -e DB_SERVER_HOST="postgres-server" \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix_pwd" \
      -e POSTGRES_DB="zabbix" \
      -e ZBX_ENABLE_SNMP_TRAPS="true" \
      --network=zabbix-net \
      -p 10051:10051 \
```

```
      --volumes-from zabbix-snmptraps \
      --restart unless-stopped \
      -d zabbix/zabbix-server-pgsql:alpine-5.0-latest
```

Zabbix 服务器实例将 10051 / TCP 端口（Zabbix trapper）公开给主机。::: 5. 启动 Zabbix Web 界面，并将实例与创建的 PostgreSQL 服务器和 Zabbix 服务器实例链接

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

Zabbix Web 界面实例向主机公开 443 / TCP 端口（HTTPS）。目录/ etc / ssl / nginx 必须包含具有所需名称的证书。::: ** 示例 3 **

该示例演示了如何在 Red Hat 8 上使用 podman 运行具有 MySQL 数据库支持的 Zabbix 服务器，基于 Nginx Web 服务器的 Zabbix Web 界面以及 Zabbix Java gateway。

1. 使用名称 zabbix 和公开的端口（Web 界面，Zabbix server trapper）创建新的 Pod：

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

2.（可选）在 zabbix pod 位置启动 Zabbix agent 容器：

```
podman run --name zabbix-agent \
    -eZBX_SERVER_HOST="127.0.0.1,localhost" \
    --restart=always \
    --pod=zabbix \
    -d registry.connect.redhat.com/zabbix/zabbix-agent-50:latest
```

3. 在主机上创建./mysql/目录，然后启动 Oracle MySQL server 8.0：

```
podman run --name mysql-server -t \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -v ./mysql/:/var/lib/mysql/:Z \
      --restart=always \
      --pod=zabbix \
      -d mysql:8.0 \
      --character-set-server=utf8 --collation-server=utf8_bin \
      --default-authentication-plugin=mysql_native_password
```

4. 启动 Zabbix server 容器：

```
podman run --name zabbix-server-mysql -t \
                  -e DB_SERVER_HOST="127.0.0.1" \
                  -e MYSQL_DATABASE="zabbix" \
                  -e MYSQL_USER="zabbix" \
                  -e MYSQL_PASSWORD="zabbix_pwd" \
                  -e MYSQL_ROOT_PASSWORD="root_pwd" \
                  -e ZBX_JAVAGATEWAY="127.0.0.1" \
                  --restart=always \
                  --pod=zabbix \
                  -d registry.connect.redhat.com/zabbix/zabbix-server-mysql-50
```

5. 启动 Zabbix Java Gateway 容器：

```
podman run --name zabbix-java-gateway -t \
      --restart=always \
      --pod=zabbix \
      -d registry.connect.redhat.com/zabbix/zabbix-java-gateway-50
```

Note:
Pod zabbix 从 zabbix-web-mysql 容器的 8080 / TCP 向主机公开 80 / TCP 端口（HTTP）。

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。

以下为几个不同版本的 compose 文件：

文件名 **	述 **
`docker-compose_v3_alpine_mysql_latest.yaml`	该 compose 文件运行基于 Alpine Linux 的 Zabbix 5.0 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 5.0 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 5.0 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 5.0 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 CentOS8 的 Zabbix 5.0 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 CentOS8 的 Zabbix 5.0 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 CentOS8 的 Zabbix 5.0 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 CentOS8 的 Zabbix 5.0 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 20.04 的 Zabbix 5.0 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 20.04 的 Zabbix 5.0 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 20.04 的 Zabbix 5.0 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 20.04 的 Zabbix 5.0 最新版本的组件，支持 PosegreSQL 数据库。

Attention:
可用的 Docker compose 文件支持 Docker Compose 的版本 3。

存储

撰写文件被配置为支持主机上的本地存储。当您使用 compose 文件运行 Zabbix 组件时，Docker Compose 将在文件夹中使用 compose 文件创建一个 zbx_env 目录。该目录将包含与上述 “卷” 部分中描述的结构相同的目录以及用于数据库存储的目录。

此外，还有卷 `/etc/localtime` 和 `/etc/timezone` 下的文件为只读模式。

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。

示例

** 示例 1 **

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

该命令将为每个 Zabbix 组件下载最新的 Zabbix 5.0 映像，并在分离模式下运行它们。

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。

** 示例 2 **

```
# git checkout 5.0
# docker-compose -f ./docker-compose_v3_ubuntu_mysql_local.yaml up -d
```

该命令将下载基本映像 Ubuntu 20.04（本地），然后在本地构建 Zabbix 5.0 组件并以分离模式运行它们。

Installation with OpenShift

Overview

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

Supported features

Zabbix Operator comes with a few possible installation options:

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

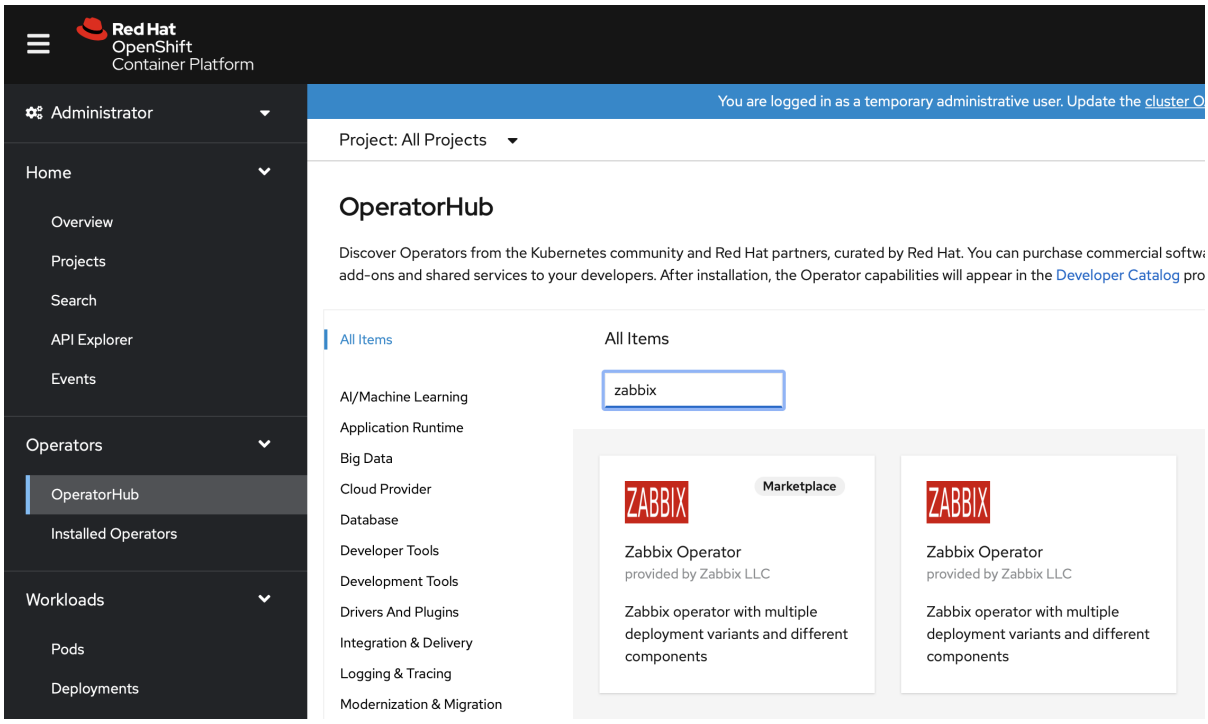
Currently Zabbix Operator is based on the Zabbix 5.0 LTS version and supports OpenShift 4.0, 4.1, 4.2, 4.3, 4.4 and 4.5.

Installing Zabbix Operator

Using RedHat Marketplace

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

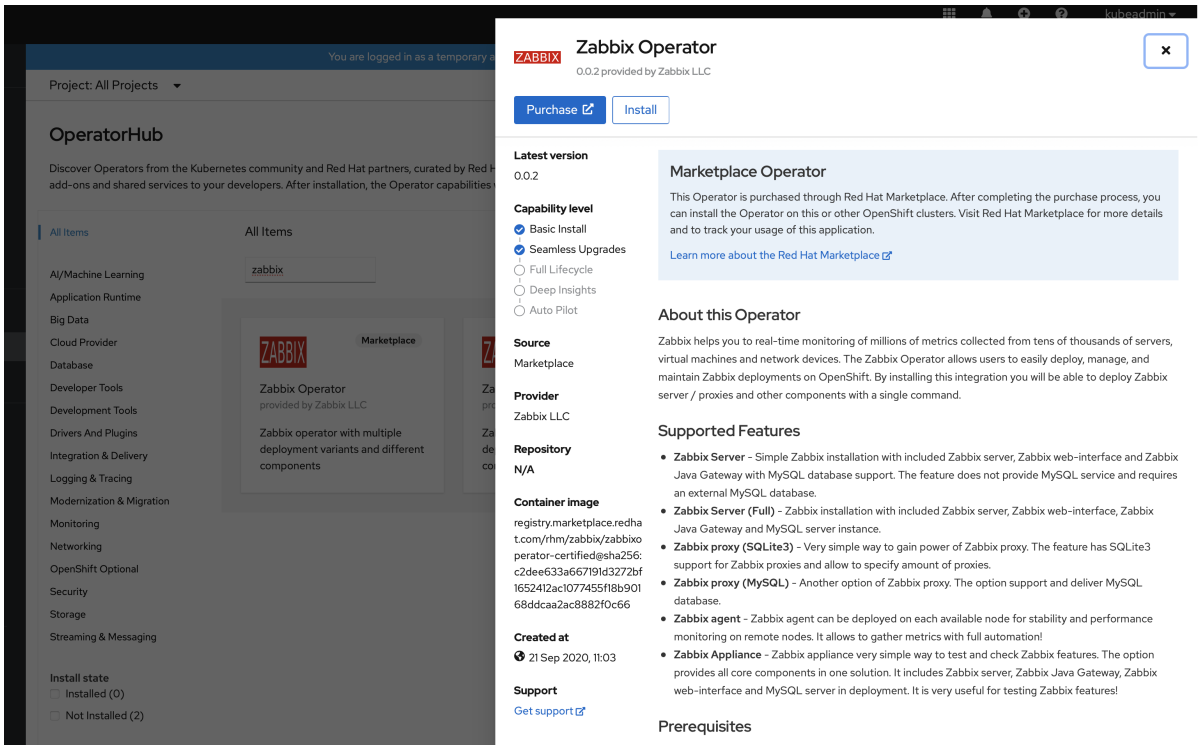
1. Select the OperatorHub from the Operators submenu and search for Zabbix.



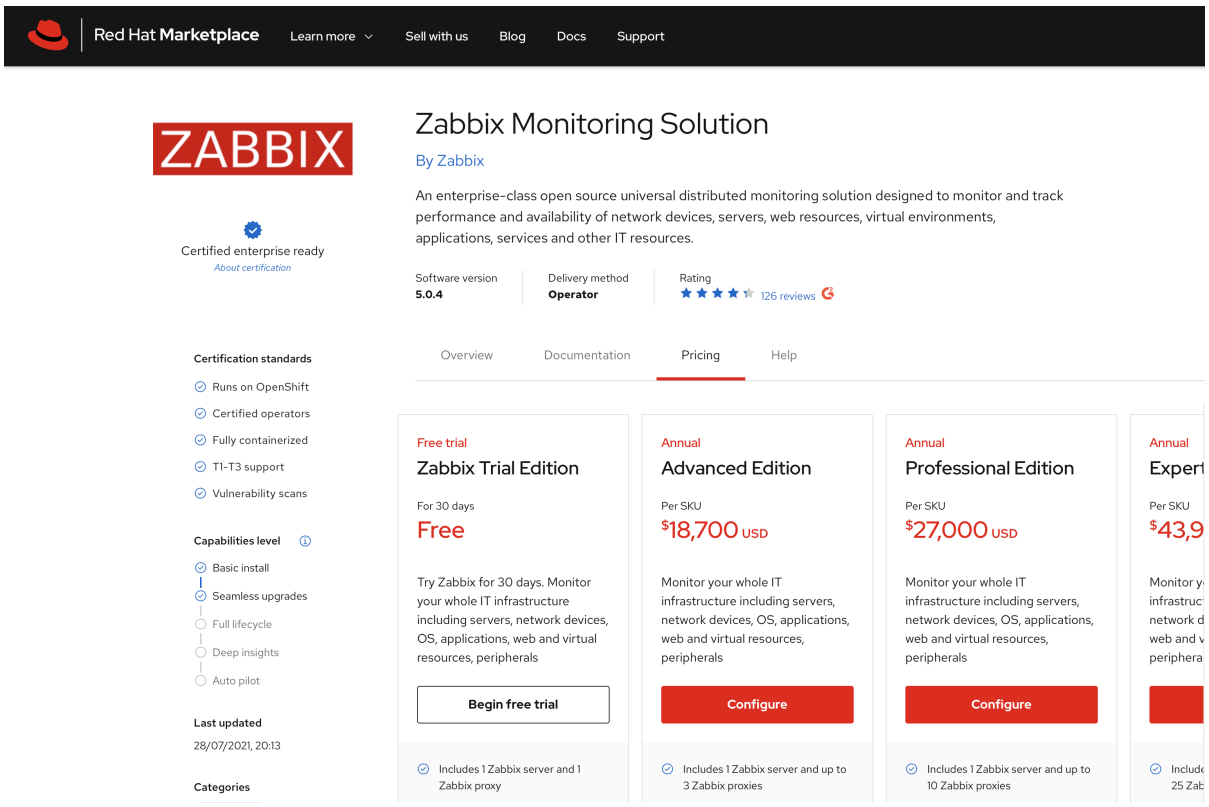
Choose the RedHat Marketplace option.

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

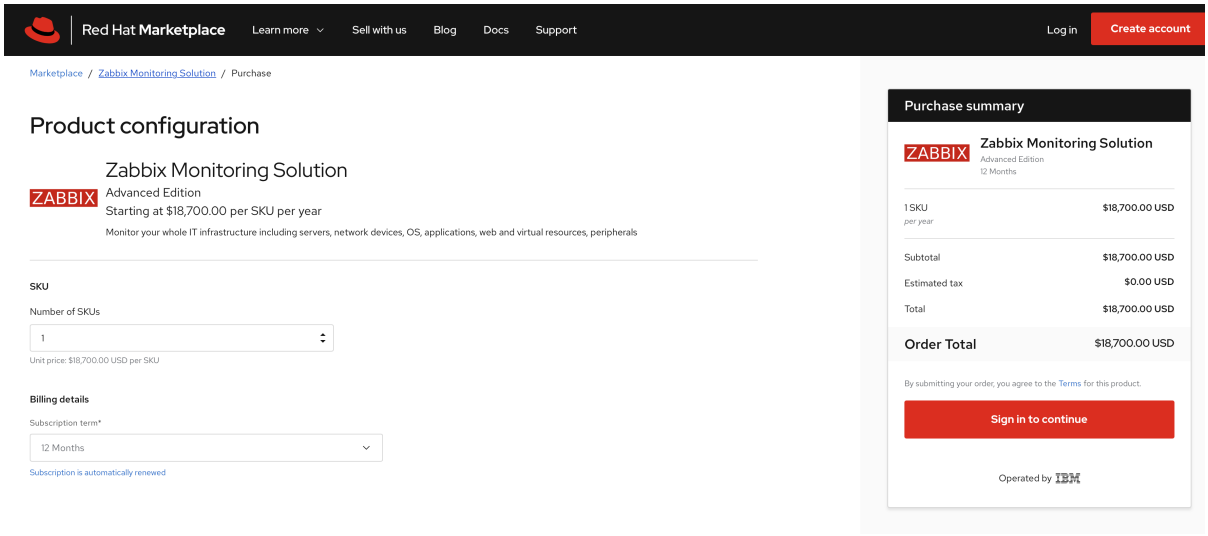
> **Attention:**
> Openshift needs to be registered with the Red Hat Marketplace portal.



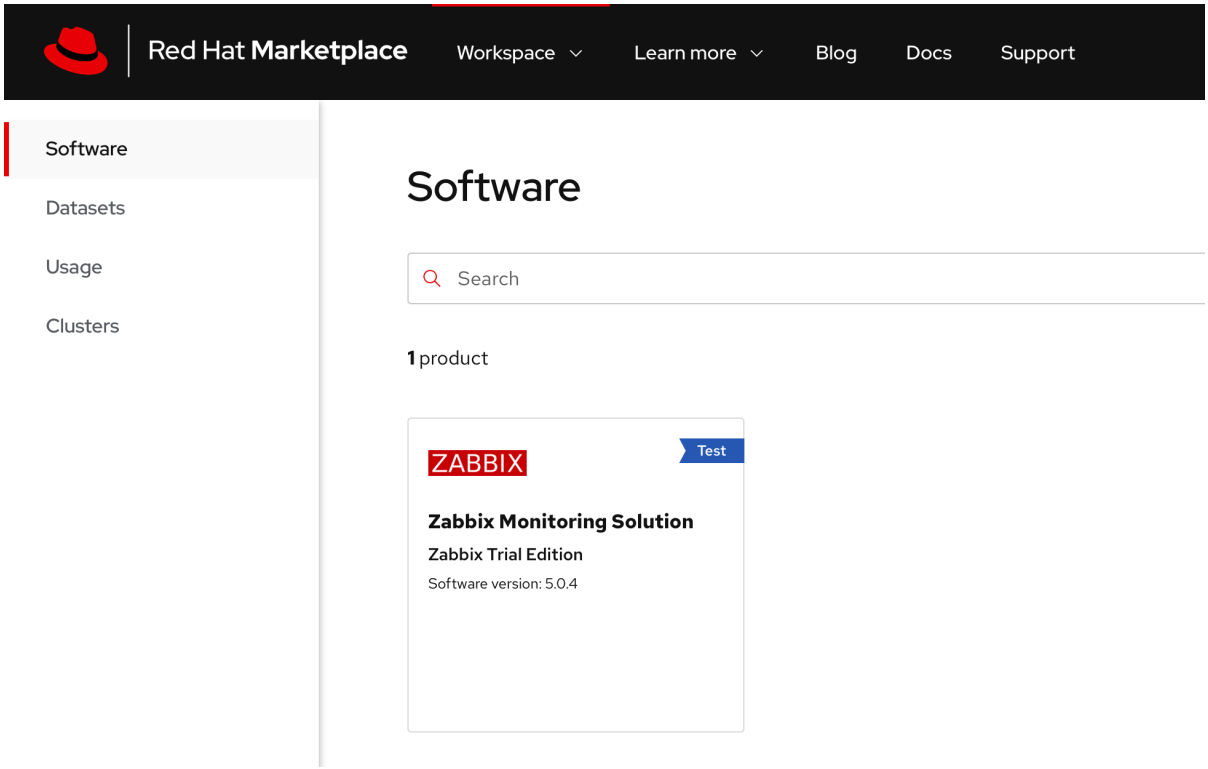
3. Select the most suitable install option.



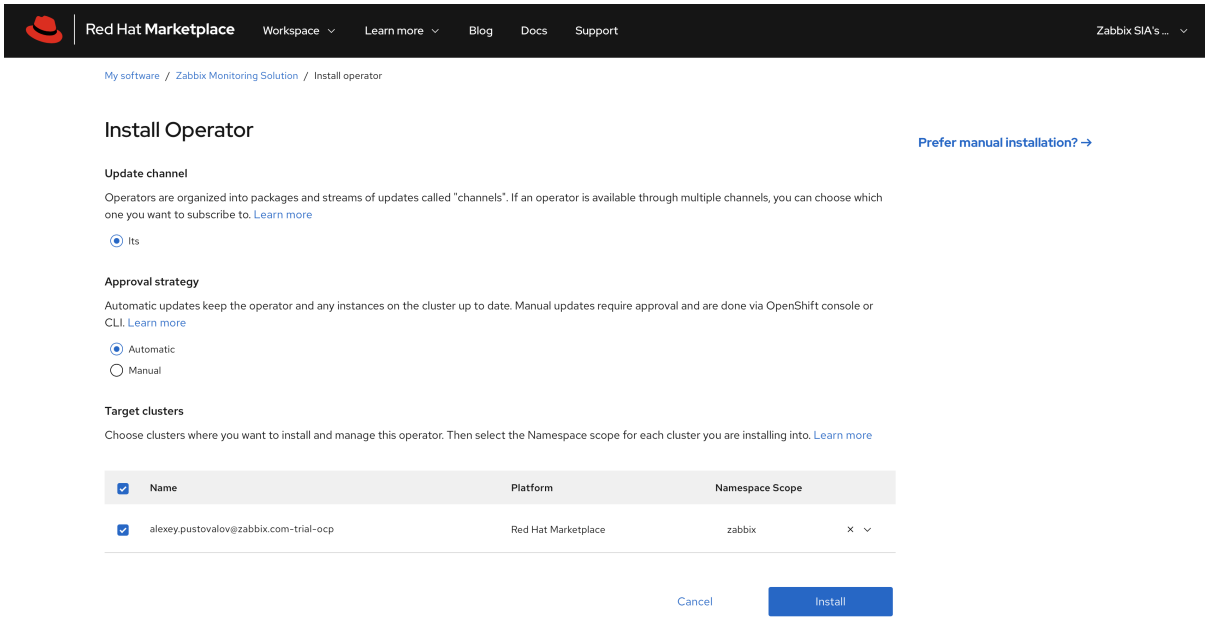
4. Specify the product configuration to fit your needs.



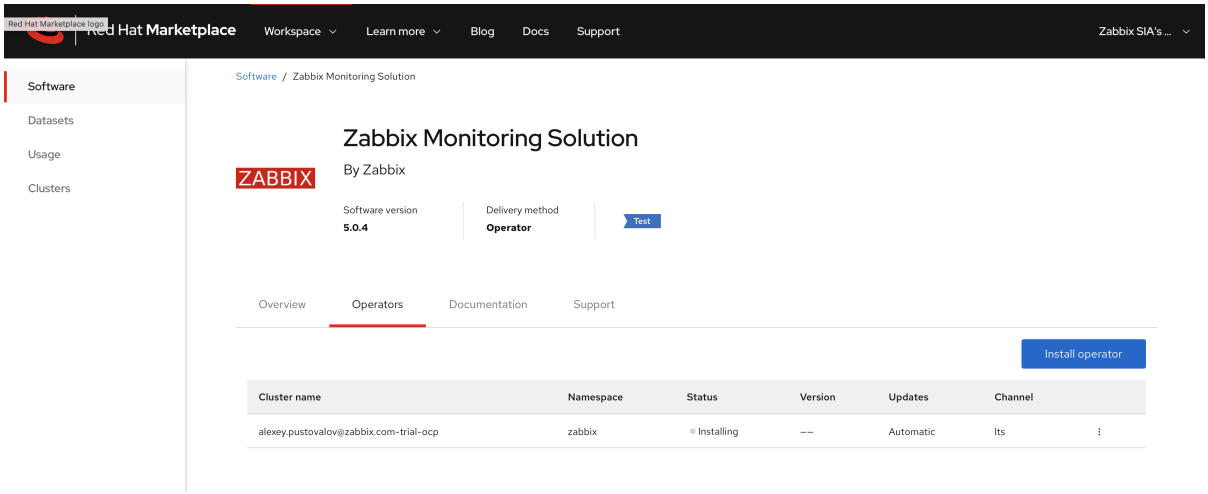
5. Navigate to your software within Red Hat Marketplace and install the Zabbix Operator software as specified in the image.



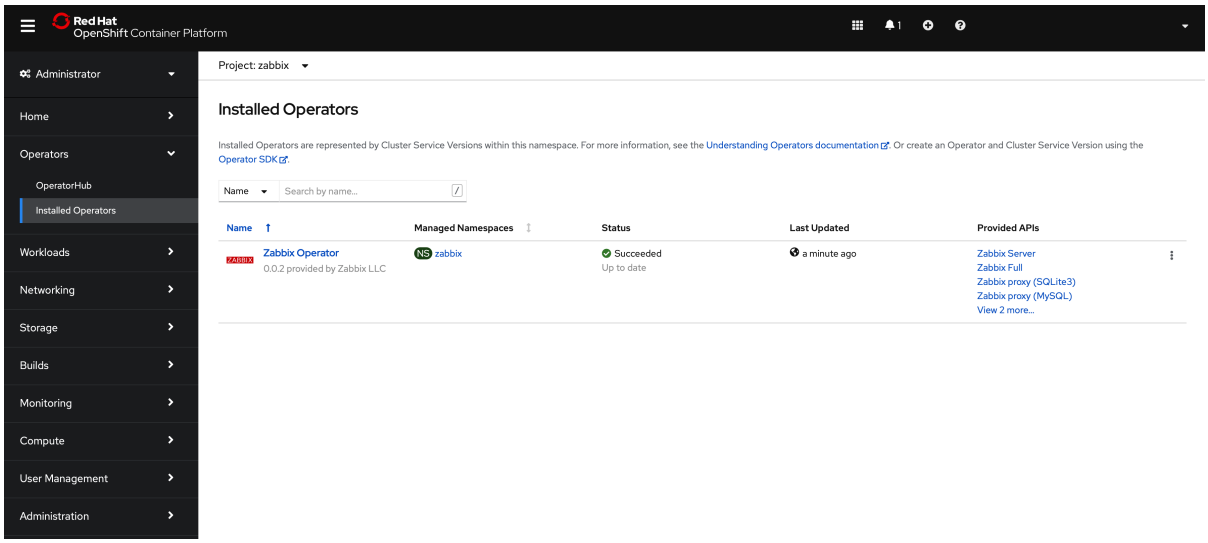
6. Install the Operator. Set the update approval strategy to *Automatic* to ensure that you always have the latest version of Zabbix components installed.



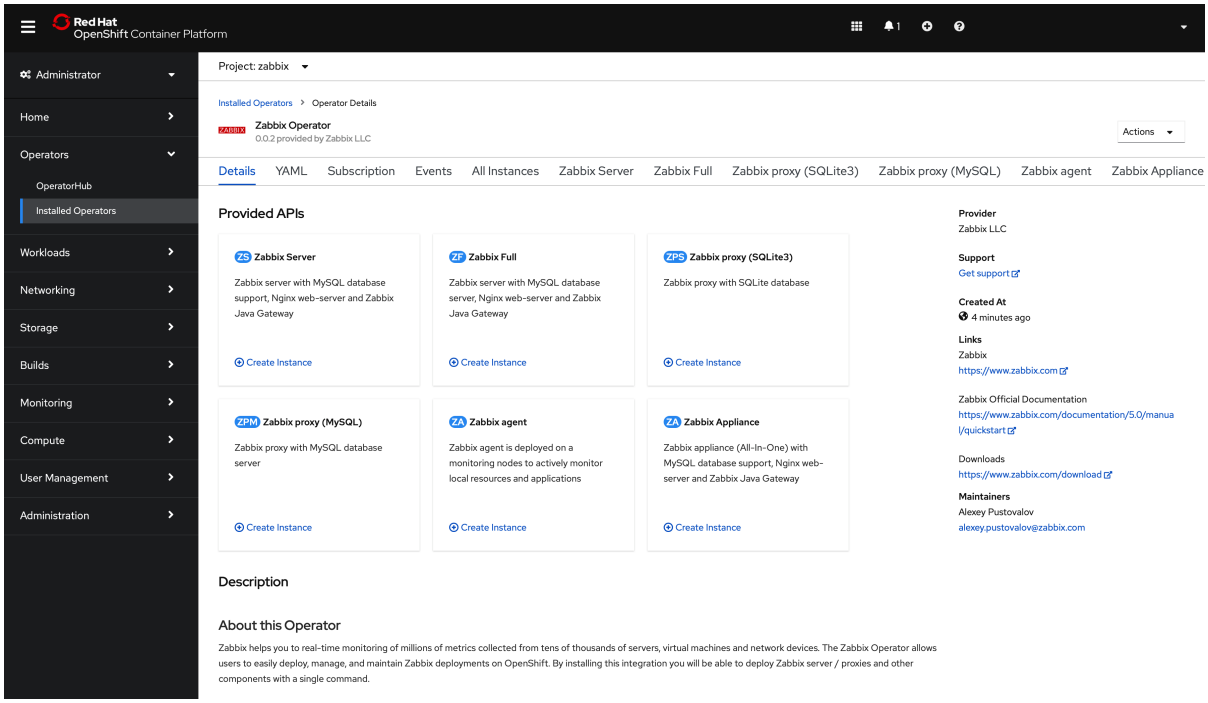
7. The Zabbix Operator is now installed into your specified cluster.



8. Go to Operators → Installed Operators.



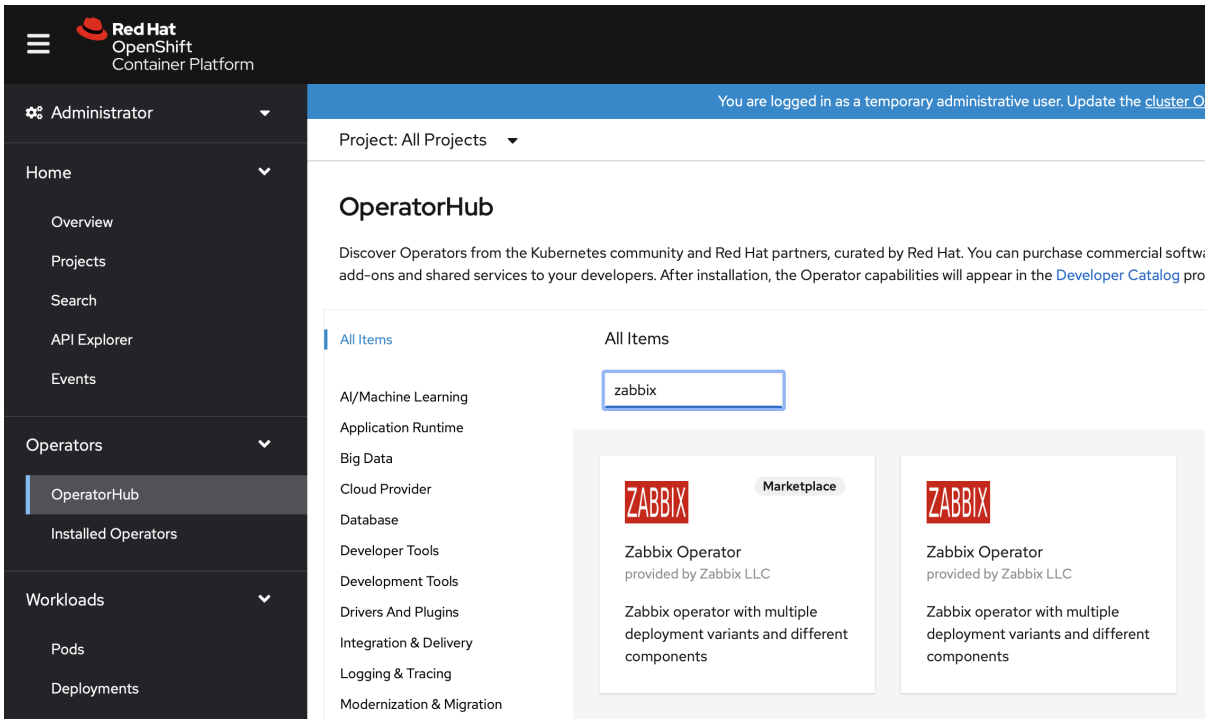
9. Open the ”Zabbix Operator” configuration page.



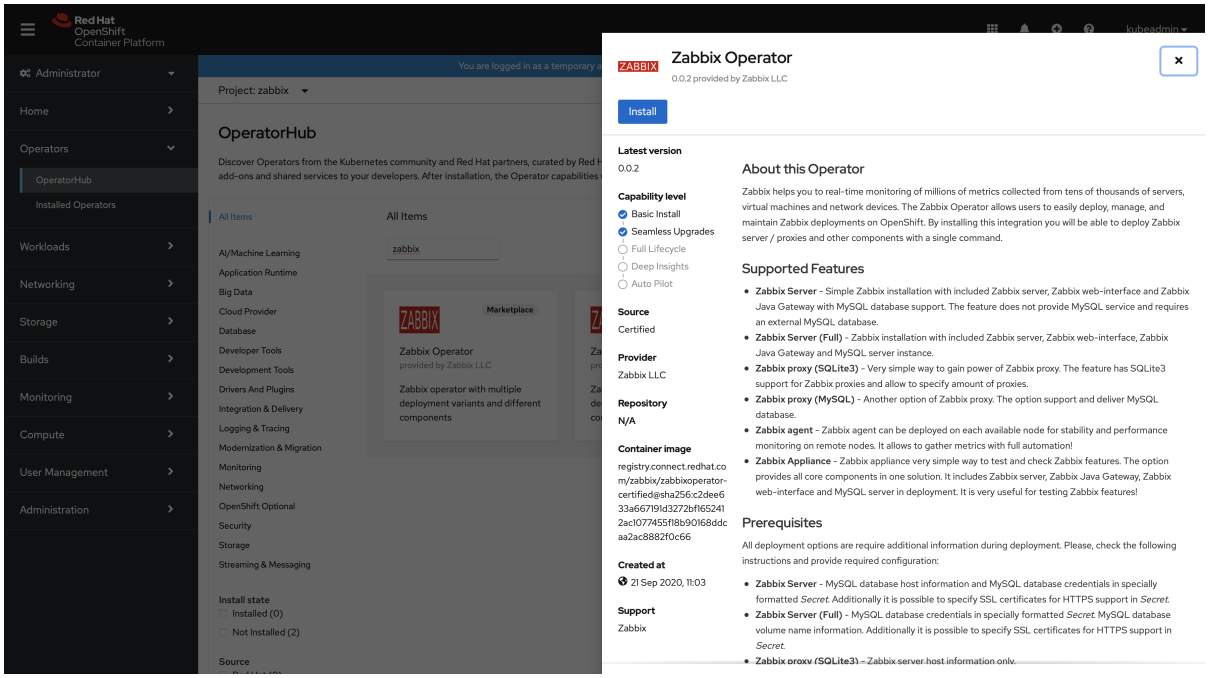
Using OperatorHub

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

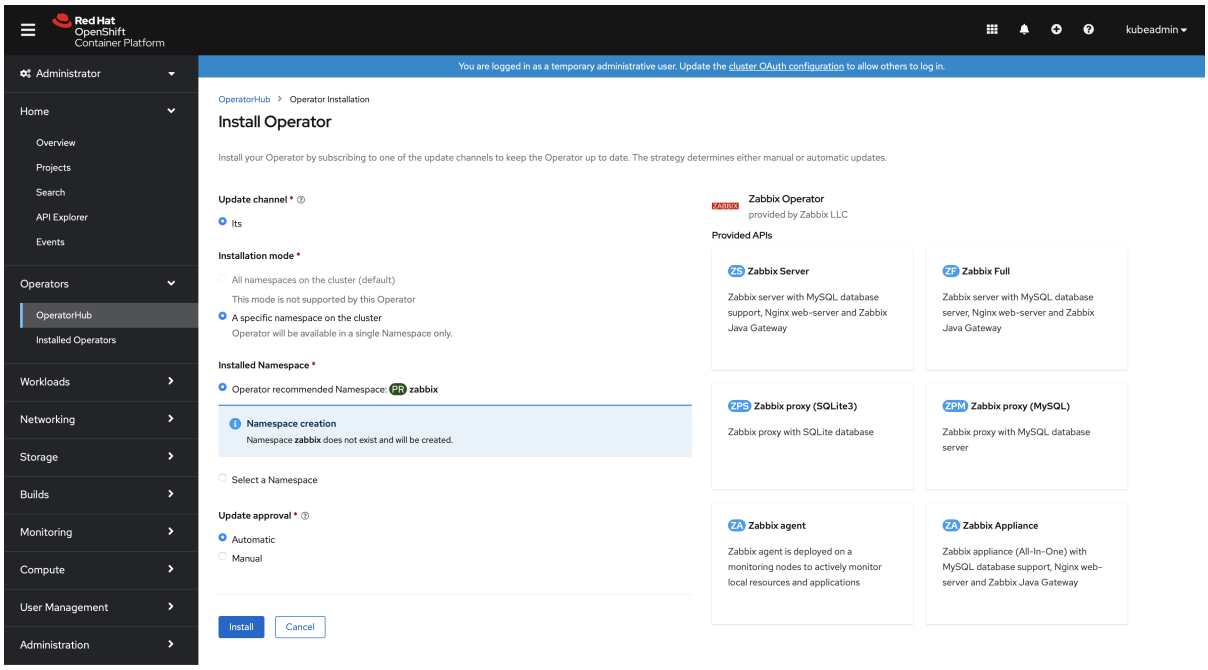
1. Select OperatorHub from the Operators submenu and search for Zabbix.



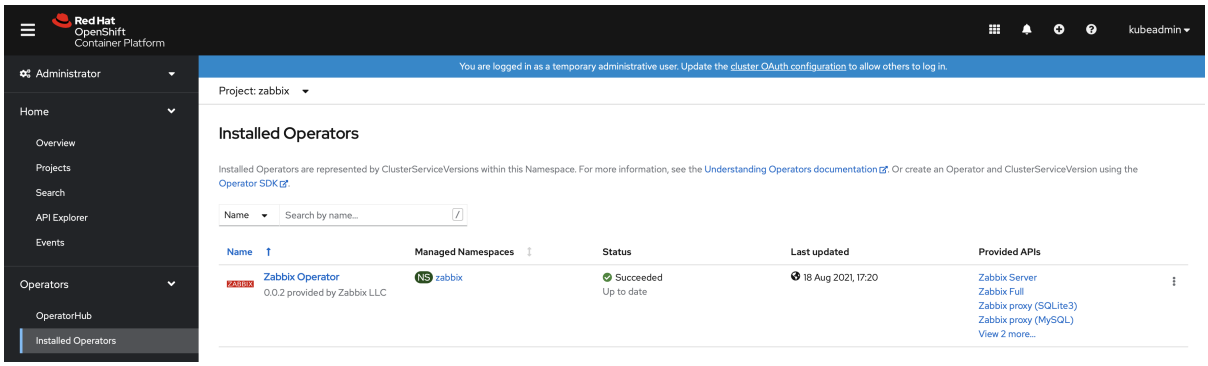
2. Select Zabbix Operator and click on Install.



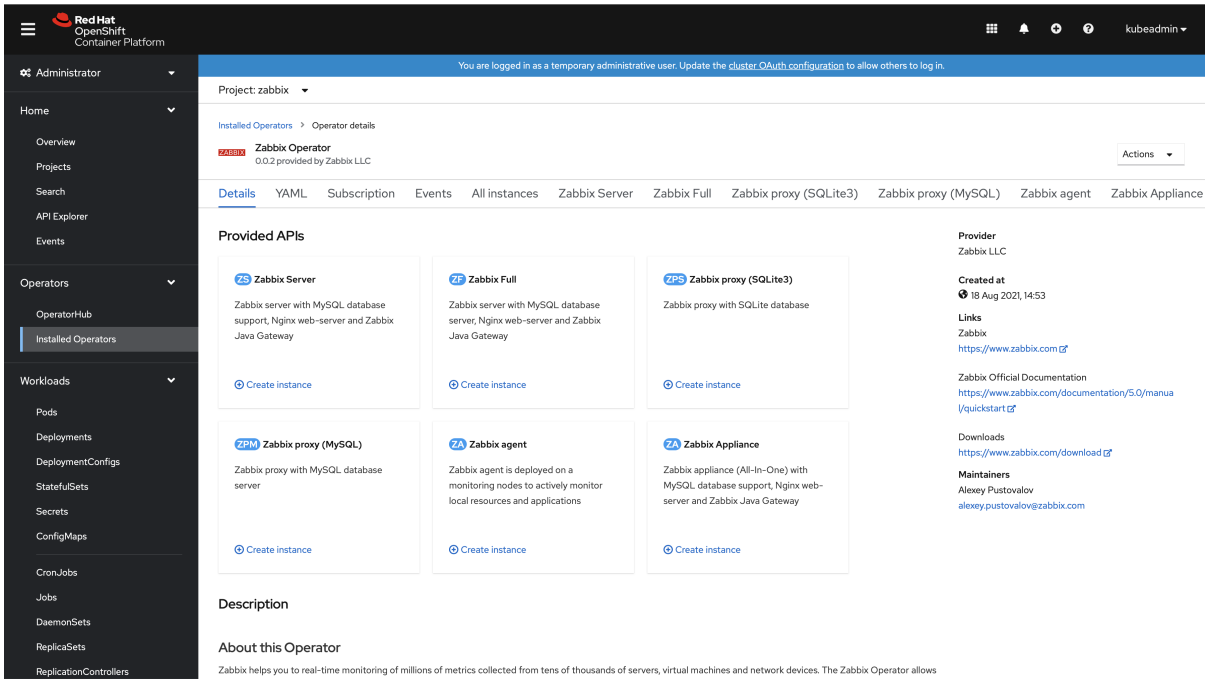
3. Select the installation options.



4. Go to Operators → Installed Operators.

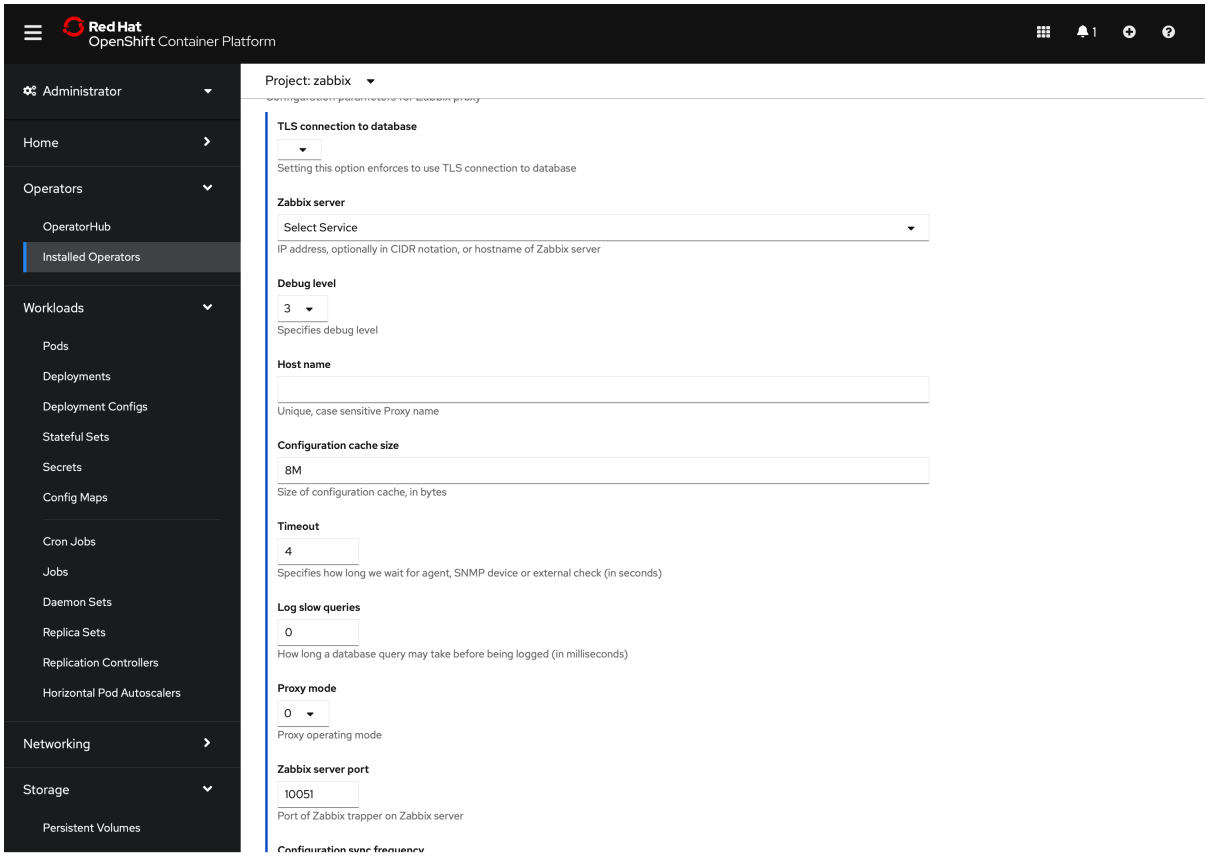


5. Open the ”Zabbix Operator” configuration page.

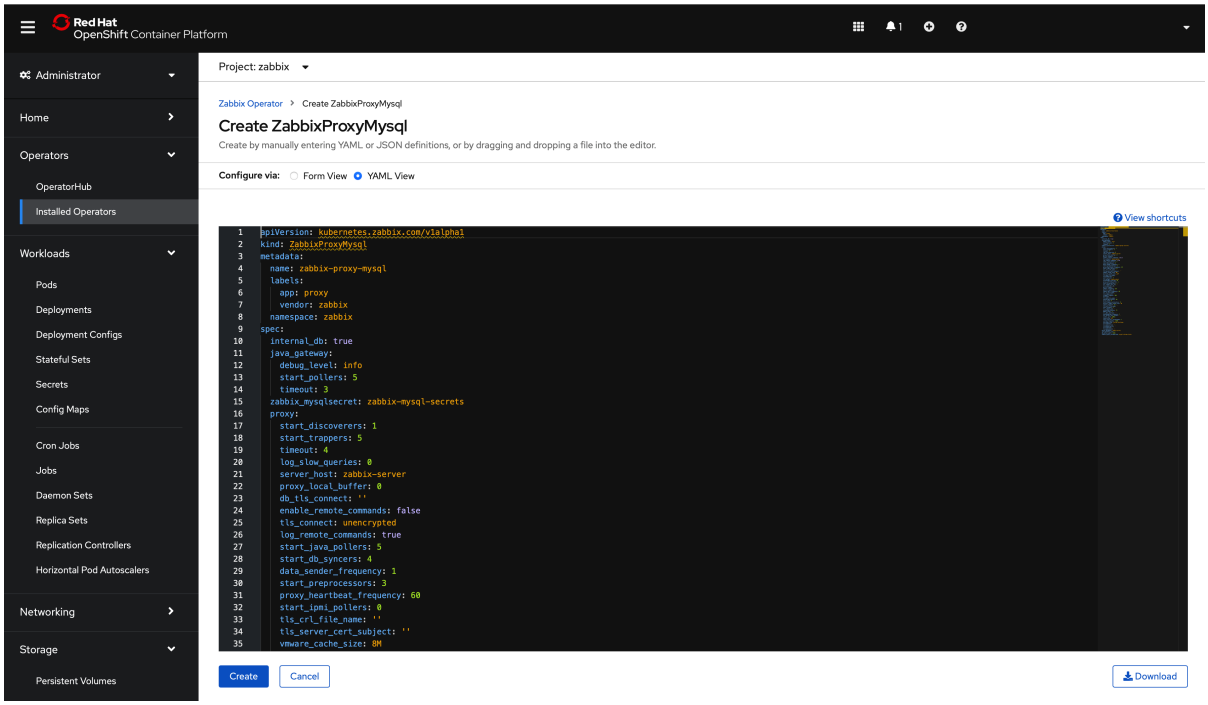


Configuration

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



The YAML section provides all available options with default values:



Zabbix server

This operand has a few prerequisites:

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:8.0
        name: mysql
        args:
          - mysqld
          - '--character-set-server=utf8'
          - '--collation-server=utf8_bin'
          - '--default-authentication-plugin=mysql_native_password'
        env:
```

```
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: Welcome1!
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
---
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
  - port: 3306
  selector:
    app: mysql
  clusterIP: None
```

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

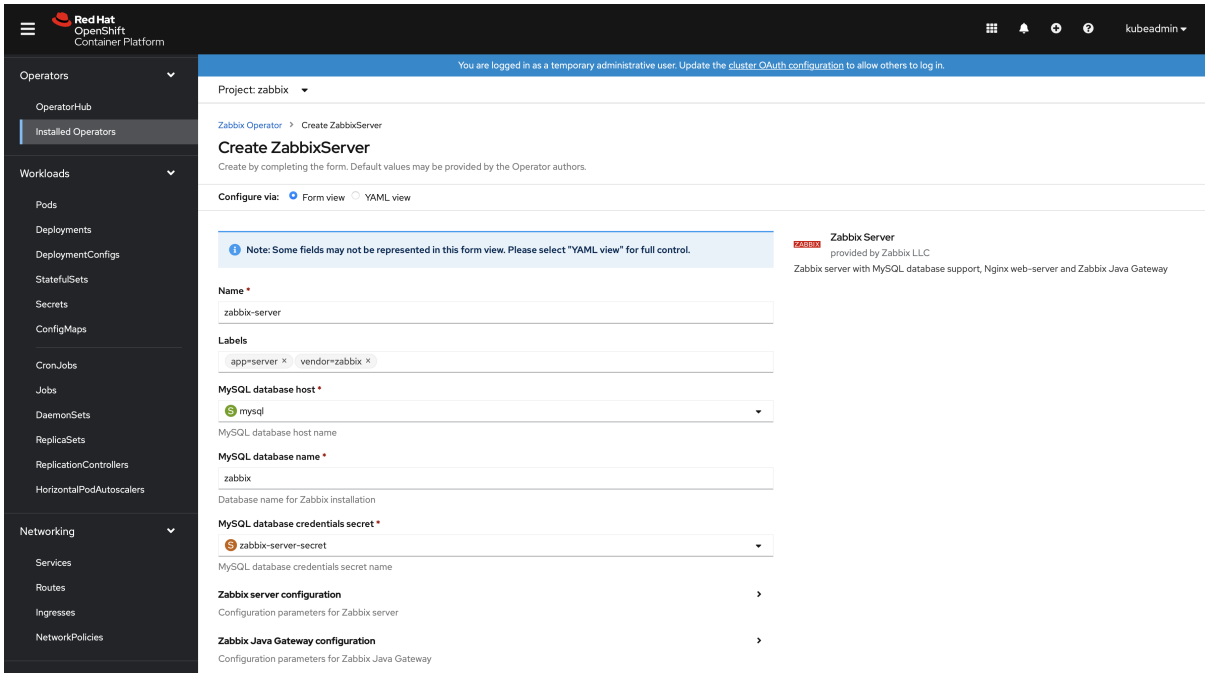
2. MySQL credentials using secret - must be secret with mysql_root_password, mysql_zabbix_username and mysql_zabbix_password data. For example:

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

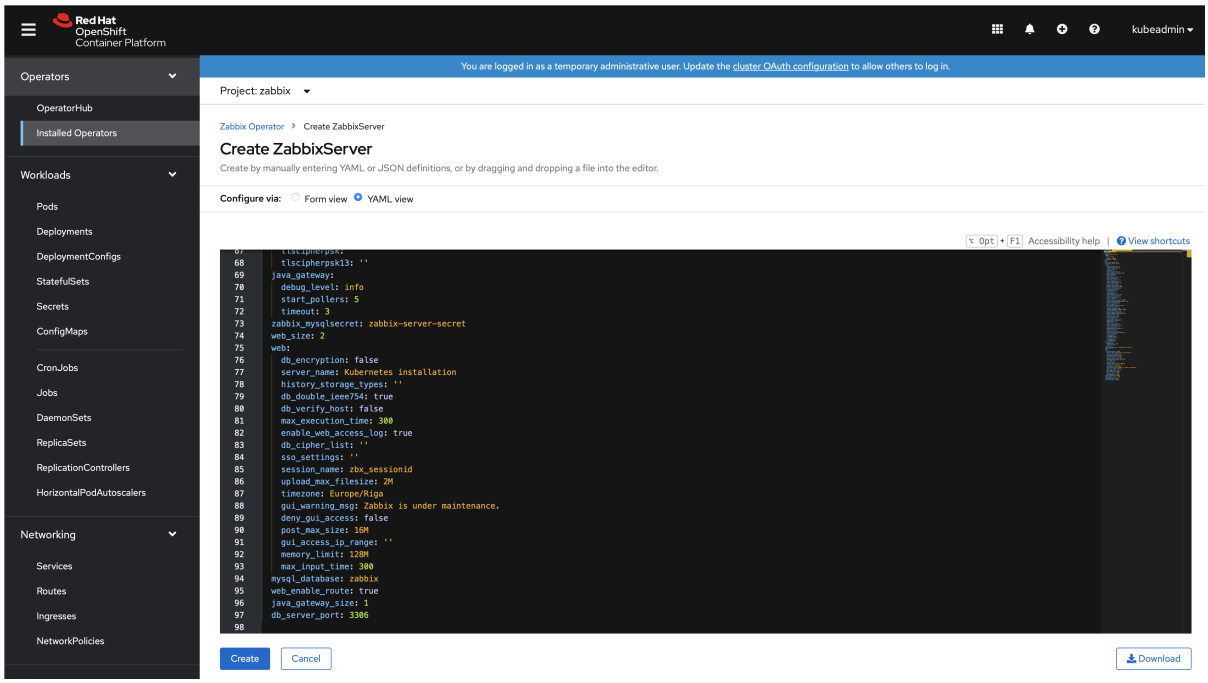
where all fields are encoded using base64. For example:

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

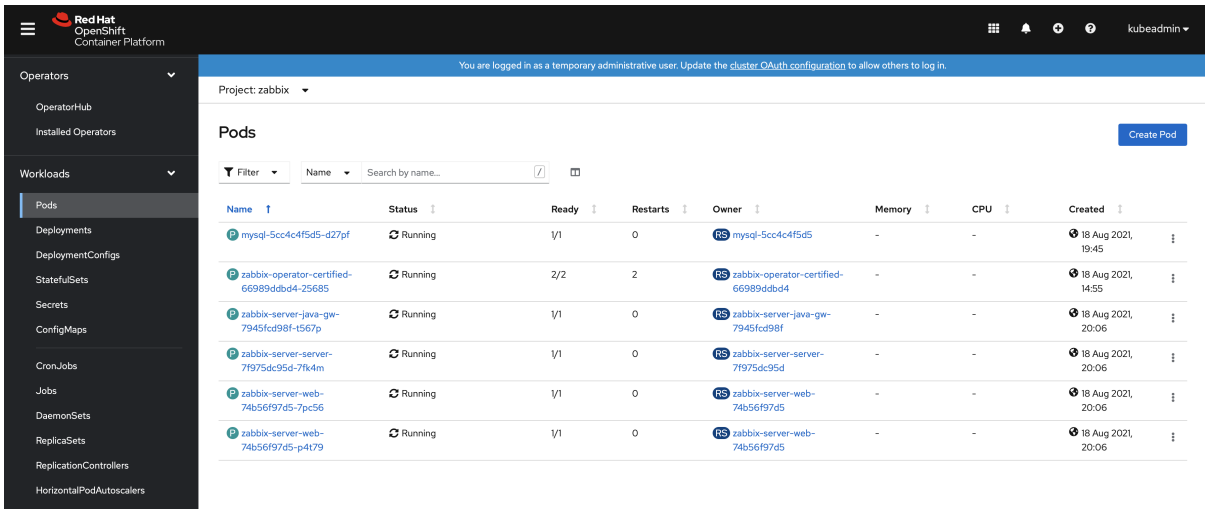
An example of ”Zabbix Server” operand configuration:



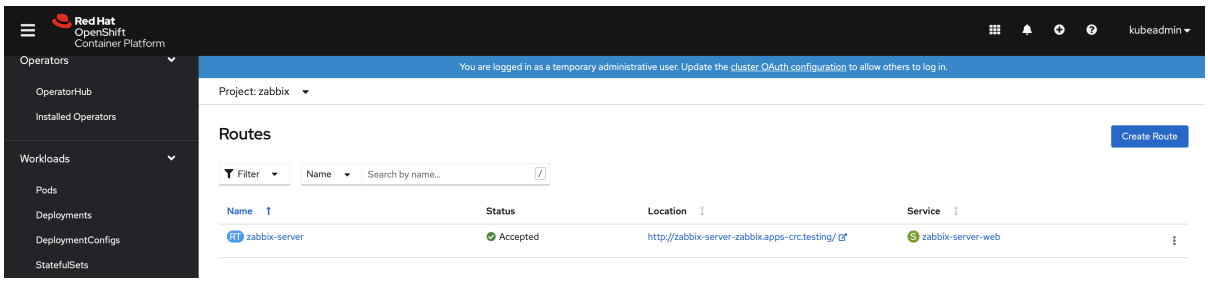
All configuration options are available using the form view, but it is possible to use the YAML view as well. For example:



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



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



Zabbix full

This operand has a few prerequisites:

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

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

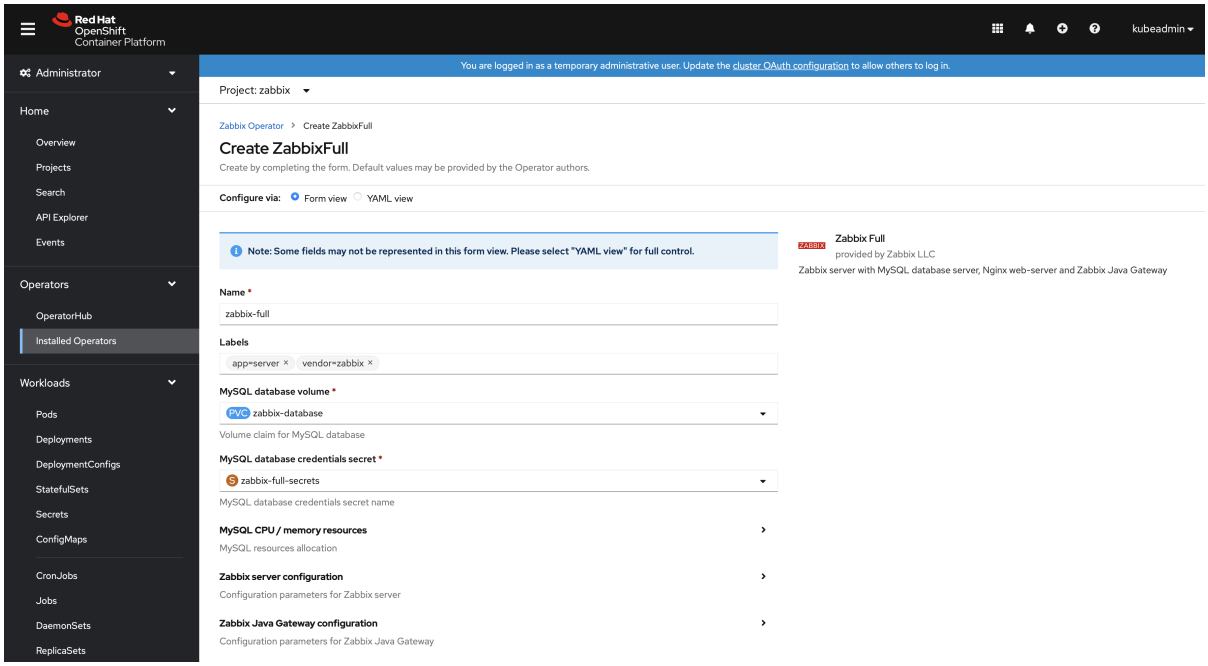
2. MySQL credentials using secret - must be secret with mysql_root_password, mysql_zabbix_username and mysql_zabbix_password data. For example:

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

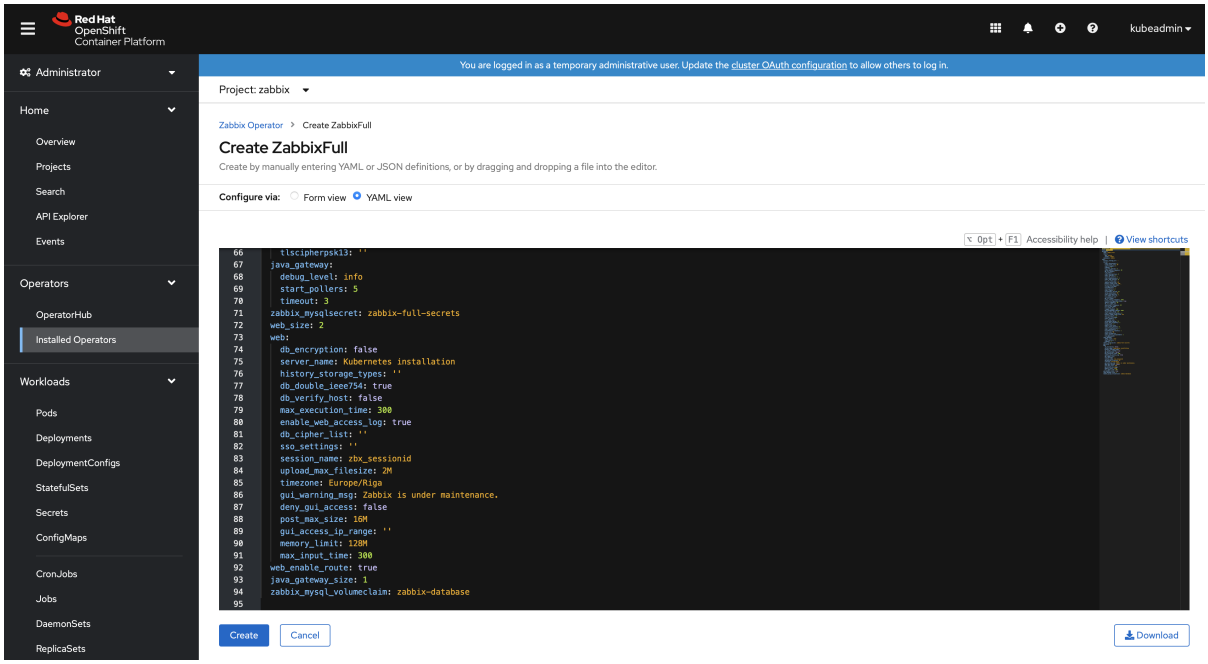
where all fields are encoded using base64. For example:

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

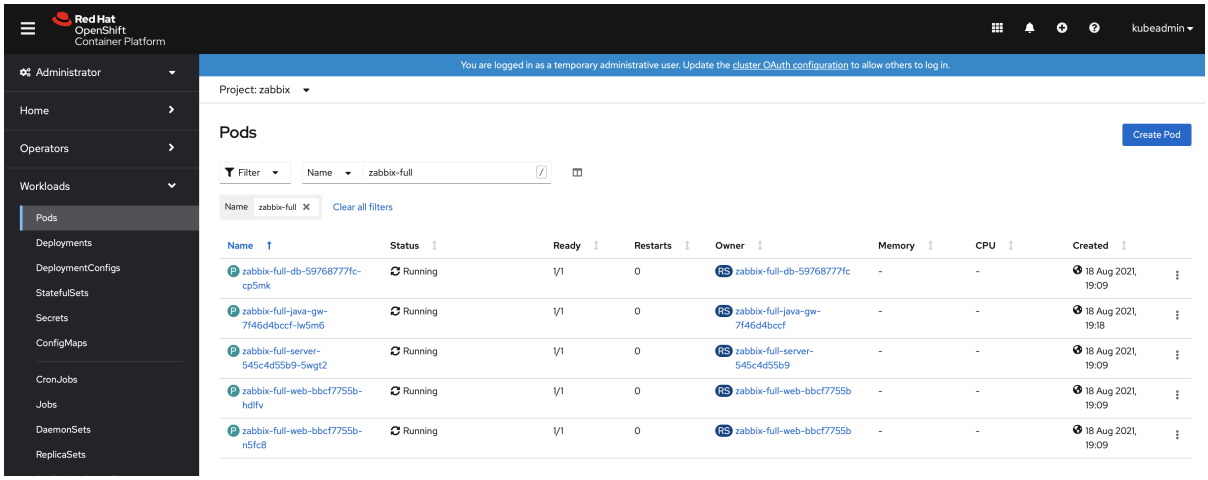
An example of ”Zabbix Full” operand configuration:



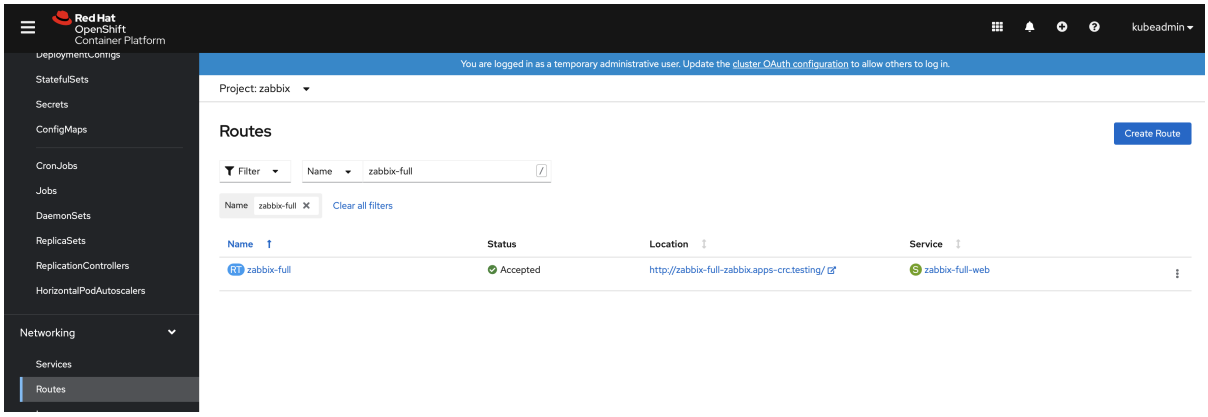
All configuration options are available using the form view, but it is possible to use the YAML view as well. For example:



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

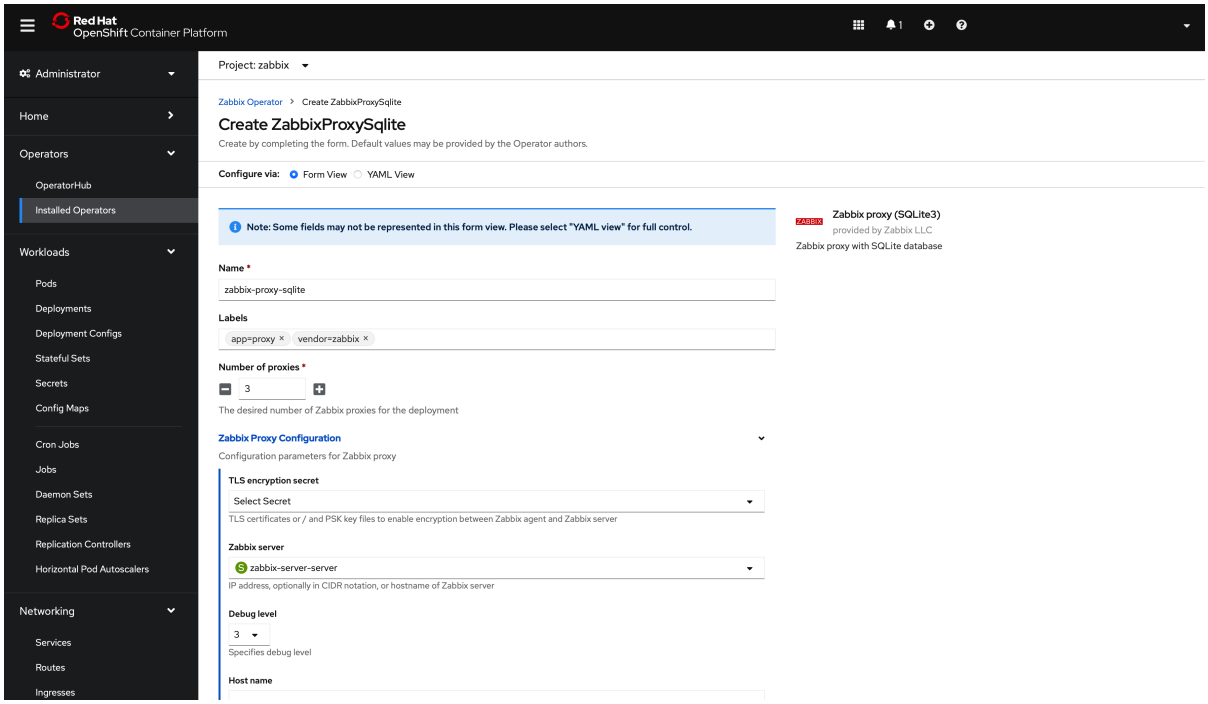


The route for Zabbix web interface is located under Networking → Routes. The URL provides access to the Zabbix web interface. In the following example it is `http://zabbix-full-zabbix.apps-crc.testing/`:

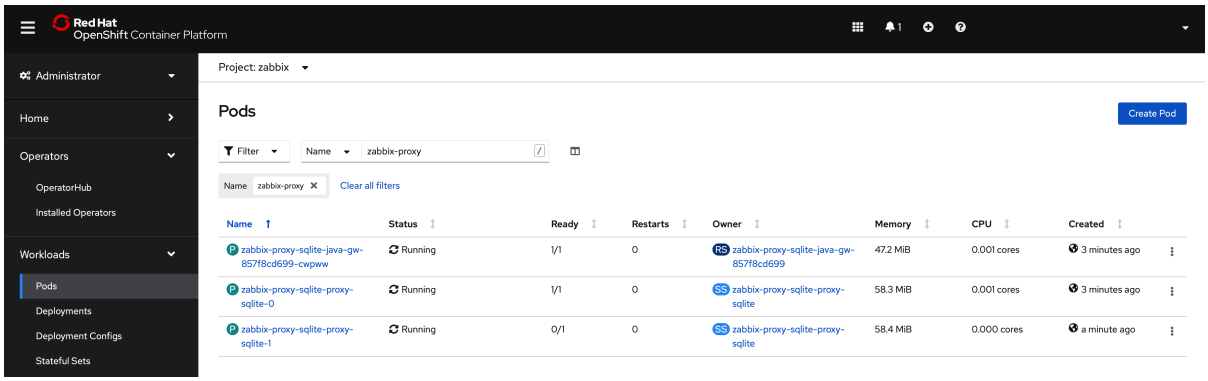


Zabbix proxy (SQLite3)

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



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

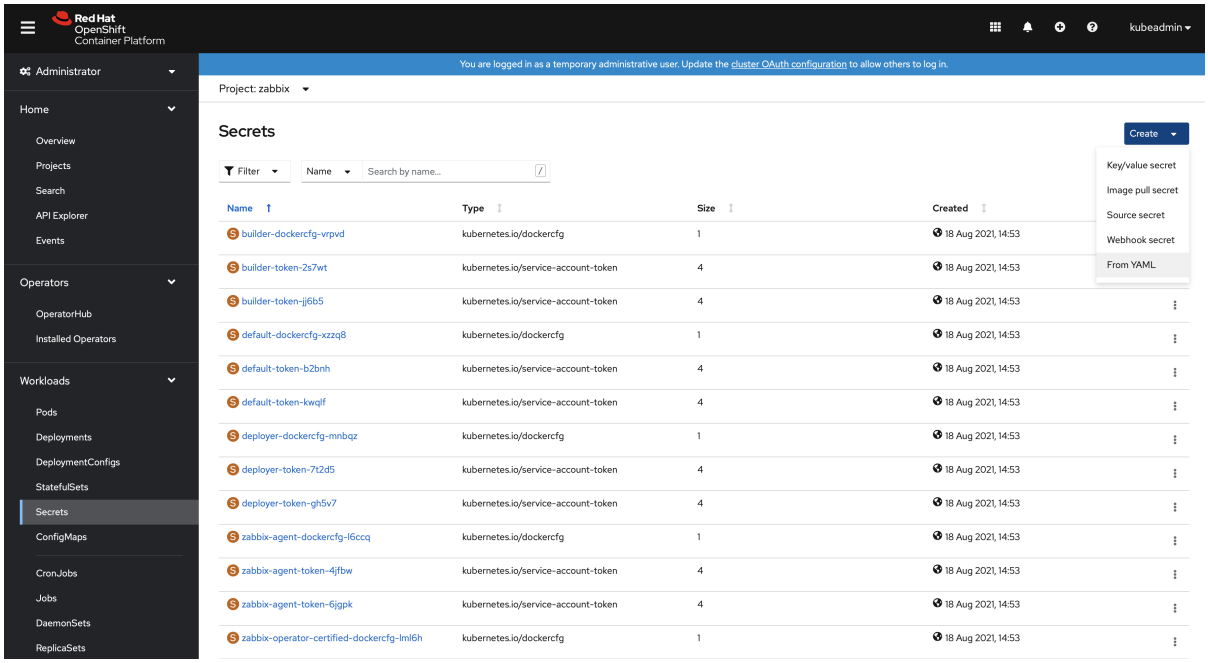


Additional information

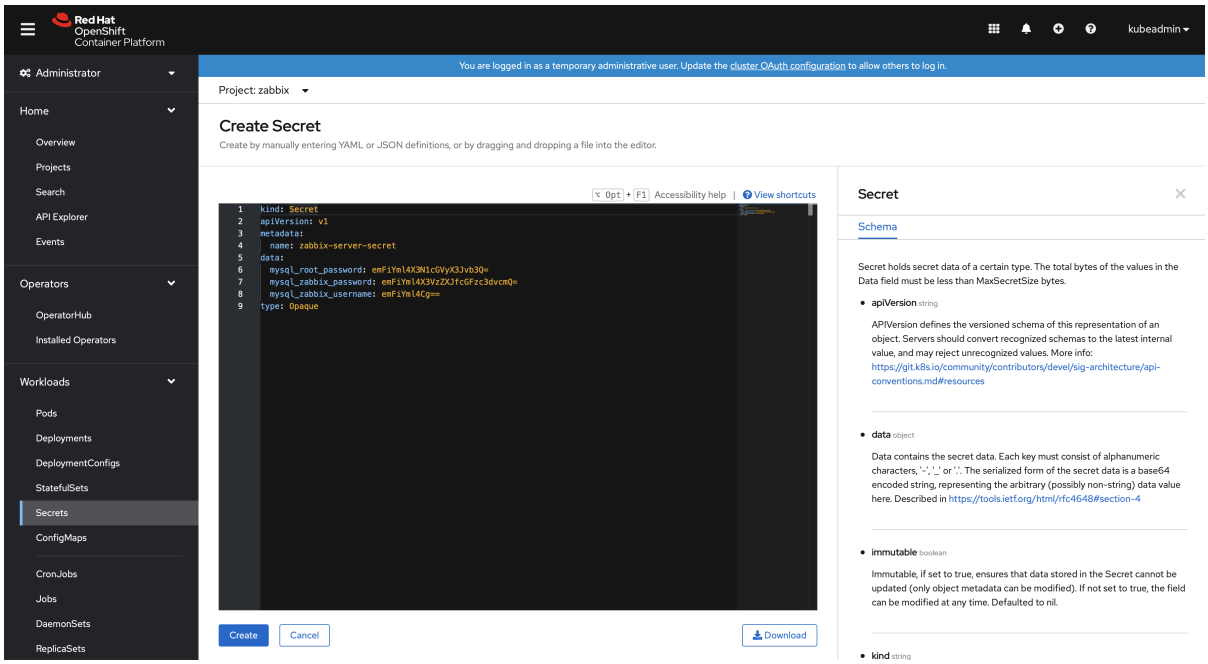
Creating new secret

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

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



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



SSL certificates for HTTPS

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

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

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

MySQL database certificate base encryption

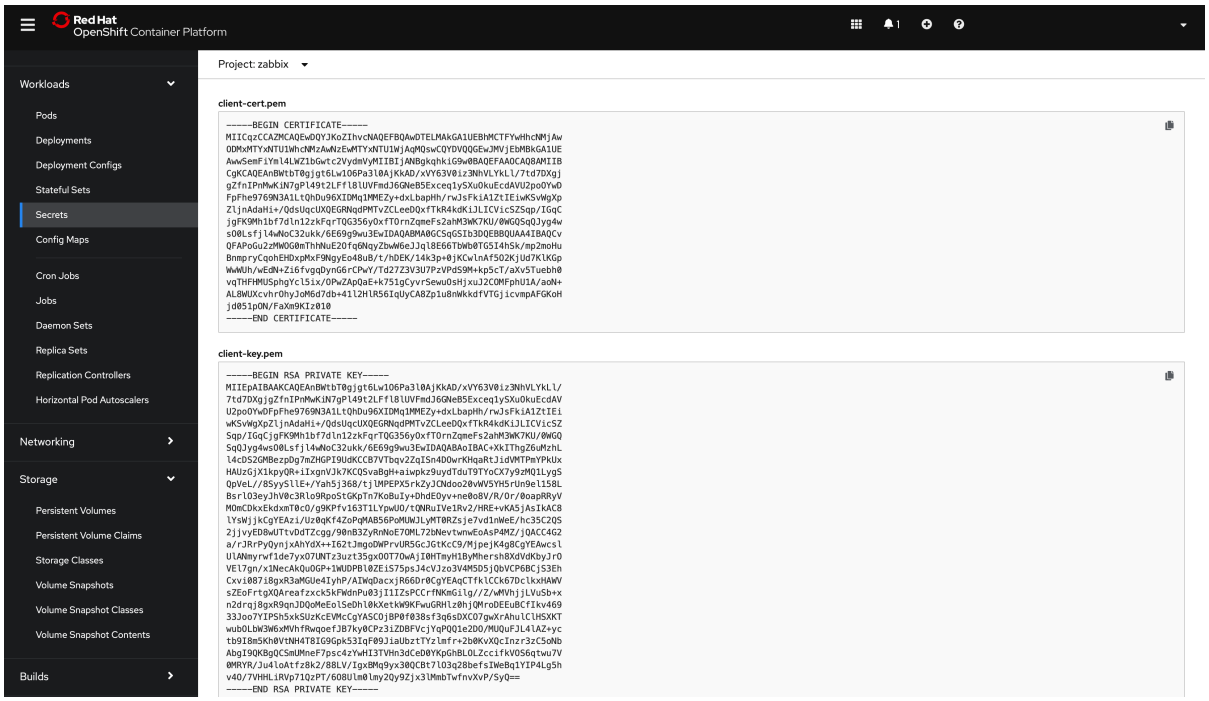
MySQL database side:

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

Zabbix components side:

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

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



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

Known issues

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

6 Web 界面安装

以下部分介绍如何一步一步安装 Zabbix Web 界面。Zabbix Web 界面使用 PHP 语言编写, 所以 Zabbix Web 界面必须在支持 PHP 环境的 web 服务器上运行.

Note:
如果需要使用除了英语以外的其他语言，在 Web 服务器上必须安装相关的语言支持. 如果有需要使用其他的语言环境，在请查看”User profile” 的相关章节” 另请参见”

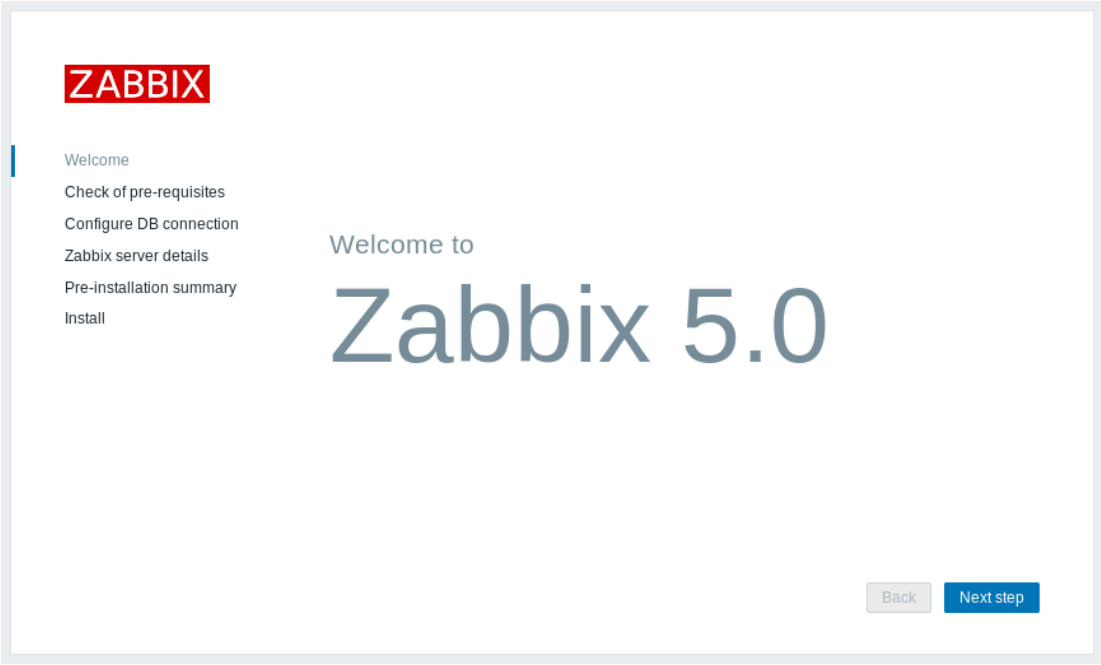
Note:
You can find out more about setting up SSL for Zabbix frontend by referring to these best practices.

欢迎页面

从浏览器上打开 Zabbix 前端访问 URL. 如果你是从 packages 方式安装 Zabbix, URL 是:

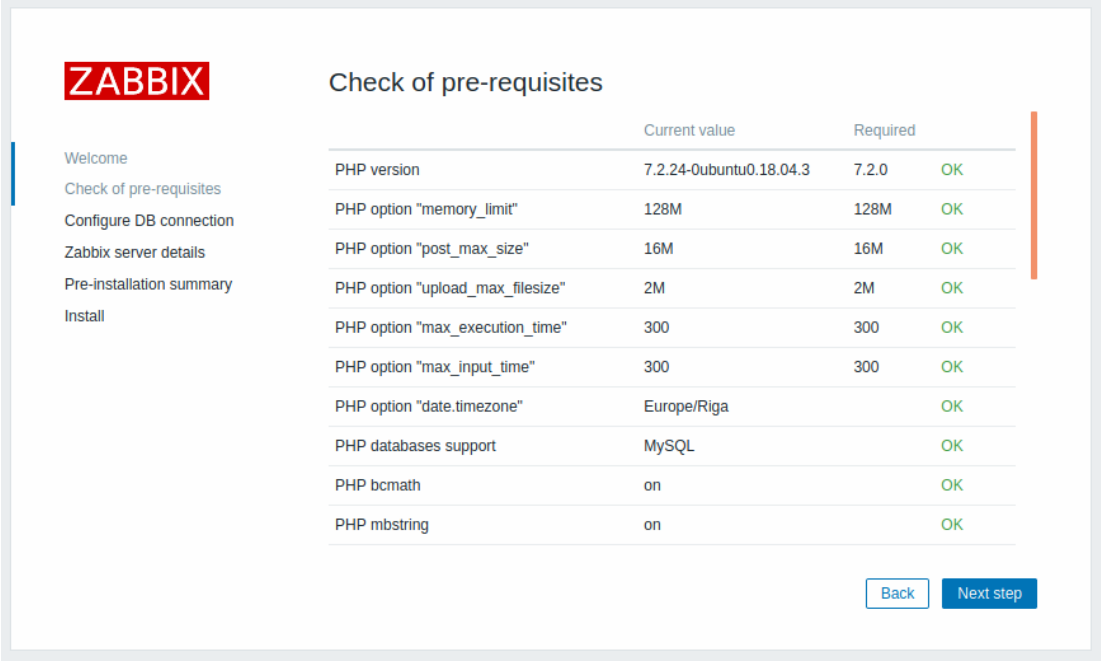
- 对应 Apache: *http://<server_ip_or_name>/zabbix*
- 对应 Nginx: *http://<server_ip_or_name>*

您看到的第一个 Web 前端安装向导页面如下.



先决条件检查

请确保先满足所有软件先决条件.



先决条件最小满	要求描述	
PHP version	7.2.0	
PHP memory_limit option	128MB	在配置文件 php.ini: memory_limit = 128M
PHP post_max_size option	16MB	在配置文件 php.ini: post_max_size = 16M
PHP upload_max_filesize option	2MB	在配置文件 php.ini: upload_max_fi = 2M
PHP max_execution_time option	300 秒 (0 和 -1 值被允许) 在配置文件	php.ini: max_execution = 300
PHP max_input_time option	300 秒 (0 和 -1 值被允许) 在配置文件	php.ini: max_input_tim = 300
PHP session.auto_start option	必须关闭在配置	件 php.ini: session.auto_s = 0
Database support	其中之一: MySQL, Oracle, PostgreSQL. 以下必	安装其中一个模块: mysql, oci8, pgsql
bcmath		php-bcmath
mbstring		php-mbstring
PHP mbstring.func_overload option	必须关闭在配置	件 php.ini: mbstring.func_ = 0

先决条件最小满	要求描述	
sockets		php-net-socket. 需要用户脚本支持.
gd	2.0.28	php-gd. PHP GD 扩展性必须支持 PNG 格式的图片 (--*with-png-dir*), JPEG (--*with-jpeg-dir*) images and FreeType 2 (--*with-freetype-dir*).
libxml	2.6.15	php-xml
xmlwriter		php-xmlwriter
xmlreader		php-xmlreader
ctype		php-ctype
session		php-session

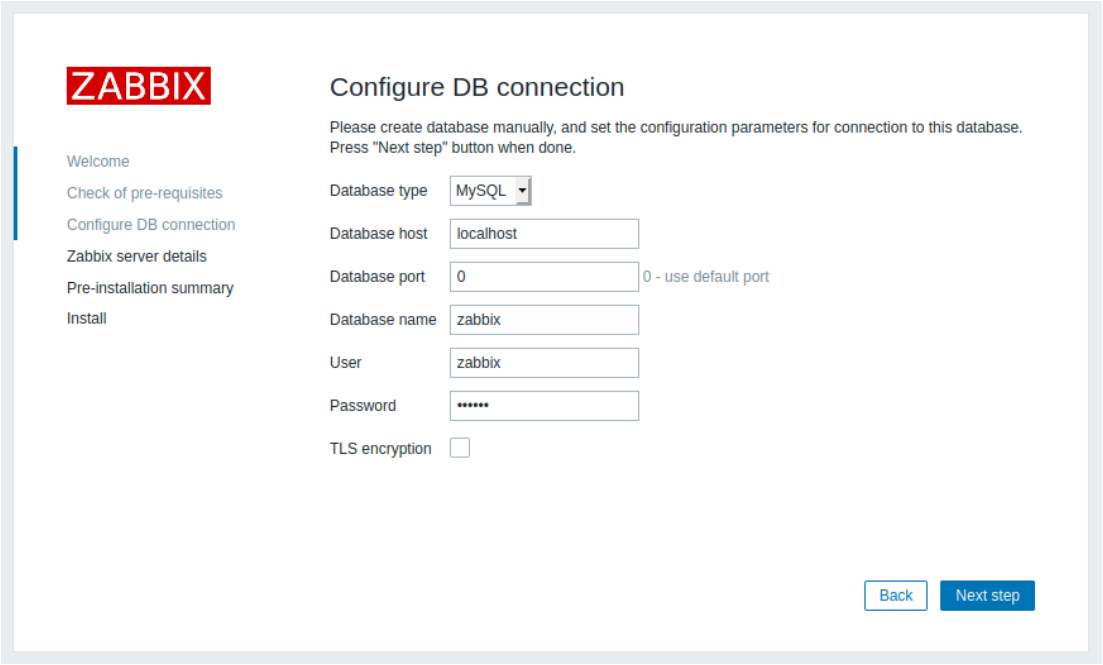
先决条件最小满	要求描述	
gettext		php-gettext 从 Zabbix 2.2.1 版本开始, PHP gettext 扩展性不是安装 Zabbix 的强制性要求. 如果没有安装 gettext, 前端也能够正常运行, 然而, 翻译将不可用.

可选的先决条件也可能出现在清单中。失败的可选先决条件显示为橙色，并处于 *Warning* 状态。如果可选前提条件失败，安装程序也可以继续。

<note important> 如果需要更改 Apache 用户或用户组，则必须验证对会话文件夹的权限。否则，Zabbix 安装程序可能无法继续。

配置数据库连接

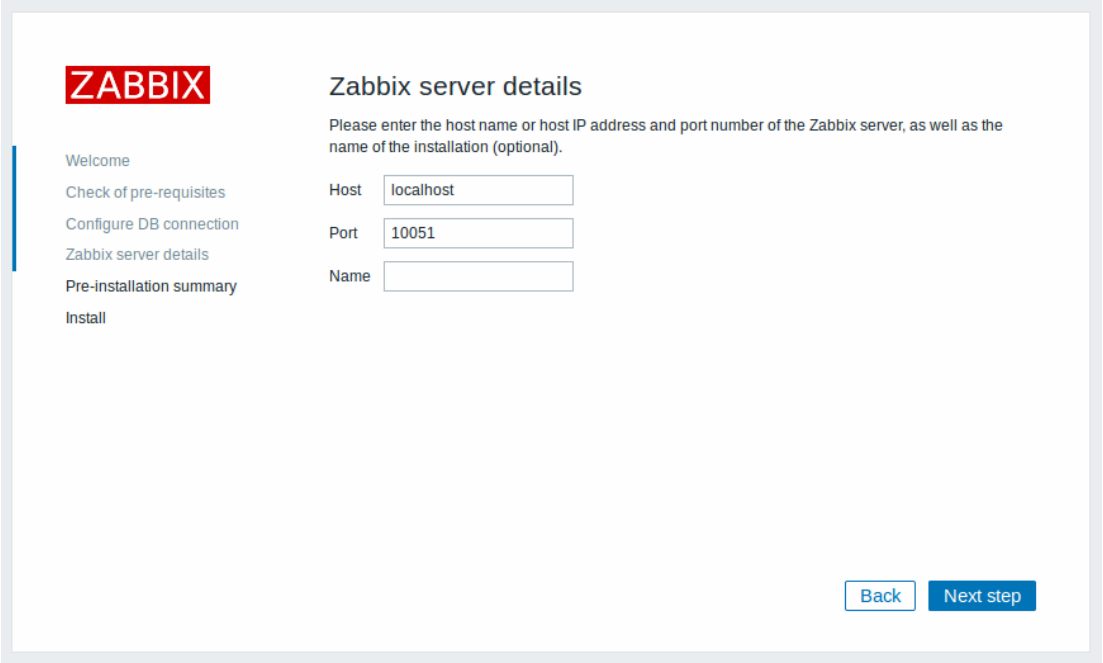
输入连接数据库所需的详细信息. Zabbix 数据库必须先建立好.



如果选中 *TLS* 加密选项选项, 额外需要填写的字段会显示并需要填写配置 TLS 连接信息 (只支持 MySQL 或 PostgreSQL 数据库).

Zabbix 服务器详情

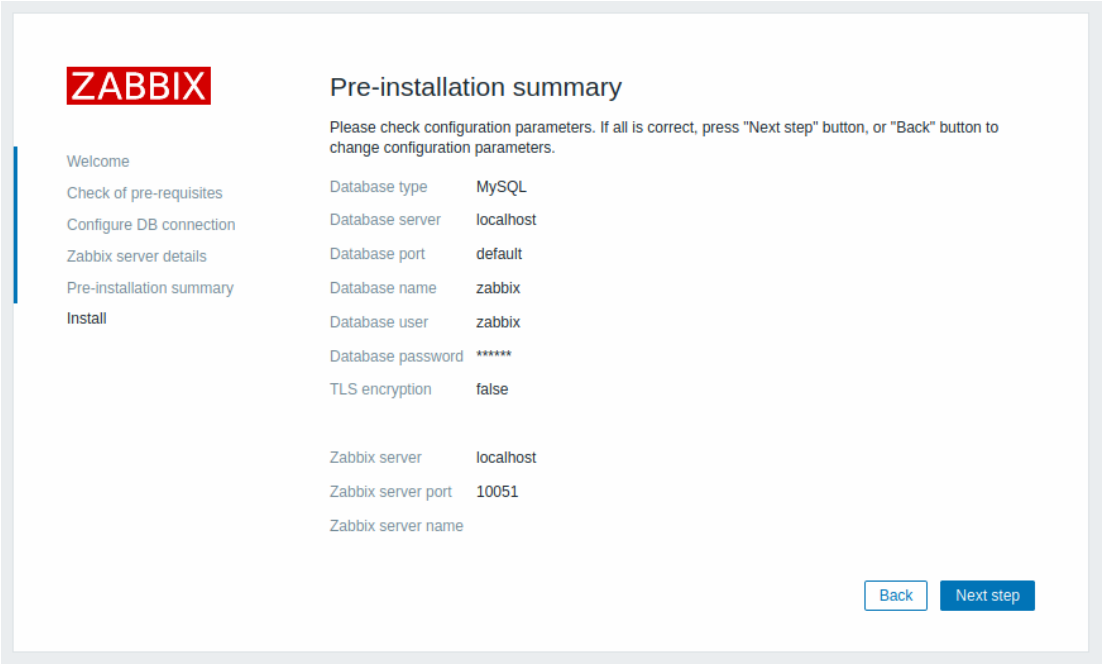
请输入 Zabbix 服务器详情.



可选的输入 Zabbix 服务器的名字, 然而, 如果输入并提交了, Zabbix 服务器的名字将会显示在菜单和页面的标题.

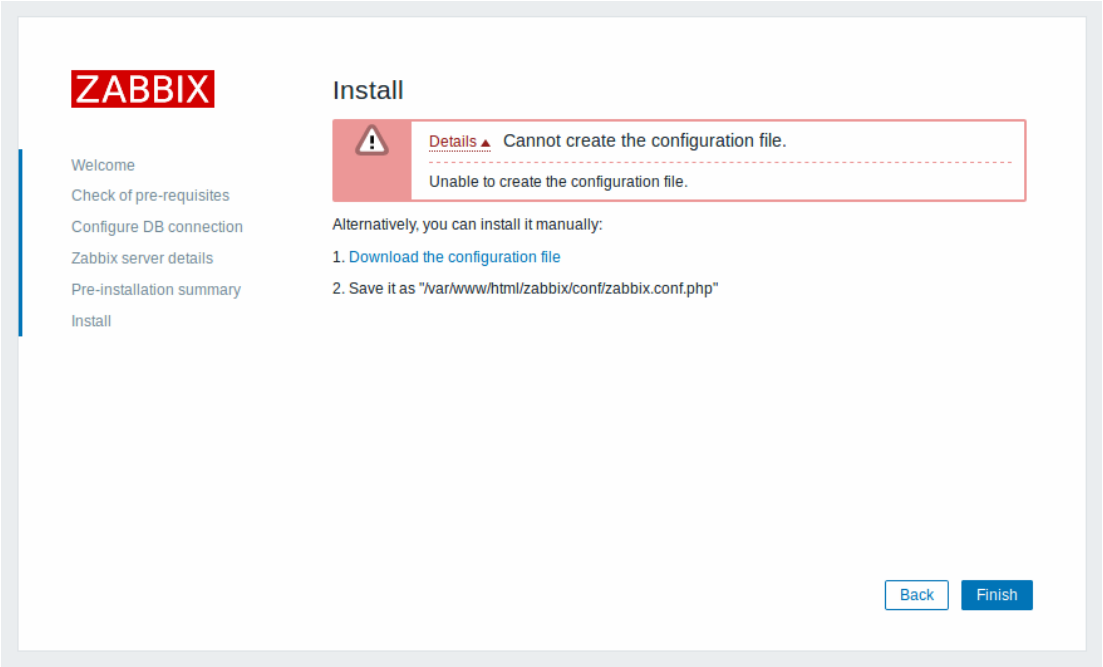
安装前总结

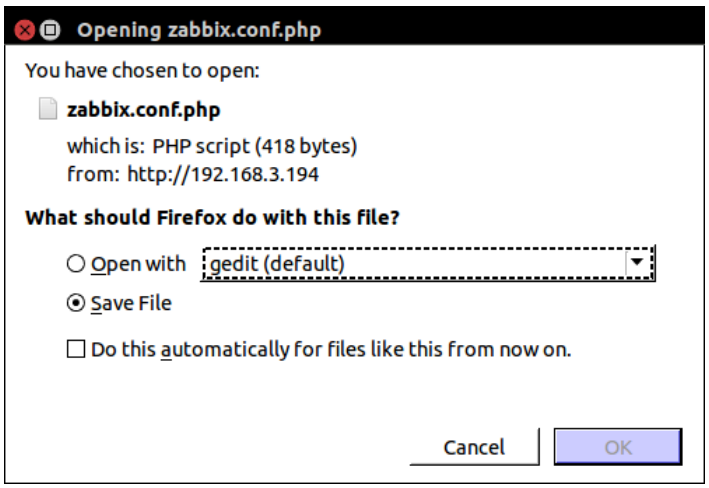
回顾所有配置.



Install

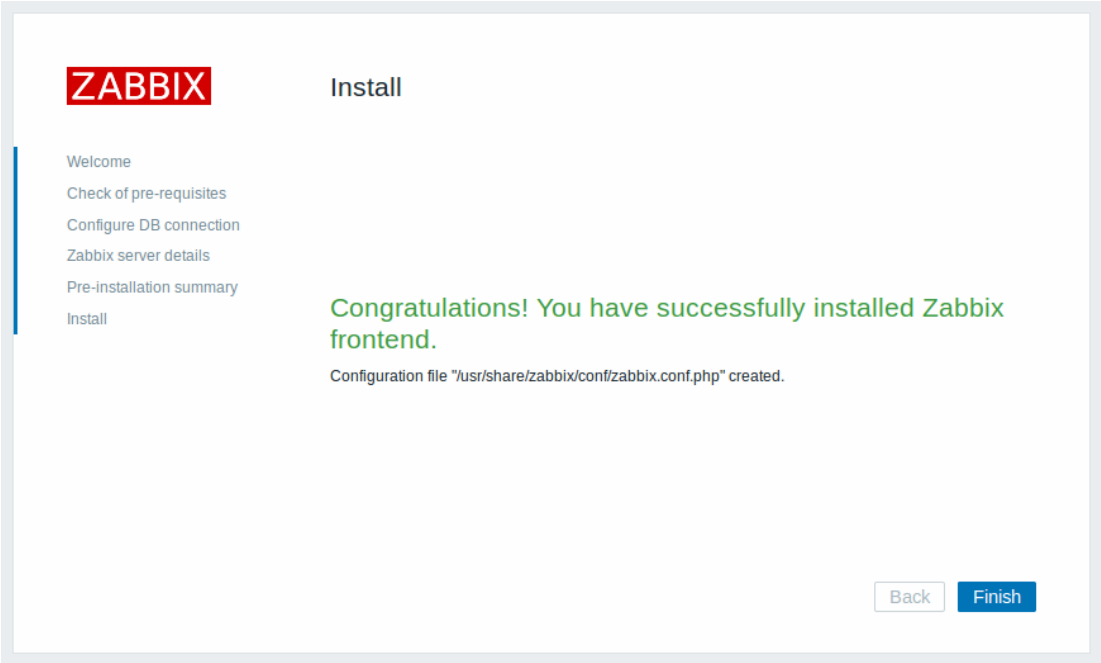
如果是从源代码处安装，请下载配置文件，并将其放在 web 服务器 HTML documents 子目录下，您所复制的 Zabbix PHP 文件的 conf/目录下。.





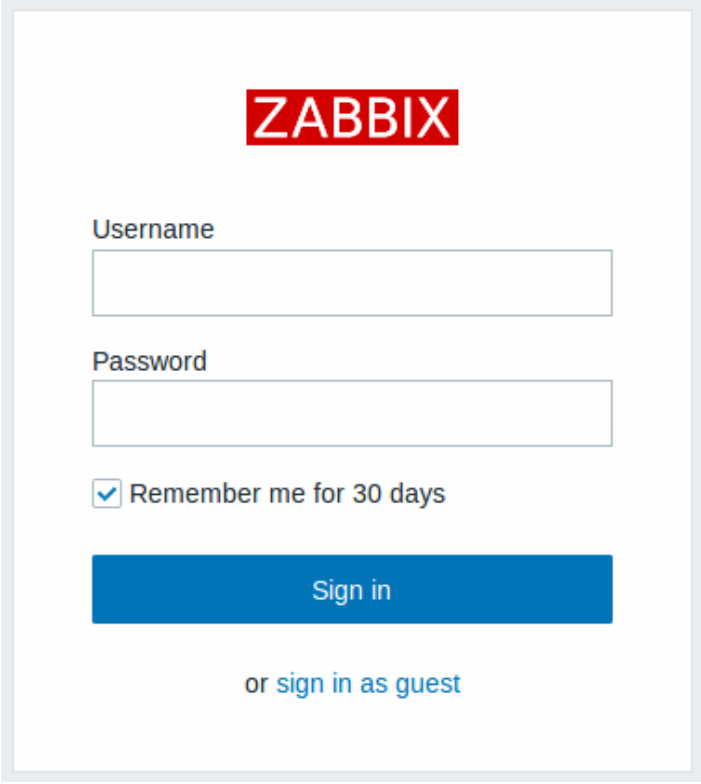
<note tip> 如果 webserver 用户对 conf/目录有写访问权，配置文件将自动保存，并且可以立即进入下一步.

完成安装.



登录

Zabbix 前端已经安装完成! 缺省用户名是 **Admin**, 密码 **zabbix**.



下一步开始你的 Zabbix 旅行.

1 RHEL/CentOS 7 前端安装

概述

从 Zabbix 5.0 版本开始，Zabbix 前端需要 PHP 7.2 版或更高版本。非常不幸的是, RHEL/CentOS 7 缺省只提供 PHP 5.4 版本. 本章节介绍在 RHEL/CentOS 7 上安装 Zabbix 前端的建议方法。

使用 Red Hat 软件集合中的 PHP 和 Nginx

如果你从官方提供的安装包repo.zabbix.com完成了 Zabbix 5.0 的干净的安装, 使用 yum 搜索 Zabbix 时，您可能会注意到缺少前端包。.

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

这是因为前端包被移动到了一个专用的前端子目录 `frontend` .
然而, Zabbix 前端是可以被安装的，前提是 PHP 7.2 依赖条件已经提供.

Note:
为了方便起见，已经从主 `zabbix-web` 包中删除了对 PHP 的任何直接依赖。这为解决 PHP7.2 依赖关系的方法提供了更大的灵活性。

建议使用 Red Hat 软件集合中的 PHP 包。Red Hat Software Collections.
启用 PHP 包，执行:

在 RHEL 环境下

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

在 CentOS 环境下

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

在 Oracle Linux 环境下

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

此时，执行

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

会返回显示新的 rh-php7* 列表.

然后, 编辑 `/etc/yum.repos.d/zabbix.repo` 文件 (如果没有此文件, 先安装 zabbix-release). 打开 `zabbix-frontend` 存储库.

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

把 `enabled=0` 替代成 `enabled=1`.

在此阶段，通过 `yum` 搜索 `Zabbix` 将返回 `zabbix-web` 包和四个新包。这四个包是:

```
zabbix-nginx-conf-scl.noarch : Nginx的Zabbix前端配置 (scl 版本)
zabbix-web-deps-scl.noarch : 用于从redhat软件集合安装zabbix-web包所需PHP依赖项的便利包
zabbix-web-mysql-scl.noarch : 用于MySQL数据库的Zabbix web前端包 (scl 版本)
zabbix-web-pgsql-scl.noarch : 用于PostgreSQL数据库的Zabbix web前端包(scl 版本)
```

在安装 MySQL 数据库所需的 `zabbix-web-mysql-scl` 或者 PostgreSQL 数据库所需的 `zabbix-web-pgsql-scl`. 取决于 Web 服务器的需要，也请安装 `zabbix-apache-conf-scl` 或者 `zabbix-nginx-conf-scl`.

Note:
在 Zabbix 4.4 版本中，已经加入了对 Nginx 的支持, 但是官方的 RHEL/CentOS 7 存储库中没有可用的 web 服务器. 因此, 它必须通过第三方仓库提供, 由用户安装。尤其是 `epel`. 在 Zabbix 5.0, 如果您选择使用 Red Hat 软件集合, 无需使用任何第三方存储库，因为 SCL 中提供 Nginx. 只需安装 `zabbix-nginx-conf-scl` 包.

新包的技术细节

zabbix-web-deps-scl

这个包用于从 Red Hat 软件集合中提取 Zabbix 前端的常见 PHP 依赖项。.

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

它还包含用于 Zabbix 的 php fpm 池，因为在这种配置中，前端可以通过 fastcgi 与 Apache 和 Nginx 一起工作。
配置文件位于/etc/opt/rh/rh-php72/php-fpm.d/zabbix.conf.

zabbix-web-mysql-scl

元软件包用于获取 zabbix-web 包、PHP 对 MySQL 数据库模块的支持以及常见的 PHP 依赖项.

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

zabbix-web-pgsql-scl

元软件包用于获取 zabbix-web 包、PHP 对 PostgreSQL 数据库模块的支持以及常见的 PHP 依赖项。

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

zabbix-apache-conf-scl

这个包用于获取 apache 并包含/etc/httpd/cond.d/zabbix.conf 文件.

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

zabbix-nginx-conf-scl

这个包用于从 Red Hat 软件集合中提取 Nginx.

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

它还包含 Nginx 服务器所需的 Zabbix 配置文件，文件在 /etc/opt/rh/rh-nginx116/nginx/conf.d/zabbix.conf.

使用第三方 PHP 存储库

如果由于某些原因不能够使用 Red Hat 软件集合, 可以用以下的替代办法:

- 使用任何可以提供 PHP 的第三方存储库。
- 从源代码构建 PHP。

Zabbix 前端所需的 PHP 模块是 php-gd, php-bcmath, php-mbstring, php-xml, php-ldap 和 php-json.

从旧版本 Zabbix 升级至 Zabbix 5.0 版本

在旧版本升级至 Zabbix 5.0 版本时，需要特别注意一些事项。

<note important>请查看通用升级指引. :::

Red Hat 软件集合中的包旨在避免与主存储库中的文件冲突.
每一个特定的包都被安装到一个单独的环境中，专门用于它的组.
例如, 来自 rh-php72-php* 组的在 /etc/opt/rh/rh-php72/ 目录下会有对应的配置文件, 日志会生成在 /var/opt/rh/rh-php72/log/ 目录下, 等等. 这些包提供的服务具有不寻常的名称，如 rh-php72-php-fpm or rh-nginx116-nginx。

官方的 zabbix5.0 前端包将 php-fpm 与 Apache 和 Nginx 结合使用

在 Apache 环境下的升级进程

本章节提供了有关将 Zabbix 前端和服务端从 4.0 版本或 4.4 版本升级到 5.0 版本，与 Apache 相关的特定说明. 与 Nginx 相关的指引请参见在 Nginx 环境下的升级进程.
下面的说明是针对已经安装 MySQL 支持的 Zabbix 服务。将命令中的 “mysql” 替换为 “pgsql” 可以适用于 PostgreSQL 数据库
下面假设 Zabbix 前端和 Zabbix 服务安装在同一台服务器上。如果您的 Zabbix 相关服务安装与之不同，请根据实际情况调整。
清除旧的 **Zabbix** 前端

在升级开始之前，你必须把已有的 Zabbix 前端清除。旧的配置文件将会被 rpm 移动到 /etc/httpd/conf.d/zabbix.conf.rpmsave.

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

安装 **SCL** 存储库

在 RHEL 环境下执行

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

在 CentOS 环境下执行

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

在 Oracle Linux 环境下执行

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

安装 **Zabbix 5.0** 发行包并启用 **Zabbix** 前端存储库

安装 zabbix-release-5.0 包.

```
rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el7.noarch.rpm
yum clean all
```

编辑 /etc/yum.repos.d/zabbix.repo file. 把 enabled=0 替换成 enabled=1.

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

安装新的前端包

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

官方 Zabbix 5.0 前端包使用 php-fpm. 在/etc/opt/rh/rh-php72/php-fpm.d/zabbix.conf 文件中更新时区.

** 更新剩余的包并重启 Zabbix server**

```
yum update zabbix-*
```

> **Attention:**
> 重启 Zabbix server 将会升级数据库. 请确保数据库已经备份.

```
systemctl restart zabbix-server
```

** 更新剩余的服务 **

启用并开启 php-fpm 服务。

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

重启 Apache.

```
systemctl restart httpd
```

在 Nginx 环境下的升级进程

遵循上面描述的 apache 升级过程，但做一些调整.
以下几个步骤需要执行:

在升级之前，请确保停止并禁用旧的 Nginx 和 php-fpm。执行:

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

为 php-fpm 编辑 `zabbix.conf` 文件时, 添加用户 `nginx` 到 `listen.acl_users`

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

确保 `zabbix-nginx-conf-scl` 包已经被安装，而不是 `zabbix-apache-conf-scl` 包被安装.

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

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

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

启动并启用 Nginx 和 php-fpm

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

Updating to PHP 7.3 packages

Overview

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

Steps

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

```
# yum search zabbix-web

...

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

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

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

or

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

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

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

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

Set your timezone.

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

4. Update the web server configuration

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

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

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

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

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

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

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

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

Restart the web server.

```
systemctl restart httpd
```

or

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

Remove the old rh-php72 packages.

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

2 Debian/Ubuntu 前端安装

概述

从 Zabbix 5.0 版本开始，Zabbix 前端需要 PHP 7.2 版或更高版本。非常不幸的是, 旧版本的 Debian & Ubuntu 提供 PHP 7.2 以下的版本。

发行版支持的 PHP 版本

发行版版本 PHP	本
Debian 10 (buster)	7.3
Debian 9 (stretch)	7.0
Debian 8 (jessie)	5.6
Ubuntu 20.04 (focal)	7.4
Ubuntu 18.04 (bionic)	7.2
Ubuntu 16.04 (xenial)	7.0
Ubuntu 14.04 (trusty)	5.5
Raspbian 10 (buster)	7.3
Raspbian 8 (stretch)	7.0

在 *stretch, jessie, xenial* 和 *trusty* 的发行版中, PHP 7.2 依赖并不可用, 因此 Zabbix 前端或更高版本不能简单地安装. 考虑到这方面的原因, 在上述发行版上，`zabbix-frontend-php` 包已经被替换成为 `zabbix-frontend-php-deprecated`。
主要区别在于没有对任何 php 或 web 服务器包的直接依赖。因此，用户可以（而且必须）自己提供这些依赖关系. 换句话说, 安装 `zabbix-frontend-php-deprecated` 包并不会提供可用的 Zabbix 前端. 必须手动安装 web 服务器以及 PHP7.2 及其模块（从源代码处使用 PPAs/build PHP）. 我们不支持任何特别的方法.

> **Note:**
> 在老版本的 Debian/Ubuntu 上获得 php7.2 或更高版本的官方方法是升级到 buster/bionic 发行版

Zabbix 前端所需的 PHP 模块是 `php-gd`, `php-bcmath`, `php-mbstring`, `php-xml`, `php-ldap` 和 `php-json`.

7 升级步骤

概述

本章节提供了关于升级至 Zabbix **5.0** 的信息：

- 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。

可以从 Zabbix 4.4.x，4.2.x，4.0.x，3.4.x，3.2.x，3.0.x，2.4.x，2.2.x 和 2.0.x 直接升级到 Zabbix 5.0.x. 要从早期版本进行升级，请参阅 Zabbix 文档以获取 2.0 或更早的版本。

从二进制包升级

概述

本节提供使用 Zabbix 提供的官方 RPM 和 DEB 软件包成功升级所需的步骤，以用于：

- Red Hat Enterprise Linux/CentOS
- Debian/Ubuntu

操作系统存储库中的 ZABBIX 软件包通常，操作系统发行版（尤其是基于 Debian 的发行版）提供了自己的 Zabbix 软件包。请注意，Zabbix 不支持这些软件包，它们通常已过时，并且缺少最新功能和错误修复。官方仅支持https://repo.zabbix.com/□repo.zabbix.com中的软件包。

如果要从 OS 发行版提供的软件包升级（或在某个时候安装了它们），请按照以下过程切换到官方 Zabbix 软件包：

1. 请先卸载旧软件包。
2. 检查卸载后可能剩余的残留文件。
3. 按照 Zabbix 提供的https://www.zabbix.com/download?zabbix=5.0□□□□□安装官方软件包。

请勿进行直接更新，因为这可能会导致安装失败。

Zabbix packages from OS repositories

Often, OS distributions (in particular, Debian-based distributions) provide their own Zabbix packages.
Note, that these packages are not supported by Zabbix, they are typically out of date and lack the latest features and bug fixes.
Only the packages from repo.zabbix.com are officially supported.

If you are upgrading from packages provided by OS distributions (or had them installed at some point), follow this procedure to switch to official Zabbix packages:

1. Always uninstall the old packages first.
2. Check for residual files that may have been left after deinstallation.
3. Install official packages following installation instructions provided by Zabbix.

Never do a direct update, as this may result in a broken installation.

1 Red Hat Enterprise Linux/CentOS

概述

本节提供了使用用于 Red Hat Enterprise Linux / CentOS 的官方 Zabbix 软件包从 Zabbix 4.4.x 成功升级到 Zabbix 5.0.x所需的步骤。

虽然不是强制性升级 Zabbix agent（但建议升级），但是 Zabbix server 和 agent 必须具有相同的主版本。因此，在 server-proxy 设置中，必须停止并升级 Zabbix server 和所有 agent。在 agent 升级期间，不再使 agent 保持运行状态将带来任何好处，因为在 agent 升级期间，其旧数据将被丢弃，并且在 agent 配置与 server 同步之前不会收集新数据。

请注意，对于 agent 上的 SQLite 数据库，升级之前来自 agent 的历史记录数据将丢失，因为不支持 SQLite 数据库升级，并且必须手动删除 SQLite 数据库文件。首次启动 agent 并且缺少 SQLite 数据库文件时，agent 会自动创建它。

根据数据库大小，数据库升级到版本 5.0 可能会花费很长时间。

<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:**
> RHEL / CentOS 7 用户，升级前端时请注意 PHP> = 7.2 版本要求！请务必阅读相关文档！

请阅读下面的升级说明：

早期版本详细的	本升级说明版本之间升级的重要说明/	更
4.4.x	For: Zabbix5.0	IBM DB2 的支持下降;所需的最低 PHP 版本从 5.4.0 升级到 7.2.0；升级了所需的最低数据库版本。
4.2.x	For: Zabbix 4.4 Zabbix5.0	Jabber，Ez Tex-ting 媒体类型已删除。

早期版本详细的	本升级说明版本之间升级的重要说明/	更
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix5.0	较旧的代理不再可以将数据报告给已升级的服务器;较新的 agent 不再能够与较旧的 Zabbix server 一起使用。

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	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix5.0	‘libpthread’和‘zlib’库现在是必需的;支持删除纯文本协议，并且标头是必需的;不再支持1.4版之前的Zab-bix agent;现在，被动代理配置中的Server参数是必需的。

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	For: Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix5.0	SQLite 支持作为 Zabbix server/前端的后端数据库删除; 支持 Perl 兼容正则表达式 (PCRE), 而不是 POSIX 扩展; Zabbix server 必需的'libpcre' 和'libevent' 库; 添加了退出代码检查, 以检查用户参数, 远程命令和 system.run []

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 LTS	For: Zabbix3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix5.0	数据库升级可能会很慢，具体取决于历史记录表的大小。

您可能还需要检查 5.0 的要求。

<note tip> 在升级过程中运行两个并行的 SSH 会话可能很方便，在一个过程中执行升级步骤，而在另一个过程中监视 server/proxy 日志。例如，在第二个 SSH 会话中运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，向您显示最新的日志文件条目和实时可能的错误。这对于生产实例可能至关重要。。:::

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。

```
# systemctl stop zabbix-server
```

如果需要升级 Zabbix proxy，那么同样停止 Zabbix proxy 进程。

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

<note important> 无法再启动已升级的 server，并且无法使用较旧但尚未升级的 proxy 将数据报告给较新的 server。从 4.1 之前的任何版本升级到 5.0（或更高版本）时，此方法从未被 Zabbix 推荐或支持，现已正式禁用，因为 server 将忽略来自未升级 proxy 的数据。:::

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。

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。

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:

RHEL/CentOS 8

```
# rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
```

RHEL/CentOS 7

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

5 升级 Zabbix 组件

运行以下命令以升级 Zabbix 组件：

```
# yum upgrade zabbix-server-mysql zabbix-web-mysql zabbix-agent
```

如果使用 PostgreSQL，请在命令中将 `mysql` 替换为 `pgsql`。如果要升级 proxy，请在命令中用 `proxy` 替换 `server`。如果要升级 agent 2，请在命令中将 `zabbix-agent` 替换为 `zabbix-agent2`。

要在 RHEL 8 上使用 Apache 正确升级 Web 前端，请运行：

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

并对此文件进行必要的更改。

更改要升级 RHEL 7 上的 Web 前端，请遵循本页上的说明（安装 PHP 7.2 或更高版本需要额外的步骤）。特别是，如果使用 Apache Web 服务器，请确保安装 `zabbix-apache-conf-scl` 软件包。

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

6 检查 Zabbix 组件配置文件的参数

有关强制性更改的详细信息，请参阅升级说明。

7 启动 Zabbix 进程

启动升级后的 Zabbix 组件。

```
# systemctl start zabbix-server
# systemctl start zabbix-proxy
# systemctl start zabbix-agent
# systemctl start zabbix-agent2
```

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。

Zabbix 次要版本之间的升级

可以在 5.0.x 的次要版本之间进行升级（例如，从 5.0.1 升级到 5.0.3）。在次要版本之间升级很容易。

要执行 Zabbix 次要版本升级，需要运行：

```
$ sudo yum upgrade 'zabbix-*'
```

在升级 Zabbix server 的次要版本时，只需运行以下命令：

```
$ sudo yum upgrade 'zabbix-server-*'
```

在升级 Zabbix agent 的次要版本时，只需运行以下命令：

```
$ sudo yum upgrade 'zabbix-agent-*'
```

或者，对于 Zabbix agent 2：

```
$ sudo yum upgrade 'zabbix-agent2-*'
```

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。

2 Debian/Ubuntu

概述

本节提供了使用 Debian / Ubuntu 官方 Zabbix 软件包从 Zabbix 4.4.x 成功升级到 Zabbix 5.0.x 所需的步骤。

虽然不是强制性升级 Zabbix agent（但建议升级），但是 Zabbix server 和 proxy 必须具有相同的主版本。因此，在 server-proxy 设置中，必须停止并升级 Zabbix server 和所有 proxy。在 proxy 升级期间，继续使 proxy 保持运行不会带来任何好处，因为在 proxy 升级期间，它们的旧数据将被丢弃，并且在 proxy 配置与 server 同步之前不会收集任何新数据。

请注意，对于代理上的 SQLite 数据库，升级之前来自 proxy 的历史记录数据将丢失，因为不支持 SQLite 数据库升级，并且必须手动删除 SQLite 数据库文件。首次启动 proxy 并且缺少 SQLite 数据库文件时，proxy 会自动创建它。

根据数据库大小，数据库升级到版本 5.0 可能会花费很长时间。

<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:
Ubuntu 16.04 或更早版本和 Debian 9 或更早版本的用户请注意您的发行版中缺少官方的 PHP 7.2 软件包。请务必阅读相关文档！

请阅读下面的升级说明：

早期版本详细的	本升级说明版本之间升级的重要说明/	更
4.4.x	For: Zabbix5.0	IBM DB2 的支持下降;所需的最低 PHP 版本从 5.4.0 升级到 7.2.0；升级了所需的最低数据库版本。
4.2.x	For: Zabbix 4.4 Zabbix5.0	Jabber，Ez Tex-ting 媒体类型已删除。

早期版本详细的	本升级说明版本之间升级的重要说明/	更
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix5.0	较旧的代理不再可以将数据报告给已升级的服务器;较新的 agent 不再能够与较旧的 Zabbix server 一起使用。

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	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix5.0	‘libpthread’和‘zlib’库现在是必需的;支持删除纯文本协议，并且标头是必需的;不再支持1.4版之前的Zab-bix agent;现在，被动代理配置中的Server参数是必需的。

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	For: Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix5.0	SQLite 支持作为 Zabbix server/前端的后端数据库删除; 支持 Perl 兼容正则表达式 (PCRE), 而不是 POSIX 扩展; Zabbix server 必需的'libpcre' 和'libevent' 库; 添加了退出代码检查, 以检查用户参数, 远程命令和 system.run []

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 LTS	For: Zabbix3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix5.0	数据库升级可能会很慢，具体取决于历史记录表的大小。

您可能还需要检查 5.0 的要求。

<note tip> 在升级过程中运行两个并行的 SSH 会话可能很方便，在一个过程中执行升级步骤，而在另一个过程中监视 server/proxy 日志。例如，在第二个 SSH 会话中运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，向您显示最新的日志文件条目和实时可能的错误。这对于生产实例可能至关重要。。:::

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。

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

如果需要升级 Zabbix proxy，那么同样停止 Zabbix proxy 进程。

```
# service zabbix-proxy stop
```

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。

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。

配置文件：

```
# 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：

```
# 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：

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

然后再安装新的软件仓库包：

在 **Debian 10** 上运行：

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

在 **Debian 9** 上运行：

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

在 **Debian 8** 上运行：

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

在 **Ubuntu 20.04** 上运行：

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

在 **Ubuntu 18.04** 上运行：

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

在 **Ubuntu 16.04** 上运行：

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

在 **Ubuntu 14.04** 上运行：

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

更新软件仓库信息。

```
# apt-get update
```

5 升级 Zabbix 组件

运行以下命令以升级 Zabbix 组件：

```
# apt-get install --only-upgrade zabbix-server-mysql zabbix-frontend-php zabbix-agent
```

如果使用 PostgreSQL，请在命令中将 `mysql` 替换为 `pgsql`。如果要升级 proxy，请在命令中用 `proxy` 替换 `server`。如果要升级 agent 2，请在命令中将 `zabbix-agent` 替换为 `zabbix-agent2`。

然后，要正确地使用 Apache 升级 Web 前端，还请运行：

```
# apt-get install zabbix-apache-conf
```

Debian 10（buster）/ Ubuntu 18.04（bionic）/ Raspbian 10（buster）之前的发行版不提供 PHP 7.2 或更高版本，这是 Zabbix 前端 5.0 所必需的。有关在较早的发行版上安装 Zabbix 前端的信息，请参阅此页面。

6 检查 Zabbix 组件配置文件的参数

有关强制性更改的详细信息，请参阅升级说明。

有关新的可选参数，请参阅 “新增功能” 部分。

7 启动 Zabbix 进程

启动升级后的 Zabbix 组件。

```
# service zabbix-server start
# service zabbix-proxy start
# service zabbix-agent start
# service zabbix-agent2 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 5.0.1 升级至 5.0.3），是非常容易的：

在升级 Zabbix 所有组件的次要版本时，只需运行以下命令：

```
$ 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：

```
$ 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：

```
$ sudo apt install --only-upgrade 'zabbix-agent.*'
```

或者，对于 Zabbix agent2：

```
$ sudo apt install --only-upgrade 'zabbix-agent2.*'
```

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 4.4.x 成功升级至 Zabbix 5.0.x 所需的步骤。

虽然不是强制性升级 Zabbix agents（但建议升级），但是 Zabbix server 和 proxy 必须具有相同的主版本。因此，在 server-proxy 设置中，必须停止并升级 Zabbix server 和所有 proxy。保持 proxy 不再运行将带来任何好处，因为在 proxy 升级期间，它们的旧数据将被丢弃，并且在 proxy 配置与 server 同步之前不会收集新数据。

<note important> 无法再启动已升级的 server，并且无法使用较旧但尚未升级的 proxy 将数据报告给较新的 server。当从 5.0 之前的任何版本升级到 5.0（或更高版本）时，此方法从未被 Zabbix 推荐或支持，现已正式禁用，因为 server 将忽略来自未升级 proxy 的数据。:::

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。

根据其数据库大小，数据库升级到 5.0 版本可能需要很长时间。

<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！ :::

请阅读下面的升级说明：

早期版本详细的	本升级说明版本之间升级的重要说明/	更
4.4.x	For: Zabbix 5.0	对 IBM DB2 的支持下降了；所需的最低 PHP 版本从 5.4.0 升级到 7.2.0；升级了所需的最低数据库版本；更改了 Zabbix PHP 文件目录。
4.2.x	For: Zabbix 4.4 Zabbix 5.0	Jabber，Ez Texting 媒体类型已删除。

早期版本详细的	本升级说明版本之间升级的重要说明/	更
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	较旧的proxy不再可以将数据报告给已升级的server；较新的proxy不再能够与较旧的Zab-bix server一起使用。

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	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	‘libpthread’和‘zlib’库现在是必需的；支持删除纯文本协议，并且标头是必需的；不再支持1.4版之前的Zabbix agents；现在，被动代理配置中的Server参数是必需的。

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	For: Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	SQLite 支持作为 Zabbix server/frontend 的后端数据库删除；支持 Perl 兼容正则表达式（PCRE），而不是 POSIX 扩展；Zabbix server 必需的’libpcre’和’libevent’库；添加了退出代码检查，以检查用户参数，远程命令和 system.run[] 项

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 LTS	For: Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	数据库升级可能会很慢，具体取决于历史记录表的大小。
2.4.x	For: Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	所需的最低 PHP 版本从 5.3.0 升级到 5.4.0 必须指定 Log-File agent 参数
2.2.x LTS	For: Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	基于节点的分布式监视已删除

早期版本详细的	本升级说明版本之间升级的重要说明/	更
2.0.x	For: Zabbix 2.2 Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	所需的最低 PHP 版本从 5.1.6 升级到 5.3.0；区分大小写的 MySQL 数据库是服务器正常工作所必需的；字符集 utf8 和 utf8_bin 排序规则是 Zab-bix server 与 MySQL 数据库一起正常工作所必需的。

早期版本详细的	本升级说明版本之间升级的重要说明/	更

您可能还需要检查 5.0 的要求。

<note tip> 在升级过程中运行两个并行的 SSH 会话可能很方便，在一个过程中执行升级步骤，而在另一个过程中监视 server/proxy 日志。例如，在第二个 SSH 会话中运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，向您显示最新的日志文件条目和实时可能的错误。这对于生产实例可能至关重要。:::

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。

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。

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。

4 从源代码包安装新的 Zabbix server

使用此说明 从源代码编译 Zabbixserver。

5 检查 Zabbix server 配置文件的参数

有关强制性更改的详细信息，请参阅升级说明。

有关新的可选参数，请参阅 “5 Zabbix 5.0.0 新特征| 新增功能]]” 部分。

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。

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。

```
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：

- 请确保数据库用户拥有足够的权限（create table, drop table, create index, drop index）；
- 请确保磁盘有足够的空间。

7 安装新的 Zabbix web 界面

其最小的需要为 PHP 7.2.0 版本。如果升级请按照安装说明进行操作。

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。

Zabbix Proxy 升级步骤

1 停止 Zabbix proxy 进程

停止 Zabbix proxy 进程。

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。

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。

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 5.0.0 新特征章节。

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。

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 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。

本节中描述的升级过程可用于升级 Zabbix agent 和 Zabbix agent 2。

1 停止 Zabbix agent 进程

停止 Zabbix agent 进程。

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。

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。

或者，从 Zabbix 下载页面 下载预编译的 Zabbix agent 包。

4 检查 Zabbix agent 配置文件的参数

此版本中没有对agent或agent 2参数进行任何强制性更改。

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。

Zabbix 次要版本之间的升级

在 5.0.x 的次要版本之间升级（例如，从 5.0.1 升级到 5.0.3）时，需要对 server/proxy/agent 执行与主要版本之间升级相同的操作。唯一的区别是，在次要版本之间升级时，不会对数据库进行任何更改。

8 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。

. ###### Installation from packages

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

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

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

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

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。

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。

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。

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。

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。

Possible deadlocks with MySQL/MariaDB

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

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。

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。

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。

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。

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）。

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。

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。

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`。

Simple checks

There is a bug in **fping** versions earlier than v3.10 that mishandles duplicate echo replay packets. This may cause unexpected results for `icmpping`, `icmppingloss`, `icmppingsec` items. It is recommended to use the latest version of **fping**. Please see ZBX-11726 for more details.

Errors with fping execution in rootless containers

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

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

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

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

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

SNMP checks

If the OpenBSD operating system is used, a use-after-free bug in the Net-SNMP library up to the 5.7.3 version can cause a crash of Zabbix server if the SourceIP parameter is set in the Zabbix server configuration file. As a workaround, please do not set the SourceIP parameter. The same problem applies also for Linux, but it does not cause Zabbix server to stop working. A local patch for the net-snmp package on OpenBSD was applied and will be released with OpenBSD 6.3.

SNMP data spikes

Spikes in SNMP data have been observed that may be related to certain physical factors like voltage spikes in the mains. See ZBX-14318 more details.

SNMP traps

The "net-snmp-perl" package, needed for SNMP traps, has been removed in RHEL/CentOS 8.0-8.2; re-added in RHEL 8.3.

So if you are using RHEL 8.0-8.2, the best solution is to upgrade to RHEL 8.3; if you are using CentOS 8.0-8.2, you may wait for CentOS 8.3 or use a package from EPEL.

Please also see ZBX-17192 for more information.

Alerter process crash in Centos/RHEL 7

Instances of a Zabbix server alerter process crash have been encountered in Centos/RHEL 7. Please see ZBX-10461 for details.

Flipping frontend locales

It has been observed that frontend locales may flip without apparent logic, i. e. some pages (or parts of pages) are displayed in one language while other pages (or parts of pages) in a different language. Typically the problem may appear when there are several users, some of whom use one locale, while others use another.

A known workaround to this is to disable multithreading in PHP and Apache.

The problem is related to how setting the locale works in PHP: locale information is maintained per process, not per thread. So in a multi-thread environment, when there are several projects run by same Apache process, it is possible that the locale gets changed in another thread and that changes how data can be processed in the Zabbix thread.

For more information, please see related problem reports:

- ZBX-10911 (Problem with flipping frontend locales)
- ZBX-16297 (Problem with number processing in graphs using the `bcdiv` function of BC Math functions)

PHP 7.3 opcache configuration

If "opcache" is enabled in the PHP 7.3 configuration, Zabbix frontend may show a blank screen when loaded for the first time. This is a registered PHP bug. To work around this, please set the "opcache.optimization_level" parameter to `0x7FFFBFDF` in the PHP configuration (php.ini file).

Graphs

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

Log file monitoring

`log[]` and `logrt[]` items repeatedly reread log file from the beginning if file system is 100% full and the log file is being appended (see ZBX-10884 for more information).

Slow MySQL queries

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

Permission checks for dashboards with graphs

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

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

For MySQL, run:

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

For PostgreSQL, run:

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

For Oracle, run:

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

Creation of deterministic triggers should be enabled for the time of patch application. On MySQL and MariaDB, this requires `GLOBAL log_bin_trust_function_creators = 1` to be set if binary logging is enabled and there is no superuser privileges and `log_bin_trust_function_creators = 1` is not set in MySQL configuration file. To set the variable using MySQL console, run:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Once the patch has been successfully applied, `log_bin_trust_function_creators` can be disabled:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
```

Triggers are also created for PostgreSQL and Oracle database.

Non-existing item value retrieval with MySQL 5.6/5.7

Zabbix server generates slow select queries in case of non-existing values for items. This 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.

Slow event info retrieval with older MySQL databases

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

API login

A large number of open user sessions can be created when using custom scripts with the `user.login` method without a following `user.logout`.

IPv6 address issue in SNMPv3 traps

Due to a net-snmp bug, IPv6 address may not be correctly displayed when using SNMPv3 in SNMP traps. For more details and a possible workaround, see ZBX-14541.

Trimmed long IPv6 IP address in failed login information

Failed login attempt message will display only the first 39 characters of a stored IP address as that's the character limit in the database field. That means that IPv6 IP addresses longer than 39 characters will be shown incompletely.

Zabbix agent checks on Windows

Non-existing DNS entries in a `Server` parameter of Zabbix agent configuration file (zabbix_agentd.conf) may increase Zabbix agent response time on Windows. This happens because Windows DNS caching daemon doesn't cache negative responses for IPv4 addresses. However, for IPv6 addresses negative responses are cached, so a possible workaround to this is disabling IPv4 on the host.

Setup wizard on SUSE with NGINX and php-fpm

Frontend setup wizard cannot save configuration file on SUSE with NGINX + php-fpm. This is caused by a setting in /usr/lib/systemd/system/php-fpm.service unit, which prevents Zabbix from writing to /etc. (introduced in PHP 7.4).

There are two workaround options available:

- Set the ProtectSystem option to 'true' instead of 'full' in the php-fpm systemd unit.
- Manually save /etc/zabbix/web/zabbix.conf.php file.

MySQL custom error codes

If Zabbix is used with MySQL installation on Azure, an unclear error message *[9002] Some errors occurred* may appear in Zabbix logs. This generic error text is sent to Zabbix server or proxy by the database. To get more information about the cause of the error, check Azure logs.

1 Compilation issues

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

Compiling Zabbix agent on HP-UX

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

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

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

Library in a non-standard location

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

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

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

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

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

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

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

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

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

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

9 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.

New templates

See the list of new templates in Zabbix 5.0.0

Changes in 5.0.1

New PostgreSQL template is available:

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

Changes in 5.0.2

New templates are available:

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

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

- *Template Module Brocade_Foundry Performance SNMPv2 → Template Module Brocade_Foundry Performance SNMP*
- *Template Module Cisco CISCO-MEMORY-POOL-MIB SNMPv2 → Template Module Cisco CISCO-MEMORY-POOL-MIB SNMP*
- *Template Module EtherLike-MIB SNMPv1*, *Template Module EtherLike-MIB SNMPv2 → Template Module EtherLike-MIB SNMP*
- *Template Module Generic SNMPv1*, *Template Module Generic SNMPv2 → Template Module Generic SNMP*
- *Template Module HOST-RESOURCES-MIB storage SNMPv1*, *Template Module HOST-RESOURCES-MIB storage SNMPv2 → Template Module HOST-RESOURCES-MIB storage SNMP*
- *Template Module Interfaces SNMPv1*, *Template Module Interfaces SNMPv2 → Template Module Interfaces SNMP*
- *Template Module Interfaces Simple SNMPv1*, *Template Module Interfaces Simple SNMPv2 → Template Module Interfaces Simple SNMP*
- *Template Module Interfaces Windows SNMPv2 → Template Module Interfaces Windows SNMP*
- *Template Module Linux memory SNMPv2 → Template Module Linux memory SNMP*
- *Template Net Alcatel Timetra TiMOS SNMPv2 → Template Net Alcatel Timetra TiMOS SNMP*
- *Template Net Arista SNMPv2 → Template Net Arista SNMP*
- *Template Net Brocade FC SNMPv2 → Template Net Brocade FC SNMP*

- *Template Net D-Link DES 7200 SNMPv2* → *Template Net D-Link DES 7200 SNMP*
- *Template Net D-Link DES_DGS Switch SNMPv2* → *Template Net D-Link DES_DGS Switch SNMP*
- *Template Net Dell Force S-Series SNMPv2* → *Template Net Dell Force S-Series SNMP*
- *Template Net Extreme EXOS SNMPv2* → *Template Net Extreme EXOS SNMP*
- *Template Net HP Comware HH3C SNMPv2* → *Template Net HP Comware HH3C SNMP*
- *Template Net HP Enterprise Switch SNMPv2* → *Template Net HP Enterprise Switch SNMP*
- *Template Net Huawei VRP SNMPv2* → *Template Net Huawei VRP SNMP*
- *Template Net Intel_Qlogic Infiniband SNMPv2* → *Template Net Intel_Qlogic Infiniband SNMP*
- *Template Net Juniper SNMPv2* → *Template Net Juniper SNMP*
- *Template Net Mellanox SNMPv2* → *Template Net Mellanox SNMP*
- *Template Net Mikrotik SNMPv2* → *Template Net Mikrotik SNMP*
- *Template Net Netgear Fastpath SNMPv2* → *Template Net Netgear Fastpath SNMP*
- *Template Net Network Generic Device SNMPv1*, *Template Net Network Generic Device SNMPv2* → *Template Net Network Generic Device SNMP*
- *Template Net QTech QSW SNMPv2* → *Template Net QTech QSW SNMP*
- *Template Net TP-LINK SNMPv2* → *Template Net TP-LINK SNMP*
- *Template Net Ubiquiti AirOS SNMPv1* → *Template Net Ubiquiti AirOS SNMP*
- *Template OS Windows SNMPv2* → *Template OS Windows SNMP*
- *Template Server Cisco UCS SNMPv2* → *Template Server Cisco UCS SNMP*
- *Template Server Dell iDRAC SNMPv2* → *Template Server Dell iDRAC SNMP*
- *Template Server HP iLO SNMPv2* → *Template Server HP iLO SNMP*
- *Template Server IBM IMM SNMPv1*, *Template Server IBM IMM SNMPv2* → *Template Server IBM IMM SNMP*
- *Template Server Supermicro Aten SNMPv2* → *Template Server Supermicro Aten SNMP*

Changes in 5.0.3

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

Template VM VMware has been updated:

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

Changes in 5.0.4

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

Changes in 5.0.5

New templates are available:

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

Changes in 5.0.6

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

New templates are available:

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

Changes in 5.0.8

New templates are available:

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

Changes in 5.0.9

A new template is available:

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

Changes in 5.0.10

New templates are available:

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

Changes in 5.0.11

New templates are available:

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

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

Changes in 5.0.12

New templates are available:

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

Changes in 5.0.13

New templates are available:

- *AAM1212-51 IES-612 SNMP* - monitoring of Zyxel AAM1212-51 / IES-612 via SNMP;
- *Cisco UCS Manager SNMP* - monitoring of Cisco UCS Manager via SNMP;
- *DELL PowerEdge R720 by HTTP* - monitoring of DELL PowerEdge R720 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- // DELL PowerEdge R720 SNMP// - monitoring of DELL PowerEdge R720 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R740 by HTTP* - monitoring of DELL PowerEdge R740 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R740 SNMP* - monitoring of DELL PowerEdge R740 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R820 by HTTP* - monitoring of DELL PowerEdge R820 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);

- *DELL PowerEdge R820 SNMP* - monitoring of DELL PowerEdge R820 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R840 by HTTP* - monitoring of DELL PowerEdge R840 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R840 SNMP* - monitoring of DELL PowerEdge R840 servers with iDRAC version 7 and later via SNMP;
- *ES3500-8PD SNMP* - monitoring of Zyxel ES3500-8PD via SNMP;
- *GS-4012F SNMP* - monitoring of Zyxel GS-4012F via SNMP;
- *HPE ProLiant BL460 SNMP* - monitoring of HPE ProLiant BL460 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant BL920 SNMP* - monitoring of HPE ProLiant BL920 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant DL360 SNMP* - monitoring of HPE ProLiant DL360 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant DL380 SNMP* - monitoring of HPE ProLiant DL380 servers with HP iLO version 4 and later via SNMP;
- *IES-500x SNMP* - monitoring of Zyxel IES-500x via SNMP;
- *IES1248-51 SNMP* - monitoring of Zyxel IES1248-51 via SNMP;
- *MES-3528 SNMP* - monitoring of Zyxel MES-3528 via SNMP;
- *MES3500-10 SNMP* - monitoring of Zyxel MES3500-10 via SNMP;
- *MES3500-24 SNMP* - monitoring of Zyxel MES3500-24 via SNMP;
- *MGS-3712 SNMP* - monitoring of Zyxel MGS-3712 via SNMP;
- *MGS-3712F SNMP* - monitoring of Zyxel MGS-3712F via SNMP;
- *MES3500-24S SNMP* - monitoring of Zyxel MES3500-24S via SNMP;
- *MGS3520-28x SNMP* - monitoring of Zyxel MGS3520-28x via SNMP;
- *NGINX Plus by HTTP* - see setup instructions for HTTP templates;
- *XGS-4728F SNMP* - monitoring of Zyxel XGS-4728F via SNMP.

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

Changes in 5.0.14

New templates are available:

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

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

Changes in 5.0.15

A new template is available:

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

Changes in 5.0.20

A new template *pfSense SNMP* is available.

Changes in 5.0.22

The template *Template App Generic Java JMX* now contains two discovery rules: - Garbage collector discovery - Memory pool discovery

Changes in 5.0.23

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

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

Changes in 5.0.26

PostgreSQL Agent 2 template updated.

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

Changes in 5.0.27

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

Changes in 5.0.31

Template DB Oracle by Zabbix agent 2 has been updated:

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

10 Zabbix 5.0.0 升级说明

这些说明用于从 Zabbix 4.4.x 升级到 Zabbix 5.0.0。所有笔记分为：

- `Critical` - 升级过程和 Zabbix 功能更改有关的最关键信息
- `Informational` - 描述 Zabbix 功能变化的所有剩余信息

可以从 Zabbix 4.4.0 之前的版本升级到 Zabbix 5.0.0。有关从以前的 Zabbix 版本进行升级的所有相关信息，请参阅升级过程部分upgrade procedure。

CRITICAL 最低要求的 PHP 版本

所需的最低 PHP 版本已从 5.4.0 升级到 ** 7.2.0 **。

此更改还会影响从某些发行版中的软件包安装 Zabbix 前端的能力。请参阅有关在RHEL/CentOS 7 上从软件包安装 Zabbix 前端的详细说明，以及受影响的Debian/Ubuntu 版本。

不再支持 IBM DB2

IBM DB2 数据库不能再用作 Zabbix 的后端数据库。

不再支持 Internet Explorer 11

Zabbix 不再支持 Microsoft Internet Explorer 11。

不再支持 mbedTLS（PolarSSL）加密库

Zabbix 不再支持 mbedTLS（PolarSSL）加密库。支持的加密库是 GnuTLS 和 OpenSSL。

所需的最低数据库版本

Zabbix 5.0.0 所需的最低数据库版本 已提高至:

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

在 MariaDB 10.2.1 及之前的版本中升级

如果数据库表是使用 MariaDB 10.2.1 及更低版本创建的，则升级 Zabbix 可能会失败，因为在那些版本中，默认行格式是紧凑的。可以通过将行格式更改为动态来解决此问题 (参见 ZBX-17690)。

启用数字（浮点）值的扩展范围

数值（浮点）数据类型现在支持约 15 位精度，范围从约-1.79E + 308 到 1.79E + 308(除了PostgreSQL 11 和早期版本). 对于新安装，默认情况下是这样。但是，在升级现有安装时，必须应用手动数据库升级补丁。

如果不应用补丁, 则前端中的System information 将显示: "Database history tables upgraded: No".

< 注意重要事项 > 该修补程序将更改历史记录和趋势表的数据列，这些数据列通常包含大量数据，因此预计需要一些时间才能完成。由于确切的估算值取决于服务器性能，数据库管理系统的配置和版本，并且无法预测，因此建议先在生产环境之外测试补丁程序。

请为您的数据库执行适当的补丁程序 (SQL file)：

- database/mysql/double.sql
- database/postgresql/double.sql
- database/oracle/double.sql

请注意，在使用软件包进行升级时，您 k 可以在 Zabbix Git 仓库中找到以下脚本:

- MySQL
- PostgreSQL
- Oracle

警告：重要！
* 仅对数据库服务器运行这些脚本。
* 在运行这些脚本之前，请确保 Zabbix 服务已停止。之后重新启动服务。

请注意，使用 TimescaleDB，compression support 仅在应用此修补程序后才能打开。

> **Note:**
> 升级数据库表后，还请在/ui/conf/zabbix.conf.php 中将 $DB['DOUBLE_IEEE754'] 值设置或更新为 true。

Docker 映像实现了非 root 权限

Zabbix Docker 映像已更新，以实现非根容器最佳实践。由于更改：

- 容器用户的所有目录都受到限制，容器所需的目录除外。例如，Zabbix 组件配置文件目录：/etc/zabbix/。
- 端口 80 和 443 已更改为 8080 和 8443，因为非特权用户限制使用所有 <1024 的端口。

已知问题：基于 Nginx 的映像在 root 下无法运行。即将修复。

INFORMATIONAL 主机接口级别的 SNMP 凭据

设置 SNMP 接口凭据已从监控项级别移至主机interface level. 有一个 **automatic** 升级过程, 可将现有 SNMP 监控项移至其相应的接口。因此，例如，如果在升级之前有：

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

升级后，将有 2 个 SNMP 接口：

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

升级之前，如果有 2 个相同的 SNMPv3 监控项具有不同的密码：

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

升级后，将有 2 个 SNMP 接口：

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

更改了 Zabbix PHP 文件目录

下载的 Zabbix 前端 PHP 文件现在位于 ui 目录中，而不是 frontends/php。使用 Zabbix 源进行安装时，这是相关的。

更改确认屏幕 URL

问题更新（确认）屏幕的 URL 参数已更改。例如，如果以前的页面参数是：

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

在新版本中，它们是：

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

在相关的开发中，当从仪表盘小部件成功更新问题时，仅重新加载该小部件，而不是整个页面。因此，另一个显示相同问题的窗口小部件的内容将保持不变，直到下一次计划的窗口小部件刷新或完成页面刷新为止。

没有数据触发对代理可用性敏感

默认情况下，现在没有数据触发器对proxy availability敏感。

全屏模式由隐藏菜单代替

全屏模式已从前端的监控部分删除。包含 “fullscreen” 的前端 URL 将不再起作用。现在，通过隐藏新的vertical menu，可以达到相同的效果（仅显示页面标题和内容）。kiosk 模式（仅页面内容，完全没有页面标题）仍然存在。

下拉第一项的选项已删除

用于配置frontend defaults 的屏幕不再具有 *Dropdown first entry* 选项，因为在前端主机组和主机选择的下拉列表已替换为 multiselect 字段。

配置参数

代理参数 EnableRemoteCommands(将来可能会被弃用并删除) 和与新的DenyKey/AllowKey 参数仍然受支持。升级现有代理时，除非您执行以下操作，否则将不允许使用远程命令：

- Set EnableRemoteCommands=1
- 删除或者注释配置 DenyKey=system.run[*]

在这种情况下，将允许无限制地使用远程命令。要创建限制，请结合使用 AllowKey 和 DenyKey 参数。

监控项 key 限制

监控项 key 的最大允许长度已从 256 个字符增加到 2048 个字符。

最低 NET-SNMP 版本

现在可以手动清除 Zabbix 服务器和代理上的 SNMP 缓存。由于添加了新的运行时控制选项，SNMP 支持现在需要 Net-SNMP 5.3.0 或更高版本。

Redis 插件更新

配置参数 `Plugins.Redis.Password` 已被删除，现在可以通过 key 的参数传输密码。有关详细信息，请参见 Redis plugin 。

支持的 Elasticsearch 版本已更改

现在支持 Elasticsearch 7.X 版。不再支持较旧版本的 Elasticsearch 。

Supported Elasticsearch versions changed

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

11 Zabbix 5.0.1 升级说明

此版本没有任何升级说明。

12 Zabbix 5.0.2 升级说明

最新数据

在最新数据 页面中:

- 通过 最近一次检查这个字段进行排序的功能已被删除
- 不支持折叠/扩展监控项的应用集 (在 Zabbix 5.0.3 版本又可以支持该功能了)

用户宏上下文中的正则表达式支持

现在，使用以下语法在用户宏上下文中支持正则表达式：

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

如果现有宏在上下文中不太可能包含 `regex:something` 字符串，则它们将不会自动转换为 `regex:"^quoted something$"`。更改必须手动完成。

有关更多信息，请参见user macros with context 。

不推荐或不支持在 zabbix agent 使用 EnableRemoteCommands 参数

现在代理参数 EnableRemoteCommands:

- 不推荐在 Zabbix agent 使用
- 不支持在 Zabbix agent2 使用

使用 AllowKey/DenyKey 参数替代。

增强的 URL 小部件安全性

现在，URL 仪表板小部件和 URL 屏幕元素将检索到的 URL 内容放入沙箱中。默认情况下，所有沙箱限制均已启用。可以在 *defines.inc.php* 文件中修改 “沙箱” 属性设置，但是出于安全原因，不建议关闭沙箱。要了解有关沙盒属性的更多信息，请参见 iframe HTML 元素说明的 sandbox 部分。

13 Zabbix 5.0.3 升级说明

IBM AIX 上的 Zabbix 代理

用于 AIX 的 Zabbix 代理已增强，可以监视物理 CPU 使用率。一个 `system.cpu.util[]` 监控项 key 具有附加的第四个参数：

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

<logical_or_physical> 键参数允许指定监控项应监控逻辑还是物理 CPU 利用率 (支持的值: `logical`, `physical`)。此新功能使用 AIX libperfstat 函数 `perfstat_cpu_util()` ，该函数可从 AIX 6.1 TL07 和 AIX 7.1 TL01 获得（据我们所知）。如果您的 AIX 系统未安装这些技术级别，请使用较旧的代理版本。您可以通过运行 `oslevel` 命令来检查安装了哪个 AIX 版本和技术级别（TL）(更多信息请查阅 IBM 文档)

14 Zabbix 5.0.4 升级说明

Agnet 2 运行时控制参数名字修改

在 Agent2 中，关于日志级别增加/减少的运行控制 参数名字已更改为与 Zabbix agent 的相关参数一致。

5.0.4 版本的命名 5.0.	版本之前的命名
log_level_increase	loglevel increase
log_level_decrease	loglevel decrease

15 Zabbix 5.0.5 升级说明

丢弃超出历史/趋势存储周期配置的数据

从现在开始，即使内部的 housekeeping 被禁用，超过配置的历史和趋势存储周期的值也将被丢弃。您可能要在升级后重新调整历史记录/趋势存储周期。

数据库连接加密设置

在 Zabbix 前端安装过程中，用于加密前端和数据库之间的连接的参数已被修改。Zabbix Web 界面的数据库连接配置的步骤中现在具有一个新复选框证书验证，标记时会出现其他选项。在特定配置中不可用的参数将被禁用。例如，如果使用 MySQL 并将// Database host 设置为 *localhost*，则无法选中 Database TLS 加密//复选框。请参阅安全连接到数据库以获取完整描述。

SourceIP 和 webhooks

SourceIP 配置参数现在用于 webhooks

API changes

See the list of API changes in Zabbix 5.0.5.

16 Zabbix 5.0.6 升级说明

在 web 监控项中的位置宏

弃用位置宏 ($1, $2, ...) 在创建 Web监控项时将不在监控项名称中使用位置宏。

如果在现有的 Web 监控项中使用了位置宏，则一经更新 Web 场景，监控项名称就在数据库更新为不使用位置宏。

Web monitoring items

Removed support of {HOST.*} macro in name property of web scenario and web scenario step.

17 Zabbix 5.0.7 升级说明

此次要版本没有任何升级说明。

Zabbix agent 2 plugins

For Ceph, Docker, Memcached, MySQL, Oracle, and Redis plugins, a Uri can no longer be specified as a 1st level plugin parameter. For PostgreSQL plugin a Uri and Database name can no longer be specified as a 1st level plugin parameters. Now these parameters can only be provided at a named session level. All existing parameters in a format Plugins.<PluginName>.Uri and the Plugins.Postgres.Database parameter should be removed from the configuration file, otherwise, the agent will fail to start.

For Uri parameters, specifying a scheme is no longer mandatory.

For the default set of parameters, you can create a named session 'Default' and specify these parameters in the session. See an example below.

Before Zabbix 5.0.7	Since Zabbix 5.0.7
Plugins.Ceph.Uri="https://localhost"	Plugins.Ceph.Sessions.Default.Uri=localhost

18 Zabbix 5.0.8 升级说明

主机/模板克隆

要克隆的对象（项、触发器、图等）的长列表已从主机和模板完全克隆表单中删除。

通知报告

如果在通知媒介类型下拉列表中选择了'All'→ 每一种媒体类型发送的通知数将不再显示在报告的总数之后的括号中。

19 Upgrade notes for 5.0.9

This minor version has no upgrade notes.

20 Upgrade notes for 5.0.10

This minor version has no upgrade notes.

21 Upgrade notes for 5.0.11

VMware event collector

The behavior of VMware event collector has been changed to fix a memory overload issue.

22 Upgrade notes for 5.0.12

This minor version has no upgrade notes.

23 Upgrade notes for 5.0.13

This minor version has no upgrade notes.

24 Upgrade notes for 5.0.14

This minor version has no upgrade notes.

25 Upgrade notes for 5.0.15

This minor version has no upgrade notes.

26 Upgrade notes for 5.0.16

Items

Zabbix agent 2 items **proc.num**, **proc.cpu.utilization**, **proc.mem** have been updated to use the latest functions introduced in Go 1.16 and thus provide better performance. Go version 1.16 or newer is now required for compiling Zabbix agent 2 to ensure correct work of these items and avoid an issue observed when compiling with older Go versions.

27 Upgrade notes for 5.0.17

This minor version has no upgrade notes.

28 Upgrade notes for 5.0.18

This minor version has no upgrade notes.

29 Upgrade notes for 5.0.19

This minor version has no upgrade notes.

30 Upgrade notes for 5.0.20

Item changes

Native support for the items **system.hw.chassis**, **system.hw.devices**, **vfs.dir.count** and **vfs.dir.size** has been added to Zabbix agent 2.

31 Upgrade notes for 5.0.21

Item changes

Native support for the items **net.dns** and **net.dns.record** has been added to Zabbix agent 2. On Zabbix agent 2 for Windows, these items now allow custom DNS IP addresses in the `ip` parameter and no longer ignore `timeout` and `count` parameters.

32 Upgrade notes for 5.0.22

This minor version has no upgrade notes.

33 Upgrade notes for 5.0.23

This minor version has no upgrade notes.

34 Upgrade notes for 5.0.24

This minor version has no upgrade notes.

35 Upgrade notes for 5.0.25

This minor version has no upgrade notes.

36 Upgrade notes for 5.0.26

This minor version has no upgrade notes.

37 Upgrade notes for 5.0.27

This minor version has no upgrade notes.

38 Upgrade notes for 5.0.28

This minor version has no upgrade notes.

39 Upgrade notes for 5.0.29

This minor version has no upgrade notes.

40 Upgrade notes for 5.0.30

This minor version has no upgrade notes.

41 Upgrade notes for 5.0.31

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

Optimized API queries API database queries, created when searching through names in the *hosts* and *items* tables, have been optimized and will now be processed more efficiently. As a result of this change, deterministic triggers need to be created during an upgrade.

On MySQL and MariaDB, this requires `GLOBAL log_bin_trust_function_creators = 1` to be set if binary logging is enabled and there is no superuser privileges and `log_bin_trust_function_creators = 1` is not set in MySQL configuration file. To set the variable using MySQL console, run:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Once the upgrade has been successfully completed, `log_bin_trust_function_creators` can be disabled:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
```

Triggers are also created for PostgreSQL and Oracle database.

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

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

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

42 Upgrade notes for 5.0.32

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

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

43 Upgrade notes for 5.0.33

This minor version does not have any upgrade notes.

44 Upgrade notes for 5.0.34

This minor version does not have any upgrade notes.

45 Upgrade notes for 5.0.35

This minor version does not have any upgrade notes.

46 Upgrade notes for 5.0.36

This minor version does not have any upgrade notes.

47 Upgrade notes for 5.0.37

This minor version does not have any upgrade notes.

48 Upgrade notes for 5.0.38

This minor version does not have any upgrade notes.

49 Upgrade notes for 5.0.39

This minor version does not have any upgrade notes.

50 Upgrade notes for 5.0.40

This minor version does not have any upgrade notes.

51 Upgrade notes for 5.0.41

This minor version has no upgrade notes.

52 Upgrade notes for 5.0.42

Zabbix agent 2 support on Windows

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

5. 快速入门

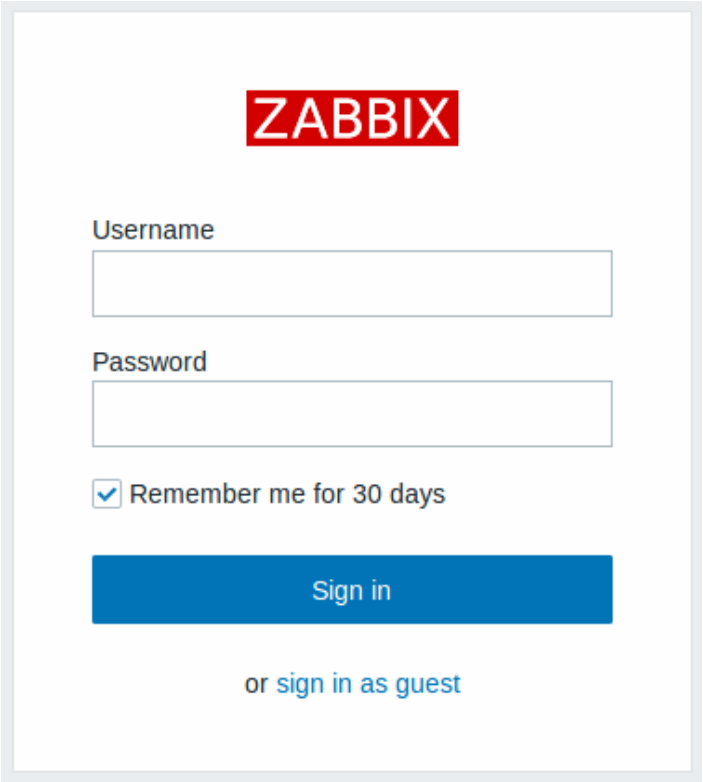
请使用侧边栏访问快速入门部分的内容。

1 登陆和配置用户

简介

本章你会学习到如何登陆 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。

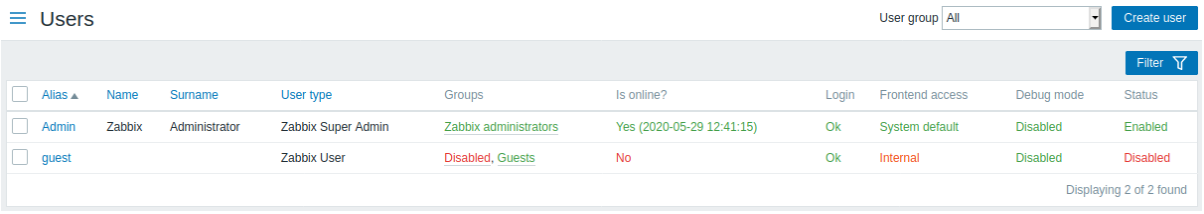
暴力破解攻击的保护机制

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 接口将暂停 30 秒。

在下次成功登陆后，将会在界面上显示登录尝试失败的 IP 地址。

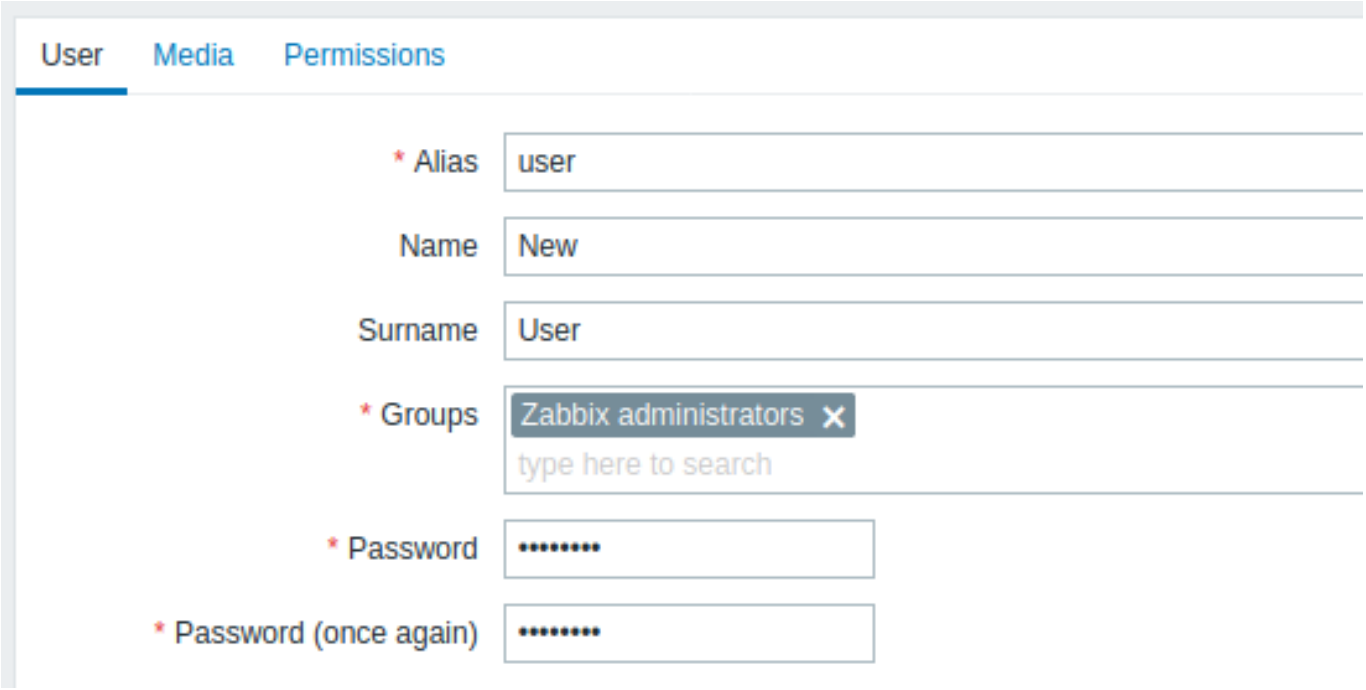
增加用户

可以在管理（*Administration*）→ 用户（*Users*）下查看用户信息。



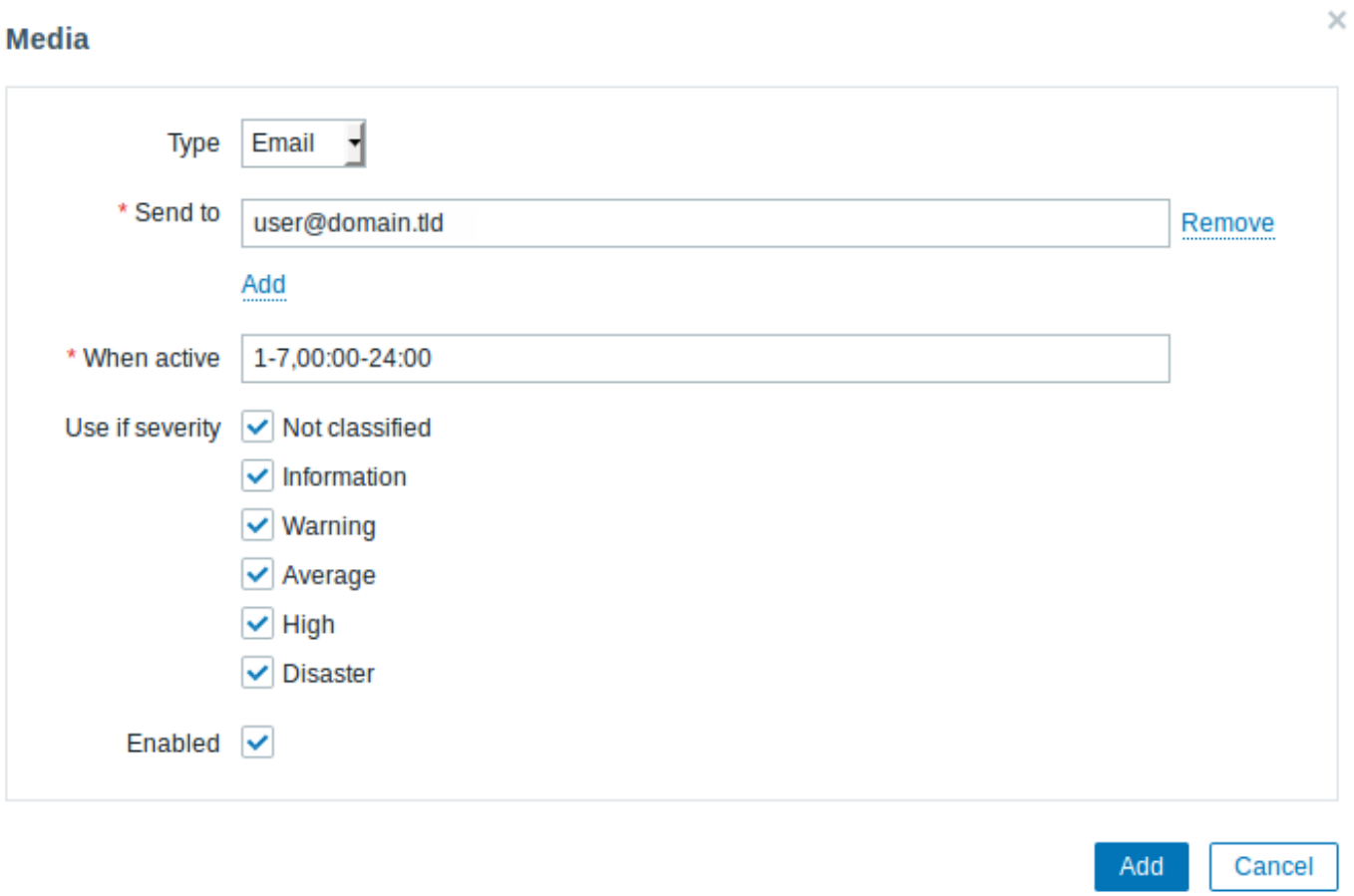
点击 创建用户（*Create user*）以增加用户。

在添加用户的表单中，请确保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。

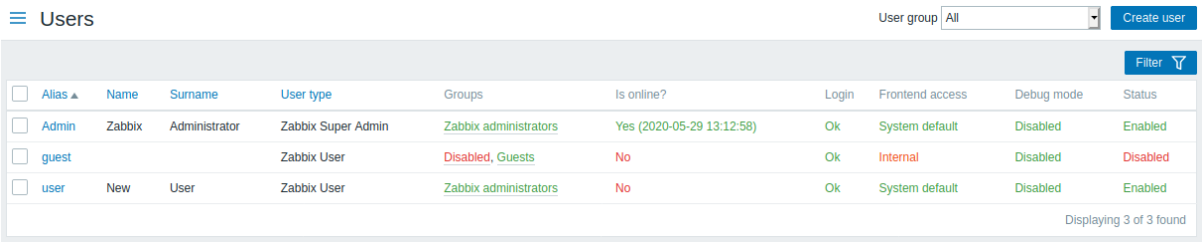
默认情况下，没有为新增的用户定义媒介（media，即通知发送方式）。如需要创建，可以到’ 媒介（Media）’ 标签下，然后点击增加（*Add*）。



在这个对话框中，为用户输入一个 Email 地址。

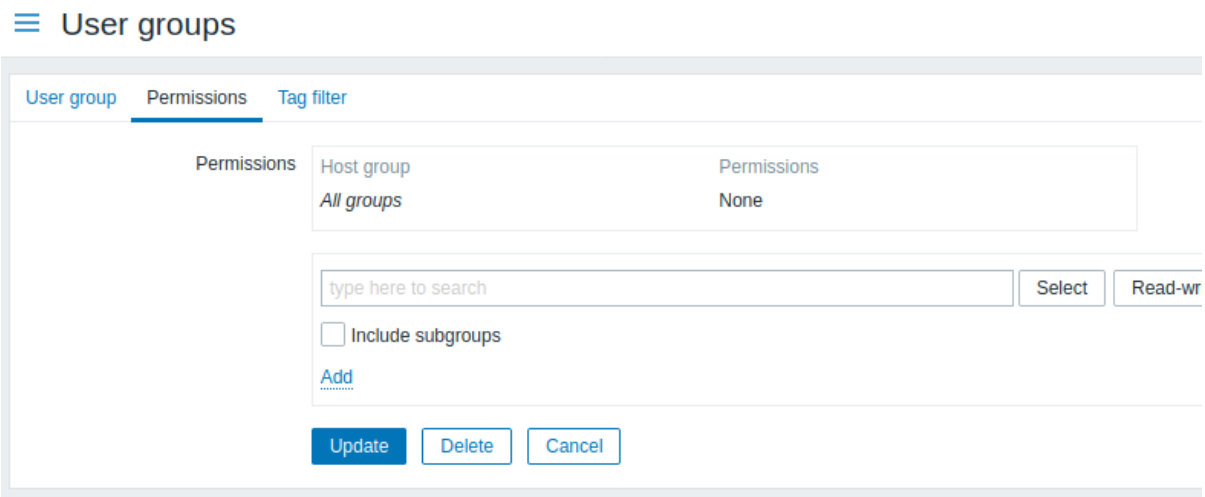
你可以为媒介指定一个时间活动周期，(访问时间周期说明页面，查看该字段格式的描述)。默认情况下，媒介一直是活动的。你也可以通过自定义触发器严重等级来激活媒介，默认所有的等级都保持开启。

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。

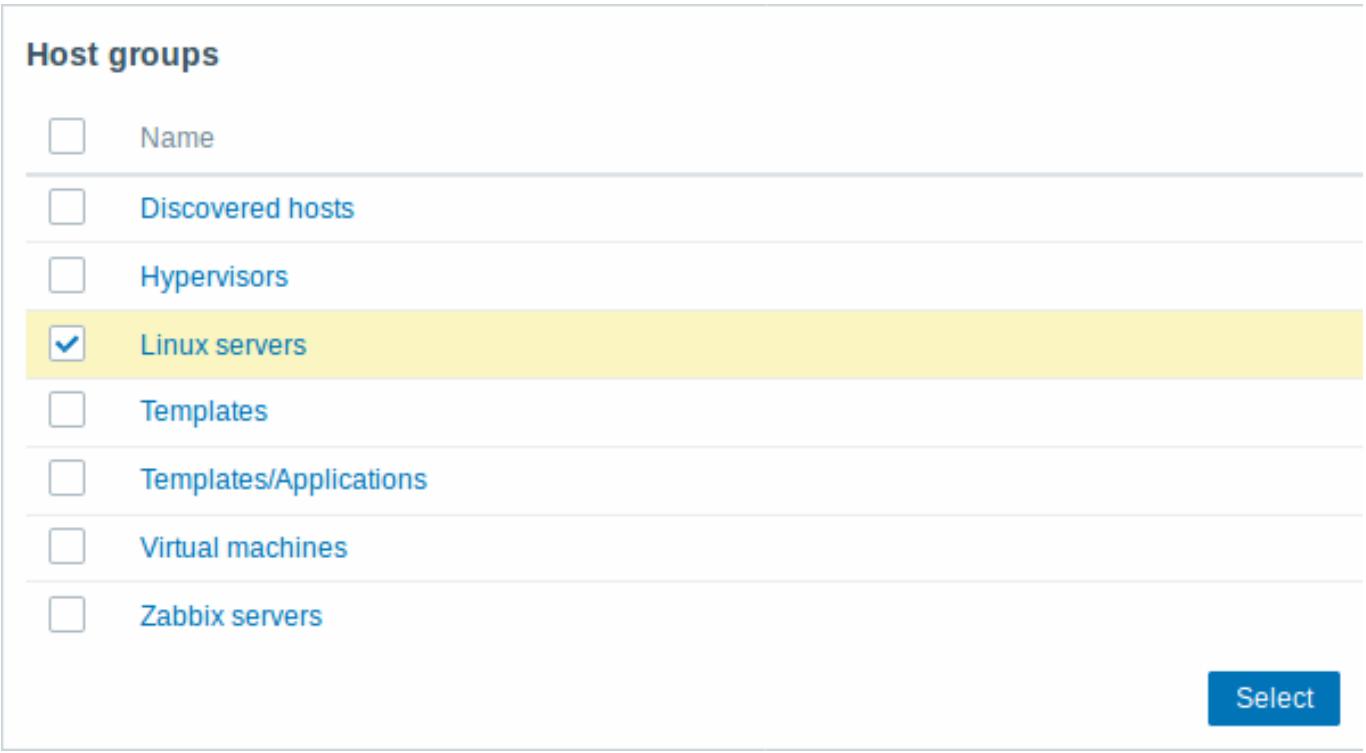


添加权限

默认情况下，新用户没有访问主机的权限。若要授予用户权限，请单击 “组” 列中的用户组 (在本例中为”administrators” 组)。在 “组属性” 表单中，转到 “权限” 选项卡。



此用户是要有只读访问 Linux Server 组的权限，所以点击用户组选择字段旁边的 Select。



在此弹出框中，选中在 “Linux servers” 旁边的复选框，然后单击 “选择”。Linux server 就会显示在选择清单中。单击 “Read” 按钮设置权限级别，然后添加到权限列表中。在 “用户组属性” 表单中，单击 “更新。

重要提醒：在 Zabbix 中，主机的访问权限被分配给用户组, 而不是单独的用户。

权限设置完成了！您可以尝试使用新用户的凭据登录。

2 新建主机

简介

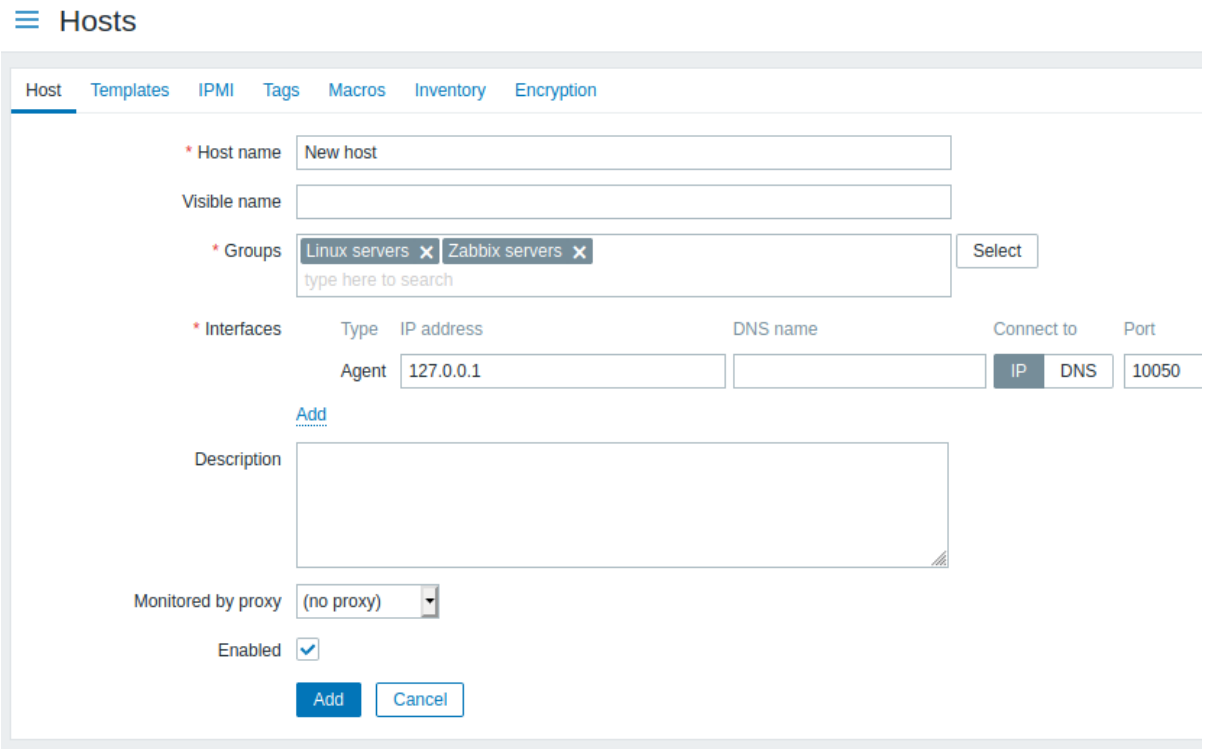
通过本节，你将会学习到如何建立一个新的主机。

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。它可以是一台物理服务器，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。



所有必填字段均以红色星标标示。

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；或者输入一个不存在的组名，zabbix 会创建这个组。

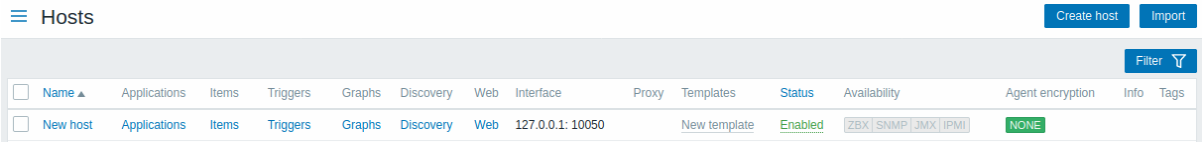
Note:
所有访问权限都分配到主机组，而不是单独的主机。这也是主机需要属于至少一个组的原因。

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。

其他选项将会使用默认值。

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。



可用性列包含每个接口的主机可用性指标。我们已经定义了 Zabbix 代理接口，因此我们可以使用代理可用性图标 (上面有 'ZBX') 来判断主机可用性：

- - 表示主机状态尚未建立，尚未发生监控指标检查
- ZBX 表示主机可用，监控指标检查已成功
- ZBX 表示主机不可用，监控指标检查失败（将鼠标光标移动到图标上以查看错误消息）。可能是由于接口凭证不正确造成了通信问题。检查 zabbix server 是否正在运行，并稍后尝试刷新页面。

3 新建监控项

简介

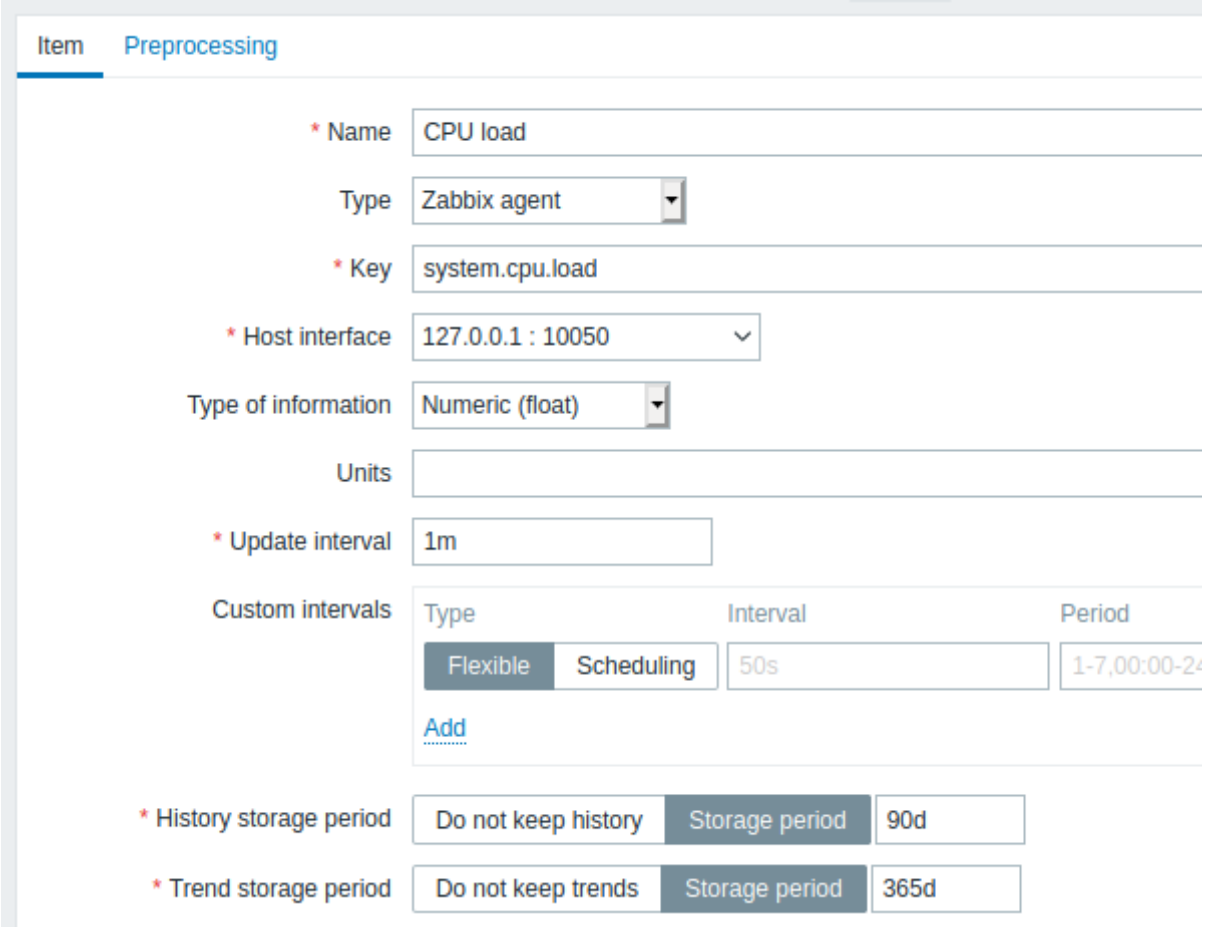
本节你会学习如何新建一个监控项（Item）。

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，点击监控项这个链接，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。

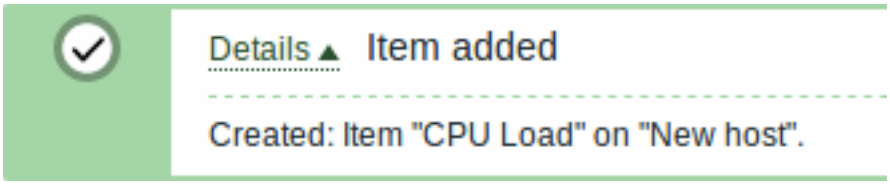
信息类型（***Type of information***）

- 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。这是一种可以减轻数据库保留许多历史值的很好的做法。

我们暂时保持其他选项的默认值。

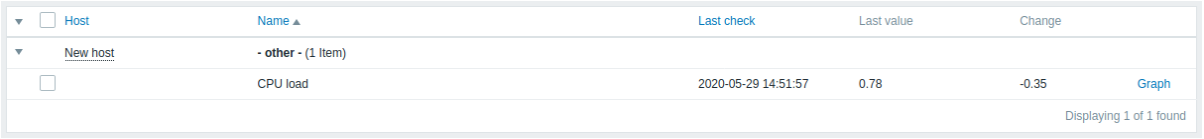
当完成后，点击添加（*Add*）。新的监控项将出现在监控项列表中。点击列表中的详细（*Details*）以查看具体细节。



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。

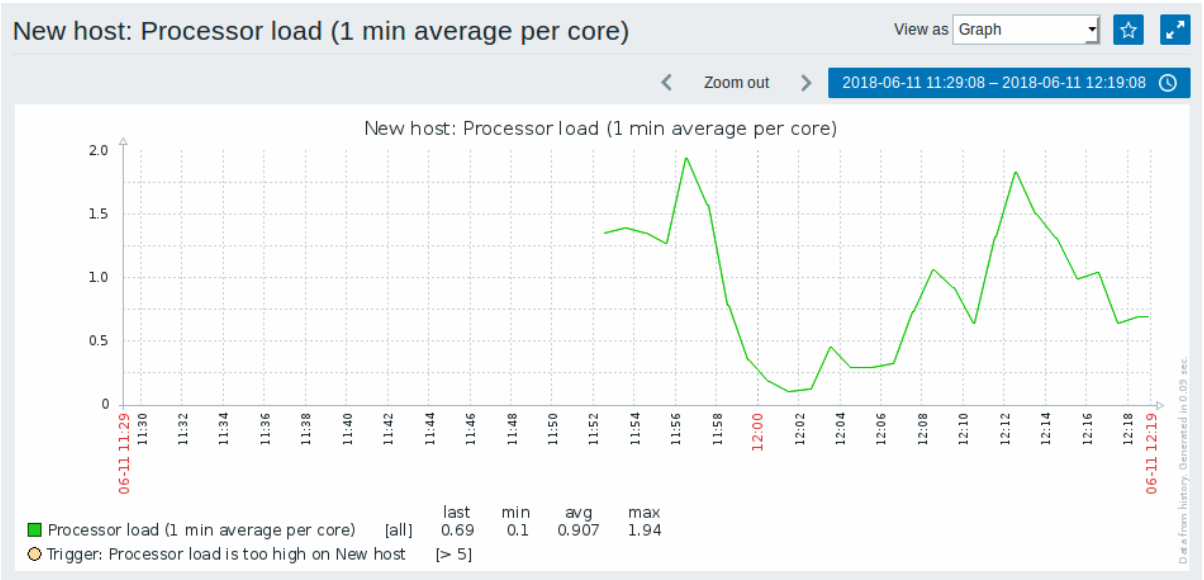
如果你在没有看到类似截图中的监控项信息，请确认：

- 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

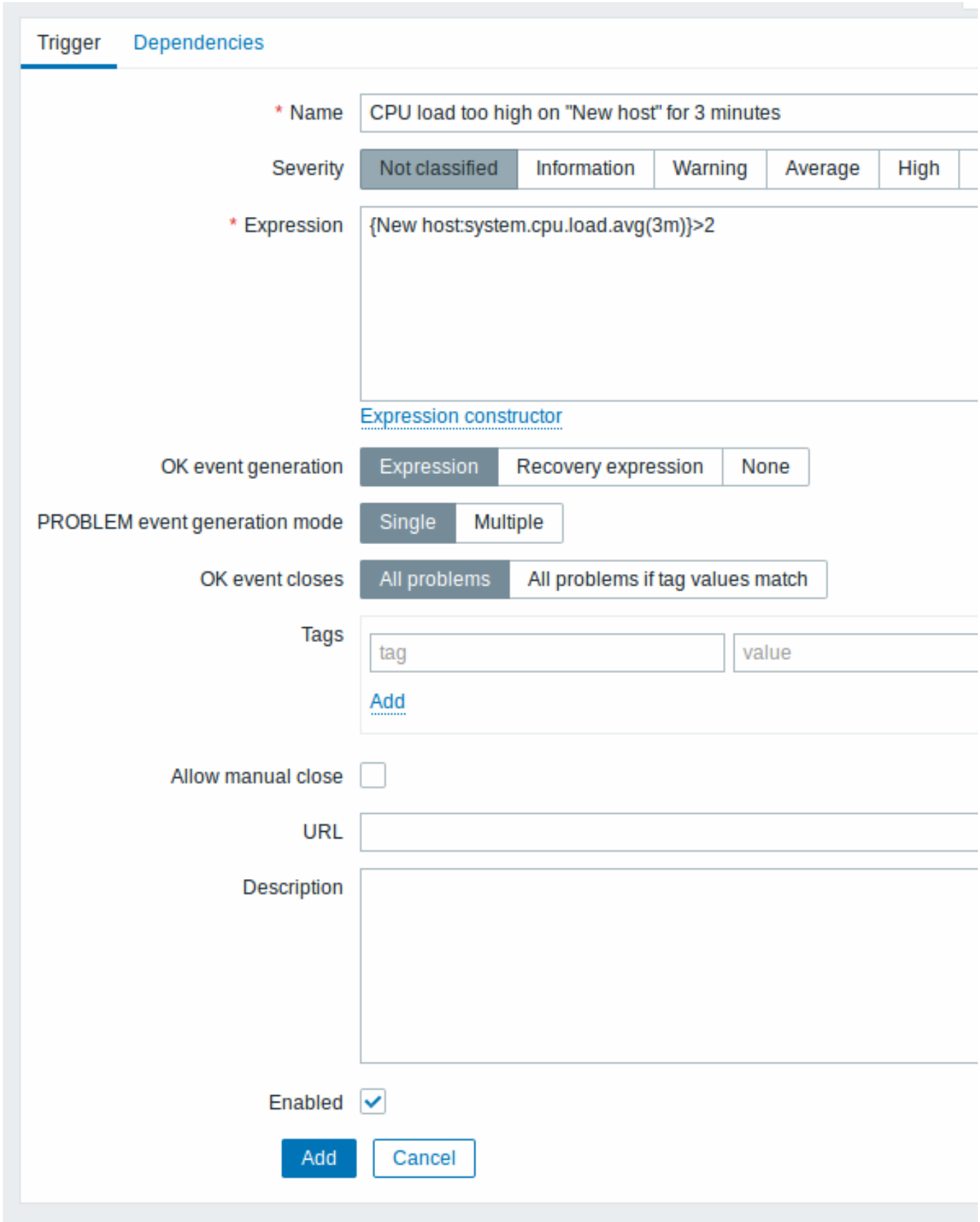
本节你会学习如何配置一个触发器（trigger）。

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。



对于我们的这个触发器，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显示在列表和其他地方。

表达式（*Expression*）

- 输入：{New host:system.cpu.load.avg(3m)}>2

这个是触发器的表达式。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

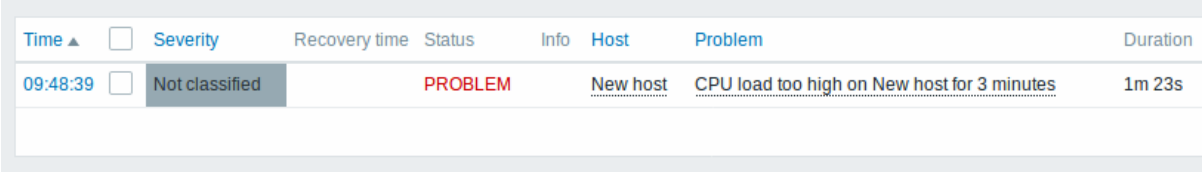
个特定的表达式大致是说如果 3 分钟内，CPU 负载的平均值超过 2，那么就触发了问题的阈值。你可以查看更多的触发器表达式语法信息。

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。



状态列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）。

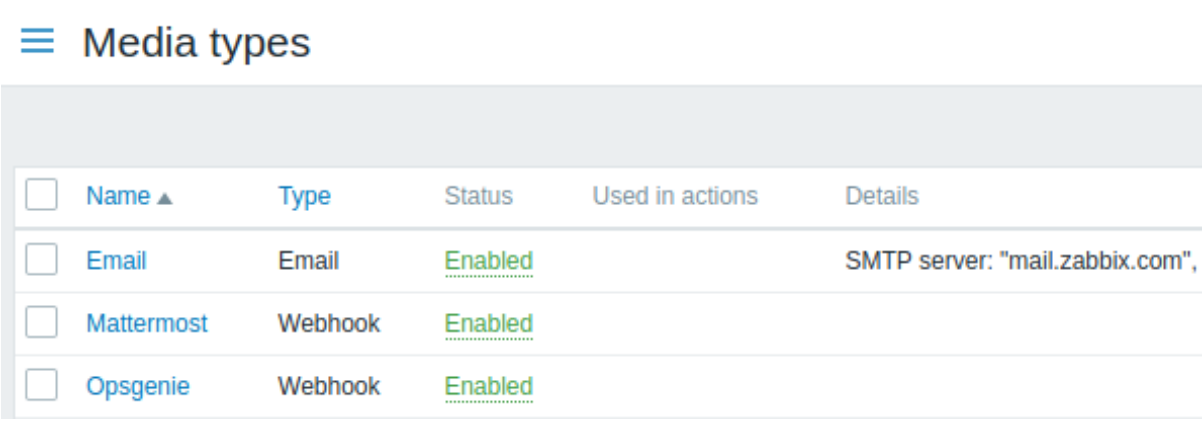
当监控项收集了数据后，触发器会根据异常状态触发报警。根据一些报警机制，它也会通知我们一些重要的事件，而不需要我们直接在 Zabbix 前端进行查看。

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。

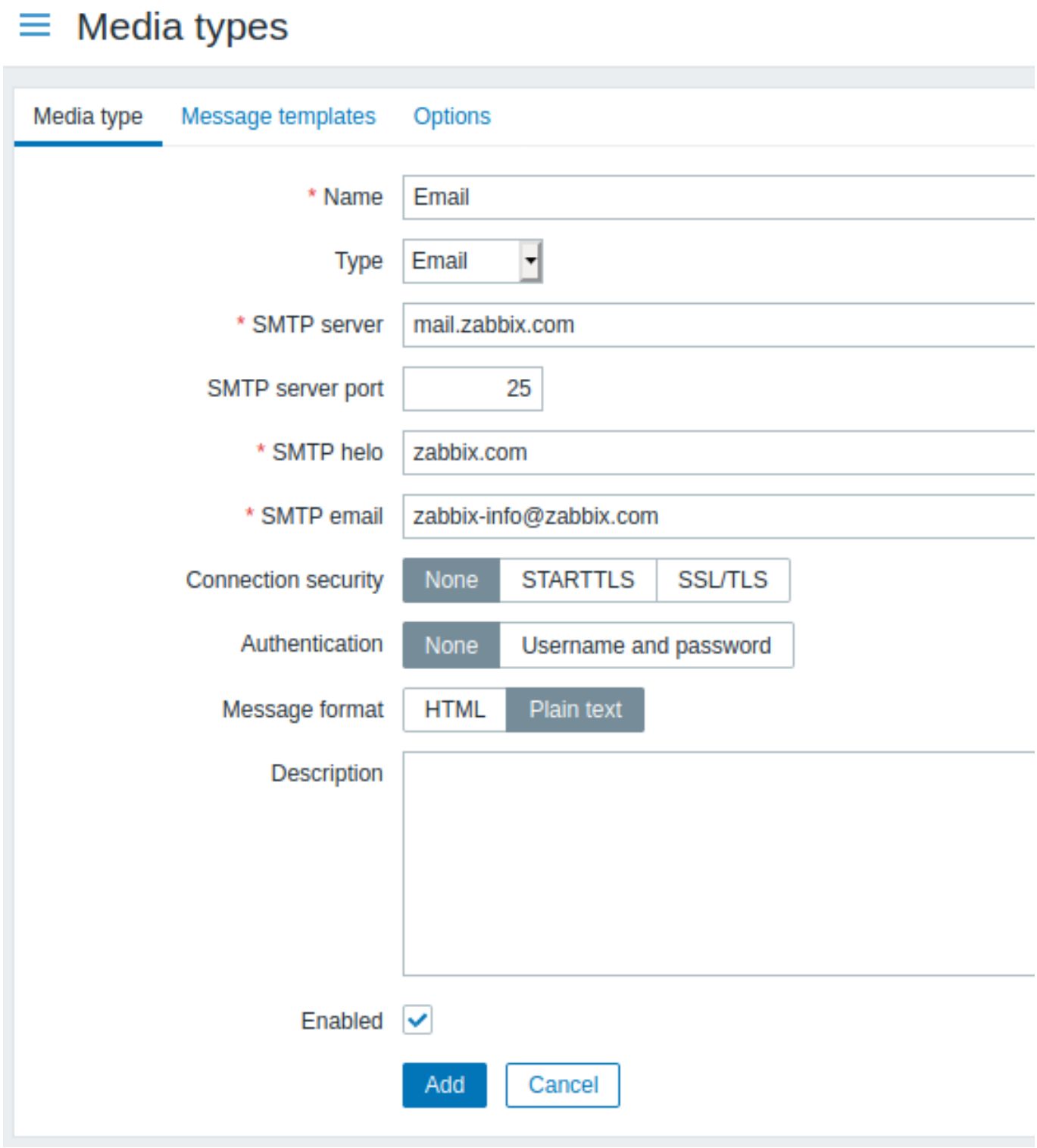
E-mail 设置

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。

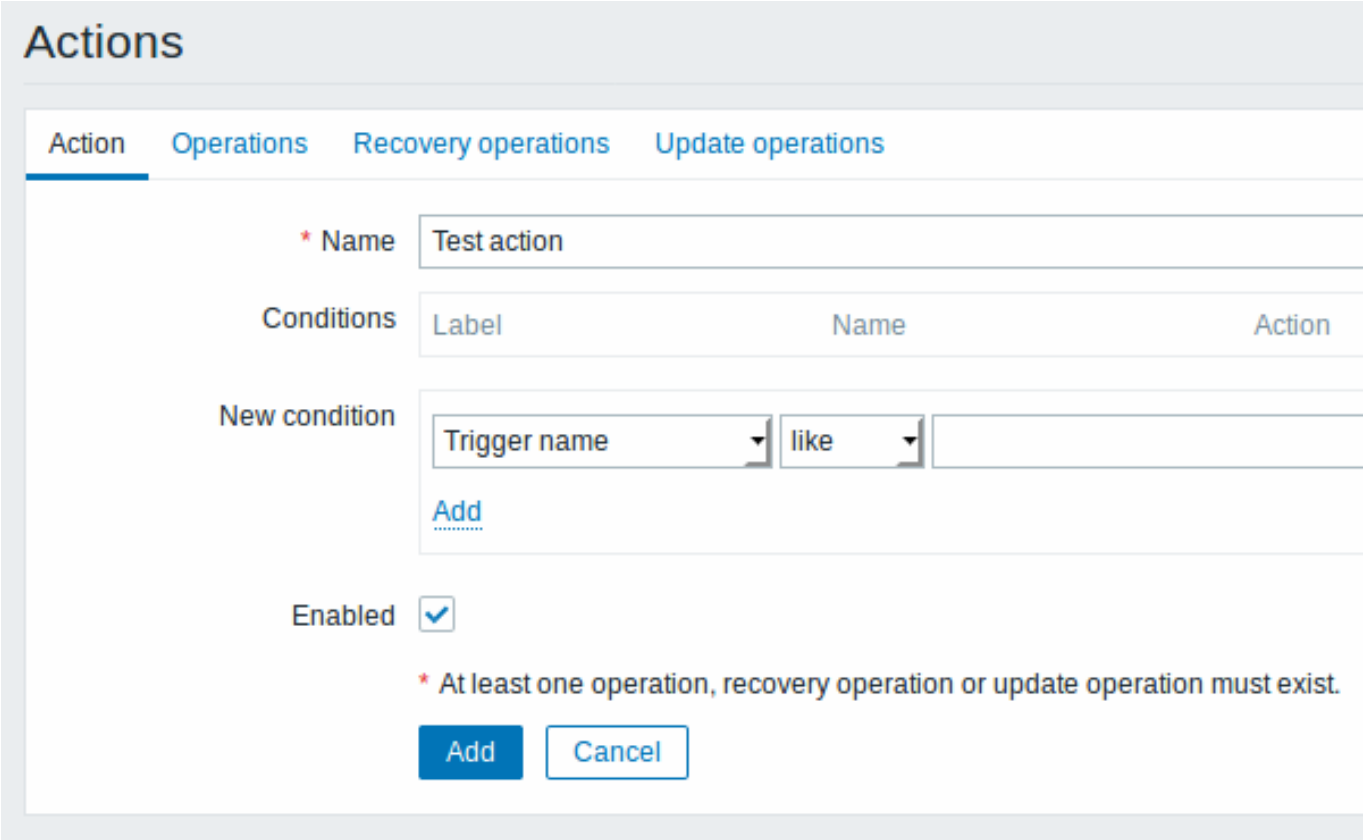
Note:
'SMTP email' 将作为 Zabbix 通知的' 发件人（From）' 地址。

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*）。

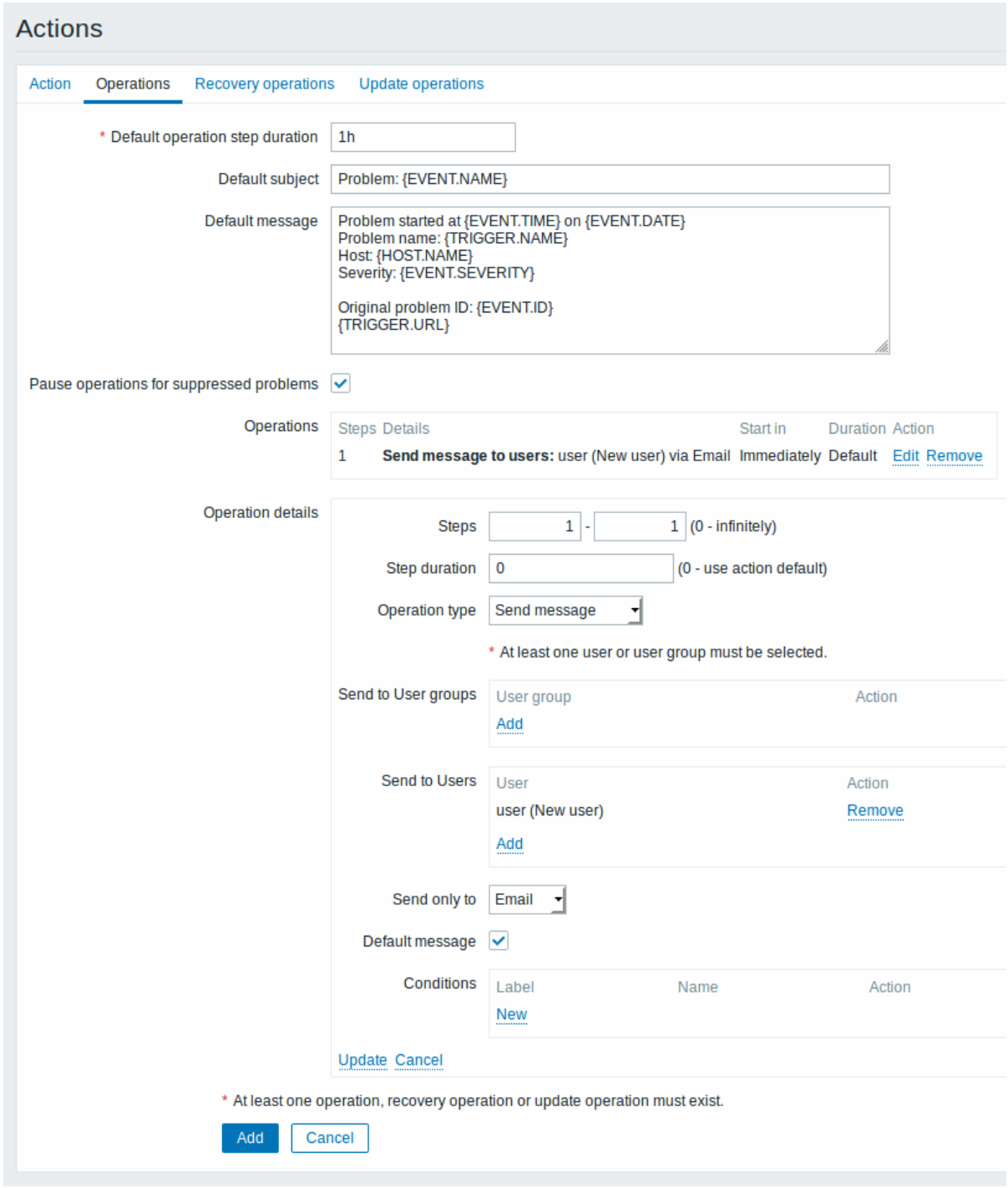


所有必填字段均以红色星标标示。

在这个表单中，输入这个动作的名称。

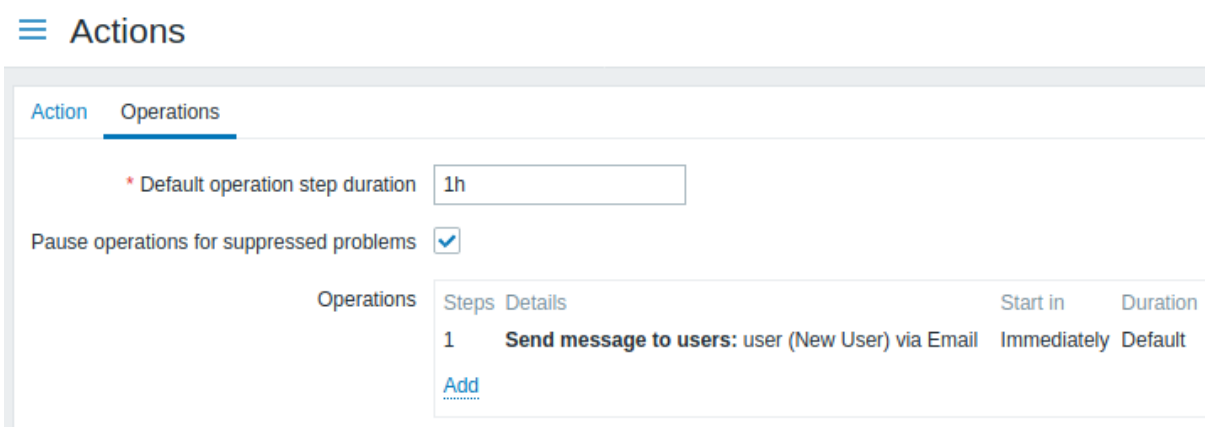
在大多数简单的例子中，如果我们不添加更多的指定条件，这个动作会在任意一个触发器从'Ok' 变为'Problem' 时被触发。

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*），然后操作就完成添加了。



配置一个简单的动作就这些步骤了，最后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，我们需要你增加主机的负载，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。

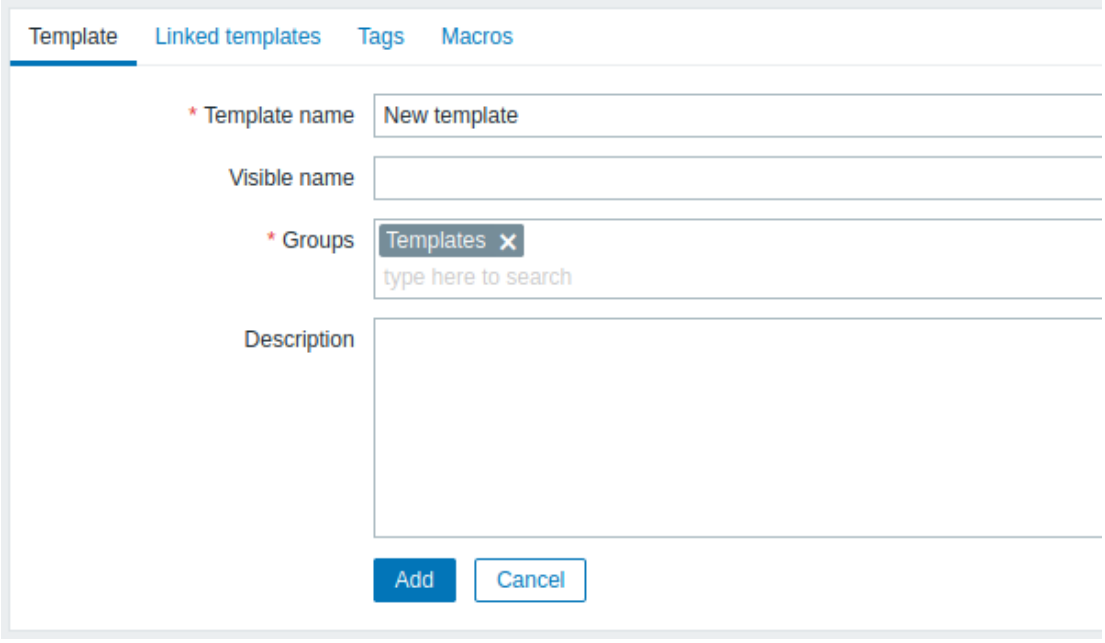
虽然这些步骤提供了很大的灵活性，但仍然需要很多步骤才能完成。如果我们需要配置上千台主机，一些自动化操作会带来更多便利性。

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：

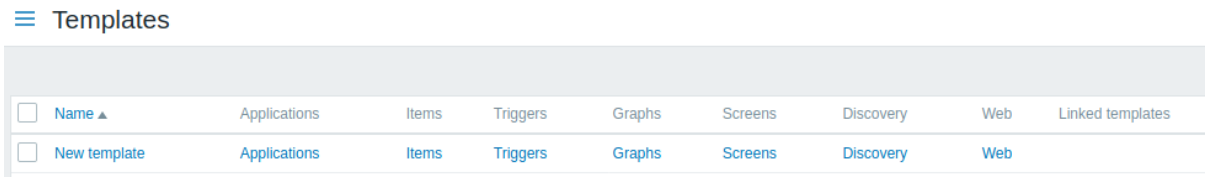
模版名称（***Template name***）

- 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。



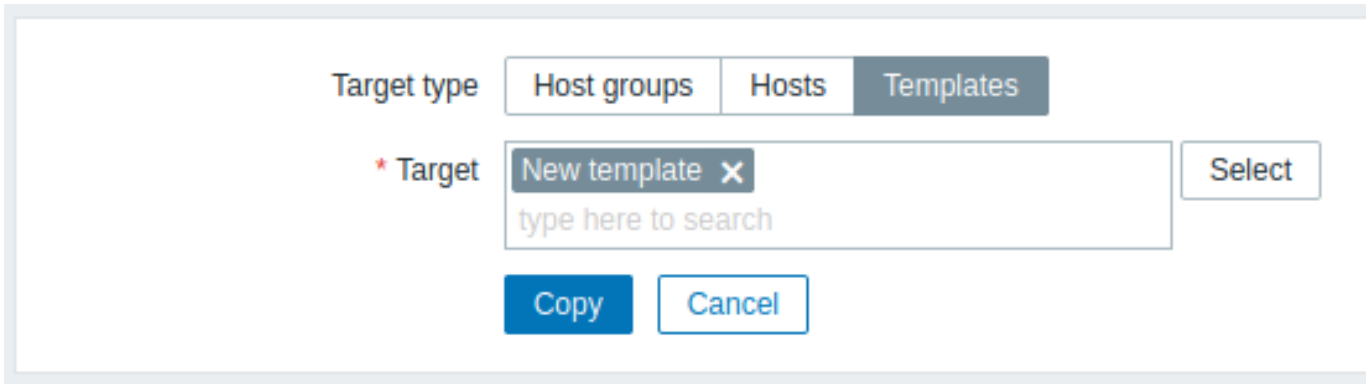
你可以在这看到模版信息。但这个模版中没有任何信息——没有监控项、触发器或者其他对象。

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*）

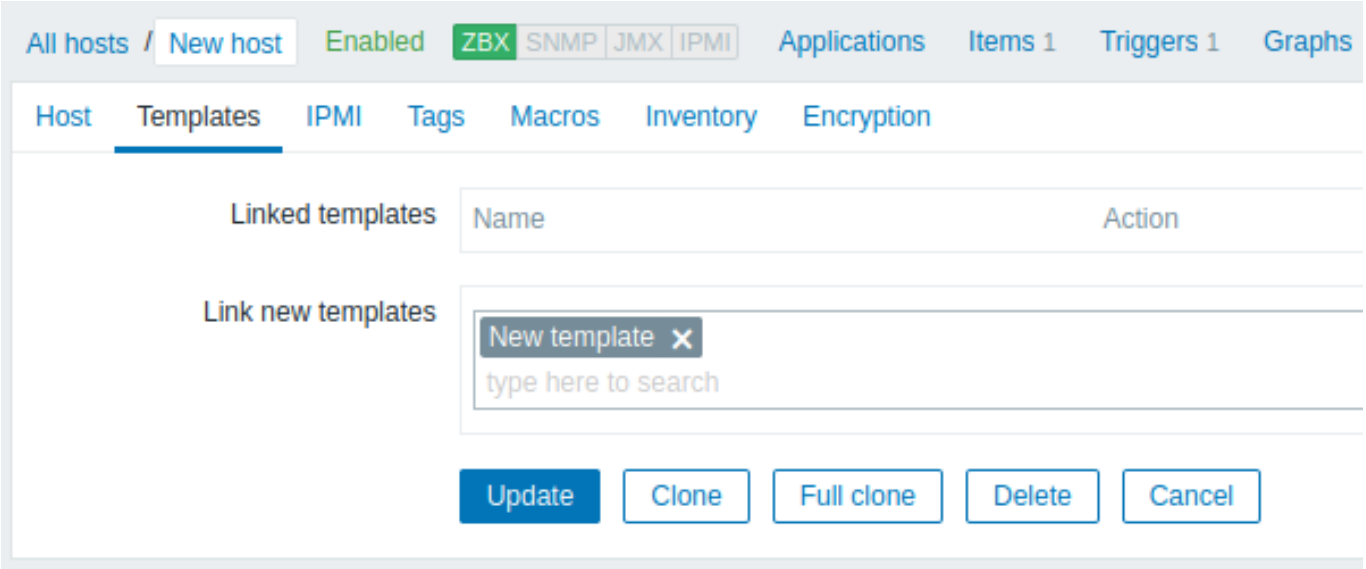
你现在可以前往配置（*Configuration*）→ 模版（*Templates*），' 新模版（New template）' 中会有一个新的监控项。

我们目前只创建了一个监控项，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。

在链接新模版（*Link new templates*）输入我们刚刚新创建的模板名字，我们所创建的模板的名称应该显示在下拉列表中，请向下滚动以进行选择。请查看模版是否出现在链接新模版（*Link new templates*）文本框。



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

概述　除了手动安装或者重新使用现有的服务器来运行 Zabbix 外，用户可通过下载Zabbix 应用或者包含 Zabbix 应用的光盘镜像。

Zabbix 设备和安装 CD 版本基于以下操作系统：

Zabbix 应用版本	操作系
5.0.0	CentOS 8 (x86_64)

Zabbix 设备安装 CD 可用于即时部署 Zabbix 服务器（MySQL）。

系统要求：

- 内存：1.5 GB
- 磁盘空间：应至少为虚拟机分配 8 GB。

|<| |<| |-|

Zabbix 设备包含一个 Zabbix 服务器（已配置并在 MySQL 上运行）和一个前端。

Zabbix 虚拟应用具有以下格式：

- VMWare (.vmx)
- Open virtualization format (.ovf)
- Microsoft Hyper-V 2012 (.vhdx)
- Microsoft Hyper-V 2008 (.vhd)
- KVM, Parallels, QEMU, USB stick, VirtualBox, Xen (.raw)
- KVM, QEMU (.qcow2)

首先，启动应用并将浏览器指向设备通过 DHCP 接收到的 IP。

Attention:
主机必须启用 DHCP。

要从虚拟机内部获取 IP 地址，请运行：

```
ip addr show
```

要访问 Zabbix 前端，请访问 **http://<host_ip>** （要在 VM 网络设置中启用从主机的浏览器桥接模式访问）。

<note 提示：> 如果应用在 Hyper-V 中启动失败，你可能需要按 Ctrl+Alt+F2 键切换 ttp 会话窗口。:::

1 对 **CENTOS 8** 配置的更改　该设备基于 CentOS8。对 CentOS 基本配置进行了一些更改。

1.1 存储库

Zabbix 官方yum 软件仓库已经添加到 */etc/yum.repos.d* 中：

```
[zabbix]
name=Zabbix Official Repository - $basearch
baseurl=http://repo.zabbix.com/zabbix/5.0/rhel/8/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
```

1.2 防火墙

设备使用具有预定义规则的 iptables 防火墙：

- 已开启 SSH 端口 (22 TCP)；
- 已开启 Zabbix agent (10050 TCP) 和 Zabbix trapper (10051 TCP) 端口；
- 已开启 HTTP (80 TCP) 和 HTTPS (443 TCP) 端口；
- 已开启 SNMP trap 端口 (162 UDP)；
- 已开启 outgoing connections to NTP 端口 (53 UDP)；
- ICMP 数据包限制为每秒钟 5 个数据包；
- 所有其他传入连接均被删除。

1.3 使用静态 IP 地址

默认情况下，应用使用 DHCP 获取 IP 地址。如果要指定静态 IP 地址，需要：

- 以 root 用户身份登录；
- 打开网卡配置文件 */etc/sysconfig/network-scripts/ifcfg-eth0* ；
- 将 *BOOTPROTO=dhcp* 替换为 *BOOTPROTO=none* ；
- 添加以下行：
  - *IPADDR=<IP address of the appliance>*
  - *PREFIX=<CIDR prefix>*
  - *GATEWAY=<gateway IP address>*
  - *DNS1=<DNS server IP address>*
- 运行 **systemctl restart network** 命令重启网卡。

如果需要，请查阅 Red Hat 官方文档。

1.4 更改时区

默认情况下，设备使用 UTC 作为系统时钟。要更改时区，需要将相应的配置文件从 */usr/share/zoneinfo* 复制到 */etc/localtime*，例如：

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

2 ZABBIX 配置　Zabbix 应用设置具有以下密码和配置更改：

2.1 登录凭证 (login:password)

系统：

- root:zabbix

Zabbix 前端：

- Admin:zabbix

数据库：

- root:<random>
- zabbix:<random>

<note 注意：> 数据库密码是在安装过程中随机生成的。根密码存储在/root/.my.cnf 文件中。不需要在 “ root” 帐户下输入密码。
根密码存储在 /root/.my.cnf 文件中。不需要在”root” 账户下输入密码。:::

要更改数据库用户密码，必须在以下位置进行更改：

- MySQL;
- /etc/zabbix/zabbix_server.conf;
- /etc/zabbix/web/zabbix.conf.php.

<note 注意：> 用户 `zabbix_srv` 和 `zabbix_web` 分别为服务器和前端定义。:::

2.2 文件位置

- 配置文件位于 **/etc/zabbix**。
- Zabbix server、proxy 和 agent 日志文件在 **/var/log/zabbix**。
- Zabbix 前端相关配置在 **/usr/share/zabbix**。
- 用户 **zabbix** 的主目录是 **/var/lib/zabbix**。

2.3 对 ZABBIX 配置的更改

- 前端时区设置为 Europe/Riga （可以在 **/etc/php-fpm.d/zabbix.conf** 配置文件中更改时区）；

3 前端访问　默认情况下，允许从任何地方访问前端。

可以通过 *http://<host>* 访问前端。

可以在 **/etc/nginx/conf.d/zabbix.conf** 文件中对访问路径进行自定义。修改此文件后，必须重新启动 Nginx。为此，请以 **root** 用户身份使用 SSH 登录并执行：

```
systemctl restart nginx
```

4 防火墙　默认情况下，仅上文配置更改中列出的端口是打开的。要打开其他端口，请修改配置文件*”/etc/sysconfig/iptables”* 并重新加载防火墙规则：

```
systemctl reload iptables
```

5 升级　Zabbix 应用软件包可能已升级。为此，请运行：

```
dnf update zabbix*
```

6 系统服务　提供系统服务：

```
systemctl list-units zabbix*
```

7 格式特定的注释　7.1 VMWARE

vmdk 格式的映像可直接在 VMware Player，Server 和 Workstation 产品中使用。如果想要在 ESX、ESXi 和 vSphere 中使用，必须使用VMware converter进行转换。

7.2 HDD / FLASH 闪存镜像 (raw)

```
dd if=./zabbix_appliance_5.0.0.raw of=/dev/sdc bs=4k conv=fdatasync
```

用你的 Flash/HDD 磁盘设备替换 */dev/sdc* 。

7. 配置

请使用左侧导航栏来访问 “配置” 这一章节的内容。

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）。

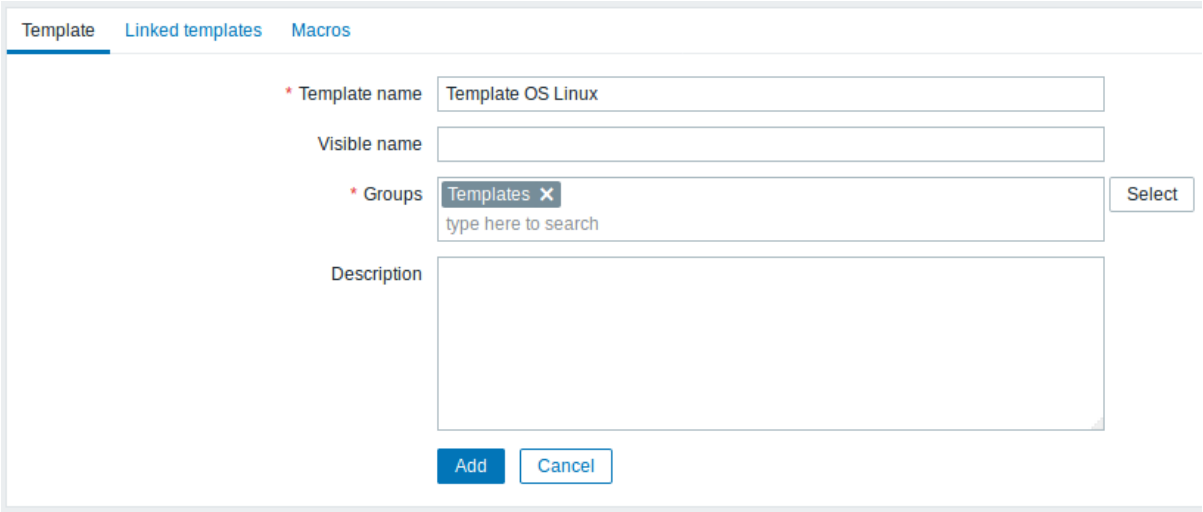
创建模板

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。

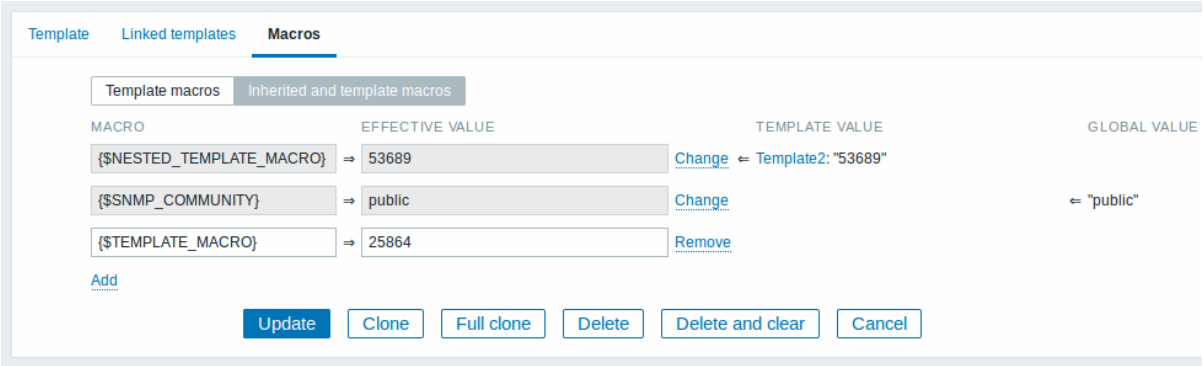
链接的模板选项卡允许您将一个或多个“嵌套”模板链接到此模板。所有实体（项目，触发器，图表等）将从链接的模板继承。

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

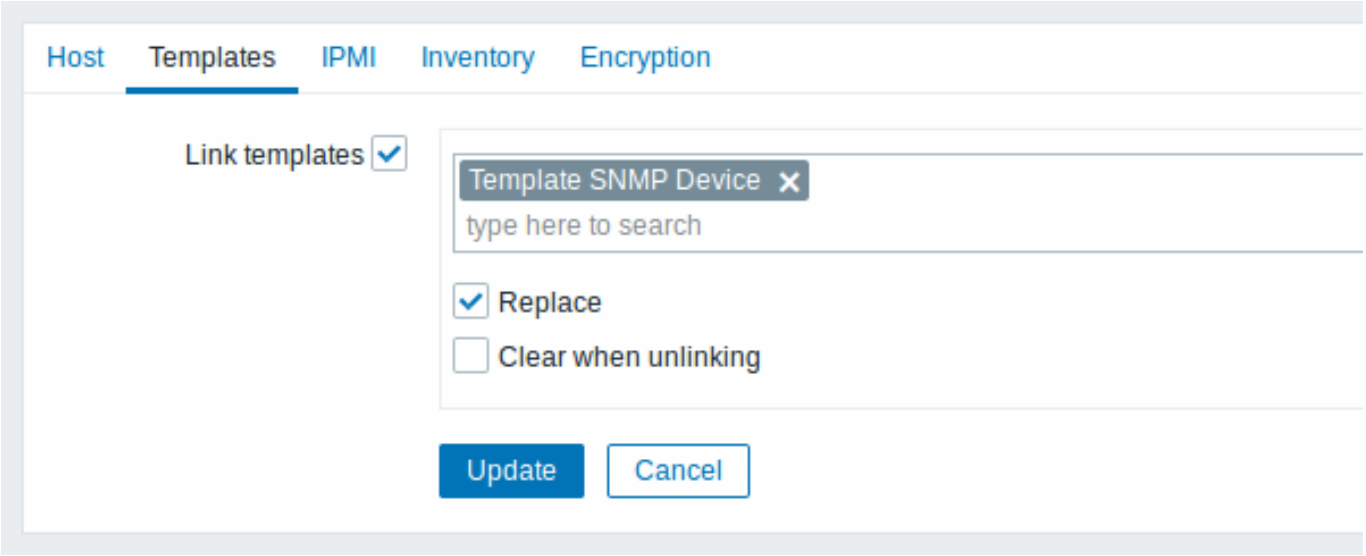
将模板链接到多个主机

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别单列出来用于对应各种服务，应用程序等都是有意义的，所以您可能会制作出相当多的模板，所有这些模板都可能需要链接到复数个主机。为了让过程简化，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
- 导航至链接的模板选项卡
- 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

完成上述步骤，模板本身拥有的实体，以及所有链接的模板的实体（如监控项，触发器，自定义图表等）将出现在模板配置中但不包含连接的模板的聚合图形，聚合图形模板仅由主机继承。

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。

4 批量更新

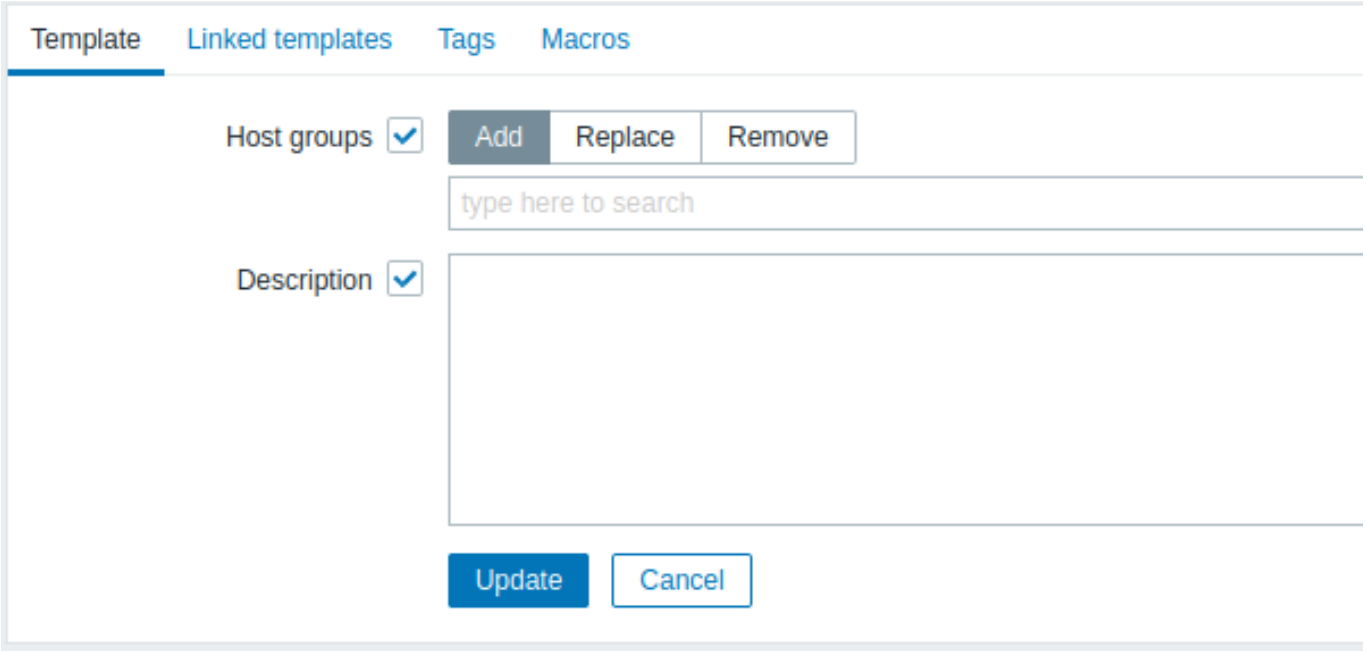
概述

有时您可能需要同时更改多个模板的某些属性。您可以使用批量更新功能，而不是打开每个模板进行编辑。

使用批量更新

批量更新一些模板，操作如下:

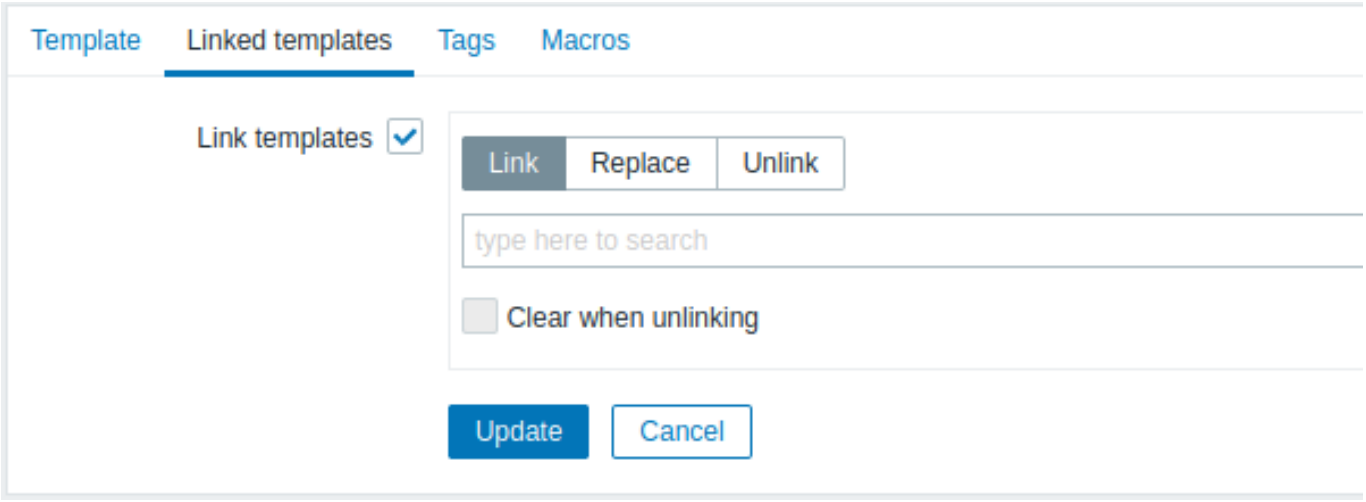
- 在模板列表 中，标记要更新模板前的复选框
- 点击列表下面的 批量更新
- 页面跳转到带有所需属性 (模板、链接的模板或标记) 的选项卡
- 标记要更新的任何属性的复选框，并为它们输入一个新值



当选择更新 主机群组复选框时，将有以下选项可以选择:

- 添加 - 允许从现有的主机组中指定额外的主机组或为模板输入全新的主机组。
- 替换 - 将从任何现有主机组中删除模板，并将它们替换为在此字段中指定的模板 (现有或新的主机组)。
- 移除 - 将从模板中删除特定的主机组。

快捷搜索 - 在搜索框输入关键字后，会自动提供一个匹配主机组的下拉列表。如果主机组是新创建的，它也会出现在下拉列表中，并且在字符串后面用新 *(new)* 表示。数量较多时，只需向下滚动选择即可。

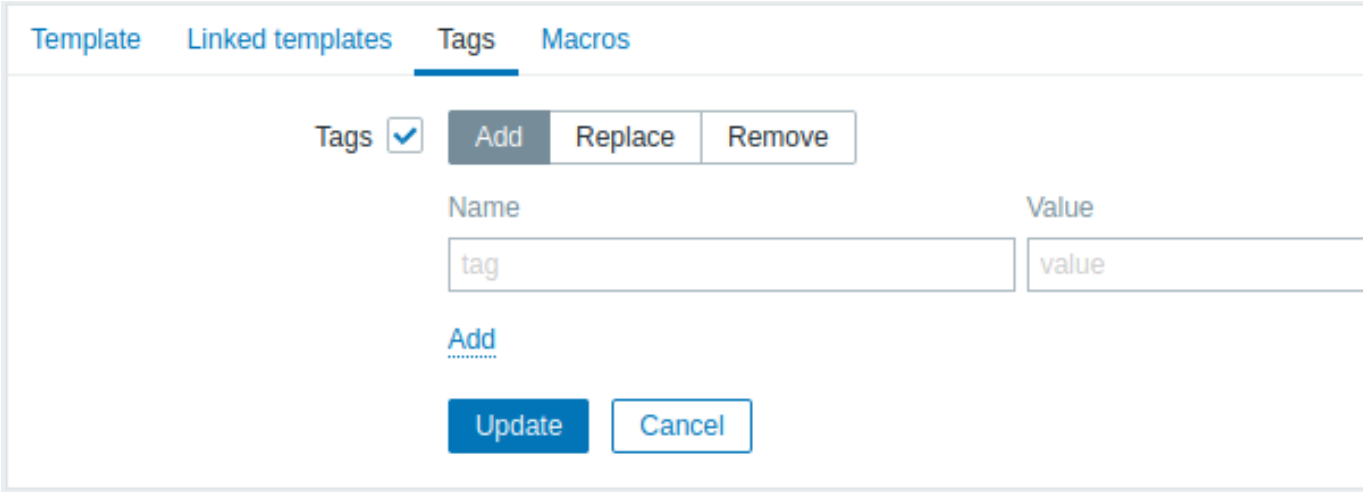


当选择 模板链接复选框时，可以使用以下选项:

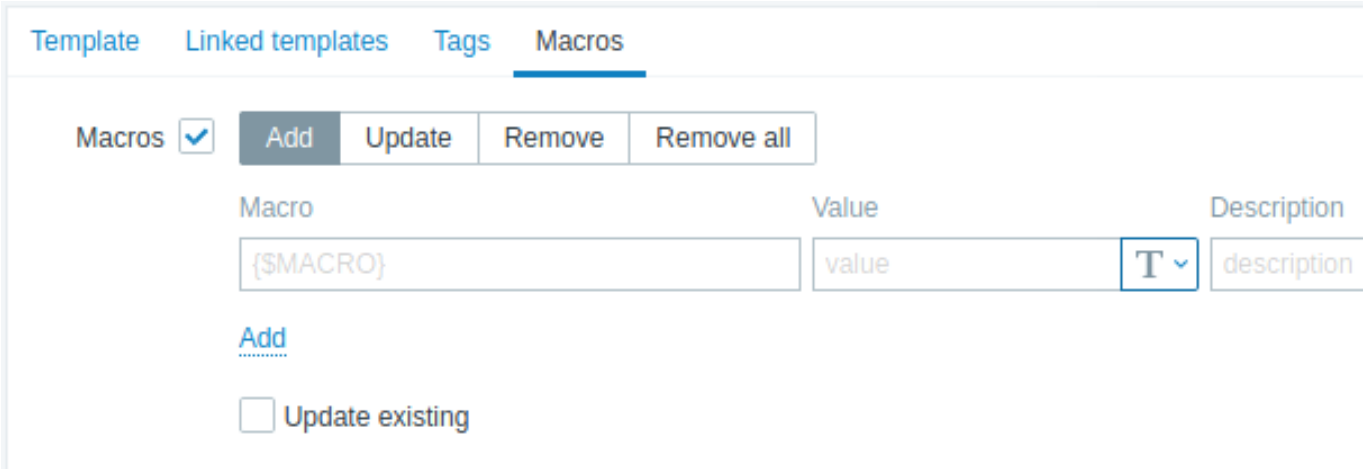
- 链接 - 指定要链接的附加模板。
- 替换 - 指定要链接的模板，同时取消链接之前链接到模板的任何模板。
- 取消链接 - 指定要取消链接的模板。

快捷搜索 -在搜索框输入模板名关键字后，将出现一个提供匹配模板的下拉菜单。只需向下滚动选择要链接/取消链接的模板即可。

清除当无链接选项不仅允许取消任何先前关联的模板的关联，而且还可以删除从它们继承的所有元素 (监控项、触发器等)。



注意，这些 {INVENTORY.*}，{HOST.HOST}，{HOST.NAME}，{HOST.CONN}，{HOST.DNS}，{HOST.IP}，{HOST.PORT} 和 {HOST.ID} 被在标记 (tags) 中支持的用户宏，在具有相同名称但不同值的标记时不被认为是“重复”，可以添加到相同的模板中。



当选择 宏复选框时，可以使用以下选项:

- 添加 - 允许为模板指定额外的用户宏。如果选中更新现有的复选框，则只更新指定宏名称的值、类型和描述。反之，如果模板中已经存在同名的宏，则不会更新该宏。
- 更新 - 将替换列表中指定的宏的值、类型和描述。如果选中添加缺失复选框，则模板上先前不存在的宏将被添加为新的宏。反之，则只有模板中已经存在的宏才会被更新。
- 移除 - 将从模板中删除指定的宏。如果选中除选中复选框，则除列表中指定的宏之外的所有宏都将被删除。如果未选中，则只删除列表中指定的宏。
- 全部移除 - 将从模板中删除所有的宏。如果未选中我确认移除所有宏复选框，将弹出需要选择该选项的警告提示框。

完成所有需要的更改后，点击更新按钮。所有选择模板的属性都会相应地更新。

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 监控的首要任务之一。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称之为“服务器 X”的主机，然后才能向其添加监视项。

主机组是由主机组成的。

前往创建主机。

1 创建主机

概述

按照以下步骤在 Zabbix 前端创建一台主机：

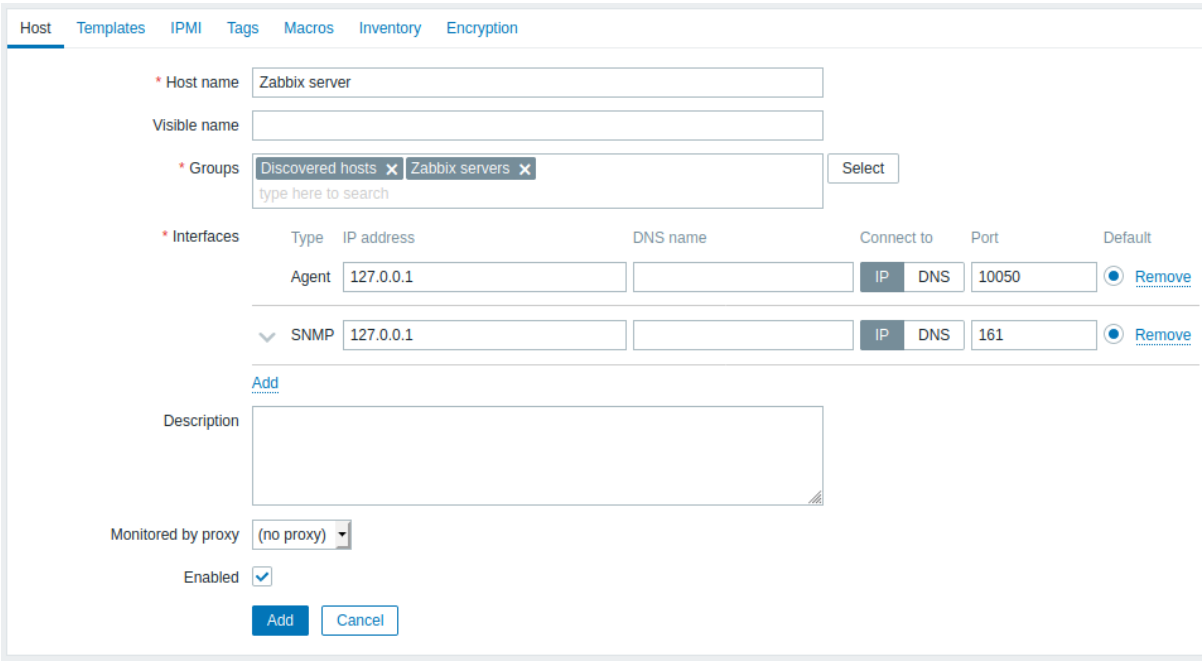
- 进入：配置 → 主机
- 单击右侧 创建主机 (或者在主机名上单击以编辑一台已有的主机)
- 在表单中输入主机的相关参数

你还可以在已经存在的主机上使用 *Clone*（克隆）和 *Full clone*（全克隆）按钮来创建一台新的主机，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（例如更改的监控项采集间隔、修改正则表达式或在低级自动发现规则添加监控项原型）都不会再自动更改到新主机；相反，他们将与最初模板一致。

配置

这个 **Host** 标签页包含了通用的主机属性：



所有必填字段都标有红色星号。

属性	描
Host name（主机名）输入唯一	主机名。允许有字母、数字、空格、点、破折号和下划线。但是，不允许在最前或最后使用空格。注意：在要配置的主机上运行 Zabbix agent 的情况下，此 agent 配置文件的参数 *Host-name*

属性	描
Visible name（可见名）显示的名	。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（群组）选择主	所属的主机组。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，可以创建新组并将主机链接到主机组。

属性	描
Interfaces（接口）支持多	主机接口类型: *Agent*，*SNMP*，*JMX* 和 *IPMI*。要增加新接口，在 *Interfaces** (接口) 处点击 *Add* (添加) 并输入 *IP/DNS*，*Connect to* (连接到) 和 *Port* (端口) 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（*IP* 地址）主机的	P 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界面。
Description（描述）填写主	描述。

属性	描	
Monitored by proxy（*agent* 代理程序）主机可以被		abbix server 或者 Zab-bix proxy 监控: **(no proxy)** - 主机被 Zab-bix sever 监控 **Proxy name** - 主机被 Zab-bix proxy“代理服务器名称”监控

属性	描
Enabled（启用）选中此	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。

通过 **Templates** 选项卡，你可以将模板链接到主机。所有实体（监控项、触发器、图形和应用集）将从模板继承。

要链接一个新模板，请开始在 *Link new templates*（链接到模板）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*（添加）。

要取消链接模板，请使用 *Linked templates*（取消模板链接）区域的两个选项之一:

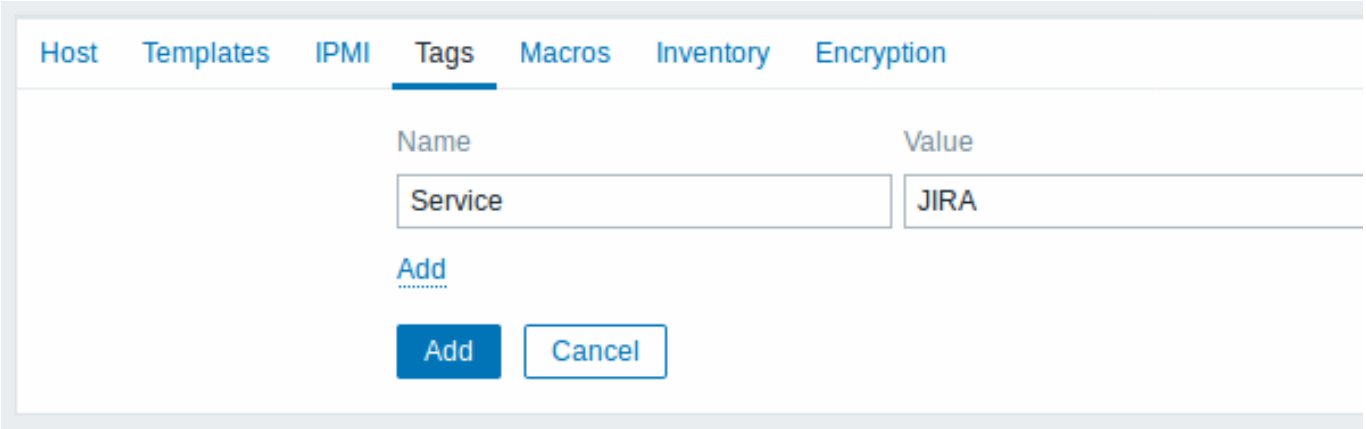
- *Unlink*（取消链接） - 取消链接模板，但保留它的监控项、触发器和图表
- *Unlink and clear*（取消链接并清理） - 取消链接模板并删除所有它的监控项、触发器和图表

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。

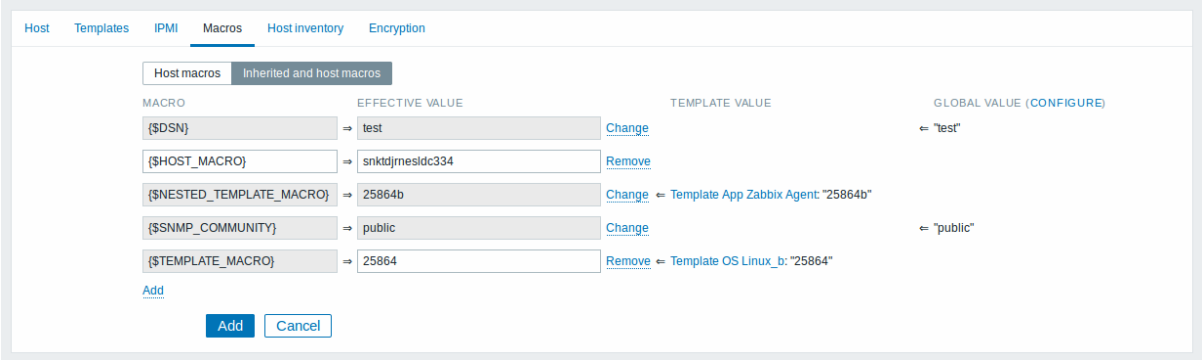
通过 **Tags**（标签）选项卡，你可以定义主机级别的标签。该主机的所有问题都将使用此处输入的值进行标记。



标签支持以下宏：用户宏、{INVENTORY.*} 宏、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID}。

通过 **Macros**（宏）选项卡，你可以定义主机级别的用户宏。也支持添加描述。

如果您选择了 *Inherited and host macros*（继承宏和主机宏）选项，您还可以在这里查看模板和全局的用户宏。这里将显示主机的所有已定义的用户宏以及它们解析的值以及其来源。



为方便起见，提供了指向各个模板和全局宏配置的链接。还可以在主机级别上编辑模板/全局宏，从而在主机上有效地创建宏的副本。

通过 **Host inventory**（主机资产清单）选项卡，你可以手动输入主机资产 信息。也可以选择启用 自动资产信息填充，或者禁用此主机的资产信息填充。

通过 **Encryption**（加密）选项卡，你可以与主机建立加密 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.

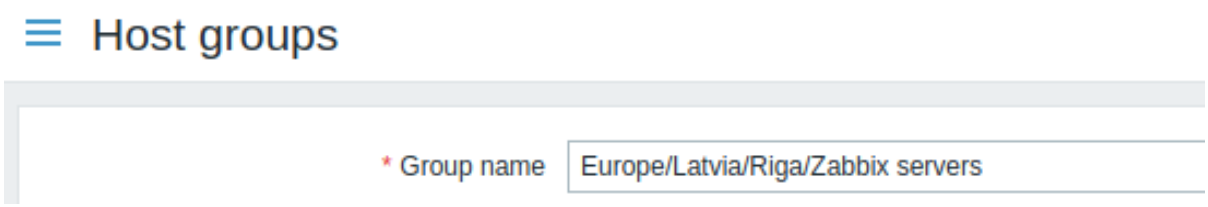
参数	描
PSK	预共享密钥（hex-string）。如果 Zabbix 使用 GnuTLS 或者 OpenSSL 库，最大长度：512 位十六进制数，如果 Zabbix 使用 mbed TLS（PolarSSL）库，则是 64 位十六进制（32 字节 PSK）。示例：1f87b595725ac5

创建主机组

要在 Zabbix 页面创建主机组，请执行以下步骤:

- 进入: *Configuration → Host groups*
- 单击页面右上角的 *Create Group*（创建组）

- 在表单中输入组的相关参数
-



所有必填字段都标有红色星号。

参数描	
Group name（组名）输入唯	的主机组名称。要创建嵌套的主机组，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，创建父主机组取决于

参数描	
Apply permissions to all subgroups	此复选框仅对 Zabbix Super Admin 用户可用，并且仅在编辑现有主机组时才可用。\\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

参数描

嵌套主机组的权限

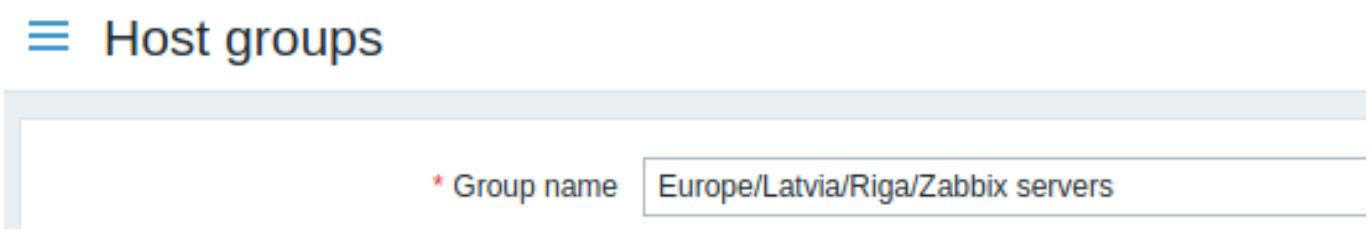
- 在为现有父级主机组创建子级主机组时，从父级继承对子级的用户组 权限 (例如，创建 `Riga/Zabbix servers` 时 `Riga` 已经存在)
- 在为现有子主机组创建父主机组时，不会设置父级的权限 (例如，创建 `Riga` 时 `Riga/Zabbix servers` 已经存在)

Creating a host group

> **Attention:**
> Only Super Admin users can create host groups.

To create a host group in Zabbix frontend, do the following:

- Go to: *Configuration* → *Host groups*
- Click on *Create Group* in the upper right corner of the screen
- Enter parameters of the group in the form



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Group name	Enter a unique host group name. To create a nested host group, use the '/' forward slash separator, for example `Europe/Latvia/Riga/Zabbix servers`. You can create this group even if none of the three parent host groups (`Europe/Latvia/Riga`) exist. In this case creating these parent host groups is up to the user; they will not be created automatically. Leading and trailing slashes, several slashes in a row are not allowed. Escaping of '/' is not supported. Nested representation of host groups is supported since Zabbix 3.2.0.
Apply permissions and tag filters to all subgroups	Checkbox is available to Zabbix Super Admin users only and only when editing an existing host group. Mark this checkbox and click on *Update* to apply the same level of permissions/tag filters to all nested host groups. For user groups that may have had differing permissions assigned to nested host groups, the permission level of the parent host group will be enforced on the nested groups. This is a one-time option that is not saved in the database. This option is supported since Zabbix 3.4.0.

Permissions to nested host groups

- When creating a child host group to an existing parent host group, user group permissions to the child are inherited from the parent (for example, when creating `Riga/Zabbix servers` if `Riga` already exists)
- When creating a parent host group to an existing child host group, no permissions to the parent are set (for example, when creating `Riga` if `Riga/Zabbix servers` already exists)

2 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 前端页面有一个特殊的 *Inventory*（资产记录）菜单。但你一开始不会看到任何数据，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录入的。

构建资产库

手动模式

配置主机时，你可以在 *Host inventory*（主机资产清单）选项卡中输入设备类型、序列号、位置、负责人等详细信息 - 这些数据将填充进资产信息。

如果主机资产信息中包含 URL，并以 “http” 或 “https” 开头，则会在 *Inventory*（资产）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*（主机资产清单）选项卡中的清单模式必须设置为 *Automatic*（自动）。

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。

以下是对资产自动填充特别有用的监控项：

- 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

可以在主机配置表单中选择资产模式。

默认情况下，新主机的资产模式是根据 *Administration*（管理）→ *General*（一般）→ *Other*（其他） 中的默认主机资产模式设置选择的。

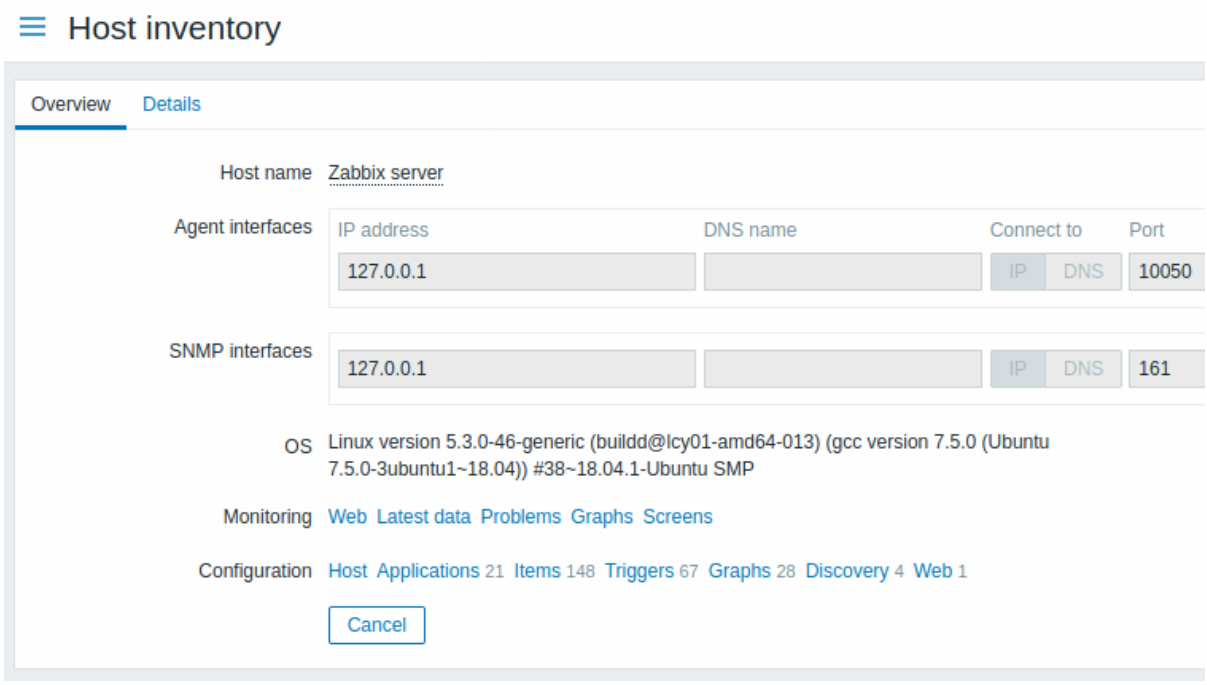
对于通过网络发现或自动注册操作添加的主机，可以定义 *Set host inventory mode*（设置主机资产记录模式）操作，选择手动或自动模式。此操作将覆盖 *Default host inventory mode*（默认主机资产记录模式）设置。

资产清单概述

Inventory（资产记录）菜单中提供了所有现有资产数据的详细信息。

在 *Inventory*（资产记录）→ *Overview*（概览）你可以通过资产的各个字段获取主机数。

在 *Inventory*（资产记录）→ *Hosts*（主机）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（概览）标签展示：

参数	描
Host name	主机名。 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，表示主机正处于维护状态。
Visible name	主机可见名（如果已定义）
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（如果已定义）。
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。

参数	描
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supported macro（支持的宏） 页面。

3 批量更新

概述

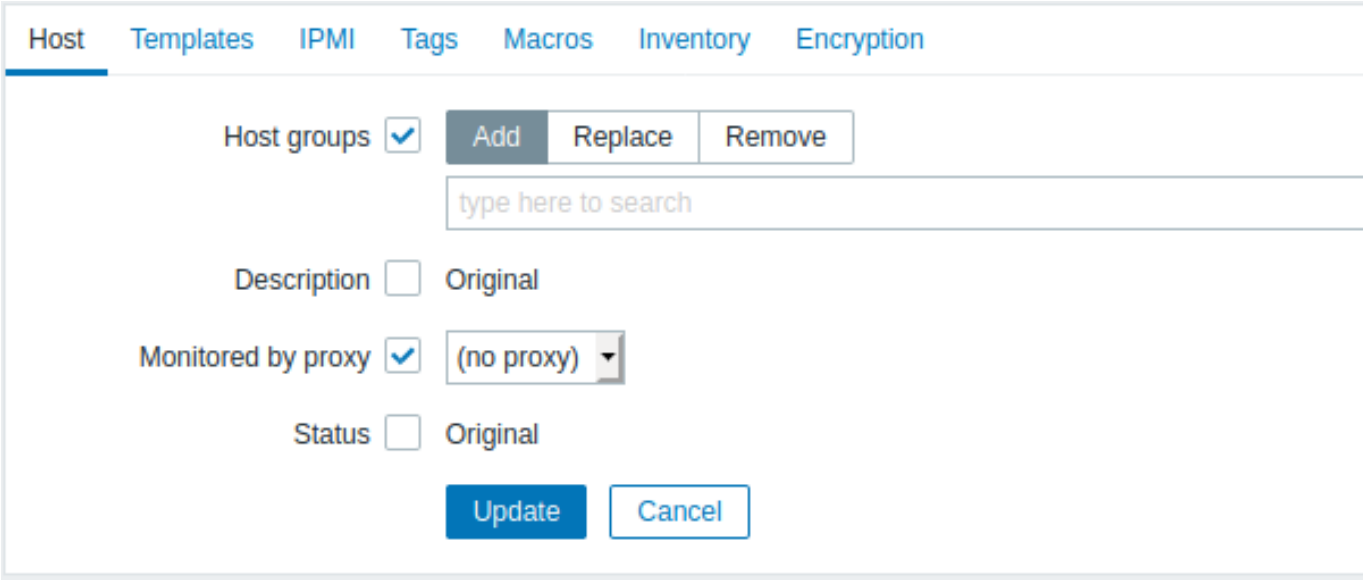
有时你可能想要一次更改多个主机的某些属性，那么你可以使用批量更新功能来代替打开每个主机进行编辑。

使用批量更新

要批量更新某些主机，请执行以下操作:

- 在主机列表中，选中要进行批量更新的主机前面的复选框
- 点击页面下方的 *Mass update*（批量更新）按钮

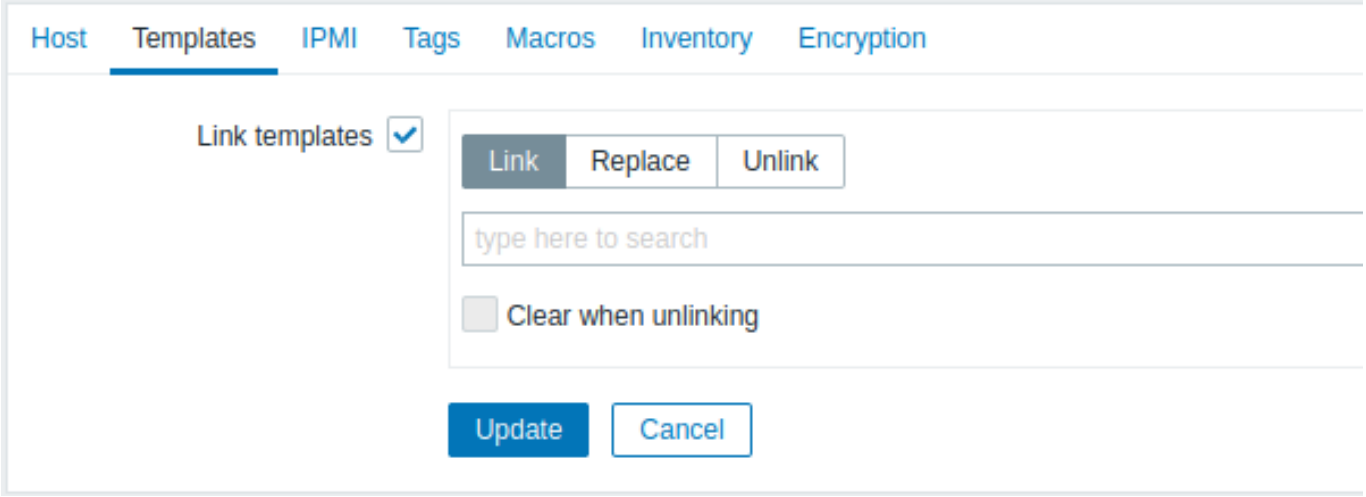
- 跳转到属性对应的选项卡 (*Host*（主机）、*Templates*（模板）、*IPMI*、*Inventory*（资产记录）或 *Encryption*（加密））
- 选中要更新的任何属性的复选框，并输入新值



当选择主机组对应的更新按钮时，可以使用以下选项：

- *Add*（添加） - 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。
- *Replace*（替换） - 将从任何现有主机组中删除主机，并替换为此字段中指定的主机组（现有或新的主机组）。
- *Remove*（移除） - 将从主机中删除特定主机组。

这些字段都是可以自动填充的 - 开始输入时会提供匹配的主机组的下拉列表。新的主机组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。

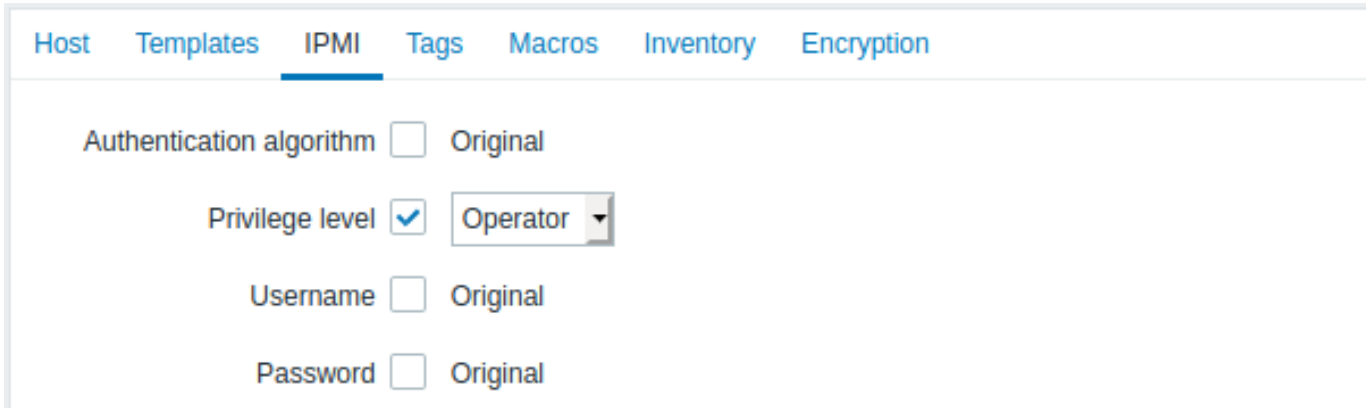


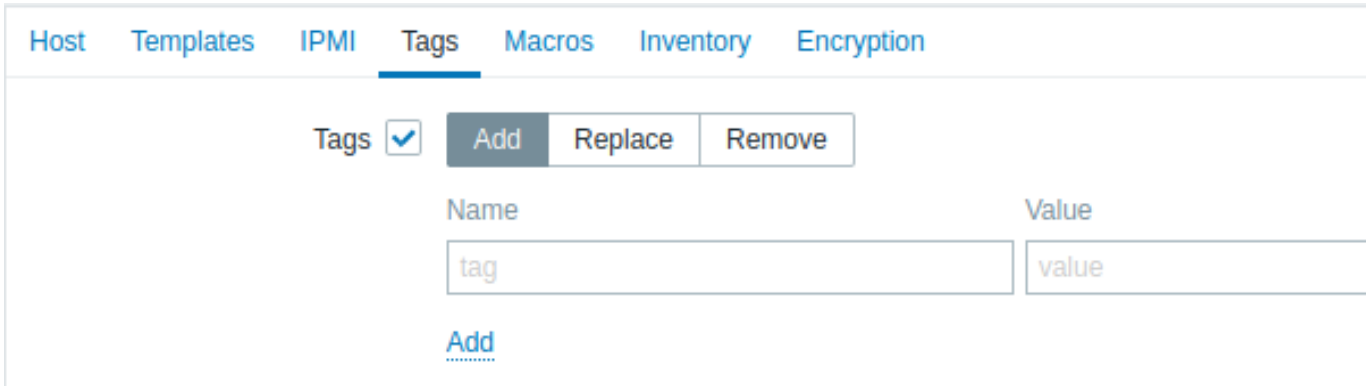
当选择模板对应的更新按钮时，可以使用以下选项：

- *Link*（链接） - 指定要链接的其他模板。
- *Replace*（替换） - 指定要链接的模板，同时取消之前链接到主机的任何模板。
- *Unlink*（取消链接） - 指定要取消链接的模板。

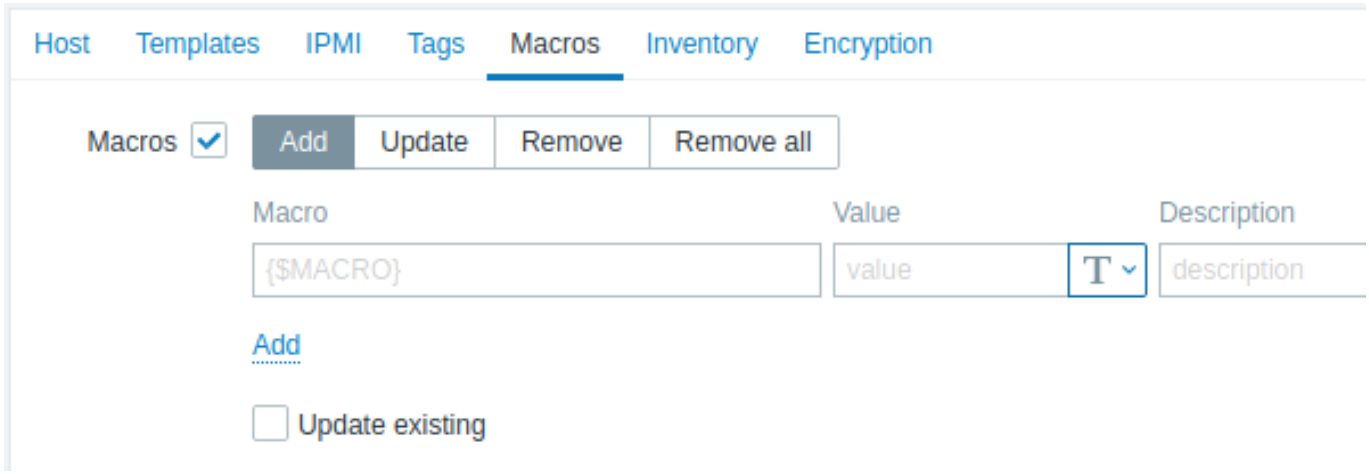
要指定链接/取消链接的模板，请在搜索框（在此输入搜索）字段中输入模板名，直到出现一个提供匹配模板的下拉菜单。只需向下滚动即可选择所需的模板。

当选中 *Clear when unlinking*（当取消链接时清除）复选框，在取消与任何以前链接的模板的关联的同时还会删除所有继承自该模板的元素（监控项、触发器等）。



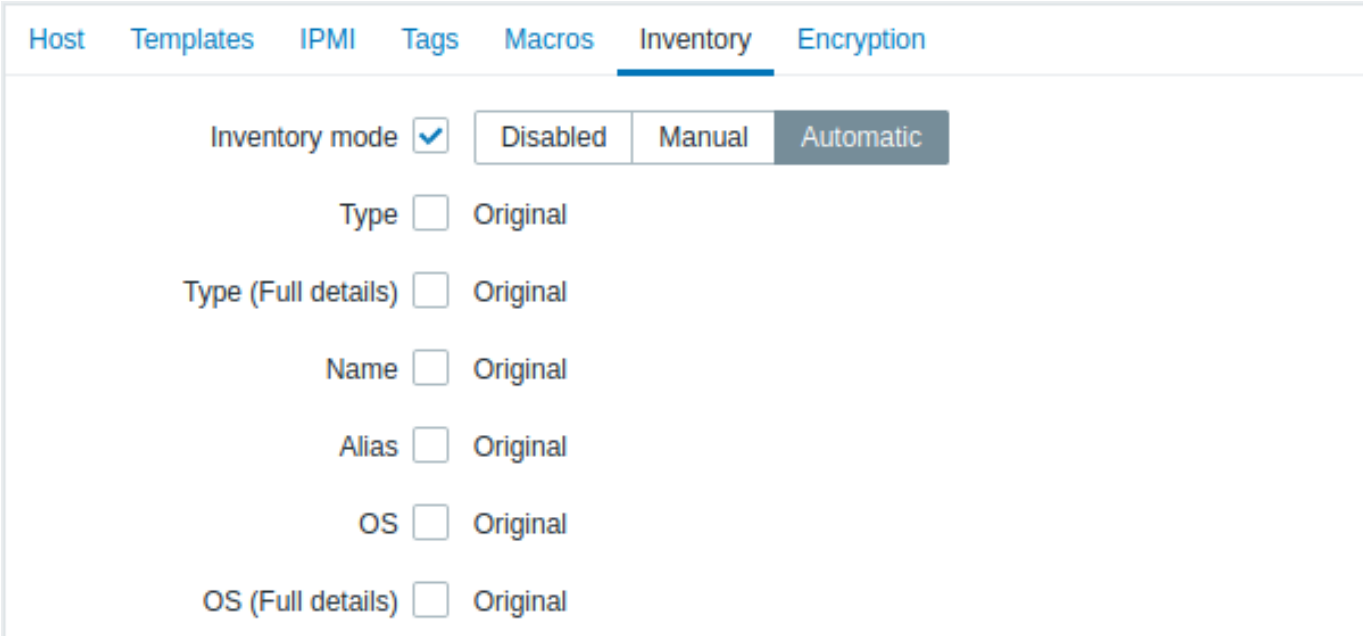


标记中支持用户宏、{INVENTORY.*} 宏、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 宏。注意，名称相同但值不同的标记不视为“重复项”，可以添加到同一主机。

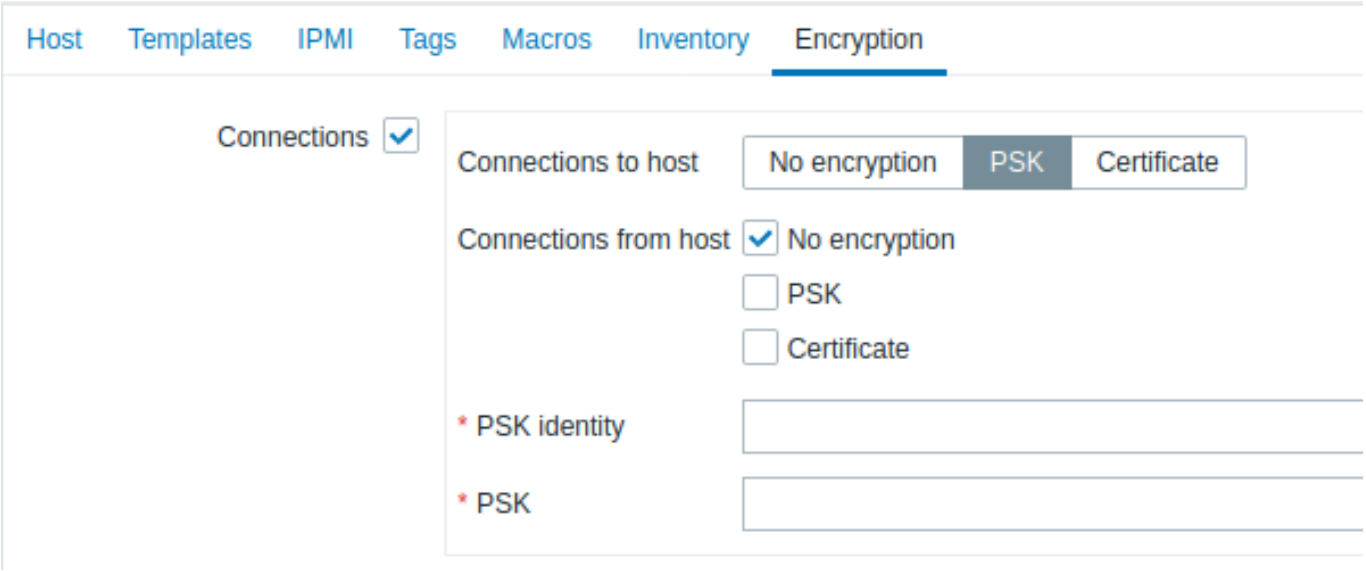


选择宏相对应的更新按钮时，可以使用以下选项：

- *Add*（添加） - 允许为主机指定额外的用户宏。如果选中了 *Update existing*（更新现有的）复选框，则将更新指定宏名称的值、类型和描述。不选的话，如果主机上已存在具有该名称的宏，则不会对其进行更新。
- *Update*（更新） - 将替换此列表中指定的宏的值、类型和描述。如果选中 *Add missing*（添加所缺的）复选框，则主机上以前不存在的宏将被添加为新宏。如果不选，则仅更新主机上已存在的宏。
- *Remove*（移除） - 从主机中删除指定的宏。如果选中了 *Except selected*（除选定对象之外）复选框，则除列表中指定的宏之外的所有宏将被删除。如果不选，则仅删除列表中指定的宏。
- *Remove all*（移除所有） - 从主机上删除所有用户宏。如果选中了 *I confirm to remove all macros*（我确认删除所有的宏）复选框，将弹出一个要求确认删除所有宏的窗口。



为了能够批量更新资产记录，*Inventory mode*（资产记录模式）应该设置为’ 手动’ 或’ 自动’。



完成所有必要的更改后，单击 *Update*（更新），所有选定主机的属性将相应更新。

2 监控项

概述

监控项是从主机收集数据的项目。

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链接到主机。然而，为了系统性能达到最佳，您可能需要对模板进行微调，使其只有必要的监控项和频繁的监视。

在单个监控项中，你可以指定从主机收集哪些数据。

为此，你可以使用监控项键值 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。

要用键值 key 指定更多的参数，请在 key 后的方括号中添加这些参数。例如，system.cpu.load**[avg5]** 将返回最近 5 分钟的 CPU 负载平均值，而 net.if.in**[eth0]** 将显示接口 eth0 的传入流量。

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

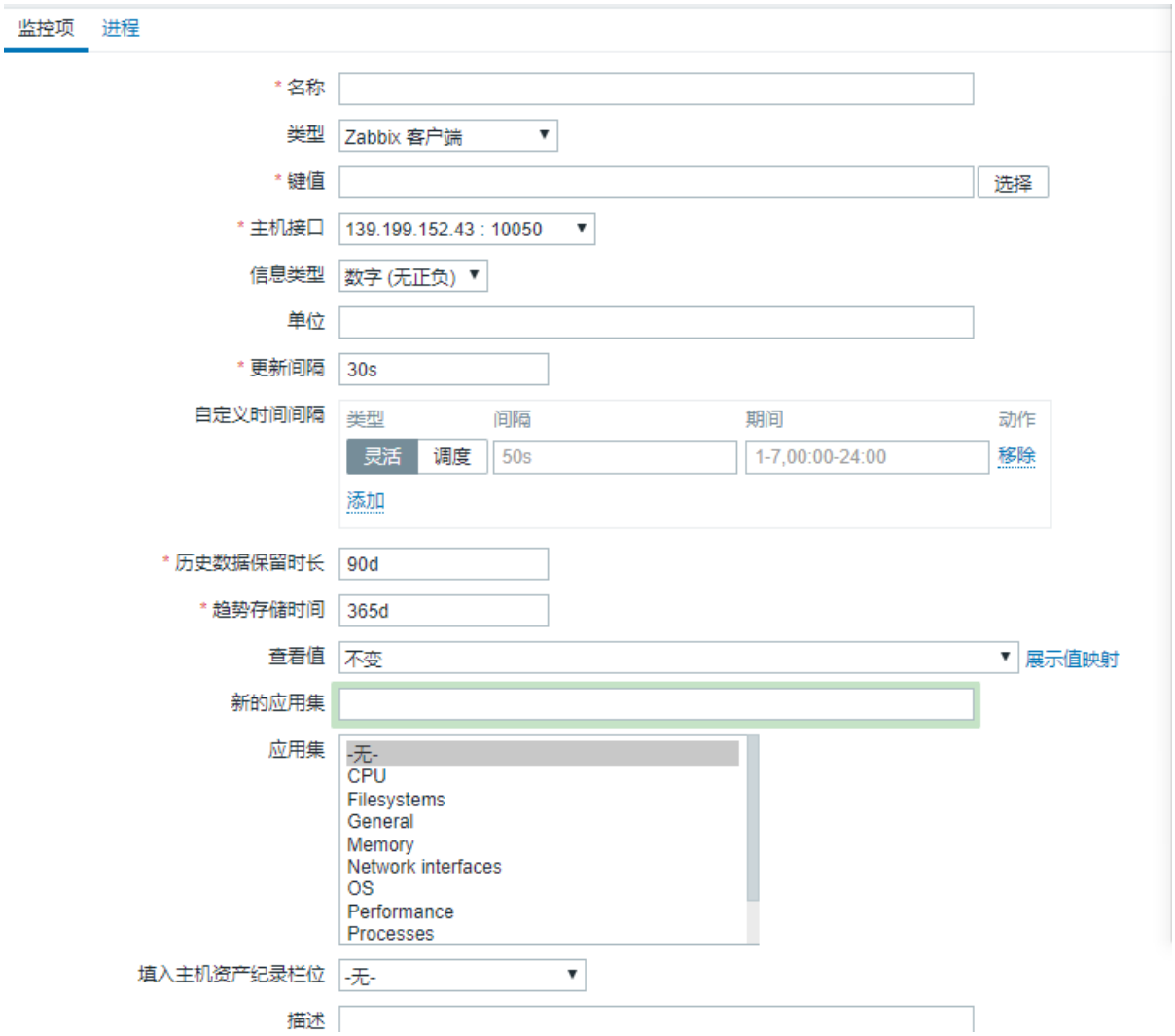
要在 Zabbix 管理页面创建一个监控项，请执行以下操作：

- 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

你也可以打开一个已经存在的监控项，点击克隆按钮，然后重命名保存。

配置

监控项选项卡包含了常规监控项属性：



所有必填字段都用红色星号标记。

参数描

名称监	项的名称。 注意，现在不建议使用位置宏（$1,$2... $9-指代项键的第一个，第二个.....第九个参数）。 例如：$1 上的可用磁盘空间 如果监控项的键值是”vfs.fs.size[/,free]”，描述将自动更改为”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（最多 2048 个字符）。 支持的监控项键值 能够在各个监控项类型中找到。 键值在单个主机中必须是唯一的。 如果键值类型是 ‘Zabbix 客户端’、‘Zabbix 客户端 (主动式)’、‘简单检查’ 或 ‘Zabbix 整合’，则这个键值必须被 Zabbix agent 或者 Zabbix server 支持。 也可以查看: 标准的键值格式。
主机接口定义主	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 **(**浮点数**)** - 64 位浮点数 允许大约 15 位的精度，范围从-1.79E+308 到 1.79E+308(PostgreSQL 11 和更早版本除外)。也支持用科学计数法接收值。例如:1.23E+7，1e308，1.1E-4。 字符 - 短文本数据 日志 - 具有可选日志相关属性 (timestamp（时间戳），source（源），severity（严重性），logeventid（日志事件 id）) 的长文本数据 文本 - 长文本数据。可参见文本数据限制。
单位如	设置了单位，Zabbix 在接收到数据后会进行处理，使其匹配设置的单位。 默认情况下，如果原始值超过 1000，则除以 1000 再显示。例如，如果将单位设置为 *bps* 并接收到值 881764，它将显示为 881.76 Kbps。 JEDEC存储器标准用于处理 B(字节)，Bps(字节/秒) 单位，它除以 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”。想要正确转换，接收的值必须是数字（无正负）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”。 注意，如果单位带有前缀 “!”，单元前缀/处理不会应用到监控项的值。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。 单个宏必须填充整个字段。不支持字段中的多个宏或文本混合的宏。\\注意：仅当自定义间隔存在非零值时，更新间隔才能设置为 “ 0”。如果设置为 “ 0”，并且存在自定义间隔（灵活的或计划的）且具有非零值，则将在自定义间隔持续时间内轮询该项目。 注意项目成为活动后或更新间隔更改后的第一个项目轮询可能发生在配置值之前。 注意可以通过点击执行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**（灵活） - 为更新间隔 (不同频率的间隔) 创建一个特例 **Scheduling**（调度） - 创建自定义轮询时间表。 详细信息请查看自定义时间间隔。 间隔字段支持时间单位，例如 30s, 1m, 2h, 1d. 支持用户宏。 从 Zabbix 3.0.0 开始支持计划。 注意: 不适用于 Zabbix Agent 的活动监控项。
历史数据保留时长以下可供选择：	**Do not keep history**-不存储监控项历史数据。如果只有依赖项需要保留历史记录，则对主监控项很有用。 此设置不能被全局管家设置覆盖。 **Storage period**-指定将详细历史数据保留在数据库中的时长（1 小时至 25 年）。过期数据将被 housekeeper 管家删除。按秒存储。支持时间后缀，例如 2h，1d。支持用户宏。 *Administration*（管理）→ *General*（一般）→ 管家 中可以覆盖该值。 如果存在全局设置，则显示绿色告警信息 History storage period 1w Overridden by global housekeeping settings (1d) ， 例如被全局管家设置 (1d) 覆盖。 建议保留最小可能天数的历史数据，以减轻数据库存储压力。可以保留更长的趋势数据来替代历史数据。 历史数据与趋势数据的区别请参考历史与趋势。
趋势数据保留时长以下可供选择：	**Do not keep trends** - 将不会存储趋势数据。 此设置不能被全局管家设置覆盖。 **Storage period** - 指定在数据库中保存聚合（每小时，最大，平均，计数）历史数据的时长（1 天至 25 年）。管家将删除较旧的数据。按秒存储。支持时间后缀，例如 2h，1d。支持用户宏。在 *Administration*（管理）→ *General*（一般）→ 管家中可以覆盖该值。 如果存在全局设置，将显示一条绿色的警告消息： Trend storage period 365d Overridden by global housekeeping settings (7d) ， 例如被全局管家设置 (7d) 覆盖。 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。 它适用于数字（无正负）、浮点数、字符类型。 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**:** 年 ***(1970-2038)* *** M**:** 月 ***(01-12)* *** d**:** 日期 ***(01-31)* *** h**:** 小时 ***(00-23)* *** m**:** 分钟 ***(00-59)* *** s**:** 秒 *(00-59)* 如果留空，则不会解析时间戳。 例如，参考 Zabbix Agent 日志文件中以下行：“23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).” 它以 6 个字符的 PID 作为开始，后跟日期、时间和其余部分。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，可以是 “ yMdhms” 以外的任何字符。
新的应用集该监控项	所属的新应用集的名称。
应用集将监	项链接到一个或多个现有应用集。
填入主机资产纪录栏位你可以选中该监控项	将其填充到主机资产记录字段。如果主机启用了主机资产记录自动添加，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:
监控项类型的特定字段在对应页面会有描述。

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该监控项。

监控项值预处理

预处理选项卡允许为接收到的值定义预处理规则。

测试

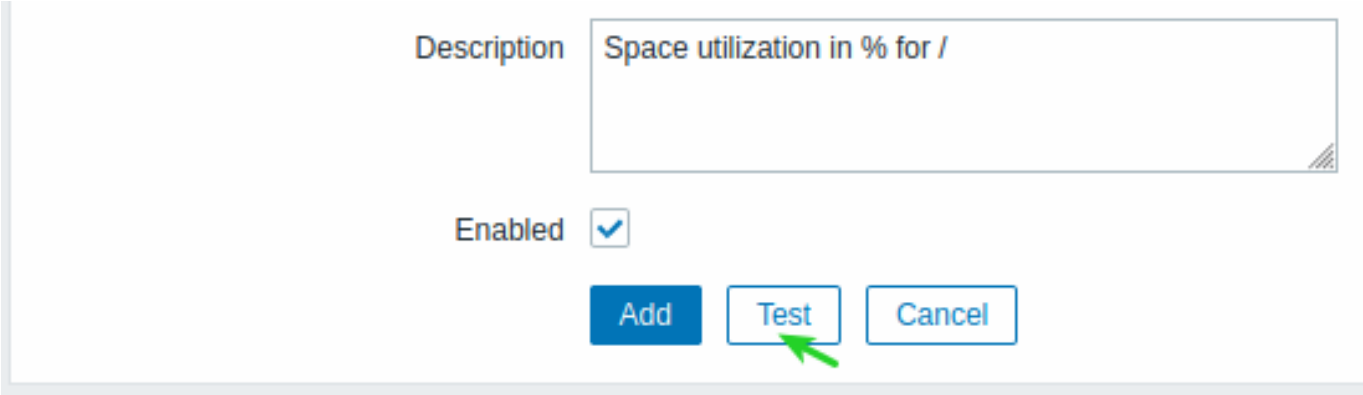
可以对监控项进行测试，如果配置正确，则可以返回实际值。甚至可以在保存项目之前进行测试。

可以对主机和模板的监控项、监控项原型和自动发现规则进行测试。agent（主动式）类型的监控项不能测试。

监控项测试可用于以下被动式监控项:

- Zabbix agent
- SNMP agent (v1, v2, v3)
- IPMI agent
- SSH checks
- Telnet checks
- JMX agent
- Simple checks (except `icmpping*`, `vmware.*` items)
- Zabbix internal
- Zabbix aggregate
- Calculated items
- External checks
- Database monitor
- HTTP agent

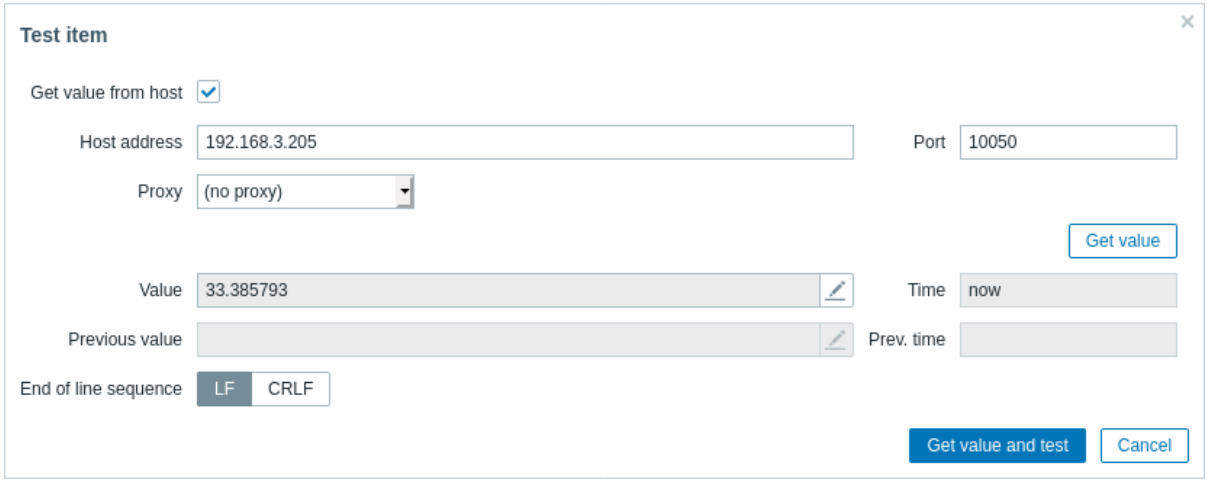
要测试监控项，请单击监控项配置表单底部的 测试按钮。请注意，对于无法测试的监控项（例如简单检查之外的其他主动检查），测试按钮将被禁用。



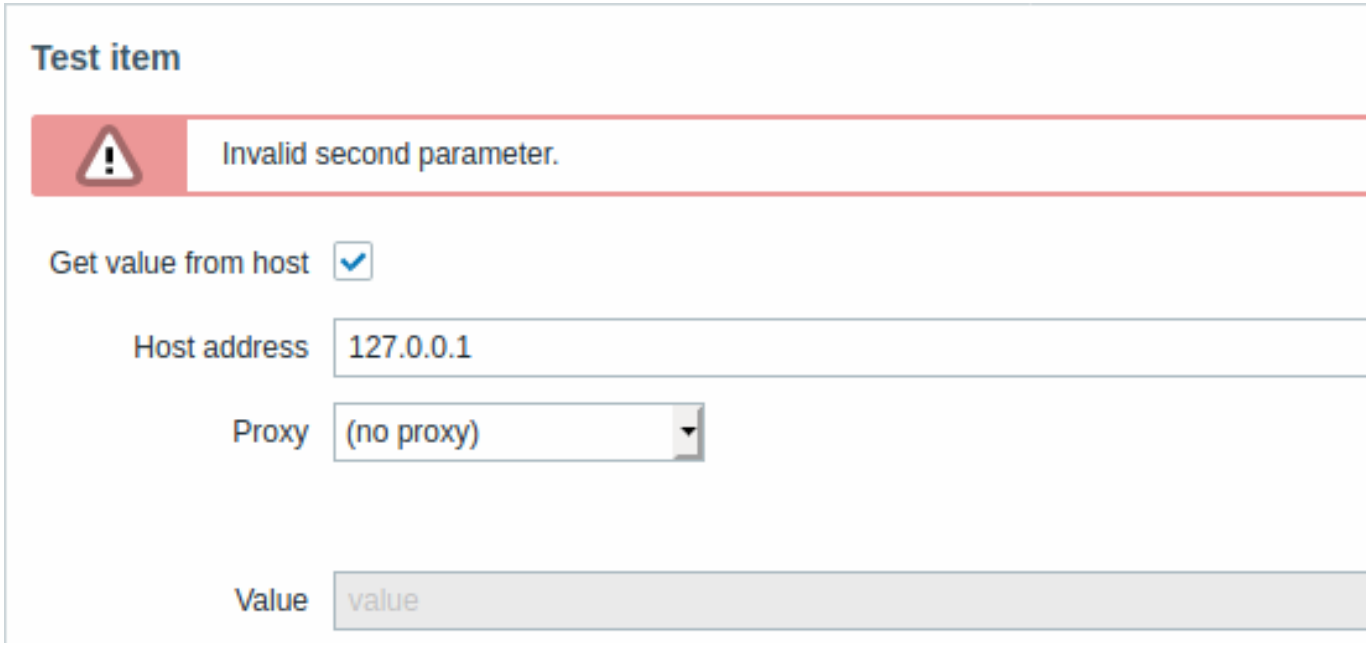
项目测试表单包含主机所需的参数 (主机 IP 地址、端口、agent 代理程序/无 agent 代理程序)。这些字段是上下文相关的:

- 所需的参数值可能是已填充的，例如，这个监控项是主机的监控项，所需信息就会从 agent 主机的接口传递过来
- 模板的监控项的相关参数需要手动填充
- 当在特定的监控项类型中上下文不需要的字段会被禁用（例如，在可计算和 zabbix 整合类型的监控项中主机地址地段被禁用，在可计算类型的监控项中，proxy 字段被禁用）

要测试监控项，点击 *Get value* (获取值)。如果成功检索到值，它会自动填充进 *Value* (值) 字段，将当前值 (如果有的话) 移动到 *Previous value* 字段，同时计算 *prev.time* 字段的值，即两个值 (两次测试) 之间的时间差，如果在检索值中检测到 “\n\r”，则尝试检测 EOL 序列并切换到 CRLF。



如果配置不正确，则返回错误提示，并描述可能的原因。



一个从主机接收成功的值必定也可以用于测试预处理。

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**（被动）检查（参见更多详细信息）。 注意当执行立即检查时，配置缓存不会更新，因此该值不会反映最新更改的监控项配置。
清除历史和趋势	删除监控项历史和趋势数据。
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后端存储数据库。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的限制：

数据库信息	型		
	Character（字符）Log	日志）Text（文本	
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

单位黑名单

默认情况下，为监控项指定单位会自动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 !B。为了更好地说明在有黑名单和没有黑名单的情况下转换的方式，请参见以下值和单位示例：

```
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, !%。

自定义脚本限制

可用的自定义脚本长度取决于使用的数据库:

数据库 //	符长度限制 // //字节限制 /	
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited
SQLite (only Zabbix proxy)	65535	not limited

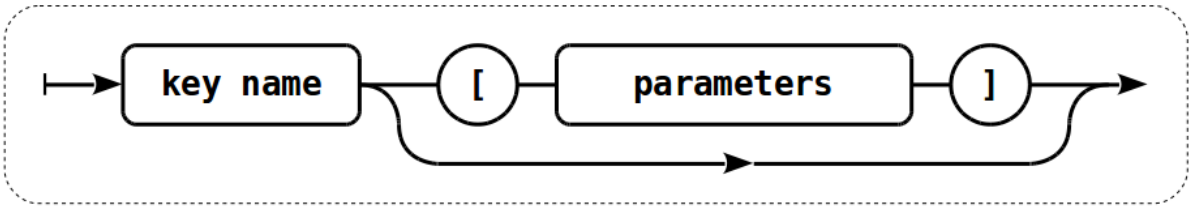
不支持的监控项

如果由于某种原因无法检索到值，则该监控项可能不受支持。但是该监控项仍会以固定时间间隔重新检索，可在管理页面中进行配置。

不支持的项目被报告为 “不支持” 状态。

1 监控项键值的格式

监控项键值的格式（包括键值的参数）必须遵循语法规则。以下插图描述了支持的语法。可以通过跟随箭头来确定每个点上允许的元素和字符 - 如果可以通过线到达某个块，则允许，否则 - 不允许。



要构建有效的监控项键值，首先要指定键值的名称，然后选择是否具有参数，后面的两行描述了这一点。

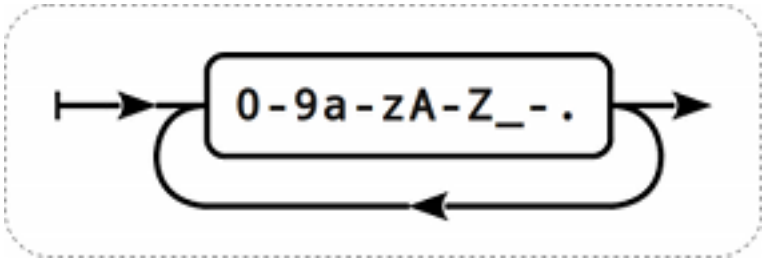
键值名称

Key（键值）名称本身具有字符范围的限制，允许的字符是：

0-9a-zA-Z_-.

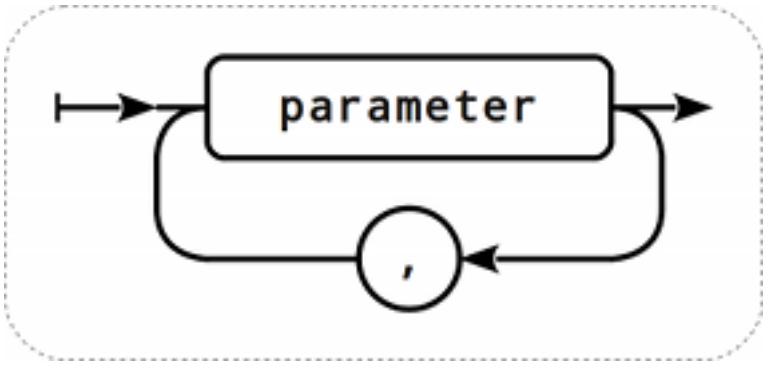
即：

- 所有数字；
- 所有小写字母；
- 所有大写字母；
- 下划线；
- 破折号；
- 点。

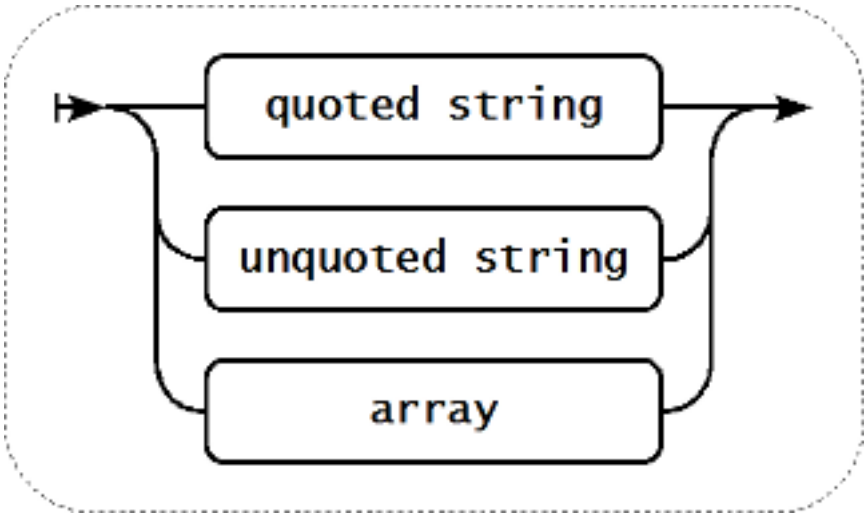


键值参数

一个键值可以包含由多个逗号分隔的参数。



每个参数可以是带引号、不带引号的字符串或数组。



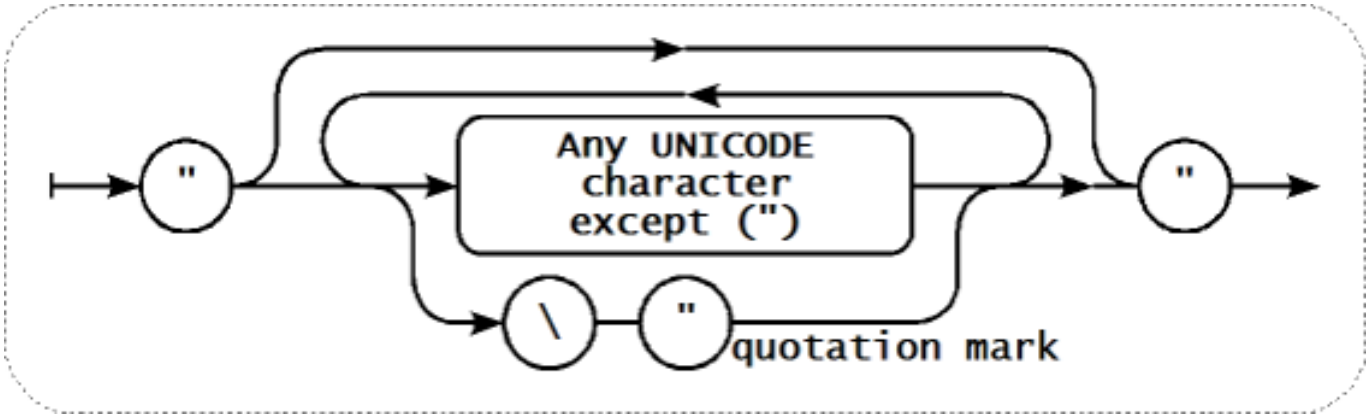
如果参数为空，则使用默认值。在这种情况下，如果指定了其它参数，则必须添加对应数量的逗号。例如，键值 **icmpping[„200„500]** 将指定每 ping 一次的时间间隔为 200 毫秒，超时时间为 500 毫秒，所有其它参数为默认值。

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。

如果键值参数的字符串中包含逗号，则该参数必须用引号引起来。

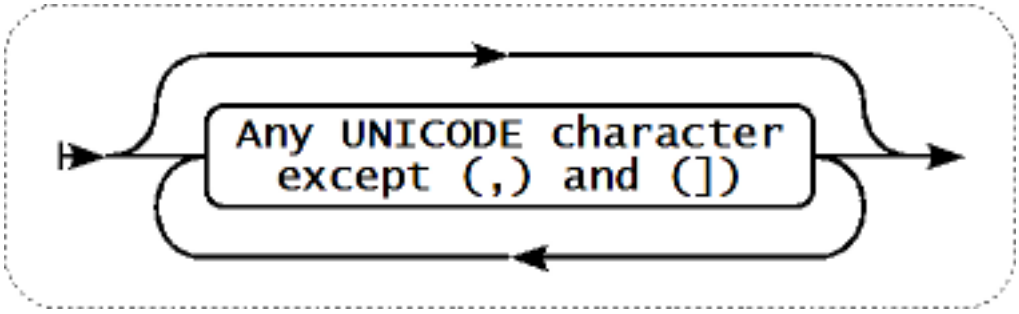
如果键值参数的字符串包含引号，则该参数必须用引号括起来，并且作为参数字符串一部分的每个引号都必须用反斜杠 (\) 进行转义。



< 提示 >：要引用监控项键值参数，只能使用双引号，不支持单引号。:::

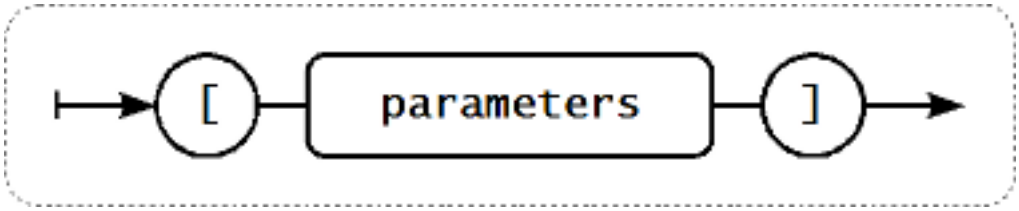
参数 - 不带引号的字符串

如果键值的参数是不带引号的字符串，则允许使用任何 Unicode 字符，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

如果键值的参数是数组，它需要被方括号括起来，其中各个参数需要符合多个参数的规则和语法。



Attention:
不支持多级参数数组，例如 `[a,[b,[c,d]],e]`。

2 自定义时间间隔

概述

监控项的自定义时间间隔有两种类型可以选择。灵活，允许重定义默认更新间隔，调度，可以使监控项在特定时间或时间序列生效。

灵活

灵活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。

注意，如果灵活间隔等于周期的长度，则该监控项将被精确采集一次。如果灵活间隔大于周期，则可能会采集该监控项一次或者完全采集(因此不建议这样配置)。如果灵活间隔小于周期，监控项将至少被采集一次。

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 秒采集一次。
0	1-7,00:00-7:00	监控项不会在夜间采集。
0	7-7,00:00-24:00	监控项不会在星期日采集。
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 开始建议使用调度方式来实现。

调度

调度用于在特定时间检查监控项。虽然默认的自定义时间间隔是灵活，但是调度常用于指定独立执行的检查计划。

调度定义为: `md<filter>wd<filter>h<filter>m<filter>s<filter>` 其中:

- **md** - month days（一月的第几天）
- **wd** - week days（一周的第几天）
- **h** - hours（小时）
- **m** - minutes（分）
- **s** – seconds（秒）

`<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。

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

前缀描	*&l	;from>* *&l	;to>*
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 前端，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。

如果同一个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

自定义时间间隔相关

谨记，在配置自定义时间间隔时，将考虑更新间隔的值。下表显示了自定义间隔和更新间隔之间的相关性。

更新间隔灵活	调度	果	
0			错误
0	非 0		活
0		非 0	度
非 0			新间隔
非 0	0	灵	
非 0		0 更	间隔和调度

2 监控项值预处理

概述

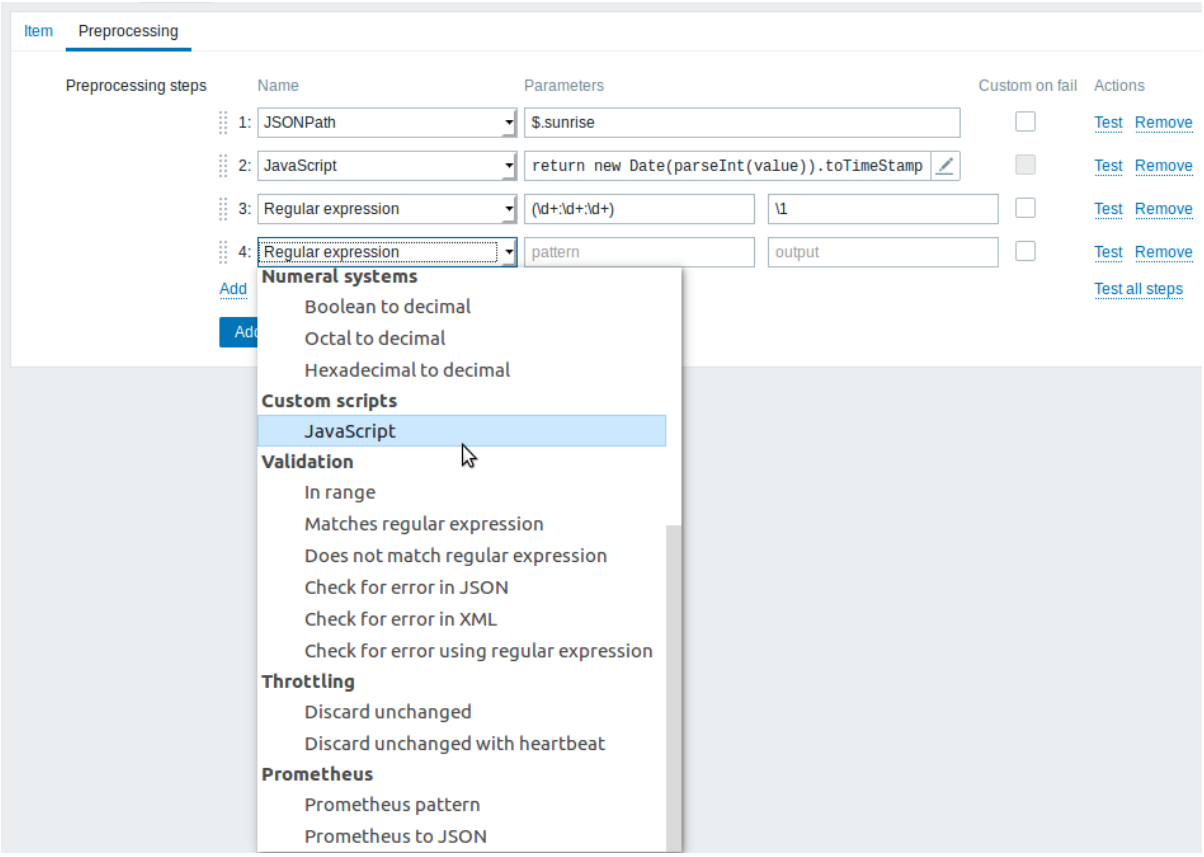
预处理允许为接收到的监控项值定义转换规则。在将值保存到数据库之前，可以进行一次或多次转换。转换按照它们定义的顺序执行。预处理是由 Zabbix server 或者 Zabbix proxy (代理监控项) 执行。

参考:

- 预处理详情
- 用法示例

配置

预处理规则在项目的预处理选项卡中进行配置 。



<note important> 如果任何预处理步骤失败监控项将不支持, 除非在自定义失败选项中指定了自定义错误处理。

对于日志监控项，日志元数据 (没有值) 将总是重置监控项的不支持的状态，使监控项再次受到支持，即使在从 agent 接收到日志值后发生了初始错误。:::

用户宏 在项值预处理参数中支持带有上下文的用户宏, 包含 JavaScript 代码。

Note:
当宏被它的值替换时，上下文将被忽略。宏值将直接插入到代码中，在将值放入 JavaScript 代码之前不能添加额外的转义。请注意，在某些情况下，这可能会导致 JavaScript 错误。

类型转	方式描述
文本	

类型转	方式描述	
	正则表达式将值与 &	t;pattern>正则表达式匹配，并将值替换为<out-put>。正则表达式支持提取最多10个带有\N序列的捕获组.如果不匹配输入值，则将不支持该监控项。参数:**pattern**-正则表达式**output**-

类型转	方式描述	
	替换查	搜索字符串并将其替换为另一个字符串(或不替换)。所有搜索到字符串都将被替换.参数: **search string** - 查找和替换的字符串,区分大小写(必需) **replacement** - 替换搜索到的字符

类型转	方式描述	
	修整删	值开始或者结束位置的指定字符。
	修整右边删除值	束位置的指定字符。
	修整左边删除值	始位置的指定字符。

Structured data

类型转	方式描述	
	XML XPath	使用XPath功能从XML数据中提取值或片段. 使用这个选项, Zab-bix server必须使用libxml支持编译。 例如: number(/doc 将从 <document><提取10 number(/doc 将从 <document><attribute="中提取10 /document/i 将从 <document><中提取<item><valu 注意，不支持

类型转	方式描述
	JSON Path

使用JSONPath功能从JSON数据中提取值或片段. 如果你使用 *Custom on fail* 复选框, 在预处理步骤失败的情况下，该项不会变得不受支持，并且可以指定自定义错误处理选项: 丢弃

类型转	方式描述	
	CSV to JSON	将CSV文件数据转换为JSON格式。有关更多信息，请参见: CSV转换JSON的预处理. 从4.4.0开始支持。
Arithmetic		

类型转	方式描述	
	Custom multiplier	将该值乘以指定的整数或浮点值.使用此选项转换接收到的以等变成B、Bps、KB、MBps为单位的值。否则Zab-bix无法正确设置(K, M, G等)后缀。注意如果监控项类型是数

类型转	方式描述
Change	

类型转	方式描述
	简单更改计算当值与先前值的差值。公式为**value**-**prev_value** *value* - 当前值; *prev_value* - 先前值 这个设置对于度量一个不断增长的值很有用。 如果当前值小于之前的值, Zabbix 会丢弃这个差异(不存储

类型转	方式描述	
	每秒更改计算值	变化(当前值与上一个值之间的差异)速度每秒.公式为(**value**-**prev_value**)/(**t** **prev_time**), *value* - 当前值; *prev_value* - 先前值; *time* - 当前时间; *prev_time* - 先前时间.这个设置对于获取持续增长值的每秒速度

类型转	方式描述
Numeral systems	

类型转	方式描述	
	布尔值转十进制将值从布尔格	转换为十进制。文本表示被翻译成0或1。因此，'TRUE'为1，'FALSE'为0。所有值都以不区分大小写的方式匹配。当前确认的值为: *TRUE* - true, t, yes, y, on, up, running, enabled, available, ok, master

类型转	方式描述
	八进制转十进制将值从八进制式转换为十进制.如果你使用 *Custom on fail* 复选框,在预处理步骤失败的情况下,该项不会变得不受支持,并且可以指定自定义错误处理选项:丢弃该值、设置指定值

类型转	方式描述	
	十六进制转十进制将十六进制转换	十进制.如果你使用 *Custom on fail* 复选框,在预处理步骤失败的情况下,该项不会变得不受支持,并且可以指定自定义错误处理选项:丢弃该值、设置指定值或设置指

类型转	方式描述	
Custom scripts		
	Javascript	在单击参数字段或铅笔图标时出现的块中输入JavaScript代码。注意，可用的JavaScript长度取决于所使用的使用的数据库. 有关更多信息，请参见: Javascript预处理.
Validation		

类型转	方式描述	
	In range	通过指定最小值/最大值(包括)来定义一个值的范围。接受的数值包含(任意数字，可选的小数部分和可选的指数部分，负数)。可以使用用户宏和低级发现宏。最小值应

类型转	方式描述	
	Matches regular expression	指定一个必须能匹配的正则表达式的值. 如果你使用 *Custom on fail* 复选框, 在预处理步骤失败的情况下, 该项不会变得不受支持, 并且可以指定自定义错误处理选

类型转	方式描述	
	Does not match regular expression	指定一个必须不能匹配的正则表达式的值. 如果你使用 *Custom on fail* 复选框, 在预处理步骤失败的情况下, 该项不会变得不受支持, 并且可以指定自定义错误处理

类型转	方式描述	
	Check for error in JSON	检查位于JSON-path的应用程序级错误消息。如果成功且消息不为空，则停止处理；否则，继续使用此预处理步骤之前的值进行处理。注意，没有添加预处理步骤信息的

类型转	方式描述	
	Check for error in XML	检查位于XPath的应用程序级错误消息。如果成功且消息不为空，则停止处理;否则，继续使用此预处理步骤之前的值进行处理。注意，没有添加预处理步骤信息的话，

类型转	方式描述	
	Check for error using a regular expression	使用正则表达式检查应用程序级错误消息。如果成功且消息不为空，则停止处理;否则，继续使用此预处理步骤之前的值进行处理。注意，没有添加预处理步骤信

类型转	方式描述
Throttling	

类型转	方式描述
	Discard unchanged 如果一个值没有改变，则丢弃它。如果一个值被丢弃，它就不会保存在数据库中，Zabbix Server也不知道收到了这个值。不会对触发器表达式求值，因此，不会创建/解决相关

类型转	方式描述	
	Discard unchanged with heartbeat	如果一个值在定义的时间段内(以秒为单位)没有更改，则丢弃该值。支持正整数值来指定秒数(最小值为1秒)。该字段可使用时间后缀(如30s、1m、2h、1d)。用户宏和低级

类型转	方式描述	
Prometheus		
	Prometheus pattern	使用以下查询从Prometheus指标提取所需的数据。查看Prometheus检查更多细节。
	Prometheus to JSON	将所需的Prometheus指标转换为JSON。查看Prometheus检查更多细节。

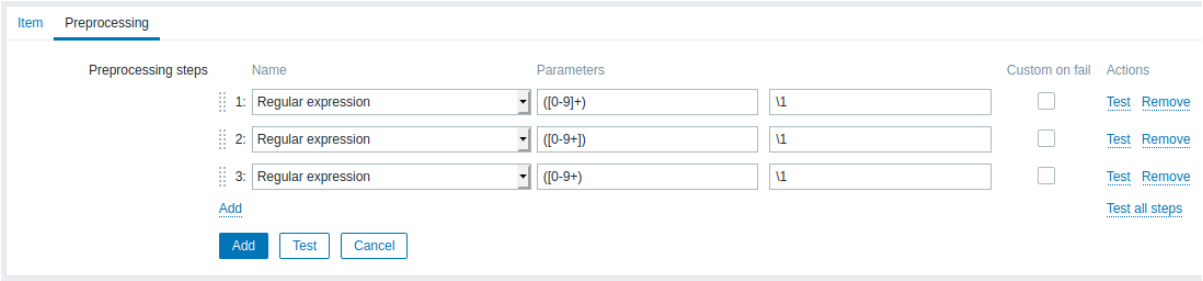
Attention:
对于更改和限制预处理步骤，需要使用以前的值来计算/比较新值。以前的值由预处理管理器处理，预处理步骤配置在进行更改或Zabbix server/proxy 重新启动时重置。由于先前的值重置:

- 对于简单改变, 每秒该表步骤-下一个值将被忽略，因为没有先前的值来计算更改;
- 对于丢弃没有改变的数据，带心跳检查丢弃不变化的数据步骤 - 下一个值永远不会被丢弃, 即使它应该因为丢弃规则而被丢弃，也不会丢弃。

Note:
如果你使用一个自定义计算改变每秒监控项信息的类型，将监控项存储类型设置为数字 *(无符号)* 但计算值实际上是一个浮点数, 计算值将通过削减小数部分，存储为整数的值。

测试

测试预处理步骤有助于确保复杂的预处理管道产生预期的结果，而无需等待项目值被接收和预处理。

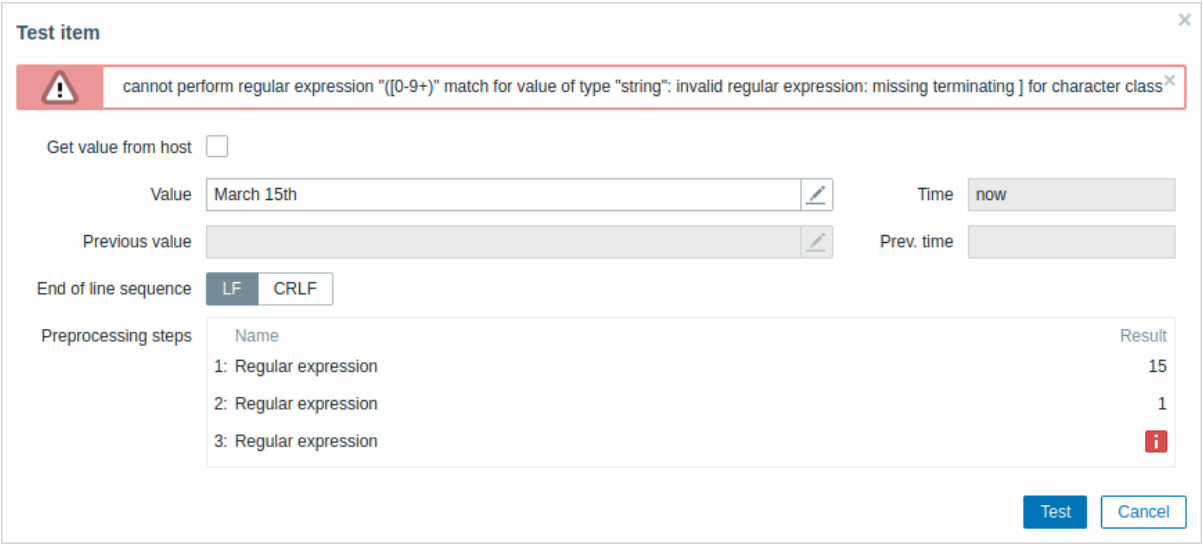


可以测试：

- 与假设值相比
- 与主机的实际值相比较

每个预处理步骤可以单独测试，也可以一起测试所有步骤。当您分别在动作块中单击// 测试 或 Test all steps //按钮时，将打开一个测试窗口。

测试假设值



参数	描
Get value from host	如果要测试假设值，请将此复选框保留为未标记。另请参见: 测试实际值.

参数	描
Value	输入要测试的输入值。单击参数字段或查看/编辑按钮将打开一个文本区域窗口，用于输入值或代码。
Time	显示输入值的时间：now (只读).

参数	描
Previous value	输入要比较的前一个输入值。仅适用于简单更改和 *Throttling* 预处理步骤。

参数	描
Previous time	输入之前要比较的输入时间。仅适用于简单更改和 *Throttling* 预处理步骤。默认值基于项的“Update interval”字段值(如果为“1m”，则此字段用“now-1m”填充)。如果未指定任何内容

参数	描
Macros	如果使用任何宏，它们将与其值一起列出。这些值是可编辑的，用于测试目的，但更改将仅保存在测试上下文中。

参数	描
End of line sequence	为多行输入值选择行尾序列：**LF** - LF (换行) 序列 **CRLF** - CRLF (回车换行) 序列。

参数描	
Preprocessing steps	将列出预处理步骤；单击 *Test* 按钮后，将显示每个步骤的测试结果。如果步骤在测试中失败，将显示错误图标。错误描述显示在鼠标上。如果为该步骤指定了“失败时

参数	描
Result	当所有步骤一起测试时(单击 *Test all steps* 按钮)，所有情况下都会显示测试预处理步骤的最终结果。还将显示监控项的值转换类型，例如“结果转换为数字(unsigned)”。

参数描

单击 *Test* 查看每个预处理步骤后的结果。

测试值存储在单个步骤或所有步骤的测试会话之间，允许用户更改预处理步骤或项目配置，然后返回到测试窗口，而无需重新输入信息。但是，值在页面刷新时丢失。

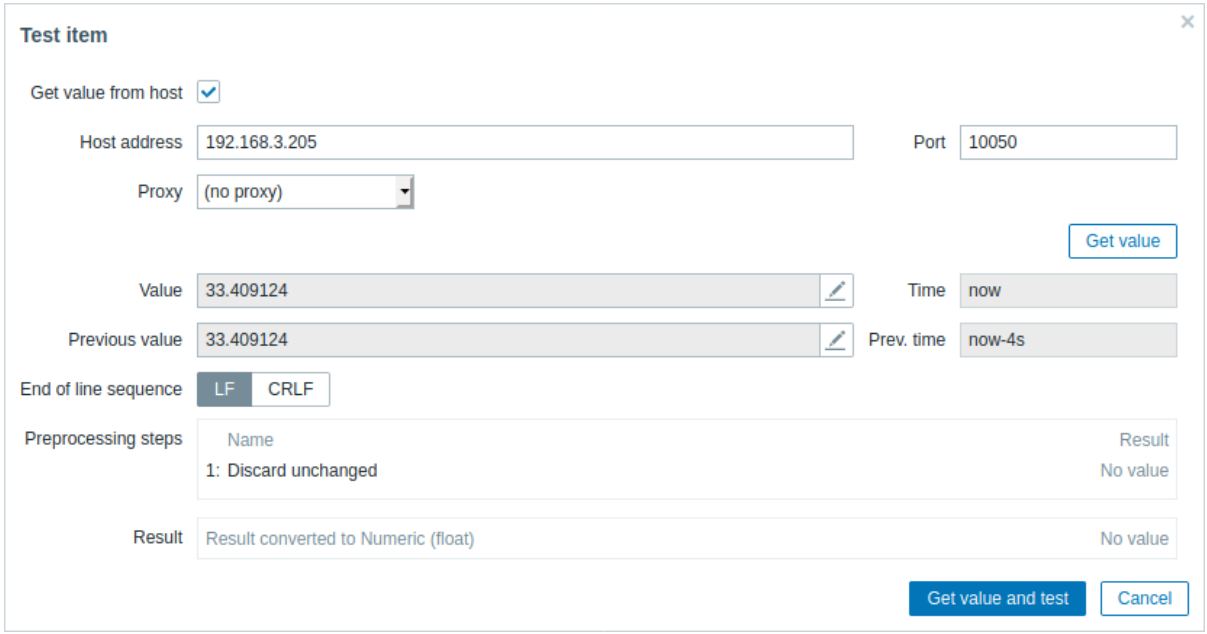
该测试由 Zabbix 服务器完成。前端将相应的请求发送到服务器，然后等待结果。请求包含输入值和预处理步骤（使用扩展的用户宏）。对于简单更改和 *Throttling* 步骤，可以指定可选的上一个值和时间。服务器响应每个预处理步骤的结果。

所有技术错误或输入验证错误均显示在测试窗口顶部的错误框中。

测试真实值

要根据实际值测试预处理，请执行以下操作：

- 标记从主机获取值复选框
- 输入或验证主机参数（主机地址、端口、代理名称/无代理）。这些字段是上下文感知的：
  - 在可能的情况下（例如，对于需要代理的监控项）通过从主机的选定代理接口获取信息来预先填充这些值
  - 必须手动填写模板项目的值
  - 如果在项目类型的上下文中不需要该字段，则该字段将被禁用（例如，主机地址字段对于计算项目和聚合项目将被禁用，代理字段对于计算项目将被禁用）
- 单击 *Get value and test* 以测试预处理



如果在项目配置窗体（“显示值” 字段）中指定了值映射，则监控项测试对话框将在最终结果之后显示另一行，名为 “已应用值映射的结果”。

具体从主机获取实际值的参数：

参数描	
Get value from host	标记此复选框以从主机获取实际值。
Host address	输入主机地址。 此字段由项目主机接口的地址自动填充。
Port	输入主机端口。 此字段由项目主机界面的端口自动填充。
Proxy	如果主机受代理监视，请指定代理。 该字段由主机的代理自动填充（如果有）。

对于更多的参数, 请看测试假设值 上面.

1 使用举例

概述

本节给出使用预处理步骤来完成一些实际任务的示例。

过滤 VMware 事件日志记录

使用正则表达式预处理过滤 VMWare 事件日志中不必要的事件。

1. 在正常工作的 VMWare 虚拟化环境主机上，检查事件日志项 `vmware.eventlog[<url>,<mode>]` 是否存在，并且工作正常。注意，如果在创建主机期间链接了 *Template VM VMWare* Template，则事件日志项可能已经存在于 hypervisor 上。

2. 在 VMWare Hypervisor 主机上创建一个 “Log” 类型的依赖监控项 ，并将该事件日志项设置为其主监控项。

在依赖项的 “预处理” 选项卡中选择 “Matches regular expression” 选项并设置参数，例如:

```
".* logged in .*" - 过滤事件日志中的所有日志事件
"\bUser\s+\K\S+"  -  只筛选事件日志中带有用户名的行
```

Attention:
如果正则表达式不匹配，则依赖项将不受支持，并给出相应的错误消息。为了避免这种情况，请标记 “Custom on fail” 复选框，并选择丢弃不匹配的值，例如：

另一种允许使用匹配组和输出控制的方法是在 “预处理” 选项卡中选择 “Regular expression” 选项并设置参数，例如:

```
pattern: ".*logged in.*", output: "\0"   - 过滤事件日志中的每行日志事件
pattern "User (.*?)(?=\ )", output: "\1" - 仅截取事件日志中筛选用户名信息
```

2 预处理详情

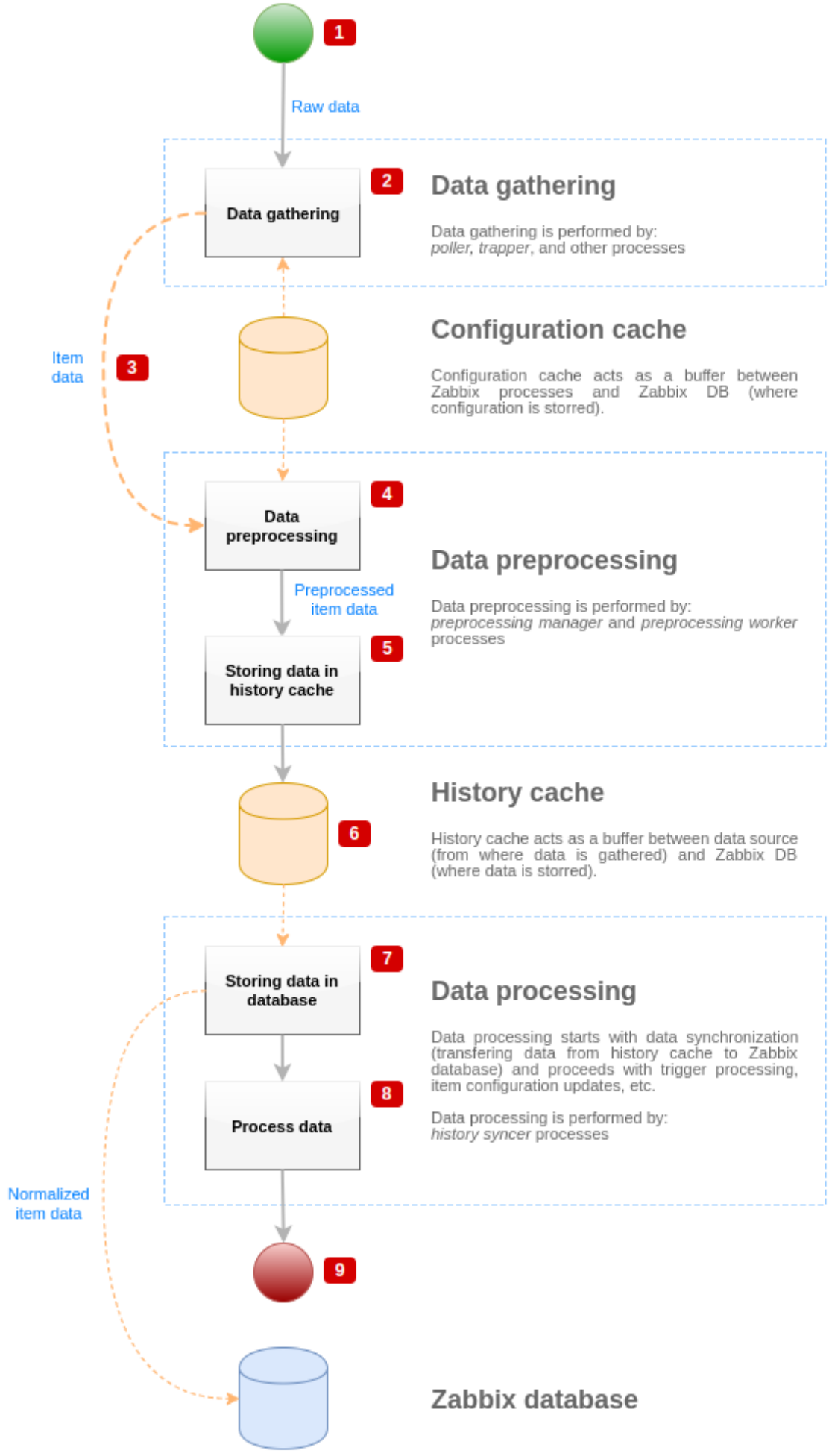
概述

本节提供监控项值预处理的详细信息。项值预处理允许为接收到的项值定义并执行转换规则。

预处理是由 Preprocessing manager 进程管理的，它是在 Zabbix 3.4 中添加的，以及执行预处理步骤的预处理工作器。来自不同数据收集器的所有值 (带或不带预处理) 在添加到历史缓存之前都要经过预处理管理器。数据收集器 (pollers, trappers 等) 和预处理过程之间使用基于套接字的 IPC 通信。Zabbix server 或 Zabbix proxy(用于由代理监视的项目) 正在执行预处理步骤。

监控项值预处理

为了可视化从数据源到 Zabbix 数据库的数据流，我们可以使用以下简化图:


1
Raw data
Data gathering
2
Data gathering
Data gathering is performed by:
poller, trapper, and other processes
Item data
3
Configuration cache
Configuration cache acts as a buffer between Zabbix processes and Zabbix DB (where configuration is storred).
Data preprocessing
4
Data preprocessing
Data preprocessing is performed by:
preprocessing manager and preprocessing worker processes
Preprocessed item data
Storing data in history cache
5
6
History cache
History cache acts as a buffer between data source (from where data is gathered) and Zabbix DB (where data is storred).
Storing data in database
7
Process data
8
Data processing
Data processing starts with data synchronization (transfering data from history cache to Zabbix database) and proceeds with trigger processing, item configuration updates, etc.
Data processing is performed by:
history syncer processes
Normalized item data
9
Zabbix database

上面的图表仅以一种简化的形式显示了监控项值预处理相关的过程、对象和动作。该图不显示条件方向更改、错误处理或循环。预处理管理进程的本地数据缓存也不显示，因为它不会直接影响数据流。这张图的目的是展示项目价值处理的过程以及它们相互作用的方式。

- 数据收集从数据源的原始数据开始。此时，数据只包含 ID、时间戳和值 (也可以是多个值)
- 无论使用哪种类型的数据采集者, 概念都是相同与主动或被动检查、捕获器监控项等, 因为它只改变了数据格式和通信起动器 (数据采集者是等待一个连接和数据, 或数据采集者发起通信和请求数据)。验证原始数据，从配置缓存中检索项配置 (使用配置数据丰富数据)。
- 基于套接字的 IPC 机制用于将数据从数据收集器传递到预处理管理器。此时，数据收集器继续收集数据，而不等待预处理管理器的响应。
- 进行数据预处理。这包括执行预处理步骤和相关项处理。

Note:
如果任何预处理步骤失败，则在执行预处理时，项可以将其状态更改为不支持。

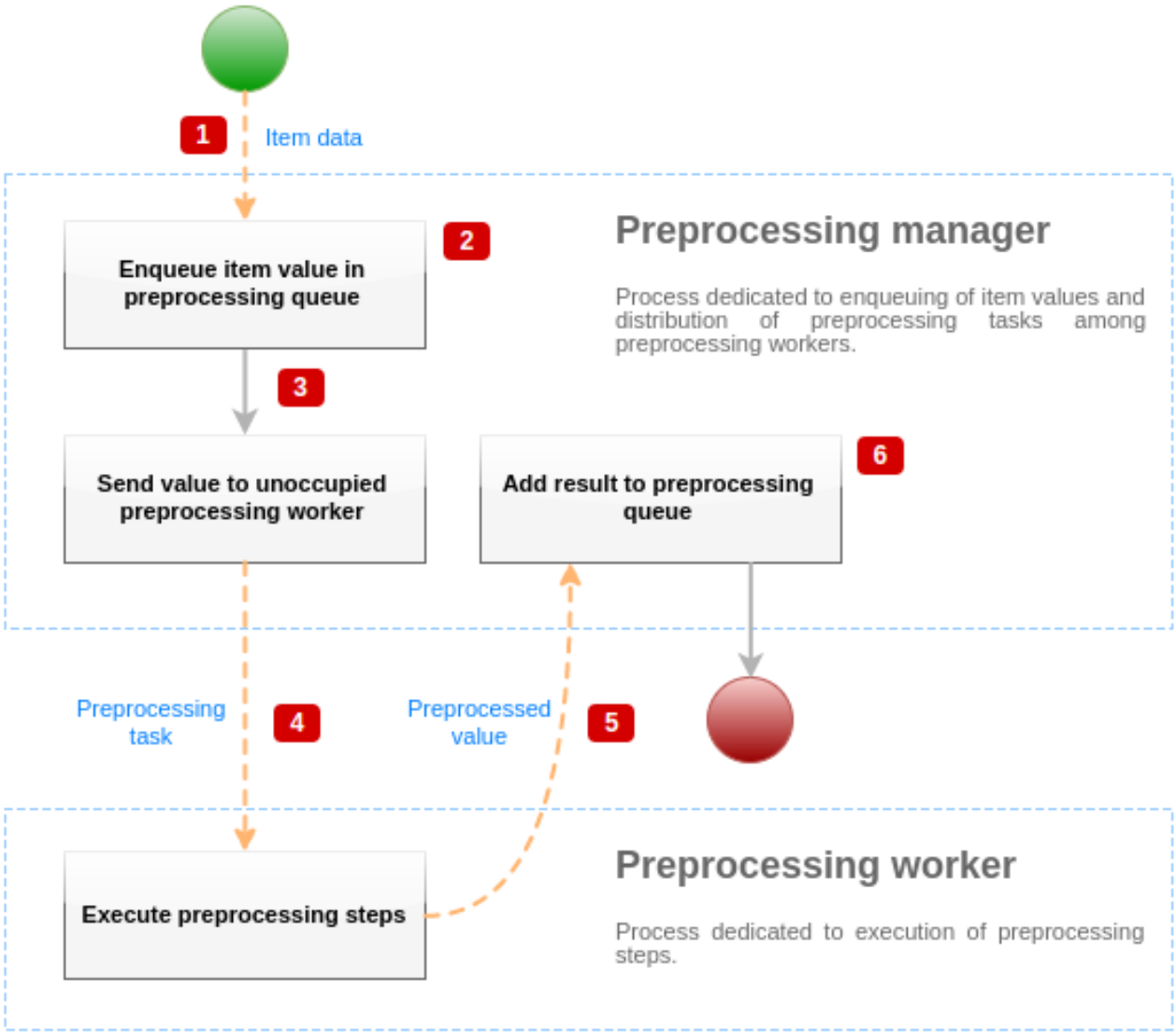
- 预处理管理器将本地数据缓存中的历史数据正在刷新到历史缓存中。
- 此时，数据流将停止，直到历史缓存的下一次同步 (当历史同步器进程执行数据同步时)。
- 同步过程从数据规范化开始，将数据存储在 Zabbix 数据库中。数据规范化执行到所需的项目类型 (在项目配置中定义的类型) 的转换，包括基于这些类型允许的预定义大小 (HISTORY_STR_VALUE_LEN 表示字符串，HISTORY_TEXT_VALUE_LEN 表示文本，HISTORY_LOG_VALUE_LEN 表示日志值) 截断文本数据。数据在标准化完成后被发送到 Zabbix 数据库。

Note:
如果数据规范化失败 (例如，当文本值不能转换为数字时)，监控项将其状态更改为不支持。

- 正在处理收集的数据—检查触发器，如果项目变得不支持，则更新项目配置，等等。
- 从监控项值处理的角度来看，这被认为是数据流的结束。

监控项值预处理

为了可视化数据预处理过程，我们可以使用以下简化图:



上面的图表仅以简化的形式显示了监控项值预处理相关的过程、对象和主要动作。该图不显示条件方向更改、错误处理或循环。图中只显示了一个预处理进程 (在实际场景中可以使用多个预处理进程)，只处理一个监控项值，我们假设该监控项需要执行至少一个预处理步骤。这个图的目的是展示监控项值预处理后端调用流程图。

- 监控数据和监控项值使用基于套接字的 IPC 机制传递给预处理管理器。
- 监控项被放在预处理队列中。

Note:
监控项可以放在预处理队列的末尾或开头。Zabbix 内部监控项总是放置在预处理队列的开头，而其他类型的项则在最后进入队列。

- 此时，数据流将停止，直到至少有一个未被占用 (即没有执行任何任务) 的预处理进程为止。
- 当预处理进程可用时，将向其发送预处理任务。
- 在预处理完成之后（预处理步骤的失败和成功执行），预处理值被传递回预处理管理器。
- 预处理管理器将结果转换为所需格式（由项值类型定义），并将结果放入预处理队列。如果当前监控项有依赖项，则依赖监控项也将添加到预处理队列中。依赖项在主监控项后面的预处理队列中排队，但仅适用于设置了值且不处于不支持状态的主监控项。

监控项值处理流水线

监控项值处理由多个进程在多个步骤 (或阶段) 中执行。这可能会导致:

- 依赖监控项可以接收值，而主项不能。这可以通过以下用例实现:

* 主监控项的值类型为“UINT”（可以使用Zabbix采集器监控项），依赖监控项的值类型为“TEXT”。
* 主监控项和依赖监控项都不需要预处理步骤。
* 文本值（如“abc”）应传递给主监控项。
* 由于没有要执行的预处理步骤，预处理管理器将检查主监控项是否处于不受支持的状态，以及是否设置了值（两者
* 当主监控项和依赖监控项都达到历史同步阶段时，由于值转换错误（文本数据不能转换为无符号整数），主监控项

结果，依赖监控项接收一个值，而主监控项将其状态更改为不支持。

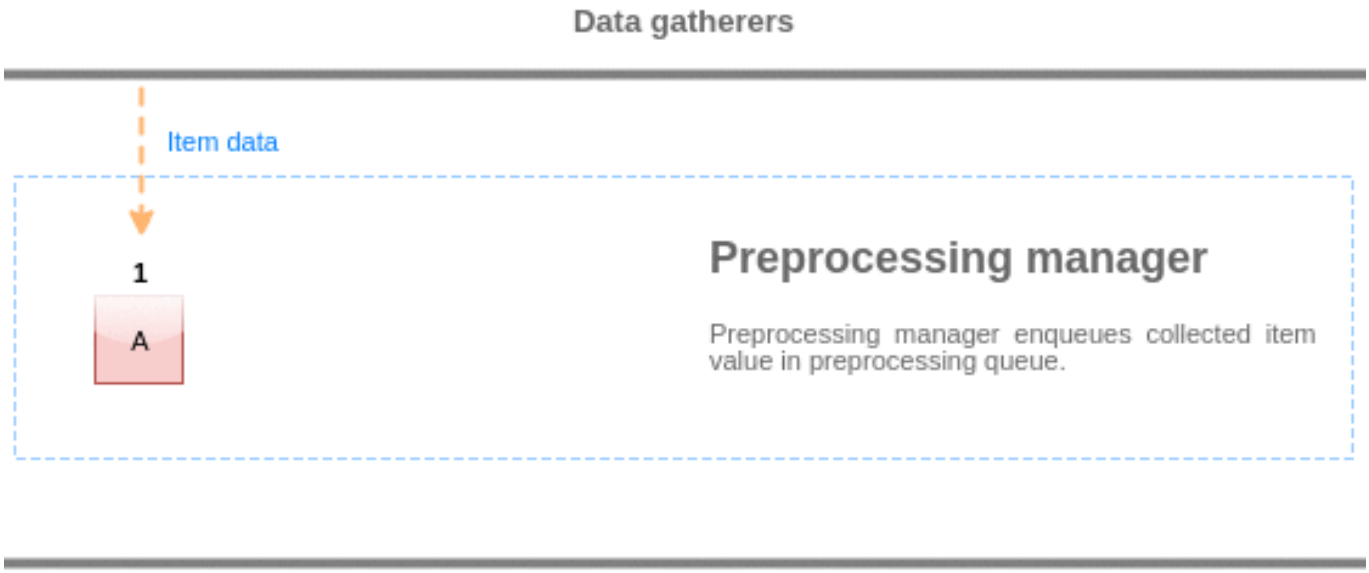
- 依赖监控项接收主监控项历史记录中不存在的值。除了主监控项类型之外，用例与前一个非常相似。例如，如果 “CHAR” 类型用于主监控项，则主监控项值将在历史同步阶段被截断，而依赖项将从主项的初始值 (未被截断的) 中接收它们的值。

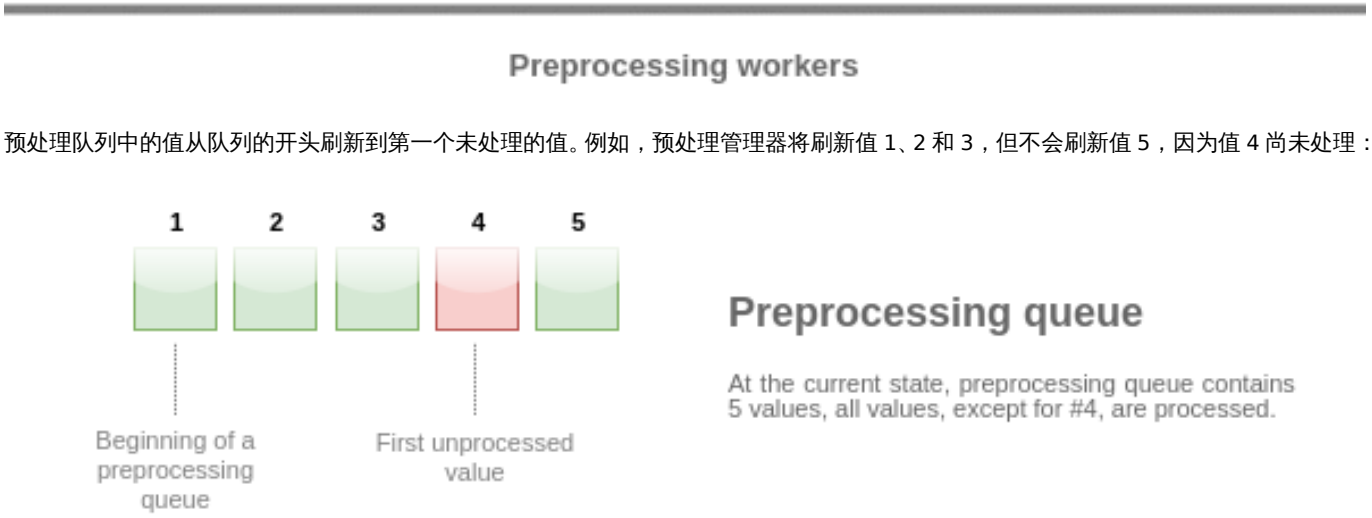
预处理队列

预处理队列是一种 FIFO 数据结构，用于存储值，以保留预处理管理器对值进行重新排序的顺序。FIFO 逻辑有多个例外：

- 内部项在队列的开头排队
- 依赖监控项总是排在主监控项之后

为了可视化预处理队列的逻辑，我们可以使用下图：





预处理队列中的值从队列的开头刷新到第一个未处理的值。例如，预处理管理器将刷新值 1、2 和 3，但不会刷新值 5，因为值 4 尚未处理：

刷新后，队列（4 和 5）中只剩下两个值，值被添加到预处理管理器的本地数据缓存中，然后值从本地缓存传输到历史缓存中。预处理管理器可以以单项模式或批量模式（用于批量接收的依赖项和值）刷新本地数据缓存中的值。

预处理进程

Zabbix 服务器配置文件允许用户设置预处理工作进程的数量。应该使用 StartPreprocessors 配置参数来设置预处理工作程序的预分支实例数。可以由许多因素确定最佳的预处理人员数量，包括 “可预处理” 项目的数量（需要执行任何预处理步骤的项目），数据收集过程的数量，项目预处理的平均步骤数量等。

但是，假设没有大体量 XML/JSON 格式数据解析这样的繁重的预处理操作，则预处理工作程序的数量可以匹配数据收集器的总数。这样，大多数情况下（收集器中的数据成批散布的情况除外）至少要有一个空闲的预处理器来收集数据。

<note warning> 太多的数据收集进程（轮询程序，不支持检查轮询程序，HTTP 轮询程序，Java 轮询程序，pinger 轮询程序，Zabbix 采集器轮询程序，proxy 轮询程序）与 IPMI 管理器，SNMP trap 程序和预处理器一起会耗尽预处理管理器的按进程文件描述符限制。这将导致 Zabbix 服务器停止（通常在启动后不久，但有时可能需要更多时间）。为了避免这种情况，应修改配置文件或提高限制。:::

3 JSONPath 功能

概述

本节详细介绍监控项值的预处理步骤中如何使用 JSONPath 功能。

JSONPath 是由点分隔的段，每段可以是一个简单的单词，如 JSON key 的名称，星号 * 或者也可以是括在 [] 中的更复杂的结构。括号段之前的分隔点是可选的，可以省略。例如：

路径描	
`$.object.name`	返回 object.name 的值
`$.object['name']`	返回 object.name 的值
`$.object.['name']`	返回 object.name 的值
`$["object"]['name']`	返回 object.name 的值
`$.['object'].["name"]`	返回 object.name 的值
`$.object.history.length()`	返回数组 object.history 的长度

路径	描
$[?(@.name == 'Object')].price.first()	返回名称为'Object'的第一个对象的price字段
$[?(@.name == 'Object')].history.first().length()	返回名称为'Object'的第一个对象history字段数组的长度
$[?(@.price > 10)].length()	返回price字段大于10对象数量

参考: 转义 JSONPath 中的 LLD 宏值中的特殊字符.

支持的格式

格式	描
`<name>`	通过 name 匹配对象属性
`*`	匹配所有对象属性
`['<name>']`	通过 name 匹配对象属性
`['<name>', '<name>', ...]`	通过列出的 name 列表，匹配对象属性
`[<index>]`	通过索引匹配数组元素
`[<number>, <number>, ...]`	通过索引列表匹配数组元素

格式描	
[*]	匹配所有对象属性或数组元素

格式描	
[<start>:<end>]	通过定义范围来匹配数组元素: **<start>** - 要匹配的第一个索引(包括)。如果未指定，则从头开始匹配所有数组元素。如果为负，则指定从数组末尾开始的偏移量。 **<end>** - 最后要

格式描	
[?(<expression>)]	通过应用过滤器表达式来匹配对象/数组元素

To find a matching segment ignoring its ancestry (detached segment) it must be prefixed with '..', 例如 $..name 或者 $..['name'] 返回所有 name 属性的值

可以通过向 JSONPath 添加"~" 后缀来提取匹配的元素名称。它返回匹配对象的名称或匹配数组项的字符串格式的索引。输出格式遵循与其他 JSONPath 查询相同的规则 - 以"as is" 形式返回确定路径结果，并以数组形式返回不确定路径结果。但是，提取与特定路径匹配的元素的名称没有意义。

筛选表达式

筛选表达式是固定符号的一种表达式。

支持的操作：

操作描	例子	
"<text>"	Text 变量. '	alue: \'1\"
'<text>'		"value: '1'"
<number>	支持科学计数法的变量 123	
<jsonpath starting with $>	JSONPath 从输入文档根节点引用的值; 仅支持确定路径 $.object.name	
<jsonpath starting with @>	JSONPath 从当前对象/元素引用的值; 仅支持确定路径。@.name	

支持的运算符:

运算符类型	描述	结果	
-	binary	减法数	
+	binary	加法数	
/	binary	除法数	
*	binary	乘法数	
==	binary	等于布	值 (1 or 0)
!=	binary	不等于布尔	(1 or 0)
	binary	小于布	值 (1 or 0)
<=	binary	小于或等于布尔值	1 or 0)
>	binary	大于布	值 (1 or 0)
>=	binary	大于或等于布尔值	1 or 0)
=~	binary	匹配正则布尔值	(1 or 0)
!	unary	布尔值非布尔值	(1 or 0)
||	binary	布尔值或布尔值	(1 or 0)
&&	binary	布尔值且布尔值	(1 or 0)

函数

可以在 JSONPath 的末尾使用函数。如果前一个函数返回后一个函数接受的值，则可以链接多个函数。支持的函数:

函数描	输入	输出
avg	输入数组中数字的平均值数字数组数字	
min	输入数组中数字的最小值数字数组数字	
max	输入数组中数字的最大值数字数组数字	
sum	输入数组中的数字总和数字数组数字	
length	输入数组中的元素数数组	数字
first	第一个数组元素数组	取决于输入数组内　的 JSON 构造（对象，数组，值）

JSONPath 聚合函数接受带引号的数字值。这意味着如果需要聚合，则将值从字符串类型转换为数字。输入不兼容将导致函数产生错误。

输出值

JSONPath 可以分为确定路径和不确定路径。确定定路径只能返回 null 或单个匹配项。不确定路径可以返回多个匹配项，基本上是具有分离的 JSONPath，多个名称/索引列表，数组切片或表达式段。但是，使用函数时，JSONPath 变得确定，因为函数始终输出单个值。确定路径返回其引用的对象/数组/值，而不确定路径返回匹配的对象/数组/值的数组。

空值

空值 (空格, tab 字符) 可以在括号符号段和表达式中自由使用，例如，`$[ 'a' ][ 0 ][ ?( $.b == 'c' ) ][ : -1 ].first( )`。

字符串

字符串应该用单引号’ 或双引号” 括起来。在字符串中，单引号或双引号 (取决于用于将其括起来的引号) 和反斜杠 “\” 用反斜杠\字符进行转义。

例子

输入数据

```
{
  "books": [
    {
      "category": "reference",
      "author": "Nigel Rees",
      "title": "Sayings of the Century",
      "price": 8.95,
      "id": 1
    },
    {
      "category": "fiction",
      "author": "Evelyn Waugh",
      "title": "Sword of Honour",
      "price": 12.99,
      "id": 2
    },
    {
      "category": "fiction",
      "author": "Herman Melville",
      "title": "Moby Dick",
      "isbn": "0-553-21311-3",
      "price": 8.99,
      "id": 3
    },
    {
      "category": "fiction",
      "author": "J. R. R. Tolkien",
      "title": "The Lord of the Rings",
      "isbn": "0-395-19395-8",
      "price": 22.99,
      "id": 4
    }
  ],
  "services": {
    "delivery": {
      "servicegroup": 1000,
```

```
      "description": "Next day delivery in local town",
      "active": true,
      "price": 5
    },
    "bookbinding": {
      "servicegroup": 1001,
      "description": "Printing and assembling book in A5 format",
      "active": true,
      "price": 154.99
    },
    "restoration": {
      "servicegroup": 1002,
      "description": "Various restoration methods",
      "active": false,
      "methods": [
        {
          "description": "Checmical cleaning",
          "price": 46
        },
        {
          "description": "Pressing pages damaged by moisture",
          "price": 24.5
        },
        {
          "description": "Rebinding torn book",
          "price": 99.49
        }
      ]
    }
  },
  "filters": {
    "price": 10,
    "category": "fiction",
    "no filters": "no \"filters\""
  },
  "closed message": "Store is closed",
  "tags": [
    "a",
    "b",
    "c",
    "d",
    "e"
  ]
}
```

JSONPath	类型结	注释
`$.filters.price`	明确的 10	
`$.filters.category`	明确的 fi	tion
`$.filters['no filters']`	明确的 no	”filters”
`$.filters`	明确的 {	<”price”: 10, ”category”: ”fiction”, ”no filters”: ”no \”filters\”” }
`$.books[1].title`	明确的 Sw	rd of Honour
`$.books[-1].author`	明确的 J.	R. R. Tolkien
`$.books.length()`	明确的 4	
`$.tags[:]`	不确定的 [”	”, ”b”, ”c”, ”d”, ”e”]
`$.tags[2:]`	不确定的 [”	”, ”d”, ”e”]
`$.tags[:3]`	不确定的 [”	”, ”b”, ”c”]
`$.tags[1:4]`	不确定的 [”	”, ”c”, ”d”]

JSONPath	类型结	注释	
`$.tags[-2:]`	不确定的 [”	”, ”e”]	
`$.tags[:-3]`	不确定的 [”	”, ”b”]	
`$.tags[:-3].length()`	不确定的 2		
`$.books[0, 2].title`	不确定的 [”	ayings of the Century”, ”Moby Dick”]	
`$.books[1]['author', "title"]`	不确定的 [”	velyn Waugh”, ”Sword of Honour”]	
`$..id`	不确定的 [1	2, 3, 4]	
`$.services..price`	不确定的 [5	154.99, 46, 24.5, 99.49]	
`$.books[?(@.id == 4 - 0.4 * 5)].title`	不确定的 [”	word of Honour”] 这个查	表明可以在查询中使用算术操作。当然，这个查询可以简化为 `$.books[?(@.i == 2)].title`
`$.books[?(@.id == 2 || @.id == 4)].title`	不确定的 [”	word of Honour”, ”The Lord of the Rings”]	
`$.books[?(!(@.id == 2))].title`	不确定的 [”	ayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]	
`$.books[?(@.id != 2)].title`	不确定的 [”	ayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]	
`$.books[?(@.title =~ " of ")].title`	不确定的 [”	ayings of the Century”, ”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(@.price > 12.99)].title`	不确定的 [”	he Lord of the Rings”]	
`$.books[?(@.author > "Herman Melville")].title`	不确定的 [”	ayings of the Century”, ”The Lord of the Rings”]	
`$.books[?(@.price > $.filters.price)].title`	不确定的 [”	word of Honour”, ”The Lord of the Rings”]	

JSONPath	类型结	注释
`$.books[?(@.category == $.filters.category)].title`	不确定的 [”	word of Honour”,”Moby Dick”,”The Lord of the Rings”]
`$..[?(@.id)]`	不确定的 [	<{ ”category”: ”reference”, ”author”: ”Nigel Rees”, ”title”: ”Sayings of the Century”, ”price”: 8.95, ”id”: 1 }, { ”category”: ”fiction”, ”author”: ”Evelyn Waugh”, ”title”: ”Sword of Honour”, ”price”: 12.99, ”id”: 2 }, { ”category”: ”fiction”, ”author”: ”Herman Melville”, ”title”: ”Moby Dick”, ”isbn”: ”0-553-21311-3”, ”price”: 8.99, ”id”: 3 }, { ”category”: ”fiction”, ”author”: ”J. R. R. Tolkien”, ”title”: ”The Lord of the Rings”, ”isbn”: ”0-395-19395-8”, ”price”: 22.99, ”id”: 4 }]
`$.services..[?(@.price > 50)].description`	不确定的’[	Printing and assembling book in A5 format”, ”Rebinding torn book”]
`$..id.length()`	明确的 4	
`$.books[?(@.id == 2)].title.first()`	明确的 Sw	rd of Honour

JSONPath	类型结	注释
`$..tags.first().length()`	明确的 5	$. tags is in- def- i- nite path, so it re- turns an ar- ray of matched ele- ments - [[”a”, ”b”, ”c”, ”d”, ”e”]], first() re- turns the first ele- ment - [”a”, ”b”, ”c”, ”d”, ”e”] and fi- nally length() cal- cu- lates its length - 5.
`$.books[*].price.min()`	明确的 8.	5
`$..price.max()`	明确的 15	.99
`$.books[?(@.category == "fiction")].price.avg()`	明确的 14	99

JSONPath	类型结	注释	
$.books[?(@.category == $.filters.xyz)].title	不确定的	对于确	路径和不确定路径，不匹配的查询返回 NULL
$.services[?(@.active=="true")].servicegroup	不确定的	00,1001] 文本常	必须在布尔值比较中使用
$.services[?(@.active=="false")].servicegroup	不确定的	02] 文本常	必须在布尔值比较中使用
$.services[?(@.servicegroup=="1002")]~.first()	明确的	toration	

转义 JSONPath 中的 LLD 宏值中的特殊字符

当 JSONPath 预处理中使用低级发现宏并解析它们的值时，将应用以下特殊字符转义规则:

- 只考虑转义反斜杠 (\) 和双引号 (") 字符;
- 如果解析的宏值包含这些字符，每个字符都用反斜杠转义;
- 如果解析的宏值包含这些字符，则每个字符都用反斜杠进行转义;

*如果它们已经用反斜杠转义，则不认为是转义，并且反斜杠和以下特殊字符将再次转义。
*

例子:

JSONPath	LLD 宏值替	后
$.[?(@.value == "{#MACRO}")]	special "value"	$.[?(@.value == "special \"value\"")]

JSONPath	LLD 宏值替	后
	c:\temp	$.[?(@.value == ”c:\\temp”)]
	a\\b	$.[?(@.value == ”a\\\\b”)]

在表达式中使用时，可能有特殊字符的宏应该用双引号括起来:

JSONPath	LLD 宏值替	后结果	
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]	OK
$.[?(@.value == {#MACRO})]		$.[?(@.value == special \”value\”)]	**Bad JSONPath expression**

当在路径中使用宏时，可能有特殊字符的宏应该用方括号和双引号括起来:

JSONPath	LLD 宏值替	后结果	
$.[”{#MACRO}”].value	c:\temp	$.[”c:\\temp”].value	OK
$.{#MACRO}.value		$.c:\\temp.value	**Bad JSONPath expression**

4 JavaScript 预处理

概览

本节详细描述通过 JavaScript 进行预处理。

JavaScript 预处理

JavaScript 预处理是通过调用带有单个参数值和用户提供的函数体的 JavaScript 函数来完成的。预处理步骤的结果是从这个函数返回的值，例如，要执行华氏到摄氏度的转换，用户必须输入:

```
return (value - 32) * 5 / 9
```

在 JavaScript 的预处理参数中，这些参数将被 server 封装到 JavaScript 函数中:

```
function (value)
{
   return (value - 32) * 5 / 9
}
```

输入参数值总是作为字符串传递。返回值通过 ToString() 方法自动强制转换为字符串 (如果失败，则返回错误为字符串值)，但有几个例外:

- 返回未定义的值将导致错误
- 返回 null 值将导致输入值被丢弃，很像在 “Custom on fail” 操作上的 “Discard value” 预处理。

可以通过抛出值/对象 (通常是字符串或错误对象) 返回错误。例子:

```
if (value == 0)
    throw "Zero input value"
return 1/value
```

每个脚本都有一个 10 秒的执行超时 (取决于脚本，超时触发的时间可能更长); 超过该值将返回错误。此外，还强制执行 10 兆字节的堆限制。JavaScript 预处理步骤字节码将在下次应用该步骤时缓存并重用。对该项的预处理步骤的任何更改将导致缓存的脚本在稍后被重置和重新编译。连续的运行失败 (连续 3 次) 将导致引擎被重新初始化，以减少一个脚本破坏下一个脚本的执行环境的可能性 (DebugLevel 4 或更高级别的日志记录此操作)。JavaScript 预处理是用 Duktape (https://duktape.org/) JavaScript 引擎实现的。

在脚本中使用宏

在 JavaScript 代码中使用用户宏是可能的。如果一个脚本包含用户宏，这些宏将在执行特定的预处理步骤之前由服务器/代理解析。注意，当在前端测试预处理步骤时，宏值不会被提取，需要手动输入。

> **Note:**
> 当宏被它的值替换时，Context 被忽略。宏值按原样插入到代码中，在将值放入 JavaScript 代码之前不可能添加额外的转义。请注意，在某些情况下，这可能会导致 JavaScript 错误。

在下面的示例中，如果接收到的值超过了 {$THRESHOLD} 的宏值，则返回该宏值 (如果存在):

```
var threshold = '{$THRESHOLD}';
return (!isNaN(threshold) && value > threshold) ? threshold : value;
```

全局 JavaScript 函数

其他的全局 JavaScript 函数已经用 Duktape 实现:

- btoa(string) - 将字符串编码为 base64 字符串
- atob(base64_string) - 解码 base64 字符串

```
try {
    b64 = btoa("utf8 string");
    utf8 = atob(b64);
}
catch (error) {
    return {'error.name' : error.name, 'error.message' : error.message}
}
```

参考: 额外的 JavaScript 对象

额外的 JavaScript 对象

概览

本节描述 Zabbix 添加到用 Duktape 实现的 JavaScript 语言中。

内置对象

Zabbix

Zabbix 对象提供了与内部 Zabbix 功能的交互。

方法描
Log(级别，消息）使用 &　t;loglevel> 日志级别 (参见配置文件 DebugLevel 参数) 将 <message> 写入 Zabbix 日志。

例子:

```
Zabbix.Log(3, "this is a log entry written with 'Warning' log level")
```

CurlHttpRequest

这个对象封装了 cURL 句柄，允许进行简单的 HTTP 请求。错误被作为异常抛出。

方法	描
`AddHeader(name, value)`	添加HTTP报头字段。此字段用于所有后续请求，直到使用ClearHeader()方法清除为止

方法	描
ClearHeader()	清理HTTP头。如果没有设置报头字段，如果发送的数据是json格式的，CurlHttpRequest将设置Content-Type为application/json，否则为text/plain

方法	描
GetHeaders()	返回接收的HTTP报头字段的对象。这个方法从Zabbix 5.0.4开始就可用了
Get(url, data)	将HTTP GET请求发送到带有可选 *data* 负载的URL，并返回响应

方法	描
`Put(url, data)`	将 HTTP PUT 请求发送到带有可选 *data* 负载的 URL，并返回响应
`Post(url, data)`	将 HTTP POST 请求发送到带有可选 *data* 负载的 URL，并返回响应

方法描	
Delete(url, data)	将 HTTP DELETE 请求发送到带有可选 *data* 负载的 URL，并返回响应
Status()	返回最后一个 HTTP 请求的状态码
SetProxy(proxy)	设置 HTTP 代理为"proxy"值。如果该参数为空，则不使用代理

例子:

```
try {
    Zabbix.Log(4, 'jira webhook script value='+value);
```

```
    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new CurlHttpRequest(),
    fields = {},
    resp;

    req.AddHeader('Content-Type: application/json');
    req.AddHeader('Authorization: Basic '+params.authentication);

    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {"key": params.project_key};
    fields.issuetype = {"id": params.issue_id};
    resp = req.Post('https://tsupport.zabbix.lan/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.Status() != 201) {
        throw 'Response code: '+req.Status();
    }

    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;
} catch (error) {
    Zabbix.Log(4, 'jira issue creation failed json : '+JSON.stringify({"fields": fields}));
    Zabbix.Log(4, 'jira issue creation failed : '+error);

    result = {};
}

return JSON.stringify(result);
```

Global JavaScript functions

Additional global JavaScript functions have been implemented with Duktape:

- btoa(string) - encodes string to base64 string
- atob(base64_string) - decodes base64 string

```
try {
    b64 = btoa("utf8 string");
    utf8 = atob(b64);
}
catch (error) {
    return {'error.name' : error.name, 'error.message' : error.message}
}
```

- md5(string) - calculates the MD5 hash of a string. This function is supported since Zabbix 5.0.9
- sha256(string) - calculates the SHA256 hash of a string. This function is supported since Zabbix 5.0.9

5 CSV 转换成 JSON 预处理

概览

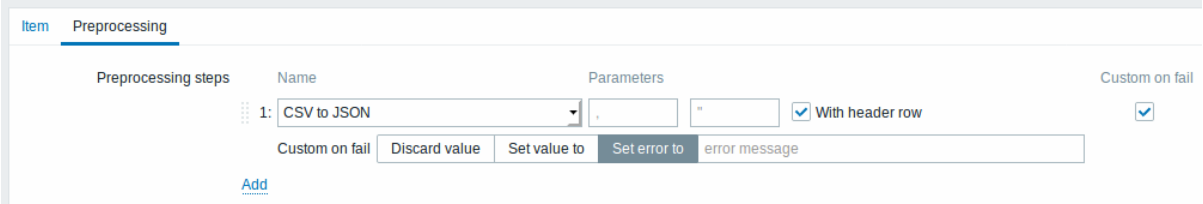
在这个预处理步骤中，可以将 CSV 文件数据转换为 JSON 格式。支持:

- items (item 原型)
- 低级服务发现规则

配置

配置 CSV 到 JSON 的预处理步骤:

- 转到 Preprocessing 选项卡item/discovery rule 配置
- 点击添加
- 选择 CSV 到 JSON 选项



第一个参数允许设置自定义分隔符。注意, 如果 CSV 输入的第一行开始”Sep=”, 紧随其后的是一个 utf-8 字符, 字符将被用作分隔符的第一个参数没有设置, 如果第一个参数没有设置分隔符并不是从”Sep=” 检索, 然后一个逗号作为分隔符。

第二个可选参数允许设置引号符号。

* 如果 *With header row* 复选框被标记，标题行值将被解释为列名 (参见header processing了解更多信息)。* 如果 *Custom on fail* 复选框被标记，那么在预处理步骤失败的情况下，该项将不会不受支持。另外，可以设置自定义错误处理选项: 丢弃该值，设置指定值或设置指定的错误消息。

头处理

CSV 文件头行可以用两种不同的方式处理:

* 如果*With header row* 复选框被标记-标题行值被解释为列名。在这种情况下，列名必须是唯一的，数据行不应该包含比标题行更多的列; * 如果*With header row* 复选框没有标记-标题行解释为数据。自动生成列名 (1,2,3,4...)

CSV 文件例子:

```
Nr,Item name,Key,Qty
1,active agent item,agent.hostname,33
"2","passive agent item","agent.version","44"
3,"active,passive agent items",agent.ping,55
```

Note:
输入中的引号字段中的引号字符必须在其前面加上另一个引号字符进行转义

处理标题行

期望有标题行时的 JSON 输出:

```
[
   {
      "Nr":"1",
      "Item name":"active agent item",
      "Key":"agent.hostname",
      "Qty":"33"
   },
   {
      "Nr":"2",
      "Item name":"passive agent item",
      "Key":"agent.version",
      "Qty":"44"
   },
   {
      "Nr":"3",
      "Item name":"active,passive agent items",
      "Key":"agent.ping",
      "Qty":"55"
   }
]
```

无标题行处理

不需要标题行时的 JSON 输出:

```
[
   {
      "1":"Nr",
      "2":"Item name",
      "3":"Key"
      "4":"Qty"
   },
   {
      "1":"1",
      "2":"active agent item",
      "3":"agent.hostname"
      "4":"33"
   },
   {
      "1":"2",
      "2":"passive agent item",
      "3":"agent.version"
      "4":"44"
   },
   {
      "1":"3",
      "2":"active,passive agent items",
      "3":"agent.ping"
      "4":"55"
   }
]
```

3 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	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。 /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". <* 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，触发器

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） **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	在 Windows 上，该选项从 64 位计数器获取值（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 *speed* 信息。	64 位接口统计计数器在 Windows Vista 和 Windows Server 2008 中引入。
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%或%TEMP%子类的环境变量是不支持的。 - 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	$HOME 和%TEMP% **regex_incl** 对硬链接计数一次。 (https://en.wikipedia.org (PCRE). regex_incl 和 regex_excl 在计算条目大小时都应用于文件和目录，但在选择遍历的子目录时会被忽略（如果 regex_incl 是 “(?i)^.+\.zip$” 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。

Key	Description	Return value	Parameters	Comments
agent.hostname	Agent host name.	String		Returns the actual value of the agent hostname from a configuration file.
agent.ping	Agent availability check.	Nothing - unavailable 1 - available		Use the **nodata()** trigger function to check for host unavailability.
agent.variant	Variant of Zabbix agent (Zabbix agent or Zabbix agent 2).	Integer		Example of returned value: 1 - Zabbix agent 2 - Zabbix agent 2 Supported since Zabbix 5.0.18.
agent.version	Version of Zabbix agent.	String		Example of returned value: 1.8.2
kernel.maxfiles	Maximum number of opened files supported by OS.	Integer		
kernel.maxproc	Maximum number of processes supported by OS.	Integer		
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>]				

Key					
	Log file monitoring.	Log	**file** - full path and name of log file **regexp** - regular expression[4] describing the required pattern **encoding** - code page identifier **maxlines** - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf **mode** (since version 2.0) - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **output** (since version 2.2) - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if	The item must be configured as an active check. If file is missing or permissions do not allow access, item turns unsupported. If `output` is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the `output` parameter is ignored. Content extraction using the `output` parameter takes place on the agent. Examples: => log[/var/log/syslog] => log[/var/log/syslog,error] => log[/home/zabbix/logs/logfile„ *Using `output` parameter for extracting a number from log record:* => log[/app1/app.log,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors"„„\1] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records,

Key
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]

Key				
	Count of matched lines in log file monitoring.	Integer	**file** - full path and name of log file **regexp** - regular expression[4] describing the required pattern **encoding** - code page identifier **maxproclines** - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf. **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **maxdelay** - maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the maxdelay notes before using it! **options** (since Zabbix 4.4.7) - additional options: *mtime-noreread* - non-unique records, reread only if the file size changes	The item must be configured as an active check. If file is missing or permissions do not allow access, item turns unsupported. See also additional information on log monitoring. This item is not supported for Windows Event Log. Supported since Zabbix 3.2.0.

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

Key				
	Log file monitoring with log rotation support.	Log	**file_regexp** - absolute path to file and the file name described by a regular expression[4]. *Note* that only the file name is a regular expression **regexp** - regular expression[4] describing the required content pattern **encoding** - code page identifier **maxlines** - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf **mode** (since version 2.0) - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **output** (since version 2.2) - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an	The item must be configured as an active check. Log rotation is based on the last modification time of files. Note that logrt is designed to work with one currently active log file, with several other matching inactive files rotated. If, for example, a directory has many active log files, a separate logrt item should be created for each one. Otherwise if one logrt item picks up too many files it may lead to exhausted memory and a crash of monitoring. If `output` is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the `output` parameter is ignored. Content extraction using the `output` parameter takes place on the agent. Examples:

Key
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]

Key				
	Count of matched lines in log file monitoring with log rotation support.	Integer	**file_regexp** - absolute path to file and regular expression[4] describing the file name pattern **regexp** - regular expression[4] describing the required content pattern **encoding** - code page identifier **maxproclines** - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf. **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **maxdelay** - maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the maxdelay notes before using it! **options** (since version 4.0; *mtime-reread*, *mtime-noreread*	The item must be configured as an active check. Log rotation is based on the last modification time of files. See also additional information on log monitoring. This item is not supported for Windows Event Log. Supported since Zabbix 3.2.0.

Key				
net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]	Checks if DNS service is up.	0 - DNS is down (server did not respond or DNS resolution failed) 1 - DNS is up	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows) - number of tries for the request (default is 2) **protocol** (since version 3.0)- the protocol used to perform DNS queries: *udp* (default) or *tcp*	Example: => net.dns[8.8.8.8,example.com, The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (except for Windows), *HINFO*, *MINFO*, *TXT*, *SRV* Internationalized domain names are not supported, please use IDNA encoded names instead. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): *net.tcp.dns*
net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]				

Key				
	Performs a DNS query.	Character string with the required type of information	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows) - number of tries for the request (default is 2) **protocol** (since version 3.0) - the protocol used to perform DNS queries: *udp* (default) or *tcp*	Example: => net.dns.record[8.8.8.8,exampl The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (except for Windows), *HINFO*, *MINFO*, *TXT*, *SRV* Internationalized domain names are not supported, please use IDNA encoded names instead. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): *net.tcp.dns.query*
net.if.collisions[if]	Number of out-of-window collisions.	Integer	**if** - network interface name	
net.if.discovery				

Key			
	List of network interfaces. Used for low-level discovery.	JSON object	Supported since Zabbix agent version 2.0. On FreeBSD, OpenBSD and NetBSD supported since Zabbix agent version 2.2. Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.
net.if.in[if,<mode>]			

Key				
	Incoming traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets *overruns (fifo)* - the number of FIFO buffer errors *frame* - the number of packet framing errors *compressed* - the number of compressed packets transmitted or received by the device driver *multicast* - the number of multicast frames received by the device driver	On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported. The network interface GUID as the first parameter on Windows is supported since Zabbix 5.0.16. Examples: => net.if.in[eth0,errors] => net.if.in[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with the *Change per second* preprocessing step in order to get bytes per second statistics.
net.if.out[if,<mode>]				

Key				
	Outgoing traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets *overruns (fifo)* - the number of FIFO buffer errors *collisions (colls)* - the number of collisions detected on the interface *carrier* - the number of carrier losses detected by the device driver *compressed* - the number of compressed packets transmitted by the device driver	On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported. The network interface GUID as the first parameter on Windows is supported since Zabbix 5.0.16. Examples: => net.if.out[eth0,errors] => net.if.out[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with the *Change per second* preprocessing step in order to get bytes per second statistics.
net.if.total[if,<mode>]				

Key				
	Sum of incoming and outgoing traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets *overruns (fifo)* - the number of FIFO buffer errors *compressed* - the number of compressed packets transmitted or received by the device driver	On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. The network interface GUID as the first parameter on Windows is supported since Zabbix 5.0.16. Examples: => net.if.total[eth0,errors] => net.if.total[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with the *Change per second* preprocessing step in order to get bytes per second statistics. Note that dropped packets are supported only if both net.if.in and net.if.out work for dropped packets on your platform.

Key				
net.tcp.listen[port]	Checks if this TCP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - TCP port number	Example: => net.tcp.listen[80] On Linux supported since Zabbix agent version 1.8.4 Since Zabbix 3.0.0, on Linux kernels 2.6.14 and above, information about listening TCP sockets is obtained from the kernel's NETLINK interface, if possible. Otherwise, the information is retrieved from /proc/net/tcp and /proc/net/tcp6 files.
net.tcp.port[<ip>,port]	Checks if it is possible to make TCP connection to specified port.	0 - cannot connect 1 - can connect	**ip** - IP or DNS name (default is 127.0.0.1) **port** - port number	Example: => net.tcp.port[,80] → can be used to test availability of web server running on port 80. For simple TCP performance testing use net.tcp.service.perf[tcp,<ip>, Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). Old naming: *check_port[*]*

Key
net.tcp.service[service,<ip>,<port>]

Key				
	Checks if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: => net.tcp.service[ftp,45] → can be used to test the availability of FTP server on TCP port 45. Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.port for checks like these. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2 (since Zabbix 5.0.3). Note that the telnet check looks for a login prompt (':' at the end). See also known issues of checking HTTPS service. *https* and *telnet* services are supported since Zabbix 2.0. Old naming: *check_service[*]*

Key				
net.tcp.service.perf[service,<ip>,<port>]	Checks performance of TCP service.	0 - service is down seconds - the number of seconds spent while connecting to the service	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: => net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server. Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.service.perf[tcp,<ip>, for checks like these. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2 (since Zabbix 5.0.3). Note that the telnet check looks for a login prompt (':' at the end). See also known issues of checking HTTPS service. *https* and *telnet* services are supported since Zabbix 2.0. Old naming: *check_service_perf[*]*
net.udp.listen[port]	Checks if this UDP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - UDP port number	Example: => net.udp.listen[68] On Linux supported since Zabbix agent version 1.8.4

Key				
net.udp.service[service,<ip>,<port>]	Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: => net.udp.service[ntp„45] → can be used to test the availability of NTP service on UDP port 45. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]	Checks performance of UDP service.	0 - service is down seconds - the number of seconds spent waiting for response from the service	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: => net.udp.service.perf[ntp] → can be used to test response time from NTP service. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]				

Key				
	Process CPU utilization percentage.	Float	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **type** - CPU utilization type: *total* (default), *user*, *system* **cmdline** - filter by command line (it is a regular expression[4]) **mode** - data gathering mode: *avg1* (default), *avg5*, *avg15* **zone** - target zone: *current* (default), *all*. This parameter is supported on Solaris only.	Examples: => proc.cpu.util[,root] → CPU utilization of all processes running under the ”root” user => proc.cpu.util[zabbix_server,za → CPU utilization of all zabbix_server processes running under the zabbix user The returned value is based on single CPU core utilization percentage. For example CPU utilization of a process fully using two cores is 200%. The process CPU utilization data is gathered by a collector which supports the maximum of 1024 unique (by name, user and command line) queries. Queries not accessed during the last 24 hours are removed from the collector. *Note* that when setting the `zone` parameter to *current* (or default) in case the agent has been compiled on a Solaris without zone support, but running on a newer Solaris where zones are supported, then the agent will return NOT-SUPPORTED (the agent

Key
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]

Key					
	Memory used by process in bytes.	Integer - with `mode` as *max*, *min*, *sum* Float - with `mode` as *avg*	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **mode** - possible values: *avg*, *max*, *min*, *sum* (default) **cmdline** - filter by command line (it is a regular expression[4]) **memtype** - type of memory used by process	Examples: => proc.mem[,root] → memory used by all processes running under the ”root” user => proc.mem[zabbix_server,zabb → memory used by all zabbix_server processes running under the zabbix user => proc.mem[,oracle,max,oracleZ → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line *Note*: When several processes use shared memory, the sum of memory used by processes may result in large, unrealistic values. See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific). When this item is invoked from the command line and contains a command line parameter (e.g. using the agent test mode: `zabbix_agentd -t proc.mem[,,,apache2]`), one extra process will be

Key
proc.num[<name>,<user>,<state>,<cmdline>,<zone>]

Key				
	The number of processes.	Integer	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **state** (*disk* and *trace* options since version 3.4.0) - possible values: *all* (default), *disk* - uninterruptible sleep, *run* - running, *sleep* - interruptible sleep, *trace* - stopped, *zomb* - zombie **cmdline** - filter by command line (it is a regular expression[4]) **zone** - target zone: *current* (default), *all*. This parameter is supported on Solaris only.	Examples: => proc.num[,mysql] → number of processes running under the mysql user => proc.num[apache2,www-data] → number of apache2 processes running under the www-data user => proc.num[,oracle,sleep,oracle → number of processes in sleep state running under oracle having oracleZABBIX in its command line See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific). On Windows, only the `name` and `user` parameters are supported. When this item is invoked from the command line and contains a command line parameter (e.g. using the agent test mode: `zabbix_agentd -t proc.num[,,,apache2]`), one extra process will be counted, as the agent will count itself. *Note* that when setting the `zone` parameter to

Key				
sensor[device,sensor,<mode>]	Hardware sensor reading.	Float	**device** - device name **sensor** - sensor name **mode** - possible values: *avg*, *max*, *min* (if this parameter is omitted, device and sensor are treated verbatim).	Reads /proc/sys/dev/sensors on Linux 2.4. Example: => sensor[w83781d-i2c-0-2d,temp1] Prior to Zabbix 1.8.4, the *sensor[temp1]* format was used. Reads /sys/class/hwmon on Linux 2.6+. See a more detailed description of sensor item on Linux. Reads the *hw.sensors* MIB on OpenBSD. Examples: => sensor[cpu0,temp0] → temperature of one CPU => sensor[”cpu[0-2]$”,temp,avg] → average temperature of the first three CPU’s Supported on OpenBSD since Zabbix 1.8.4.
system.boottime	System boot time.	Integer (Unix timestamp)		
system.cpu.discovery	List of detected CPUs/CPU cores. Used for low-level discovery.	JSON object		Supported on all platforms since 2.4.0.
system.cpu.intr	Device interrupts.	Integer		
system.cpu.load[<cpu>,<mode>]				

Key				
	CPU load.	Float	**cpu** - possible values: *all* (default), *percpu* (since version 2.0; total load divided by online CPU count) **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	Example: => system.cpu.load[,avg5] Old naming: *system.cpu.loadX*
system.cpu.num[<type>]				
	Number of CPUs.	Integer	**type** - possible values: *online* (default), *max*	Example: => system.cpu.num
system.cpu.switches				
	Count of context switches.	Integer		Old naming: *system.tem[switches]*
system.cpu.util[<cpu>,<type>,<mode>,<logical_or_physical>]				

Key				
	CPU utilization percentage.	Float	**cpu** - *<CPU number>* or *all* (default) **type** - possible values: *user* (default), *idle*, *nice*, *system* (default for Windows), *iowait*, *interrupt*, *softirq*, *steal*, *guest* (on Linux kernels 2.6.24 and above), *guest_nice* (on Linux kernels 2.6.33 and above). See also platform-specific details for this parameter. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15* **logical_or_physical** (since version 5.0.3; AIX only) - possible values: *logical* (default), *physical*.	Example: => system.cpu.util[0,user,avg5] Old naming: *system.cpu.idleX*, *system.cpu.niceX*, *system.cpu.systemX*, *system.cpu.userX*
system.hostname[<type>, <transform>]				

Key				
	System host name.	String	**type** (before version 5.0.18 supported on Windows only) - possible values: *netbios* (default on Windows), *host* (default on Linux), *shorthost* (since version 5.0.18; returns part of the hostname before the first dot, a full string for names without dots). **transform** (since version 5.0.18) - possible values: *none* (default), *lower* (convert to lowercase)	The value is acquired by either GetComputerName() (for **netbios**) or gethostname() (for **host**) functions on Windows and by "hostname" command on other systems. Examples of returned values: *on Linux*: => system.hostname → linux-w7x1 => system.hostname → example.com => system.hostname[shorthost] → example *on Windows*: => system.hostname → WIN-SERV2008-I6 => system.hostname[host] → Win-Serv2008-I6LonG => system.hostname[host,lower] → win-serv2008-i6long See also a more detailed description.
system.hw.chassis[<info>]				

Key				
	Chassis information.	String	**info** - one of full (default), model, serial, type or vendor	Example: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] This key depends on the availability of the SMBIOS table. Will try to read the DMI table from sysfs, if sysfs access fails then try reading directly from memory. **Root permissions** are required because the value is acquired by reading from sysfs or memory. Supported since Zabbix agent version 2.0.
system.hw.cpu[<cpu>,<info>]				

Key				
	CPU information.	String or integer	**cpu** - *<CPU number>* or *all* (default) **info** - possible values: *full* (default), *curfreq*, *maxfreq*, *model* or *vendor*	Example: => system.hw.cpu[0,vendor] → AuthenticAMD Gathers info from /proc/cpuinfo and /sys/devices/system/cpu/[cpur If a CPU number and *curfreq* or *maxfreq* is specified, a numeric value is returned (Hz). Supported since Zabbix agent version 2.0.
system.hw.devices[<type>]				
	Listing of PCI or USB devices.	Text	**type** - *pci* (default) or *usb*	Example: => system.hw.devices[pci] → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [..] Returns the output of either lspci or lsusb utility (executed without any parameters) Supported since Zabbix agent version 2.0.
system.hw.macaddr[<interface>,<format>]				

Key				
	Listing of MAC addresses.	String	**interface** - *all* (default) or a regular expression[4] **format** - *full* (default) or *short*	Lists MAC addresses of the interfaces whose name matches the given `interface` regular expression[4] (*all* lists for all interfaces). Example: => system.hw.macaddr[”eth0$”,full] → [eth0] 00:11:22:33:44:55 If `format` is specified as *short*, interface names and identical MAC addresses are not listed. Supported since Zabbix agent version 2.0.
system.localtime[<type>]	System time.	Integer - with `type` as *utc* String - with `type` as *local*	**type** (since version 2.0) - possible values: *utc* - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds. *local* - the time in the ’yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm’ format	Must be used as a passive check only. Example: => system.localtime[local] → create an item using this key and then use it to display host time in the *Clock* screen element.
system.run[command,<mode>]				

Key				
	Run specified command on the host.	Text result of the command 1 - with `mode` as *nowait* (regardless of command result)	**command** - command for execution **mode** - possible values: *wait* - wait end of execution (default), *nowait* - do not wait	Up to 512KB of data can be returned, including trailing whitespace that is truncated. To be processed correctly, the output of the command must be text. Example: => system.run[ls -l /] → detailed file list of root directory. *Note*: system.run items are disabled by default. Learn how to enable them. The return value of the item is standard output together with standard error produced by command. The exit code is not checked. Empty result is allowed starting with Zabbix 2.4.0. See also: Command execution.
system.stat[resource,<type>]				

Key			
	System statistics.	Integer or float	**ent** - number of processor units this partition is entitled to receive (float) **kthr,<type>** - information about kernel thread states: *r* - average number of runnable kernel threads (float) *b* - average number of kernel threads placed in the Virtual Memory Manager wait queue (float) **memory,<type>** - information about the usage of virtual and real memory: *avm* - active virtual pages (integer) *fre* - size of the free list (integer) **page,<type>** - information about page faults and paging activity: *fi* - file page-ins per second (float) *fo* - file page-outs per second (float) *pi* - pages paged in from paging space (float) *po* - pages paged out to paging space (float) *fr* - pages freed (page replacement) (float) *sr* - pages scanned by page-replacement algorithm (float) **faults,<type>** - trap and

Key				
				Comments This item is supported on AIX only, since Zabbix 1.8.1. Take note of the following limitations in these items: => system.stat[cpu,app] - supported only on AIX LPAR of type ”Shared” => system.stat[cpu,ec] - supported on AIX LPAR of type ”Shared” and ”Dedicated” (”Dedicated” always returns 100 (percent)) => system.stat[cpu,lbusy] - supported only on AIX LPAR of type ”Shared” => system.stat[cpu,pc] - supported on AIX LPAR of type ”Shared” and ”Dedicated” => system.stat[ent] - supported on AIX LPAR of type ”Shared” and ”Dedicated”
system.sw.arch	Software architecture information.	String		Example: => system.sw.arch → i686 Info is acquired from uname() function. Supported since Zabbix agent version 2.0.
system.sw.os[<info>]				

Key				
	Operating system information.	String	**info** - possible values: *full* (default), *short* or *name*	Example: => system.sw.os[short]→ Ubuntu 2.6.35-28.50-generic 2.6.35.11 Info is acquired from (note that not all files and options are present in all distributions): /proc/version (*full*) /proc/version_signature (*short*) PRETTY_NAME parameter from /etc/os-release on systems supporting it, or /etc/issue.net (*name*) Supported since Zabbix agent version 2.0.
system.sw.packages[<package>,<manager>,<format>]				

Key				
	Listing of installed packages.	Text	**package** - *all* (default) or a regular expression[4] **manager** - *all* (default) or a package manager **format** - *full* (default) or *short*	Lists (alphabetically) installed packages whose name matches the given `package` regular expression[4] (*all* lists them all). Example: => system.sw.packages[mini,dpkg,sh → python-minimal, python2.6-minimal, ubuntu-minimal Supported package managers (executed command): dpkg (dpkg --get-selections) pkgtool (ls /var/log/packages) rpm (rpm -qa) pacman (pacman -Q) If `format` is specified as *full*, packages are grouped by package managers (each manager on a separate line beginning with its name in square brackets). If `format` is specified as *short*, packages are not grouped and are listed on a single line. Supported since Zabbix agent version 2.0.
system.swap.in[<device>,<type>]				

Key				
	Swap in (from device into memory) statistics.	Integer	**device** - device used for swapping (default is *all*) **type** - possible values: *count* (number of swapins), *sectors* (sectors swapped in), *pages* (pages swapped in). See also platform-specific details for this parameter.	Example: => system.swap.in[,pages] The source of this information is: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.out[<device>,<type>]				
	Swap out (from memory onto device) statistics.	Integer	**device** - device used for swapping (default is *all*) **type** - possible values: *count* (number of swapouts), *sectors* (sectors swapped out), *pages* (pages swapped out). See also platform-specific details for this parameter.	Example: => system.swap.out[,pages] The source of this information is: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.size[<device>,<type>]				

Key				
	Swap space size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**device** - device used for swapping (default is *all*) **type** - possible values: *free* (free swap space, default), *pfree* (free swap space, in percent), *pused* (used swap space, in percent), *total* (total swap space), *used* (used swap space) Note that *pfree*, *pused* are not supported on Windows if swap size is 0. See also platform-specific details for this parameter.	Example: => system.swap.size[,pfree] → free swap space percentage If *device* is not specified Zabbix agent will only take into account swap devices (files), physical memory will be ignored. For example, on Solaris systems *swap -s* command includes a portion of physical memory and swap devices (unlike *swap -l*). Note that this key might report incorrect swap space size/percentage on virtualized (VMware ESXi, VirtualBox) Windows platforms. In this case you may use the `perf_counter[\700(_Tota` key to obtain correct swap space percentage. Old naming: *system.swap.free*, *system.swap.total*
system.uname				

Key			
	Identification of the system.	String	Example of returned value (Unix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 Example of returned value (Windows): Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86 On Unix since Zabbix 2.2.0 the value for this item is obtained with uname() system call. Previously it was obtained by invoking ”uname -a”. The value of this item might differ from the output of ”uname -a” and does not include additional information that ”uname -a” prints based on other sources. On Windows since Zabbix 3.0 the value for this item is obtained from Win32_OperatingSystem and Win32_Processor WMI classes. Previously it was obtained from volatile Windows APIs and undocumented registry keys.

Key			
system.uptime			
	System uptime in seconds.	Integer	In item configuration, use **s** or **uptime** units to get readable values.
system.users.num			
	Number of users logged in.	Integer	**who** command is used on the agent side to obtain the value.
vfs.dev.discovery			
	List of block devices and their type. Used for low-level discovery.	JSON object	This item is supported on Linux platform only. Supported since Zabbix 4.4.0.
vfs.dev.read[<device>,<type>,<mode>]			

Key					
	Disk read statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps* *Note*: if using an update interval of three hours or more[2], will always return '0'	**device** - disk device (default is *all* [3]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Note that 'type' parameter support and defaults depend on the platform. See platform-specific details. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	You may use relative device names (for example, `sda`) as well as an optional /dev/ prefix (for example, `/dev/sda`). LVM logical volumes are supported. Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Example: => vfs.dev.read[,operations] *sps*, *ops* and *bps* on supported platforms used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 devices (1023 individual and one for *all*). Old naming: *io[*]*
vfs.dev.write[<device>,<type>,<mode>]					

Key					
		Disk write statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps* *Note*: if using an update interval of three hours or more[2], will always return '0'	**device** - disk device (default is *all* [3]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Note that 'type' parameter support and defaults depend on the platform. See platform-specific details. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	You may use relative device names (for example, `sda`) as well as an optional /dev/ prefix (for example, `/dev/sda`). LVM logical volumes are supported. Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Example: => vfs.dev.write[,operations] *sps*, *ops* and *bps* on supported platforms used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 (1023 individual and one for *all*). Old naming: *io[*]*
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age					

Key					
	Directory entry count.	Integer	**dir** - absolute path to directory **regex_incl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) **regex_excl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value) **types_incl** - directory entry types to count, possible values: *file* - regular file, *dir* - subdirectory, *sym* - symbolic link, *sock* - socket, *bdev* - block device, *cdev* - character device, *fifo* - FIFO, *dev* - synonymous with "bdev,cdev", *all* - all types (default), i.e. "file,dir,sym,sock,bdev,cdev,fifo". Multiple types must be separated with comma and quoted. **types_excl** - directory entry types (see <types_incl>) to NOT count. If some entry type is in both <types_incl> and <types_excl>, directory	Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported. Pseudo-directories "." and ".." are never counted. Symbolic links are never followed for directory traversal. On Windows, directory symlinks are skipped and hard links are counted only once. Both `regex_incl` and `regex_excl` are being applied to files and directories when calculating entry size, but are ignored when picking subdirectories to traverse (if regex_incl is "(?i)^.+\.zip$" and max_depth is not set, then all subdirectories will be traversed, but only files of type zip will be counted). Execution time is limited by the default timeout value in agent configuration (3 sec). Since large directory traversal may take longer than that, no data will be returned and

Key
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]

Key				
	Directory size (in bytes).	Integer	**dir** - absolute path to directory **regex_incl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) **regex_excl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value) **mode** - possible values: *apparent* (default) - gets apparent file sizes rather than disk usage (acts as `du -sb dir`), *disk* - gets disk usage (acts as `du -s -B1 dir`). Unlike du command, vfs.dir.size item takes hidden files in account when calculating directory size (acts as `du -sb .[^.]* * ` within dir). **max_depth** - maximum depth of subdirectories to traverse. **-1** (default) - unlimited, **0** - no descending into subdirectories. **regex_excl_dir** - regular expression describing the	Only directories with at least read permission for *zabbix* user are calculated. On Windows any symlink is skipped and hard links are taken into account only once. With large directories or slow drives this item may time out due to the Timeout setting in agent and server/proxy configuration files. Increase the timeout values as necessary. Examples: ⇒ vfs.dir.size[/tmp,log] - calculates size of all files in /tmp which contain 'log' ⇒ vfs.dir.size[/tmp,log,^.+\.old$] - calculates size of all files in /tmp which contain 'log', excluding files containing '.old' The file size limit depends on large file support. Supported since Zabbix 3.4.0.

Key				
vfs.file.cksum[file]	File checksum, calculated by the UNIX cksum algorithm.	Integer	**file** - full path to file	Example: => vfs.file.cksum[/etc/passwd] Example of returned value: 1938292000 Old naming: *cksum* The file size limit depends on large file support.
vfs.file.contents[file,<encoding>]	Retrieving contents of a file.	Text	**file** - full path to file **encoding** - code page identifier	Returns an empty string if the file is empty or contains LF/CR characters only. Byte order mark (BOM) is excluded from the output. Example: => vfs.file.contents[/etc/passwd] This item is limited to files no larger than 64 Kbytes. Supported since Zabbix agent version 2.0.
vfs.file.exists[file,<types_incl>,<types_excl>]				

Key				
	Checks if file exists.	0 - not found 1 - file of the specified type exists	**file** - full path to file **types_incl** - list of file types to include, possible values: *file* (regular file, default (if types_excl is not set)), *dir* (directory), *sym* (symbolic link), *sock* (socket), *bdev* (block device), *cdev* (character device), *fifo* (FIFO), *dev* (synonymous with "bdev,cdev"), *all* (all mentioned types, default if types_excl is set). **types_excl** - list of file types to exclude, see types_incl for possible values (by default no types are excluded)	Multiple types must be separated with a comma and the entire set enclosed in quotes "". On Windows the double quotes have to be backslash '\' escaped and the whole item key enclosed in double quotes when using the command line utility for calling zabbix_get.exe or agent2. If the same type is in both <types_incl> and <types_excl>, files of this type are excluded. Examples: => vfs.file.exists[/tmp/application => vfs.file.exists[/tmp/application => vfs.file.exists[/tmp/application The file size limit depends on large file support. The `types_incl`, `types_excl` parameters are supported since Zabbix 5.0.2. Note that the item may turn unsupported in Windows if a directory is searched within a non-existing directory, e.g. vfs.file.exists[C:\no\dir,dir] (where 'no' does not exist).

Key				
vfs.file.md5sum[file]	MD5 checksum of file.	Character string (MD5 hash of the file)	**file** - full path to file	Example: => vfs.file.md5sum[/usr/local/etc/ Example of returned value: b5052decb577e0fffd622d6dd The file size limit (64 MB) for this item was removed in version 1.8.6. The file size limit depends on large file support.
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]				

Key					
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]	Find string in a file.	The line containing the matched string, or as specified by the optional `output` parameter	**file** - full path to file **regexp** - regular expression[4] describing the required pattern **encoding** - code page identifier **start line** - the number of first line to search (first line of file by default). **end line** - the number of last line to search (last line of file by default). **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).	Only the first matching line is returned. An empty string is returned if no line matched the expression. Byte order mark (BOM) is excluded from the output. Content extraction using the `output` parameter takes place on the agent. The `start line`, `end line` and `output` parameters are supported from version 2.2. Examples: => vfs.file.regexp[/etc/passwd,zal => vfs.file.regexp[/path/to/some/f 9]+)$”„3,5,\1] => vfs.file.regexp[/etc/passwd,”^ 9]+)”„„\1] → getting the ID of user *zabbix*

Key				
	Find string in a file.	0 - match not found 1 - found	**file** - full path to file **regexp** - regular expression[4] describing the required pattern **encoding** - code page identifier **start line** - the number of first line to search (first line of file by default). **end line** - the number of last line to search (last line of file by default).	Byte order mark (BOM) is ignored. The `start line` and `end line` parameters are supported from version 2.2. Example: => vfs.file.regmatch[/var/log/app.
vfs.file.size[file]	File size (in bytes).	Integer	**file** - full path to file	The file must have read permissions for user *zabbix*. Example: => vfs.file.size[/var/log/syslog] The file size limit depends on large file support.
vfs.file.time[file,<mode>]	File time information.	Integer (Unix timestamp)	**file** - full path to the file **mode** - possible values: *modify* (default) - last time of modifying file content, *access* - last time of reading file, *change* - last time of changing file properties	Example: => vfs.file.time[/etc/passwd,modi The file size limit depends on large file support.
vfs.fs.discovery				

Key				
	List of mounted filesystems and their types. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 2.0. {#FSDRIVETYPE} macro is supported on Windows since Zabbix agent version 3.0.
vfs.fs.get				
	List of mounted filesystems, their types, disk space and inode statistics. Can be used for low-level discovery.	JSON object		Supported since Zabbix agent version 4.4.5.
vfs.fs.inode[fs,<mode>]				
	Number or percentage of inodes.	Integer - for number Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, //pfree // (free, percentage), *pused* (used, percentage)	Example: => vfs.fs.inode[/,pfree] Old naming: *vfs.fs.inode.free[*]*, *vfs.fs.inode.pfree[*]*, *vfs.fs.inode.total[*]*
vfs.fs.size[fs,<mode>]				
	Disk space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), *pused* (used, percentage)	In case of a mounted volume, disk space for local file system is returned. Example: => vfs.fs.size[/tmp,free] Reserved space of a file system is taken into account and not included when using the *free* mode. Old naming: *vfs.fs.free[*]*, *vfs.fs.total[*]*, *vfs.fs.used[*]*, *vfs.fs.pfree[*]*, *vfs.fs.pused[*]*
vm.memory.size[<mode>]				

Key				
	Memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**mode** - possible values: *total* (default), *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*, *used*, *pused* (used, percentage), *available*, *pavailable* (available, percentage) See also platform-specific support and additional details for this parameter.	This item accepts three categories of parameters: 1) *total* - total amount of memory; 2) platform-specific memory types: *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*; 3) user-level estimates on how much memory is used and available: *used*, *pused*, *available*, *pavailable*.
web.page.get[host,<path>,<port>]				

Key				
web.page.perf[host,<path>,<port>]	Get content of web page.	Web page source as text (including headers)	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[5]. Missing scheme will be treated as *http*. If URL is specified `path` and `port` must be empty. Specifying user name/password when connecting to servers that require authentication, for example: `http://user:password@www.example.com` is only possible with cURL support [5]. Punycode is supported in hostnames. **path** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP)	This item turns unsupported if the resource specified in `host` does not exist or is unavailable. `host` can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. Example: => web.page.get[www.example.c => web.page.get[https://www.exa => web.page.get[https://blog.exa => web.page.get[localhost:80] => web.page.get[”[::1]/server-status”]

Key					
	Loading time of full web page (in seconds).	Float	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[5]. Missing scheme will be treated as *http*. If URL is specified `path` and `port` must be empty. Specifying user name/password when connecting to servers that require authentication, for example: `http://user:password@www.example.com` is only possible with cURL support [5]. Punycode is supported in hostnames. **path** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP)	This item turns unsupported if the resource specified in `host` does not exist or is unavailable. `host` can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. Example: => web.page.perf[www.example.c => web.page.perf[https://www.ex
web.page.regexp[host,<path>,<port>,regexp,<length>,<output>]					

Key					
		Find string on a web page.	The matched string, or as specified by the optional `output` parameter	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[5]. Missing scheme will be treated as *http*. If URL is specified `path` and `port` must be empty. Specifying user name/password when connecting to servers that require authentication, for example: `http://user:password@www.example.com` is only possible with cURL support [5]. Punycode is supported in hostnames. **path** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP) **regexp** - regular expression[4] describing the required pattern **length** - maximum number of characters to return **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match	This item turns unsupported if the resource specified in `host` does not exist or is unavailable. `host` can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled. Content extraction using the `output` parameter takes place on the agent. The `output` parameter is supported from version 2.2. Example: => web.page.regexp[www.examp => web.page.regexp[https://www

Key				
zabbix.stats[<ip>,<port>]	Return a set of Zabbix server or proxy internal metrics remotely.	JSON object	**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1) **port** - port of server/proxy to be remotely queried (default is 10051)	Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance. A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats.
zabbix.stats[<ip>,<port>,queue,<from>,<to>]	Return number of monitored items in the queue which are delayed on Zabbix server or proxy remotely.	JSON object	**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1) **port** - port of server/proxy to be remotely queried (default is 10051) **queue** - constant (to be used as is) **from** - delayed by at least (default is 6 seconds) **to** - delayed by at most (default is infinity)	Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance.

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。

Troubleshooting agent items

- If used with the passive agent, *Timeout* value in server configuration may need to be higher than *Timeout* in the agent configuration file. Otherwise the item may not get any value because the server request to agent timed out first.

Mandatory and optional parameters

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

Usage with command-line utilities

Note that when testing or using item keys with zabbix_agentd or zabbix_get from the command line you should consider shell syntax too.

For example, if a certain parameter of the key has to be enclosed in double quotes you have to explicitly escape double quotes, otherwise they will be trimmed by the shell as special characters and will not be passed to the Zabbix utility.

Examples:

```
$ zabbix_agentd -t 'vfs.dir.count[/var/log,,,"file,dir",,0]'
```

```
$ zabbix_agentd -t vfs.dir.count[/var/log,,,\"file,dir\",,0]
```

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[pavailable] → 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。

Zabbix agent 2

监控项 keys

该表仅描述 Zabbix agent 2 可用的监控项 keys 的详细信息。

Note:
不带尖括号 < > 的参数是必需的。用尖括号 < > 标记的参数是可选的。

Key	描述 *	返回值 ** ** 参数	* 注释	
ceph.df.details [<connString>, <user>, <apikey>]	在 pool 中的集群数据使用和分布详细信息。JSON 对象	**connString** - U	l 或 session 名称。 此监控项由 [Ceph plu-gin](/m	**user, password** - Ceph 登录凭证。 nual/config/items/plugins#plu of-the-box) 支持。
ceph.osd.stats [<connString>, <user>, <apikey>]	汇总和按 OSD 统计。JSON 对	**connSt	ing** - URI 或 session 名称。此监控项由 [Ceph p	**user, password** - Ceph 登录凭证。 ugin](/manual/config/items/plu of-the-box) 支持。
ceph.osd.discovery [<connString>, <user>, <apikey>]	发现的 OSD 列表。用于低级别发现。JSON 对象	**connString**	URI 或 session 名称。 此监控项由 [Ceph plugin]	**user, password** - Ceph 登录凭证。 /manual/config/items/plugins# of-the-box) 支持。
ceph.osd.dump [<connString>, <user>, <apikey>]	OSD 的使用阈值和状态。JSON 对象	**connStri	g** - URI 或 session 名称。此监控项由 [Ceph pl	**user, password** - Ceph 登录凭证。 gin](/manual/config/items/plug of-the-box) 支持。
ceph.ping [<connString>, <user>, <apikey>]	测试是否可以建立与 Ceph 的连接。0 - 测试连接失败（一	由错误引起，比如认证和配置问题）**connString** - URI 或 session 名称。1 - 测试连接成功 **use	此监控项由 [Ceph plu-gin](/manual/config/ password** - Ceph 登录凭证。	lugins#plugins_supplied_out-of-the-box) 支持。 items/,
ceph.pool.discovery [<connString>, <user>, <apikey>]	已发现池的列表。用于低级别发现。JSON 对象	**connString** -	RI 或 session 名称。 此监控项由 [Ceph plugin](/	**user, password** - Ceph 登录凭证。 anual/config/items/plugins#pl of-the-box) 支持。
ceph.status [<connString>, <user>, <apikey>]				

Key				
	集群总体状态。JSON 对	**connSt	ing** - URI 或 session 名称。此监控项由 [Ceph**user, password** - Ceph 登录凭证。	lugin](/manual/config/items/pl of-the-box) 支持。
docker.container_info [<ID>]	容器的信息。[Cont	inerInspect](https://docs.docker.com/engine/api/v1.28/#operation/C API 输出的 JSON 信息 **ID** — 容	的 ID 或名字此监控项从 Zabbix 5.0.	开始支持。 必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.container_stats [<ID>]	容器资源使用统计。[Contain	rStats](https://docs.docker.com/engine/api/v1.28/#operation/Conta API 输出的 JSON 信息 **ID** — 容器的 I	或名字 Docker 插件支持此监控项。	。 必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.containers	容器列表。[Con	ainerList](https://docs.docker.com/engine/api/v1.28/#operation/Con API 输出的 JSON 信息 -	Docker 插件支持此监控项。	此监控项。 必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.containers.discovery[<options>]	容器列表。用于低级别发现。JSON 对象	**options** —	否应该发现所有或仅发现正在运行的容器。支持的值： Docker 插件支持此监控项。 *true* - 发现所有容器; *false* - 仅发现运行的容器 (默认值)。必须将 Agent2 用户	<'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.data_usage	当前系统数据使用情况。[SystemDat	Usage](https://docs.docker.com/engine/api/v1.28/#operation/Syste API 输出的 JSON 信息 -	Docker 插件支持此监控项。	。 必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.images				

Key				
	镜像列表。[Ima	eList](https://docs.docker.com/engine/api/v1.28/#operation/ImageL API 输出的 JSON 信息 -	Docker 插件支持此监控项。	此监控项。 必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.images.discovery				
	镜像列表。用于低级别发现。JSON 对象	-	Docker 插件支持此监控项	必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.info				
	系统信息。[Sys	emInfo](https://docs.docker.com/engine/api/v1.28/#operation/Syste API 输出的 JSON 信息 -	Docker 插件支持此监控项。	此监控项。 必须将 Agent2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
docker.ping				
	测试 Docker daemon 运行情况。1 - 测试	接成功 - 0 - 测试连接失败	Docker 插件支持此监必须将 Ag	项。 nt2 用户（'zabbix'）添加到'docker' 用户组否则将因为权限限制而无法获取信息。
memcached.ping[<connString>,<user>,<password>]				
	测试连接情况。1 - 测试	接成功 **connString 0 - 测试连接失败（一般由错误引起，比如认证和配置问题）	* - URI 或 session 名称。此监控项由 Memcached p	ugin 支持。
memcached.stats[<connString>,<user>,<password>,<type>]				
	STATS 命令输出信息。JSON -	输出序列化的 JSON 信息 **connString**	- URI 或 session 名称。 此监控项由 Memcached plu**user, password** - Memchached 登录凭证. **type** - 要返回的统计类型: *items*, *sizes*, *slabs* or *settings* （默认为空将返回常规统计信息）。	in 支持。
mysql.db.discovery[<connString>, <username>, <password>]				

Key				
	mysql 数据库列表。用于低级别发现。LLD JSON 格式的”s	ow databases”SQL 返回结果。**connString** - URI	session 名称。此监控项由 MySQL plugin 支持。**username, password** - MySQL 登录凭证。	
mysql.db.size[<connString>, <username>, <password>,dbName]				
	数据库大小，单位为 bytes。”select c	alesce(sum(data_length + index_length),0) as size from information_schema.tables where table_schema=?” SQL 返回结果。**connString**	URI 或 session 名称。 此监控项由 MySQL plugin 支 **username, password** - MySQL 登录凭证。**dbName** - Database name.	。
mysql.get_status_variables[<connString>, <username>, <password>]				
	global status 变量值。JSO	格式的”show global status” SQL 返回结果。**connStrin	** - URI 或 session 名称。此监控项由 MySQL plug**username, password** - MySQL 登录凭证.	n 支持。
mysql.ping[<connString>, <username>, <password>]				
	测试连接情况。1 - 测试	接成功 **connString 0 - 测试连接失败（一般由错误引起，比如认证和配置问题）.	* - URI 或 session 名称。此监控项由 MySQL plugi**username, password** - MySQL 登录凭证.	支持。
mysql.replication.discovery[<connString>, <username>, <password>]				
	MySQL 副本列表。用于低级别发现。LLD JSON 格式的”	how slave status” SQL 返回结果。**connString** - URI	或 session 名称。 此监控项由 MySQL plugin 支持。**username, password** - MySQL 登录凭证。	
mysql.replication.get_slave_status[<connString>, <username>, <password>, <masterHost>]				

Key				
	副本状态。JSON	式的”show slave status” SQL 返回结果。 **connString	* - URI 或 session 名称。 此监控项由 MySQL plugi**username, password** - MySQL 登录凭证。 **masterHost** - 主服务器主机名	支持。
mysql.version[<connString>, <username>, <password>]				
	MySQL 版本。 My	QL 实例版本。 **connS	ring** - URI 或 session 名称。 此监控项由 MySQL**username, password** - MySQL 登录凭证。	lugin 支持。
oracle.diskgroups.stats[<connString>,<user>,<password>,<service>]				
	ASM 磁盘组统计信息。JSON 对象	**connStr	ng** - URI 或 session 名称。 此监控项由 Oracle p**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	ugin 支持。
oracle.diskgroups.discovery[<connString>,<user>,<password>,<service>]				
	ASM 磁盘组列表。用于低级别发现。JSON 对象	**connString**	URI 或 session 名称。 此监控项由 Oracle plugin 支 **user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称	。
oracle.archive.info[<connString>,<user>,<password>,<service>]				
	Archive 日志统计信息。JSON 对	**connSt	ing** - URI 或 session 名称。 此监控项由 Oracle**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	lugin 支持。
oracle.cdb.info[<connString>,<user>,<password>,<service>]				

Key				
	CDB 信息。JS	N 对象 **co	nString** - URI 或 session 名称。 此监控项由 Ora**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	le plugin 支持。
oracle.custom.query[<connString>,<user>,<password>,<service>, queryName, <args...>]				
	用户自定义 SQL 查询结果。JSON 对象	**connStrin	** - URI 或 session 名称。 此监控项由 Oracle plu**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。 **queryName** — 自定义查询的名称（不带扩展名的 sql 文件名称）。 **args...** — 传递给查询的一个或多个逗号分隔的参数	in 支持。
oracle.datafiles.stats[<connString>,<user>,<password>,<service>]				
	数据文件统计信息。JSON 对象	**connStri	g** - URI 或 session 名称。 此监控项由 Oracle pl**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	gin 支持。
oracle.db.discovery[<connString>,<user>,<password>,<service>]				
	数据库列表。用于低级别发现. JSON 对象	**connString**	- URI 或 session 名称。 此监控项由 Oracle plugin**user, password** - Oracle 登录凭证. **service** - Oracle 服务名称。	持。
oracle.fra.stats[<connString>,<user>,<password>,<service>]				

Key				
	FRA 统计信息。JSON	对象 **conn	tring** - URI 或 session 名称。此监控项由 Oracl**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	plugin 支持。
oracle.instance.info[<connString>,<user>,<password>,<service>]				
	实例统计信息。JSON 对	**connSt	ing** - URI 或 session 名称。此监控项由 Oracle**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	lugin 支持。
oracle.pdb.info[<connString>,<user>,<password>,<service>]				
	PDB 信息。JS	N 对象 **co	nString** - URI 或 session 名称。 此监控项由 Ora**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	le plugin 支持。
oracle.pdb.discovery[<connString>,<user>,<password>,<service>]				
	PDB 列表。用于低级别发现。JSON 对象	**connString	* - URI 或 session 名称。此监控项由 Oracle plug**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	n 支持。
oracle.pga.stats[<connString>,<user>,<password>,<service>]				
	PGA 统计信息。JSON	对象 **conn	tring** - URI 或 session 名称。此监控项由 Oracl**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	plugin 支持。
oracle.ping[<connString>,<user>,<password>,<service>]				

Key				
	测试 Oracle 连接。0 -	试连接失败（一般由错误引起，比如认证和配置问题） **connString** - URI 或 sessio1 - 测试连接成功 **use	名称。 此监控项由 Oracle plugin 支持。, password** - Oracle 登录凭证. **service** - Oracle 服务名称。	
oracle.proc.stats[<connString>,<user>,<password>,<service>]				
	进程统计信息。JSON 对	**connSt	ing** - URI 或 session 名称。 此监控项由 Oracle**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	lugin 支持。
oracle.redolog.info[<connString>,<user>,<password>,<service>]				
	控制文件中的日志文件信息。JSON 对象	**connString**	- URI 或 session 名称。 此监控项由 Oracle plugin**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	持。
oracle.sga.stats[<connString>,<user>,<password>,<service>]				
	SGA 统计信息。JSON	对象 **conn	tring** - URI 或 session 名称。 此监控项由 Oracl**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。	plugin 支持。
oracle.sessions.stats[<connString>,<user>,<password>,<service>,<lockMaxTime>]				
	Sessions 统计信息。JSON	对象 **conn	tring** - URI 或 session 名称。 此监控项由 Oracl**user, password** - Oracle 登录凭证。 **service** - Oracle 服务名称。 **lockMaxTime** - 会话锁定的最大持续时间（以秒为单位)。默认值：600 秒	plugin 支持。
oracle.sys.metrics[<connString>,<user>,<password>,<service>,<duration>]				

Key				
	系统指标列表。JSON 对	**connSt	ing** - URI 或 session 名称。此监控项由 Oracle**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。**duration** - 系统指标采集间隔，以秒为单位。可能的值: *60* — long duration (默认值), *15* — short duration.	lugin 支持。
oracle.sys.params[<connString>,<user>,<password>,<service>]				
	系统参数列表。JSON 对	**connSt	ing** - URI 或 session 名称。此监控项由 Oracle**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	lugin 支持。
oracle.ts.stats[<connString>,<user>,<password>,<service>]				
	表空间统计信息。JSON 对象	**connStr	ng** - URI 或 session 名称。此监控项由 Oracle p**user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	ugin 支持。
oracle.ts.discovery[<connString>,<user>,<password>,<service>]				
	表空间列表。用于低级别发现。JSON 对象	**connString**	URI 或 session 名称。 此监控项由 Oracle plugin 支 **user, password** - Oracle 登录凭证。**service** - Oracle 服务名称。	。
oracle.user.info[<connString>,<user>,<password>,<username>]				

Key				
	表空间列表。用于低级别发现。JSON 对象	**connString**	URI 或 session 名称。 此监控项由 Oracle plugin 支 **user, password** - Oracle 登录凭证. **service** - Oracle 服务名称。 **username** - 用户名，不支持小写用户名。默认值: 当前用户。	。
pgsql.autovacuum.count[<uri>,<username>,<password>,<dbName>]				
	autovacuum 数量。SQ	查询结果。**uri**	- URI 或 session 名称。 从 Zabbix 5.0**username, password** - PostgreSQL 登录凭证。 **dbName** - Database name.	1 开始支持此监控项。
pgsql.archive[<uri>,<username>,<password>,<dbName>]				
	archived 文件信息。SQL 查	结果（JSON 格式）。**uri** - URI	或 session 名称。 返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称。**	<ql.archive.count_archived_fil - 已经成功归档的 WAL 文件数量。 gsql.archive.failed_trying_to_a - 归档 WAL 文件的失败尝试次数。 **pgsql.archive.count_files_t** - 需要归档的文件数。 **pgsql.archive.size_files_to_** - 需要归档的文件大小。 从 Zabbix 5.0.1 开始支持此监控项。
pgsql.bgwriter[<uri>,<username>,<password>,<dbName>]				

Key				
	数据库集群的检查点总数，按检查点类型细分。SQL 查询结果 (JSON 格式)。	**uri** - URI 或 session 名称。	返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称。 **	<ql.bgwriter.buffers_alloc** - 分配的缓冲区数 gsql.bgwriter.buffers_backend - 后端直接写入的缓冲区数。 **pgsql.bgwriter.maxwritten** - 后端写入器由于写入了太多缓冲区而停止清除扫描的次数。 **pgsql.bgwriter.buffers_bac** - 后端必须执行自己的 fsync 调用而不是写入器的次数。 **pgsql.bgwriter.buffers_clea** - 后端写入器写入的缓冲区数。 **pgsql.bgwriter.buffers_che** - 在检查点期间写入的缓冲区数。 **pgsql.bgwriter.checkpoints** - 已执行的计划检查点的数量。 **pgsql.bgwriter.checkpoints** - 已执行的请求检查点的数量。 **pgsql.bgwriter.checkpoint_** - 在将文件写入磁盘的检查点处理过程中花费的总时间，以毫秒为单位。 **pgsql.bgwriter.sync_time** - 在检查点处理过程中文件与磁盘同步花费的总时间。 从 Zabbix 5.0.1 开始支持此监控项。
pgsql.cache.hit[<uri>,<username>,<password>,<dbName>]	PostgreSQL 缓冲区缓存命中率。SQL 查询返回结	(百分比)。**uri** - URI 或 sess	on 名称。 从 Zabbix 5.0.1 开始支持此监控项。 **username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称	
pgsql.connections[<uri>,<username>,<password>,<dbName>]				

Key					
pgsql.dbstat[<uri>,<username>,<password>,dbName]	按类型的连接信息。JSON 对象.	**uri** -	RI 或 session 名称。 返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称。**	<ql.connections.active** - active 的连接 gsql.connections.fastpath_fun - 使用 fast-path 功能的连接 **pgsql.connections.idle** - 空闲的连接 **pgsql.connections.idle_in_t** - 空闲的连接（in transaction） **pgsql.connections.prepare** - 预处理连接数。 **pgsql.connections.total** - 连接总数 **pgsql.connections.total_pc** - 关于 PostgreSQL 服务器的 “ max_connections” 设置的总连接数百分比。 **pgsql.connections.waiting** - 查询中的连接数。 **pgsql.connections.idle_in_t** - 空闲的连接（in transaction）且事务中的语句之一导致错误。 从 Zabbix 5.0.1 开始支持此监控项。

Key					
	数据库统计信息。用于低级别发现。SQL 查询结果（JSON 格式）	**uri** - URI 或 session	称。 返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称 *	<ql.dbstat.numbackends[”{#D - 当前连接到该数据库的后端数。 pgsql.dbstat.sum.blk_read_tim - 数据库后端读取数据文件块所花费的时间，以毫秒为单位。 **pgsql.dbstat.sum.blk_write** - 数据库中后端写入数据文件块所花费的时间，以毫秒为单位。 **pgsql.dbstat.sum.checksu** - 数据页或在共享对象上校验失败次数, 如果数据校验功能被禁止则返回 NULL (仅 PostgreSQL 12 支持)。 **pgsql.dbstat.blks_read.rat** - 数据库中读取的磁盘块数。 **pgsql.dbstat.deadlocks.rat** - 数据库中检测到的死锁发生次数。 **pgsql.dbstat.blks_hit.rate[** - 缓冲区高速缓存中已命中的次数（仅包括 PostgreSQL Pro 缓冲区高速缓存中的命中）。 **pgsql.dbstat.xact_rollback.** - 数据库中回滚事务数。 **pgsql.dbstat.xact_commit.** - 数据库中已提交的事务数。 **pgsql.dbstat.tup_updated.** - 数据库中更新的行数。 **pgsql.dbstat.tup_returned.** - 数据库中查询返回的行数。 **pgsql.dbstat.tup_inserted.** - 数据库中插入的行数。 **pgsql.dbstat.tup_fetched.r** - 数据库中 fetched 的行数。 **pgsql.dbstat.tup_deleted.r** - 数据库中删除的行数。 **pgsql.dbstat.conflicts.rate** - 由于与备用服务器上的恢复冲突而取消查询的数据库统计信息。 **pgsql.dbstat.temp_files.rat** - 数据库中查询

Key
pgsql.dbstat.sum[<uri>,<username>,<password>, <dbName>]

Key				
	集群中所有数据库的摘要数据。JSON 格式的 SQL 查询结	。**uri** - URI 或 sessi	n 名称。 返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称 *	<ql.dbstat.numbackends** - 当前连接到该数据库的后端数。 pgsql.dbstat.sum.blk_read_tim - 数据库中后端读取数据文件块所花费的时间，以毫秒为单位。 **pgsql.dbstat.sum.blk_write** - 数据库中后端写入数据文件块所花费的时间，以毫秒为单位。 **pgsql.dbstat.sum.checksu** - 数据页或在共享对象上校验失败次数, 如果数据校验功能被禁止则返回 NULL (仅 PostgreSQL 12 支持). **pgsql.dbstat.sum.xact_com** - 数据库中已提交的事务数。 **pgsql.dbstat.sum.conflicts** - 由于与备用服务器上的恢复冲突而取消查询的数据库统计信息。 **pgsql.dbstat.sum.deadlock** - 数据库中死锁发生次数 **pgsql.dbstat.sum.blks_rea** - 数据库中读取的磁盘块数。 **pgsql.dbstat.sum.blks_hit** - 缓冲区高速缓存中已命中的次数（仅包括 PostgreSQL Pro 缓冲区高速缓存中的命中）。 **pgsql.dbstat.sum.temp_by** - 数据库中查询写入临时文件的数据总量。包括来自所有临时文件的数据，忽略 log_temp_files 设置和创建临时文件的原因（例如，排序或散列）。 **pgsql.dbstat.sum.temp_file** - 数据库中查询创建的临时文件数。包括所有临时文件的数量，忽略 log_temp_files 设置和创建临时文件的原因（例如，排序或散

Key			
pgsql.db.age[<uri>,<username>,<password>, dbName]	最老的数据库 FrozenXID 持续时间。用于低级别发现。SQL 查询结果。	**uri** - URI 或 session	名称。 此监控项从 Zabbix 5.0.1 开始支持。 **username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称。
pgsql.db.bloating_tables[<uri>,<username>,<password>, <dbName>]	每个数据库 bloating 表数量。用于低级别发现. SQL 查询结果。	**uri** - URI 或 sess	on 名称。 此监控项从 Zabbix 5.0.1 开始支持。 **username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称。
pgsql.db.discovery[<uri>,<username>,<password>, <dbName>]	PostgreSQL 数据库列表。用于低级别发现. LLD JSON 格式的 S	L 查询结果。**uri** - URI 或 sess	on 名称。 此监控项从 Zabbix 5.0.1 开始支持。 **username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称。
pgsql.db.size[<uri>,<username>,<password>, dbName]	数据库大小（bytes）。用于低级别发现。SQL 查询结果。	**uri** - URI 或 sess	on 名称。 此监控项从 Zabbix 5.0.1 开始支持。 **username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称。
pgsql.locks[<uri>,<username>,<password>, <dbName>]			

Key					
		每个数据库的已授予锁的信息。用于低级别发现。SQL 查询结果（JSON 格式）。	**uri** - URI 或 session 名称。	返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称。 **	<ql.locks.shareupdateexclusiv - 共享更新排它锁的数量。 gsql.locks.accessexclusive[”{# - 访问排它锁的数量。 **pgsql.locks.accessshare[”** - 访问共享锁的数量。 **pgsql.locks.exclusive[”{#D** - the number of exclusive locks. **pgsql.locks.rowexclusive[”** - 行排它锁的数量。 **pgsql.locks.rowshare[”{#D** - 行共享锁的数量。 **pgsql.locks.share[”{#DBN** - 共享锁的数量。 **pgsql.locks.sharerowexclu** - 共享行排它锁的数量。 此监控项从 Zabbix 5.0.1 开始支持。
pgsql.oldest.xid[<uri>,<username>,<password>,<dbName>]					
		最老的 XID 持续时间。SQI 查询结果	**uri** - UR	或 session 名称。 此监控项从 Zabbix 5.0.**username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称。	开始支持。
pgsql.ping[<uri>,<username>,<password>,<dbName>]					
		测试连接情况。 1 - 测试	接成功 **uri** - UR 0 - 测试连接失败（一般由错误引起，比如认证和配置问题）。 **dbName** - Database 名称。	或 session 名称。 此监控项从 Zabbix 5.0.**username, password** - PostgreSQL 登录凭证。	开始支持。
pgsql.replication.count[<uri>,<username>,<password>,<dbName>]					
		standby 服务数量。SQL 查	结果。**uri** -	URI 或 session 名称。 此监控项从 Zabbix 5**username, password** - PostgreSQL 登录凭证。 **dbName** - Database 名称。	0.1 开始支持。

Key				
pgsql.replication.recovery_role[<uri>,<username>,<password>,<dbName>]				
	Recovery 状态。0	master 模式 **ur1 - recovery** 仍在进行 **(standby** 模式**)** use	** - URI 或 session 名称。此监控项从 Zabname, password** - PostgreSQL 登录凭证。**dbName** - Database 名称。	ix 5.0.1 开始支持。
pgsql.replication.status[<uri>,<username>,<password>,<dbName>]				
	Replication 状态。0	streaming is down **1 - streaming is up 2 - master 模式 *	ri** - URI 或 session 名称。此监控项从 Z**username, password** - PostgreSQL 登录凭证。dbName** - Database 名称。	bbix 5.0.1 开始支持。
pgsql.replication_lag.b[<uri>,<username>,<password>,<dbName>]				
	Replication 延迟大小（bytes)。SQL 查询结	。**uri** - U	I 或 session 名称。 此监控项从 Zabbix 5.0**username, password** - PostgreSQL 登录凭证。**dbName** - Database 名称。	1 开始支持。
pgsql.replication_lag.sec[<uri>,<username>,<password>,<dbName>]				
	Replication 延迟时间（秒)。SQL 查询结果	**uri** - UR	或 session 名称。 此监控项从 Zabbix 5.0.**username, password** - PostgreSQL 登录凭证。**dbName** - Database 名称。	开始支持。
pgsql.uptime[<uri>,<username>,<password>,<dbName>]				
	PostgreSQL 启动时间（毫秒)。SQL 查询结果。	**uri** - URI	或 session 名称。 此监控项从 Zabbix 5.0.1**username, password** - PostgreSQL 登录凭证。**dbName** - Database 名称。	始支持。
pgsql.wal.stat[<uri>,<username>,<password>,<dbName>]				

Key				
	WAL 统计信息。JSON	式的 SQL 查询结果。**uri** - UR	或 session 名称。 返回的数据由依赖项处理: **username, password** - PostgreSQL 登录凭证。 **pg**dbName** - Database 名称。**	<ql.wal.count** — WAL 文件数量。 gsql.wal.write** - WAL lsn 以使用大小。(单位为 bytes). 此监控项从 Zabbix 5.0.1 开始支持。
redis.config[<connString>,<password>,<pattern>]				
	返回与模式匹配的 Redis 实例配置参数。JSON - 使用了全局样式	式（包含通配符），则返回 JSON **connString** - URI 或 session 名称。 single value - 未使用全局样式模式（不包含任何通配符），则返回 **pattern** - glob-sty	此监控项从 Zabbix 4.4.5 开始支持。 **password** - Redis 密码。 e 模式 (* 默认值).	
redis.info[<connString>,<password>,<section>]				
	INFO 命令输出信息。JSON 对	**connSt	ing** - URI 或 session 名称。 此监控项从 Zab-bix**password** - Redis 密码。 **section** - 参见section(*default* 默认值)。	.4.5 开始支持。
redis.ping[<connString>,<password>]				
	测试连接情况。 1 - 测试	接成功 **connString 0 - 测试连接失败（一般由错误引起，比如认证和配置问题）	* - URI 或 session 名称。 此监控项从 Zabbix 4.4.**password** - Redis 密码。	开始支持。
redis.slowlog.count[<connString>,<password>]				
	从 Redis 启动以来 slowlog 数量。整型	**connStr	ng** - URI 或 session 名称。 此监控项从 Zabbix 4**password** - Redis 密码。	4.5 开始支持。
systemd.unit.get[<unit name>,<interface>]				

Key					
	Systemd unit 所有属性。JSON	对象 **unit	name** - unit 名称 (在监控项原型中可以使用宏 {#UNIT.NAME} 来发现名称) 此监控项仅支持 Linux 平台。 **interface** - unit 接口类型, 可能的值: *Unit* (默认值), *Service*, *Socket*, *Device*, *Mount*, *Automount*, *Swap*, *Target*, *Path*	<Systemd unit 的 LoadState, ActiveState 和 UnitFileState 返回示例: `"ActiveState":{"state":` 此监控项从 Zabbix 5.0.7 开始支持。
systemd.unit.info[<unit name>,<property>,<interface>]				
	Systemd unit 信息。字符	**uni	name** - unit 名称 (在监控项原型中可以使用宏 {#UNIT.NAME} 来发现名称) 此监控项允许如 [dbus API](https: **property** - unit 属性 (如：ActiveState (默认值), LoadState, Description) **interface** - unit 接口类型 (如：Unit (默认值), Socket, Service) 此监控项仅支持 L	/www.freedesktop.org/wiki/Sof 所描述一样从特定类型的接口检索特定的属性。 nux 平台。 示例: => sys-temd.unit.info[”{#UNIT.NAME - 采集已发现的 Systemd unit 活动状态信息 (active, reloading, inactive, failed, activating, deactivating) => sys-temd.unit.info[”{#UNIT.NAME - 采集已发现的 Systemd unit load 状态信息 => sys-temd.unit.info[mysqld.service - 采集服务 ID 信息 (mysqld.service) => sys-temd.unit.info[mysqld.service - 采集服务描述信息 (MySQL Server) =>systemd.unit.info[mysqld.s - 采集服务最后一次进入活动状态的时间 (1562565036283903) => sys-temd.unit.info[dbus.socket,NO - 采集此套接字单元的连接数

Key			
systemd.unit.discovery[<type>]	systemd units 列表以及详细信息。用于低级别发现。JSON 对象	**type** - 可能的值: *	ll, automount, device, mount, path, service* (default), *socket*, *swap*, *target* 此监控项仅支持 Linux 平台。

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。

Items

All hosts / Zabbix server Enabled ZBX SNMP JMX IPMI Applications 13 Items 81 Triggers 47

Item Preprocessing

* Name	SNMP: InOctets (Bps)
Type	SNMPv3 agent
* Key	SNMP-InOctets-Bps
* Host interface	127.0.0.1 : 161
* SNMP OID	.1.3.6.1.2.1.2.2.1.10.3
Context name	
Security name	
Security level	authPriv
Authentication protocol	MD5 SHA
Authentication passphrase	
Privacy protocol	DES AES
Privacy passphrase	
Port	161
Type of information	Numeric (float)

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 MIB 筛选器

介绍

MIB 是一个管理信息库。MIB 筛选器允许使用 OID(对象标识符) 的文本表示形式。

举例,

`ifHCOutOctets`

OID 的文本表示形式

`1.3.6.1.2.1.31.1.1.1.10`

在使用 Zabbix 监控 SNMP 设备时，可以使用以上任何一种，但是使用文本表示形式时感觉更舒适，该方式需要安装 MIB 筛选器。

安装 MIB 筛选器

Debian 操作系统:

```
# apt install snmp-mibs-downloader
# download-mibs
```

RedHat 操作系统:

```
# yum install net-snmp-libs
```

启用 MIB 筛选器

在 RedHat 操作系统上，默认情况下启用 mib 筛选器。在 Debian 的操作系统上您必须编辑`/etc/snmp/snmp.conf` 和取消 `mibs :` 的注释

由于许可证的原因，snmp软件不包含MIB筛选器，因此默认情况下MIB处于禁用状态，如果添加了MIB，可以通过取消注释来

测试 MIB 筛选器

可以用使用 `snmpwalk` 程序来测试 snmp 管理信息库, 如果尚未安装, 请按照以下说明进行操作。

基于 Debian 操作系统:

```
# apt install snmp
```

基于 RedHat 操作系统:

```
# yum install net-snmp-utils
```

安装完成后, 请使用下面的命令查询网络设备，注意命令一定不能输入错误:

```
$ snmpwalk -v 2c -c public <NETWORK DEVICE IP> ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 176137634
IF-MIB::ifInOctets.2 = Counter32: 0
IF-MIB::ifInOctets.3 = Counter32: 240375057
IF-MIB::ifInOctets.4 = Counter32: 220893420
[...]
```

在 Zabbix 中使用 MIB

这个非常重要.Zabbix 进程不会自动生效 MIB 筛选器的变更，因此每次更改后需要重启启动 Zabbix server 和 proxy，例如:

```
# service zabbix-server restart
```

重启后，MIB 筛选器的更改才会生效。

使用自定义 MIB 筛选器

每个 GNU/Linux 发行版都有标准的 MIB 库，但是一些设备供应商单独为他们的设备提供 MIB 库。

假设你要使用CISCO-SMI的 MIB 库。以下的说明将下载并安装:

```
# wget ftp://ftp.cisco.com/pub/mibs/v2/CISCO-SMI.my -P /tmp
# mkdir -p /usr/local/share/snmp/mibs
# grep -q '^mibdirs +/usr/local/share/snmp/mibs' /etc/snmp/snmp.conf 2>/dev/null || echo "mibdirs +/usr/lo
# cp /tmp/CISCO-SMI.my /usr/local/share/snmp/mibs
```

现在您应该能够使用它了。尝试将对象 *ciscoProducts* 从 MIB 库转换为 OID:

```
# snmptranslate -IR -On CISCO-SMI::ciscoProducts
.1.3.6.1.4.1.9.1
```

如果你收到的结果是错误信息而不是 OID。请确保先前的命令没有返回任何错误。

在使用命令行工具和 Zabbix 时，需要先将对象名称转换工作完成，这样您就可以使用自定义 MIB 库了。请注意查询时使用的 MIB 名称前缀 (CISCO-SMI::)。

不要忘记在 Zabbix 中使用这个 MIB 库之前重新启动 Zabbix server/proxy。

<note important> 请记住，MIB 文件可以有依赖项。也就是说，一个 MIB 库可能需要另一个 MIB 库。为了满足这些依赖关系，您必须安装所有受影响的 MIB 库。:::

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]’;

Note:
如果没有引用脚本名称，snmptrapd 将拒绝启动消息，类似:
Regexp modifiers ”/l” and ”/a” are mutually exclusive at (eval 2) line 1, at end of line
Regexp modifier ”/l” may not appear twice at (eval 2) line 1, at end of line

Warning:
net snmp 代理不支持带有 SNMPv3/USM 的 AES256。

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

Log rotation

Zabbix does not provide any log rotation system - that should be handled by the user. The log rotation should first rename the old file and only later delete it so that no traps are lost:

1. Zabbix opens the trap file at the last known location and goes to step 3
2. Zabbix checks if the currently opened file has been rotated by comparing the inode number to the define trap file’s inode number. If there is no opened file, Zabbix resets the last location and goes to step 1.
3. Zabbix reads the data from the currently opened file and sets the new location.
4. The new data are parsed. If this was the rotated file, the file is closed and goes back to step 2.
5. If there was no new data, Zabbix sleeps for 1 second and goes back to step 2.

File system

Because of the trap file implementation, Zabbix needs the file system to support inodes to differentiate files (the information is acquired by a stat() call).

6 Setup example

This example uses snmptrapd + SNMPTT to pass traps to Zabbix server. Setup:

1. **zabbix_server.conf** - configure Zabbix to start SNMP trapper and set the trap file:
   StartSNMPTrapper=1
   SNMPTrapperFile=/tmp/my_zabbix_traps.tmp
2. **snmptrapd.conf** - add SNMPTT as the trap handler:
   traphandle default snmptt
3. **snmptt.ini** -
   enable the use of the Perl module from the NET-SNMP package:
   net_snmp_perl_enable = 1
   configure output file and time format:

log_file = /tmp/my_zabbix_traps.tmp
date_time_format = %H:%M:%S %Y/%m/%d

4. **snmptt.conf** - define a default trap format:
   EVENT general .* "General event" Normal
   FORMAT ZBXTRAP $aA $ar
5. Create an SNMP item TEST:
   Host's SNMP interface IP: 127.0.0.1
   Key: snmptrap["General"]
   Log time format: hh:mm:ss yyyy/MM/dd

This results in:

1. Command used to send a trap:
   snmptrap -v 1 -c public 127.0.0.1 '.1.3.6.1.6.3.1.1.5.3' '0.0.0.0' 6 33 '55' .1.3.6.1.6.3.1.1.5.3 s "teststring000"
2. The received trap:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - ZBXTRAP 127.0.0.1 127.0.0.1
3. Value for item TEST:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - 127.0.0.1

Note:
This simple example uses SNMPTT as **traphandle**. For better performance on production systems, use embedded Perl to pass traps from snmptrapd to SNMPTT or directly to Zabbix.

7 See also

- Zabbix blog article on SNMP traps
- CentOS based SNMP trap tutorial on zabbix.org

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：

- 选择'IPMI agent' 作为 类型
- 在主机中输入唯一的监控项键值（例如，ipmi.fan.rpm）
- 对于 主机接口，选择 IPMI 接口（IP 和端口）。注意 IPMI 接口在主机上必须存在。

- 指定 *IPMI* 传感器（例如在 Dell Poweredge 型号服务器上的'FAN MOD 1A RPM'）用于检索对应的指标。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。

* 选择相应的信息类型（'浮点数'，对于离散传感器 - '数字(无正负)')，单位(类似'rpm')和任何其它必需监控项属性

支持的检查项

下表描述 IPMI agent 内置监控项支持的检查。

监控项键值			
▲ ipmi.get	描述返 IPMI 传感器相关信息。JSON 对象	值备注 此监控项可用于 [自	 发现 IPMI 传感器](/zh/manual/disc 从 Zabbix 5.0.0. 开始支持

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

概述

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 监控项键值

键值	描述 *	返回值 ** **	* 注解
		参数	
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]			

键值				
	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>]				
	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**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）	cks#icmp_pings) 表.
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]				

键值					
		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。

Note that for SourceIP support in LDAP simple checks (since Zabbix 5.0.21), OpenLDAP version 2.6.1 or above is required.

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 ('FpingLocation' 参数) 定义的不匹配，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

此外，Zabbix 使用 fping 参数`-i interval ms`(不要和上表中提到的参数 interval 混淆，它对应于 fping 参数-p) 和`-S source IP address`(或者旧 fping 版本的选项-I)。这些参数通过使用不同的参数组合运行自动检测。Zabbix 尝试检测 fping 允许与-i 参数一起使用的最小值 (以毫秒为单位)，尝试 3 个值:0、1 和 10。第一个成功的值将用于后续的 ICMP 检查。这个过程是由每个ICMP pinger 进程单独完成的。

从 Zabbix 5.0.4 版本开始，fping 自动检测的参数每小时都会失效，并且在下一次尝试执行 ICMP 检查时再次加载。设置 DebugLevel>=4 可以在服务器或代理日志文件中查看该进程的详细信息。

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监控项 IP 才会被写入相同的单个文件。
所有写入到单个文件的 IP 地址将被 fping 并行检查，因此 Zabbix icmp pinger 进程将花费固定的时间来处理监控项，而不管文件中的 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>]			

键值			
	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.datastore.discovery[<url>]			
	发现 VMware 的所有数据存储 (Datas-tore) JSON 对象	**url** - VM	are 服务的 URL 地址
vmware.datastore.hv.list[<url>,<datastore>]			
	VMware 虚拟管理器数据存储列表字符串	**url** - VMw	re 服务的 URL 地址 **datastore** - 数据存储名称
vmware.datastore.read[<url>,<datastore>,<mode>]			
	从数据存储执行读取操作的总时间 (毫秒) 整型 ^**[2](vmware	keys#footnotesUR url** - VMware 服务	L^地址 **datastore** - 数据存储名称 **mode** - latency (平均值，默认)，maxla-tency(最大值)
vmware.datastore.size[<url>,<datastore>,<mode>]			
	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 **dat	<store** - 数据存储名称 **mode** - 可能的值: total (默认), free, pfree (剩余百分比), uncom-mitted
vmware.datastore.write[<url>,<datastore>,<mode>]			
	从数据存储执行写入操作的总时间 (毫秒). 整型 ^**[2](vmware	keys#footnotesUR url** - VMware 服务	L^地址 **datastore** - 数据存储名称 **mode** - latency (平均值，默认)，maxla-tency(最大值)
vmware.dc.discovery[<url>]			

键值				
	发现 VMware 的所有数据中心（Datacenter）JSON 对象	**url** - VM	are 服务的 URL 地址 从 Zabbix 5.0.3 开始支	
vmware.eventlog[<url>,<mode>]				
	VMware 事件日志. Log	**u	l** - VMware 服务的 URL 地址 另请参考: 筛选 **mode** - all(默认)，skip - 跳过旧数据的处理	Mware 事件日志记录的示例
vmware.fullname[<url>]				
	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>]				
	VMware 虚拟管理器数据存储名称. JSON 对象	**url** - VM	are 服务的 URL 地址 **uuid** - VMware 虚拟机管理程序主机名	
vmware.hv.datastore.list[<url>,<uuid>]				

键值			
	VMware 虚拟管理器数据存储列表. JSON 对象	**url** - VM	are 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>]			
	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#footnotes)^ url** - VMware 服务的	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.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>]			

键值			
	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
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>]			

键值			
	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>]			
	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 ^**[2](vmware_ke	s#footnotes) 地址 url** - VMware 服务的 UR	**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 ^**[2](vmware_ke	s#footnotes) 地址 url** - VMware 服务的 UR	**uuid** - VMware 虚拟机管理程序主机名 **mode** - bps (默认)
vmware.hv.perfcounter[<url>,<uuid>,<path>,<instance>]			

键值				
	VMware 虚拟机管理程序性能计数器值. 整型 ^**[2](vm	are_keys#foo[illegible] url** - VMwa	服务的 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>]				
	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>]				

键值			
	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>]			
	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>]			

键值			
	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>]			
	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

键值				
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#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.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>]				

键值					
	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		
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	s#footnotes)^**[2](vmware_ke	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>]					

键值				
	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>]				
	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

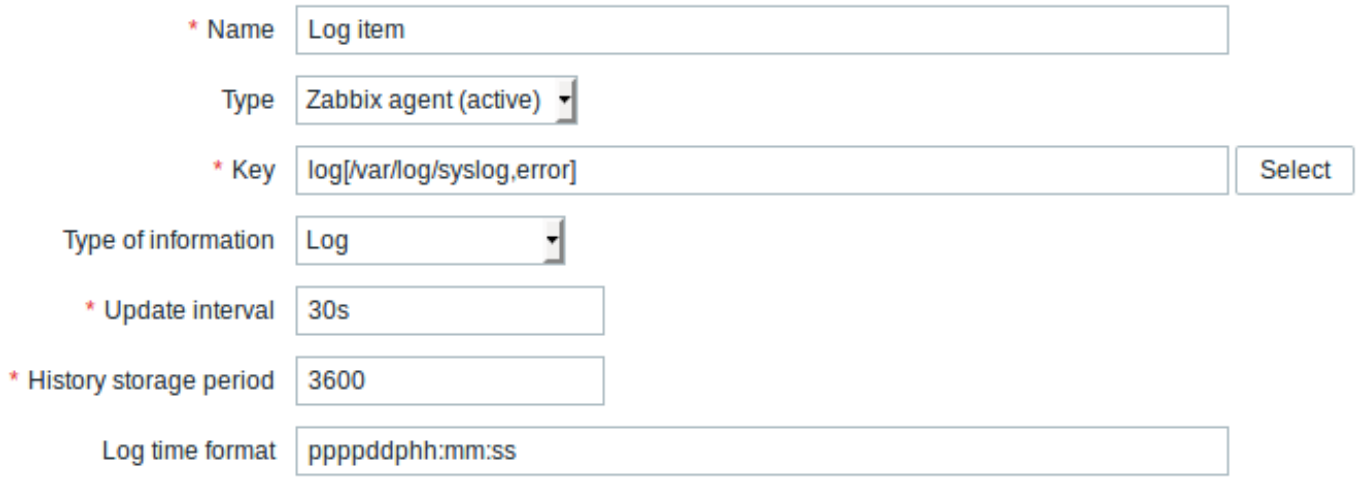
验证代理 (zabbix_agentd) 参数

确保在代理 (zabbix_agentd) 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：

类型这	选择 **Zabbix agent (active)**
Key	使用以下监控项键中的一个进行配置: **log**[] 或 **logrt**[] 这两个监控项键允许监视日志并按正则方式过滤日志内容。例如：log[/var/log/syslog，error]。确保该文件对 "zabbix" 用户具有读取权限，否则项目状态将设置为 "unsupported"。 **log.count**[] 或 **logrt.count**[]. 使用这些监控项键及其参数的详细信息，请参见支持的Zabbix agent 监控项的键值部分。
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 (agent) 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(agent)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
    * 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(agent)从上次停止的点开始读取日志文件。
* 在代理(agent)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
* 每当日志文件变得小于代理(agent)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
* 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 [[zh:manua
* 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。

Purpose of persistent files

When Zabbix agent is started it receives a list of active checks from Zabbix server or proxy. For log*[] metrics it receives the processed log size and the modification time for finding where to start log file monitoring from. Depending on the actual log file size and modification time reported by file system the agent decides either to continue log file monitoring from the processed log size or re-analyze the log file from the beginning.

A running agent maintains a larger set of atributes for tracking all monitored log files between checks. This in-memory state is lost when the agent is stopped.

The new optional parameter **persistent_dir** specifies a directory for storing this state of log[], log.count[], logrt[] or logrt.count[] item in a file. The state of log item is restored from the persistent file after the Zabbix agent is restarted.

The primary use-case is monitoring of log file located on a mirrored file system. Until some moment in time the log file is written to both mirrors. Then mirrors are split. On the active copy the log file is still growing, getting new records. Zabbix agent analyzes it and sends processed logs size and modification time to server. On the passive copy the log file stays the same, well behind the active copy. Later the operating system and Zabbix agent are rebooted from the passive copy. The processed log size and modification time the Zabbix agent receives from server may not be valid for situation on the passive copy. To continue log file monitoring from the place the agent left off at the moment of file system mirror split the agent restores its state from the persistent file.

Agent operation with persistent file

On startup Zabbix agent knows nothing about persistent files. Only after receiving a list of active checks from Zabbix server (proxy) the agent sees that some log items should be backed by persistent files under specified directories.

During agent operation the persistent files are opened for writing (with fopen(filename, ”w”)) and overwritten with the latest data. The chance of losing persistent file data if the overwriting and file system mirror split happen at the same time is very small, no special handling for it. Writing into persistent file is NOT followed by enforced synchronization to storage media (fsync() is not called).

Overwriting with the latest data is done after successful reporting of matching log file record or metadata (processed log size and modification time) to Zabbix server. That may happen as often as every item check if log file keeps changing.

After receiving a list of active checks the agent scans persistent file directory and removes obsolete persistent files. Removing is done with delay 24 hours because log files in NOTSUPPORTED state are not included in the list of active checks but they may become SUPPORTED later and their persistent files will be useful. If agent is stopped and started again before 24 hours expire, then the obsolete files will not be deleted as Zabbix agent is not getting info about their location from Zabbix server anymore.

No special actions during agent shutdown.

After receiving a list of active checks the agent marks obsolete persistent files for removal. A persistent file becomes obsolete if: 1) the corresponding log item is no longer monitored, 2) a log item is reconfigured with a different **persistent_dir** location than before.

Removing is done with delay 24 hours because log files in NOTSUPPORTED state are not included in the list of active checks but they may become SUPPORTED later and their persistent files will be useful.

If the agent is stopped before 24 hours expire, then the obsolete files will not be deleted as Zabbix agent is not getting info about their location from Zabbix server anymore.

Warning:
Reconfiguring a log item’s **persistent_dir** back to the old **persistent_dir** location while the agent is stopped, without deleting the old persistent file by user - will cause restoring the agent state from the old persistent file resulting in missed messages or false alerts.

Naming and location of persistent files

Zabbix agent distinguishes active checks by their keys. For example, logrt[/home/zabbix/test.log] and logrt[/home/zabbix/test.log,] are different items. Modifying the item logrt[/home/zabbix/test.log„,10] in frontend to logrt[/home/zabbix/test.log„,20] will result in deleting the item logrt[/home/zabbix/test.log„,10] from the agent’s list of active checks and creating logrt[/home/zabbix/test.log„,20] item (some attributes are carried across modification in frontend/server, not in agent).

The file name is composed of MD5 sum of item key with item key length appended to reduce possibility of collisions. For example, the state of logrt[/home/zabbix50/test.log„„„„„/home/zabbix50/agent_private] item will be kept in persistent file c963ade4008054813bbc0a650bb8e09266.

Multiple log items can use the same value of **persistent_dir**.

persistent_dir is specified by taking into account specific file system layouts, mount points and mount options and storage mirroring configuration - the persistent file should be on the same mirrored filesystem as the monitored log file.

If **persistent_dir** directory cannot be created or does not exist, or access rights for Zabbix agent does not allow to create/write/read/delete files the log item becomes NOTSUPPORTED.

If access rights to persistent storage files are removed during agent operation or other errors occur (e.g. disk full) then errors are logged into the agent log file but the log item does not become NOTSUPPORTED.

Load on I/O

Item's persistent file is updated after successful sending of every batch of data (containing item's data) to server. For example, default 'BufferSize' is 100. If a log item has found 70 matching records then the first 50 records will be sent in one batch, persistent file will be updated, then remaining 20 records will be sent (maybe with some delay when more data is accumulated) in the 2nd batch, and the persistent file will be updated again.

Actions if communication fails between agent and server

Each matching line from `log[]` and `logrt[]` item and a result of each `log.count[]` and `logrt.count[]` item check requires a free slot in the designated 50% area in the agent send buffer. The buffer elements are regularly sent to server (or proxy) and the buffer slots are free again.

While there are free slots in the designated log area in the agent send buffer and communication fails between agent and server (or proxy) the log monitoring results are accumulated in the send buffer. This helps to mitigate short communication failures.

During longer communication failures all log slots get occupied and the following actions are taken:

- `log[]` and `logrt[]` item checks are stopped. When communication is restored and free slots in the buffer are available the checks are resumed from the previous position. No matching lines are lost, they are just reported later.
- `log.count[]` and `logrt.count[]` checks are stopped if `maxdelay = 0` (default). Behavior is similar to `log[]` and `logrt[]` items as described above. Note that this can affect `log.count[]` and `logrt.count[]` results: for example, one check counts 100 matching lines in a log file, but as there are no free slots in the buffer the check is stopped. When communication is restored the agent counts the same 100 matching lines and also 70 new matching lines. The agent now sends count = 170 as if they were found in one check.
- `log.count[]` and `logrt.count[]` checks with `maxdelay > 0`: if there was no "jump" during the check, then behavior is similar to described above. If a "jump" over log file lines took place then the position after "jump" is kept and the counted result is discarded. So, the agent tries to keep up with a growing log file even in case of communication failure.

7 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。

Performance

Using some internal items may negatively affect performance. These items are:

- zabbix[host,,items]
- zabbix[host,,items_unsupported]
- zabbix[hosts]
- zabbix[items]
- zabbix[items_unsupported]
- zabbix[queue]
- zabbix[required_performance]
- zabbix[stats,,,queue]
- zabbix[triggers]

The System information and Queue frontend sections are also affected.

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]				

键值				
	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]				
	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]				

键值				
	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
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]((	zh/manual/discovery/lo 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[lld_queue]				
	在低级发现预处理队列中队列数量	整数	此监控项可用于监控低级发现预处理	列长度 \\此监控项从 Zab-bix**4.2.0.** 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	/zh/manual/web_interfa 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[stats,<ip>,<port>]				

键值					
		远程 Zabbix 服务器或代理内部度量	JSON 对象	**ip** - 要远程查	的 Zabbix 服务器/代理 (proxy) 的 IP/DNS/网络掩码 (NET-MASK) 列表 (默认：127.0.0.1) **port** - 要远程查询的 Zabbix 服务器/代理 (proxy) 的端口号 (默认：10051) 注意被查询Zabbix 服务器/代理 (proxy)只允许配置参数 `StatsAllowedIP` 中指定 IP 列表中的 IP 进行查询 此项返回一组选定的内部度量数据。有关详细信息，请参阅远程监视 Zabbix stats 从 Zabbix **4.2.0.** 开始支持
zabbix[stats,<ip>,<port>,queue,<from>,<to>]					

键值				
	远程 Zabbix 服务器或代理内部队列度量	JSON 对象	**ip** - 要远程查询的	abbix 服务器/代理 (proxy) 的 IP/DNS/网络掩码 (NET-MASK) 列表 (默认：127.0.0.1) **port** - 要远程查询的 Zabbix 服务器/代理 (proxy) 的端口号 (默认：10051) **from** - 至少延迟 (默认：6s) **to** - 最多延迟 (默认：infinity) 注意被查询Zabbix 服务器/代理 (proxy)只允许配置参数 StatsAllowedIP 中指定 IP 列表中的 IP 进行查询 此项返回一组选定的内部度量数据。有关详细信息，请参阅远程监视 Zabbix stats 从 Zabbix **4.2.0.** 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]				

键值				
	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>]				
	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>]				

键值				
	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[version]	Zabbix server 或 zabbix proxy 的版本	字符串	这个监控项从	Zabbix 5.0.0. 开始支持。 返回值样例: 5.0.0beta1
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		
		pused	已用历史缓冲区的百分比浮点数	从 Zabbix 4.0.	. 开始支持
	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		
		pused	已用历史索引缓冲区的百分比浮点数	从 Zabbix 4.0.0.	开始支持
	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	*
		pused	已用历史趋势缓冲区的百分比浮点数	从 Zabbix 4.0.0.	开始支持

9 SSH 检查

概述

SSH 检查不依赖于 Zabbix agent，可对无 agent 代理的设备进行监控。

要执行 SSH 检查，Zabbix 服务器必须首先配置SSH2(libssh2 或 libssh) 支持，也可参见需求文档

> **Attention:**
> 从 RHEL/CentOS 8 开始，只支持 libssh。

配置

密码验证

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。

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）。

密钥文件认证

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。

以 root 身份打开 Zabbix server 的配置文件，查找以下行

```
# SSHKeyLocation=
```

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
```

保存文件并重启 zabbix_server 服务

/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。

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* 。

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。

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* 。

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
```

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
```

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
```

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
```

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
```

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+        |
+-----------------+
```

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。

<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 配置方式

对于每台被 SSH 检测的主机，此步骤只需要执行一次。

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.
```

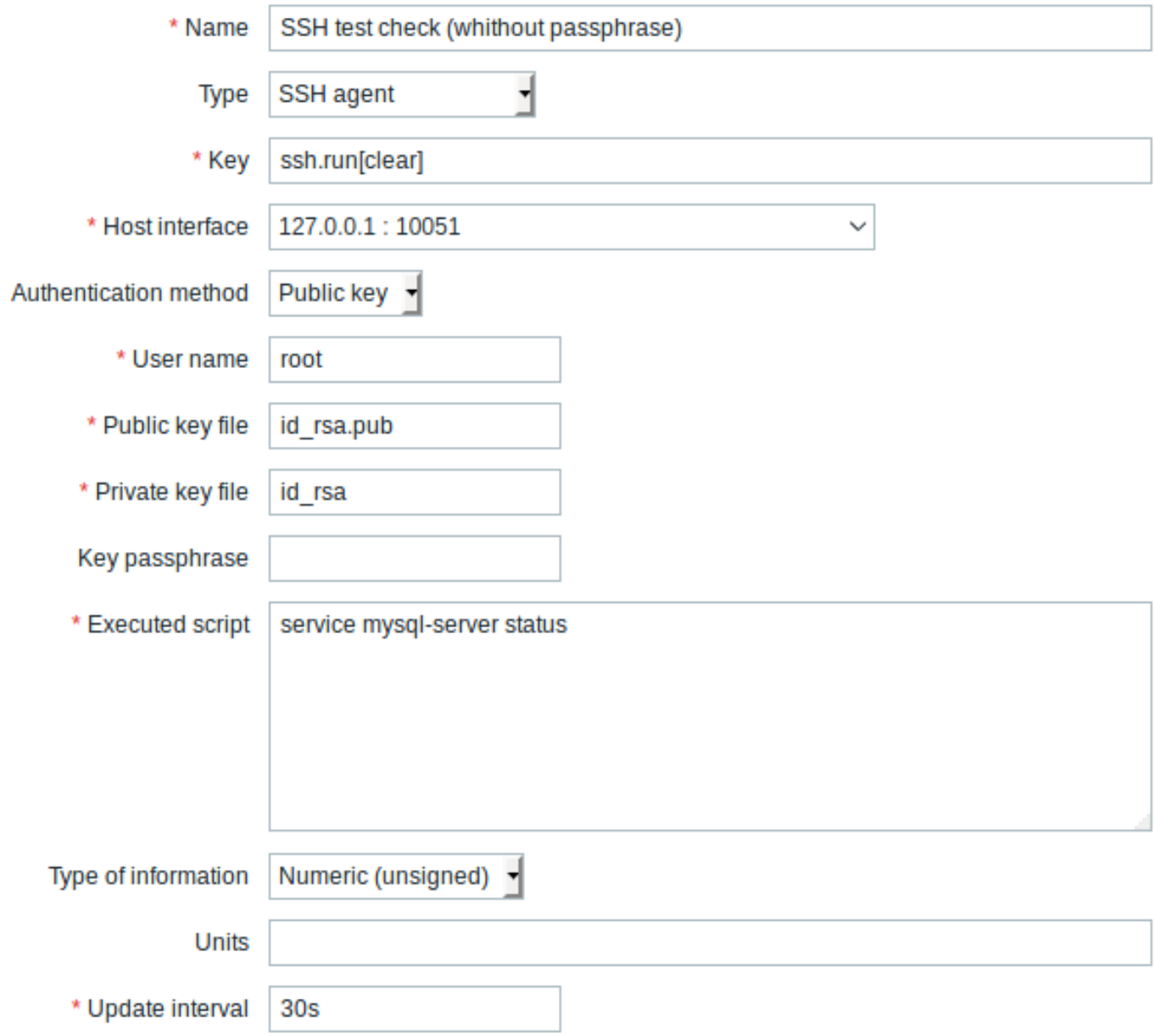
现在可以使用 *zabbix* 用户的默认私钥 (*/home/zabbix/.ssh/id_rsa*) 检查 SSH 登陆了：

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

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。

监控项配置

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。



所有标有红色星号的为必填项。

需要为 SSH 监控项提供特定信息的字段是：

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*

<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：

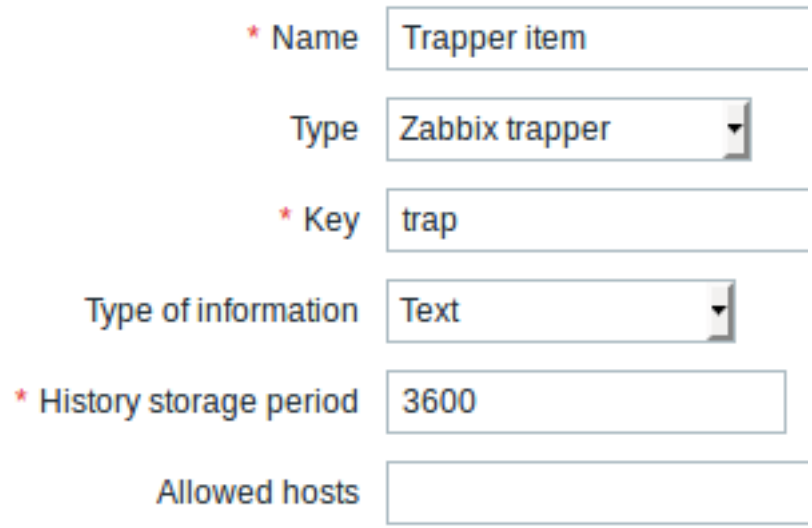
- 在 Zabbix 中建立一个捕捉器监控项
- 将数据发送到 Zabbix

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）

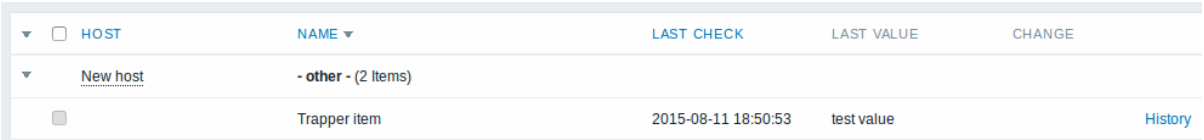
-k - 指定我们之前定义的监控项的键值

-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



请注意，如果发送单个数值，数据图将在该值的时间点的左侧和右侧显示一条水平线。

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。

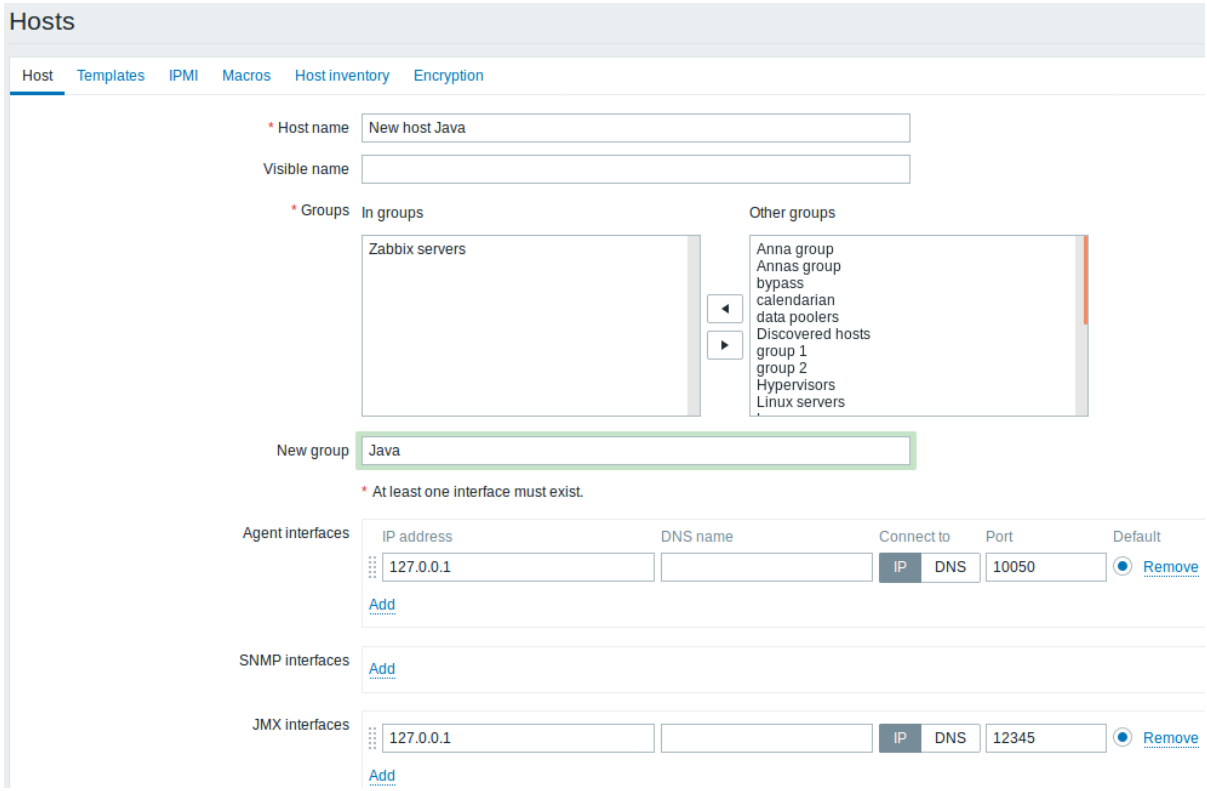
详细说明请参考 使用 JMX 监控和管理 。

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。

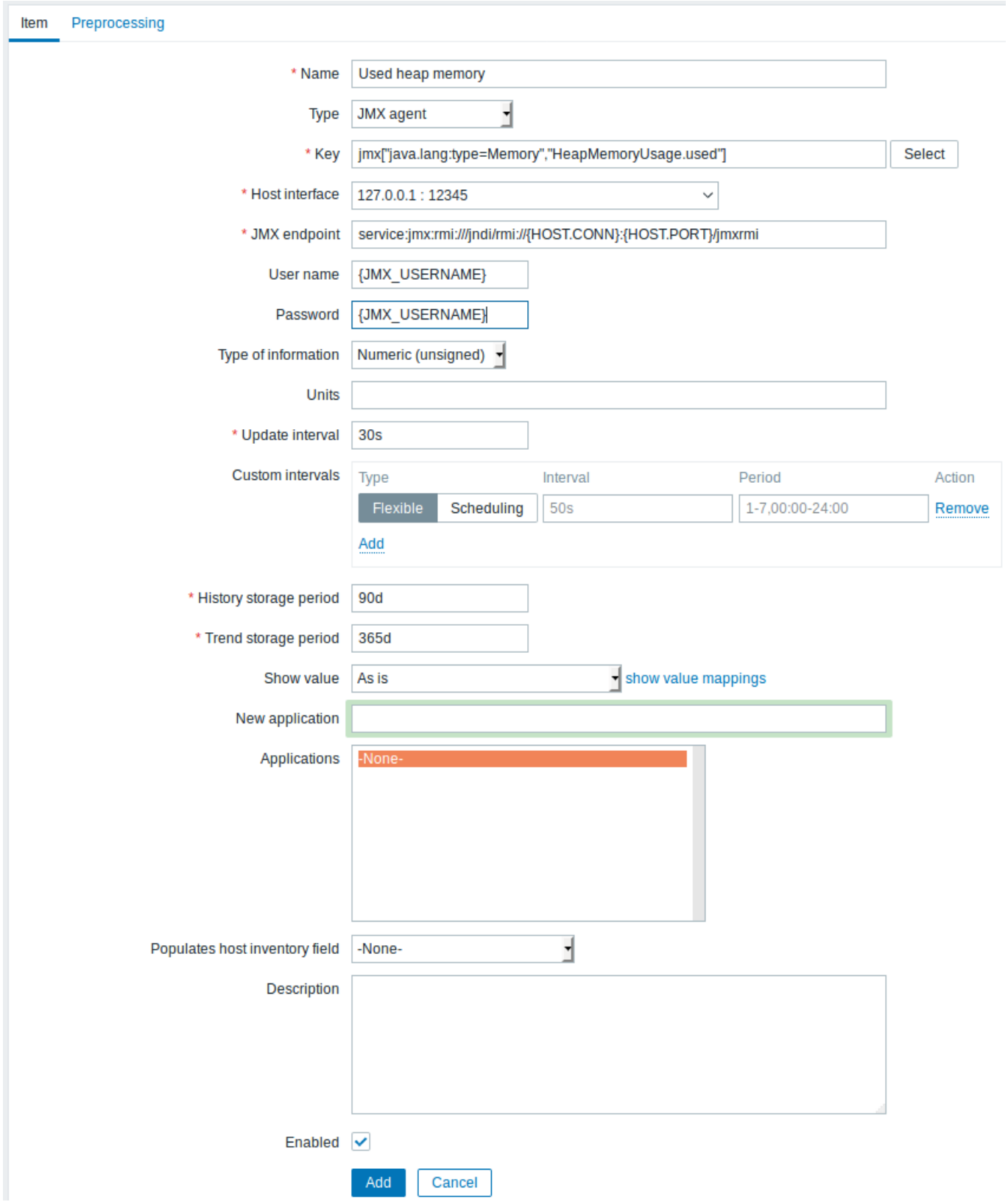


标有红色星号的为必填项。

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]
```

属性返回列表数据

列表数据属性由一个或多个复合属性组成。如果在属性名参数中指定了这样一个属性，那么这个监控项的值将以 JSON 格式返回属性的完整结构。列表数据属性中的单个元素值可以使用预处理进行检索。

列表数据属性数据样例:

```
jmx[com.example:type=Hello,foodinfo]
```

监控项值:

```
[
    {
        "a": "apple",
        "b": "banana",
        "c": "cherry"
    },
    {
        "a": "potato",
        "b": "lettuce",
        "c": "onion"
    }
]
```

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，请参见监控项键值的格式。这就是全部了，祝 JMX 监控快乐！

非基本数据类型

从 Zabbix 4.0.0 开始，可以使用自定义的 *MBeans* 返回非基本数据类型数据，其重写了 toString() 方法。

JBoss EAP 6.4 的自定义端点示例

自定义端点允许使用默认 RMI 以外的其他传输协议。

为了说明这种功能，我们以配置 JBoss EAP 6.4 监控为例。首先，需要做一些前期准备：

```
* 已经安装了 Zabbix Java gateway，如果还没有安装，那么你可以按照帮助文档进行安装。
* Zabbix server 和 Java gateway 被安装在目录 ''/usr/local/''
* JBoss 已经被安装在目录 ''/opt/jboss-eap-6.4/'' 并运行在 ''standalone''模式
* 我们假设所有这些组件都在同一主机上工作
* Firewall 和 SELinux 被关闭或做了相应的配置
```

让我们在配置文件 zabbix_server.conf 中进行一些简单的设置:

```
JavaGateway=127.0.0.1
StartJavaPollers=5
```

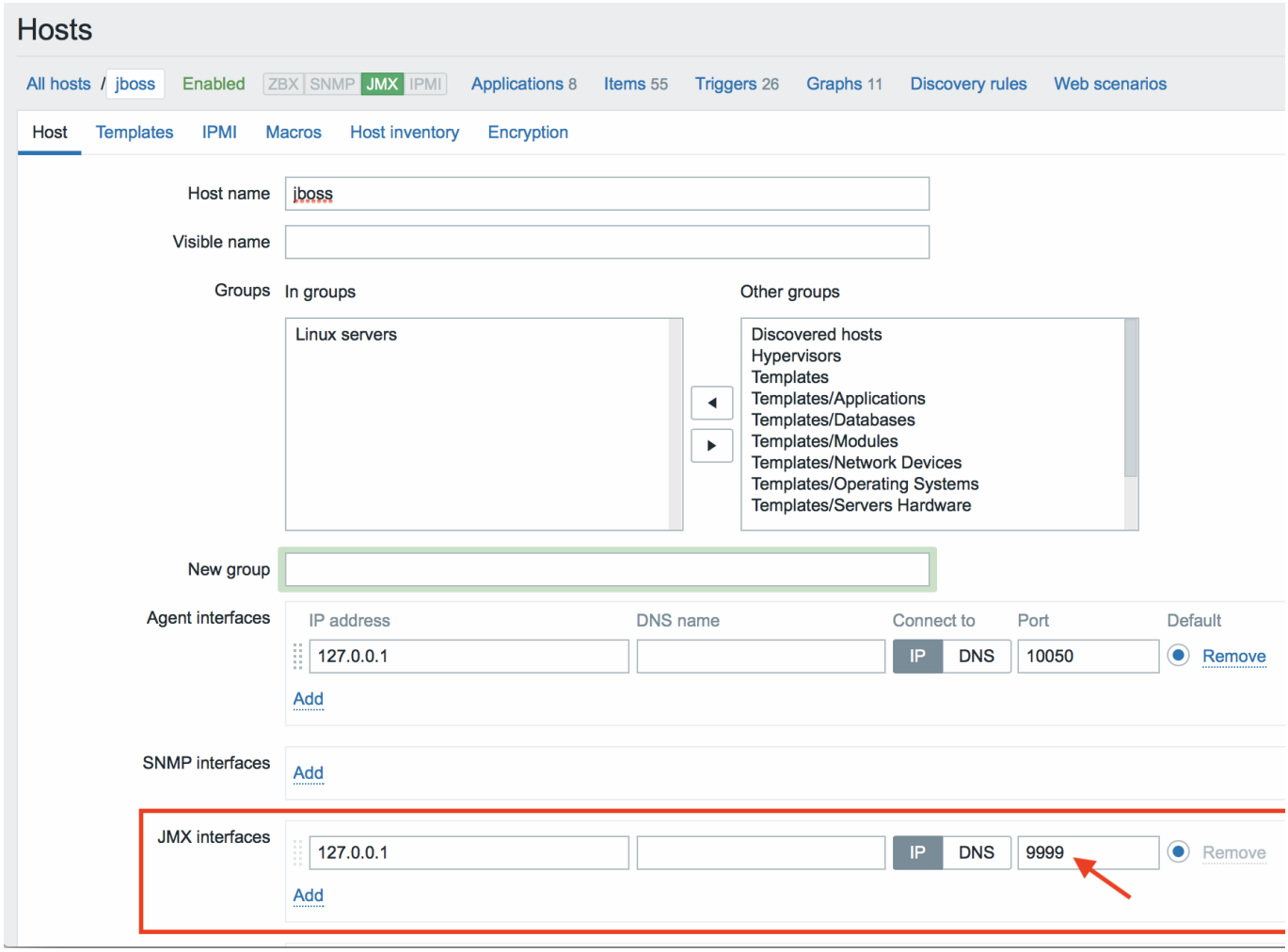
并在配置文件 zabbix_java/settings.sh 或 zabbix_java_gateway.conf 中修改:

```
START_POLLERS=5
```

检查并确认 JBoss 的标准管理端口已经启用：

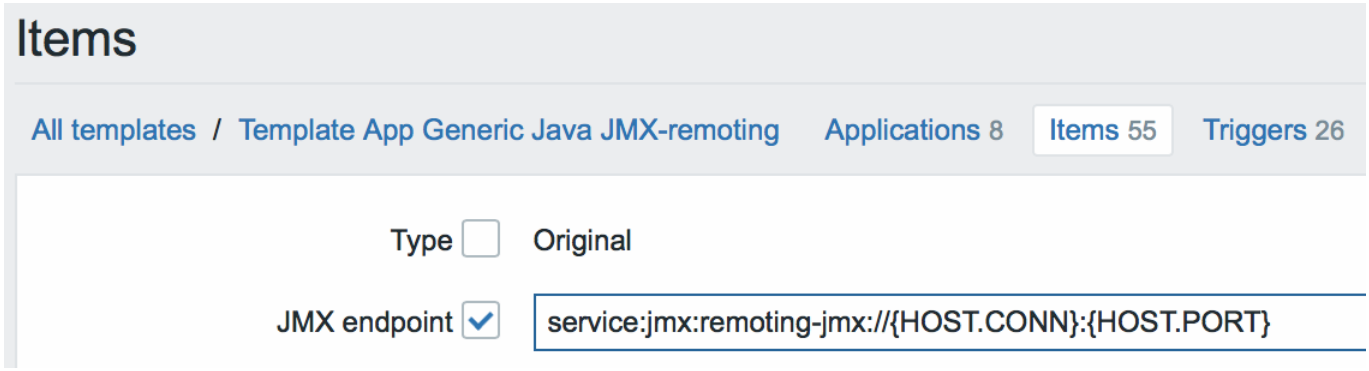
```
$ netstat -natp | grep 9999
tcp        0      0 127.0.0.1:9999          0.0.0.0:*               LISTEN      10148/java
```

现在让我们在 Zabbix 服务器中创建一个配置有 JMX 接口（127.0.0.1:9999）的主机。



若我们知道当前 JBoss 版本使用的是 JBoss Remoting 协议而不是 RMI 时，我们需要相应地批量更新 JMX 模板中的 *JMX endpoint* 参数：

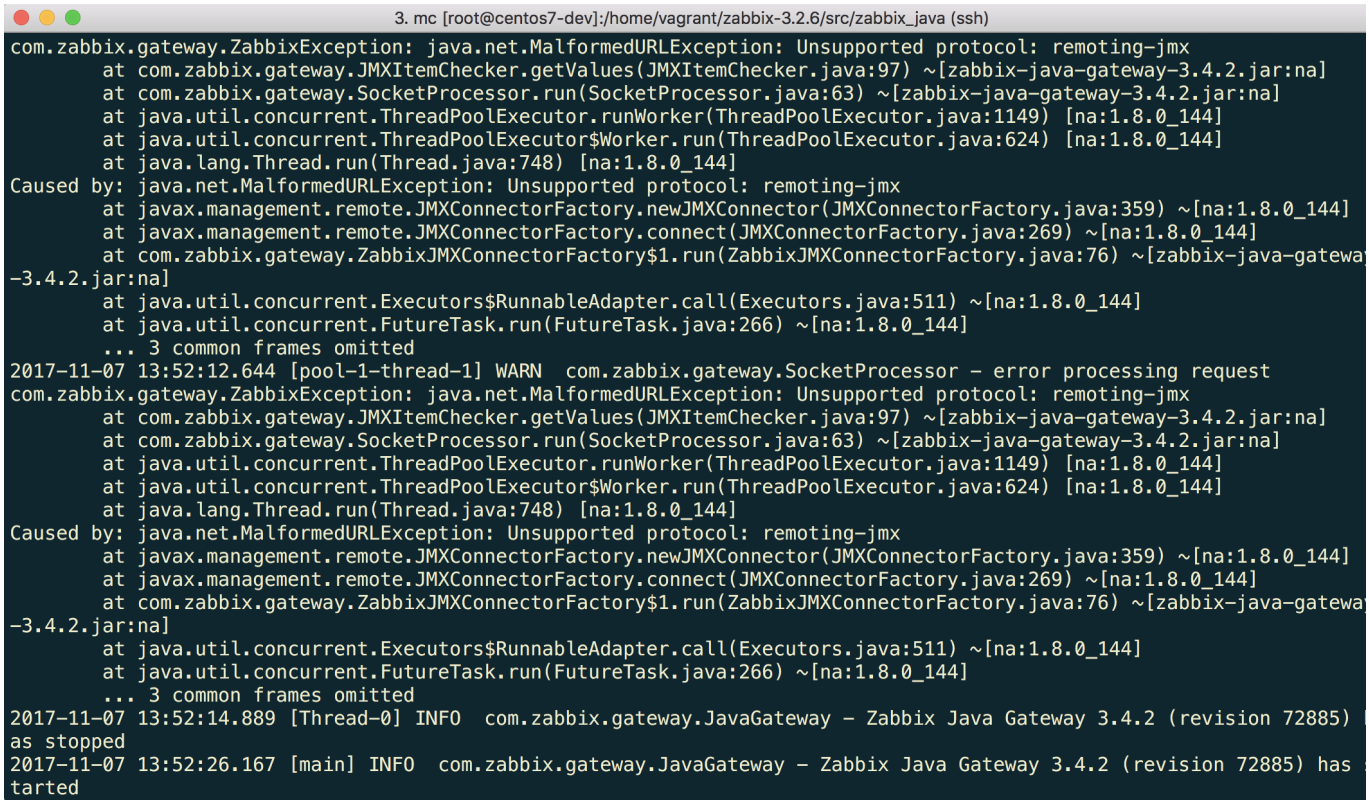
```
service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}
```



更新配置缓存:

```
$ /usr/local/sbin/zabbix_server -R config_cache_reload
```

注意，您可能首先会遇到如下错误.



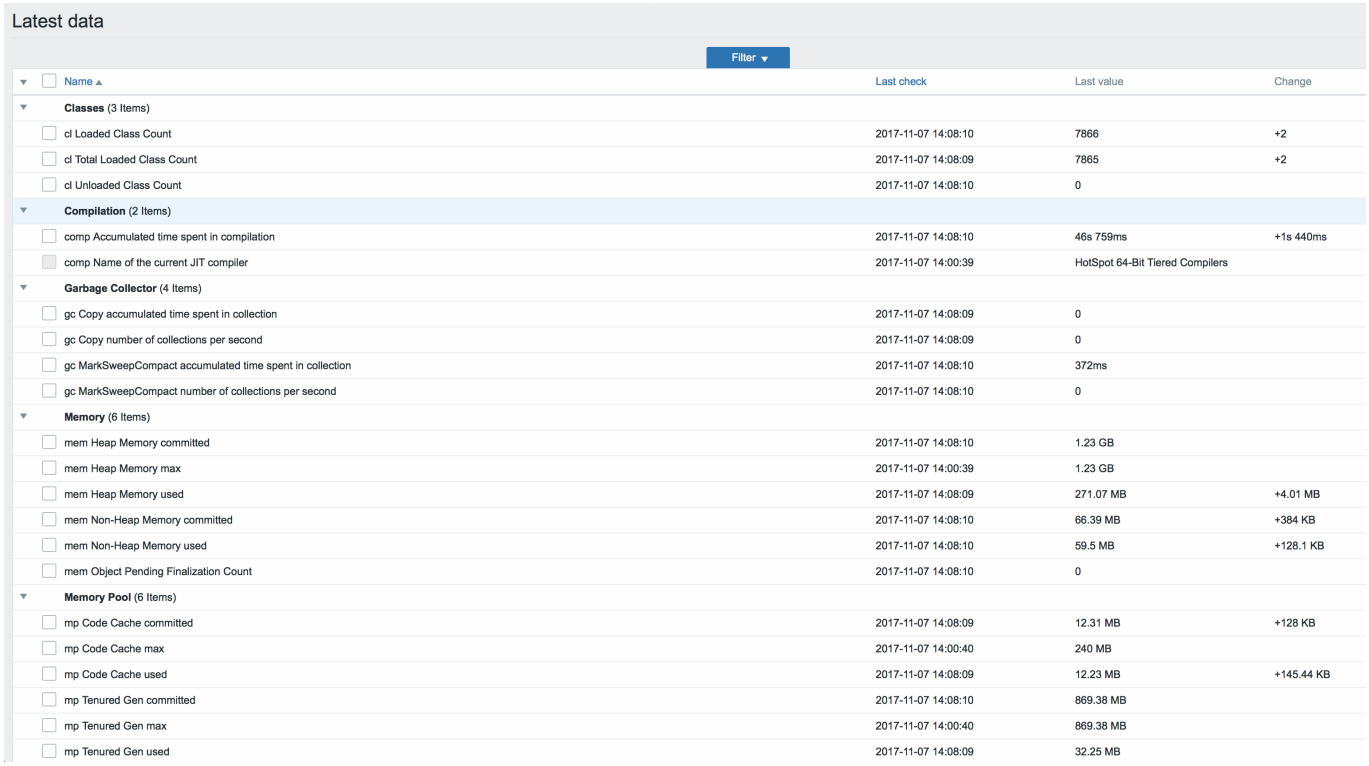
“Unsupported protocol: remoting-jmx” 意味着 *Java gateway* 不知道如何使用指定的协议。这可以通过创建一个包含以下内容的 `~/needed_modules.txt` 文件解决问题：

```
  jboss-as-remoting
  jboss-logging
  jboss-logmanager
  jboss-marshalling
  jboss-remoting
  jboss-sasl
  jcl-over-slf4j
  jul-to-slf4j-stub
  log4j-jboss-logmanager
  remoting-jmx
  slf4j-api
  xnio-api
  xnio-nio</pre>
```

然后执行命令:

```
  $ for i in $(cat ~/needed_modules.txt); do find /opt/jboss-eap-6.4 -iname ${i}*.jar -exec cp {} /usr/loc
```

这样，*Java gateway* 将拥有使用 jmx 远程处理所需的所有模块。剩下的就是重新启动 *Java gateway* ，如果一切都做对了，稍候就会在 Zabbix 中看到 JMX 的监控数据了：



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

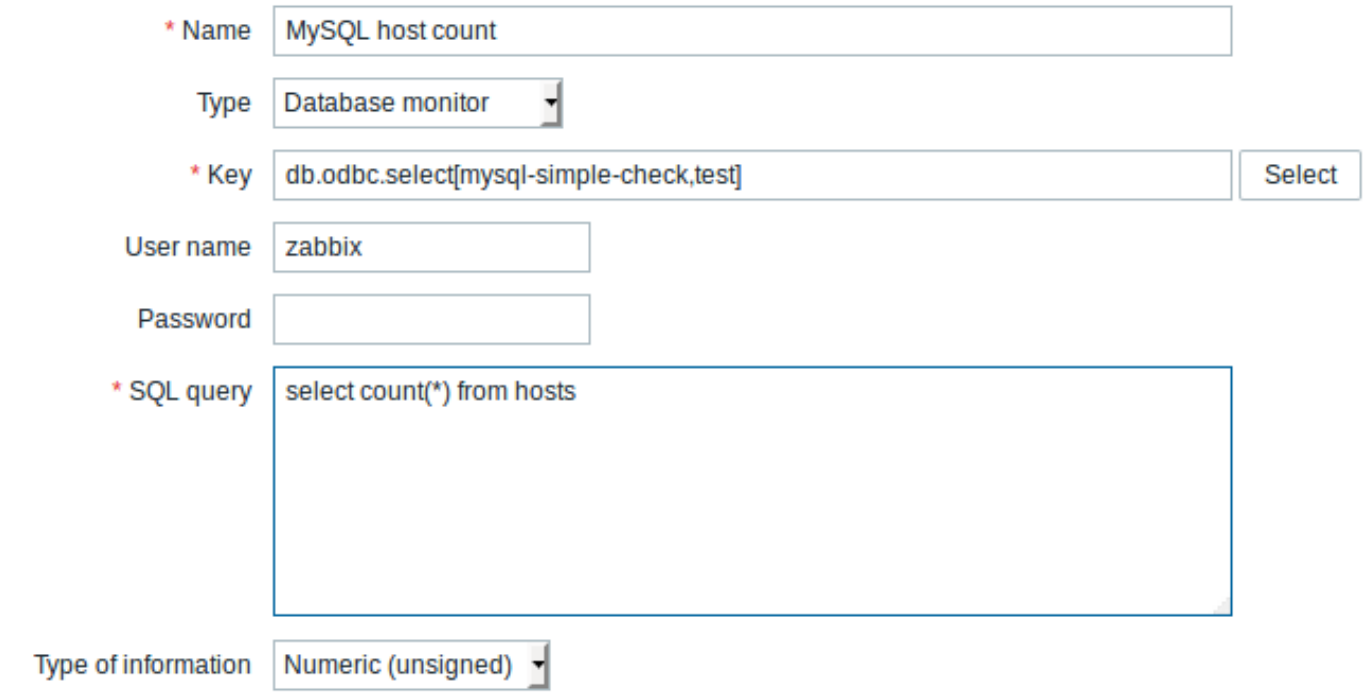
要启用 ODBC 支持，Zabbix 应该使用以下标志进行编译：

```
  --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 或 Zabbix proxy (如果主机被 Zabbix proxy 监控) 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

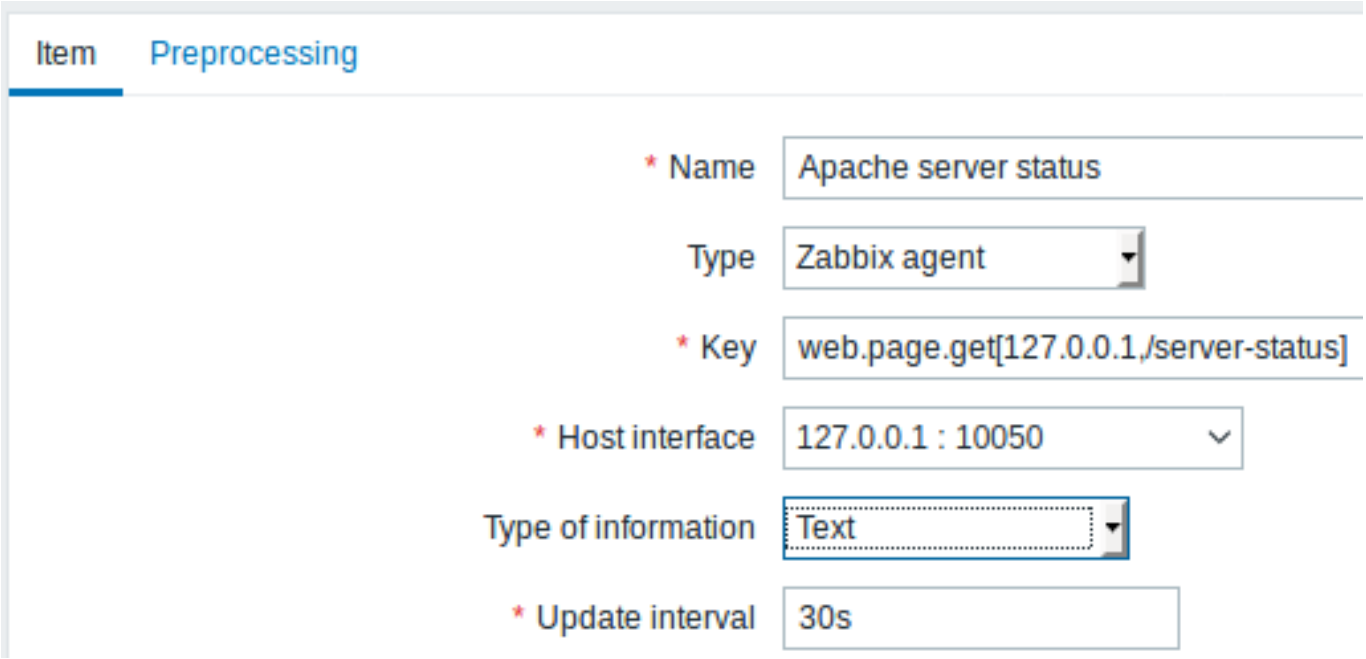
- 只允许相同的主机（模板）从属项
- 监控项原型可以依赖于来自同一主机的另一个监控项原型或常规监控项

- 主监控项的从属项最大计数被限制为 29999（不考虑依赖级别的数量）
- 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)

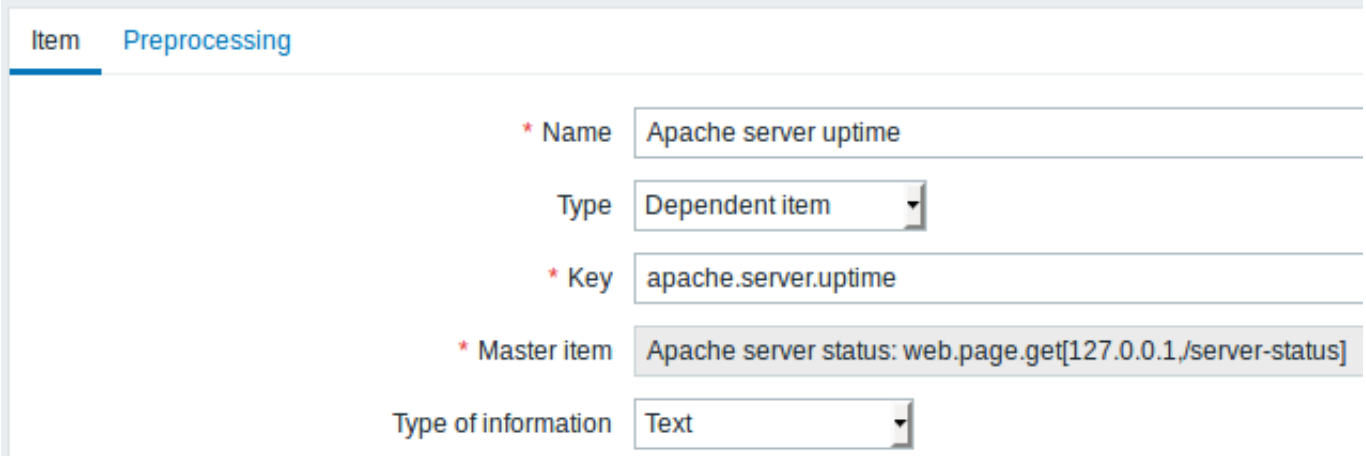
- 进入: *Configuration* → *Hosts*
- 在主机那一行点击 *Items*
- 点击 *Create item*
- 下表中输入监控项的参数



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

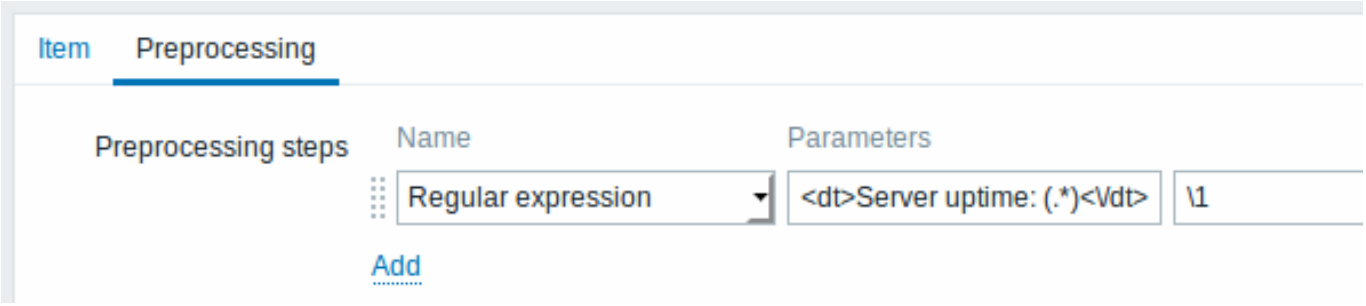


所有标有红色星号的为必填字段。

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

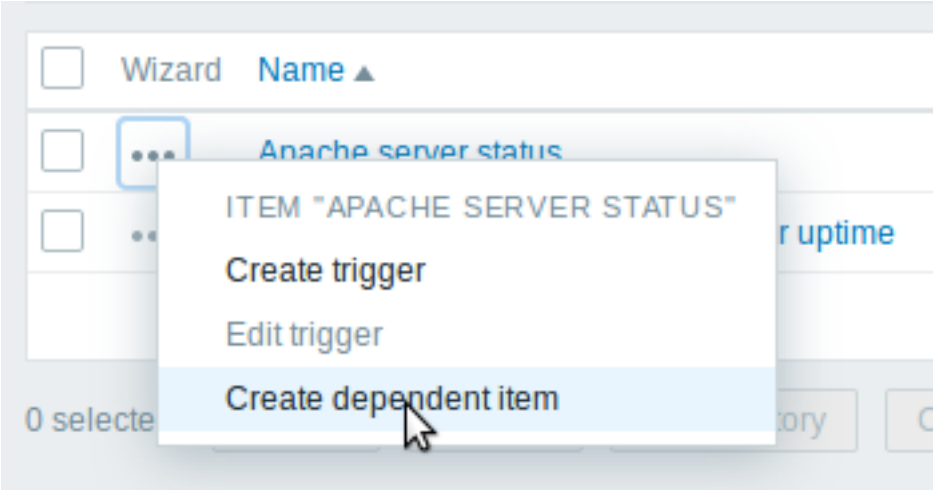
你可以使用监控项的值预处理 来提取主监控项值的所需部分。



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。

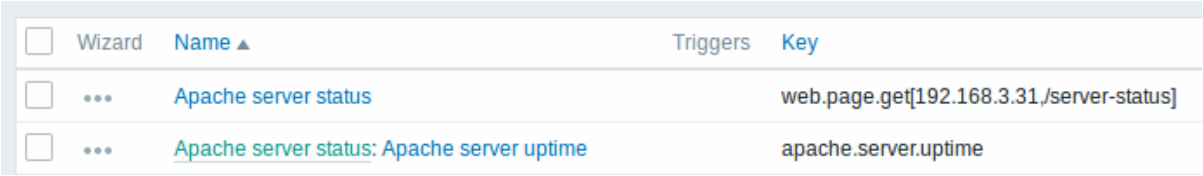
点击 *ADD* 保存从属监控项。

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 监控项检查由 Zabbix 服务器执行。但是，当主机由 Zabbix proxy 监控时，HTTP 项检查由 proxy 执行。

HTTP 监控项检查不需要任何 agent 运行在被监控的主机上。

HTTP agent 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）。最大重定向数硬编码为 10（用 cURL 的参数 CURLOPT_MAXREDIRS）

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: `{ "header": { "<key>": "<value>", "<key2>": "<value>" }, "body": <body> }`
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
                    }
                }
            }]
        }
    }
}
```

- 接收：

```
{
    "_scroll_id": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAAJRZZUTJWQVktVFN
    "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
            }
```

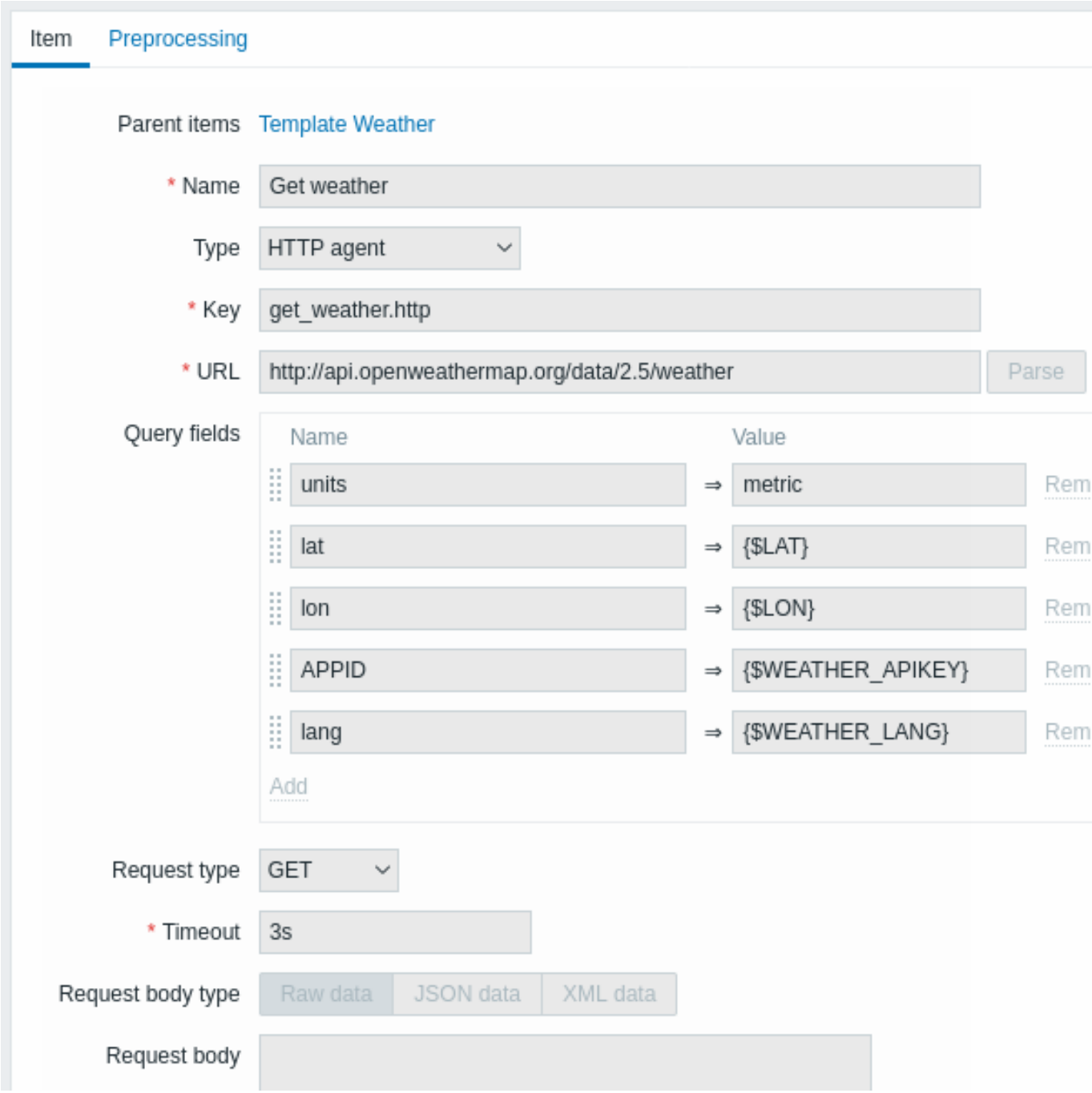
```
        }]
    }
}
```

- 现在使用 JSONPath 预处理步骤获取项值：`$.hits.hits[0]._source.value`

Example 4

Retrieving weather information by connecting to the Openweathermap public service.

- Configure a master item for bulk data collection in a single JSON:



Note the usage of macros in query fields. Refer to the Openweathermap API for how to fill them.

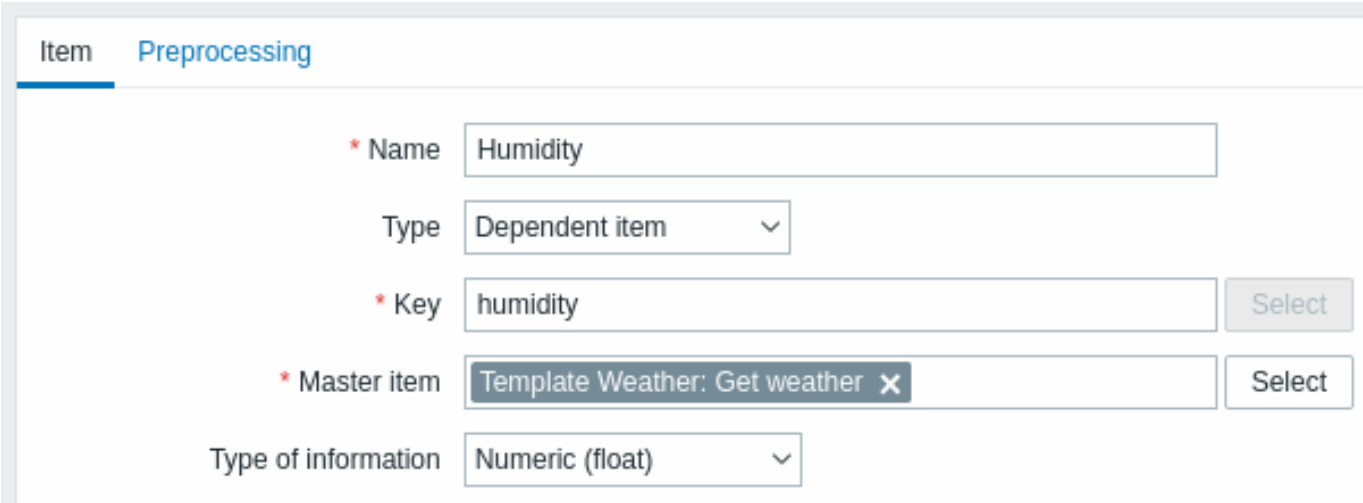
Sample JSON returned in response to HTTP agent:

```
{
    "body": {
        "coord": {
            "lon": 40.01,
            "lat": 56.11
        },
        "weather": [{
```

```
            "id": 801,
            "main": "Clouds",
            "description": "few clouds",
            "icon": "02n"
        }],
        "base": "stations",
        "main": {
            "temp": 15.14,
            "pressure": 1012.6,
            "humidity": 66,
            "temp_min": 15.14,
            "temp_max": 15.14,
            "sea_level": 1030.91,
            "grnd_level": 1012.6
        },
        "wind": {
            "speed": 1.86,
            "deg": 246.001
        },
        "clouds": {
            "all": 20
        },
        "dt": 1526509427,
        "sys": {
            "message": 0.0035,
            "country": "RU",
            "sunrise": 1526432608,
            "sunset": 1526491828
        },
        "id": 487837,
        "name": "Stavrovo",
        "cod": 200
    }
}
```

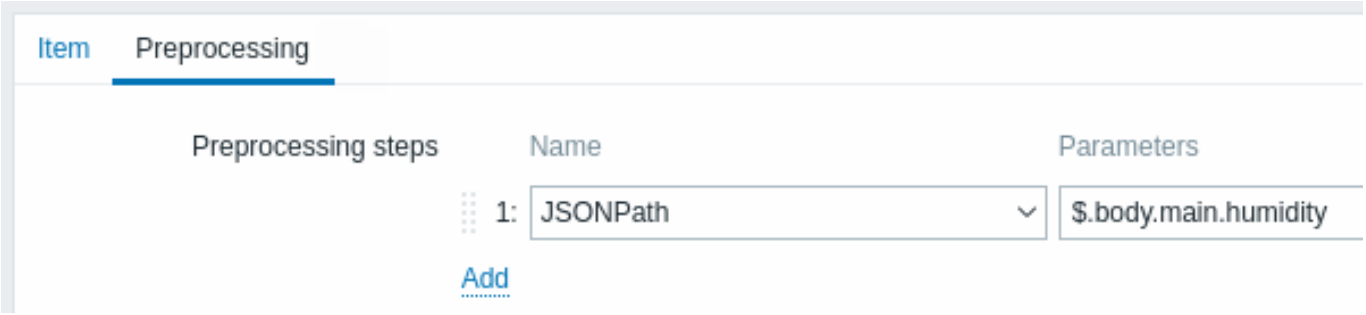

The next task is to configure dependent items that extract data from the JSON.

- Configure a sample dependent item for humidity:

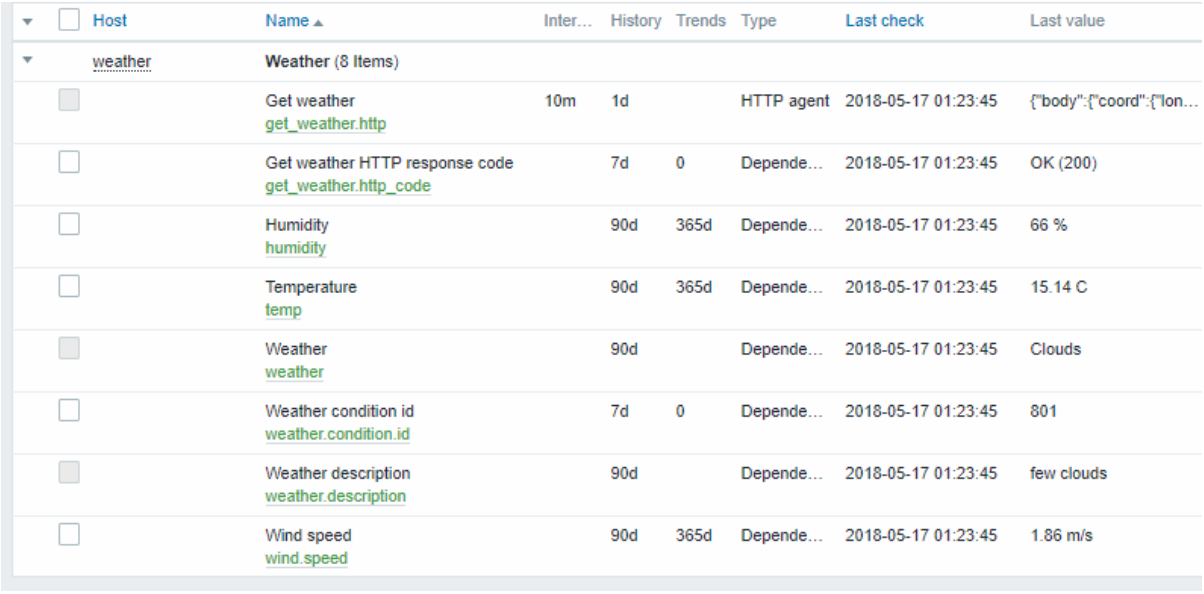


Other weather metrics such as 'Temperature' are added in the same manner.

- Sample dependent item value preprocessing with JSONPath:



- Check the result of weather data in *Latest data*:

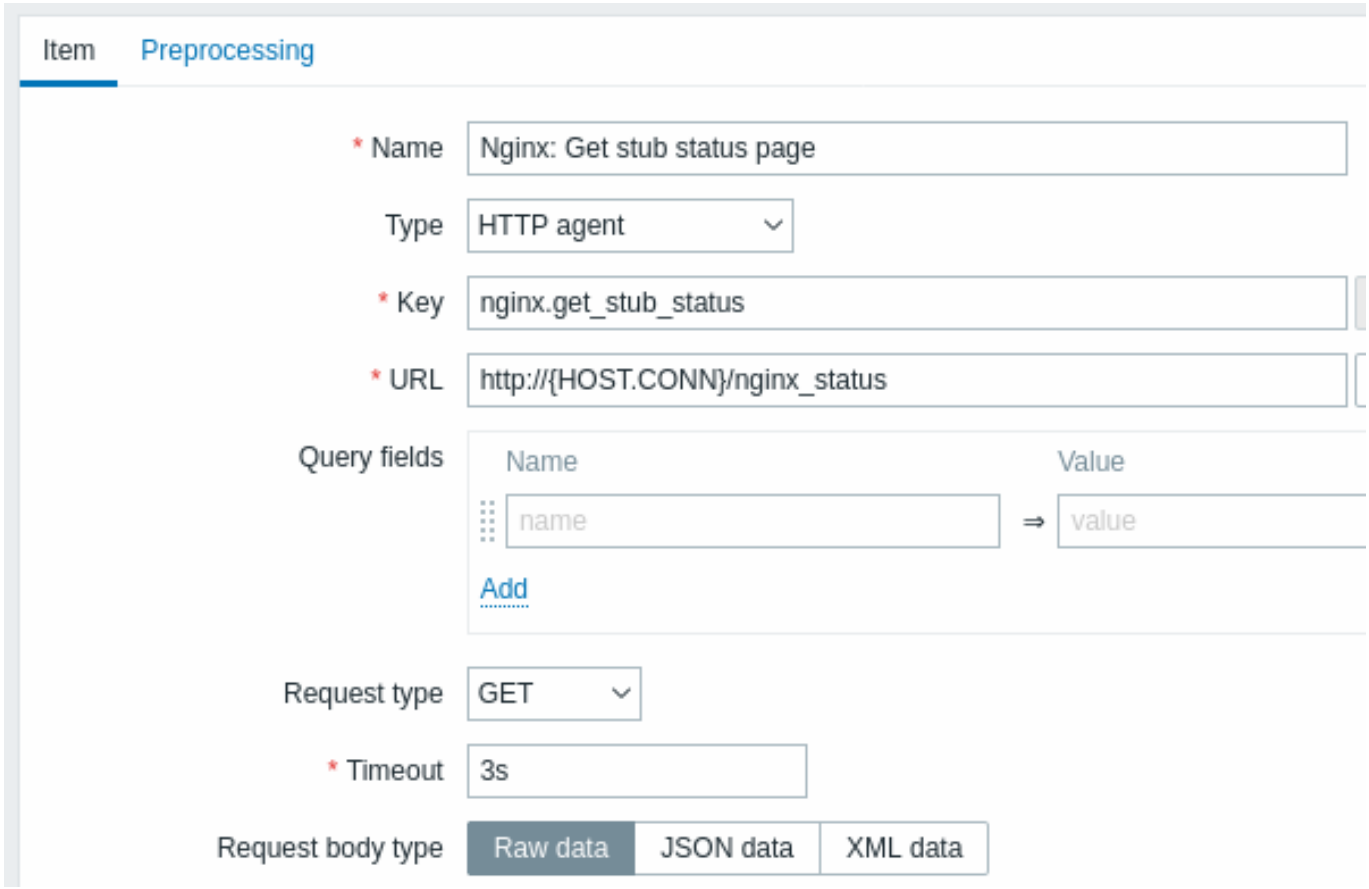


Host	Name ▲	Inter...	History	Trends	Type	Last check	Last value
weather	**Weather** (8 Items)						
	Get weather get_weather.http	10m	1d		HTTP agent	2018-05-17 01:23:45	{"body":{"coord":{"lon...
	Get weather HTTP response code get_weather.http_code		7d	0	Depende...	2018-05-17 01:23:45	OK (200)
	Humidity humidity		90d	365d	Depende...	2018-05-17 01:23:45	66 %
	Temperature temp		90d	365d	Depende...	2018-05-17 01:23:45	15.14 C
	Weather weather		90d		Depende...	2018-05-17 01:23:45	Clouds
	Weather condition id weather.condition.id		7d	0	Depende...	2018-05-17 01:23:45	801
	Weather description weather.description		90d		Depende...	2018-05-17 01:23:45	few clouds
	Wind speed wind.speed		90d	365d	Depende...	2018-05-17 01:23:45	1.86 m/s

Example 5

Connecting to Nginx status page and getting its metrics in bulk.

- Configure Nginx following the official guide.
- Configure a master item for bulk data collection:

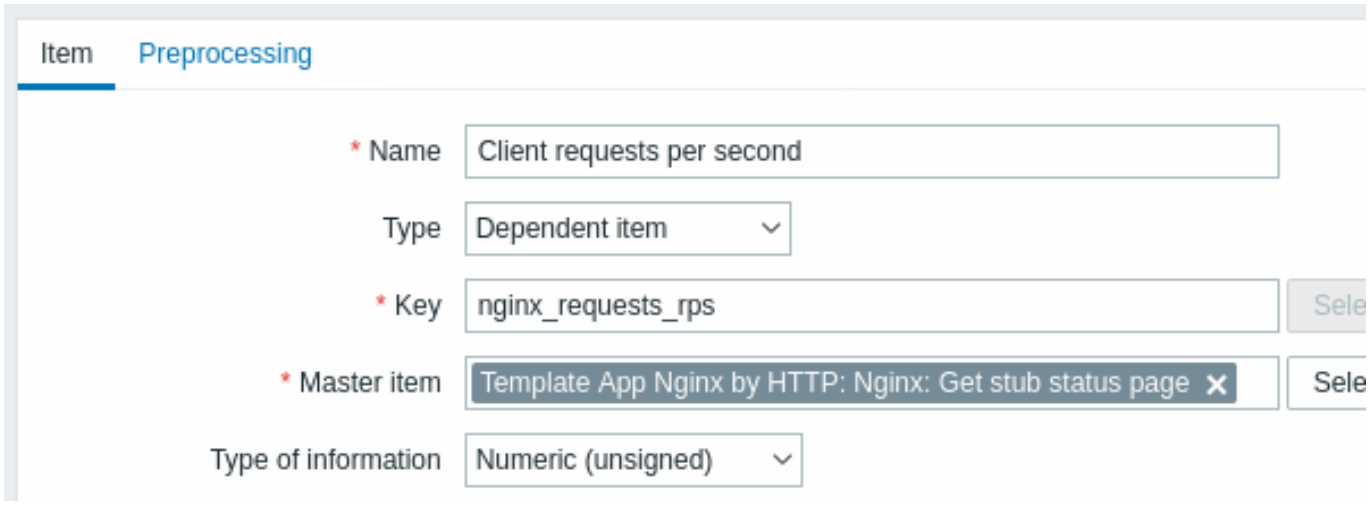


Sample Nginx stub status output:

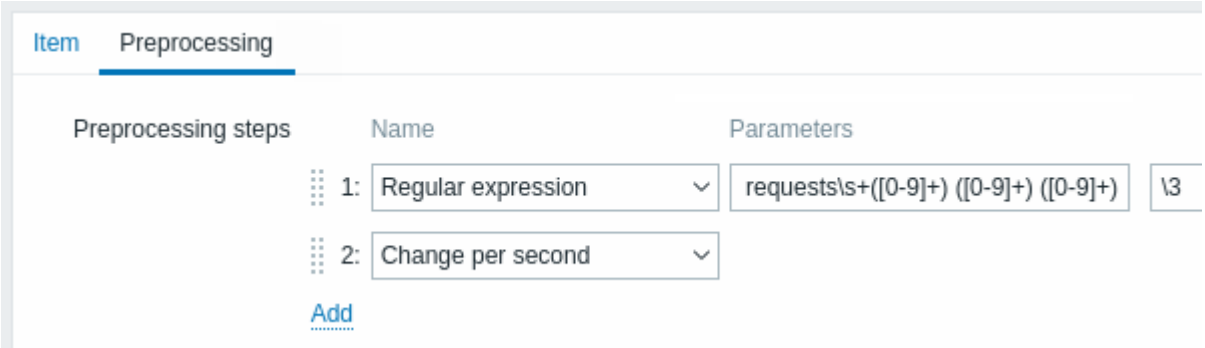
```
Active connections: 1 Active connections:
server accepts handled requests
 52 52 52
Reading: 0 Writing: 1 Waiting: 0
```

The next task is to configure dependent items that extract data.

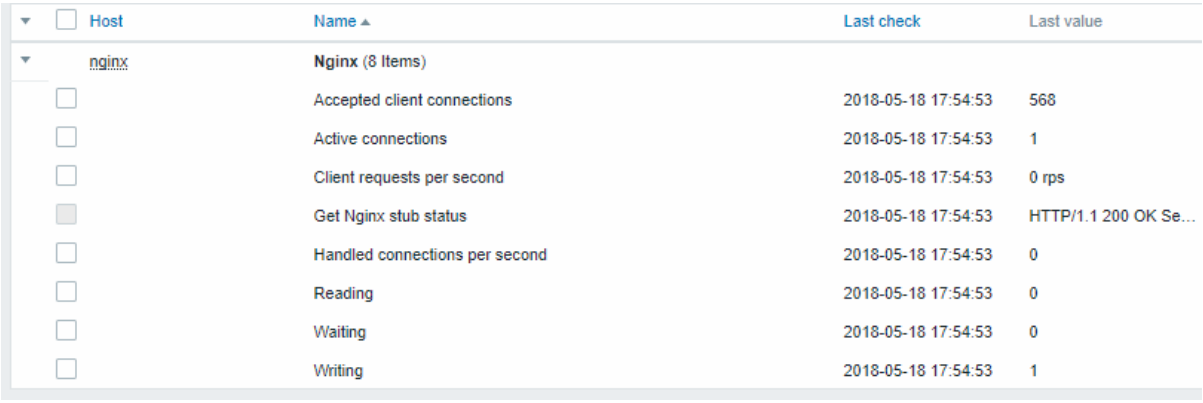
- Configure a sample dependent item for requests per second:



- Sample dependent item value preprocessing with regular expression:



- Check the complete result from stub module in *Latest data*:



18 Prometheus 数据处理

概述

Zabbix 可以查询 Prometheus 行格式标准的度量数据。

收集 Prometheus 数据需要如下两步:

- 一个主 HTTP 监控项数据端点, 例如：`https://<prometheus host>/metrics`
- 一个预处理配置，预定步骤为 Prometheus 选项的依赖监控项，用于从主 HTTP 监控项收集的度量中查询所需数据。

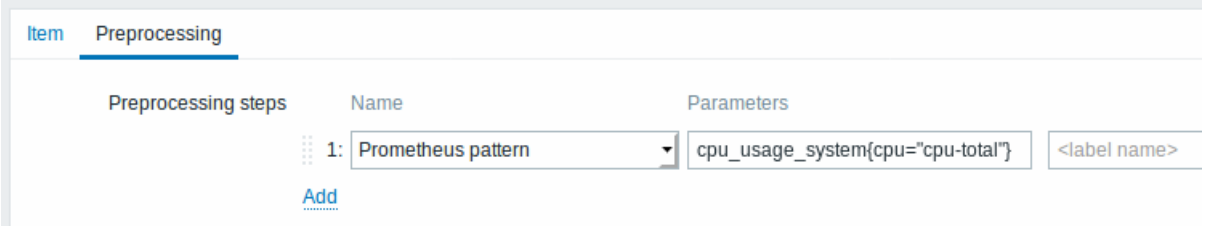
有两个 Prometheus 预处理的预定步骤配置选项:

- *Prometheus* 正则 - 用于普通监控项中查询 Prometheus 数据
- *Prometheus* 转 *JSON* - 用于普通监控项及低级发现中. 在这种情况下，将返回 JSON 格式的 Prometheus 数据.

配置

如果您配置了 HTTP 主监控项, 你需要创建一个使用 Prometheus 预定步骤配置的依赖监控项，配置步骤如下:

- 点击监控项配置表单的类型（Type），选择相关项目（Dependent item）并配置主监控项
- 切换到预处理（Preprocessing）选项页
- 选择一个 *Prometheus* 预定步骤（Preprocessing steps），选择 *Prometheus* 正则或 *Prometheus* 转 *JSON*



参数描	样例
Pattern	要定义所需的数据的正则规则，可以使用类似于 Prometheus 查询语言的语法规则 (参见对照表), 例如.: *wmi_os_physical_memory_free_bytes* - 被选定的 *metric* 名称 cpu {__name__=”<metric name>”} - 被选定的 metric 名称 *cpu {__name__=~”<regex>”}* - 匹配被选定 *metric* 名称的正则表达式 cpu_usage {<label name>=”<label value>”,...} - 被选定的 label 名称 *wmi {<label name>=~”<regex>”,...}* - 匹配被选定 *label* 名称的正则表达式 wmi_os_ti{__name__=~”.*”}==<value> - 被选定的 metric 值 或以上两者的结合: <metric name>{<label1 name>=”<label1 value>”,<label2 name>=~”<regex>”,...}==<value> Label 值可以是 UTF-8 字符的任意序列，但反斜杠、双引号和换行符必须转义为 \\, \" 和 \n；其他字符不必转义。
Output	定义 Label 名称（可选）。在这种情况下，将返回与 Label 名称对应的值。 此字段仅适用于 *Prometheus* 正则选项。

Prometheus 转 JSON

可以使用低级（low-level）发现处理 Prometheus 数据，此时需要配置预处理步骤选项为 *Prometheus* 转 *JSON* 将 Prometheus 数据转换为 JSON 格式数据。

详细参见使用 Prometheus 数据发现.

查询语法对照表

下表列出了 PromQL 和 Zabbix Prometheus 预处理查询语法之间的差异。

	PromQL 语法 Z	bbix Prometheus 预处理语法
差异	etheus 服务器 Promet	eus 格式的纯文本
查询目标		
Pro		
返回	向量 (Instant vector) Metri	或 label 值 (Prometheus 正则)
瞬		每条度量数据的 JSON 列表 (Prometheus 转 JSON)

PromQL 语法 Z	bbix Prometheus 预处理语法
Label匹配运算符	!=, =~, !~ **=* , =~
**=*	PCRE
Label或Metric名称匹配的Regular表达式	RE2
比较运算符参见[相似按metric名称等于字符串进行选定<metric	表](https://prometheus.io/docs/prometheus/latest/querying/operators/#comparison-binary-operators) 只有 **==* (等于) 支持值过滤
ame> or {__name__=”<metric name>”} <metric	ame> or {__name__=”<metric name>”}

PromQL 语法 Z	bbix Prometheus 预处理语法
按 ~”<regex>”} {__name__ met- ric 名 称 匹 配 正 则 表 达 式 进 行 选 定 {__name__	~”<regex>”}
按 ame>=”<label value>”,...} {<label <la- bel name> 的 值 等 于 字 符 串 进 行 选 定 {<la- bel	ame>=”<label value>”,...}
按 e>=~”<regex>”,...} {<label na <la- bel name> 的 值 匹 配 正 则 表 达 式 进 行 选 定 {<la- bel na	e>=~”<regex>”,...}

	PromQL 语法 Z	bbix Prometheus 预处理语法
按值等于字符串进行选定{__name	_=~".*"} == <value> {__name	_=~".*"} == <value>

4 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。由于触发器触发基于历史数据，因此不会有触发器的功能。:::

<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。

随着数据的流入，Zabbix 服务器会在运行时在趋势缓存中累积趋势数据。在以下情况下，服务器会将趋势刷新到数据库中（前端可以在其中找到它们）：

- 一个新的小时开始或者服务端收到该监控项的新值
- 一个新的小时将在不到 5 分钟内结束（没有新值）
- 服务器停止

要在图表上查看趋势，您需要至少等到下一小时的开始（如果监控项经常更新），最多等到下一小时的结束（如果监控项很少更新），最多需要 2 个小时。
当服务器刷新趋势缓存并且该小时数据库中已经有趋势数据时（例如，服务器已在半小时中重启），服务器需要使用更新语句而不是简单的插入。因此，在更大的初始化安装上，如果需要重新启动，最好在一小时结束时停止服务器，在下一小时开始时启动，以避免趋势数据重叠。
历史表不以任何方式参与趋势生成。

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。

5 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。

6 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 */
```

```c
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。

7 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。

8 批量更新

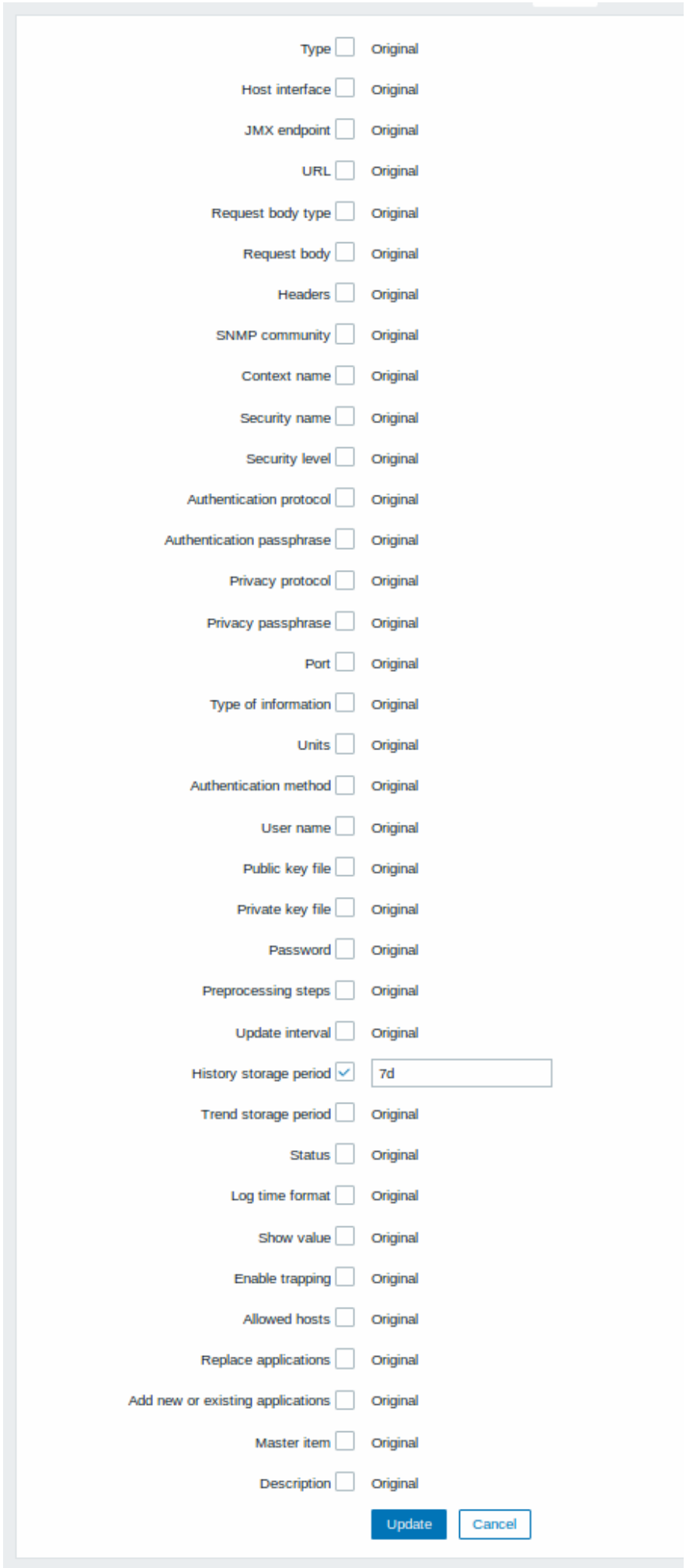
概览

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


Type Original
Host interface Original
JMX endpoint Original
URL Original
Request body type Original
Request body Original
Headers Original
SNMP community Original
Context name Original
Security name Original
Security level Original
Authentication protocol Original
Authentication passphrase Original
Privacy protocol Original
Privacy passphrase Original
Port Original
Type of information Original
Units Original
Authentication method Original
User name Original
Public key file Original
Private key file Original
Password Original
Preprocessing steps Original
Update interval Original
History storage period 7d
Trend storage period Original
Status Original
Log time format Original
Show value Original
Enable trapping Original
Allowed hosts Original
Replace applications Original
Add new or existing applications Original
Master item Original
Description Original
Update
Cancel

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。

9 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。

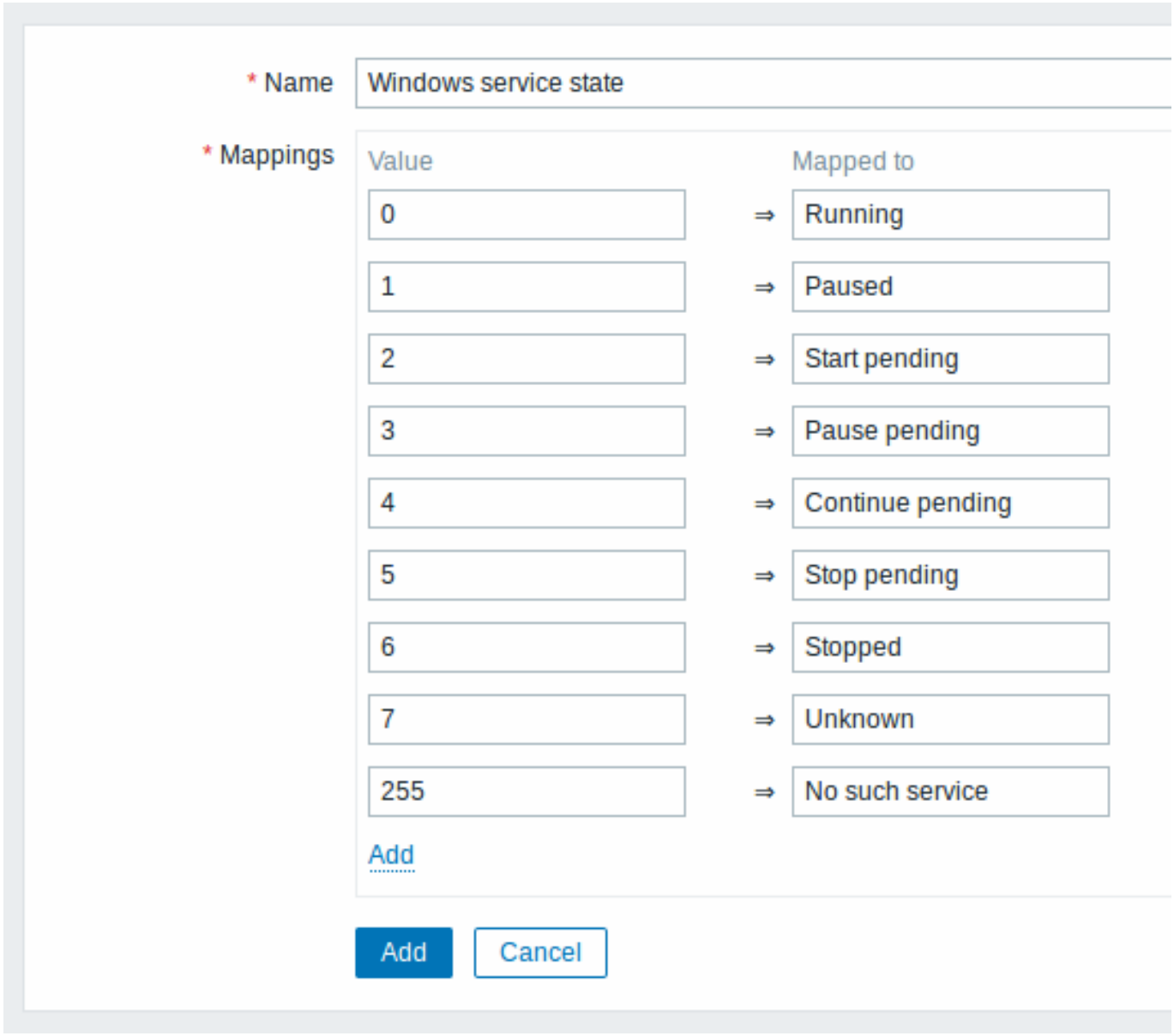
> **Note:**
> 值映射能被用来替换 数字（无符号），数字（浮点）和 字符类型的监控项信息

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：

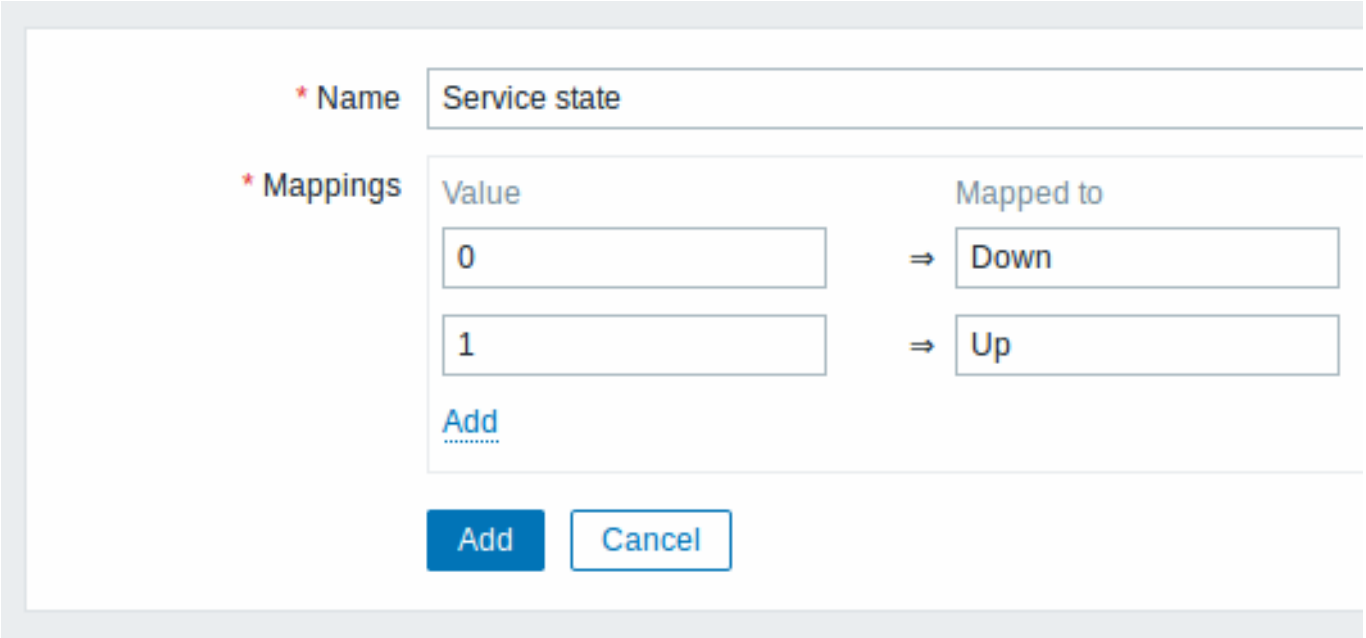
参数	描
名称	值映射的名称，应当时唯一的
映射单	映射 - 一对值与字符串表示.

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。

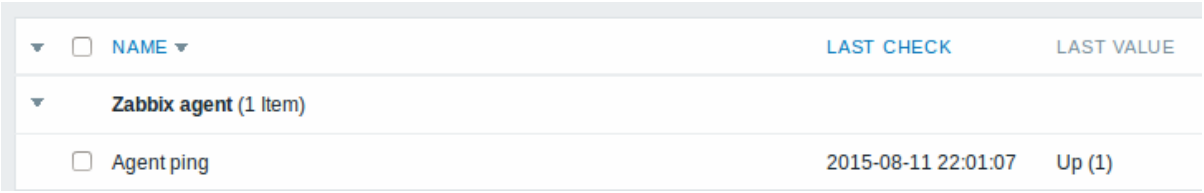
值映射如何工作的

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。

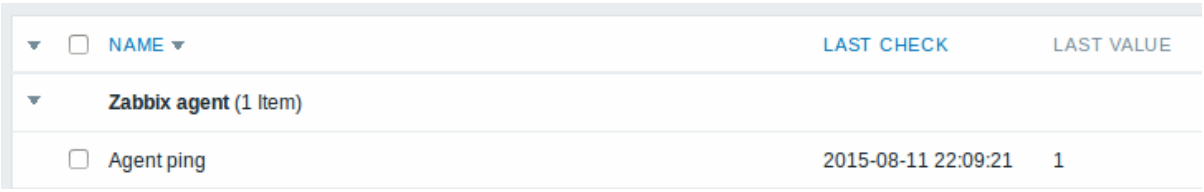
这样配置以后，在监控中 → 最新数据会以映射的值 “Up” 显示（括号中显示的时原始值）。



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。

10 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。

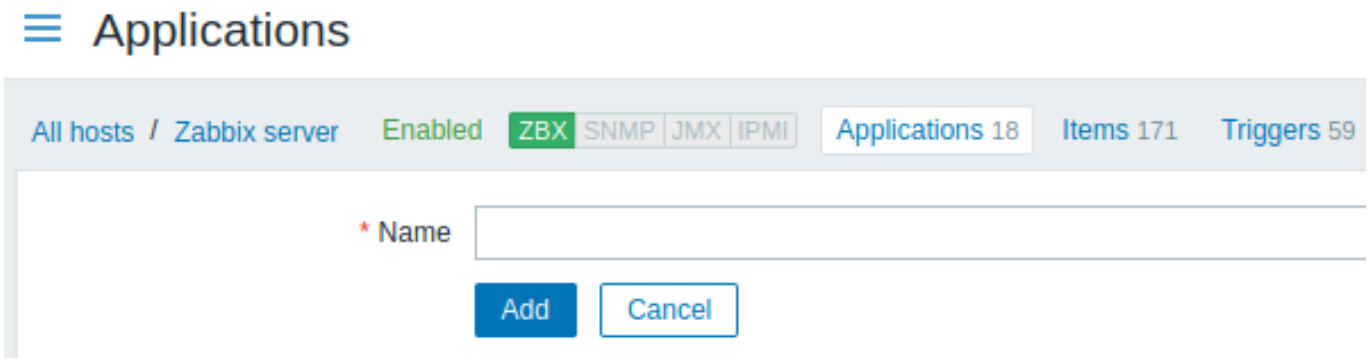
如果你正在使用应用，那么在监控中-最新数据中，你将看到按各应用分组下的监控项和 Web 场景。

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。

11 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：

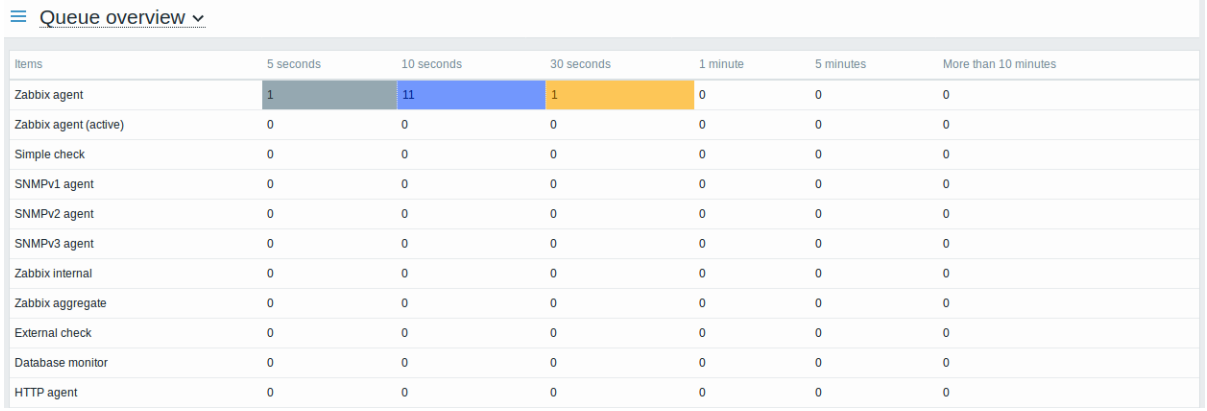
- log, logrt and eventlog 相关的 ZABBIX Agent（主动）监控项
- SNMP trap 类型监控项
- trapper 类型监控项
- web 场景监控的监控项

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。

12 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]。

13 立刻执行

概览

在 Zabbix 中，检查新监控项的值是一个基于配置的更新间隔的循环过程。虽然对于大多数监控项来说更新间隔非常短，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，在实际情况中，可能需要更快的检查新的值。一例如，立刻获取可发现资源的更改。为了满足这种需要，可以重新调整被动检查并立即检索新值。

此功能仅支持被动检查。支持以下监控项的类型：

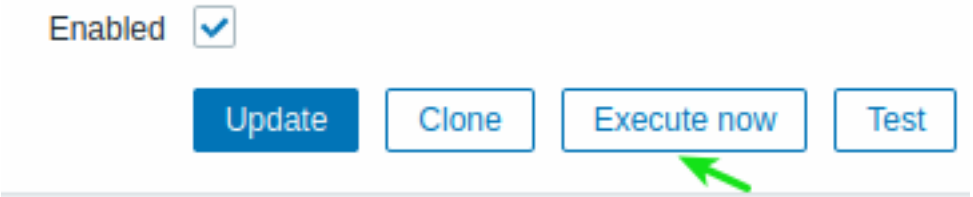
- Zabbix agent (被动模式)
- SNMPv1/v2/v3 agent
- IPMI agent
- Simple check
- Zabbix internal
- Zabbix aggregate
- External check
- Database monitor
- JMX agent
- SSH agent
- Telnet
- Calculated
- HTTP agent

<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；当为配置要检查的监控项时，请使用 Test 选项。:::

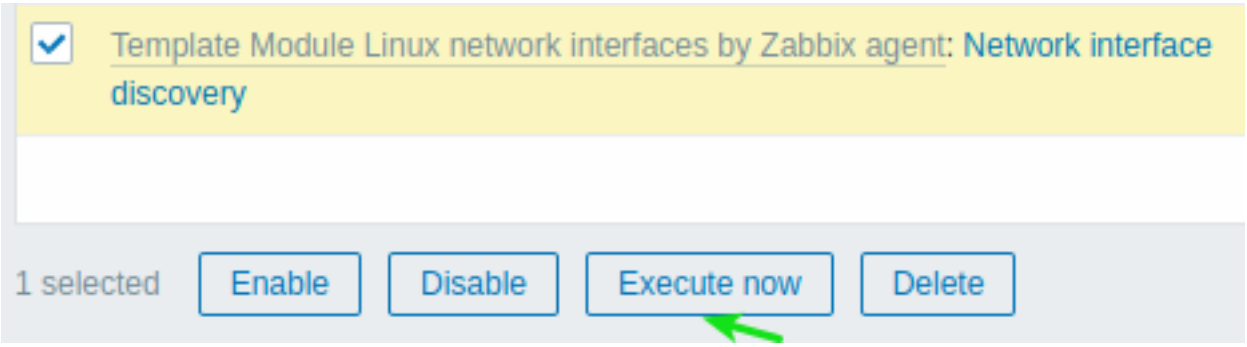
配置

要立刻执行被动检查：

- 在已存在的监控项（或自动发现规则）配置表单中点击立即执行:



- 在监控项（或发现规则列表）中，选定监控项/规则后，单击立即执行:



在后一种情况下，可以选择多个项目/规则，一次性对它们进行 “立即执行”。

14 插件

插件扩展了 Zabbix 监控功能。这些插件是用 Go 编程语言编写的，只支持 Zabbix agent2。它们提供了可加载模块（用 C 编写）和其它扩展 Zabbix 功能的方法，例如用户参数（agent 指标），外部检查（无 agent 监控）和 `system.run[]` Zabbix agent 监控项 。

以下是 agent2 及其插件功能：

*单个配置文件（所有插件配置参数与agent本身的参数位于同一个文件中）；
*基于调度和任务并发的任务队列管理；
*插件级超时。

开箱即用的插件

所有支持 Zabbix agent2 的指标都由插件收集。以下是 Zabbix agent2 可直接使用插件：

插件名称描述	支持监控项	备注
Agent	Zabbix agent 使用的监控项。agent.	ostname, agent.ping, agent.version 与 Zabbi agent 支持的参数keys相同。
Ceph	监控 Ceph。ce	h.df.details, ceph.osd.stats, ceph.osd.discovery, ceph.osd.dump, 支持 ceph.ping, ceph.pool.discovery, ceph.status keys 仅能用于 Zab-bix agent2。
CPU	监控系统 CPU (number of CPUs/CPU cores, discovered CPUs, utilization percentage)。syst	m.cpu.discovery, sys-tem.cpu.num, system.cpu.util 与 Zab ix agent 支持的参数keys相同。
Docker	监控 Docker 容器。dock	r.container_info, docker.container_stats, docker.containers, docker.containers.discovery, 可使用 *docker.data_usage* 监控模板。 docker.images, docker.images.discovery, docker.info, docker.ping pp Docker* 支持的keys 仅能用于 Zab-bix agent2。
File	文件属性信息搜集。vfs.file	cksum, vfs.file.contents, vfs.file.exists, vfs.file.md5sum, 与 Zab-bixvfs.file.regexp, vfs.file.regmatch, vfs.file.size, vfs.file.time gent 支持的参数keys相同。
Kernel	监控内核。kern	l.maxfiles, kernel.maxproc 与 Zab ix agent 支持的参数keys相同。

插件名称描述	支持监控项	备注	
Log	监控日志文件。log, l	g.count, logrt, logrt.count 与 Zabbi	agent 支持的参数keys相同。
Memcached	监控缓存服务器。memcach	d.ping, mem-chached.stats 可使用 *App	Memchached* 监控模版。 支持的keys仅能用于 Zab-bix agent2。
MySQL	监控 MySQL 数据库及其相关内容。mysql.db.di	covery, mysql.db.size, mysql.get_status_va 以使用 *DB MyS- mysql.ping, mysql.replication.di mysql.replication.ge mysql.version	L by Zab-bix agent 2* 监 covery, t_slave_status 控模版。 支持的keys仅能用于 Zab-bix agent2。 默认配置: URI=tcp://lo username=roo ”pass-word= ”.

插件名称描述	支持监控项	备注
NetIf	监控网络接口。net.if	collisions, net.if.discovery, net.if.in, net.if.out, net.if.total 与 Zabbi agent 支持的参数keys相同。

插件名称描述	支持监控项	备注
Oracle	监控 Oracle 数据库。oracl	.diskgroups.stats, ora-cle.diskgroups.discovery, ora-cle.archive.info, ora-cle.archive.discovery, 仅适用于 oracle.cdb.info, ora-cle.custom.query, ora-cle.datafiles.stats, ora-cle.db.discovery, oracle.fra.stats, ora-cle.instance.info, ora-cle.pdb.info, ora-cle.pdb.discovery, oracle.pga.stats, oracle.ping, or-acle.proc.stats, ora-cle.redolog.info, oracle.sga.stats, ora-cle.sessions.stats, ora-cle.sys.metrics, ora-cle.sys.params, oracle.ts.stats, ora-cle.ts.discovery, oracle.user.info 默认配置：URI=tcp://lo service=XE (ser-vice name)。 插件的功能可以通过自定义用户定义

abbix Agent2* 可以使用监, * 控模版。

在使用插件之前安装 Or-a-cle In-stant Client 。

支持的keys仅能用于 Zab-bix agent2 。

插件名称描述	支持监控项	备注
PostgreSQL	监控 PostgreSQL 及其相关内容。pgsql.pi	g, pgsql.db.discovery, pgsql.db.size, pgsql.db.age, pgsql.database.bloating_tables, * 仅能用于 Zabpgsql.replication, pgsql.replication_lag.sec, pgsql.replication_lag.b, pgsql.replication.count, pgsql.replication.status, pgsql.replication.recovery_role, pgsql.cache.hit, pgsql.connections, pgsql.archive, pgsql.bgwriter, pgsql.dbstat.sum, pgsql.dbstat, pgsql.wal.stat, pgsql.locks, pgsql.pgsql.oldest.xid, pgsql.uptime	ix agent2* 可以使用监控模版。 支持的keys 仅能用于 Zab-bix agent2。
Proc	监控 CPU 使用率。proc.	pu.util 与 Zabb	x agent 支持的参数key相同。
Redis	监控 Redis 服务器。redis	config, redis.info, redis.ping, re-dis.slowlog.count 可以使用 *	B Re-dis* 监控模版. 支持的keys 仅能用于 Zab-bix agent2。

插件名称描述	支持监控项	备注	
Swap	交换空间大小/比率。system.s	ap.size 支持 Zabbix	5.0.5 及其以上版本.与 Zab-bix agent 支持的参数key相同。
SystemRun	执行特定的命令。system.	un 与 Zabbix	agent 支持的参数key相同。
Systemd	监控系统服务。system	.unit.discovery, sys-temd.unit.get, sys-temd.unit.info 支持的 [k	ys](/manual/con仅能用于 Zab-bix agent2。
TCP	TCP 连通性检测。net.t	p.port 与 Zabb	x agent 支持的参数key相同。
UDP	监控 UDP 服务及其性能。net.udp.	ervice, net.udp.service.per与 Zabbix	gent 支持的参数keys相同。
Uname	查询系统信息。system	hostname, sys-tem.sw.arch, system.uname 与 Zabbi	agent 支持的参数keys相同。

插件名称描述	支持监控项	备注	
Uptime	系统指标信息搜集。system.u	time 与 Zabbix	gent 支持的参数key相同。
VFSDev	虚拟文件系统设备指标搜集。vfs.dev.disc	very, vfs.dev.read, vfs.dev.write 与 Zabbix agen	支持的参数keys相同。
Web	Web 页面监控。web.	age.get, web.page.perf, web.page.regexp 与 Zab	ix agent 支持的参数keys相同。
ZabbixAsync	异步指标搜集。net.tc	.listen, net.udp.listen, sensor, sys- tem.boottime, sys- tem.cpu.intr, sys- tem.cpu.load, 与 Zabbisys- tem.cpu.switches, system.hw.cpu, sys- tem.hw.macaddr, sys- tem.localtime, system.sw.os, system.swap.in, sys- tem.swap.out, vfs.fs.discovery	agent 支持的参数keys相同。
ZabbixStats	Zabbix server/proxy 内部指标或者队列延迟数量统计。zabbix.stats	与 Zabbix agent 支	的参数keys相同。

插件名称描述	支持监控项	备注
ZabbixSync	同步指标搜集。net.dn	, net.dns.record, net.tcp.service, net.tcp.service.perf, proc.mem, 与 Zabbix agent 支持的参数keys相同。 proc.num, system.hw.chassis, system.hw.devices, system.sw.packages, system.users.num, vfs.dir.count, vfs.dir.size, vfs.fs.get, vfs.fs.inode, vfs.fs.size, vm.memory.size

配置插件

本节介绍了常见的配置原则和最佳实践。

有关插件配置的具体示例，请参见：

- Ceph
- Memcached
- MySQL
- Oracle
- PostgreSQL
- Redis

所有插件都是使用 Zabbix agent2 的 *plugins.** 参数配置文件配置的。与其它 agent 参数不同，它不是参数的键/值类型。它是一个单独的部分，可以描述插件的特定参数。每个参数应具有以下结构：

Plugins.<PluginName>.<Parameter>=<Value>

参数名称应符合以下要求：

- 建议将插件的名称大写；
- 参数应大写；
- 不允许使用特殊字符；
- 嵌套不受最大级别的限制；
- 参数的数量不受限制。

Default values

Since Zabbix 5.0.35, you can set default values for the connection-related parameters (URI, username, password, etc.) in the configuration file in the format:

Plugins.<PluginName>.Default.<Parameter>=<Value>

For example, *Plugins.Mysql.Default.Username=zabbix*, *Plugins.MongoDB.Default.Uri=tcp://127.0.0.1:27017*, etc.

If a value for such parameter is not provided in an item key or in the named session parameters, the plugin will use the default value. If a default parameter is also undefined, hardcoded defaults will be used.

If an item key does not have any parameters, Zabbix agent 2 will attempt to collect the metric using values defined in the default parameters section.

Named sessions

Named sessions represent an additional level of plugin parameters and can be used to specify separate sets of authentication parameters for each of the instances being monitored. Each named session parameter should have the following structure:

Plugins.<PluginName>.Sessions.<SessionName>.<Parameter>=<Value>

A session name can be used as a connString item key parameter instead of specifying a URI, username, and/or password separately.

In item keys, the first parameter can be either a connString or a URI. If the first key parameter doesn't match any session name, it will be treated as a URI. Note that embedding credentials into a URI is not supported, use named session parameters instead.

The list of available named session parameters depends on the plugin, see Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows) for details.

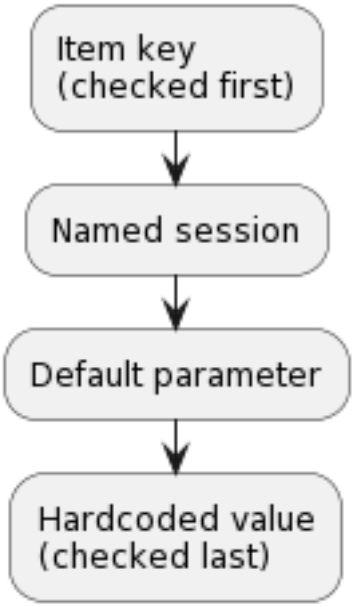
In Zabbix versions before 5.0.34, when providing a connString (session name) in key parameters, item key parameters for username and password must be empty. The values will be taken from the session parameters. If an authentication parameter is not specified for the named session, a hardcoded default value will be used.

Since Zabbix 5.0.34, it is possible to override session parameters by specifying new values in the item key parameters (see example).

Since Zabbix 5.0.35, if a parameter is not defined for the named session, Zabbix agent 2 will use the value defined in the default plugin parameter.

Parameter priority

Since version 5.0.34, Zabbix agent 2 plugins search for connection-related parameter values in the following order:



1. The first item key parameter is compared to session names. If no match is found it is treated as an actual value; in this case, step 3 will be skipped. If a match is found, the parameter value (usually, a URI) must be defined in the named session.
2. Other parameters will be taken from the item key if defined.
3. If an item key parameter (for example, password) is empty, plugin will look for the corresponding named session parameter.
4. If the session parameter is also not specified, the value defined in the corresponding default parameter will be used.
5. If all else fails, the plugin will use the hardcoded default value.

Example 1

Monitoring of two instances "MySQL1" and "MySQL2".

Configuration parameters:

```
Plugins.Mysql.Sessions.MySQL1.Uri=tcp://127.0.0.1:3306
Plugins.Mysql.Sessions.MySQL1.User=mysql1_user
Plugins.Mysql.Sessions.MySQL1.Password=unique_password
Plugins.Mysql.Sessions.MySQL2.Uri=tcp://192.0.2.0:3306
Plugins.Mysql.Sessions.MySQL2.User=mysql2_user
Plugins.Mysql.Sessions.MySQL2.Password=different_password
```

Item keys: `mysql.ping[MySQL1]`, `mysql.ping[MySQL2]`

Example 2

Providing some of the parameters in the item key (supported since Zabbix 5.0.34).

Configuration parameters:

```
Plugins.Postgres.Sessions.Session1.Uri=tcp://192.0.2.234:5432
Plugins.Postgres.Sessions.Session1.User=old_username
Plugins.Postgres.Sessions.Session1.Password=session_password
```

Item key: `pgsql.ping[session1,new_username,,postgres]`

As a result of this configuration, the agent will connect to PostgreSQL using the following parameters:

- URI from session parameter: *192.0.2.234:5432*
- Username from the item key: *new_username*
- Password from session parameter (since it is omitted in the item key): *session_password*
- Database name from the item key: *postgres*

Example 3

Collecting a metric using default configuration parameters.

Configuration parameters:

```
Plugins.Postgres.Default.Uri=tcp://192.0.2.234:5432
Plugins.Postgres.Default.User=zabbix
Plugins.Postgres.Default.Password=password
```

Item key: `pgsql.ping[,,,postgres]`

As a result of this configuration, the agent will connect to PostgreSQL using the parameters:

- Default URI: *192.0.2.234:5432*
- Default username: *zabbix*
- Default password: *password*
- Database name from the item key: *postgres*

连接

一些插件支持同时从多个实例收集度量。可以监视本地和远程实例。支持 TCP 和 Unix 套接字连接。

建议配置插件，使实例的连接保持在打开状态。这样做的好处是减少了网络拥塞、延迟以及由于连接数较少而导致的 CPU 和内存使用。客户端库负责这个。

> **Note:**
> 未使用的连接保持打开的时间段可以由 *Plugins.<PluginName>.KeepAlive* 参数确定。
> 示例：*Plugins.Memcached.KeepAlive*

编写插件

插件是定义结构并实现一个或多个插件接口（Exporter、Collector、Runner、Watcher）的 Go 包：

- 插件导出器 **plugin.Exporter**

Exporter 是执行轮询并返回值（values）、nothing 和 error 的最简单接口。它接受预先准备好的项键、参数和上下文。Exporter 接口是唯一可以同时访问的接口。所有其它插件接口访问都是独占的，当插件已经在执行某些任务时，不能调用任何方法。此外，每个插件最多只能调用 100 个 concurrent Export ()，可以根据需要减少每个插件的调用次数。

- 插件收集器 **plugin.Collector**

收集器用于插件需要定期收集数据时。此接口通常与导出器接口一起用于导出收集的数据。

- 插件配置器 **plugin.Configurator**

Configurator 用于从 agent2配置文件向插件提供配置参数。

- 插件运行器 **plugin.Runner**

Runner 接口提供了在插件启动（激活）时执行一些初始化，在插件停止（停用）时执行取消初始化的方法。例如，插件可以通过实现 Runner 接口来启动/停止一些后台 goroutine。

- 插件. 观察者 **plugin.Watcher**

Watcher 允许插件实现自己的度量轮询，而不使用 agent 的内部调度程序，例如在基于陷阱的插件中。

默认情况下，插件处于非活动状态，只有在监视插件提供的度量时才会激活。

插件位于插件目录树中，按含义分组，例如 `plugins/system/uptime/uptime.go`.

实施步骤

插件必须导入 zabbix.com/pkg/plugin 包。

```
import "zabbix.com/pkg/plugin"
```

插件必须定义结构并嵌入 plugin.Base 结构。

```
type Plugin struct {
    plugin.Base
}
var impl Plugin
```

一个插件必须实现一个或多个插件接口。

```
func (p *Plugin) Export(key string, params []string, ctx plugin.ContextProvider) (result interface{}, err
    if len(params) > 0 {
        p.Debugf("received %d parameters while expected none", len(params))
        return nil, errors.New("Too many parameters")
    }
    return time.Now().Format(time.RFC3339)
}
```

插件必须在初始化期间注册自身。

```
func init() {
    plugin.RegisterMetrics(&impl, "Time", "system.time", "Returns time string in RFC 3999 format.")
}
```

其中 RegisterMetrics 参数为：

- 指向插件实现的指针
- 插件名称（大写）
- 指标 #1 名称（项目键）
- 指标 #1 描述（以大写字符开始，以点结束）
- 指标 #2 名称（项目键）（可选）
- 指标 #2 描述（以大写字符开始，以点结束）（可选）
- ...

如果需要日志记录，插件必须使用 plugin.Base 提供的日志记录功能（见上面的例子）。它基本上是一个标准日志的包装器，但是它会在日志信息前面加上 [<plugin name>]。

15 限制 **agent** 检查

概述

可以通过创建项目黑名单、白名单或白名单/黑名单的组合来限制 agent 的检查。

为此，请结合使用两个 agent配置参数：

''AllowKey=<pattern>''-允许哪些检查；<pattern>使用通配符（）表达式指定
''DenyKey=<pattern>''-拒绝哪些检查；<pattern>使用通配符（）表达式指定

请注意：

- 在默认情况下，全部 system.run[*] 项（远程命令、脚本）是禁用的，即使没有指定拒绝键；
- 从 Zabbix 5.0.2 agent 参数 EnableRemoteCommands：

  * deprecated Zabbix agent已经弃用
  * unsupported Zabbix agent2不支持

因此，要允许所有远程命令，请指定 AllowKey=system.run[*] 参数。(在 Zabbix 5.0.2 版本前，agent 配置中还需要 EnableRemoteCommands=1。)

要仅允许某些远程命令，请创建更具体的 AllowKey[] 参数的白名单。要禁止特定的远程命令，请在 AllowKey=system.run[*] 之前添加 DenyKey 参数。

重要规则

- 没有拒绝规则的白名单只允许 system.run[*] 项。对于所有其它项，如果没有 DenyKey 参数，则不允许使用 AllowKey 参数；在这种情况下，Zabbix agent 将不会仅使用 **AllowKey** 参数启动。
- 顺序很重要。指定参数在配置文件中按其出现顺序逐一检查：
  - 一旦项键与允许/拒绝规则匹配，该项即被允许或拒绝；并且规则检查停止。因此，如果一个项同时匹配允许规则和拒绝规则，那么结果将取决于哪个规则排在第一位。

    - 顺序还影响 EnableRemoteCommands 参数（如果使用）。
- 支持无限数量的 AllowKey/DenyKey 参数。
- AllowKey、DenyKey 规则不影响 HostnameItem、HostMetadataItem、HostInterfaceItem 配置参数。
- 键模式是一个通配符表达式，其中通配符（*）与特定位置的任意数量的任意字符匹配。它可以用于关键字名称和参数。
- 如果在 agent 配置中不允许某个特定的项密钥，则该项将被报告为不受支持（没有给出有关原因的提示）；
- 带有--print（-p）命令行选项的 Zabbix agent 将不显示配置不允许的密钥；
- 带有--test（-t）命令行选项的 Zabbix agent 将返回”Unsupported item key.” 配置不允许的键的状态；
- 拒绝的远程命令不会记录在 agent 日志中（如果 LogRemoteCommands=1）。

用例

拒绝特定检查

* 黑名单一个特定的检查与 DenyKey 参数。不允许匹配密钥。除 system.run[] 项外，允许使用所有不匹配的密钥。

例如：

```
# 拒绝安全数据访问
DenyKey=vfs.file.contents[/etc/passwd,*]
```

<note important> 黑名单可能不是一个好的选择，因为新的 Zabbix 版本可能具有不受现有配置明确限制的新密钥。这可能会导致安全漏洞。:::

拒绝特定命令，允许其它命令

- 使用 DenyKey 参数将特定命令列入黑名单。使用 AllowKey 参数白名单所有其它命令。

```
# 不允许特定命令
DenyKey=system.run[ls -l /]

# 允许其它脚本
AllowKey=system.run[*]
```

允许特定检查，拒绝其它检查

- 使用 AllowKey 参数的白名单特定检查，使用 `DenyKey=*` 拒绝其它检查

例如：

```
# 允许读取日志:
AllowKey=vfs.file.*[/var/log/*]

# 允许本地时间检查
AllowKey=system.localtime[*]

# 拒绝所有其它密钥
DenyKey=*
```

模式示例

模式描	匹配	不匹配	
*	匹配所有可能的带参数或不带参数的键。Any	None	
vfs.file.contents	匹配 vfs.file.contents 没有参数。vfs.fi	e.contents	e.contents[/etc
vfs.file.contents[]	匹配 vfs.file.contents 具有空参数。vfs.fil	.contents[]	.contents

模式描	匹配	不匹配
vfs.file.contents[*]	匹配 `vfs.file.contents` 具有任何参数；将不匹配 `vfs.file.contents` 没有方括号。 vfs.file.contents	] vfs.file.contents	vfs.file.conte
vfs.file.contents[/etc/passwd,]*	匹配 `vfs.file.contents` 的第一个参数与/etc/passwd 匹配，所有其它参数都有任何值（也为空）。 vfs.file.contents[/etc/pass	d,] vfs.file.content	d] ts[/etc/passv vfs.file.content vfs.file.content
*vfs.file.contents[*passwd*]*	匹配 `vfs.file.contents` 的第一个参数与 *passwd* 匹配，没有其它参数。 vfs.file.contents	/etc/passwd] vfs.file.contents	/etc/passwd,] vfs.file.content utf8]
*vfs.file.contents[*passwd*,*]*	匹配 `vfs.file.contents` 中只有第一个参数匹配 *passwd*，并且后面的所有参数都有任何值（也为空）。 vfs.file.contents[/etc/passwd,	vfs.file.contents[/etc/passwd,utf8]	[/etc/passwd,r
vfs.file.contents[/var/log/zabbix_server.log,,abc]*	匹配 `vfs.file.contents` 的第一个参数匹配/var/log/zabbix_server.log，匹配 “abc” 的第三个参数和任何(也是空的）第二个参数。 vfs.file.contents[/var/log/zabbi	_server.log„abc]	vfs.file.log,abc

模式描	匹配	不匹配	
vfs.file.contents[/etc/passwd,utf8]	匹配 `vfs.file.contents` 的第一个参数匹配/etc/passwd，第二个参数匹配“utf8”，没有其它参数。	sswd,utf8] vfs.file.contents[/etc/p	sswd,] vfs.file.content
*vfs.file.**	匹配以 `vfs.file.` 开头没有任何参数。	tents vfs.file.contents[	vfs.file.size vfs.file.size[/va
vfs.file.[*]*	匹配以 `vfs.file.` 开头的任何键 vfs.file 文件任何参数。	tes[] vfs.file.sizes[	vfs.file.size[/
vfs..contents*	匹配以 `vfs.` 开始并以`.contents` 结束不带任何参数的任何键。	.contents vfs.contents	vfs..cont

system.run 和 AllowKey

例如 myscript.sh 的脚本文件可以通过 Zabbix agent 在主机上以几种方式执行：

1. 作为被动或主动检查中的项目键，例如：

- system.run[myscript.sh]
- system.run[myscript.sh,wait]
- system.run[myscript.sh.nowait]

在此，用户可以添加”wait”、”nowait”，或者省略第二个参数，在 system.run[] 中使用其默认值。

2. 作为全局脚本（由用户在前端或 API 中启动）。

用户在 *Administration→Scripts*中配置此脚本，设置 “Execute on:Zabbix agent” 并放置 myscript.sh 文件输入到脚本的 “Commands” 输入字段。从前端或 API 调用时，Zabbix server 将发送到 agent：

- system.run[myscript.sh,wait] - 最高支持到 Zabbix 5.0.4 版本
- system.run[myscript.sh] - 从 5.0.5 版本开始支持

在此，用户不控制”wait”/”nowait” 参数。

3. 作为一个动作的远程命令。Zabbix server 向 agent 发送：

- system.run[myscript.sh,nowait]

在此，用户同样不控制”wait”/”nowait” 参数。

这意味着如果我们把 AllowKey 设置为：

```
AllowKey=system.run[myscript.sh]
```

则

- system.run[myscript.sh]-将被允许
- system.run[myscript.sh,wait], system.run[myscript.sh,nowait] 将不被允许-如果作为操作步骤调用，脚本将不会运行

要允许所有描述的变体，您可以添加：

```
AllowKey=system.run[myscript.sh,*]
DenyKey=system.run[*]
```

到 agent/agent2 参数。

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

Unknown state

It is possible that an unknown operand appears in a trigger expression if:

- an unsupported item is used
- the function evaluation for a supported item results in an error

In this case a trigger generally evaluates to ”unknown” (although there are some exceptions). For more details, see Expressions with unknown operands.

It is possible to get notified on unknown triggers.

1 配置一个触发器

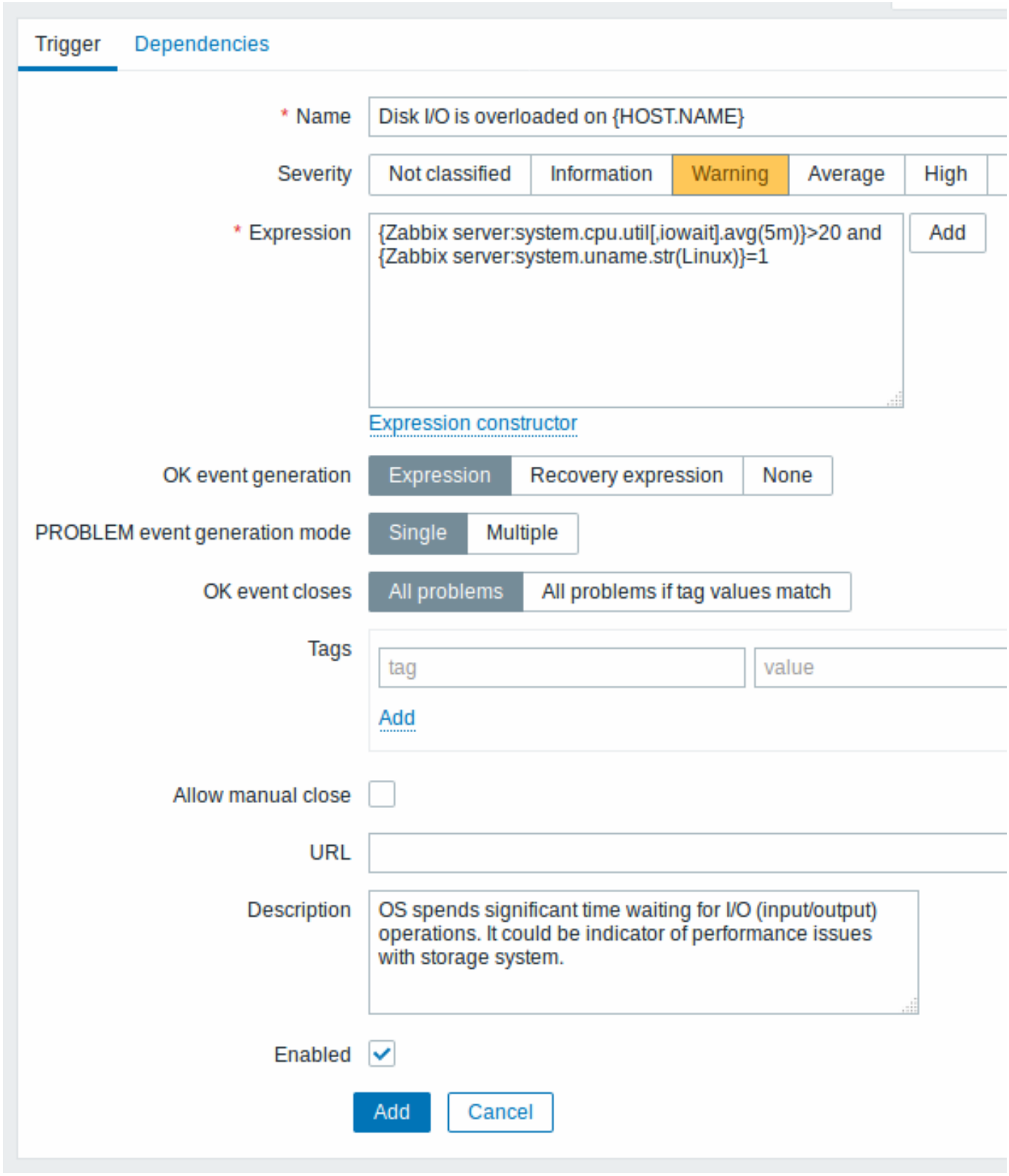
概述

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	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 - {TRIG-GER.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。

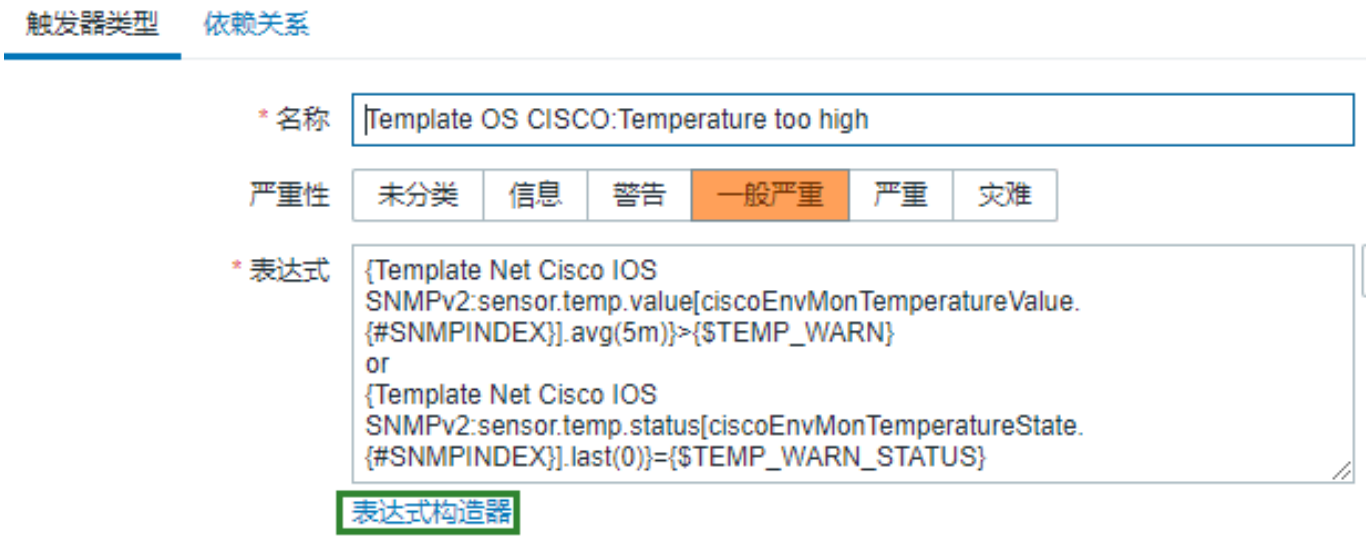
测试表达式

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。

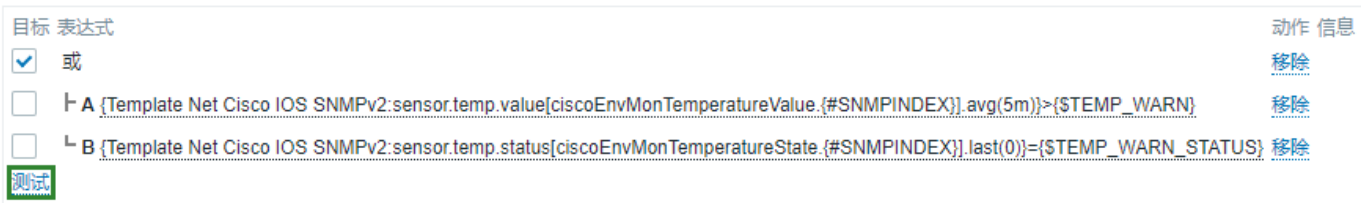
以官方模板的表达为例：

```
{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
```

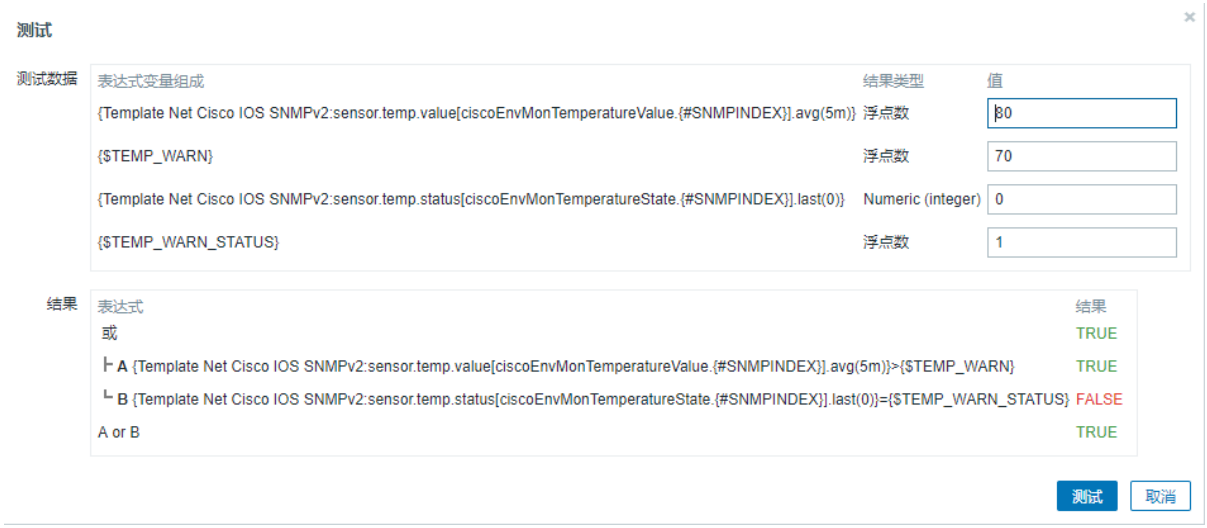
若要测试表达式，请点击表达式字段下的 表达式构造器。



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。

Example 10

CPU activity at any time with exception

Use of function time() and **not** operator:

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2
and not ({zabbix:system.cpu.load[all,avg1].dayofweek(0)}=7 and {zabbix:system.cpu.load[all,avg1].time(0)}>
and not ({zabbix:system.cpu.load[all,avg1].dayofweek(0)}=1 and {zabbix:system.cpu.load[all,avg1].time(0)}<
```

The trigger may change its state to true at any time, except for 2 hours on a week change (Sunday, 23:00 - Monday, 01:00).

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。

Example 1

Temperature in server room is too high.

Problem expression:

```
{server:temp.last()}>20
```

Recovery expression:

```
{server:temp.last()}<=15
```

Example 2

Free disk space is too low.

Problem expression: it is less than 10GB for last 5 minutes

```
{server:vfs.fs.size[/,free].max(5m)}<10G
```

Recovery expression: it is more than 40GB for last 10 minutes

```
{server:vfs.fs.size[/,free].min(10m)}>40G
```

Expressions with unsupported items and unknown values

Versions before Zabbix 3.2 are very strict about unsupported items in a trigger expression. Any unsupported item in the expression immediately renders trigger value to `Unknown`.

Since Zabbix 3.2 there is a more flexible approach to unsupported items by admitting unknown values into expression evaluation:

- For some functions their values are not affected by whether an item is supported or unsupported. Such functions are now evaluated even if they refer to unsupported items. See the list in functions and unsupported items.
- Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
  - "1 `or` Unsuported_item1.some_function() `or` Unsuported_item2.some_function() `or` ..." can be evaluated to '1' (True),
  - "0 `and` Unsuported_item1.some_function() `and` Unsuported_item2.some_function() `and` ..." can be evaluated to '0' (False).
    Zabbix tries to evaluate logical expressions taking unsupported items as `Unknown` values. In the two cases mentioned above a known value will be produced; in other cases trigger value will be `Unknown`.
- If a function evaluation for supported item results in error, the function value is `Unknown` and it takes part in further expression evaluation.

Note that unknown values may "disappear" only in logical expressions as described above. In arithmetic expressions unknown values always lead to result `Unknown` (except division by 0).

If a trigger expression with several unsupported items evaluates to `Unknown` the error message in the frontend refers to the last unsupported item evaluated.

3 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

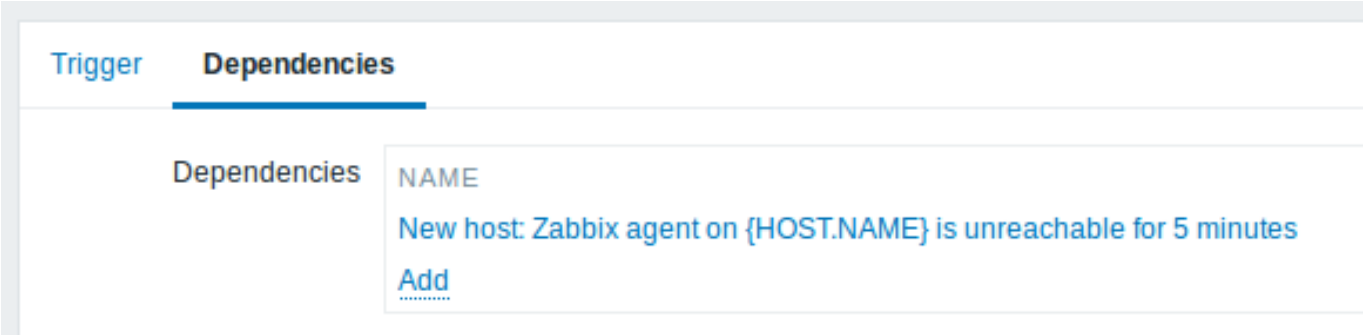
请注意，上述情况下的“次要”（依赖）触发器不会立即更新。

另外：

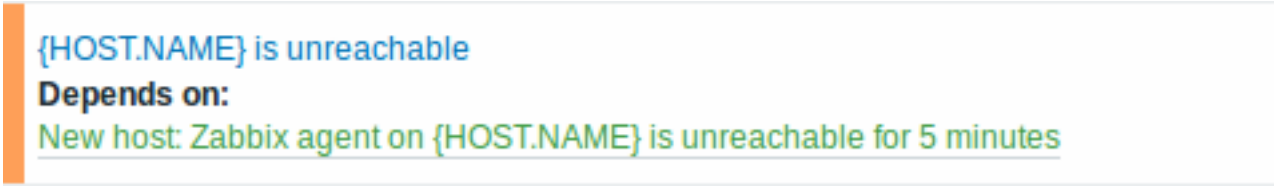
- 触发器依赖可以从任何主机触发器添加到任何其他主机触发器，只要它不会导致循环依赖。
- 触发器依赖可以从一个模板添加到另一个模板，如果模板 A 的触发器依赖于模板 B 的触发器，模板 A 只能与模板 B 一起链接到主机（或其他模板），但是模板 B 可以单独链接到主机（或其他模板）。
- 触发器依赖可以从模板触发器添加到主机触发器。在这种情况下，例如, 有一个触发器依赖于路由器（主机）触发器的模板。链接到这个模板的所有主机都将依赖于特定的路由器。
- 可以不添加从主机触发器到模板触发器的触发器依赖性。
- 触发器依赖可以从一个触发器原型添加到另一个触发器原型 (在同一个 Low-level discovery 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

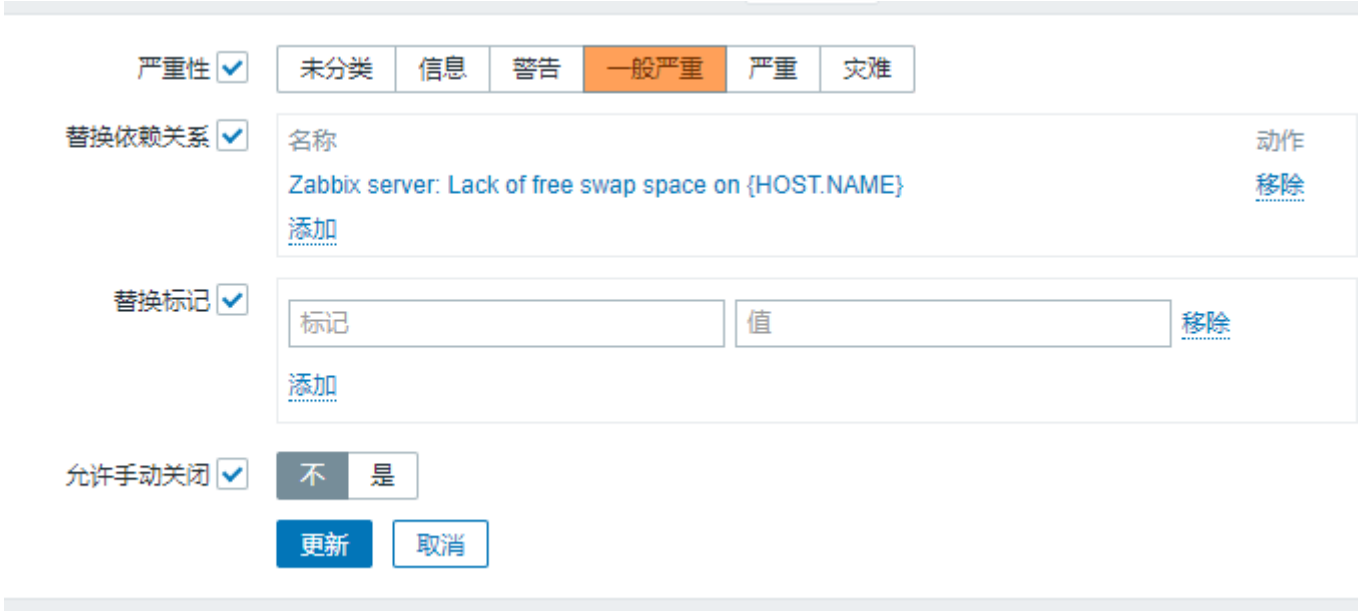
概述

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] 当 HttpOnly 设定为’true’ 时，将只能通过 HTTP 协议访问 cookie。这意味着无法通过脚本语言（例如 JavaScript）访问 Cookie。此设置可以有效地帮助减少通过 XSS 攻击进行的身份盗用（尽管并非所有浏览器都支持此功能）。^ Secure((`Secure` 表示应仅通过从来自客户端的安全的 HTTPS 连接传输 cookie。设置为’true’ 时，仅当存在安全连接时才设置 cookie。

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[2] 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。

[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)。

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[3]；
- 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)[4].

<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[5]：

```
{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

[3] 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。

[4] 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）。

[5] 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。

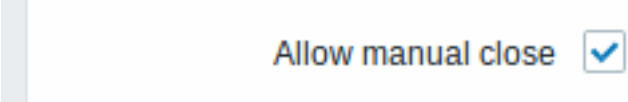
手动关闭问题事件并不意味着底层的触发器将永远不会再次进入“问题”状态。当触发器表达式中包含的任何监控项有新数据达到时，将重新计算整个表达式，并可能会再次生成问题。

配置

需要两步来手动关闭问题事件。

触发器配置

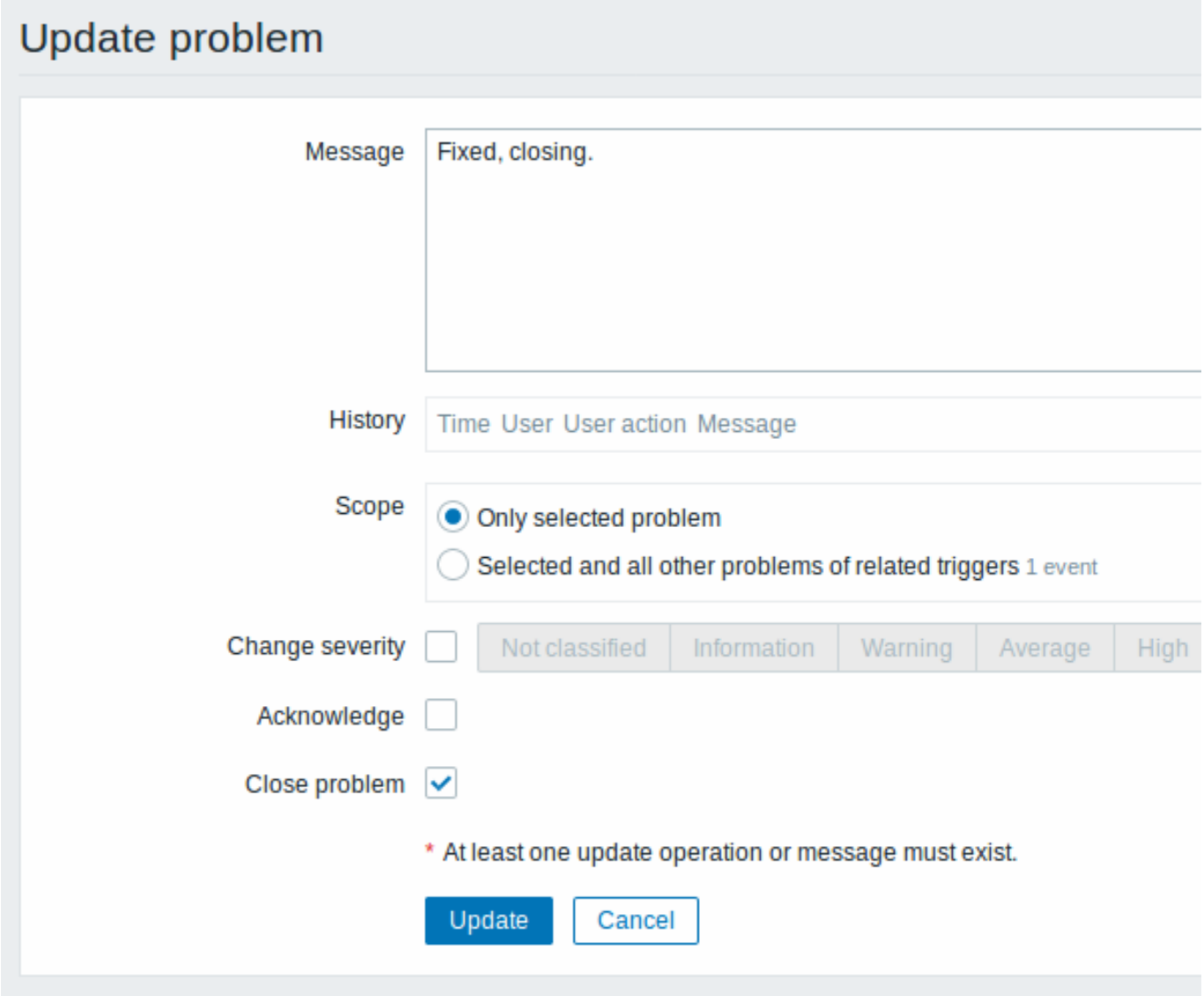
在触发器的配置页面上，启用 允许手动关闭选项。



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：

- 触发器级别的 - 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 基于触发器的事件关联

概述

基于触发器的事件关联，允许关联一个触发器报告的各个不同问题。

在 Zabbix 中，通常一个正常（恢复）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。

这需要在触发器配置页面将 生成多重问题事件选项置为启用。通常适用于日志监控、被动采集（trap）处理等。

如果这么做了，zabbix 可以根据事件标签关联问题事件。事件标签被用于提取值并创建问题事件的标签。利用这一点，还可以根据匹配的标签关闭个别问题。

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
```

事件关联地过程是将 Line1 的问题事件关联到 Line3 的恢复事件，Line2 的问题事件关联到 Line4 的恢复事件。除了完成匹配，还能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
```

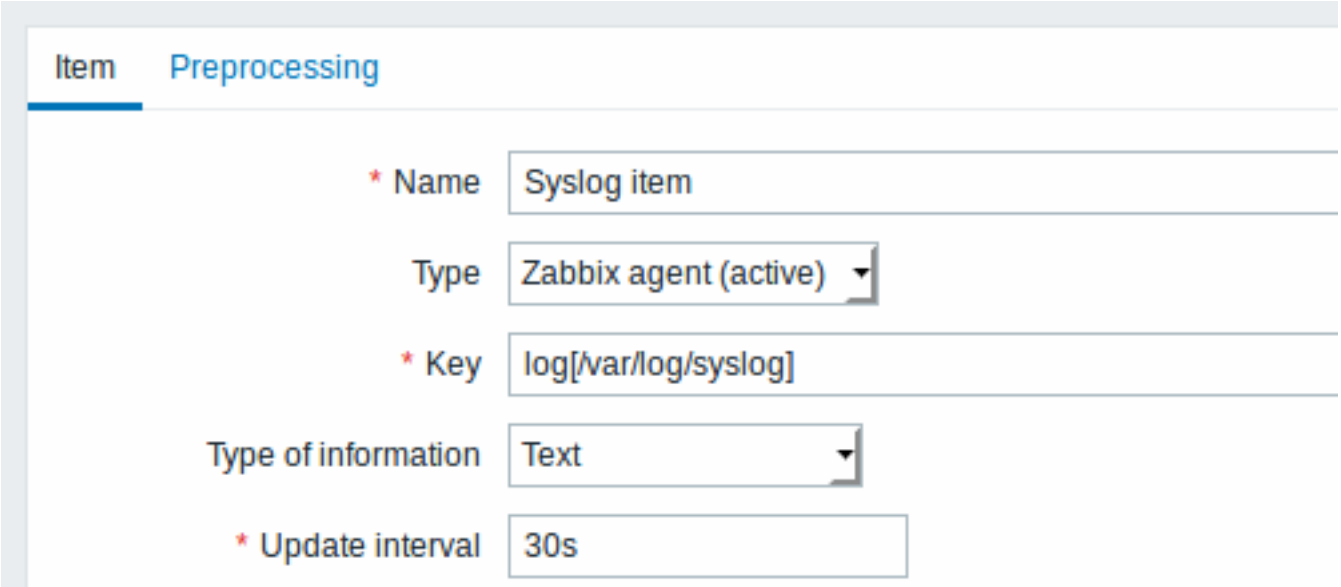
为此，需要通过标签将这些事件相关联，例如，可以标识为”Application 1” 和”Application 2”。这个过程也可以将正则表达式应用于日志中，来提取标签的值。然后，当事件创建时，他们分别给打上为”Application 1” 和”Application 2” 的标签，并且问题可以与解决方法相匹配

配置

监控项

首先，你需要设置一个监控日志文件的监控项，例如：

```
log[/var/log/syslog]
```



当这个监控项设置完毕，等待一分钟让配置变更被 Zabbix Server 读取到，然后去最新数据 页面确认数据有开始采集

触发器

在监控项运行正常后，你需要配置触发器 。

配置前，确认日志文件中哪些条目值得关注非常重要。
举个例子：
以下触发器表达式将搜索 “ Stopping” 之类的字符串以表示潜在问题：

```
{My host:log[/var/log/syslog].regexp("Stopping")}=1
```

Attention:
为了确保包含字符串”Stopping” 的每一行都被视为问题，还需要将触发器中的事件生成模式设置为’ 多重事件’。

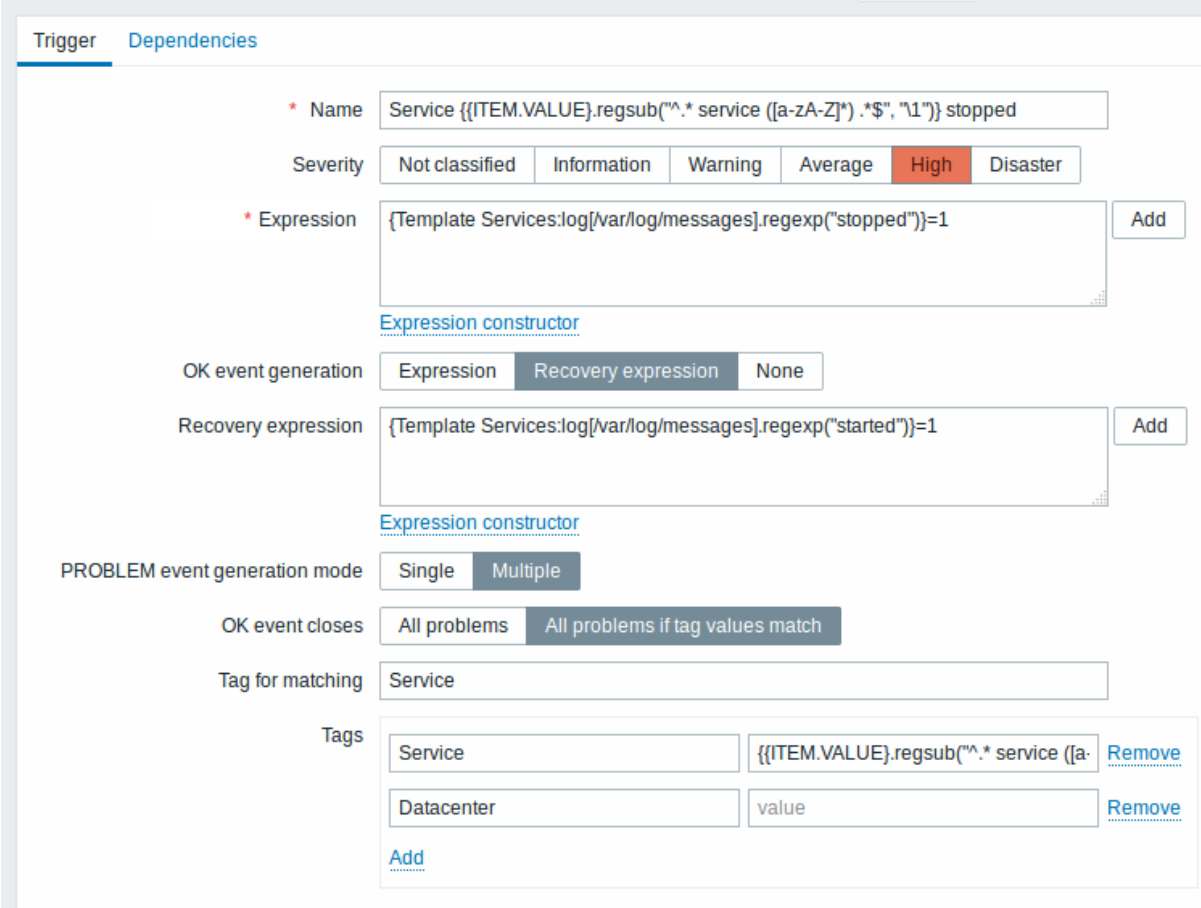
然后顶一个恢复表达式。下面的恢复表达式将会在有一行包含字符串’Starting’ 日志即恢复所有问题事件：

```
{My host:log[/var/log/syslog].regexp("Starting")}=1
```

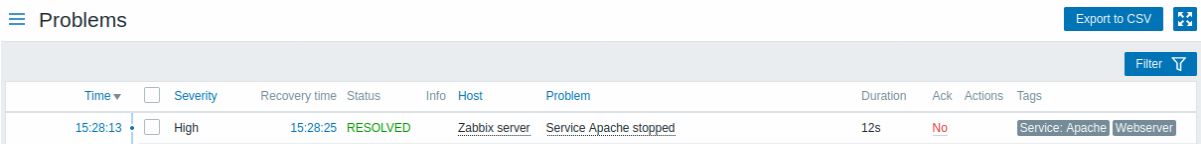
由于不希望因为关闭所有问题，而导致一些关键的故障根源问题也被以某种方式关闭。这时候就需要标签功能登场。

问题事件和恢复事件可以通过触发器配置中指定的标签匹配。以下配置会达成这一目的：

- 设定 // 问题事件生成模式// ：多重
- 设定 // 正常事件关闭 // ：标签匹配的所有问题
- 配置特定 tag 的名称用于事件匹配
- 配置事件标签从日志中提取标签的值



如果配置成功，你能够看到依据应用打上的问题事件标签，并在监测中 → 问题页面看到结果相匹配的问题被解决



<note warning> 因为当为不相关的问题创建相似的事件标签时，有可能出现错误配置，请依据下面方法研究配置！:::

- 在两个应用程序将错误和恢复消息写入同一日志文件的情况下，用户可以在标签配置中，使用 {ITEM.VALUE} 宏，并使用特定的正则表达式过滤宏的值（例如，日志的不同消息格式），以获取应用程序 A 和应用程序 B 的名称。从而在一个触发器中配置匹配具有不同标签的应用程序。但是，如果与正则表达式不匹配，则这可能无法按计划进行。不匹配的正则表达式将在问题和 OK 事件中产生空标记值，并且单个空标记值足以将它们关联起来。因此，来自应用程序 A 的恢复消息可能会意外关闭来自应用程序 B 的错误消息。
- 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。

事件标签

概述

在 Zabbix 中有一个定义自定义事件标签的选项。该标签可以在模板、主机和触发器级别定义。

定义标签后，将使用标签数据标记相应的新事件：

- 模板级别的标签——由该模板的触发器生成的主机问题将被标记
- 主机级别的标签——该主机的所有问题将会被标记
- 触发器级别的标签——该触发器产生的问题将被标记

事件从模板、主机、触发器的整个链路中继承所有标签。在标记事件时，完全相同的 tag:value 组合 (在解析之后) 会合并成一个，而不是产生重复的多个标记。

拥有自定义事件标签可提供更大的灵活性。最重要的是，可以基于事件标签进行事件关联。另外，可以根据事件标签定义动作。

事件标签的展示形式为一对 *tag name*（标签名）和 *value*（值）。可以只使用标签名或将其与一个值配对：

MySQL, Service:MySQL, Services, Services:Customer, Applications, Application:Java, Priority:High

一个（触发器、模板、主机或事件）实体可能有多个名称相同但是值不同的标签——这些标签不会被视为 “重复项”。例如，空值和有值的同名标签可以同时使用。

Note:
主机原型和从原型创建的主机不支持使用标签。

使用示例

该功能的一些使用示例如下：

1. 在前端标记触发器事件

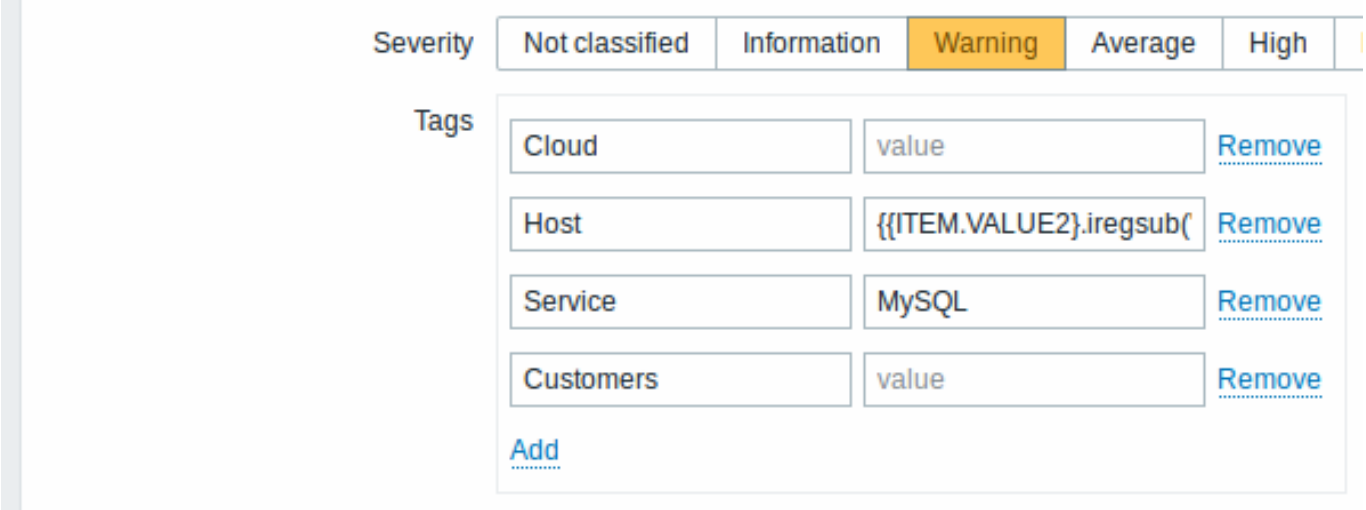
    * 在触发器级别定义标签；
    * 在//Monitoring// ——> //Problems// 页面查看所有被这些标签标记的触发器问题。
- 标记所有继承自模板的问题
    * 在模板级别定义标签，例如 'App=MySQL'；
    * 在//Monitoring// ——> //Problems// 页面查看所有被通过模板触发器创建标签标记的主机问题。
- 标记所有主机问题
    * 在主机级别定义标签，例如 'Service=JIRA'；
    * 在//Monitoring// ——> //Problems// 页面查看被这些标签标记的所有主机触发器问题。
- 识别日志文件中的问题并分别将其关闭
    * 在日志触发器中定义标签，这些标签将使用''%%{{%%ITEM.VALUE<N>}.regsub()}'' 宏解析的值来识别事件；
    * 触发器配置中''问题事件生成模式(PROBLEM event generation mode)''选择''多重(Mulitple)''；
    * 触发器配置中使用 [[zh:manual/config/event_correlation|事件关联]]：在选项''事件成功关闭(OK event close
    * 查看使用标签创建并分别关闭的问题事件。
- 用标签来过滤通知
    * 在触发器级别定义标签，以通过不同的标签标记事件；
    * 在动作条件下使用标签过滤，对仅与标签数据匹配的事件上接收通知。
- 使用从监控项的值中提取的信息作为标签值
    * 在标签值中使用宏 ''%%{{%%ITEM.VALUE<N>}.regsub()}'' ；
    * 在 //Monitoring// → //Problems// 页面查看从监控项值中提取的数据作为标签值的问题。
- 在通知中更好地识别问题
    * 在触发器级别定义标签;
    * 在问题通知中使用宏 {EVENT.TAGS} ；
    * 更容易识别通知所属的应用程序/服务。
- 通过在模板级别使用标签简化配置任务
    * 在模板触发器级别定义标签;
    * 在模板触发器创建的所有触发器上查看这些标签。
- 使用低级发现(LLD)中的标签创建触发器
    * 在触发器原型上定义标签;
    * 在标签名或值中使用LLD宏;
    * 在触发器原型创建的所有触发器上查看这些标签.

配置

事件标签可以在如下配置中进行配置:

- 模板配置 - 链接到主机时影响模板中的所有触发器

- 主机配置 - 影响主机的所有触发器
- 单个触发器配置:



可以为触发器、模板触发器和触发器原型定义事件标签。

支持的宏

以下宏可用于触发器级别标签:

- {ITEM.VALUE}, {ITEM.LASTVALUE}, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} 和 {HOST.ID} 这些宏可被填写到标签名或标签值中。
- {INVENTORY.*} 宏 可在触发器表达式中用于从一个或多个主机中获取主机资产信息 (从 4.0.0 开始支持)。
- 用户宏 标签名/值支持用户宏上下文。用户宏上下文可能包括低级发现宏。
- 低级发现宏可用于触发器原型中的标签名/值。

以下宏可用于基于触发器的通知:

- {EVENT.TAGS} 和 {EVENT.RECOVERY.TAGS} 宏将解析为事件标签或恢复事件标签的逗号分隔列表。
- {EVENT.TAGSJSON} and {EVENT.RECOVERY.TAGSJSON} 宏将解析为包含事件标签对象 或恢复事件标签对象的 JSON 数组。

以下宏可用于模板和主机级标签：

- {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} 和 {HOST.ID} 宏
- {INVENTORY.*} 宏
- 用户宏

触发器标签中使用提取的字符串

支持使用宏中提取的字符串来作为标签名或标签值-将正则表达式应用于宏 {ITEM.VALUE}, {ITEM.LASTVALUE} 或低级发现宏中，例如:

```
{{ITEM.VALUE}.regsub(pattern, output)}
{{ITEM.VALUE}.iregsub(pattern, output)}

{{#LLDMACRO}.regsub(pattern, output)}
{{#LLDMACRO}.iregsub(pattern, output)}
```

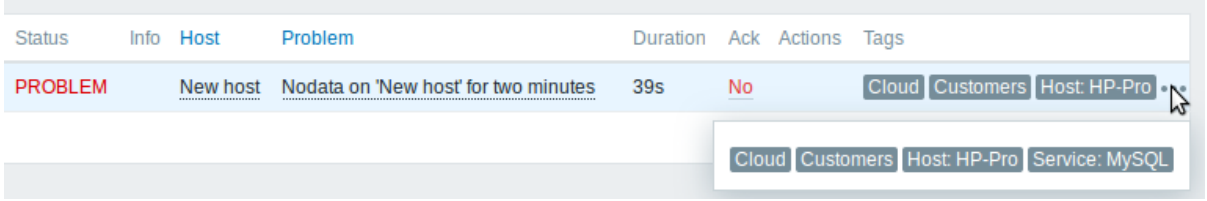
如果宏解析后的标签名或值的长度超过 255 个字符，则标签名或值将被剪切为 255 个字符。

也可参照: 在低级发现宏 中使用宏函数进行事件标记。.

查看事件标签

事件标签（如果已定义）可以在如下产生的新事件列表的页面查看:

- *Monitoring* → *Problems*
- *Monitoring* → *Problems* → *Event details*
- *Monitoring* → *Dashboard* → *Problems* widget (将鼠标移动到问题名称上时打开的弹出窗口中)



仅显示前三个标签条目。如果有三个以上的标签条目，则用三个点表示。如果将鼠标移动到这三个点上时，所有标签条目都将显示在弹出窗口中。

注意，标签的显示顺序受筛选器和页面 *Monitoring→Problems* 或仪表板小构件*Problems* 的 标签显示优先级选项的影响。

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放。迄今为止，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，一个日志触发器可以报告应用程序问题，同时有一个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都会发布获得一个问题报告怎么办？想要不需要全部的问题报告。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，通过生成相应的恢复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。

> **Attention:**
> 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）。

要 安全地配置全局关联，请遵循以下重要提示：

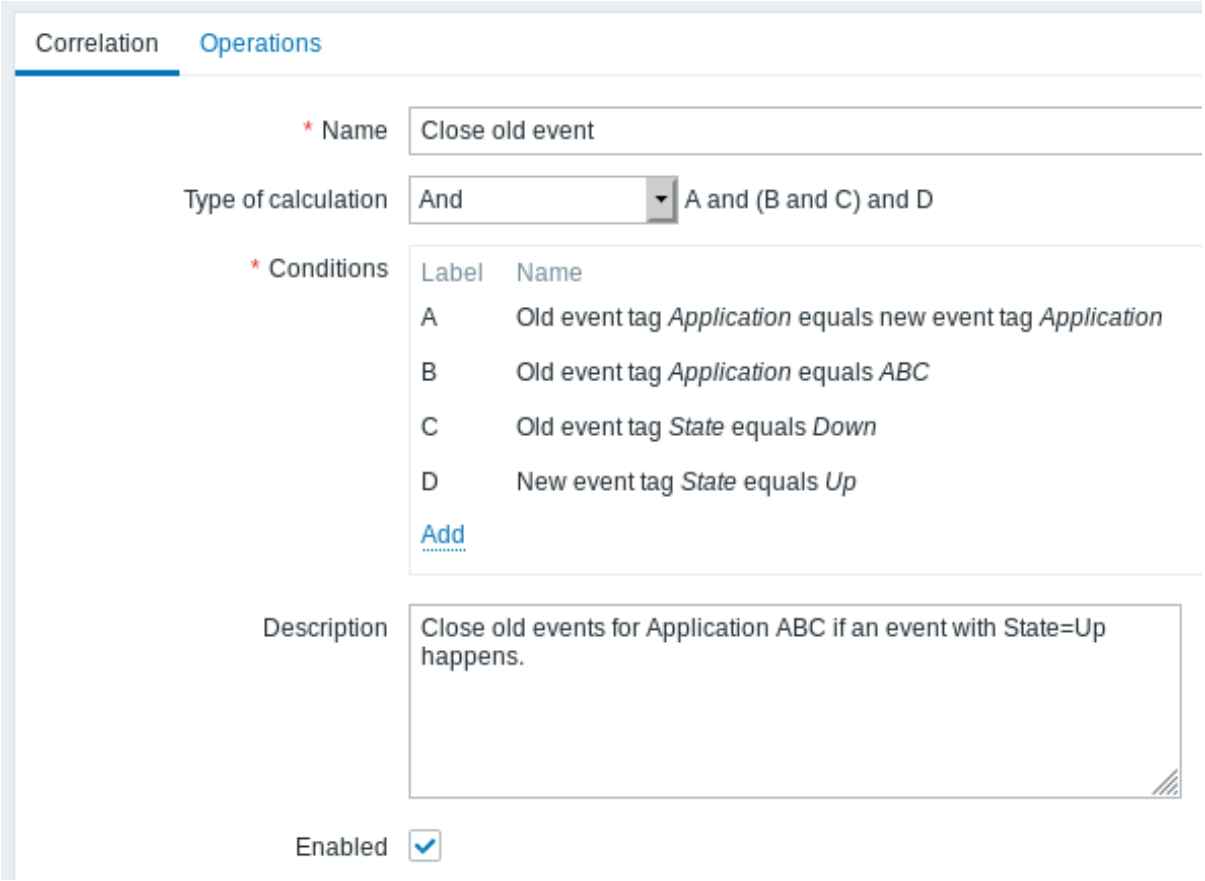
- 减少相关范围。始终为与旧事件配对的新事件设置唯一标记，并使用 新事件标记关联条件
- 使用 关闭旧事件操作时，根据旧事件添加条件（或者可以关闭所有现有问题）
- 避免使用可能最终被不同关联配置使用的常见标记名称
- 保持关联规则的数量仅限于您真正需要的数量

可参考: 已知问题。

配置

要全局配置事件关联规则：

- 打开 配置 → 事件关联页
- 单击右侧 创建相关性（点击现有规则的名称打开编辑）
- 在表单中输入关联规则的参数



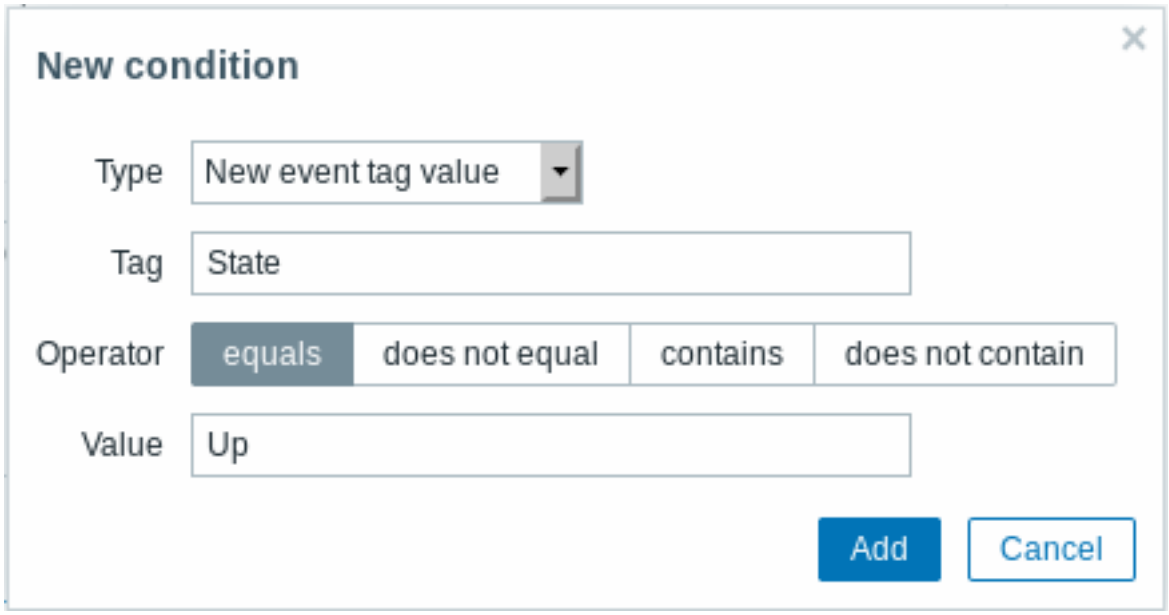
所有必填输入字段都标有红色星号。

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。详情见下方的条件配置。
说明关	规则说明。

参数描	
已启用如果	中此复选框，则将启用关联规则。

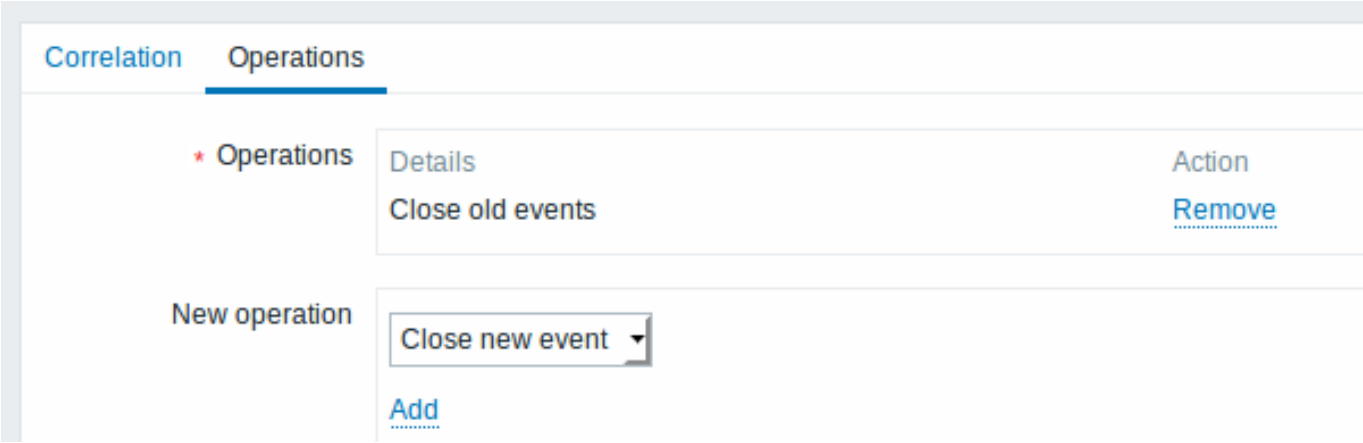
要配置一个新条件的细节，请点击条件框中的按钮 Add 。请在打开弹出窗配置条件。



参数	描
新条件设置	条关联事件的条件。注意如果没有指定旧事件条件，所有匹配的旧事件将全部被关闭。同样的，如果没有指定新事件条件，所有匹配的新事件将全部被关闭。以下条件可选 旧事件标签 - 指定匹配的旧事件标签。 新事件标签 - 指定匹配的新事件标签。 新事件主机组 - 指定匹配的新事件主机组。 事件标签对 - 同时指定匹配新事件标签和旧事件标签的值。这条规则将同时匹配新、旧事件标签的值，标签的 名称不需要匹配。这个选项对那些无法

参数描

- 在表单中选择关联规则的操作



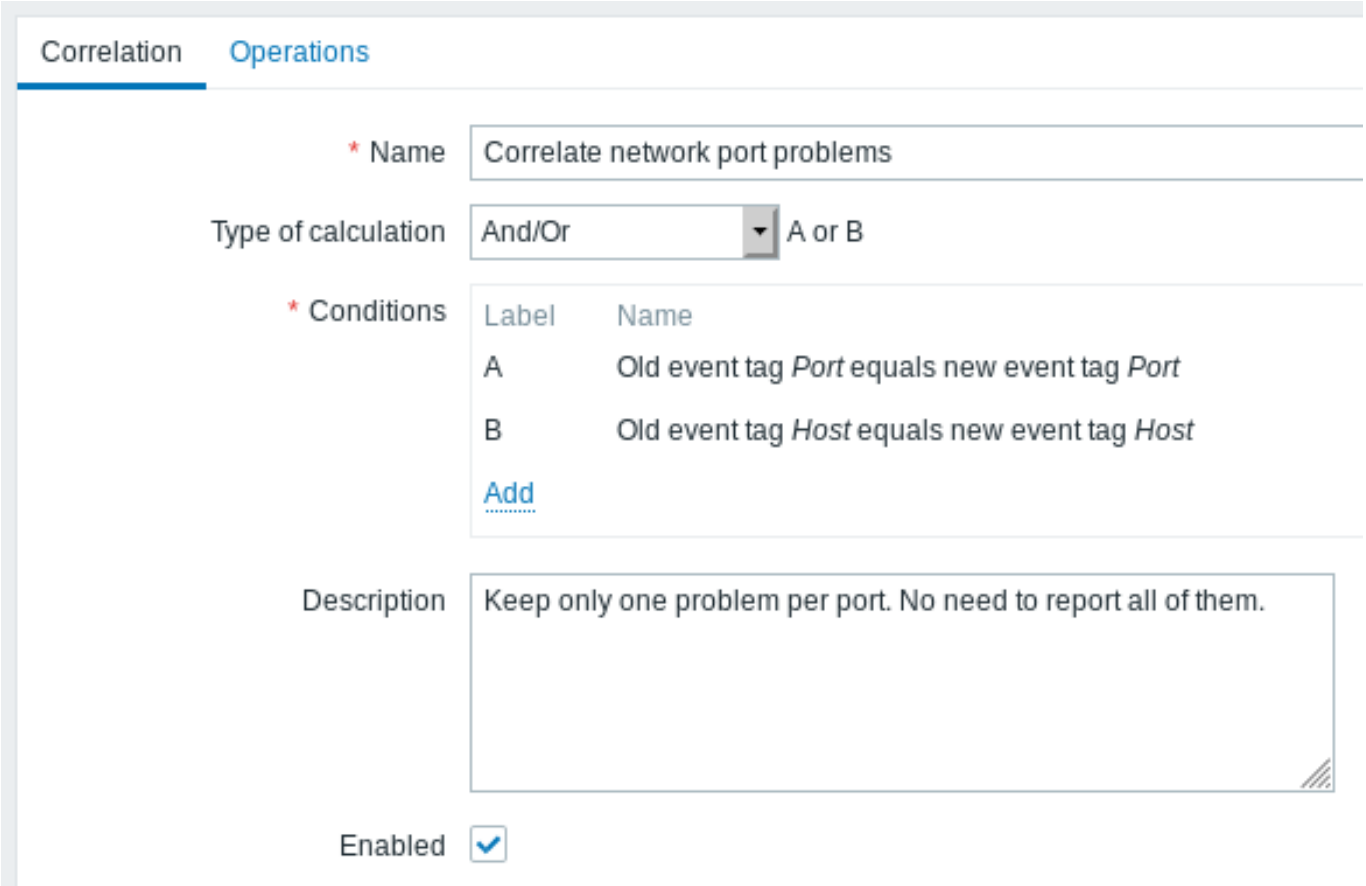
参数描	
操作从	新操作 * 框选中选择的操作列表
新操作选择	事件关联时执行的操作，然后单击添加。可以使用以下操作：关闭旧事件 - 在发生新事件时关闭旧事件。使用 关闭旧事件操作时，始终根据旧事件添加条件，或者可以关闭所有现有问题。关闭新事件 - 当事件发生时关闭新事件

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！

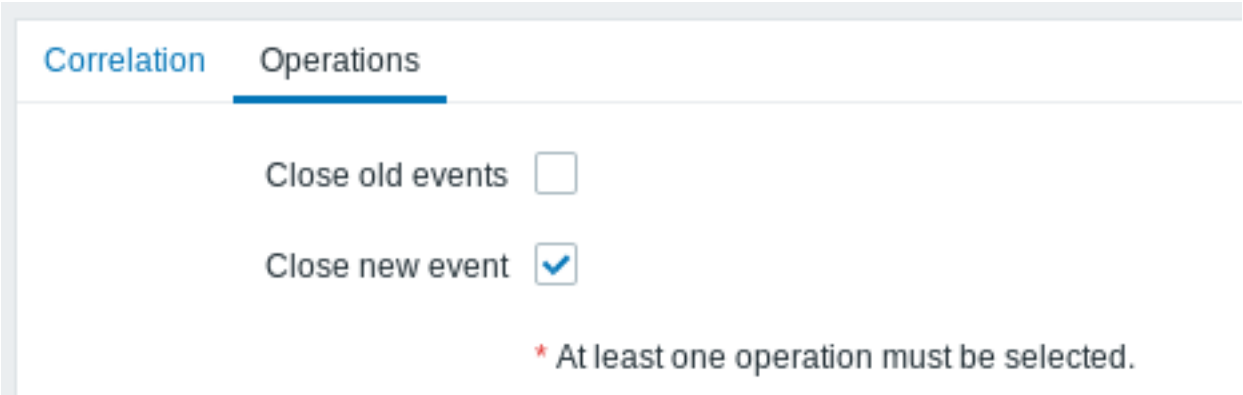
- 实际标记和标记值仅在触发器触发时可见。如果使用的正则表达式无效，则使用 * UNKNOWN * 字符串静默替换它。如果错过了具有 * UNKNOWN * 标记值的初始问题事件，则可能会出现具有相同 * UNKNOWN * 标记值的后续 OK 事件，这些事件可能会关闭它们不应关闭的问题事件。
- 如果用户使用不带宏函数的 {ITEM.VALUE} 宏作为标记值，则有 255 个字符的限制。当日志消息很长并且前 255 个字符是非特定的时，这也可能导致类似的事件标记用于不相关的问题。

示例 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* 标签，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。

6 Visualization

1 Graphs

Overview

With lots of data flowing into Zabbix, it becomes much easier for the users if they can look at a visual representation of what is going on rather than only numbers.

This is where graphs come in. Graphs allow to grasp the data flow at a glance, correlate problems, discover when something started or make a presentation of when something might turn into a problem.

Zabbix provides users with:

- built-in simple graphs of one item data
- the possibility to create more complex customized graphs
- access to a comparison of several items quickly in ad-hoc graphs
- modern customisable vector graphs

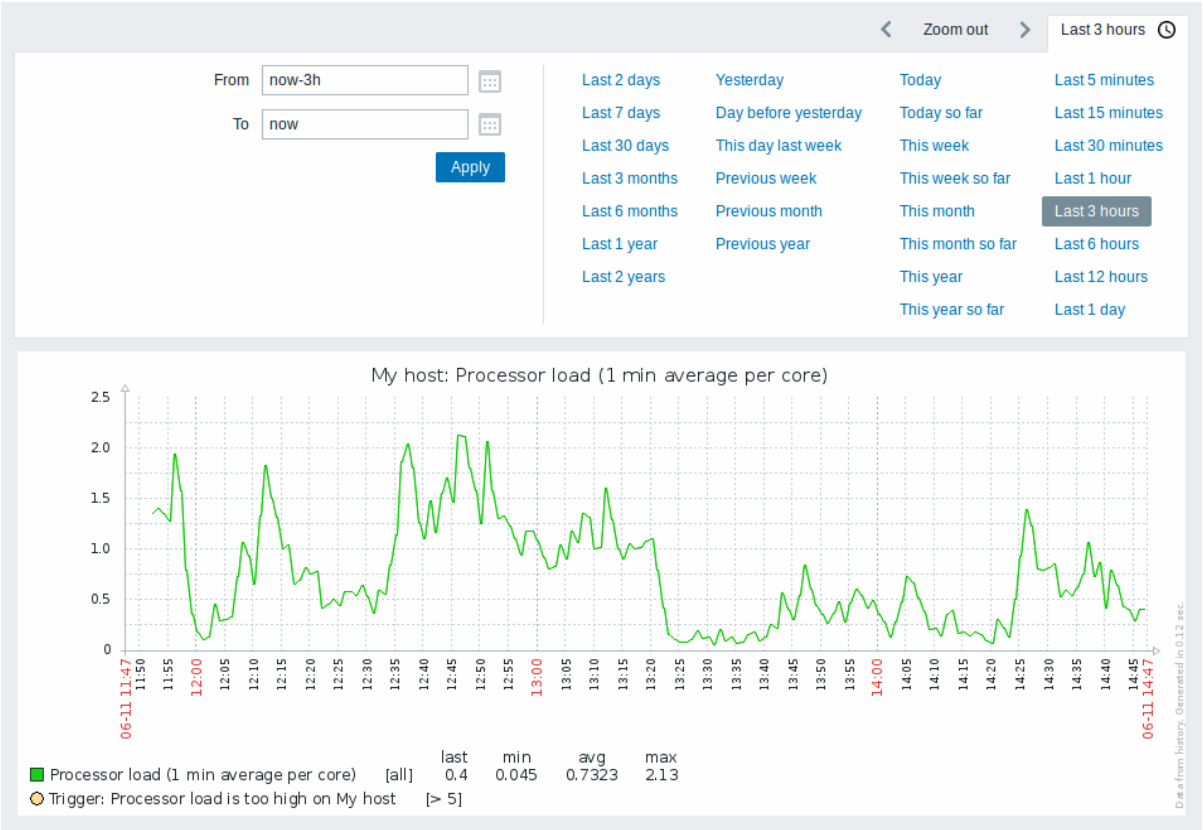
1 Simple graphs

Overview

Simple graphs are provided for the visualization of data gathered by items.

No configuration effort is required on the user part to view simple graphs. They are freely made available by Zabbix.

Just go to *Monitoring* → *Latest data* and click on the Graph link for the respective item and a graph will be displayed.



Note:
Simple graphs are provided for all numeric items. For textual items, a link to History is available in *Monitoring* → *Latest data*.

Time period selector

Take note of the time period selector above the graph. It allows to select often required periods with one mouse click.

Note that such options as *Today*, *This week*, *This month*, *This year* display the whole period, including the hours/days in the future. *Today so far*, in contrast, only displays the hours passed.

Once a period is selected, it can be moved back and forth in time by clicking on the arrow buttons. The *Zoom out* button allows to zoom out the period two times or by 50% in each direction. Zoom out is also possible by double-clicking in the graphs. The whole time period selector can be collapsed by clicking on the tab label containing the selected period string.

The *From/To* fields display the selected period in either:

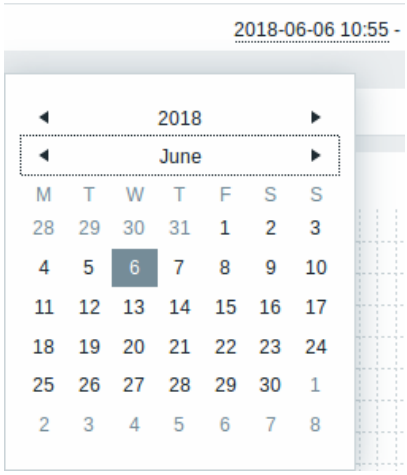
- absolute time syntax in format `Y-m-d H:i:s`

- relative time syntax, e.g.: `now-1d`

A date in relative format can contain one or several mathematical operations (- or +), e.g. `now-1d` or `now-1d-2h+5m`. For relative time the following abbreviations are supported:

- now
- s (seconds)
- m (minutes)
- h (hours)
- d (days)
- w (weeks)
- M (months)
- y (years)

It is possible to pick a specific start/end date by clicking on the calendar icon next to the *From/To* fields. In this case, the date picker pop up will open.



Within the date picker, it is possible to navigate between the blocks of year/month/date using Tab and Shift+Tab. Keyboard arrows or arrow buttons allow to select the desired value. Pressing Enter (or clicking on the desired value) activates the choice.

Another way of controlling the displayed time is to highlight an area in the graph with the left mouse button. The graph will zoom into the highlighted area once you release the left mouse button.

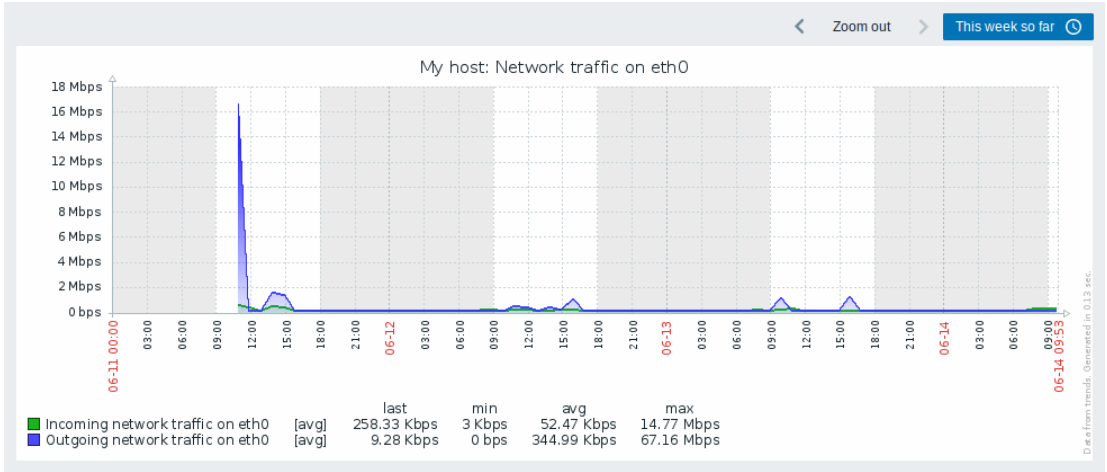
In case no time value is specified or field is left blank, time value will be set to "00:00:00". This doesn't apply to today's date selection: in that case time will be set to current value.

Recent data vs longer periods

For very recent data a **single** line is drawn connecting each received value. The single line is drawn as long as there is at least one horizontal pixel available for one value.

For data that show a longer period **three lines** are drawn - a dark green one shows the average, while a light pink and a light green line shows the maximum and minimum values at that point in time. The space between the highs and the lows is filled with yellow background.

Working time (working days) is displayed in graphs as a white background, while non-working time is displayed in gray (with the *Original blue* default frontend theme).

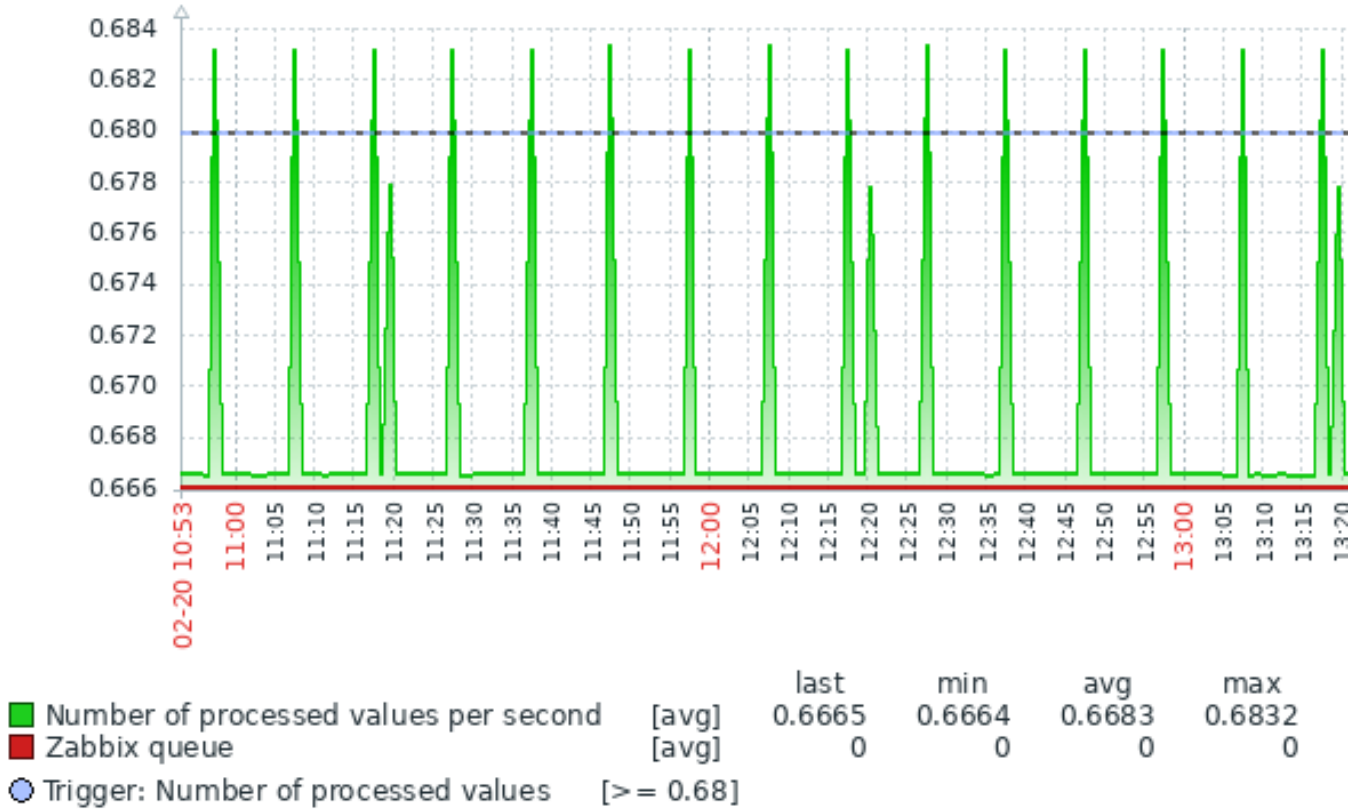


Working time is always displayed in simple graphs, whereas displaying it in custom graphs is a user preference.

Working time is not displayed if the graph shows more than 3 months.

Trigger lines

Simple triggers are displayed as lines with black dashes over trigger severity color -- take note of the blue line on the graph and the trigger information displayed in the legend. Up to 3 trigger lines can be displayed on the graph; if there are more triggers then the triggers with lower severity are prioritized. Triggers are always displayed in simple graphs, whereas displaying them in custom graphs is a user preference.



Generating from history/trends

Graphs can be drawn based on either item history or trends.

For the users who have frontend debug mode activated, a gray, vertical caption is displayed at the bottom right of a graph indicating where the data come from.

Several factors influence whether history of trends is used:

- longevity of item history. For example, item history can be kept for 14 days. In that case, any data older than the fourteen days will be coming from trends.
- data congestion in the graph. If the amount of seconds to display in a horizontal graph pixel exceeds 3600/16, trend data are displayed (even if item history is still available for the same period).
- if trends are disabled, item history is used for graph building - if available for that period. This is supported starting with Zabbix 2.2.1 (before, disabled trends would mean an empty graph for the period even if item history was available).

Absence of data

For items with a regular update interval, nothing is displayed in the graph if item data are not collected.

However, for trapper items and items with a scheduled update interval (and regular update interval set to 0), a straight line is drawn leading up to the first collected value and from the last collected value to the end of graph; the line is on the level of the first/last value respectively.

Switching to raw values

A dropdown on the upper right allows to switch from the simple graph to the *Values/500 latest values* listings. This can be useful for viewing the numeric values making up the graph.

The values represented here are raw, i.e. no units or postprocessing of values is used. Value mapping, however, is applied.

Known issues

See known issues for graphs.

2 Custom graphs

Overview

Custom graphs, as the name suggests, offer customization capabilities.

While simple graphs are good for viewing data of a single item, they do not offer configuration capabilities.

Thus, if you want to change graph style or the way lines are displayed or compare several items, for example incoming and outgoing traffic in a single graph, you need a custom graph.

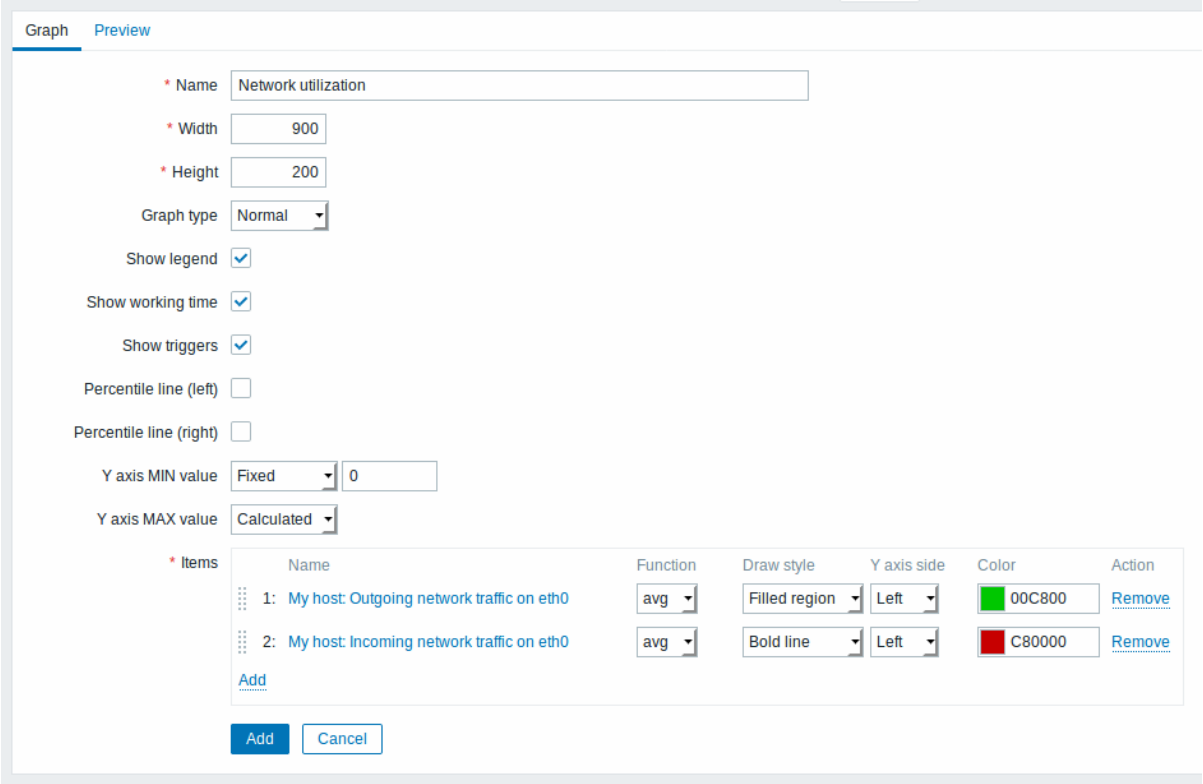
Custom graphs are configured manually.

They can be created for a host or several hosts or for a single template.

Configuring custom graphs

To create a custom graph, do the following:

- Go to *Configuration → Hosts (or Templates)*
- Click on *Graphs* in the row next to the desired host or template
- In the Graphs screen click on *Create graph*
- Edit graph attributes



All mandatory input fields are marked with a red asterisk.

Graph attributes:

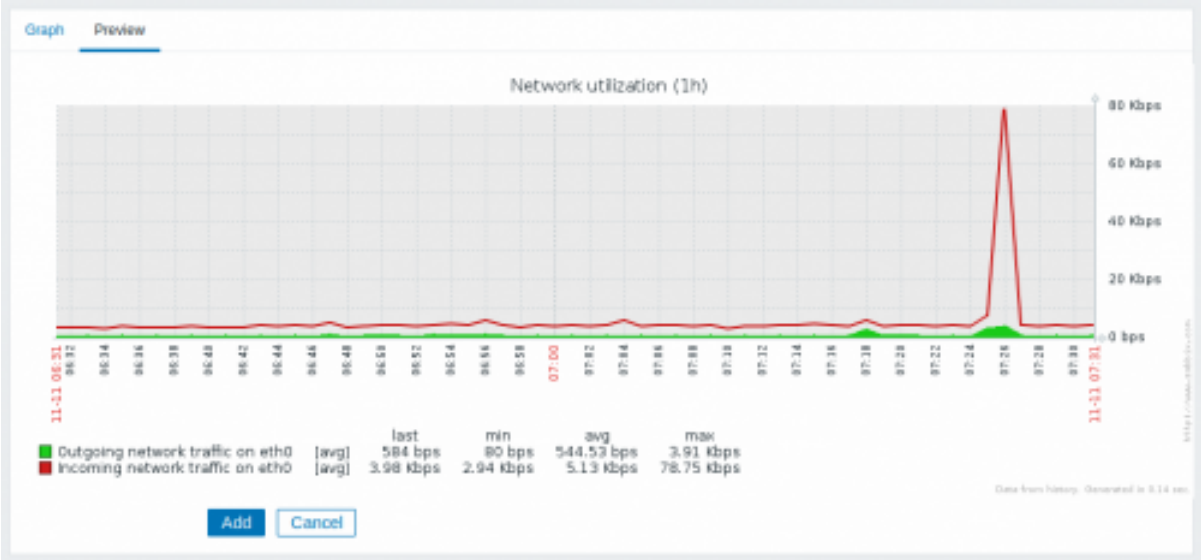
Parameter	Description
Name	Unique graph name. Starting with Zabbix 2.2, item values can be referenced in the name by using simple macros with the standard `{host:key.func(param)}` syntax. Only **avg**, **last**, **max** and **min** as functions with seconds as parameter are supported within this macro. {HOST.HOST<1-9>} macros are supported for the use within this macro, referencing the first, second, third, etc. host in the graph, for example `{{HOST.HOST1}:key.func(param)}`.
Width	Graph width in pixels (for preview and pie/exploded graphs only).

Parameter		Description
Height		Graph height in pixels.
Graph type		Graph type: **Normal** - normal graph, values displayed as lines **Stacked** - stacked graph, filled areas displayed **Pie** - pie graph **Exploded** - ”exploded” pie graph, portions displayed as ”cut out” of the pie
Show legend		Checking this box will set to display the graph legend.
Show working time		If selected, non-working hours will be shown with gray background. Not available for pie and exploded pie graphs.
Show triggers		If selected, simple triggers will be displayed as lines with black dashes over trigger severity color. Not available for pie and exploded pie graphs.
Percentile line (left)		Display percentile for left Y axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 per cent of the values fall under. Displayed as a bright green line. Only available for normal graphs.
Percentile line (right)		Display percentile for right Y axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 per cent of the values fall under. Displayed as a bright red line. Only available for normal graphs.
Y axis MIN value		Minimum value of Y axis: **Calculated** - Y axis minimum value will be automatically calculated **Fixed** - fixed minimum value for Y axis. Not available for pie and exploded pie graphs. **Item** - last value of the selected item will be the minimum value
Y axis MAX value		Maximum value of Y axis: **Calculated** - Y axis maximum value will be automatically calculated **Fixed** - fixed maximum value for Y axis. Not available for pie and exploded pie graphs. **Item** - last value of the selected item will be the maximum value
3D view		Enable 3D style. For pie and exploded pie graphs only.
Items		Items, data of which are to be displayed in this graph. Click on *Add* to select items. You can also select various displaying options (function, draw style, left/right axis display, color).
	Sort order (0→100)	Draw order. 0 will be processed first. Can be used to draw lines or regions behind (or in front of) another. You can drag and drop items by the arrow in the beginning of line to set the sort order or which item is displayed in front of the other.
	Name	Name of the selected item is displayed as a link. Clicking on the link opens the list of other available items.
	Type	Type (only available for pie and exploded pie graphs): **Simple** - value of the item is represented proportionally on the pie **Graph sum** - value of the item represents the whole pie Note that coloring of the ”graph sum” item will only be visible to the extent that it is not taken up by ”proportional” items.

Parameter		Description
	Function	Select what values will be displayed when more than one value exists per vertical graph pixel for an item: **all** - display all possible values (minimum, maximum, average) in the graph. Note that for shorter periods this setting has no effect; only for longer periods, when data congestion in a vertical graph pixel increases, 'all' starts displaying minimum, maximum and average values. This function is only available for *Normal* graph type. See also: Generating graphs from history/trends. **avg** - display the average values **last** - display the latest values. This function is only available if either *Pie/Exploded pie* is selected as graph type. **max** - display the maximum values **min** - display the minimum values
	Draw style	Select the draw style (only available for normal graphs; for stacked graphs filled region is always used) to apply to the item data - *Line*, *Bold line*, *Filled region*, *Dot*, *Dashed line*, *Gradient line*.
	Y axis side	Select the Y axis side to show the item data - *Left*, *Right*.
	Color	Select the color to apply to the item data.

Graph preview

In the *Preview* tab, a preview of the graph is displayed so you can immediately see what you are creating.



Note that the preview will not show any data for template items.

In this example, pay attention to the dashed bold line displaying the trigger level and the trigger information displayed in the legend.

Note:
No more than 3 trigger lines can be displayed. If there are more triggers then the triggers with lower severity are prioritized for display.

If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

3 Ad-hoc graphs

Overview

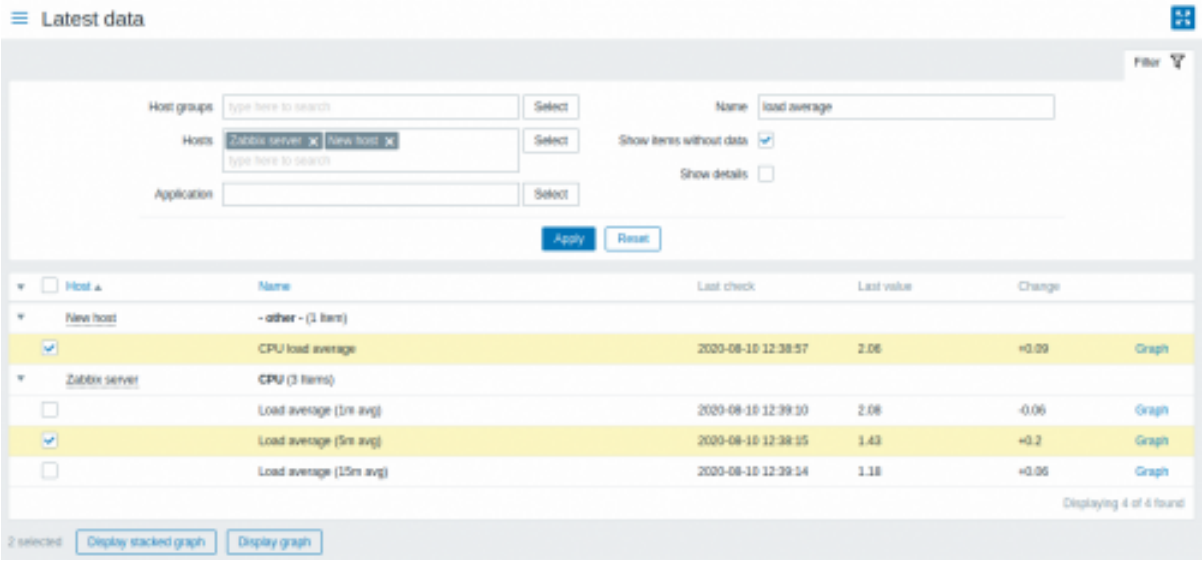
While a simple graph is great for accessing data of one item and custom graphs offer customization options, none of the two allow to quickly create a comparison graph for multiple items with little effort and no maintenance.

To address this issue, since Zabbix 2.4 it is possible to create ad-hoc graphs for several items in a very quick way.

Configuration

To create an ad-hoc graph, do the following:

- Go to *Monitoring → Latest data*
- Use filter to display items that you want
- Mark checkboxes of the items you want to graph
- Click on *Display stacked graph* or *Display graph* buttons



Your graph is created instantly:

Note that to avoid displaying too many lines in the graph, only the average value for each item is displayed (min/max value lines are not displayed). Triggers and trigger information is not displayed in the graph.

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

4 Aggregation in graphs

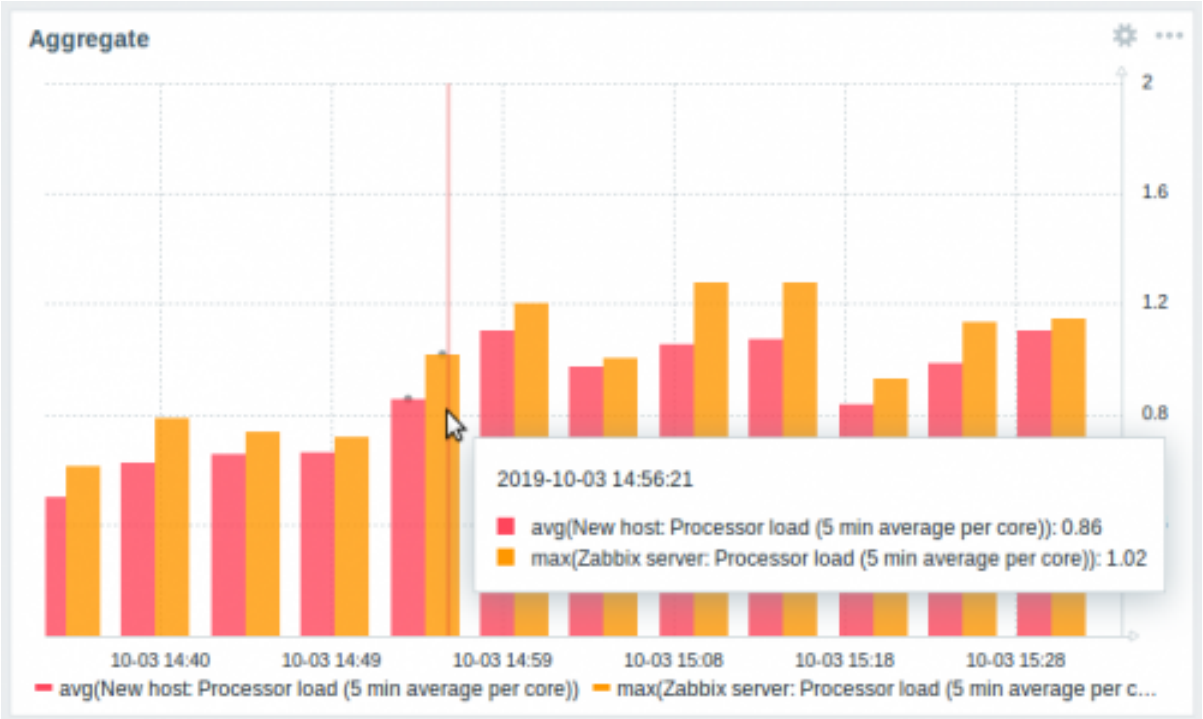
Overview

The aggregation functions, available in the graph widget of the dashboard, allow to display an aggregated value for the chosen interval (5 minutes, an hour, a day), instead of all values.

The aggregation options are as follows:

- min
- max
- avg
- count
- sum
- first (first value displayed)
- last (last value displayed)

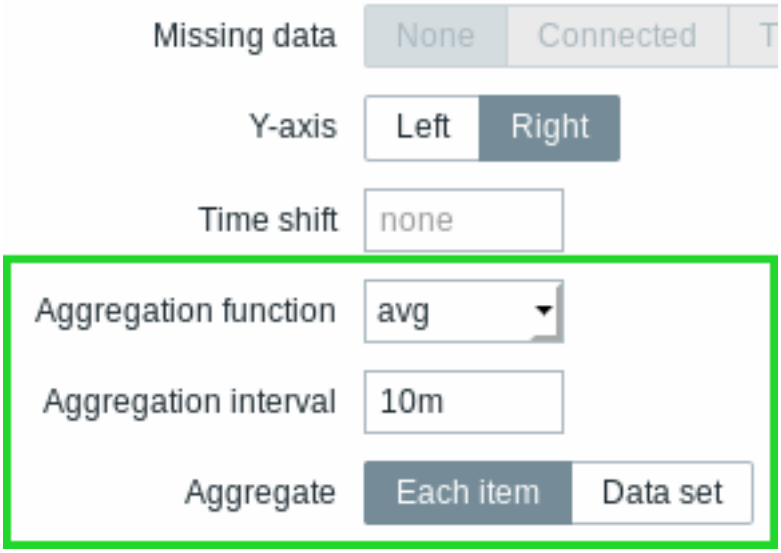
The most exciting use of data aggregation is the possibility to create nice side-by-side comparisons of data for some period:



When hovering over a point in time in the graph, date and time is displayed, in addition to items and their aggregated values. Items are displayed in parentheses, prefixed by the aggregation function used. Note that this is the date and time of the point in graph, not of the actual values.

Configuration

The options for aggregation are available in data set settings when configuring a graph widget.



You may pick the aggregation function and the time interval. As the data set may comprise several items, there is also another option allowing to show aggregated data for each item separately or for all data set items as one aggregated value.

See the widget configuration page for more details.

Use cases

Average request count to Nginx server

View the average request count per second per day to Nginx server:

- add the request count per second item to the data set
- select the aggregate function avg and specify interval 1d
- a bar graph is displayed, where each bar represents the average number of requests per second per day

Minimum weekly disk space among clusters

View the lowest disk space among clusters over a week.

- add to the data set: hosts `cluster*`, key "Free disk space on /data"
- select the aggregate function `min` and specify interval `1w`
- a bar graph is displayed, where each bar represents the minimum disk space per week for each /data volume of the cluster

2 Network maps

Overview

If you have a network to look after, you may want to have an overview of your infrastructure somewhere. For that purpose you can create maps in Zabbix - of networks and of anything you like.

All users can create network maps. The maps can be public (available to all users) or private (available to selected users).

Proceed to configuring a network map.

1 Configuring a network map

Overview

Configuring a map in Zabbix requires that you first create a map by defining its general parameters and then you start filling the actual map with elements and their links.

You can populate the map with elements that are a host, a host group, a trigger, an image or another map.

Icons are used to represent map elements. You can define the information that will be displayed with the icons and set that recent problems are displayed in a special way. You can link the icons and define information to be displayed on the links.

You can add custom URLs to be accessible by clicking on the icons. Thus you may link a host icon to host properties or a map icon to another map.

Maps are managed in *Monitoring* → *Maps*, where they can be configured, managed and viewed. In the monitoring view you can click on the icons and take advantage of the links to some scripts and URLs.

Network maps are based on vector graphics (SVG) since Zabbix 3.4.

Public and private maps

All users in Zabbix (including non-admin users) can create network maps. Maps have an owner - the user who created them. Maps can be made public or private.

- *Public* maps are visible to all users, although to see it the user must have read access to at least one map element. Public maps can be edited in case a user/ user group has read-write permissions for this map and at least read permissions to all elements of the corresponding map including triggers in the links.
- *Private* maps are visible only to their owner and the users/user groups the map is shared with by the owner. Regular (non-Super admin) users can only share with the groups and users they are member of. Admin level users can see private maps regardless of being the owner or belonging to the shared user list. Private maps can be edited by the owner of the map and in case a user/ user group has read-write permissions for this map and at least read permissions to all elements of the corresponding map including triggers in the links.

Map elements that the user does not have read permission to are displayed with a grayed out icon and all textual information on the element is hidden. However, trigger label is visible even if the user has no permission to the trigger.

To add an element to the map the user must also have at least read permission to it.

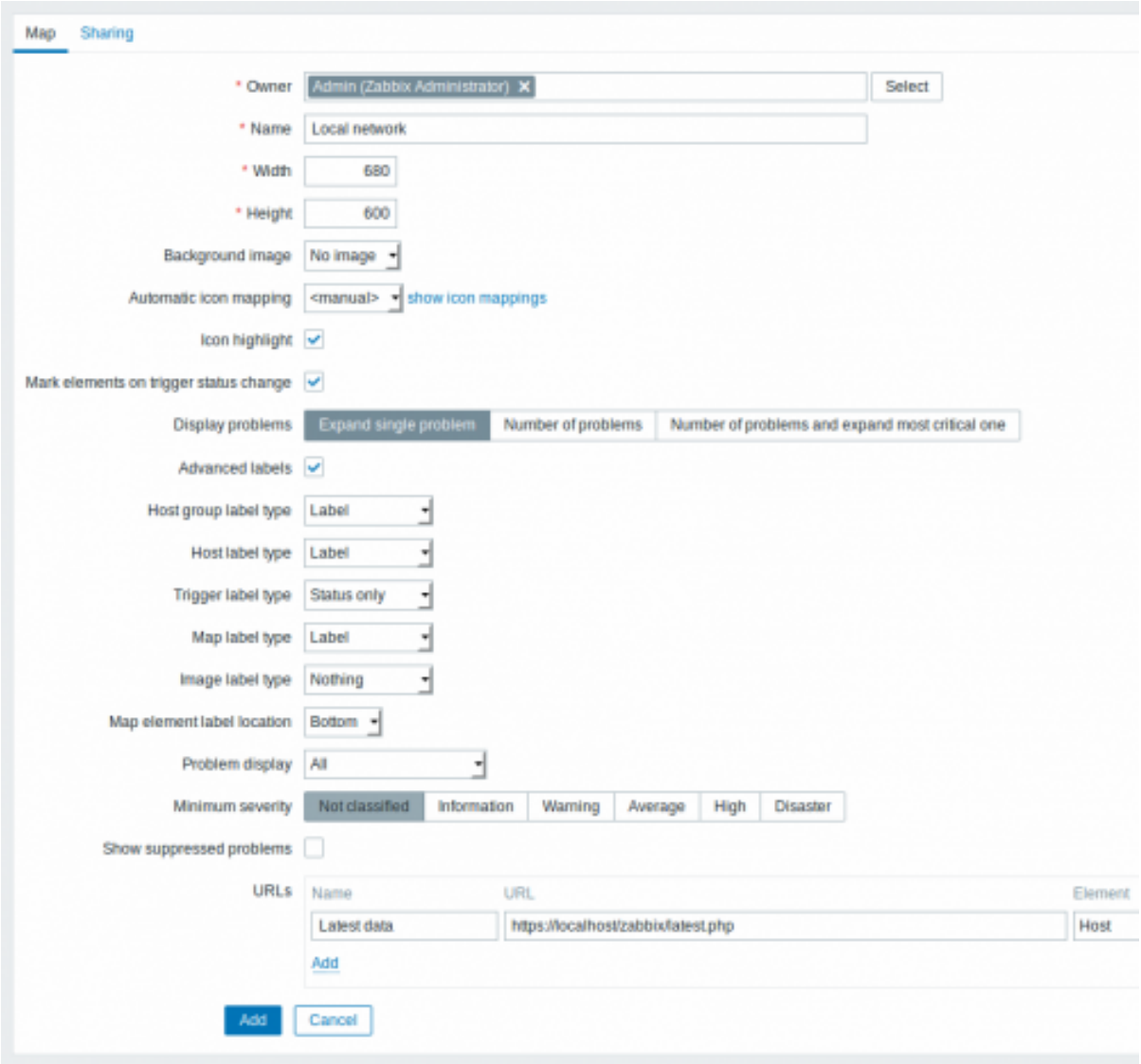
Creating a map

To create a map, do the following:

- Go to *Monitoring* → *Maps*
- Go to the view with all maps
- Click on *Create map*

You can also use the *Clone* and *Full clone* buttons in the configuration form of an existing map to create a new map. Clicking on *Clone* will retain general layout attributes of the original map, but no elements. *Full clone* will retain both the general layout attributes and all elements of the original map.

The **Map** tab contains general map attributes:



All mandatory input fields are marked with a red asterisk.

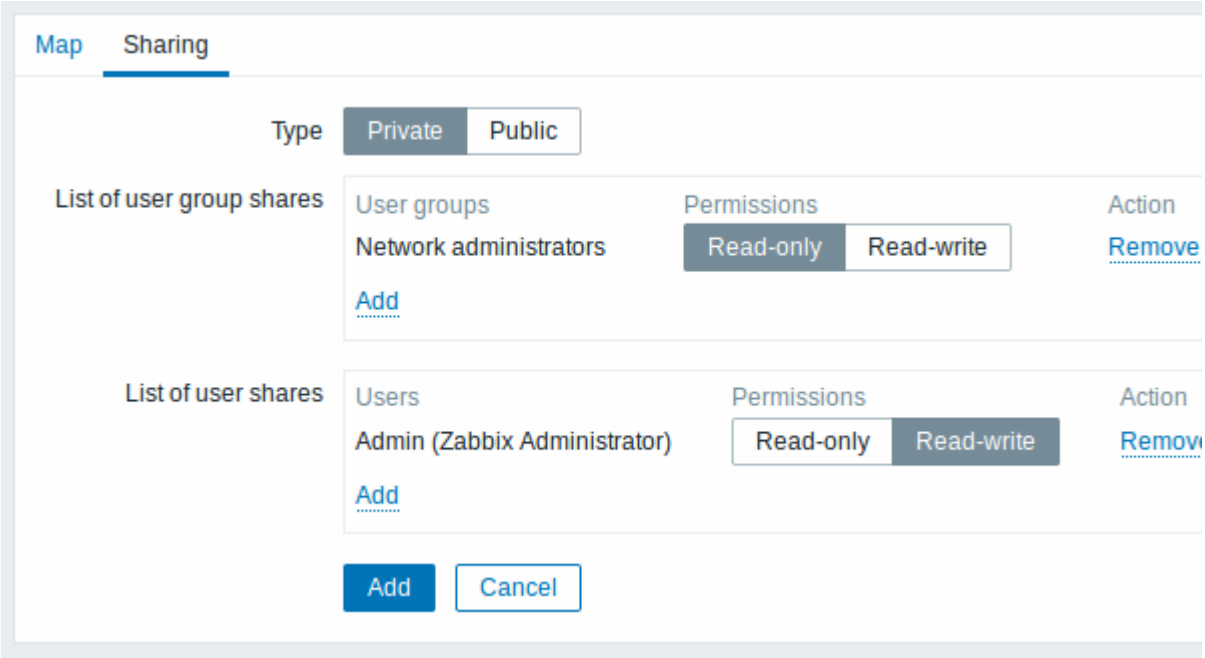
General map attributes:

Parameter	Description
Owner	Name of map owner.
Name	Unique map name.
Width	Map width in pixels.
Height	Map height in pixels.
Background image	Use background image: **No image** - no background image (white background) **Image** - selected image to be used as a background image. No scaling is performed. You may use a geographical map or any other image to enhance your map.
Automatic icon mapping	You can set to use an automatic icon mapping, configured in *Administration → General → Icon mapping*. Icon mapping allows to map certain icons against certain host inventory fields.
Icon highlighting	If you check this box, map elements will receive highlighting. Elements with an active trigger will receive a round background, in the same color as the highest severity trigger. Moreover, a thick green line will be displayed around the circle, if all problems are acknowledged. Elements with "disabled" or "in maintenance" status will get a square background, gray and orange respectively. See also: Viewing maps

Parameter	Description
Mark elements on trigger status change	A recent change of trigger status (recent problem or resolution) will be highlighted with markers (inward-pointing red triangles) on the three sides of the element icon that are free of the label. Markers are displayed for 30 minutes.
Display problems	Select how problems are displayed with a map element: **Expand single problem** - if there is only one problem, the problem name is displayed. Otherwise, the total number of problems is displayed. **Number of problems** - the total number of problems is displayed **Number of problems and expand most critical one** - name of the most critical problem and the total number of problems is displayed. 'Most critical' is determined based on problem severity and, if equal, problem event ID (higher ID or later problem displayed first). For a *trigger map element* it is based on problem severity and, if equal, trigger position in the trigger list. In case of multiple problems of the same trigger, the most recent one will be displayed.
Advanced labels	If you check this box you will be able to define separate label types for separate element types.
Map element label type	Label type used for map elements: **Label** - map element label **IP address** - IP address **Element name** - element name (for example, host name) **Status only** - status only (OK or PROBLEM) **Nothing** - no labels are displayed
Map element label location	Label location in relation to the map element: **Bottom** - beneath the map element **Left** - to the left **Right** - to the right **Top** - above the map element
Problem display	Display problem count as: **All** - full problem count will be displayed **Separated** - unacknowledged problem count will be displayed separated as a number of the total problem count **Unacknowledged only** - only the unacknowledged problem count will be displayed
Minimum trigger severity	Problems below the selected minimum severity level will not be displayed in the map. For example, with *Warning* selected, changes with *Information* and *Not classified* level triggers will not be reflected in the map. This parameter is supported starting with Zabbix 2.2.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
URLs	URLs for each element type can be defined (with a label). These will be displayed as links when a user clicks on the element in the map viewing mode. Macros can be used in map URL names and values. For a full list, see supported macros and search for 'map URL names and values'.

Sharing

The **Sharing** tab contains the map type as well as sharing options (user groups, users) for private maps:



Parameter	Description
Type	Select map type: **Private** - map is visible only to selected user groups and users **Public** - map is visible to all
List of user group shares	Select user groups that the map is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the map is accessible to. You may allow read-only or read-write access.

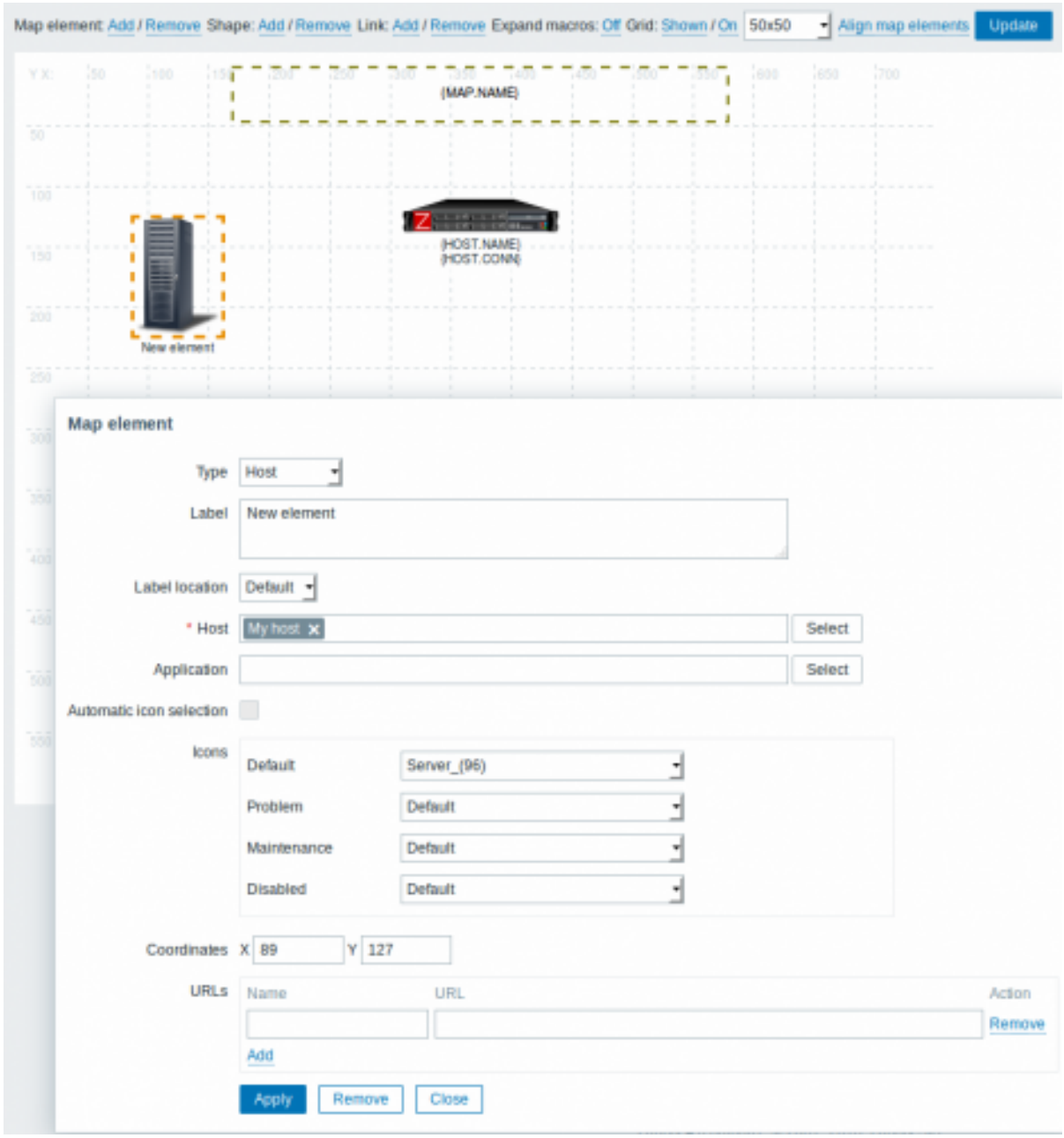
When you click on *Add* to save this map, you have created an empty map with a name, dimensions and certain preferences. Now you need to add some elements. For that, click on *Constructor* in the map list to open the editable area.

Adding elements

To add an element, click on *Add* next to `Map element`. The new element will appear at the top left corner of the map. Drag and drop it wherever you like.

Note that with the Grid option "On", elements will always align to the grid (you can pick various grid sizes from the dropdown, also hide/show the grid). If you want to put elements anywhere without alignment, turn the option to "Off". (Random elements can later again be aligned to the grid with the *Align map elements* button.)

Now that you have some elements in place, you may want to start differentiating them by giving names etc. By clicking on the element, a form is displayed and you can set the element type, give a name, choose a different icon etc.



Map element attributes:

Parameter	Description
Type	Type of the element: **Host** - icon representing status of all triggers of the selected host **Map** - icon representing status of all elements of a map **Trigger** - icon representing status of one or more triggers **Host group** - icon representing status of all triggers of all hosts belonging to the selected group **Image** - an icon, not linked to any resource
Label	Icon label, any string. Macros and multi-line strings can be used in labels. For a full list of supported macros, see supported macros and search for 'map element labels'.
Label location	Label location in relation to the icon: **Default** - map's default label location **Bottom** - beneath the icon **Left** - to the left **Right** - to the right **Top** - above the icon

Parameter	Description
Host	Enter the host, if the element type is 'Host'. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on 'x' to remove the selected.
Map	Select the map, if the element type is 'Map'.
Triggers	If the element type is 'Trigger', select one or more triggers in the *New triggers* field below and click on *Add*. The order of selected triggers can be changed, but only within the same severity of triggers. Multiple trigger selection also affects {HOST.*} macro resolution both in the construction and view modes. // 1 In construction mode// the first displayed {HOST.*} macros will be resolved depending on the first trigger in the list (based on trigger severity). // 2 View mode// depends on the Display problems parameter in General map attributes. * If *Expand single problem* mode is chosen the first displayed {HOST.*} macros will be resolved depending on the latest detected problem trigger (not mattering the severity) or the first trigger in the list (in case no problem detected); * If *Number of problems and expand most critical one* mode is chosen the first displayed {HOST.*} macros will be resolved depending on the trigger severity.
Host group	Enter the host group, if the element type is 'Host group'. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove the selected.
Application	You can select an application, allowing to only display problems of triggers that belong to the given application. This field is available for host and host group element types, and supported since Zabbix **2.4.0**.
Automatic icon selection	In this case an icon mapping will be used to determine which icon to display.
Icons	You can choose to display different icons for the element in these cases: default, problem, maintenance, disabled.
Coordinate X	X coordinate of the map element.
Coordinate Y	Y coordinate of the map element.
URLs	Element-specific URLs can be set for the element. These will be displayed as links when a user clicks on the element in the map viewing mode. If the element has its own URLs and there are map level URLs for its type defined, they will be combined in the same menu. Macros can be used in map element names and values. For a full list, see supported macros and search for 'map URL names and values'.

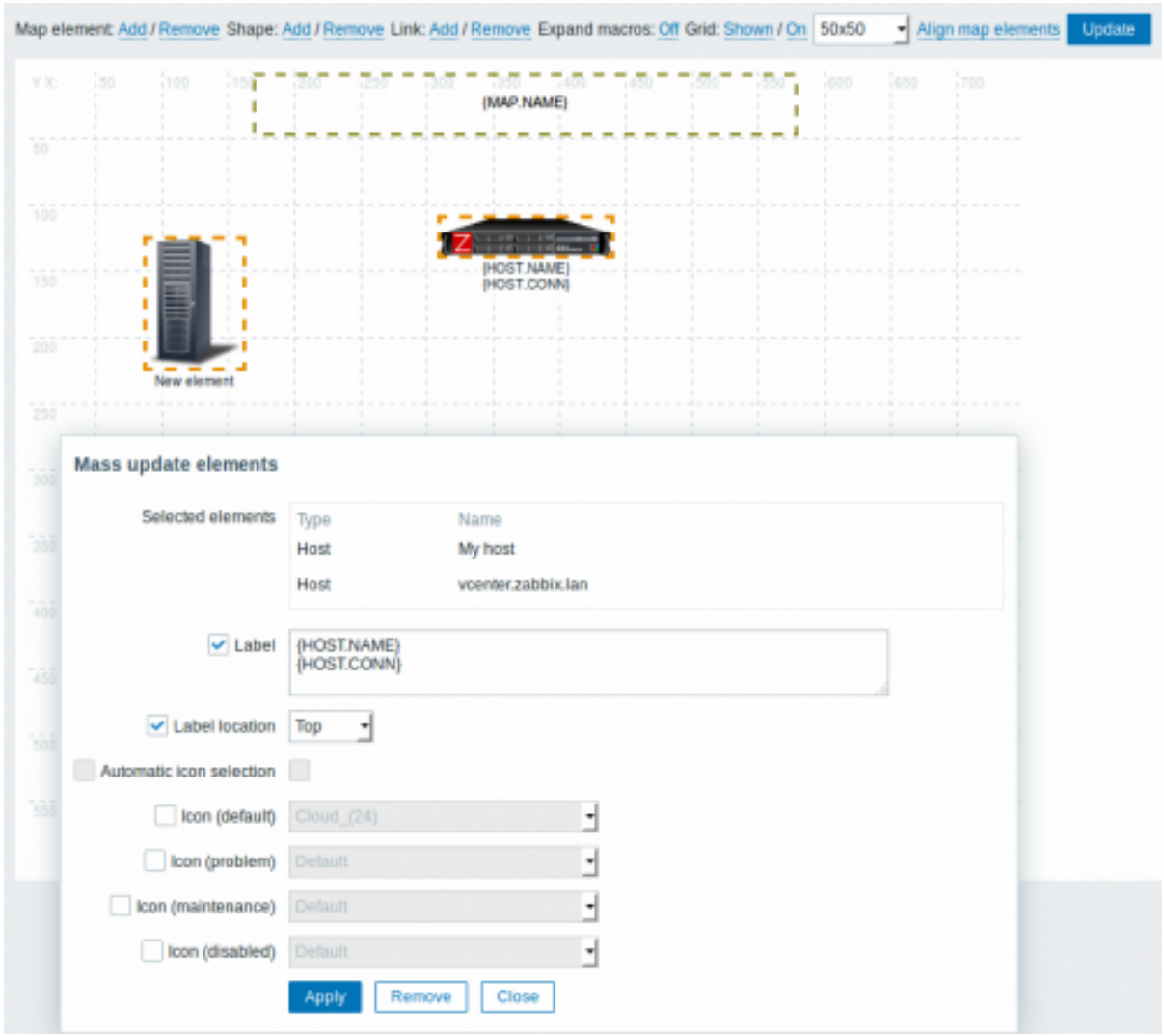
Attention:
Added elements are not automatically saved. If you navigate away from the page, all changes may be lost.
Therefore it is a good idea to click on the **Update** button in the top right corner. Once clicked, the changes are saved regardless of what you choose in the following popup.
Selected grid options are also saved with each map.

Selecting elements

To select elements, select one and then hold down *Ctrl* to select the others.

You can also select multiple elements by dragging a rectangle in the editable area and selecting all elements in it.

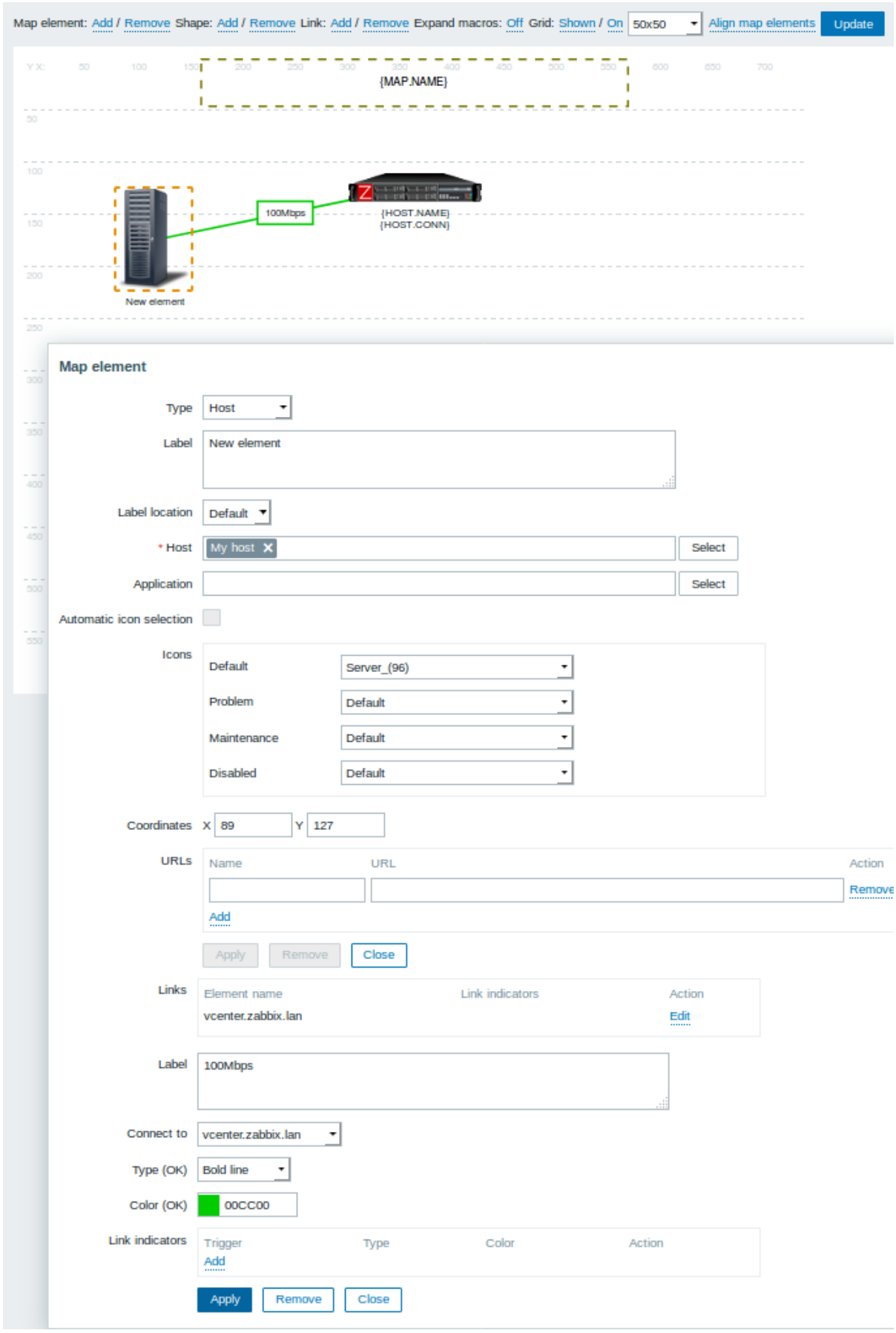
Once you select more than one element, the element property form shifts to the mass-update mode so you can change attributes of selected elements in one go. To do so, mark the attribute using the checkbox and enter a new value for it. You may use macros here (such as, say, {HOST.NAME} for the element label).



Linking elements

Once you have put some elements on the map, it is time to start linking them. To link two elements you must first select them. With the elements selected, click on *Add* next to Link.

With a link created, the single element form now contains an additional *Links* section. Click on *Edit* to edit link attributes.



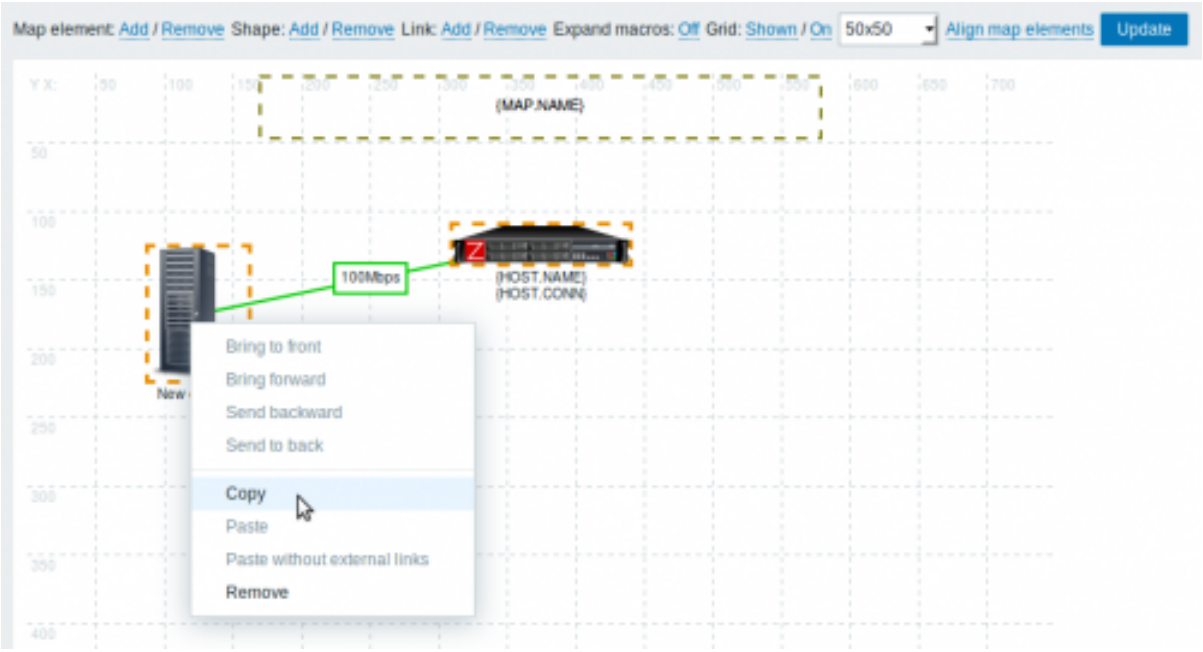
Link attributes:

Parameter	Description
Label	Label that will be rendered on top of the link. The {host:key.func(param)} macro is supported in this field, but only with `avg`, `last`, `min` and `max` trigger functions, with seconds as parameter.
Connect to	The element that the link connects to.
Type (OK)	Default link style: **Line** - single line **Bold line** - bold line **Dot** - dots **Dashed line** - dashed line
Color (OK)	Default link color.
Link indicators	List of triggers linked to the link. In case a trigger has status PROBLEM, its style is applied to the link.

Moving and copy-pasting elements

Several selected elements can be **moved** to another place in the map by clicking on one of the selected elements, holding down the mouse button and moving the cursor to the desired location.

One or more elements can be **copied** by selecting the elements, then clicking on a selected element with the right mouse button and selecting *Copy* from the menu.



To paste the elements, click on a map area with the right mouse button and select *Paste* from the menu. The *Paste without external links* option will paste the elements retaining only the links that are between the selected elements.

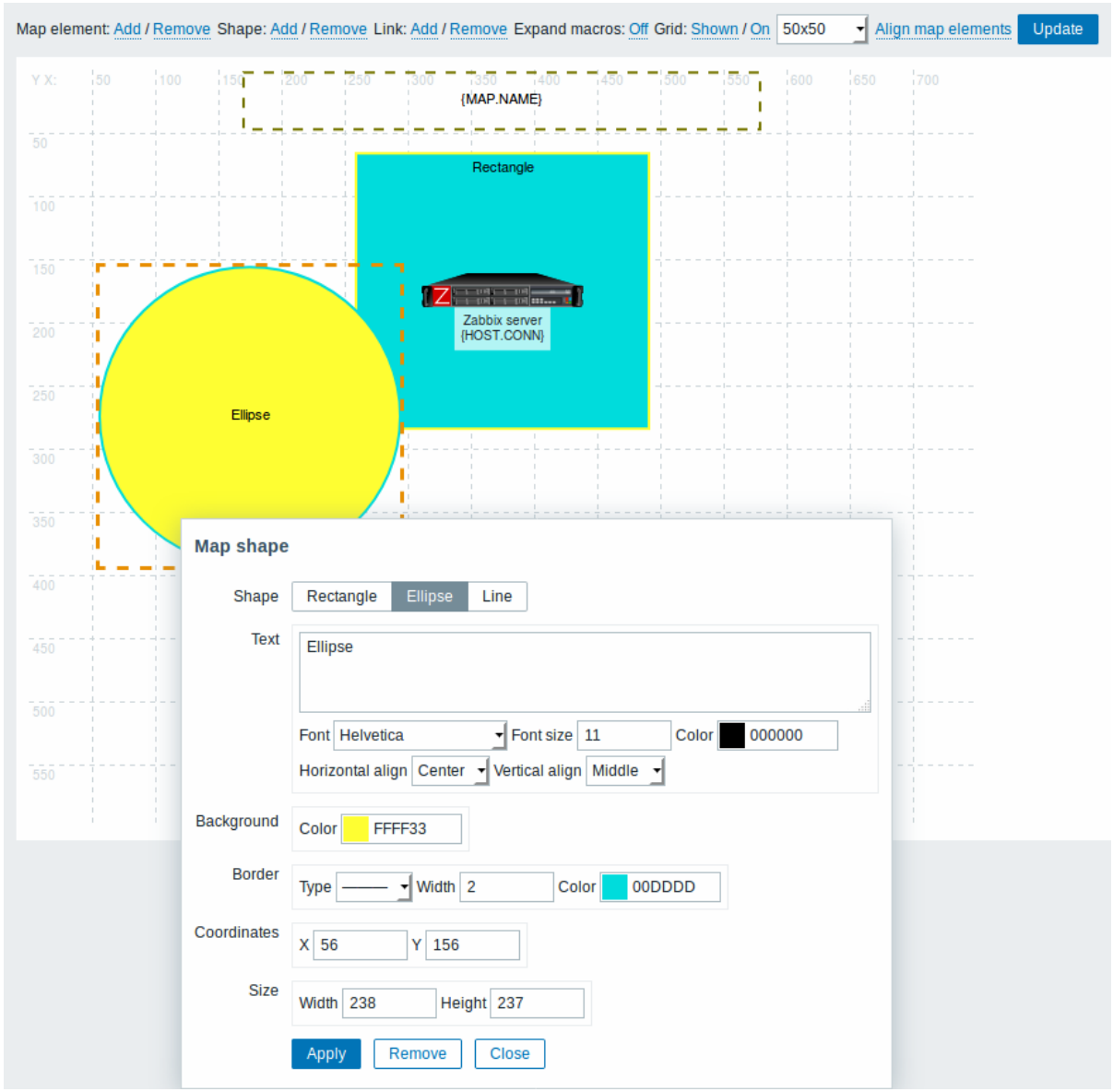
Copy-pasting works within the same browser window. Keyboard shortcuts are not supported.

Adding shapes

In addition to map elements, it is also possible to add some shapes. Shapes are not map elements; they are just a visual representation. For example, a rectangle shape can be used as a background to group some hosts. Rectangle and ellipse shapes can be added.

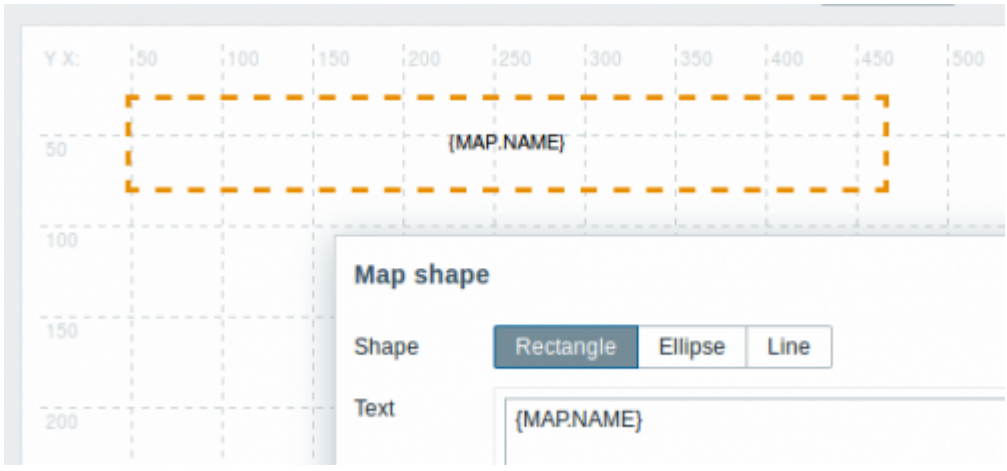
To add a shape, click on *Add* next to Shape. The new shape will appear at the top left corner of the map. Drag and drop it wherever you like.

A new shape is added with default colors. By clicking on the shape, a form is displayed and you can customize the way a shape looks, add text, etc.



To select shapes, select one and then hold down *Ctrl* to select the others. With several shapes selected, common properties can be mass updated, similarly as with elements.

Text can be added in the shapes. To display text only the shape can be made invisible by removing the shape border (select 'None' in the *Border* field). For example, take note of how the {MAP.NAME} macro, visible in the screenshot above, is actually a rectangle shape with text, which can be seen when clicking on the macro:



{MAP.NAME} resolves to the configured map name, when viewing the map.

If hyperlinks are used in the text, they become clickable when viewing the map.

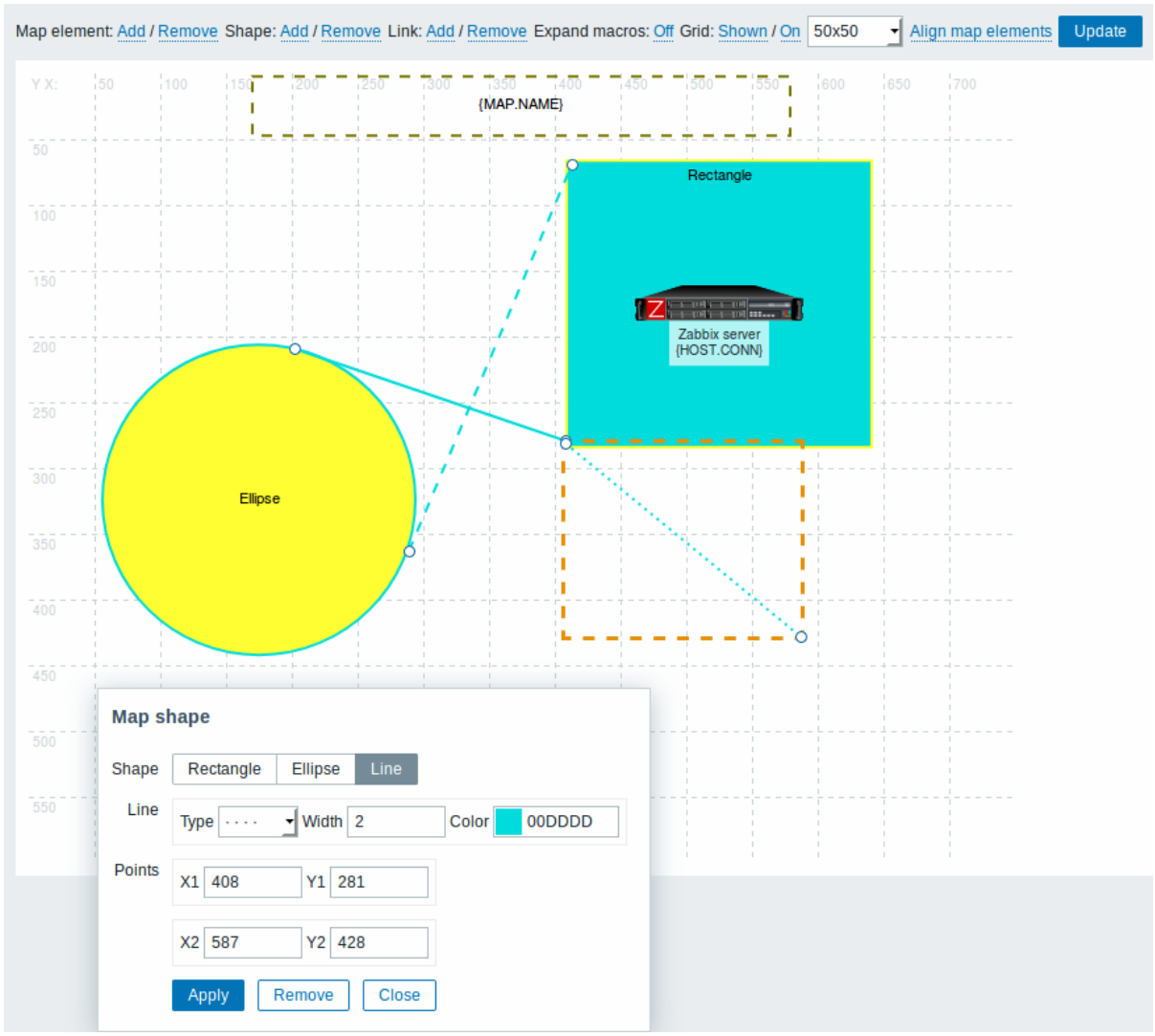
Line wrapping for text is always "on" within shapes. However, within an ellipse the lines are wrapped as though the ellipse were

a rectangle. Word wrapping is not implemented, so long words (words that do not fit the shape) are not wrapped, but are masked (constructor page) or clipped (other pages with maps).

Adding lines

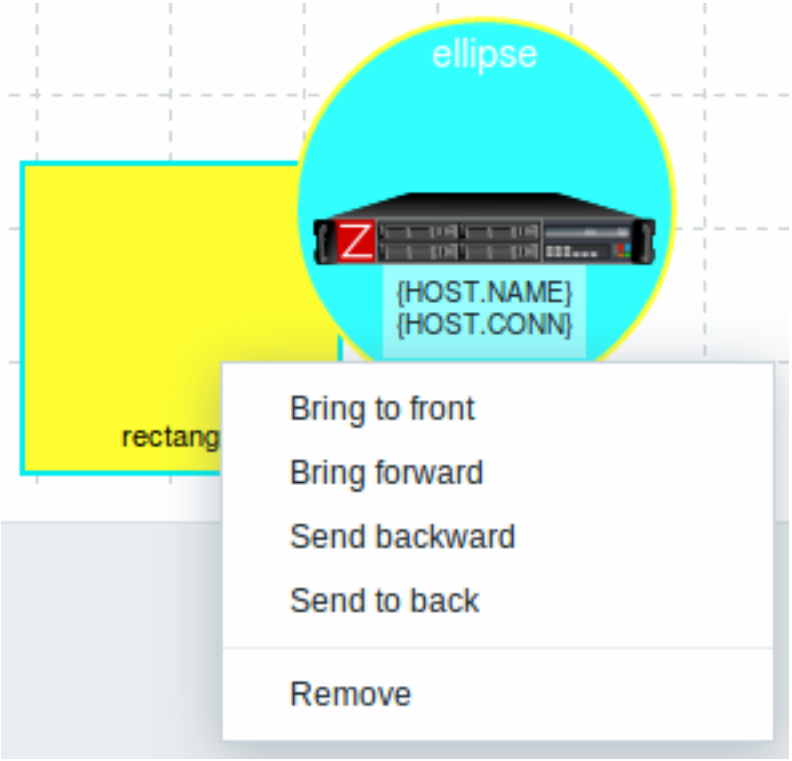
In addition to shapes, it is also possible to add some lines. Lines can be used to link elements or shapes in a map.

To add a line, click on *Add* next to Shape. A new shape will appear at the top left corner of the map. Select it and click on *Line* in the editing form to change the shape into a line. Then adjust line properties, such as line type, width, color, etc.



Ordering shapes and lines

To bring one shape in front of the other (or vice versa) click on the shape with the right mouse button bringing up the map shape menu.

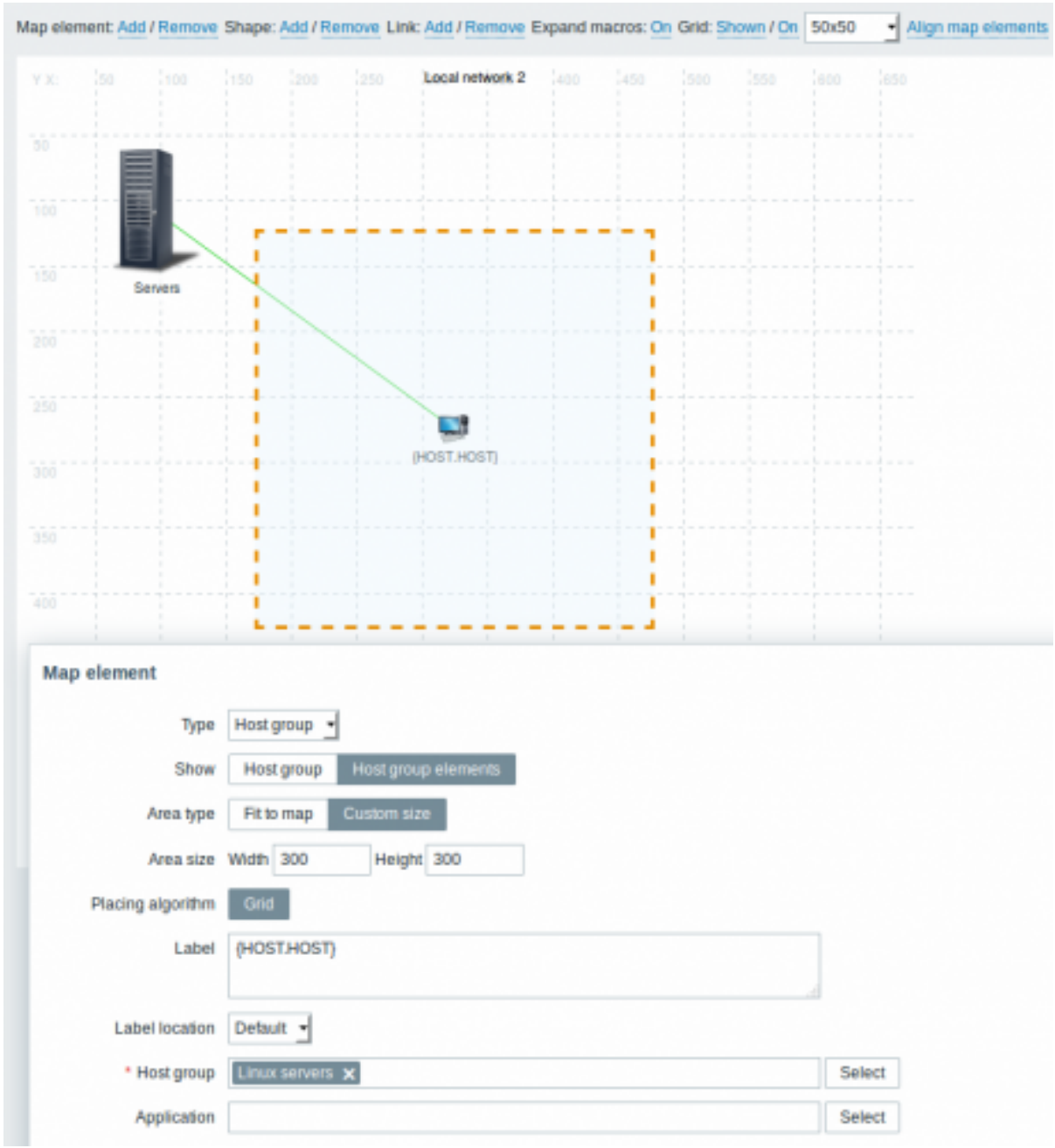


2 Host group elements

Overview

This section explains how to add a “Host group” type element when configuring a network map.

Configuration



All mandatory input fields are marked with a red asterisk.

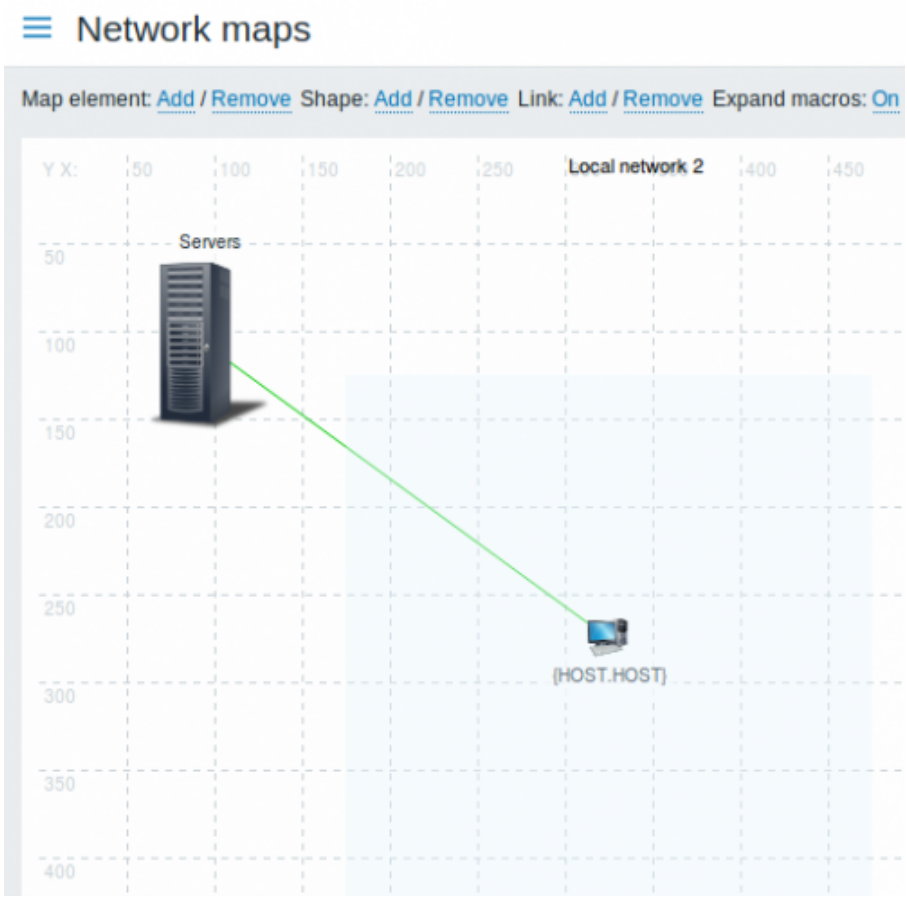
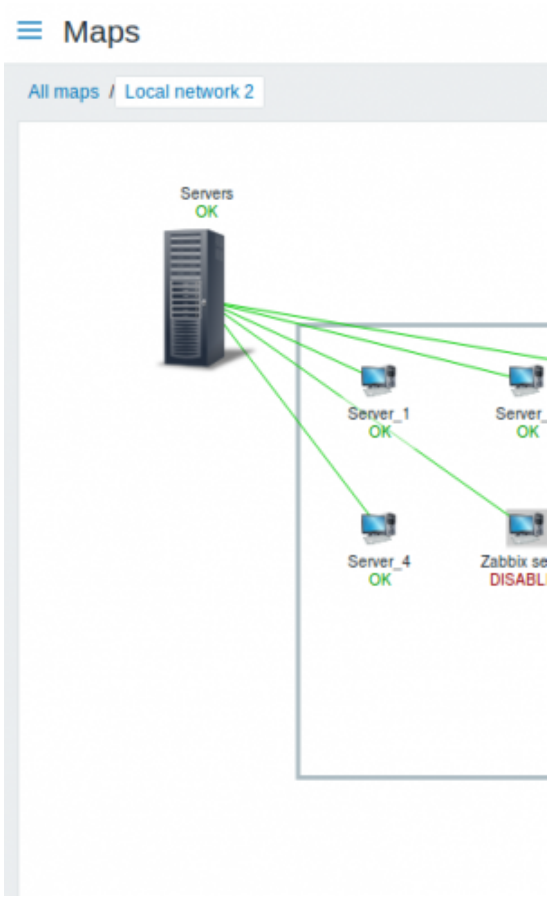
This table consists of parameters typical for *Host group* element type:

Parameter	Description
Type	Select Type of the element: **Host group** - icon representing status of all triggers of all hosts belonging to the selected group
Show	Show options: **Host group** - selecting this option will result as one single icon displaying corresponding information about the certain host group **Host group elements** - selecting this option will result as multiple icons displaying corresponding information about each single element (host) of the certain host group
Area type	This setting is available if “Host group elements” parameter is selected: **Fit to map** - all host group elements are equally placed within the map **Custom size** - manual setting of the map area for all the host group elements to be displayed

Parameter	Description
Area size	This setting is available if “Host group elements” parameter and “Area type” parameter are selected: **Width** - numeric value to be entered to specify map area width **Height** - numeric value to be entered to specify map area height
Placing algorithm	**Grid** – only available option of displaying all the host group elements
Label	Icon label, any string. Macros and multi-line strings can be used in labels. If the type of the map element is “Host group” specifying certain macros has impact on the map view displaying corresponding information about each single host. For example, if {HOST.IP} macro is used, edit map view will only display the macro {HOST.IP} itself while map view will include and display each host’s unique IP address

Viewing host group elements

This option is available if ”Host group elements” show option is chosen. When selecting ”Host group elements” as the *show* option, you will at first see only one icon for the host group. However, when you save the map and then go to the map view, you will see that the map includes all the elements (hosts) of the certain host group:

Map editing view	Map view
	

Notice how the {HOST.NAME} macro is used. In map editing the macro name in unresolved, while in map view all the unique names of the hosts are displayed.

3 Link indicators

Overview

You can assign some triggers to a link between elements in a network map. When these triggers go into a problem state, the link can reflect that.

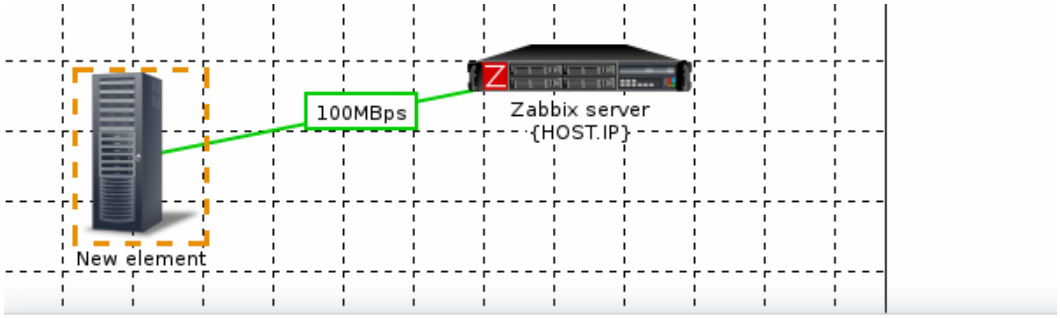
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

Should any of these triggers go into a problem state, their link style and color will be displayed on the link. So maybe your default link was a green line. Now, with the trigger in problem state, your link may become bold red (if you have defined it so).

Configuration

To assign triggers as link indicators, do the following:

- select a map element
- click on *Edit* in the *Links* section for the appropriate link
- click on *Add* in the *Link indicators* block and select one or more triggers



Map element

Type: Host

Label: New element

Label location: Default

*Host: New host × Select

Application: Select

Automatic icon selection

Icons

Default	Server_(96)
Problem	Server_(128)
Maintenance	Server_(24)
Disabled	Default

Coordinates: X 89 Y 127

URLs

NAME	URL

Add

Apply Remove Close

Links

ELEMENT NAME	LINK INDICATORS
Zabbix server	New host: Zabbix agent on New host is unreachable for 5 minutes

Label: 100MBps

Connect to: Zabbix server

Type (OK): Bold line

Color (OK): 00CC00

Link indicators

TRIGGER	TYPE	COLOU
New host: Zabbix agent on New host is unreachable for 5 minutes	Line	DI

Add

Apply Remove Close

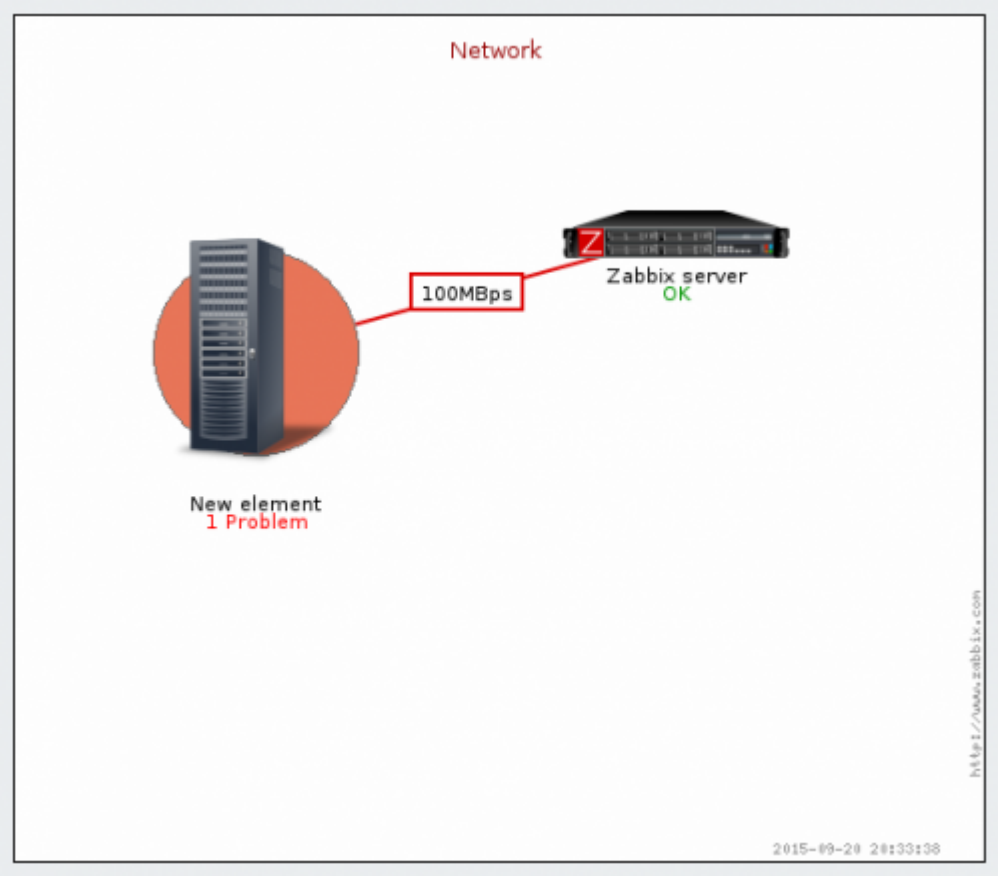
All mandatory input fields are marked with a red asterisk.

Added triggers can be seen in the *Link indicators* list.

You can set the link type and color for each trigger directly from the list. When done, click on *Apply*, close the form and click on *Update* to save the map changes.

Display

In *Monitoring* → *Maps* the respective color will be displayed on the link if the trigger goes into a problem state.



Note:
If multiple triggers go into a problem state, the problem with the highest severity will determine the link style and color. If multiple triggers with the same severity are assigned to the same map link, the one with the lowest ID takes precedence. Note also that:

1. *Minimimum trigger severity* and *Show suppressed problem* settings from map configuration affect which problems are taken into account.
2. In case with triggers with multiple problems (multiple problem generation), each problem may have severity that differs from trigger severity (changed manually), may have different tags (due to macros) and may be suppressed.

3 Screens

Overview

On Zabbix screens you can group information from various sources for a quick overview on a single screen. Building the screens is quite easy and intuitive.

Essentially a screen is a table. You choose how many cells per table and what elements to display in the cells. The following elements can be displayed:

- simple graphs
- simple graph prototypes
- user-defined custom graphs
- custom graph prototypes
- maps
- plain text information

- server information (overview)
- host information (overview)
- trigger information (overview)
- host/host group issues (status of problems)
- problems by severity
- data overview
- clock
- history of events
- history of recent actions
- URL (data taken from another location)

Global screens are managed in *Monitoring* → *Screens*, where they can be configured, managed and viewed. They can also be added to the favorites section of *Monitoring* → *Dashboard*.

Host-level screens are configured on template level and then generated for hosts once the template is linked to the hosts.

To configure a screen you must first create it by defining its general properties and then add individual elements in the cells.

All users in Zabbix (including non-admin users) can create screens. Screens have an owner - the user who created them.

Screens can be made public or private. Public screens are visible to all users.

Private screens are visible only to their owner. Private screens can be shared by the owner to other users and user groups. Regular (non-Super admin) users can only share with the groups and users they are member of. Private screens will be visible to their owner and the users the screen is shared with as long as they have read permissions to all screen elements. Admin level users, as long as they have read permissions to all screen elements, can see and edit private screens regardless of being the owner or belonging to the shared user list.

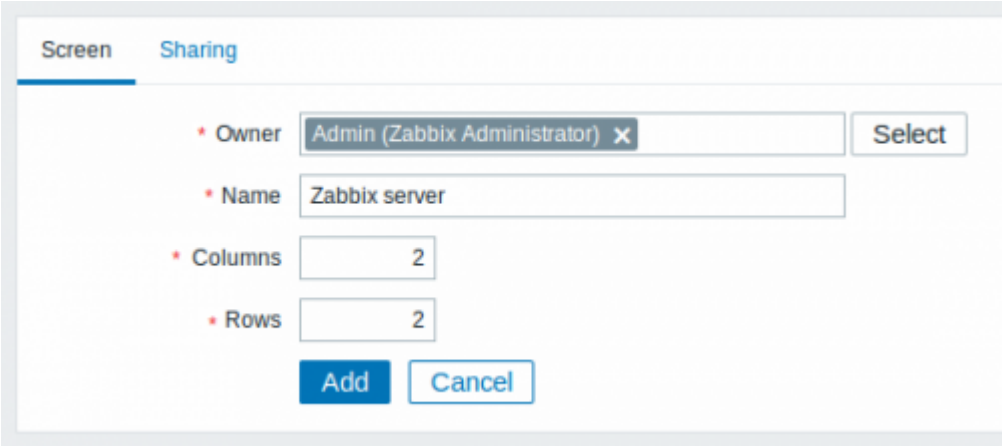
Warning:
For both public and private screens a user must have at least read permissions to all screen elements in order to see the screen. To add an element to a screen a user must also have at least read permission to it.

Creating a screen

To create a screen, do the following:

- Go to *Monitoring* → *Screens*
- Go to the view with all screens
- Click on *Create screen*

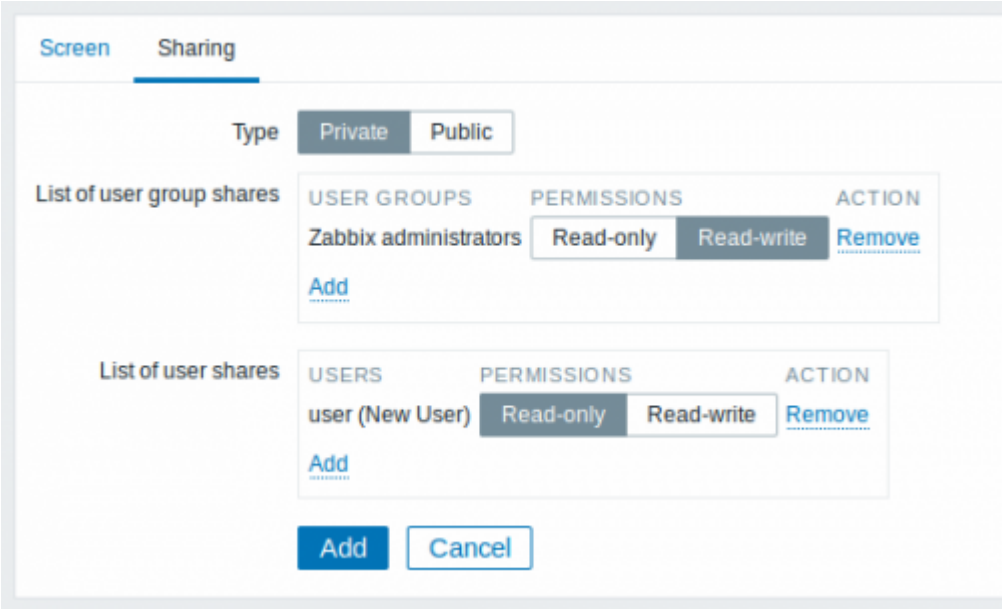
The **Screen** tab contains general screen attributes:



All mandatory input fields are marked with a red asterisk.

Give your screen a unique name and set the number of columns (vertical cells) and rows (horizontal cells).

The **Sharing** tab contains the screen type as well as sharing options (user groups, users) for private screens:



Parameter	Description
Owner	Select the screen owner.
Type	Select screen type: **Private** - screen is visible only to selected user groups and users **Public** - screen is visible to all
List of user group shares	Select user groups that the screen is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the screen is accessible to. You may allow read-only or read-write access.

Click on *Add* to save the screen.

Adding elements

To add elements to the screen, click on *Constructor* next to the screen name in the list.

On a new screen you probably only see links named *Change*. Clicking those links opens a form whereby you set what to display in each cell.

On an existing screen you click on the existing elements to open the form whereby you set what to display.

All mandatory input fields are marked with a red asterisk.

Screen element attributes:

Parameter	Description
Resource	Information displayed in the cell: **Action log** - history of recent actions **Clock** - digital or analog clock displaying current server or local time **Data overview** - latest data for a group of hosts **Graph** - single custom graph **Graph prototype** - custom graph from low-level discovery rule **History of events** - latest events **Host group issues** - status of triggers filtered by the host group (includes triggers without events, since Zabbix 2.2) **Host info** - high level host related information **Host issues** - status of triggers filtered by the host (includes triggers without events, since Zabbix 2.2) **Map** - single map **Plain text** - plain text data **Simple graph** - single simple graph **Simple graph prototype** - simple graph based on item generated by low-level discovery **System information** - high-level information about Zabbix server **Problems by severity** - displays problems by severity (similar to the Dashboard) **Trigger info** - high level trigger related information **Trigger overview** - status of triggers for a host group **URL** - include content from the specified resource See also more information on configuring each resource.
Horizontal align	Possible values: **Center** **Left** **Right**
Vertical align	Possible values: **Middle** **Top** **Bottom**
Column span	Extend cell to a number of columns, same way as HTML column spanning works.
Row span	Extend cell to a number of rows, same way as HTML row spanning works.

Take note of the '+' and '-' controls on each side of the table.

Clicking on '+' above the table will add a column. Clicking on '-' beneath the table will remove a column.

Clicking on '+' on the left side of the table will add a row. Clicking on '-' on the right side of the table will remove a row.

Attention:
If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

Dynamic elements

For some of the elements there is an extra option called *Dynamic item*. Checking this box at first does not to seem to change anything.

However, once you go to *Monitoring* → *Screens*, you may realize that now you have an extra multiselect field there for selecting the host. Thus you have a screen where some elements display the same information while others display information depending on the currently selected host.

The benefit of this is that you do not need to create extra screens just because you want to see the same graphs containing data from various hosts.

Dynamic item option is available for several screen elements:

- Graphs (custom graphs)
- Graph prototypes
- Simple graphs

- Simple graph prototypes
- Plain text
- URL

Note:
Clicking on a dynamic graph opens it in full view; although with custom graphs and graph prototypes that is currently supported with the default host only (i.e. with host 'not selected' in the dropdown). When selecting another host in the dropdown, the dynamic graph is created using item data of that host and the resulting graph is not clickable.

Note:
Dynamic URL elements will not be displayed in *Monitoring* → *Screens*, unless a host is selected. Without a selected host the "No host selected" message will be visible only.

1 Screen elements

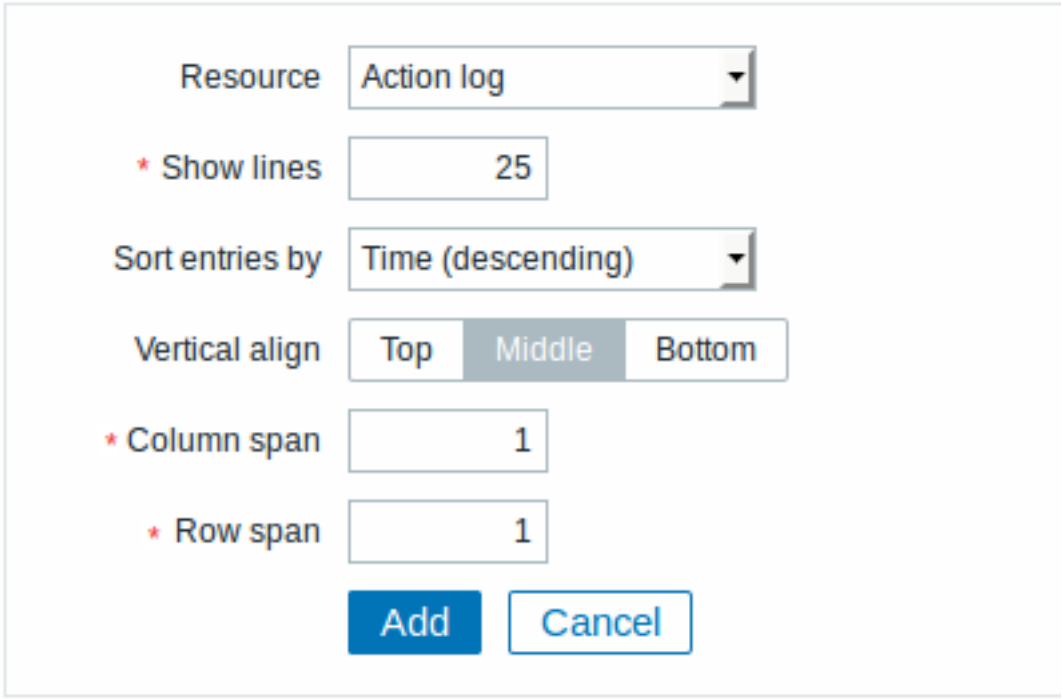
Overview

This section lists available screen elements and provides details for screen element configuration.

1 Action log

In the action log element you can display details of action operations (notifications, remote commands). It replicates information from *Reports* → *Audit*.

To configure, select *Action log* as resource:



All mandatory input fields are marked with a red asterisk.

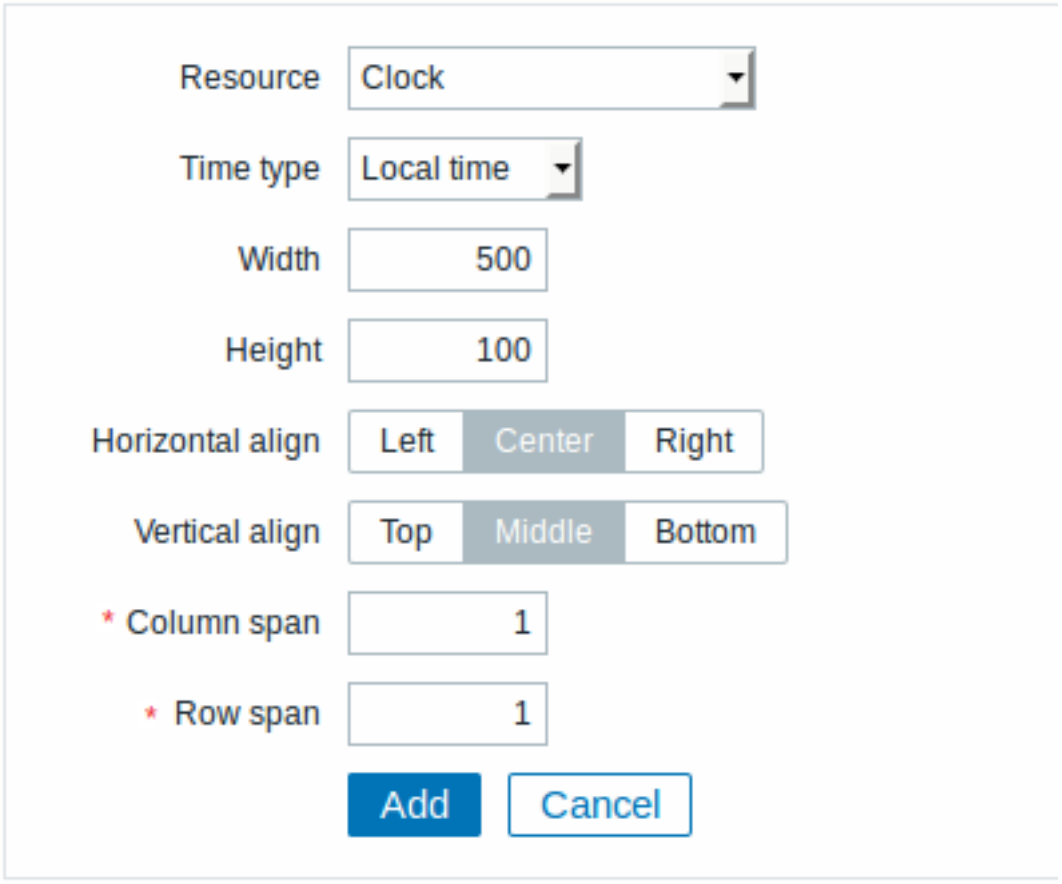
You may set the following specific options:

Show lines	Set how many action log lines will be displayed in the screen cell.
Sort entries by	Sort entries by: **Time** (descending or ascending) **Type** (descending or ascending) **Status** (descending or ascending) **Recipient** (descending or ascending).

2 Clock

In the clock element you may display local, server or specified host time.

To configure, select *Clock* as resource:



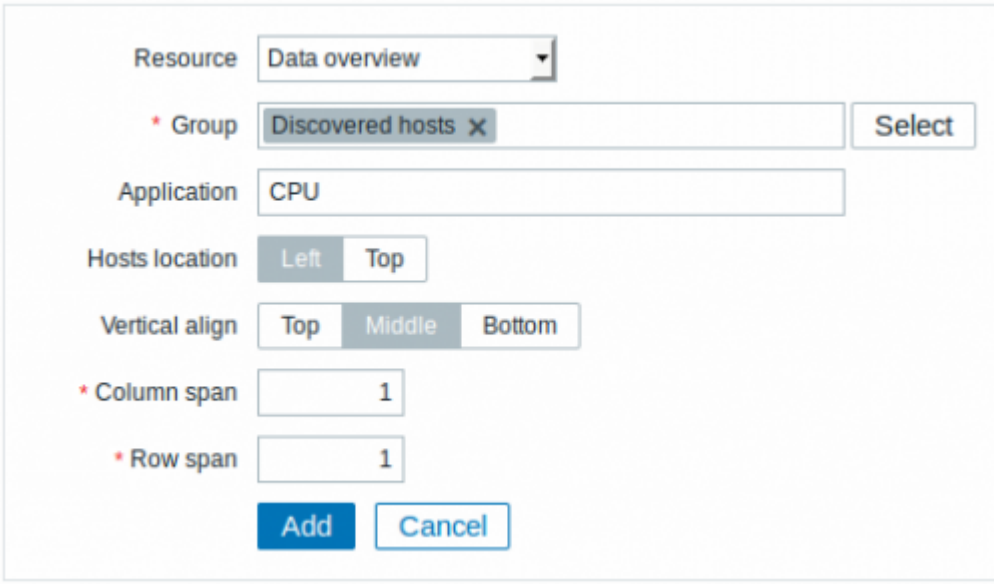
You may set the following specific options:

Time type	Select local, server or specified host time.
Item	Select the item for displaying time. To display host time, use the `system.localtime[local]` item. This item must exist on the host. This field is available only when *Host time* is selected.
Width	Select clock width.
Height	Select clock height.

3 Data overview

In the data overview element you can display the latest data for a group of hosts. It replicates information from *Monitoring* → *Overview* (when *Data* is selected as Type there).

To configure, select *Data overview* as resource:



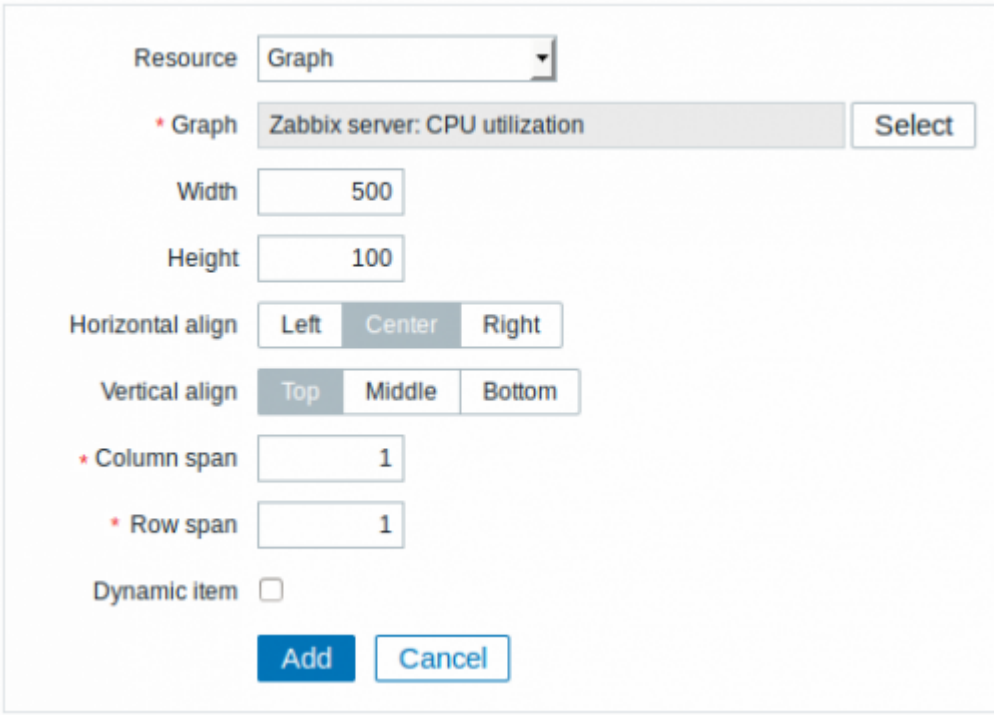
You may set the following specific options:

Group	Select host group.
Application	Enter application name.
Hosts location	Select host location - left or top.

4 Graph

In the graph element you can display a single custom graph.

To configure, select *Graph* as resource:



You may set the following specific options:

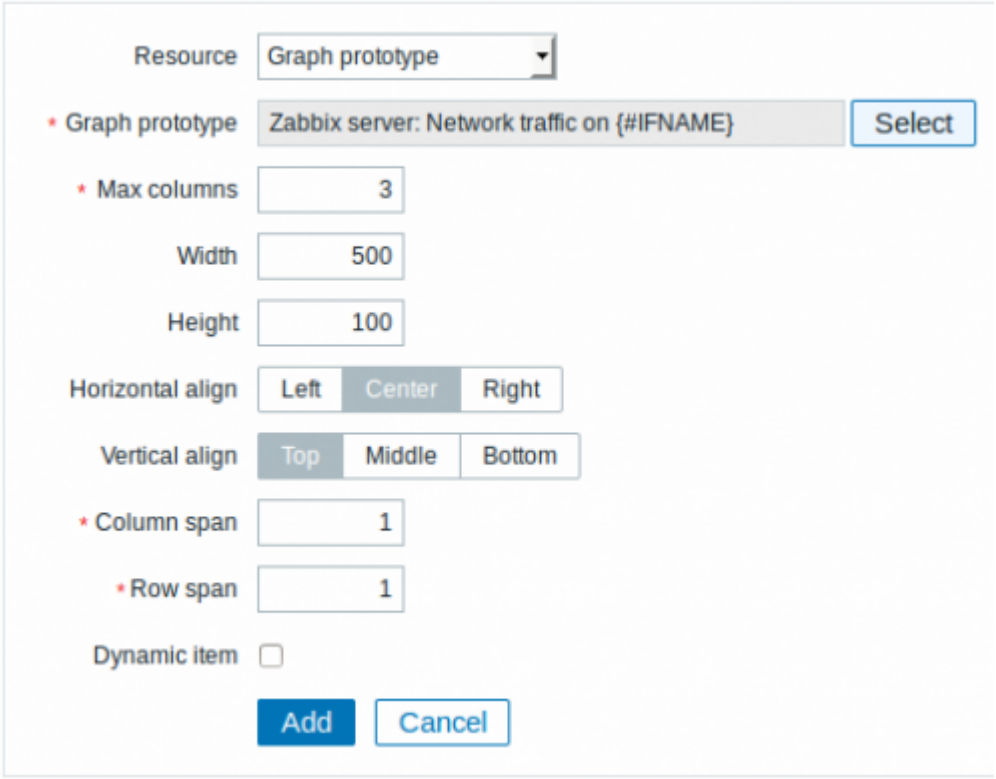
Graph	Select the graph to display.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.

Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

5 Graph prototype

In the graph prototype element you can display a custom graph from a low-level discovery rule.

To configure, select *Graph prototype* as resource:



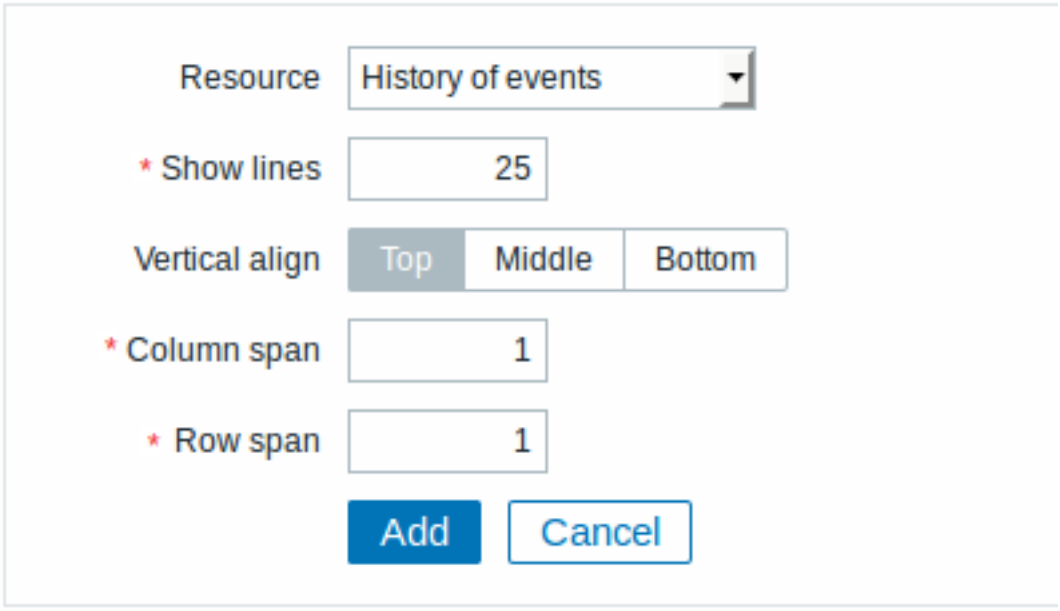
You may set the following specific options:

Graph prototype	Select the graph prototype to display.
Max columns	In how many columns generated graphs should be displayed in the screen cell. Useful when there are many LLD-generated graphs.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.
Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

6 History of events

In the history of events element you can display latest events.

To configure, select *History of events* as resource:



You may set the following specific option:

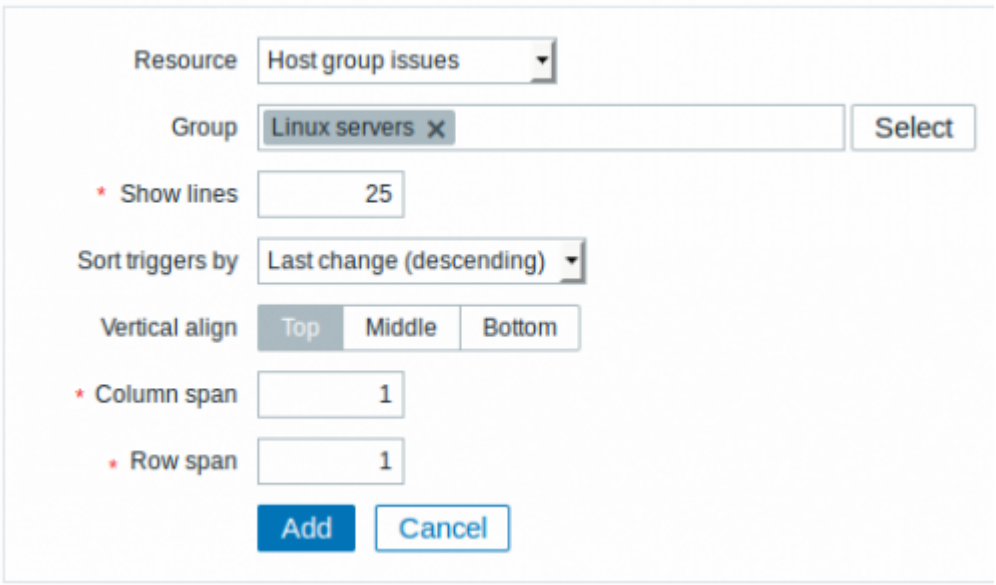
Show lines	Set how many event lines will be displayed in the screen cell.

7 Host group issues

In the host group issue element you can display problem details filtered by the selected host group.

The problem severity color displayed is originally from the underlying trigger, but can be adjusted in the problem update screen.

To configure, select *Host group issues* as resource:



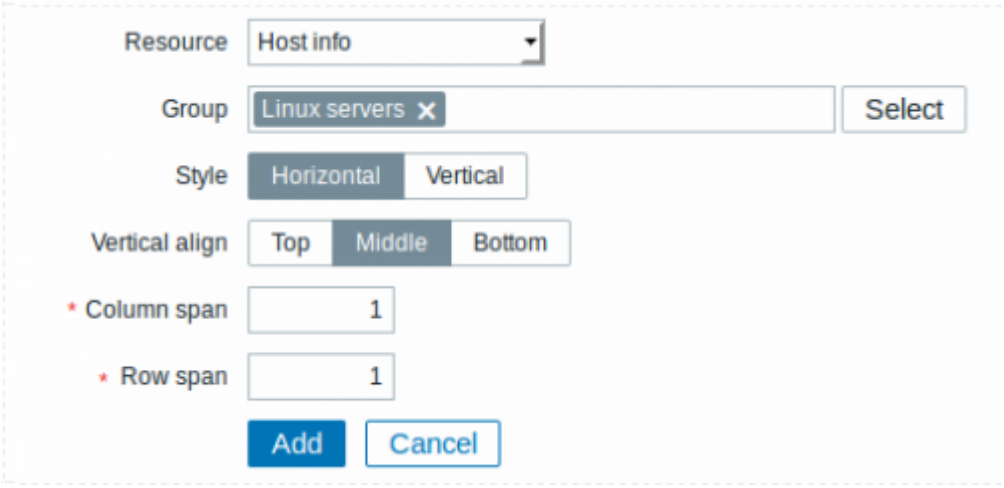
You may set the following specific options:

Group	Select host group.
Show lines	Set how many problem lines will be displayed in the screen cell.
Sort triggers by	Select from the dropdown to sort problems by last change, severity (both descending) or host (ascending).

8 Host info

In the host information element you can display high-level information about host availability.

To configure, select *Host info* as resource:



You may set the following specific options:

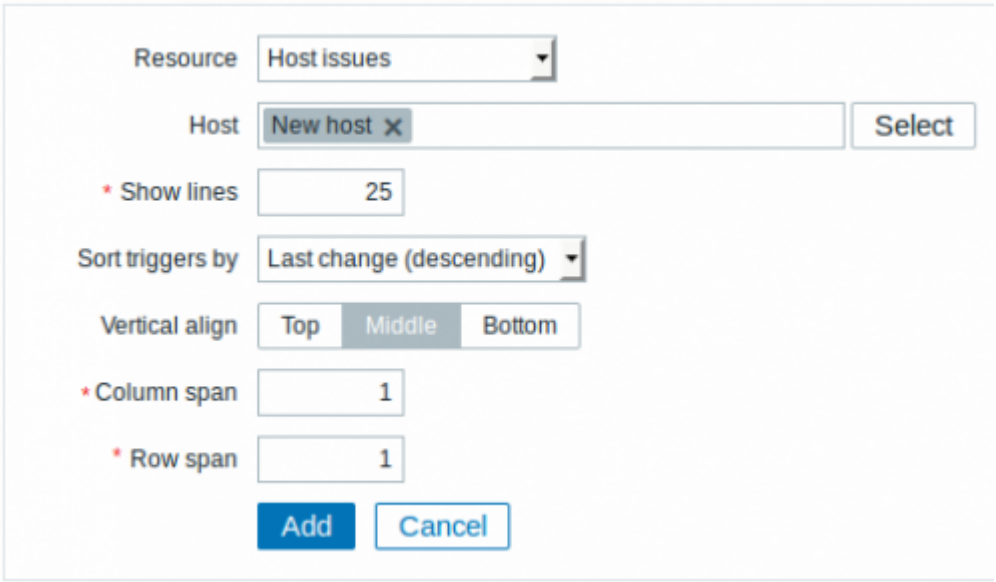
Group	Select host group(s).
Style	Select vertical or horizontal display.

9 Host issues

In the host issue element you can display problem details filtered by the selected host.

The problem severity color displayed is originally from the underlying trigger, but can be adjusted in the problem update screen.

To configure, select *Host issues* as resource:



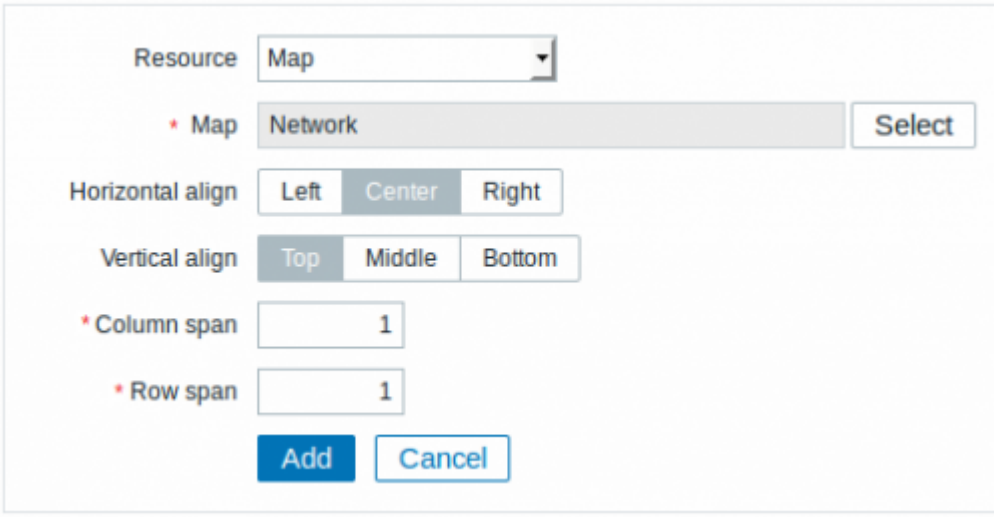
You may set the following specific options:

Host	Select the host.
Show lines	Set how many problem lines will be displayed in the screen cell.
Sort triggers by	Select from the dropdown to sort problems by last change, severity (both descending) or host (ascending).

10 Map

In the map element you can display a configured network map.

To configure, select *Map* as resource:



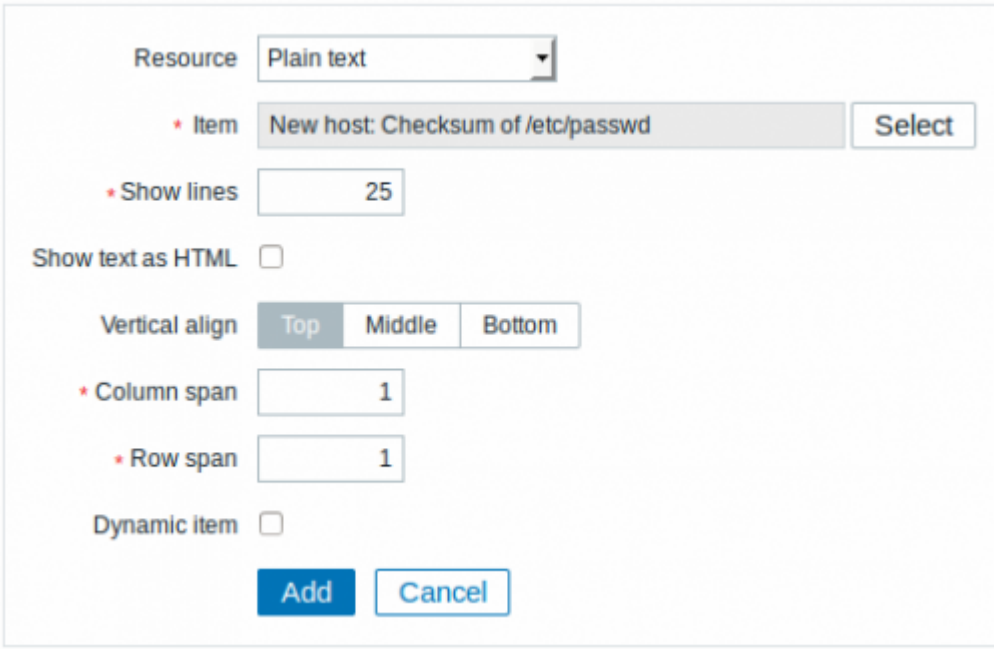
You may set the following specific options:

Map	Select the map to display.

11 Plain text

In the plain text element you can display latest item data in plain text.

To configure, select *Plain text* as resource:



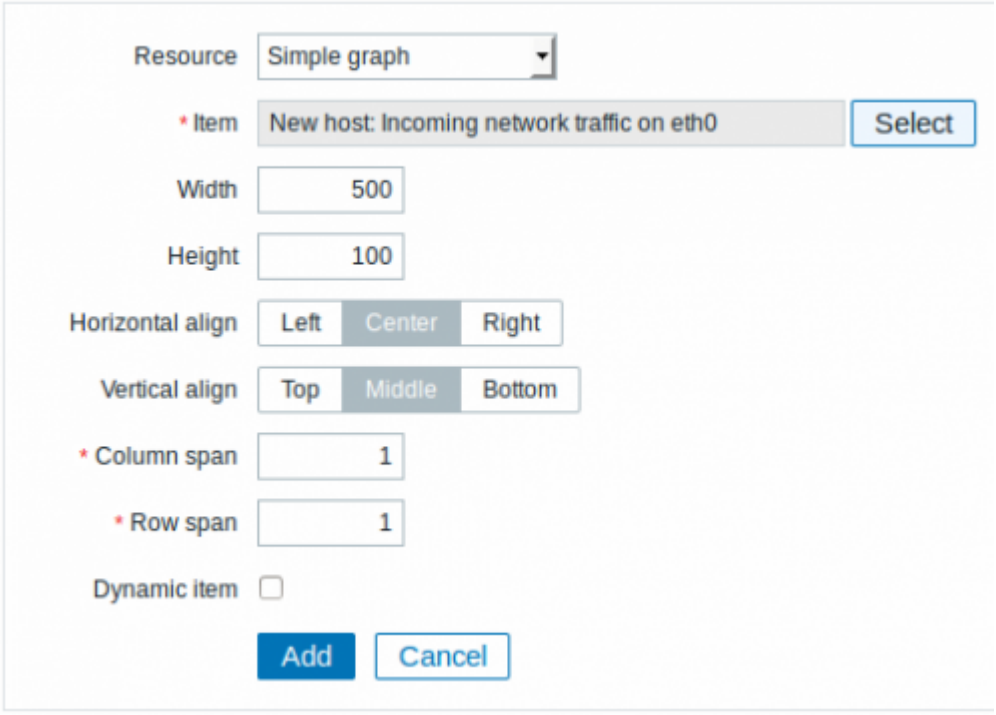
You may set the following specific options:

Item	Select the item.
Show lines	Set how many latest data lines will be displayed in the screen cell.
Show text as HTML	Set to display text as HTML.
Dynamic item	Set to display different data depending on the selected host.

12 Simple graph

In the simple graph element you can display a single simple graph.

To configure, select *Simple graph* as resource:



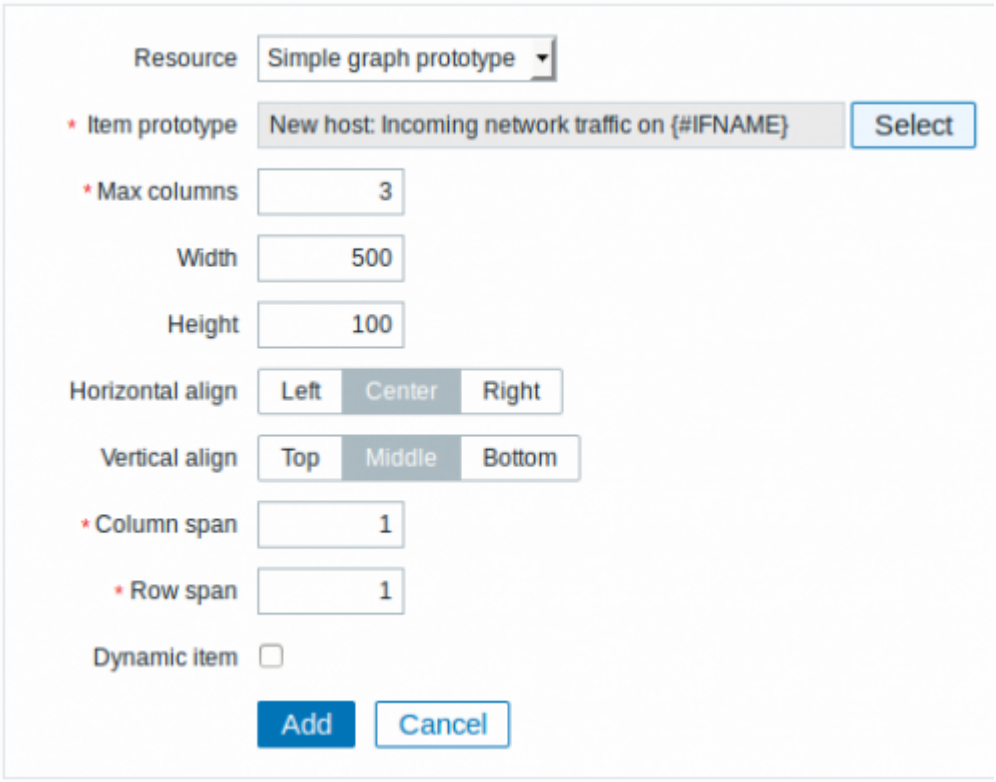
You may set the following specific options:

Item	Select the item for the simple graph.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.
Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

13 Simple graph prototype

In the simple graph prototype element you can display a simple graph based on an item generated by low-level discovery.

To configure, select *Simple graph prototype* as resource:



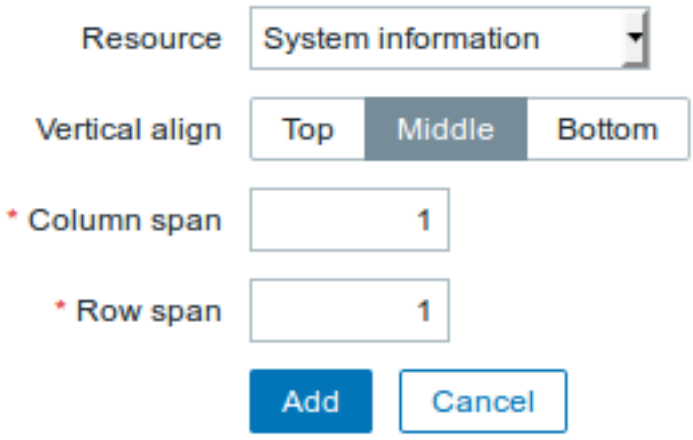
You may set the following specific options:

Item prototype	Select the item prototype for the simple graph.
Max columns	In how many columns generated graphs should be displayed in the screen cell. Useful when there are many LLD-generated graphs.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.
Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

14 System information

In the system information element you can display high-level Zabbix and Zabbix server information.

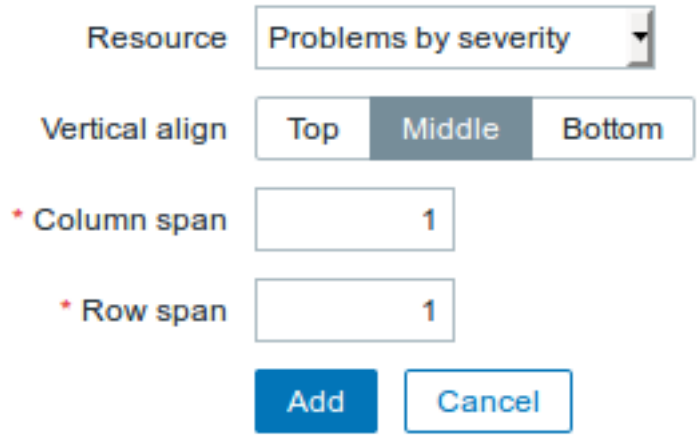
To configure, select *System information* as resource:



15 Problems by severity

In this element you can display problems by severity similarly as in the Dashboard widget.

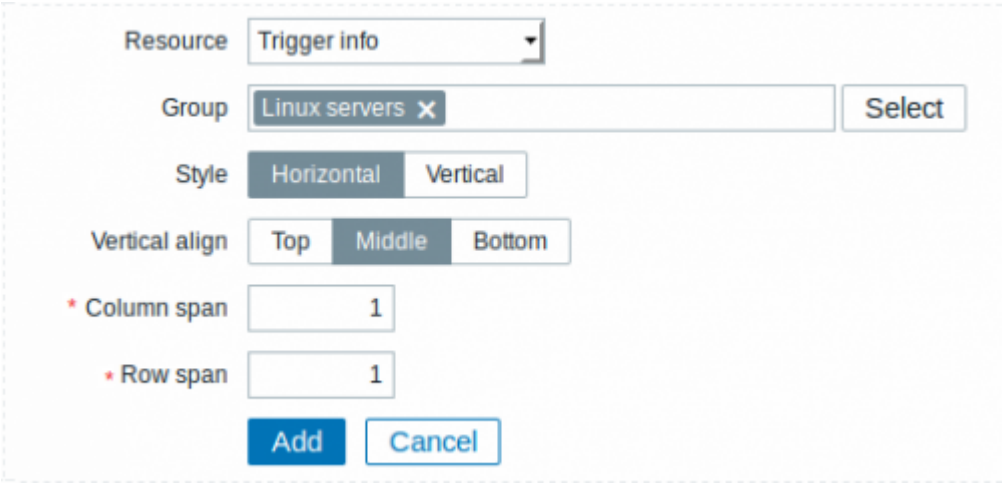
To configure, select *Problems by severity* as resource:



16 Trigger info

In the trigger info element you can display high-level information about trigger states.

To configure, select *Trigger info* as resource:



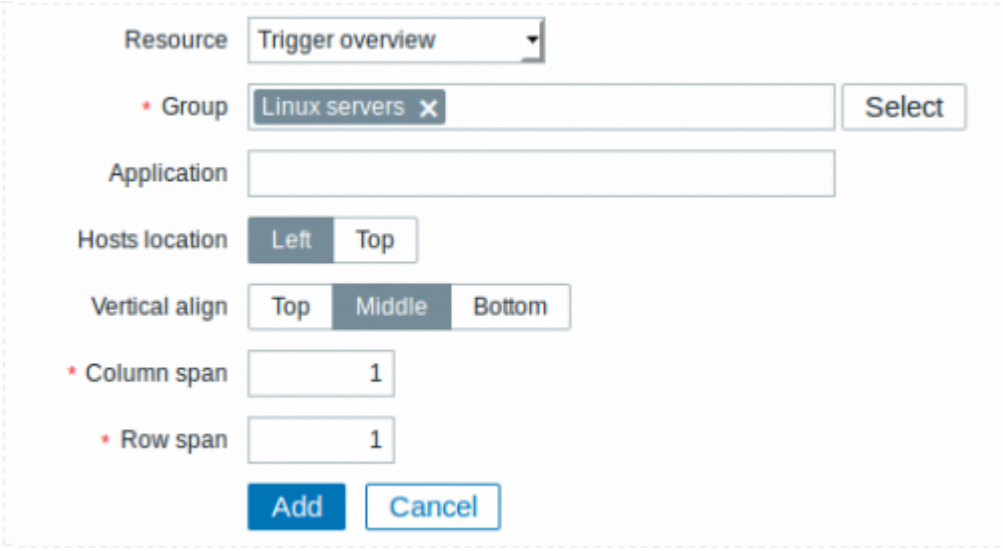
You may set the following specific options:

Group	Select the host group(s).
Style	Select vertical or horizontal display.

17 Trigger overview

In the trigger overview element you can display the trigger states for a group of hosts. It replicates information from *Monitoring* → *Overview* (when *Triggers* is selected as Type there).

To configure, select *Trigger overview* as resource:



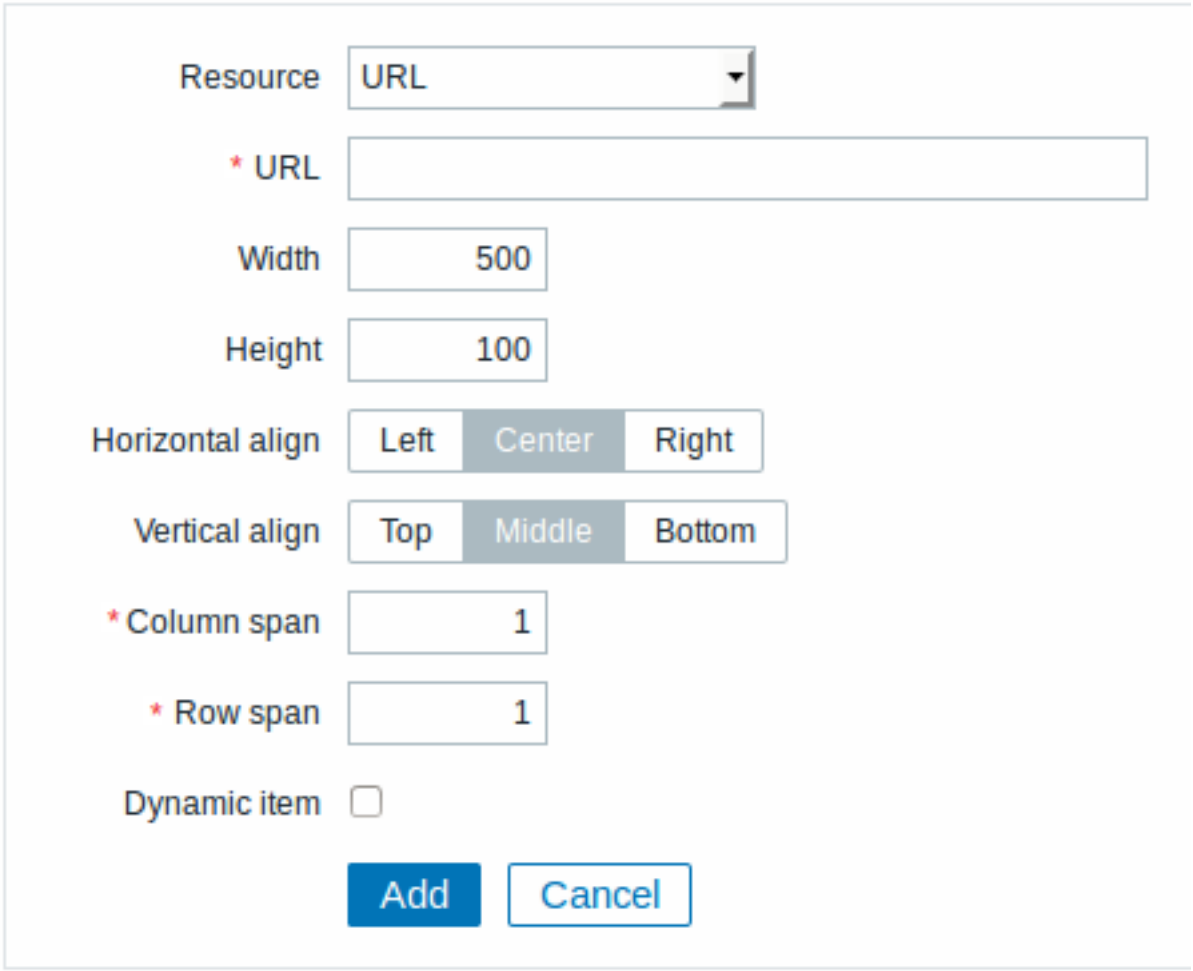
You may set the following specific options:

Group	Select the host group(s).
Application	Enter the application name.
Hosts location	Select host location - left or top.

18 URL

The URL element displays the content retrieved from the specified URL.

To configure, select *URL* as resource:



You may set the following specific options:

URL	Enter the URL to display. Relative paths are allowed since Zabbix 4.4.8.
Width	Select window width.
Height	Select window width.
Dynamic item	Set to display different URL content depending on the selected host.

Attention:
Browsers might not load an HTTP page included in a screen (using URL element), if Zabbix frontend is accessed over HTTPS.

4 Slide shows

Overview

In a slide show you can configure that a number of screens are displayed one after another at set intervals.

Sometimes you might want to switch between some configured screens. While that can be done manually, doing that more than once or twice may become very tedious. This is where the slide show function comes to rescue.

All users in Zabbix (including non-admin users) can create slide shows. Slide shows have an owner - the user who created them.

Slide shows can be made public or private. Public slide shows are visible to all users, however, they must have at least read permissions to all slide show elements (screens) to see it. To add a screen to the slide show the user must also have at least read permission to it.

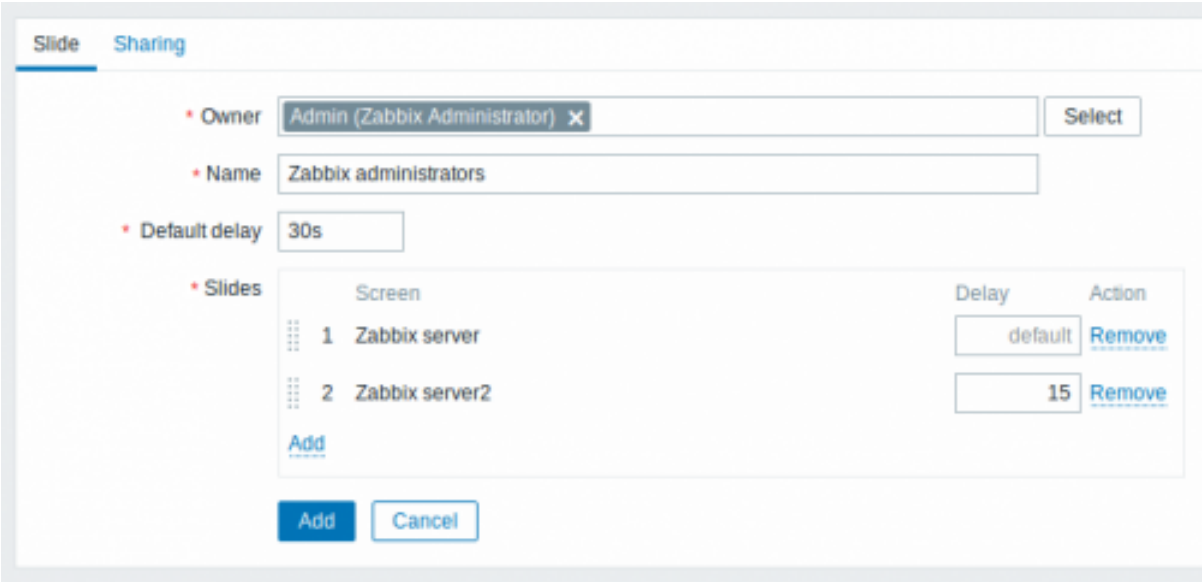
Private slide shows are visible only to their owner. Private slide shows can be shared by the owner to other users and user groups. Regular (non-Super admin) users can only share with the groups and users they are member of. Private slide shows will be visible to their owner and the users the slide show is shared with as long as they have read permissions to all included screens. Admin level users, as long as they have read permissions to all included screens, can see and edit private slide shows regardless of being the owner or belonging to the shared user list.

Configuration

To create a slide show, do the following:

- Go to *Monitoring* → *Screens*
- Select *Slide shows* in the title dropdown
- Click on *Create slide show*

The **Slide** tab contains general slide show attributes:



All mandatory input fields are marked with a red asterisk.

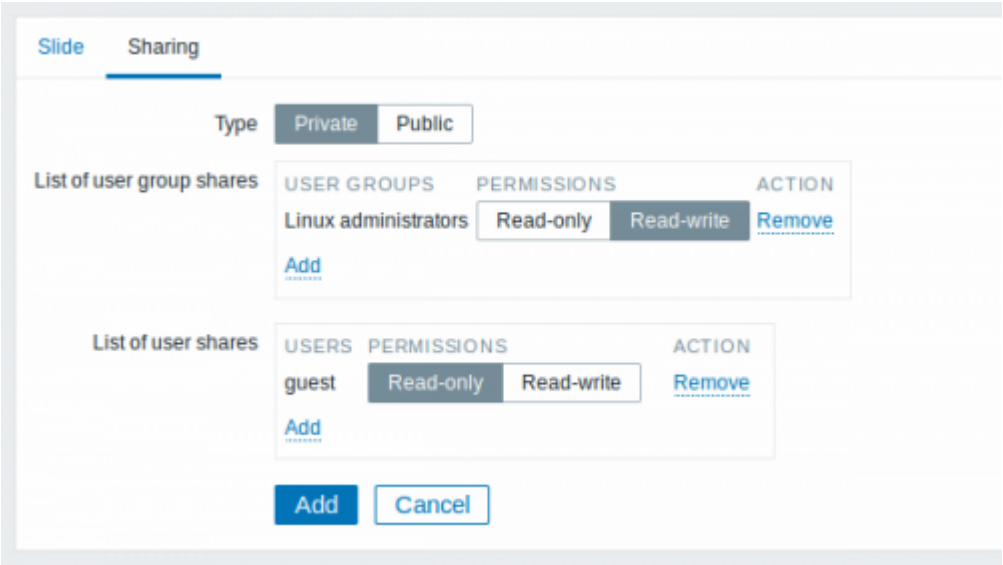
Parameter	Description
Owner	Select the slide show owner. Specifying owner is mandatory.

Parameter	Description
Name	Unique name of the slide show.
Default delay	How long one screen is displayed by default, before rotating to the next. Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.
Slides	List of screens to be rotated. Click on *Add* to select screens. The *Up/Down* arrow before the screen allows to drag a screen up and down in the sort order of display. If you want to display only, say, a single graph in the slide show, create a screen containing just that one graph.
Screen	Screen name.
Delay	A custom value for how long the screen will be displayed, in seconds. If set to 0, the *Default delay* value will be used.
Action	Click on *Remove* to remove a screen from the slide show.

The slide show in this example consists of two screens which will be displayed in the following order:

Zabbix server ⇒ Displayed for 30 seconds ⇒ Zabbix server2 ⇒ Displayed for 15 seconds ⇒ Zabbix server ⇒ Displayed for 30 seconds ⇒ Zabbix server2 ⇒ ...

The **Sharing** tab contains the slide show type as well as sharing options (user groups, users) for private slide shows:



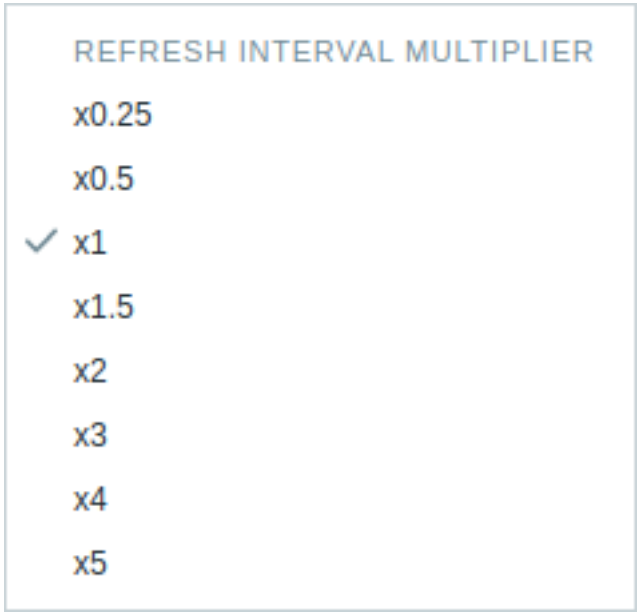
Parameter	Description
Type	Select slide show type: **Private** - slide show is visible only to selected user groups and users **Public** - slide show is visible to all
List of user group shares	Select user groups that the slide show is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the slide show is accessible to. You may allow read-only or read-write access.

Click on *Add* to save the slide show.

Display

Slide shows that are ready can be viewed in *Monitoring* → *Screens*, then choosing *Slide shows* in the title dropdown and clicking on the slide show name.

With the Menu option next to the dropdown, you can accelerate or slow down the display by choosing a slide delay multiplier:



Attention:
If a delay ends up as being less than 5 seconds (either by having entered a delay less than 5 seconds or by using the slide delay multiplier), a 5-second minimum delay will be used.

5 Host screens

Overview

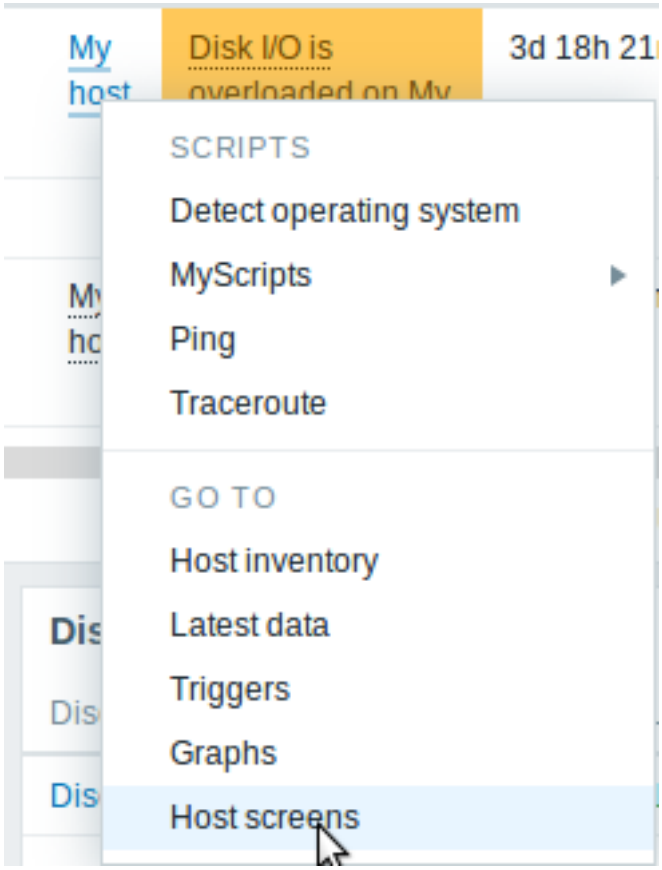
Host screens look similar to global screens, however, host screens display data about the host only. Host screens are configured on the template level and then are generated for a host, once the template is linked to the host.

Host screens *cannot* be configured or directly accessed in the *Monitoring* → *Screens* section, which is reserved for global screens. The ways to access host screens are listed below in this section.

Accessing host screens

Access to host screens is provided:

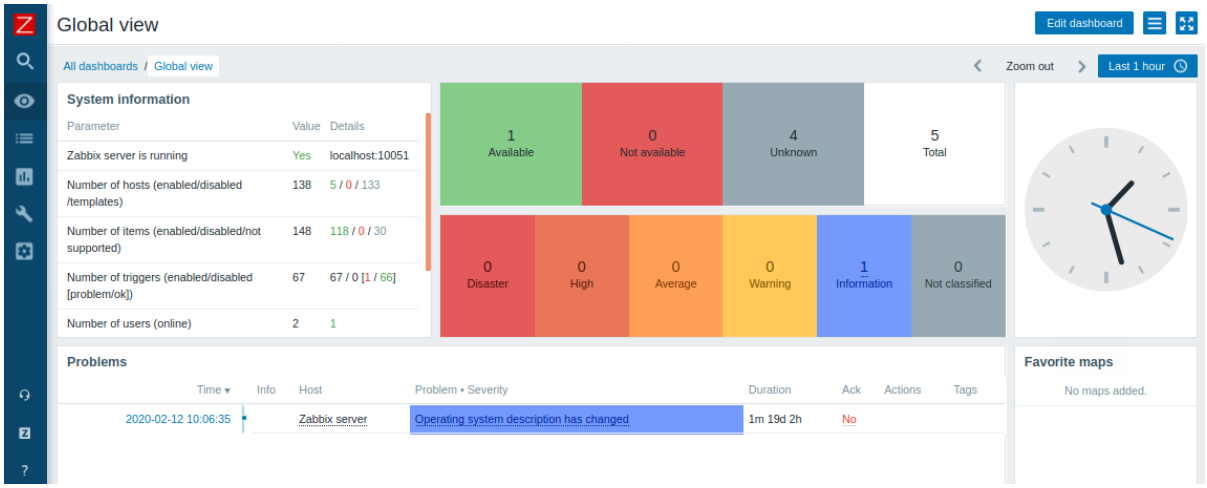
- From the host menu that is available in many frontend locations:
  - click on the host name and then select *Host screens* from the drop-down menu



- When searching for a host name in global search:
  - click on the *Screens* link provided in search results
- When clicking on a host name in *Inventory* → *Hosts*:
  - click on the *Screens* link provided

6 Dashboard

The dashboard and its widgets provide a strong visualization platform with such tools as modern graphs, maps and many more.



7 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
- 自动发现规则
- web 场景

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 开箱即用的模板

概述

Zabbix 致力于提供越来越多有用的开箱即用模板列表。开箱即用的模板已预先配置，这是加速监控作业部署的有效方法。

有以下模板可用来源：

- 在新安装的 ZABBIX 中 - 请前往配置 → 模板页;
- 如果是从旧版本升级的 ZABBIX，你可以在下载的最新版本的 ZABBIX 的 templates 目录中找到模板文件。在经过手工导入这些模板文件后，可以在配置 → 模板页找到模板。
- 也可以从Zabbix git repository下载模板文件。(请确认模板文件与您的 ZABBIX 兼容)

请使用侧边栏目录访问特定类型模板及操作须知

参考资料:

- 导入模板
- 链接模板

HTTP 模板操作

使用HTTP agent方式收集度量数据的模板正确操作步骤如下:

1. 在 Zabbix 中创建一个主机，指定监控目标的 IP 地址或 DNS 名称为主接口。这需要宏 `{HOST.CONN}` 在模板项中正确的解析。
2. 将模板链接 到步骤 1 中创建的主机 (如果模板在 Zabbix 安装中不可用，您可能需要首先导入模板的 `.xml` 文件 - 参见开箱即用的模板说明).
3. 根据需要调整必配宏的值。
4. 配置被监视的主机实例，以允许与 Zabbix 获取数据 - 请参阅 附加步骤/注释列中的说明。

> **Note:**
> 此页仅包含正确模板操作所需的最小宏的集和设置步骤。模板的详细描述，包括宏、监控项和触发器的完整列表，可以在模板的自述文件 `Readme.md` 文件（可通过单击模板名称进行访问）。

模板必	宏附加步骤	注释
Template App Apache by HTTP	**{$APACHE.STATUS.HOST}** - Apache 状态页的主机名或 IP 地址 (默认: 127.0.0.1)。Apache 模块 m**{$APACHE.STATUS.PATH}** - URL 路径 (默认: server-status?auto)。检查可 **{$APACHE.STATUS.PORT}** - Apache 状态页的端口号 (默认：80)。httpd -M**{$APACHE.STATUS.SCHEME}** - 请求协议. 支持: http (默认), https. |d_status 需要被设置 (参见 Apache [文档](https://httpd.apache.org/docs/current/mod/mod_status.html)). 性, 执行: 2>/dev/null | grep status_module Apache 配置样例: `<Location "/server-status"> SetHandler server-status Require host example.com </Location>`	
Template App Elasticsearch Cluster by HTTP	**{$ELASTICSEARCH.PORT}** - Elasticsearch 主机的端口号 (默认: 9200)。-**{$ELASTICSEARCH.SCHEME}** - 请求协议. 支持: http (默认), https. **{$ELASTICSEARCH.USERNAME}**, **{$ELASTICSEARCH.PASSWORD}** - 登录凭据, 仅当用于 Elasticsearch 身份验证时才需要.	

模板必	宏附加步骤	注释
Template App Etcd by HTTP	**{$ETCD.PORT}** - Etcd API 端点的端口号 (默认：2379)．度量数据被从 AP**{$ETCD.SCHEME}** - 请求协议，支持：http (默认), https. **{$ETCD.USER}**, **{$ETCD.PASSWORD}** - 登录凭据, 仅当用于 Etcd 身份验证时才需要。检验 Etcd 是否被配置允许本	端点的接口路径 /metrics 收集; 指定 API 端点的接口路径，请使用 --listen-metric 参数 (参见 Etcd 文档). 采集度量数据, 执行: curl -L http://localhos 检查 Etcd 可以被从 Zab-bix proxy 或 Zab-bix server 访问 执行: curl -L

模板必	宏附加步骤	注释
Template App Hadoop by HTTP	**{$HADOOP.CAPACITY_REMAINING.MIN.WARN}** - Hadoop 集群剩余容量百分比的触发器阈值 (默认：20). 度量数据是通过使用 HTTP ag**{$HADOOP.NAMENODE.HOST}** - Hadoop NameNode 节点的主机 IP 或 FQDN (默认：NameNode). **{$HADOOP.NAMENODE.PORT}** - Hadoop NameNode 节点的 web-UI 端口号 (默认：9870). **{$HADOOP.NAMENODE.RESPONSE_TIME.MAX.WARN}** - Hadoop NameNode 的 API 页最大响应时间（以秒为单位）的触发器阈值 (默认：10s). **{$HADOOP.RESOURCEMANAGER.HOST}** - Hadoop ResourceManager 节点的 IP 地址或 FQDN (默认：ResourceManager). **{$HADOOP.RESOURCEMANAGER.PORT}** - Hadoop ResourceManager 的 web-UI 端口号 (默认：8088). **{$HADOOP.RESOURCEMANAGER.RESPONSE_TIME.MAX.WARN}** - Hadoop ResourceManager API 页最大响应时间（以秒为单位）的触发器阈值 (默认：10s).|nt 和 `JSONPath` 预处理方式远程轮询 Hadoop Api 来收集的。Zabbix 服务器（或代理）执行对 ResourceManager、NodeManager、NameNode、DataNodes API 的直接请求。由于 Zabbix 批量收集数据，所有指标都可以一次性收集。	
Template App HAProxy by HTTP	**{$HAPROXY.STATS.PATH}** - HAProxy 状态页的路径 (默认：stats). HAProxy**{$HAPROXY.STATS.PORT}** - HAProxy 状态页主机或容器的端口号 (默认：8404). **{$HAPROXY.STATS.SCHEME}** - 请求协议，支持：http (默认), https.	态页需要被设置(参见 HAProxy 博客文档或模板的 Readme.md 配置样例).

模板必	宏附加步骤	注释
Template App Nginx by HTTP	**{$NGINX.STUB_STATUS.HOST}** - Nginx stub_status 主机或容器的主机名或 IP 地址 (默认：localhost). 'ngx_http_stu**{$NGINX.STUB_STATUS.PATH}** - Nginx stub_status 页的请求路径 （默认：basic_status）.\ 检查可用性，执 **{$NGINX.STUB_STATUS.PORT}** - Nginx stub_status 主机或容器的端口号 （默认：80）.\nginx -V 2**{$NGINX.STUB_STATUS.SCHEME}** - 请求协议，支持：http (默认), https.	_status_module 需要被设置 (参见 Nginx [文档](https://nginx.or 或模板的 Readme.md 配置样例). : &1 | grep -o with-http_stub_status_m

模板必	宏附加步骤	注释
	Template App PHP-FPM by HTTP	**{$PHP_FPM.HOST** - PHP-FPM 主机或容器的主机名或 IP 地址(默认：localhost). 编辑 php-fpm **{$PHP_FPM.P** - PHP-FPM 状态页(ping 页) URL 路径(默认：ping) - PHP-FPM 主机或容器的端口号(默认：80). - PHP-FPM 进程 name(默认：php-fpm). **{$PHP_FPM.SCHE** - 请

配置文件并启用状态页: path = /status /ping $ php-fpm7 -t 3. 重载 php-fpm 服务. 4. 在 Nginx 服务配置文件的 server 配置 FPM 区域, 添加(参见模板的 *Readme.md* 文档 ping.path **{$** 中带有注释的扩展 fpm). 例) location ~

模板必	宏附加步骤	注释
Template App RabbitMQ cluster by HTTP	**{$RABBITMQ.API.CLUSTER_HOST}** - RabbitMQ 集群 API 端点的主机名或 IP 地址 (默认：127.0.0.1)。 启用 RabbitMQ 管 **{$RABBITMQ.API.SCHEME}** - 请求协议，支持：http (默认), https. **{$RABBITMQ.API.USER}**, **{$RABBITMQ.API.PASSWORD}** - RabbitMQ 登录凭据 (默认用户名: zbx_monitor, 密码: zabbix)。 在 RabbitMQ	插件 (参见 RabbitMQ 文档)。 中创建一个具有必备权限用于监控的用户，执行: ” rabbitmqctl add_user zbx_monitor <PASSWORD> ” rabbitmqctl set_permissions -p / zbx_monitor %% "" "" ".*"%% rabbitmqctl set_user_tags zbx_monitor monitoring 单节点安装时，可将集群模板分

模板必	宏附加步骤	注释
Template DB ClickHouse by HTTP	**{$CLICKHOUSE.PORT}** - ClickHouse HTTP 端点的端口号 (默认：8123)。创建一个具有 w**{$CLICKHOUSE.SCHEME}** - 请求协议，支持：http (默认), https.\ **{$CLICKHOUSE.USER}**, **{$CLICKHOUSE.PASSWORD}** - ClickHouse 登录凭据 (默认用户名: zabbix, 密码: zabbix_pass).\ 参见模板的 *Rea 如果不需要身份验证，请从监控项的 HTTP agent 类型配置项的请求头中删除。|b 配置文件和查看数据库权限的 ClickHouse 用户 (参见 ClickHouse 文档). me.md* 文档中现成的一个 `zabbix.xml` 文件配置。	
Template Tel Asterisk by HTTP	**{$AMI.PORT}** - 检测服务可用性的 AMI 端口号 (默认：8088)。1. 启用 [mini-**{$AMI.SECRET}** - Asterisk Manager 的 `secret` (默认：zabbix). **{$AMI.URL}** - Asterisk Manager 的 API URL 格式 2.`<scheme>://<host>:<port>/<prefix>/rawman` (默认： http://asterisk:8088/asterisk/rawman)。3.**{$AMI.USERNAME}** - Asterisk Manager 用户名.	TTP Server](https://wiki HTTP+Server). 添加选项 `webenabled=yes` 到配置文件 *manager.conf* 的 `general` 部分. 在 Asterisk 实例中创建 *Asterisk Manager* 用户。

模板必	宏附加步骤	注释
ZooKeeper by HTTP	**{$ZOOKEEPER.COMMAND_URL}** - admin.commandURL; 用于相对于根 URL 列出和发出命令的 URL (默认：commands). 通过对 AdminServer 的请 **{$ZOOKEEPER.PORT}** - admin.serverPort; 内置 Jetty 服务的侦听端口号 (默认：8080). **{$ZOOKEEPER.SCHEME}** - 请求协议，支持：http (默认), https.	从每个 ZooKeeper 节点收集度量数据 (默认启用). 配置启用 Admin-Server 参见 ZooKeeper 文档

IPMI 摸板操作

IPMI 模板不需要任何特定的设置. 要开始监视，请将模板链接 到目标主机。(如果 Zabbix 安装中没有该模板，则可能需要先导入该模板的.xml 文件 - 有关说明，请参见开箱即用的模板 部分)。

Note:
页面仅包含最小的一组宏和正确的模板操作所需的设置步骤。在模板的 Readme.md 文件中提供了模板的详细说明，包括宏，项和触发器的完整列表（可通过单击模板名称访问）。

摸板强	宏附加步骤	注释
Template Server Chassis by IPMI	**{$IPMI.USER}**, **{$IPMI.PASSWORD}** - 访问 BMC 的凭据 (default: none) -	

JMX 摸板操作

确保通过JMX收集指标的模板正确运行步骤:

1. 确保已正确安装和设置 Zabbix Java gateway 网关。
2. 链接 模板到目标主机. 主机应设置 JMX 接口。
如果模板在您的 Zabbix 中不可用，您可能需要先导入摸板文件.xml - 查看Templates out-of-the-box 说明部分。
3. 根据需要调整强制宏的值。
4. 配置要监视的实例以允许与 Zabbix 共享数据- 请参阅 附加步骤/注释字段.

Note:
该页面仅包含最小的一组宏和正确的模板操作所需的设置步骤。在模板的 Readme.md 文件中提供了模板的详细说明，包括宏，项和触发器的完整列表（可通过单击模板名称访问）。

摸板强	宏附加步骤	注释
Apache ActiveMQ by JMX	**{$ACTIVEMQ.PORT}** - JMX 的端口 (default: 1099). 应该 **{$ACTIVEMQ.USERNAME}**, **{$ACTIVEMQ.PASSWORD}** - JMX 的登录凭据 (default username: admin, password: activemq).	据官方文档中的说明启用和配置对 Apache ActiveMQ 的 JMX 访问。
Template DB Apache Cassandra by JMX	**{$CASSANDRA.USER}**, **{$CASSANDRA.PASSWORD}** - Apache Cassandra 登录凭据 应该根 (default username: zabbix, password: zabbix).	官方文档中的说明启用和配置对 Apache Cassandra 的 JMX 访问。

摸板强	宏附加步骤	注释
Apache Kafka by JMX	**{$KAFKA.USER}**, **{$KAFKA.PASSWORD}** - Apache Kafka 登录凭据 应该根 (default username: zabbix, password: zabbix).	官方文档中的说明启用和配置对 Apache Kafka 的 JMX 访问。
Apache Tomcat by JMX	**{$TOMCAT.USER}**, **{$TOMCAT.PASSWORD}** - Apache Tomcat 登录凭据；如果 Tomcat 安装不需要身份验证，请留空 应该根据 [官方文档](https:/(default: not set).	tomcat.apache.c 10.0- doc/monitoring.h 中的说明启用和配置对 Apache Tom- cat 的 JMX 访问。(选择正确的版本)。

摸板强	宏附加步骤	注释
Ignite by JMX	**{$IGNITE.USER}**, **{$IGNITE.PASSWORD}** - Apache Ignite 登录凭据 (默认 username: zabbix, password: <secret>). 启用并配置	MX 访问 Apache Ignite. MX 树层次结构默认包含 ClassLoader。添加以下 Java 虚拟机选项 “-DIGNITE_MBEAN=false” 将排除具有 ClassLoader 名称的一级。 可以按需配置缓存和数据域指标 - 详情参

摸板强	宏附加步骤	注释

ODBC 摸板操作

确保通过ODBC monitoring监控收集度量的模板正确运行的步骤:

1. 确保 Zabbix 服务器或代理上安装了所需的 ODBC 驱动程序。
2. 将模板链接 到目标主机 (如果模板在您的 Zabbix 中不可用，您可能需要先导入模板文件.xml 文件 - 查看开箱即用的模板 说明部分。).
3. 根据需要调整强制宏的值。
4. 配置要监视的实例以允许与 Zabbix 共享数据-请参阅附加步骤/注释字段.

Note:
该页面仅包含最小的一组宏和正确的模板操作所需的设置步骤。在模板的 Readme.md 文件中提供了模板的详细说明，包括宏，项和触发器的完整列表（可通过单击模板名称访问）。

摸板强	宏附加步骤	注释
Template DB MSSQL by ODBC	**{$MSSQL.DSN}** - 系统数据源名称 (default: < 填写你的 DSN>) 创建一个 Micros**{$MSSQL.PORT}** - Microsoft SQL Server 的 TCP 端口 (default: 1433) **{$MSSQL.USER}, {$MSSQL.PASSWORD}** - Microsoft SQL 登录凭据 (default: not set) “服务	ft SQL 用户进行监视，并向该用户授予以下权限：查看服务器状态；查看任何定义 (查看 Microsoft SQL 文档获取详情). TCP 端口状态” 监控项使用 {HOST.CONN} 和 {$MSSQL.PORT} 宏来检查 Microsoft SQL 实

摸板强	宏附加步骤	注释
Template DB MySQL by ODBC	**{$MYSQL.DSN}** - 系统数据源名称 (default: < 填写你的 DSN>）要将所需的特权授予将 **{$MYSQL.USER}, {$MYSQL.PASSWORD}** - MySQL 登录凭证；密码可以为空 (default: not set) GRANT USA| 于监控的 MySQL 用户，run: E,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON %% *.* TO '<username>'@'%';%% 查阅MYSQL 文档 获取详情.	

摸板强	宏附加步骤	注释
Template DB Oracle by ODBC	**{$ORACLE.DSN}** - 系统数据源名称 (default: < 填写你的 DSN>) 1. 要创建一个用 **{$ORACLE.PORT}** - Oracle DB 的 TCP 端口 (default: 1521) C**{$ORACLE.USER}, {$ORACLE.PASSWORD}** - Oracle 登录凭证 (default: not set)--	监控的 Oracle 用户，, run: EATE USER zabbix_mon IDENTIFIED BY <PASSWORD>; 对授予 zabbix_mon 用户的访问权限。 GRANT CONNECT, CREATE SESSION TO zabbix_mon; GRANT SELECT ON V_$instance TO zabbix_mon; GRANT SELECT ON V_$database TO zabbix_mon; GRANT SELECT ON v_$sysmetric TO zabbix_mon; GRANT SELECT ON v$recovery_f TO zabbix_mon; GRANT SELECT

摸板强	宏附加步骤	注释

Zabbix agent 2 模板操作

下面步骤确保当前操作模板能正常收集监控对象数据通过Zabbix agent 2:

1. 请确保主机上已安装 agent 2，和安装的版本包含所需的插件. 如果你需要升级到 agent 2 请跟着此页面步骤。
2. 将模板链接 关联到主机 (如果在 Zabbix 中没有 agent 2 可用的模板，您可能需要首先导入模板的.xml 文件-参见开箱即用的模板 说明部分)。
3. 根据提示修改强制宏值修改。注意，用户宏可以被用来覆盖配置的参数。
4. 配置监控对象实例允许 zabbix 去获取数据。- 请参阅附加步骤/注释栏中的说明。

Attention:
Zabbix agent 2 模板和插件一起使用。而基本配置可以通过简单地调整用户宏来完成，更深层次的定制可以通过configuring the plugin 实现。举个例子，如果插件支持命名会话，通过使用自己的 URI 指定指定的会话，可以监视多个相同类型的实体 (如 MySQL1 和 MySQL2)，用户名和密码将保存在各自的配置文件中。

Note:
本页仅包含适当模板操作所需的最小宏集和设置步骤。模板的详细描述，包括宏、项和触发器的完整列表，可以在模板的自述文件中找到。Readmd 文件 (可通过单击模板名称访问)。

模板名称强制性	宏添加步骤/说明	
Template App Ceph by Zabbix agent 2	**{$CEPH.API.KEY}** - API key (default: `zabbix_pass`). 如果 {$CEPH.CONNSTRING} 是一个 URI，则 API key 必须有值。 如果 {$CEPH.CONNSTRING} 是一个会话名称，则 API key 为空。 1. 依据[文档](ht**{$CEPH.CONNSTRING}** - 连接字符串; 可以是一个会话名称或者 URL 地址，按照格式: <protocol(host:port)>. 对于 URL 地址只支持 HTTPS 架构。 例如: Prod, https://localhost:8003 (default) 2**{$CEPH.USER}** - 用户被使用在监控中 (default:`zabbix`). 如果 {$CEPH.CONNSTRING} 是一个 URI，则宏值必须有值。 如果 {$CEPH.CONNSTRING} 是一个会话名称，则宏值为空。	与 *Ceph* 插件一起使用; 支持会话名称命名。ps://docs.ceph 配置 Ceph RESTful 模块。确保 RESTful API endpoint 可以被连接使用。

模板名称强制性	宏添加步骤/说明	
Template App Docker	-	与 *Docker* 插件一起使用;不支持会话名称命名。 设置 Docker API 端点编辑插件的路径。Docker.Endpo 参数在 agent 2 配置文件 (默认: Plugins.Do 测试可用性,运行: zabbix_get -s docker-hos -k docker.inf

模板名称强制性	宏添加步骤/说明	
Template App Memcached	**{$MEMCACHED.CONN.URI}** - 连接字符串在 URI 格式; 端口是可选的; 密码没有被使用. 与 *Memcached* 模块一起使用；支如果没有设置, 插件默认使用的值是: tcp://localhost:11211. 例子: tcp://127.0.0.1:11211, tcp://localhost, unix:/var/run/memcached.sock. 测	会话名称命名. 可用性，运行: zabbix_get -s memcached-l -k memcached.p
Template DB MySQL by Zabbix agent 2	**{$MYSQL.DSN}** - MySQL 实例的系统数据源名称 (默认: < 填写你的 DSN>). 与 *MySQL* 插件一起使用下面格式定义可以是会话名称或者是一个 URI: %% <protocol(host:port or /path/to/socket)/>%% 对于 URI，只支持 TCP 和 Unix 模式。 将用于监控的权限授例子: MySQL1, tcp://localhost:3306, tcp://172.16.0.10, unix:/var/run/mysql.sock **{$MYSQL.USER}**, **{$MYSQL.PASSWORD}** - MySQL 凭证 (default: none). 如果 {$MYSQL.DSN} 是一个 URI，那么此 {$MYSQL.USER}，{$MYSQL.PASSWORD} 宏值项不为空。\ 如果 {$MYSQL.DSN} 是一个会话名称，那么此 {$MYSQL.USER}，{$MYSQL.PASSWORD} 宏值项为空. 有关的信息，请参阅 MySQL 文档 [|支持会话名称命名。 MySQL 用户，运行: RANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO '<username>'@'%'; 户权限](https://dev.mysql.com/doc/refman/8.0/en/grant.html) 和Unix sockets	

模板名称强制性	宏添加步骤/说明	
Template DB Oracle by Zabbix agent 2	**{$ORACLE.CONNSTRING}** - 连接字符串；下面格式定义可以是会话名称或者是一个 URI: <protocol(host:port or /path/to/socket)/> 与 *Oracle* 插件一起使用; 支持会话对于 URI，只支持 TCP 模式。 例子: Oracle1, tcp://localhost:1521 (default) 安 **{$ORACLE.SERVICE}** - Oracle 服务的名称 (default: ORA). 如果 {$ORACLE.CONNSTRING} 是一个 URI，那么此 {$ORACLE.CONNSTRING} 宏值项不为空。 `CREATE USER zabbix_` 如果 {$ORACLE.CONNSTRING} 是一个会话名称，那么此 {$ORACLE.CONNSTRING} 宏值项为空。\-- Grant access t **{$ORACLE.USER}**, **{$ORACLE.PASSWORD}** - Oracle 凭证 (default username: zabbix, password: zabbix_password). Required, 如果 {$ORACLE.CONNSTRING} 是一个 URI，那么此 {$ORACLE.USER}，{$ORACLE.PASSWORD} 宏值项不为空。 `GRANT SELECT ON D` 如果 {$ORACLE.CONNSTRING} 是会话名称，那么此 {$ORACLE.USER}，{$ORACLE.PASSWORD} 宏值项为空。GRANT SELECT ON	称命名。oracle 参考地址 Oracle Instant Client. 创建具有所需权限的 Oracle 用户, 运行: on IDENTIFIED BY <PASSWORD>; the zabbix_mon user. GRANT CONNECT, CREATE SESSION TO zabbix_mon A_TABLESPAC TO zabbix_mon; BA_TABLESPA TO zabbix_mon; GRANT SELECT ON DBA_USERS TO zabbix_mon GRANT SELECT ON SYS.DBA_DA

模板名称强制性	宏添加步骤/说明	
Template DB PostgreSQL by Zabbix agent 2	**{$PG.URI}** - 连接字符串; 下面格式定义可以是会话名称或者是一个 URI: 与 *PostgreSQL* 插件一起使用；支%% <protocol(host:port or /path/to/socket)/>%%. 对于 URI，只支持 TCP 和 Unix 模式。 例子: Postgres1, tcp://localhost:5432 (default), tcp://172.16.0.10 创 **{$PG.USER}**, **{$PG.PASSWORD}** - PostgreSQL 凭证 (default username: postgres, password:postgres). 如果 {$PG.URI} 是一个 URI，那么此 {$PG.USER}，{$PG.PASSWORD} 宏值项不为空。如果 {$PG.URI} 是一个会话名称，那么此 {$PG.USER}，{$PG.PASSWORD} 宏值项为空。GRANT EXECUTE ON FUNCTION pg_catalo	会话名称命名。具有所需权限的用户，适用于 Post-greSQL 10 及更新版本，运行: REATE USER 'zbx_monitor' IDEN-TI-FIED BY '<pass-word>'; .pg_ls_dir(text TO zbx_monitor;\ EXECUTE ON FUNCTION pg_catalog TO zbx_monito 编辑 pg_hba.conf 允许 Zab-bix agent 连接访问 (详细信息访问PostgreSQL

模板名称强制性	宏添加步骤/说明	
Template DB Redis	**{$REDIS.CONN.URI}** -URI 格式的连接字符串；端口可选择；密码不被使用。 与 *Redis* 模块一起使用；支持会话名如果不设置, 模块默认的值: tcp://localhost:6379	命名。测试可用性，运行: `zabbix_get -s redis-mast -k redis.ping`

Zabbix agent 模板操作

下面步骤确保当前操作模板能通过Zabbix agent正常收集监控对象数据:

1. 请确保主机上已安装 Zabbix agent。Zabbix agent active 模式检查，确保主机 ip 信息填写入参数'ServerActive' 在 Zabbix agent 配置文件中配置文件.
2. 链接 将模板关联上目标主机 (如果在 Zabbix 中没有 agent 可用的模板，您可能需要首先导入模板的.xml 文件-参见 - 看开箱即用的模板 说明部分).
3. 根据提示修改强制宏值修改。
4. 配置监控对象实例允许 zabbix 去获取数据 - 请参阅附加步骤/注释栏中的说明。

Note:
本页仅包含适当模板操作所需的最小宏集和设置步骤。模板的详细描述，包括宏、项和触发器的完整列表，都可以在模板中找到 Readme.md 文件 (点击模板名称即可访问).

模板名称	必要宏	额外的步骤/	明
Microsoft Exchange Server 2016 by Zabbix agent/Microsoft Exchange Server 2016 by Zabbix agent active			Note, 模板没有提供关于 Windows 服务状态的信息。建议配合使用 *OS Windows by Zabbix agent* 或者 *OS Windows by Zabbix agent active* 模板.

模板名称必要宏	额外的步骤/ 明
Template App Apache by Zabbix agent	**{$APACHE.STATUS.H** - Apache 状态页面的主机名或 IP 地址 (默认: 127.0.0.1) 应该设置 Apache 模块 **{$APACHE.STATUS.P** - URL 路径 (默认: server-status?auto) 检查可 **{$APACHE.STATUS.P** - Apache 状态页面的端口 (默认: 80) httpd - |mod 状态 (通过 Apache [文档](https://httpd.apach 查看详情). 性, 运行: 2>/dev/null | grep sta-tus_module Apache 配置示例: <Location "/server-status"> SetHandler server-status Require host example.com

模板名称	必要宏	额外的步骤/	明
Template App HAProxy by Zabbix agent	**{$HAPROXY.STATS.PA** - HAProxy统计页面的路径(默认: stats) 该设置启用HA **{$HAPROXY.STATS** - HAProxy的端口统计主机或容器(默认: 8404) **{$HAPROXY.STATS.S** - HAProxy的scheme统计页面. 支持协议: http(默认), https		统计页面(通过HAProxy博客查看详细信息或模板的README.md配置示例).

模板名称	必要宏	额外的步骤/	明
Template App IIS by Zabbix agent / Template App IIS by Zabbix agent active		**{$IIS.PORT}** - IIS Server 监听端口 (默认: 80) 服务应该遵 -端口检查服务 (默认: http)。查看更多细节. [Web Server](h	以下规则: tps://docs.micr us/iis/web-hosting/web-server-for-shared-**{$IIS.SERVICE}** nstalli the-web-server-role) IIS Man-age-net.tcp.port service Scripts and Tools 更多详细内容轻擦好看 IIS 文档。

模板名称必要宏	额外的步骤/	明
Template App Nginx by Zabbix agent	**{$NGINX.STUB_STAT** - 主机名或者IP地址是Nginx stub_status系统主机或者容器(默认: localhost) ngi - Nginx stub_status页面路径(默认: basic_status)\ 检查可用性 **{$NGINX.STUB_ST - Nginx stub_status主机或容器的端口(默认: 80)nginx -V	Nginx需要被设置(更多信息请查看Nginx [文档](https://ngin 或者模板的Readme.md对于配置示例)运行: 2>&1 | grep -o with-http_stub_statu

模板名称必要宏	额外的步骤/	明
Template App PHP-FPM by Zabbix agent	**{$PHP_FPM.HOST}** - PHP-FPM 状态主机或容器的主机名或 IP (默认: localhost) 打开 php-fpm **{$PHP_FPM.PING** - PHP-FPM ping 页面路径 (默认:ping) **{$PHP_FPM.PORT}** - PHP-FPM 状态主机或容器的端口 (默认:80) **{$PHP_FPM.PROCESS_NAME}** - PHP-FPM 进程名称 (默认:php-fpm) **{$PHP_FPM.STATUS.** - PHP-FPM 状态页面路径 (默认:status) 2. 验证命令	置文件并启用状态页面: atus_path = /status 1. tus /ping 置格式: $ php-fpm7 -t 3. pm.s** {$PH 重新加载 php-fpm 服务. \ping.path **{$PH** 4. 在 Ng-inx 配置中 pm) 文件 server 块 (虚拟主机) , 添加 (更多信息请查看模板 *Readme.md* 带有注释

模板名称必要宏	额外的步骤/	明
Template App RabbitMQ cluster by Zabbix agent	**{$RABBITMQ.API.CLU** - RabbitMQ 集群 API 端点的主机名或 IP 地址 (默认:127.0.0.1) 用 RabbitMQ 管理 **{$RABBITMQ.API.USER}**, **{$RABBITMQ.API.PASSWOR** - RabbitMQ 登录凭证 (默认 username: zbx_monitor, password: zabbix)	件 (see RabbitMQ 文档). 创建一个 RabbitMQ 启用户，该用户具有 Necessary 的监控权限，运行: ” rabbitmqctl add_user zbx_monitor <PASSWORD> ” `rabbitmqctl set_permiss -p / zbx_monitor %% "" "" ".*"%% rabbitmqctl set_user_tag zbx_monitor monitoring` 如果集群由多个节点组成，建

模板名称	必要宏	额外的步骤/	明
Template DB MySQL		**{$MYSQL.HOST}** - MySQL 主机或容器的主机名或 IP 地址(默认: local-host) **{$MYSQL.PORT}** - 数据库服务端口(默认: 3306)	1. 如果需要，将 mysqlad-min 实用程序的路径添加到全局环境变量 PATH 中。 2. 复制 template_db_mysql.conf 文件从 *templates* 将 Zabbix 的目录放入 Zabbix 代理配置的文件夹中(/etc/zabbix/zabbix_agentd.conf.d 默认) 然后重启 Zabbix agent. 3.

模板名称必要宏	额外的步骤/	明
Template DB PostgreSQL	**{$PG.DB}** - 库名去连接 post-gres 数据库服务 (默认: post-gres) 1. 创建一个只读用 **{$PG.HOST}** - 数据库主机地址或者 socket 目录 (默认:127.0.0.1) **{$PG.PORT}** – 数据库服务端口 (默认: 5432)\GRANT p **{$PG.USER}** - 数据库用户名 (默认: zbx_monitor) For old	zbx_monitor 可以正常访问 Post-greSQL 服务器。适用于 Post-greSQL 10 及更新版本, 运行: zbx_monitor WITH CREATE USER **** PASS-WORD ’<PASS-WORD>’ IN-HERIT; _monitor GRANT USE_monitor; r Post-greSQL ver-sions, run: CREATE USER zbx_monitor WITH PASSWORD '<PASSWORD> GRANT SELECT ON pg_stat_dat TO zbx_monitor 2. 复制 postgresql/ 到 Zab-bix agent `

模板名称必要宏	额外的步骤/	明

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（基本上是机框）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）：

- 机框故障监控（电源，风扇和温度，总体状态）
- 机框性能监控（CPU 和内存项）
- 机框资产收集（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;
- 如果是从旧版本升级的 ZABBIX，你可以在下载的最新版本的 ZABBIX 的 templates 目录中找到模板文件。在经过手工导入这些模板文件后，可以在配置 → 模板页找到模板。

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.]*$
```

更新后，会过滤掉在大多数系统上环回和空接口。

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	操作系统使用的 MIB 库	**标签(/zh/manual/gettemplates_out_of_the_box/network_	
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
Template Net Brocade_Foundry Nonstackable SNMPv2	Brocade Foundry	Brocade MLX, Foundry	Brocade MLXe, Foundry FLS648, Foundry FWSX424		FOUNDRY-SN-AGENT-MIB	Performance, Fault

模板名称提供商	设备系列	已知模型	操作系统使用的 MIB 库	**标签(/zh/manual/gettemplates_out_of_the_box/network_	
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/DGS 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
Template Net Juniper SNMPv2	Juniper	MX,SRX,EX mod-els	Juniper MX240, Juniper EX4200-24F	Junos	JUNIPER-MIB	Certified

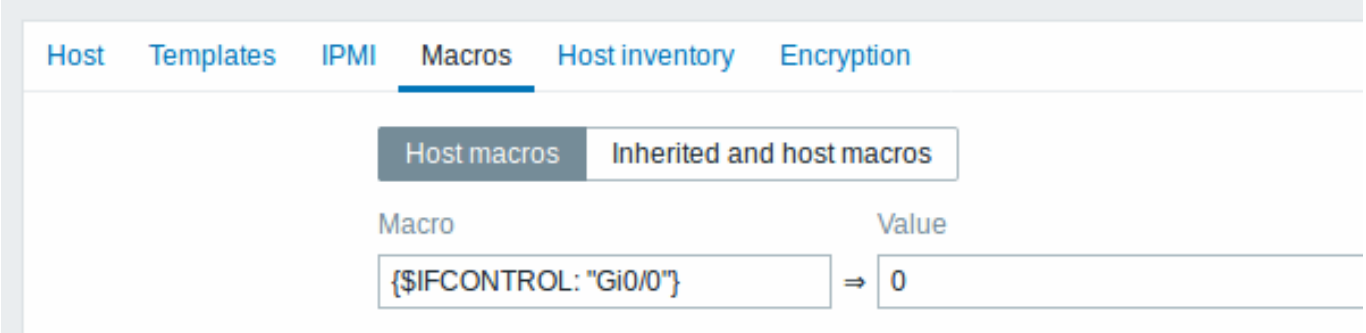
模板名称提供商	设备系列	已知模型	操作系统使用的 MIB 库	**标签(/zh/manual/guide	templates_out_of_the_box/network_
Template Net Mellanox SNMPv2	Mellanox	Mellanox In-finiband de-vices	SX1036	MLNX-OS	HOST-RESOURCES-MIB,ENTITY-MIB,ENTITY-SENSOR-MIB,MELLANOX-MIB	Certified
Template Net Mikrotik SNMPv2	Mikrotik	Mikrotik RouterOS de-vices	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,UFO	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：

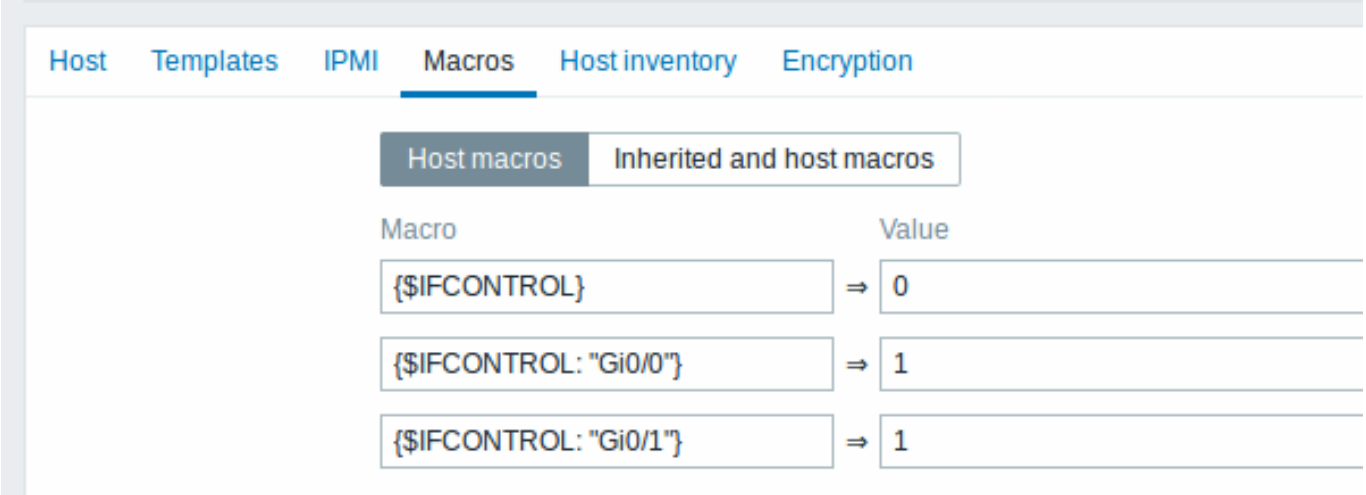
- 尽可能使用用户宏，让用户可以自主调节触发器
- 尽可能使用低级别发现来最小化不支持的项目数
- 所有模板都依赖于 Template ICMP Ping，会通过 ICMP 也会检查所有设备
- 项目不使用任何 MIB - SNMP OID 用于项目和低级别发现。因此，没有必要将任何 MIB 加载到 Zabbix 中以使模板工作。
- 当发现接口以及 ifAdminStatus = down（2）的接口时，环回网络接口被过滤

- 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} 的值。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

- Performance - 设备系列 MIB 提供了一种监控 CPU 和内存项的方法;
- Fault - 设备系列 MIB 提供监控至少一个温度传感器的方法;
- Inventory - 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;
- Certified - 涵盖上述所有三个主要类别。

9 事件通知

概述

假设我们已经配置了一些监控项和触发器，现在由于触发器状态的改变而发生了一些事件，那么接下来要考虑的就是动作。

首先，我们不希望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（如问题）时能够接收到通知。并且，当发生问题时，我们希望所有相关人员都能收到通知。

这就是为什么发送通知是 Zabbix 提供的主要动作之一。我们可以定义在某一事件发生时，应该通知何人以及何时通知。

为了能够发送和接收 Zabbix 的通知，您必须：

- 定义媒介
- 配置动作 向已定义的媒介发送消息

动作由 条件和 操作组成。总的说来，当条件满足时，则执行相应的操作。两个主要的操作是发送消息（通知）和执行远程命令。

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。

1 媒介类型

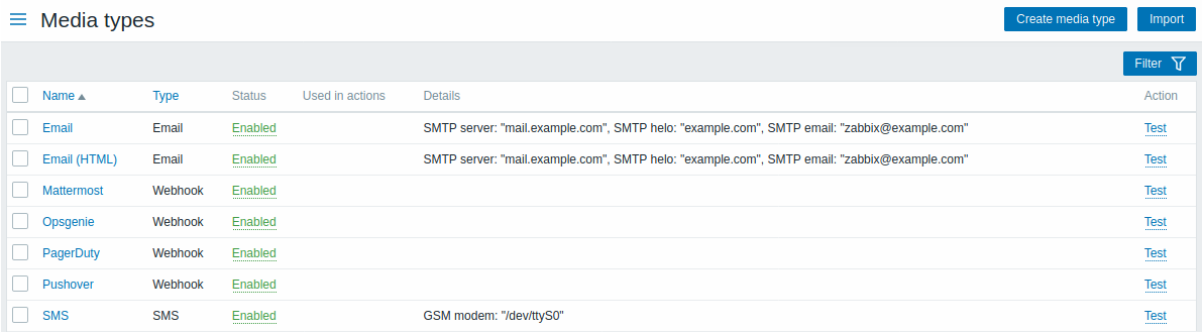
概述

媒介是 Zabbix 中用于发送通知和告警的传输通道。

您可以配置多种媒介类型：

- 电子邮件
- 短信
- 自定义报警脚本
- Webhook

媒介类型在 管理 → 媒介类型中进行配置。



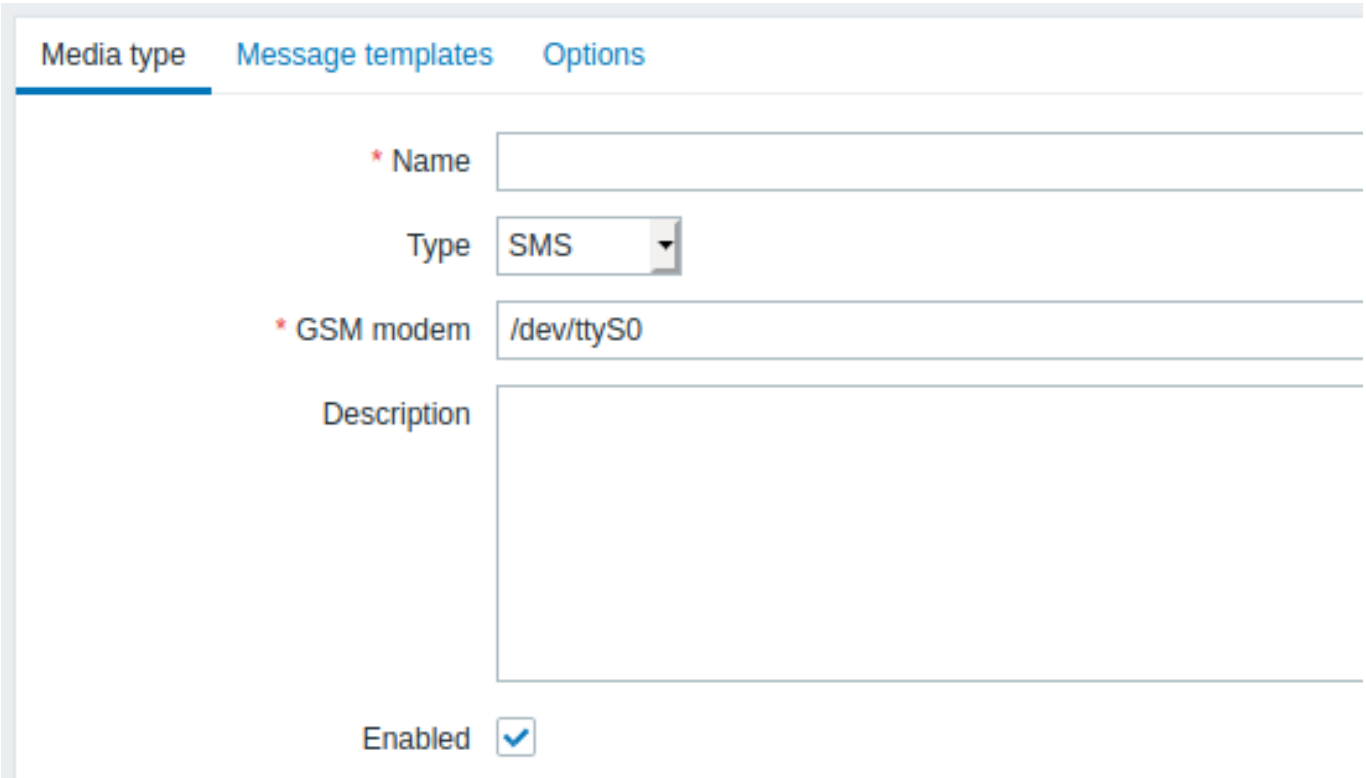
有些媒介类型是在默认数据集中预定义的。您只需要微调它们的参数，即可使其工作。

单击最后一列的 // 测试 //，可以测试配置好的媒介类型是否工作正常。（更多详细信息请参见媒介类型测试 ）。

点击 创建媒介类型按钮来创建一个新的媒体类型，就会打开一个媒体类型的配置表单。

通用参数

有些参数对于所有媒介类型都是通用的。



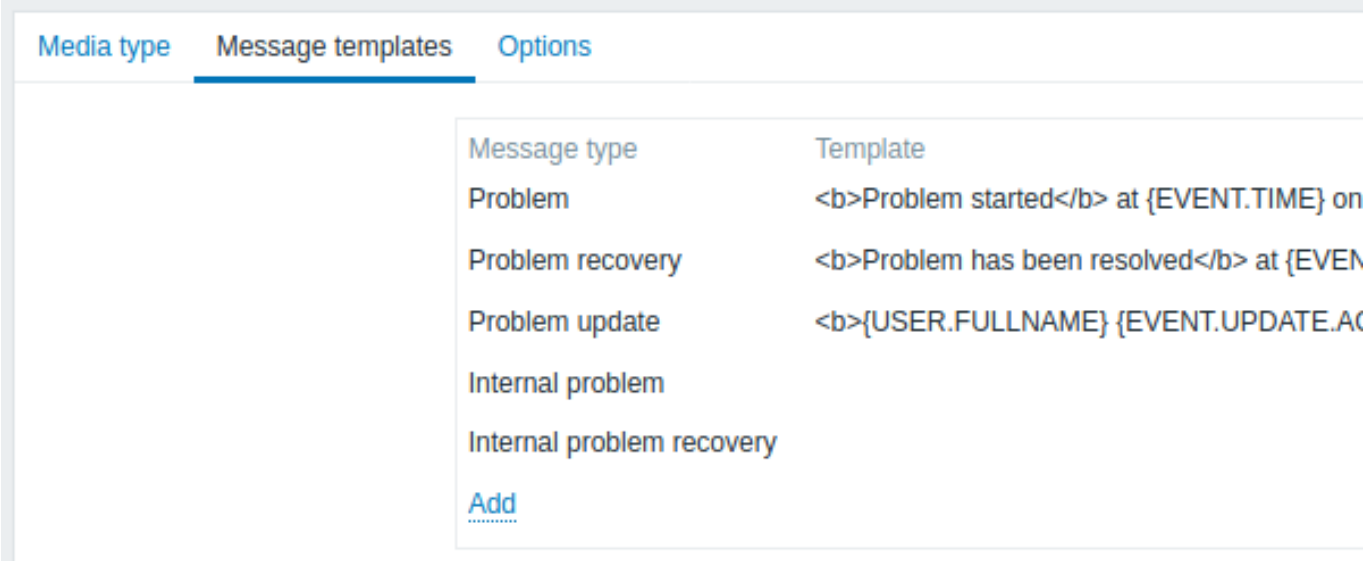
在 媒介类型选项卡中，常见的一般属性有：

参数	说
名称	媒类型的名称。
类型	选媒介类型。
描述	输媒介类型的描述。
已启用	选中选框以启用此媒介类型。

有关媒介特定的参数，请参见媒介类型的各个页面。

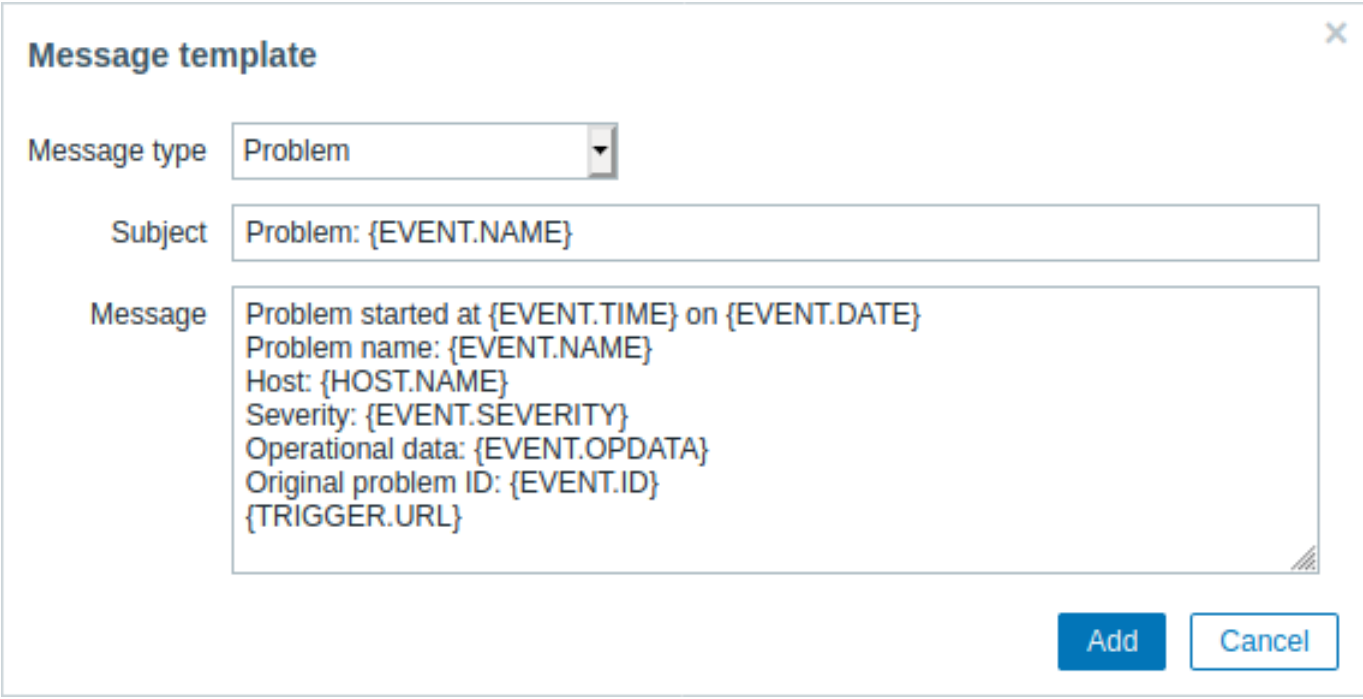
消息模板选项卡可以为以下的所有或部分事件类型设置默认的通知消息：

- 问题
- 问题恢复
- 问题更新
- 自动发现
- 自动注册
- 内部问题
- 内部问题恢复



自定义消息模板的步骤:

- 在 消息模板选项卡中，点击 Add 将打开一个 消息模板的弹出窗口。
- 选择所需的 消息类型，编辑 主题及 消息文本。
- 点击 添加保存消息模板



消息模板的参数:

参数	描	
消息类型	择为	种事件类型定义默认消息。 每种事件类型只能定义一个默认消息。

参数	描	
主题	默	消息的主题（可包含宏）。主题长度最多为 255 个字符。 主题不适用于短消息媒介类型。
消息	默	消息的内容（可包含宏）。具体能在消息中输入多少个字符取决于数据的类型（了解更多信息请参见发送消息）。

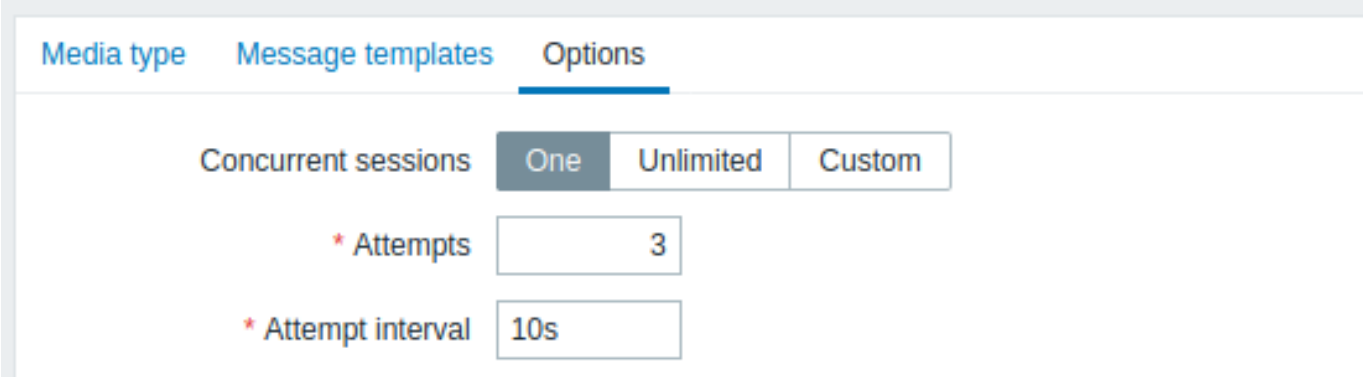
要更改现有的消息模板，请执行以下操作：在 动作那一栏点击 Edit 编辑模板或者点击 Remove 删除消息模板。

还可以为每个动作单独定义通知消息模板（详情请见动作操作 ）。动作中的自定义消息可以覆盖为媒介类型配置的消息模板。

Warning:
必须为所有媒体类型定义消息模板，包括未使用默认消息通知的 webhook 或自定义报警脚本。比如，如果没有为 Pushover webhook 定义异常消息，“Send message to Pushover webhook” 将无法发送异常通知。

选项选项卡包含了告警处理设置。每种媒介类型都可以配置相同的选项集。

所有媒介类型都是并行处理的。虽然每个媒介类型的最大并发会话数是可配置的，但服务器上报警器的进程总数只能通过 StartAlerters 参数 进行限制。由一个触发器生成的多个告警是按顺序进行处理的。因此，只有当多个触发器生成多个通知时，才能同时处理多个通知。



参数	描
并发会话为该媒	类型选择并行报警程序会话的数量：壹 - 单会话无限 - 不限制会话数量自定义 - 自定义并行会话数量Unlimited/high 意味着在发送通知时会产生更多并行会话且会话数量不断增加。Unlim-ited/high 应该在需要同时发送大量通知的大型环境中使用。
尝试次数尝试发	通知的次数。最大可设置为 10；默认值为’3’。如果设置为’1’，Zab-bix 将仅发送一次通知，即使并且如果发送失败将不会重试。

参数	描
尝试间隔在发送	败的情况下尝试重新发送通知的频率，单位为秒（0-60）。如果设置为’0’，发送失败后将立即重发。支持定义时间后缀，如 5s, 1m。

媒介类型测试

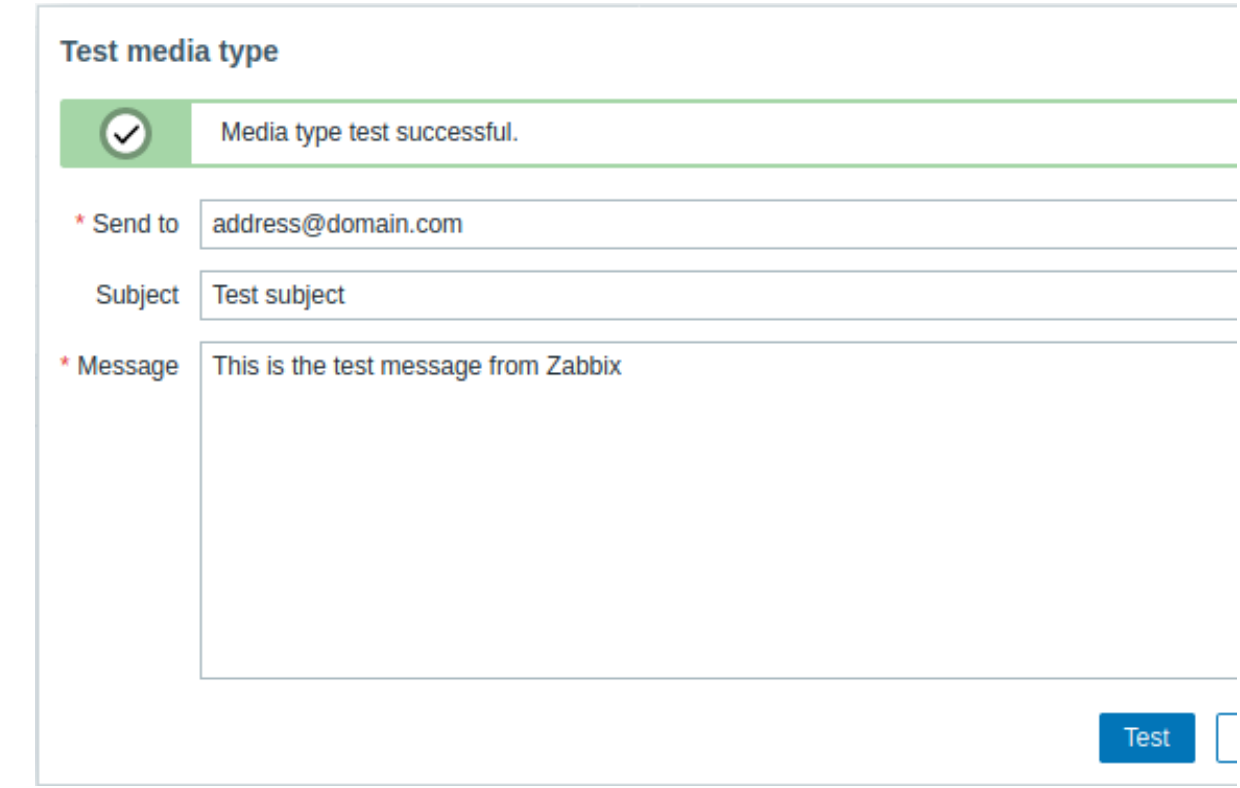
测试配置好的媒介类型是否正常工作。

电子邮件

例如，测试电子邮件媒介类型：

- 在媒介类型列表 中找到相关的电子邮件
- 点击列表最后一栏中的 测试（将打开一个测试窗口）
- 在 收件人字段输入收件人的地址，设置通知主题（可选）及消息内容。
- 点击 测试发送测试消息

测试成功或失败的消息将在同一窗口显示：

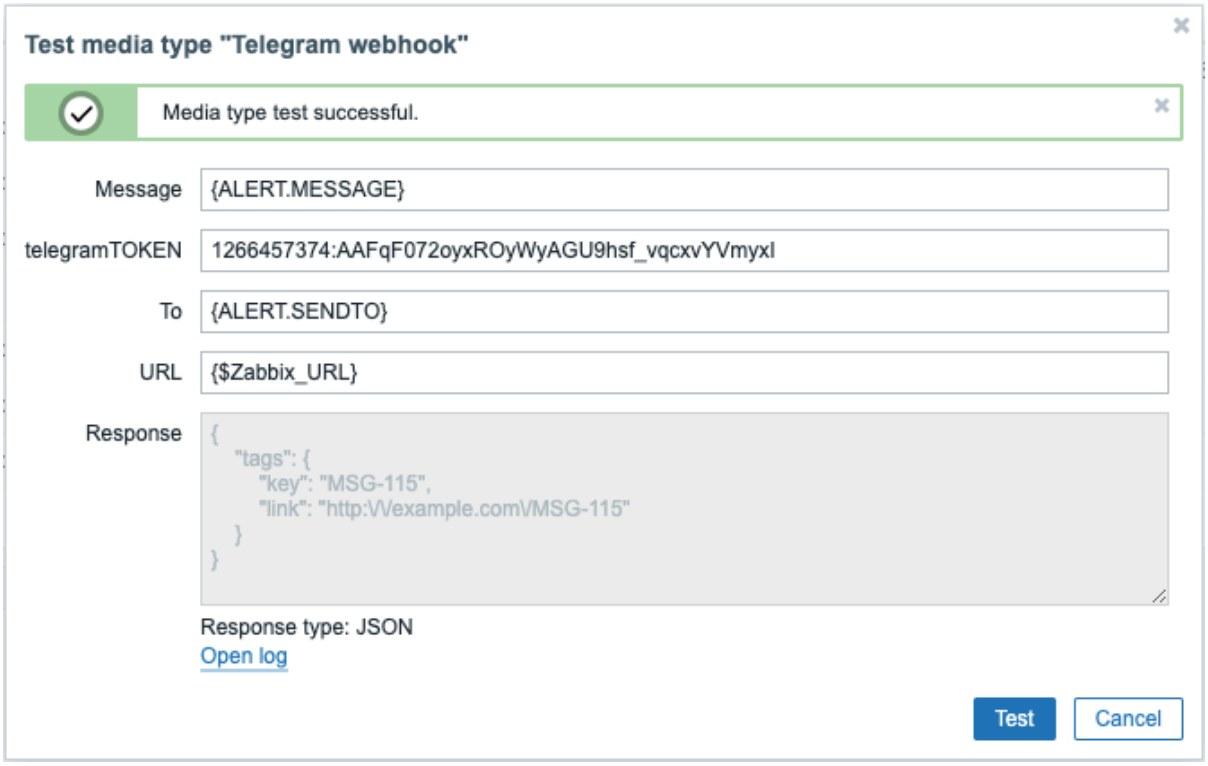


**Webhook **

测试 webhook 媒介类型:

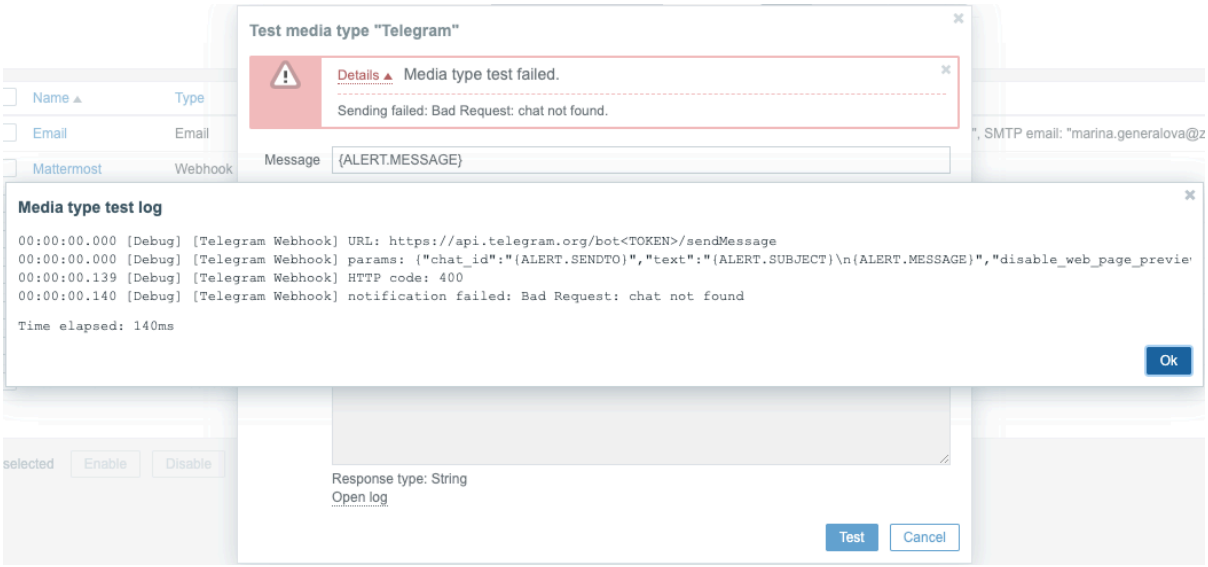
- 在媒介类型的列表 中找到相关的 webhook
- 点击列表最后一栏中的 测试（将打开一个测试窗口）
- 根据需要来编辑 webhook 参数值
- 点击 测试

默认情况下，测试 webhook 时使用的是之前在配置 webhook 过程中填入的参数。但是，我们也可以更改其属性值来进行测试。替换或删除测试窗口中的值仅影响测试过程，实际的 webhook 属性值将保持不变。



查看媒介类型测试的日志条目而不离开此测试窗口：

- 点击 打开日志（将打开一个新的弹窗）。



若 webhook 测试成功

- 将会显示 *"Media type test successful."* 的消息
- 服务器响应信息将显示在灰色的 *Response* 字段
- 在 *Response* 字段的下方会显示其响应的类型（JSON 或是 String）

若 webhook 测试失败

- 将会显示 *"Media type test failed."* 的消息, 及执行失败的详细信息。

用户媒介

只有在用户配置文件中定义了此媒介类型的媒体（电子邮件地址/电话号码/webhook 用户 id 等）时，用户才会收到该媒介类型的通知。例如，如果在用户配置文件中未定义 webhook "X" 媒介，那么在使用 webhook "X" 向用户 "Admin" 发送消息这个动作时，该消息将无法被发送出去。

定义用户媒介的步骤：

- 转到 管理 → 用户
- 打开用户属性表单
- 在报警媒介选项卡中，点击 Add

用户媒介属性：

参数	描
类型	下菜单中包含了所有已配置的媒体类型的名称。

参数描	
发送到提供	需的联系信息，消息将发送至此。 对于电子邮件媒介类型，可以通过点击地址字段下面的 Add 来添加多个地址。在这种情况下，通知将发送到所有提供的电子邮件地址。还可以在邮件收件人的 发送到字段中，以’Recipient name <address1@company.co 格式指定收件人。注意：如果提供了收件人的姓名，则电子邮件地址应该用尖括号括起来(<>)。收件人的名称支持 UTF-8 字符，不支持引号对和注释。例如: *John Aber-croft <man-*

参数	描
何时发送您可以	制消息发送的时间，例如仅限在工作日 (1-5,09:00-18:00)。有关格式的说明请参见时间段格式页面。
告警级别勾选复	框，要标记要接收何种严重程度的消息。注意如果您想接收非触发事件 的通知，请 务必勾选默认严重性 (’未分类’) 复选框。保存后，选中的触发级别将以相应的级别颜色显示，未选中的将显示为灰色。
状态用	媒介的状态。已启用 - 使用中。禁用 - 未被使用。

1 电子邮件

概述

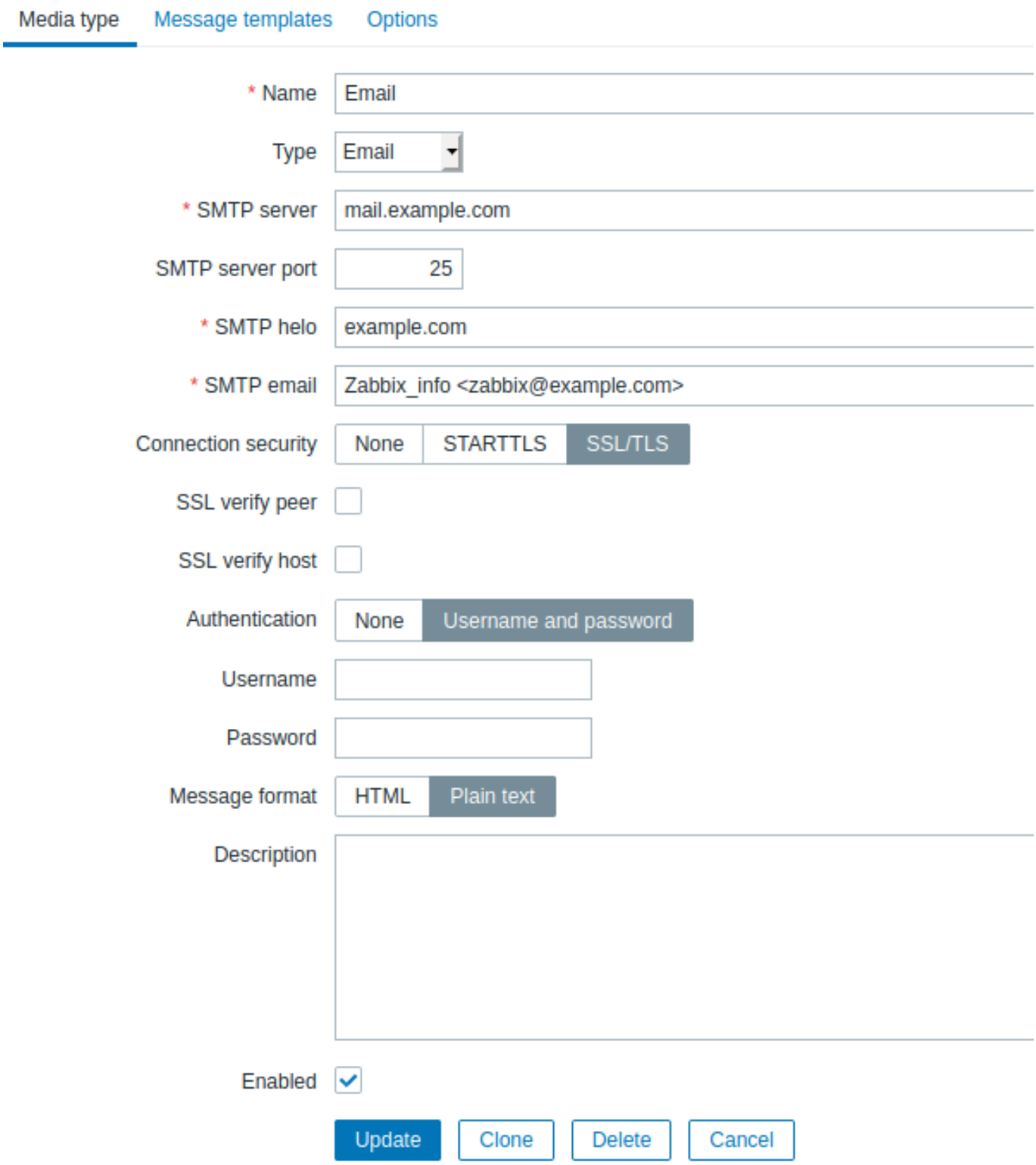
要将电子邮件配置为消息的传递通道，您需要将电子邮件配置为媒介类型，并为用户分配具体的邮件地址。

配置

配置电子邮件为媒介类型：

- 转到 管理 - > 媒介类型
- 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* ）

媒介类型选项卡中包含了一般的媒介类型属性:



标有红色星号的为必填字段。

参数	描
SMTP server	设置SMTP服务器来发送邮件。
SMTP server port	设置SMTP服务器端口来处理传出的消息。*Zabbix 3.0*之后支持此选项。
SMTP helo	设置正确的SMTP helo值，通常为域名。

参数	描
SMTP email	此处输入的地址将作为发送邮件的地址。Zabbix 2.2 版本之后，支持使用实际的电子邮件地址添加发件人显示名(如上图中 *Zabbix_info <zabbix@company.co* 的 “Zabbix_info”)。与 RFC 5322 允许的相比，Zabbix

参数	描
Connection security	选择连接安全级别： **None** - 不使用 CURLOPT_USE_SSL 选项 **STARTTLS** - 使用有 CURLUSESSL_ALL 值的 CURLOPT_USE_SSL 选项 **SSL/TLS** - 是否使用 CURLOPT_USE_SSL 为可选项 *Zabbix 3.0* 之后支持此选项。

参数	描
SSL verify peer	选中此复选框以验证 SMTP 服务器的 SSL 证书。“SSLCALocation” 服务器配置指令的值应该放到 CURLOPT_CAPATH 中以进行证书验证。设置 cURL 选项 CURLOPT_SSL_VERIFY*Zabbix 3.0* 之后支持此选项。

参数	描
SSL verify host	选中此复选框以验证SMTP服务器证书的*Common Name*字段或*Subject Alternate Name*字段是否匹配。 设置cURL选项CURLOPT_SSL_VERIFY。 *Zabbix 3.0*之后支持此选项。

参数	描
Authentication	选择认证级别： **None** - 不设置 cURL 选项（3.4.2 版本之后） **User name and password** - 意味着”AUTH=*”将认证机制的选择留给了 cURL（3.4.2 版本之前） **Normal password** - CURLOPT_LOGIN_OPT 在”AUTH=PLAIN”中设置 *Zabbix 3.0* 之后支持此选项。

参数	描
Username	认证使用的用户名。 设置 CURLOPT_USERNAME 的值。 *Zabbix 3.0* 之后支持此选项。
Password	认证使用的密码。 设置 CURLOPT_PASSWORD 值. *Zabbix 3.0* 之后支持此选项。

参数	描
Message format	选择消息的发送格式：**HTML** - 以 HTML 格式发送 **Plain text** - 以纯文本格式发送

Attention:
要使 SMTP 验证选项可用，在编译安装 Zabbix server 时，应使用 cURL 7.20.0 或更高的版本，并使用--with-libcurl 编译选项。

有关如何配置默认消息及警报处理选项，请详见通用媒体类型参数 。

用户媒介

电子邮件媒介类型配置完成后，请转到 管理 → 用户部分，编辑用户配置文件，将电子邮件媒介分配给用户。用户媒介的配置步骤请参见媒介类型 页面（适用于所有媒介类型）。

2 短信

概述

Zabbix 支持使用连接到 Zabbix 服务器串行口的串行 GSM 调制解调器发送短信。

请确保满足以下条件：

- 串行设备的速率（Linux 下通常为/dev/ttyS0）与 GSM 调制解调器的速度一致。Zabbix 没有设置串行链路的速度。它使用默认设置。
- 'zabbix' 用户对串行设备具有读/写访问权限。执行 ls -l /dev/ttyS0 命令来查看当前串口设备的权限。
- GSM 调制解调器已经输入了 PIN 码，并且在电源复位后会将其保留。或者，您可以禁用 SIM 卡上的 PIN 码。可以通过在终端软件（如 Unix minicom 或 Windows HyperTerminal）中发出命令 AT + CPIN ="NNNN" 来输入 PIN 码（NNNN 是您的 PIN 码，且必须放在引号中）。

Zabbix 已通过以下 GSM 调制解调器的测试：

- Siemens MC35
- Teltonika ModemCOM/G10

配置短信作为消息的传送通道时，需要将短信配置为媒介类型，并输入相应用户的电话号码。

配置

配置短信作为媒体类型的步骤：

- 转到 管理 - > 媒介类型
- 点击 创建媒介类型（或者点击预定义媒介类型列表中的 *SMS*）。

以下参数仅适用于短信媒介类型：

参数描	
GSM modem	设置 GSM 调制解调器串口设备的名称。

有关如何配置默认消息及警报处理选项，请详见通用媒体媒介类型参数 。要注意的是，Zabbix 是无法并行处理发送短信通知的。

用户媒介

短信媒介类型配置完成后，请转到 管理 → 用户部分，编辑用户配置文件，将短信媒介分配给用户。用户媒介的配置步骤请参见媒介类型页面（适用于所有媒介类型）。

3 自定义报警脚本

概述

如果您对目前发送告警的媒介类型不满意，那么还有另外一种方式来执行此操作。您可以创建一个脚本，按照您自己的方式来处理通知。

告警脚本在 Zabbix server 上执行。这些脚本的存放目录，是服务器配置文件 中 **AlertScriptsPath** 变量所定义的。

以下是一个报警脚本的示例：

```
#####!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF
```

<note important> 从 3.4 版本开始，Zabbix 会检查执行命令和脚本的退出代码。若退出代码不为 **0**，则均被视为命令执行 错误。在这种情况下，Zabbix 将尝试重复失败的执行。:::

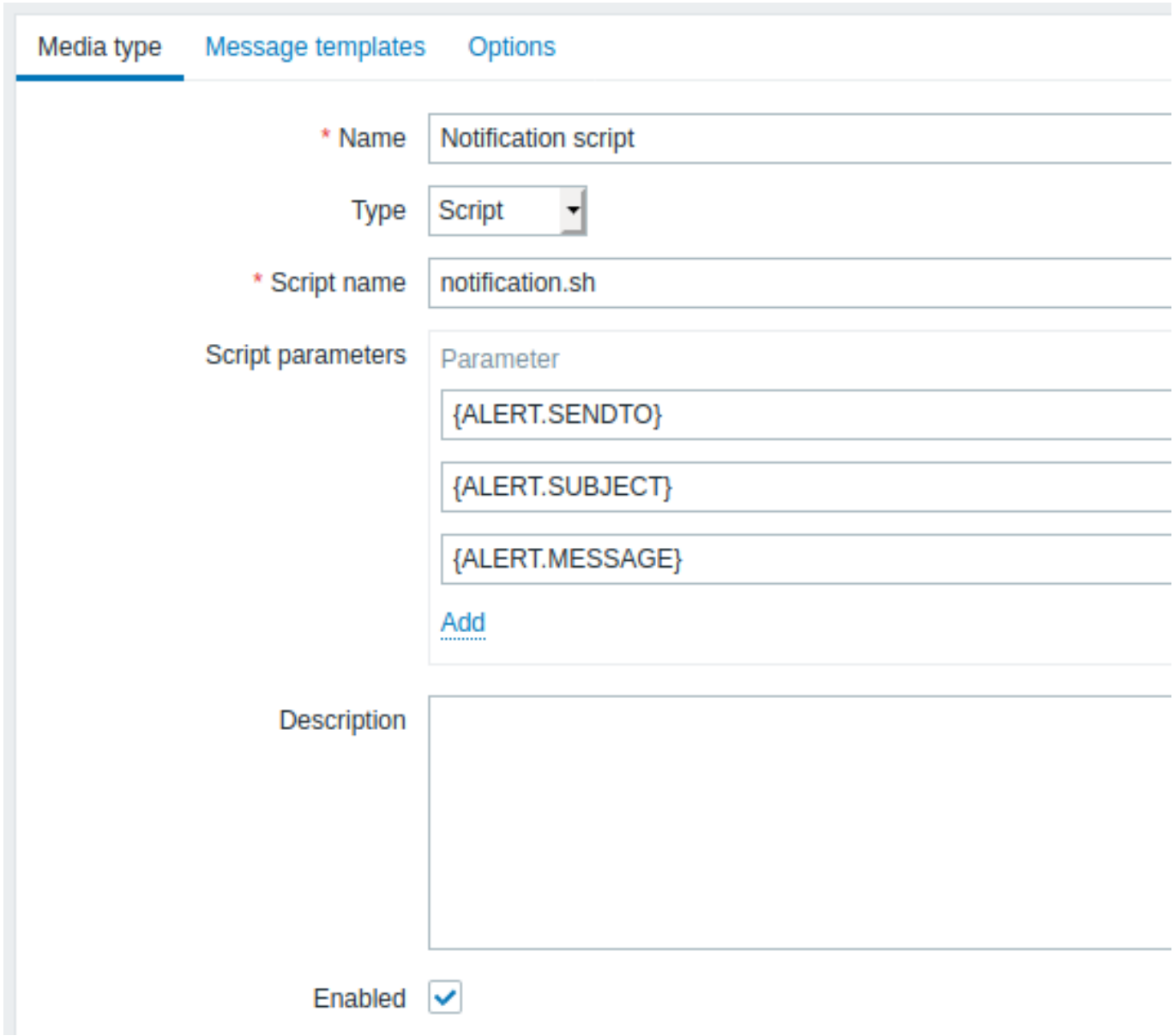
环境变量不会为脚本保留或创建，因此应该明确地来处理它们。

配置

配置自定义告警脚本为媒介类型：

- 转到 管理 - > 媒介类型
- 点击 创建媒介类型

媒介类型选项包含了媒介类型的一般属性：



红色星号标记的为必填字段。

以下为脚本媒介类型所特有的参数：

参数	描
脚本名称	输入脚 的名称。

参数描	
脚本参数向脚本	加命令行参数。脚本中支持 {ALERT.SENDTO}, {ALERT.SUBJECT} 和 {ALERT.MESSAGE} 宏 参数。从 Zabbix 3.0 开始支持自定义脚本参数。

有关如何配置默认消息和报警处理选项，请详见通用媒介类型参数 。

Warning:
即使自定义报警脚本不使用默认消息，也必须为此媒介类型使用的操作类型定义消息模板，否则将不会发送通知。

Attention:
从 Zabbix 3.4.0 开始，就已实现了媒介类型的并行处理。因此需要注意的是，如果配置了多个脚本媒介类型，这些脚本可能会被报警进程并行处理。报警进程的总数受 StartAlerters 参数 的限制。

用户媒介

媒介类型配置完成后，请转到 管理 → 用户部分，编辑用户配置文件，将此媒介分配给用户。用户媒介的配置步骤请参见媒介类型 页面 (适用于所有媒介类型)。

注意：定义用户媒介时，*Send to* 字段不能为空。如果在自定义脚本中不使用此字段，那么请在此字段中输入任何支持的字符组合，以绕过验证要求。

4 Webhook

概览

webhook 媒介类型对于使用自定义的 JavaScript 代码进行 HTTP 调用非常有用，它可以直接与外部软件 (如 Helpdesk 系统、聊天工具或信使) 集成。您可以选择导入 Zabbix 已提供的集成，也可以从头创建一个自定义集成。

集成

以下集成允许使用预定义的 webhook 媒体类型来推送 Zabbix 通知:

- Discord
- iLert
- iTop
- Jira
- Jira Service Desk
- Mattermost
- Microsoft Teams
- Opsgenie
- OTRS
- Pagerduty
- Pushover
- Redmine
- Rocket.Chat
- ServiceNow

- SIGNL4
- Slack
- SolarWinds
- SysAid
- Telegram
- TOPdesk
- Zammad
- Zendesk

Note:
除了这里列出的服务外，Zabbix 还可以集成 **Spiceworks** (无需 webhook)。要把 Zabbix 通知转换成 Spiceworks 对象，需先创建一个电子邮件媒体类型，在指定的 Zabbix 用户配置文件设置中输入 Spiceworks helpdesk 的邮件地址 (例如 help@zabbix.on.spiceworks.com)。

配置

开始使用 webhook 集成:

1. 在已下载的 Zabbix 程序的 `templates/media` 目录中，找到所需的.xml 文件；或者从 Zabbix 的git 仓库中下载
2. 将文件导入到 Zabbix 安装中。Webhook 将出现在媒体类型列表中。
3. 根据 *Readme.md* 文件的说明来配置 webhook (你也可以点击 webhook's 名称来快速访问 *Readme.md*)。

从零开始创建一个自定义的 webhook:

- 转到 // 管理 → 媒介类型 //
- 点击 // 创建媒体类型 //

媒体类型选项卡包含了针对这种媒体类型的各种属性:

Media type | Message templates | Options

* Name: Pushover

Type: Webhook

Parameters

Name	Value	Action
endpoint	https://api.pushover.net/1/messag	Remove
eventid	{EVENT.ID}	Remove
expire	1200	Remove
message	{ALERT.MESSAGE}	Remove
priority	0	Remove
retry	60	Remove
title	{ALERT.SUBJECT}	Remove
token	<PUSHOVER TOKEN HERE>	Remove
triggerid	{TRIGGER.ID}	Remove
url	{$ZABBIX.URL}	Remove
url_title	Zabbix	Remove
user	{ALERT.SENDTO}	Remove

Add

* Script: try {…

Timeout: 30s

Process tags

Include event menu entry

* Menu entry name

* Menu entry URL

Description: Please refer to setup guide here: https://git.zabbix.com/projects/ZBX/repos/zabbix/browse/templates/media/pushover

Set token parameter to your Pushover application key.
When assigning Pushover media to the Zabbix user - add the user key into send to field.

Enabled

Update | Clone | Delete | Cancel

红色星号标记的为必填字段。

webhook 媒体类型的具体参数如下:

参数描	
参数指	webhook 变量作为属性和值对。对于预先配置的 web-hook，参数列表会随着服务的不同而变化。检查 web-hook 的 *Readme.md* 参数说明文件。对于新的 web-hooks，默认情况下包含了几个常见的

参数	描
脚本	在击参数字段(或旁边的视图/编辑按钮)时出现的块中输入JavaScript代码。这段代码将执行web-hook操作。代码可以访问所有参数，它可以执行HTTP GET、POST、PUT和DELETE请求，并可控制HTTP头

参数	描
超时 J	vaScript 执行超时 (1-60s, 默认为 30s)。支持时间后缀, 例如 30s, 1m。

参数	描
Process tags	选中复选框标以将返回的JSON属性值作为标记处理。这些标记被添加到Zab-bix中已经存在的(如果有的话)问题事件标记中。

参数	描
Include event menu entry	选中复选框以在事件菜单中包含一个链接到创建的外部目标的条目。若选中此项，web-hook就不应该被用来向不同的用户发送通知(考虑创建一个专用用户)或者在几

参数	描
Menu entry name	指定菜单入口名称。支持{EVENT.TAGS.<tag name>}宏。只有当*Include event menu entry*被选中时，该字段才为必填项。

参数描	
Menu entry URL	指定菜单入口的URL。支持{EVENT.TAGS.<tname>}宏。只有当*Include event menu entry*被选中时，该字段才为必填项。

关于如何配置默认消息和警报处理选项的详细信息，请参见通用媒介类型参数

Warning:
即使 webhook 不使用默认消息，webhook 使用的操作类型的消息模板也必须定义。

用户媒介

媒介类型配置完成后, 转到 管理 → 用户部分，将 webhook 媒介分配给一个现有用户或创建一个新用户来表示 webhook。媒介类型 页面中描述了为现有用户设置用户媒体的步骤，这对于所有媒体类型都是通用的。

如果 webhook 使用标签来存储 ticket\message ID, 避免将同一个 webhook 作为媒体分配给不同的用户，因为这样做可能会导致 webhook 错误 (适用于大多数使用了 *Include event menu entry* 选项)。在这种情况下，最好的做法是创建一个专用的用户来表示 webhook:

1. 配置好 webhook 媒体类型后，前往 管理 → 用户部分，并创建一个专用的 Zabbix 用户来表示 webhook - 例如, 为 Slack webhook 添加一个别名 *Slack*。所有的设置可以保持默认值（除了媒体），因为这个用户不会登录到 Zabbix。
2. 在用户配置中，进入 *Media* 选项卡和添加 webhook ，提供所需的联系信息。如果 webhook 没有使用 发送到字段, 输入任何支持的字符组合来绕过验证要求。
3. 至少授予该用户对要向其发送警报的所有主机有可读的权限 。

配置告警动作时，请在 发送到用户字段中添加该用户 - 这将告诉 Zabbix 使用 webhook 来获取来自这个动作的通知。

配置告警动作

告警动作决定了哪些通知应该通过 webhook 发送。webhook 的配置动作 步骤与所有其他媒体类型相同，但有以下例外:

- 如果 webhook 使用标签来存储 ticket\message ID 以及后续的 update\resolve 操作, 这个 webhook 不应该用于单个问题事件的多个告警动作中。这适用于 Zabbix 提供的 Jira, Jira Service Desk, Mattermost, Opsgenie, OTRS, Redmine, ServiceNow, Slack, Zammad 和 Zendesk webhooks 以及大多数使用 *Include event menu entry* 选项的 webhook。允许在多个操作中使用 webhook，

如果这些操作或升级步骤属于同一个操作。在不同的动作中使用这个 webhook 也是可以的，由于不同的筛选器条件，操作不会应用到相同的问题事件中。

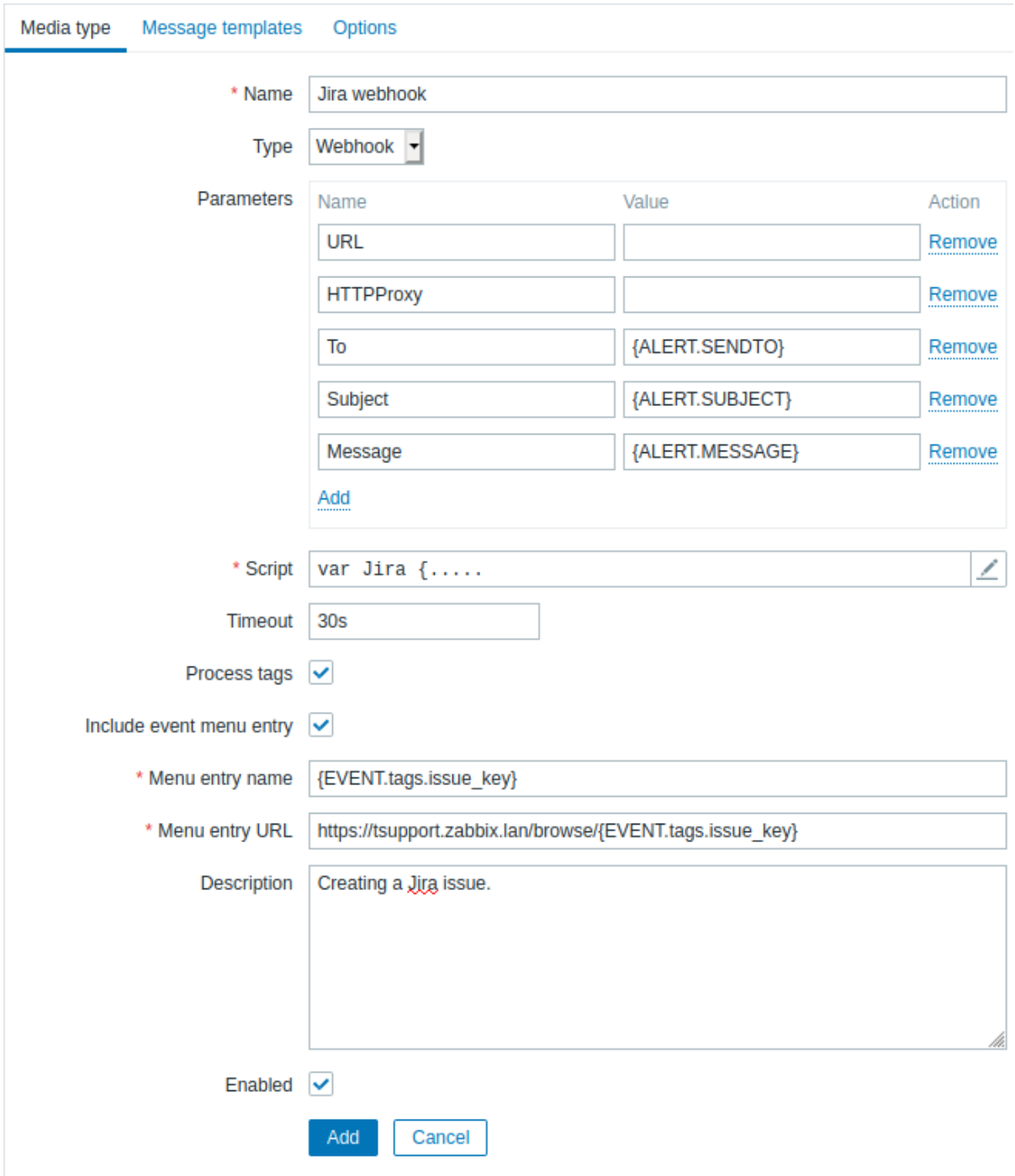
- 当在动作中使用 webhook 用于内部事件: 在动作操作配置中，选中复选框 自定义消息复选框，输入自定义消息内容，否则通知不会被发送出去。

Webhook 脚本范例

概述

尽管 Zabbix 提供了大量现成的 webhook 集成，但您可能想要创建自己的 webhook。本节提供了自定义 webhook 脚本的示例 (在 脚本参数中使用)。有关其他 webhook 参数的说明，请参见webhook 章节。

Jira webhook (自定义)



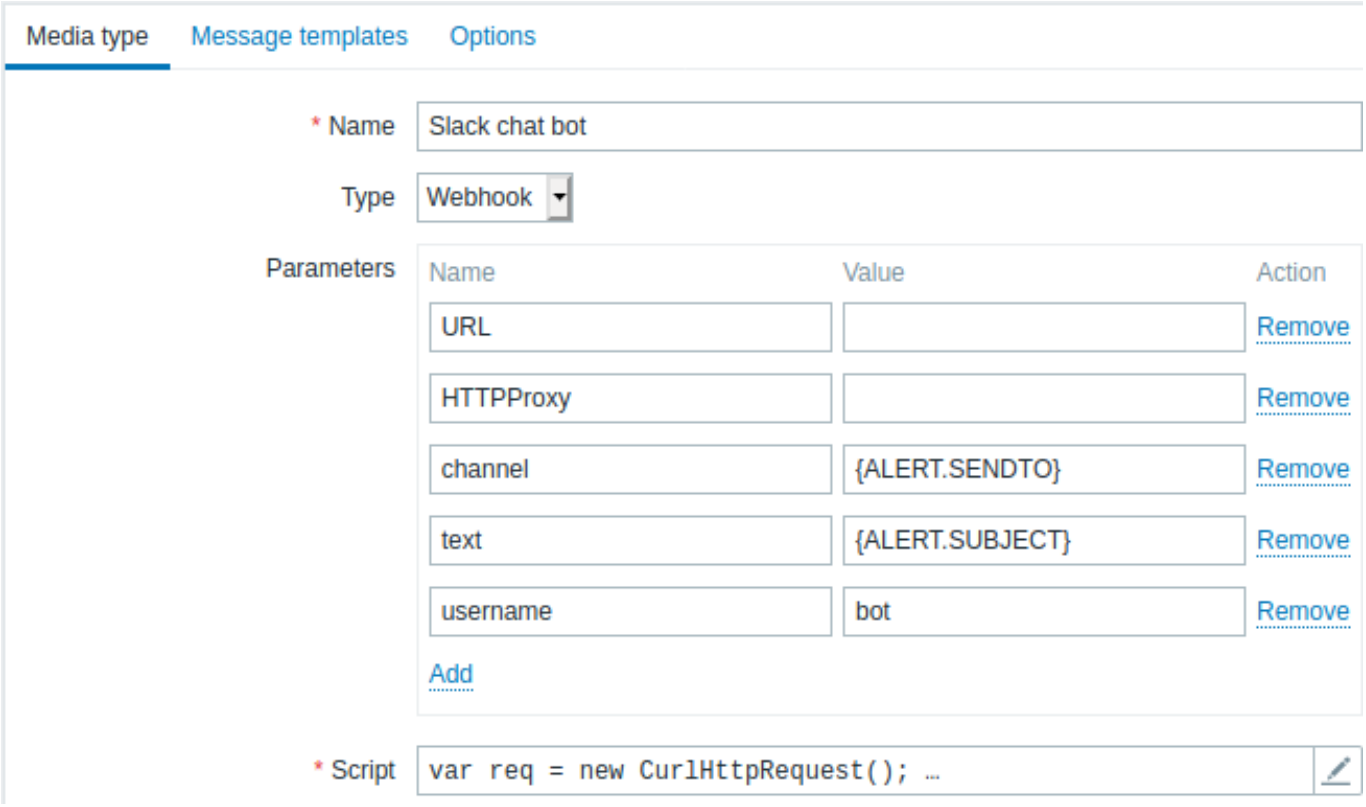
此脚本将创建一个 JIRA 问题，并返回关于所创建问题的一些信息。

```
try {
    Zabbix.Log(127, 'jira webhook script value='+value);
```

```
    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new CurlHttpRequest(),
        proxy = params.HTTPProxy;
        req.SetProxy(proxy);
    fields = {},
    resp;

    req.AddHeader('Content-Type: application/json');
    req.AddHeader('Authorization: Basic '+params.authentication);

    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {"key": params.project_key};
    fields.issuetype = {"id": params.issue_id};
    resp = req.Post('https://tsupport.zabbix.lan/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.Status() != 201) {
        throw 'Response code: '+req.Status();
    }

    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;
} catch (error) {
    Zabbix.Log(127, 'jira issue creation failed json : '+JSON.stringify({"fields": fields}));
    Zabbix.Log(127, 'jira issue creation failed : '+error);

    result = {};
}

return JSON.stringify(result);
```

Slack webhook (自定义)

此 webhook 将把 Zabbix 的通知转发到 Slack 频道中。




```
var req = new CurlHttpRequest();
params = JSON.parse(value);
proxy = params.HTTPProxy;
req.SetProxy(proxy);
req.AddHeader('Content-Type: application/x-www-form-urlencoded');

Zabbix.Log(127, 'webhook request value='+value);

req.Post('https://hooks.slack.com/services/KLFDEI9KNL/ZNA76HGCF/h9MLKJMWoUcEAz8n',
  'payload='+value
);

Zabbix.Log(127, 'response code: '+req.Status());

return JSON.stringify({
  'tags': {
    'delivered': 'slack'
  }
});
```

2 动作

概述

如果您希望对产生的事件进行一些操作（例如发送通知），则需要配置动作。

可以根据所有支持类型的事件来定义动作：

- 触发事件 - 当 trigger 的状态从 正常变为 问题或者从 问题恢复到 正常时
- 发现事件 - 当发生网络发现时
- 自动注册事件 - 当新的活动 agents 自动注册（或已注册主机元数据发生改变）时
- 内部事件 - 当监控项变成不支持状态或触发器进入未知状态时

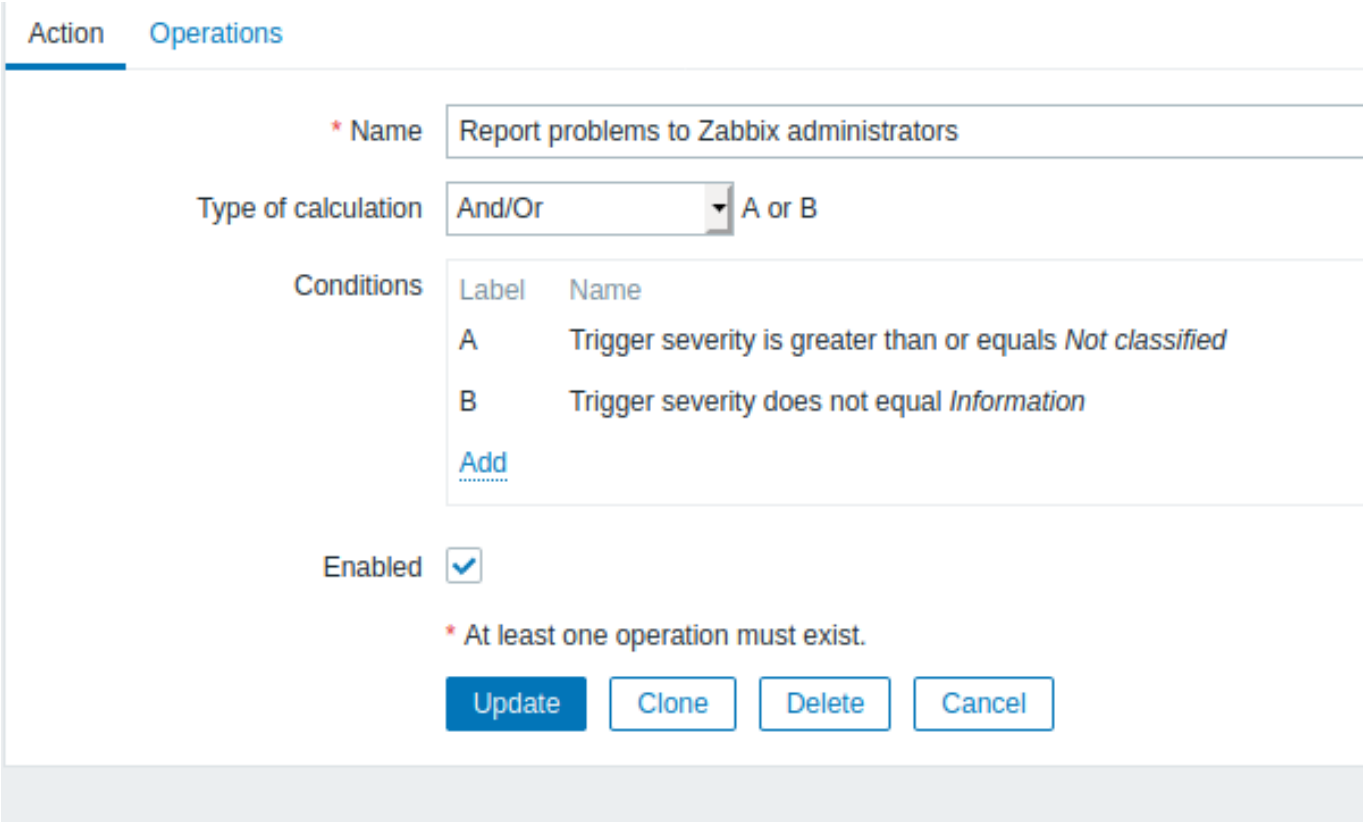
配置动作

配置动作，步骤如下：

- 转到 配置 - > 操作
- 从页面标题下拉菜单中选择所需的动作类型

- 点击 创建动作
- 给动作命名
- 选择执行操作的条件
- 选择要执行的操作

常见动作属性：



带有红色星号的为必填字段。

参数描	
名称唯	的动作名称.

参数	描
计算方式选择值	运算选项 作为动作条件(可以有多个条件)： **And** - 必须满足所有条件 **Or** - 只需满足其中一个条件 **And/Or** - 两者的组合：AND 有不同的条件类型，OR 有相同的条件类型 自定义表达式 - 用户自定义计算公式来进行条件的计算.
条件动	条件的清单。点击添加来新增条件。
已启用选中	复选框以启用该操作。否则将被禁用。

1 条件

概述

此功能定义了只有当事件与定义的条件匹配时才执行动作。条件在配置动作 时进行设置。

条件匹配区分大小写。

触发器动作

可以为基于触发器的动作设置以下条件：

条件类型支持的	作符描述	
Application	等于 指包含 不含	应用集或要排除的应用集。等于 ** - 事件属于链接到指定应用集的监控项的触发器。包含 ** - 事件属于链接到包含该字符串的应用集的监控项的触发器。不

条件类型支持的	作符描述	
Host group	等于 指不等于 **	主机组或要排除的主机组。 于 ** - 属于此主机组的事件。 不等于 - 不属于此主机组的事件。 指定父主机组也隐含选择了所有嵌套的主机组。 要仅指定父组，必须

条件类型支持的	作符描述	
Template	等于 指不等于 **	模板或要排除的模板。于 ** - 属于从此模板继承的触发器的事件不等于 - 不属于从此模板继承的触发器的事件。

条件类型支持的	作符描述	
Host	等于 指不等于 **	主机或要排除的主机于 **-** 属于此主机的事件. 不等于 **-** 不属于此主机的事件。

条件类型支持的	作符描述	
Tag name	等于 指不等于 ** 包含 不含	事件标签或要排除的事件标签。于 ** - 有该标签的事件不等于 ** - 没有该标签的事件包含 ** -标签中包含此字符串的事件不含 - 标签中不包含此字符串的

条件类型支持的	作符描述	
Tag value	等于 指不等于 ** 包含 不含	事件标签和值组合或要排除的标签和值组合于 ** - 有此标签和值的事件不等于 ** -没有此标签和值的事件包含 ** - 标签和值中包含该字符串的事件不含

条件类型支持的	作符描述	
Trigger	等于 指不等于 **	触发器或要排除的触发器。于 ** - 由该触发器生成的事件 **%%** 不等于**%%** - 除此触发器以外，由任何其他触发器生成的事件。

条件类型支持的	作符描述	
Trigger name	包含 在不含 *	发器名称中指定一个字符串或要排除的字符串。包含 ** - 事件由触发器生成，在名称中包含此字符串。不含 - 触发器名称中不包含该字符串。注意:输入的值将

条件类型支持的	作符描述	
Trigger severity	等于 指不等于 大于等于 不小于等于 ** 大	触发严重性。于 ** - 等于触发严重性于 ** - 不等于触发严重性等于 ** - 大于或等于触发严重性小于等于 - 小于或等于触发严重性

条件类型支持的	作符描述	
Time period	在 非在 *	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。 有关格式的说明请参见时间段规范页面。 从 Zabbix 3.4.0 开始，支持用户宏。

条件类型支持的	作符描述	
Problem is suppressed	不是 指是	问题是否因为主机维护而被抑制(不显示)。 *不是 - 不抑制问题。 **是 - 抑制问题.

自动发现动作

可以为基于自动注册的事件设置以下条件：

条件类型支持的	作符描述	
Host IP	等于 指不等于 **	要自动发现的主机的IP地址范围或要排除的IP范围。于 ** -主机IP在该范围内。不等于 - 主机IP不在该范围内。它可能具有以下格式：单个IP：192.168.1.33 IP地址范围：192.168.1-10.1-

条件类型支持的	作符描述	
Service type	等于 指不等于 **	已发现服务的服务类型或者要排除的服务类型。于 ** - 匹配已发现的服务。不等于 - 不匹配已发现的服务。可用的服务类型：SSH, LDAP, SMTP, FTP, HTTP, HTTPS *(自 Zab-bix 2.2 开始，支持*

条件类型支持的	作符描述	
Service port	等于 指不等于 **	已发现服务的TCP端口范围或者要排除的TCP端口范围。于 ** - 服务端口在该范围内。不等于 - 服务端口不在该范围内。

条件类型支持的	作符描述	
Discovery rule	等于 指不等于 **	自动发现规则或要排除的规则。于 ** - 使用此自动发现规则。不等于 - 使用除此规则以外的任何其他的自动发现规则。

条件类型支持的	作符描述	
Discovery check	等于 指不等于 **	自动发现检查或要排除的自动发现检查于 ** -使用此自动发现检查。 不等于 -使用除此以外的其他任何自动发现检查。

条件类型支持的	作符描述	
Discovery object	等于指	发现的对象。等于 - 等于发现的对象(设备或服务)。

条件类型支持的	作符描述	
Discovery status	等于 *	Up** -匹配’Host Up’和’Service Up’的事件 **Down** - 匹配’Host Down’和’Service Down’的事件 **Discovered** - 匹配’Host Dis-cov-ered’和’Service Dis-cov-ered’的事件 **Lost** - 匹配’Host Lost’和’Service Lost’的事件。

条件类型支持的	作符描述	
Uptime/Downtime	大于等于 ’Ho 小于等于 ** 大	t Up’ 和’Service Up’ 事件的运行时间。’Host Down’ 和’Service Down’ 事件的故障时间。等于 ** - 大于或者等于。参数以秒为单位。小于等于 - 小于或等于。参数以秒为单位。

条件类型支持的	作符描述	
Received value	等于 在不等于 大于等于 不小于等于 ** 大包含 不含	现规则中指定从agent(Zabbix, SNMP)检查接收到的值。字符串比较。如果一条规则配置了多个Zab-bix agent或SNMP检查，则将检查每个接收的值(每次检查都会生成一个新事件，该事件

条件类型支持的	作符描述	
Proxy	等于 指不等于 **	proxy或要排除的proxy。于**-使用此代理.不等于-使用除此proxy以外的任何其他proxy。

Note:
自动发现规则中的服务检查，会导致发现事件，但不会同时发生。因此，如果在动作中为 `Service type`, `Service port` 或 `Received value` 条件配了多个值，那么它们将每次与一个自动发现事件进行比较，而 不会同时与多个自动发现事件进行比较。结果，具有多个值的同类型检测的动作，可能无法正确地执行。

可以为基于主动式 agent 自动注册的动作设置以下条件：

条件类型支持的	作符描述	
Host metadata	包含 指不含 匹配 不匹配主机	主机元数据或要排除的主机元数据。包含**-**主机元数据包含该字符串。不包含**-**主机元数据不包该含字符串。数据可以在agent配置文件中设置。匹配-主机元

条件类型支持的	作符描述	
Host name	包含 指不含 匹配 不匹配 **	主机名称或要排除的主机名称。包含 ** - 主机名称包含此字符串。不含 ** - 主机名不包含此字符串。配 ** - 主机名称匹配正则表达式。不匹配 - 主机名称不包

条件类型支持的	作符描述	
Proxy	等于 指不等于 **	proxy 或要排除的 proxy。于 ** - 使用此 proxy。不等于 - 使用除此 proxy 以外的任何其他 proxy.

内部事件动作

可以为基于内部事件的动作设置以下条件:

条件类型支持的	作符描述	
Application	等于 指包含 不含	应用程序或要排除的应用集。等于 ** - 属于链接到指定应用集的监控项的事件。包含 ** - 事件属于一个监控项，该监控项链接到了包含此字符串的应用集。不

条件类型支持的	作符描述
Event type	等于 * 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** - 匹配低级别发现规则从“正常”到“不支持”状态的事件 **Trigger in "unknown" state** - 匹配

条件类型支持的	作符描述	
Host group	等于 指不等于 **	主机组或要排除的主机组。 于 ** - 属于此主机组的事件。 不等于 - 不属于此主机组的事件。

条件类型支持的	作符描述	
Template	等于 指不等于 **	模板或要排除的模板。 于 ** - 属于从此模板继承的监控项/触发器/低级别发现规则的事件。 不等于 - 不属于从此模板继承的监控项/触发器/低级别发现规则的事件。

条件类型支持的	作符描述	
Host	等于 指不等于 **	主机或要排除的主机。于 ** - 属于此主机的事件。不等于 - 不属于此主机的事件。

条件计算类型

计算条件的选项有以下几种：

- **And** - 必须满足所有条件

注意：当多个触发器被选择为 `Trigger =` 条件时，在它们之间是不允许使用"And" 来计算的。只能基于一个触发器的事件执行动作。

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** - 只要满足一个条件就足够了
- **And/Or** - 两者的结合：AND 具有不同的条件类型，而 OR 具有相同的条件类型，例如：

Host group 等于 Oracle servers
Host group 等于 MySQL servers
Trigger name 包含'Database is down'
Trigger name 包含'Database is unavailable'

等价于

(Host group 等于 Oracle servers **or** Host group 等于 MySQL servers**) and (**Trigger name 包含'Database is down' **or** Trigger name 包含'Database is unavailable'**)**

- 自定义表达式 - 用户自定义动作条件的表达式。必须包含所有的条件（以大写字母 A, B, C, ... 表示），可以包含空格、制表符、括号 ()、**and** （区分大小写）、**or** （区分大小写）和 **not** （区分大小写）。

虽然前面关于 `And/Or` 的示例可以表示为 (A or B) and (C or D)，但在自定义表达式中，您还有多种其他的计算方法：

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

((A or B) and C) or D
(not (A or B) and C) or not D
等等

由于删除对象而禁用动作

如果在动作条件/操作中使用的某个对象（主机，模板，触发器等）被删除了，那么条件/操作也会被删除，并且将禁用该动作，以避免错误地执行该动作。用户可以重新启用动作。

当删除以下对象时会发生这种情况：

- 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 “从模板中取消链接” 操作）；
- 触发器（“触发器” 条件）；
- 自动发现规则（使用 “自动发现规则” 和 “自动发现检查” 条件时）。

注意：如果远程命令有多个目标主机，我们删除了其中的一个，那么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，那么操作也将被删除。” 链接到模板” 和” 从模板取消链接” 的操作也是一样。

当删除” 发送消息” 操作中使用的用户或用户组时，动作不会被禁用。

Actions disabled due to deleted objects

If a certain object (host, template, trigger, etc) used in an action condition/operation is deleted, the condition/operation is removed and the action is disabled to avoid incorrect execution of the action. The action can be re-enabled by the user.

This behavior takes place when deleting:

- host groups (”host group” condition, ”remote command” operation on a specific host group);
- hosts (”host” condition, ”remote command” operation on a specific host);
- templates (”template” condition, ”link to template” and ”unlink from template” operations);
- triggers (”trigger” condition);
- discovery rules (when using ”discovery rule” and ”discovery check” conditions).

Note: If a remote command has many target hosts, and we delete one of them, only this host will be removed from the target list, the operation itself will remain. But, if it’s the only host, the operation will be removed, too. The same goes for ”link to template” and ”unlink from template” operations.

Actions are not disabled when deleting a user or user group used in a ”send message” operation.

2 操作

概述

您可以为所有事件定义以下操作：

- 发送信息
- 执行远程命令 (包括 IPMI)

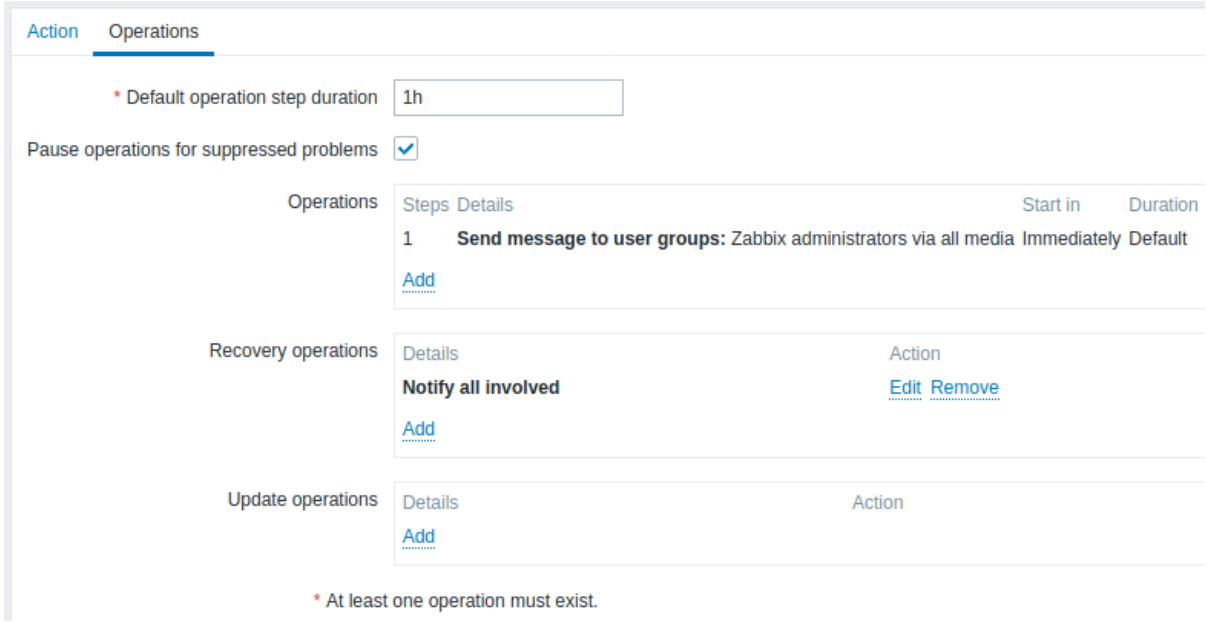
<note important> 如果用户被明确地设置了主机动作和操作的权限为”denied” 或用户根本没有该主机的访问权限，那么 Zabbix server 将不会生成告警。:::

对于自动发现和自动注册事件，还有其他可用操作：

- 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

配置操作

要配置操作，请转到动作配置 中的 操作选项卡。



配置新操作的详细信息，请点击“操作”块中的 Add 。若编辑现有的操作，点击“操作”旁边的 Edit 。将会打开一个弹出窗口，您可以在其中编辑操作步骤的详细信息。

红色星号标记的为必填字段。

常规操作属性：

参数	描
Default operation step duration	一个操作步骤默认持续时间(60 秒到一周)。例如，“1 小时”表示在执行操作时，距离下一步操作还有 1 个小时。从 Zabbix 34.0 开始，支持时间后缀 ，例如 60s, 1m, 2h, 1d。从 Zabbix 3.4.0 开始，支持用户宏 。

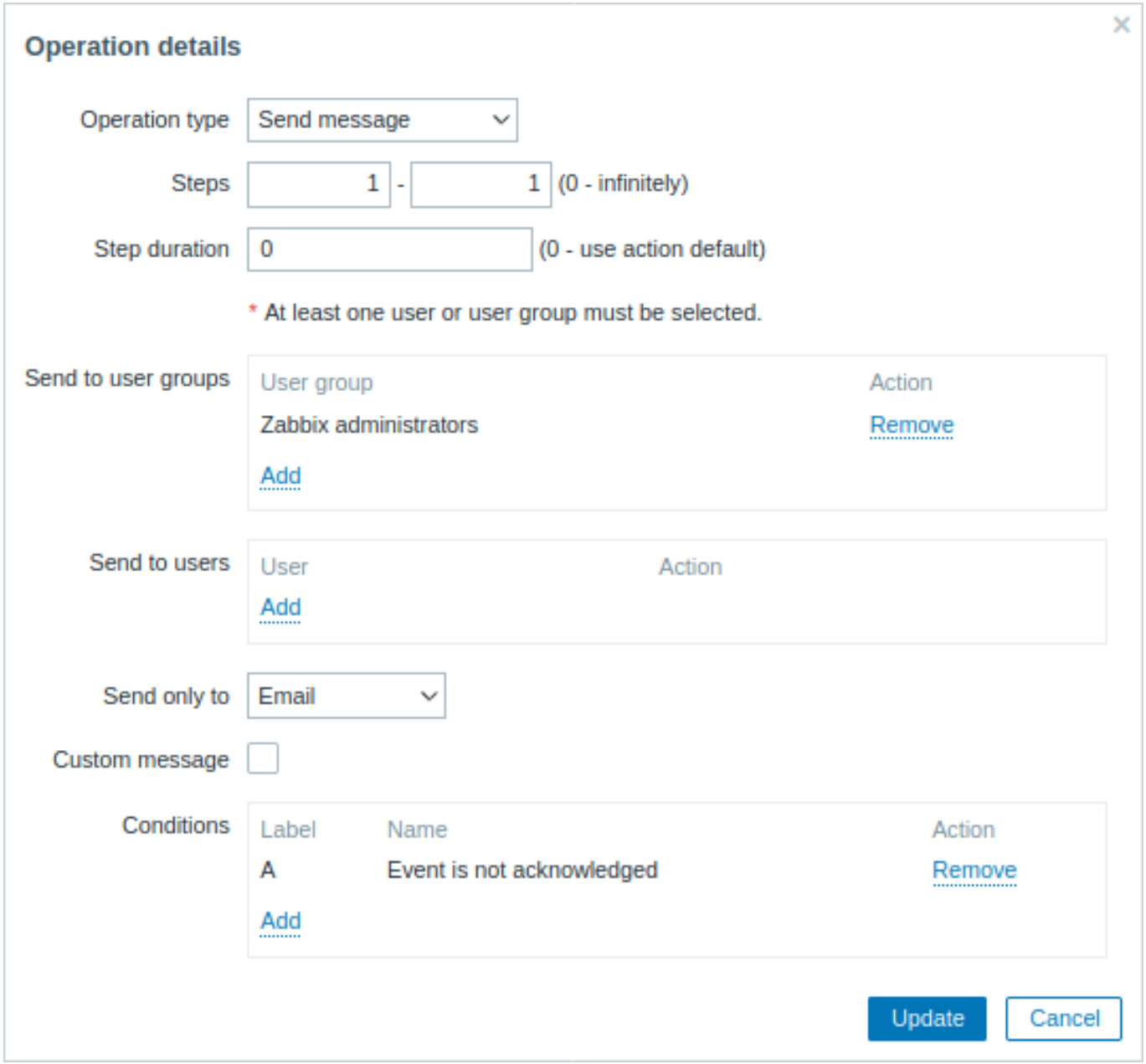
参数	描
Pause operations for suppressed problems	选中此复选框以延长维护期间的操作。当维护结束后开始执行操作时，所有的操作都将执行，包括维护过程中的事件操作。请注意，此设置只影响问题升级；恢复和更新操作不会受到影响。\\如果取消选中此复选框，即使在维护期间，操作也将毫不延迟地执行。Zabbix 3.2.0 之后支持此选项。

参数	描
Operations	显示动作操作(如果有的话)，详细信息如下： **Steps** - 分配给操作的升级步骤 **Details** - 操作的类型及其收件人/目标。 操作列表还显示了通知接收者使用的媒介类型(电子邮件，短信或脚本）以及通知收件人的姓名和姓氏(在别名之后的括号中)。 **Start in** - 事件发生后多久执行操作 **Duration (秒)** - 显示步长。如果步骤使用默认持续时间，则显示默认，如果使用自定义时长，则显示时间。 **Action** - 显示用于编辑和删除操作的链接。

参数	描
Recovery operations	显示动作操作(如果有的话)，详细信息如下： **Details** - 操作的类型及其收件人/目标。操作列表还显示了通知接收者使用的媒介类型(电子邮件，短信或脚本）以及通知收件人的姓名和姓氏(在别名之后的括号中)。 **Action** - 显示用于编辑和删除操作的链接。

参数	描
Update operations	显示动作操作(如果有的话)，详细信息如下： **Details** - 操作的类型及其收件人/目标。 操作列表还显示了通知接收者使用的媒介类型(电子邮件，短信或脚本）以及通知收件人的姓名和姓氏(在别名之后的括号中)。 **Action** - 示用于编辑和删除操作的链接。

Operation details



参数	描
Operation type	所有事件有两种操作类型：**Send message** - 发送消息给用户 **Remote command** - 执行远程命令 更多的操作可用于基于发现和自动注册的事件(见上文)。

参数	描	
Steps		在升级计划表中选择要分配操作的步骤： **From** - 从这个步骤开始执行 **To** - 执行到此步骤（0= 无穷大，执行将不会受到限制） 这些步骤的自定义持续时间（0 = 使用默认步骤持续时间）。 从 Zabbix 3.4.0 开始，支持时间后缀，例如 60s, 1m, 2h, 1d。 从 Zabbix3.4.0 开始，支持用户宏。 可以将多个操作分配给同一个步骤。如果这些操作定义了不同的持续时间，则将考虑最短的持续时间并将其应用于该步骤。
Step duration		
	操作类型：发送消息	

参数	描
	Send to user groups 点击 *Add* 选择要发送消息的用户组。若要收到通知，用户组至少要对主机具有 “读” 权限 。
	Send to users 点击 *Add* 选择要发送消息的用户。若要收到通知，用户至少要对主机具有 “读” 权限。
	Send only to 将消息发送到所有定义的媒介类型或仅发送到选定的媒介类型。
	Custom message 如果选中，则可以配置自定义消息。对于通过webhooks发送的有关内部事件的通知，必须使用自定义消息。
	Subject 自定义消息的主题。主题中可以包含宏。最大长度为 255 个字符。
	Message 自定义的消息。消息内容可以包含宏。具体能在消息中输入多少个字符取决于数据的类型(了解更多信息请参见发送消息)。

参数	描
	操作类 型：远程 命令

参数		描
	Target list	选择要执行命令的目标： **Current host** - 在导致异常事件的触发器所在的主机上执行命令。如果触发器中有多个主机, 则此选项将不起作用。 **Host** - 选择要在其上执行命令的主机。 **Host group** - 选择需要执行该命令的主机组。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，即使该主机被多次匹配(例如来自多个主机组，单台主机和从主机组中匹配)。 如果在 Zabbix server 上执行了自定义脚本，那么目标列表是没有意义的。在这种情况下选择更多目标只会导致脚本在服务器上执行更多次。

参数		描
	Type	选择命令类型： **IPMI** - 执行IPMI 命令 **Custom script** - 执行自定义命令集 **SSH** - 执行 SSH 命令 **Telnet** - 执行 Telnet 命令 **Global script** - 执行在管理 → 脚本中定义的全局脚本之一。

参数		描
	Execute on	在以下位置执行自定义脚本： **Zabbix agent** - 该脚本将由主机上的 Zabbix agent 执行 **Zabbix server (proxy)** - 该脚本将由 Zabbix server 或 proxy 执行——这取决于主机是由 server 监控还是由 proxy 监控的 **Zabbix server** - 该脚本仅由 Zabbix server 执行 要在 agent 上执行脚本，必须允许 system.run 监控项。 要在 proxy 上执行脚本，必须对其进行配置（开启 *EnableRemoteCommands* 参数），以允许从服务器远程执行命令。 如果 类型是“自定义脚本”，则该字段可用。

参数	描	
	Commands	输入命令。 所支持的宏将根据导致事件的触发表达式进行解析。例如，主机宏将解析为触发器表达式的主机（而不是目标列表的主机）。
Conditions		执行操作的条件： **Not ack** - 仅当事件未被确认时 **Ack** - 仅当事件被确认时。

完成后，点击 *Add* 将所有操作添加到 操作列表中。

1 发送消息

概述

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。

配置

为了能够发送和接收 Zabbix 的通知，您必须：

- 定义媒介 来发送消息

<note warning> 如果您想要接收如发现、agent 自动注册或内部事件等非触发类的事件通知，那么在用户媒介配置 中 必须勾选默认的触发器级别 (’ 未分类’) 。:::

- 配置动作操作 向一个已定义的媒介发送消息

<note important>Zabbix 仅向那些至少对生成事件的主机具有 “读” 权限的用户发送通知。该用户须可访问至少一台配置了触发器表达式的主机。:::

您可以配置使用升级 发送消息的自定义场景。

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）。消息的主题和正文是 base64 编码，遵循 “SMTP/MIME 电子邮件” 格式标准。

所有宏展开后的消息限制与远程命令 的消息限制相同。

跟踪消息

您可以在 监测 - > 问题中查看发送的消息的状态。

在 动作那一列您可以看到有关所采取动作的汇总信息。绿色的数字表示发送的消息，红色的则表示失败的消息。进行中表示动作已经启动。失败通知没有成功执行任何操作。

单击事件时间，可以查看事件详情，在 消息动作块中您也可以看到由于事件而发送（或未发送）的消息的详细信息。

在 报表 → 动作日志您将看到对已配置了操作的事件所采取的所有动作的详细信息。

2 远程命令

概述

使用远程命令，您可以定义在某些条件下，在监视的主机上自动执行某个预定义的命令。

因此，远程命令是一种强大的智能主动监控机制。

在功能最明显的用途中，您可以尝试：

- 自动重启某些没有响应的应用程序 (web 服务器、中间件、CRM 等)
- 使用 IPMI“reboot” 命令，重启那些不响应请求的远程服务器
- 自动释放空间不足的磁盘 (删除旧文件，清理/tmp 等)
- 根据 CPU 的负载情况，将虚拟机从一个物理机迁移到另一个物理机上
- 在 CPU(磁盘、内存等) 资源不足的情况下，向云环境添加新的节点

配置远程命令的操作类似于发送消息，唯一的区别是 Zabbix 将执行命令而不是发送消息。

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。但在 Zabbix agent 和 Zabbix proxy 上，远程命令默认是不开启的，它们可以通过以下方式启用：

- 在 agent 配置中添加 `AllowKey=system.run[*]` 参数；
- 在 proxy 配置中，将 enableremotecomcommands 参数设置为 “1”（Zabbix 5.0.2 之前版本，agent 配置也要更改）

Zabbix server 执行的远程命令按照命令执行 (包括退出代码检查) 中描述的方式运行。

即使目标主机处于维护状态，也会执行远程命令。

远程命令限制

在所有宏解析后，远程命令的限制取决于数据库和字符集的类型（非 ASCII 字符需要存储一个以上的字节）：

数据库 //	符限制 // //字节限制	//
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	无限制
SQLite (only Zabbix proxy)	65535	无限制

以下教程就有关如何设置远程命令进行了逐步说明。

配置

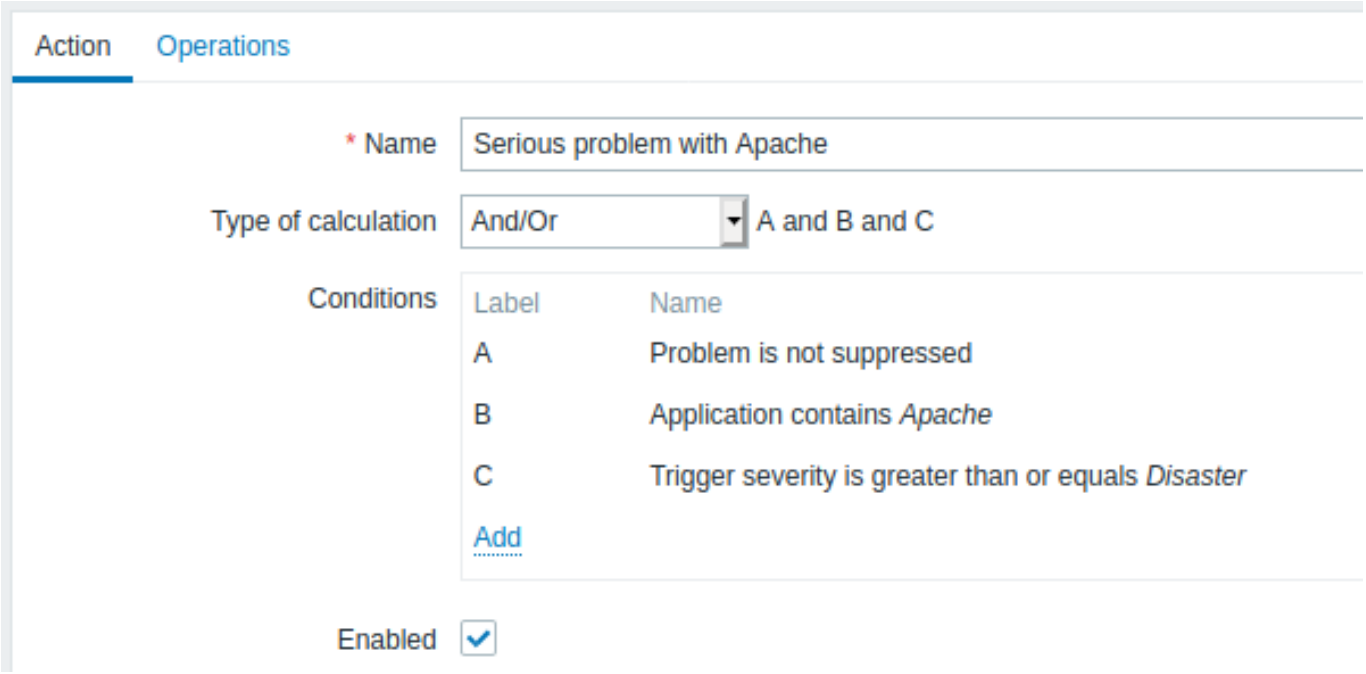
首先，必须在 agent 的配置 中启用那些在 Zabbix agent（自定义脚本）上执行的远程命令。

确保已经添加了 `AllowKey=system.run[*]` 参数。如果修改了此参数，请重启 agent 服务。

Attention:
远程命令不适用于主动模式的 Zabbix agent。

然后，在 配置 → 动作中配置一个新动作时:

- 定义适当的条件。如本例中，设置为在某一个 Apache 应用程序出现任何灾难问题时激活该动作：



标记红色星号的为必填字段。

- 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）
- 如果选择了 自定义脚本类型，则需要选择自定义脚本的执行方式（通过 Zabbix agent、Zabbix server（proxy）或仅 Zabbix server）
- 输入远程命令

例如：

```
sudo /etc/init.d/apache restart
```

在这种情况下，Zabbix 将会尝试重启 Apache 进程。使用此命令，确保该命令在 Zabbix agent 上执行 (在 执行在块中，点击 *Zabbix agent* 按钮)。

Attention:
请注意 **sudo** 的用法 - 默认情况下，Zabbix 用户没有重启系统服务的权限。有关如何配置 **sudo**，请参见下文。

Note:
Zabbix agent 应该在远程主机上运行并接受传入的连接。Zabbix agent 在后台执行命令。

通过 system.run[,nowait] 键，在 Zabbix agent 上执行远程命令时则不会超时，并且不会检查执行的结果。在 Zabbix server 和 Zabbix proxy 上，远程命令的超时时间是在 zabbix_server.conf 或 zabbix_proxy.conf 文件的 TrapperTimeout 参数中设置的，并且会检查执行结果。

访问权限

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: 确保'zabbix' 用户对已配置的命令具有执行权限。可能有人更乐于使用 ** sudo ** 来访问特权命令。要配置访问权限，请以 root 身份执行：

```
# visudo
```

可以在 *sudoers* 文件中使用以下的示例代码:

```
# 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:
在某些系统中，*sudoers* 文件将阻止非本地用户执行命令。要更改此设置，请在 */etc/sudoers* 文件中注释 **requiretty** 选项。

具有多个接口的远程命令

如果目标系统具有多个选定类型的接口（Zabbix agent 或 IPMI），则将在默认接口上执行远程命令。

可以使用除 zabbix agent 以外的其他接口，通过 SSH 和 Telnet 执行远程命令。可用的接口按以下顺序选择:

```
* Zabbix agent default interface
* SNMP default interface
* JMX default interface
* IPMI default interface
```

IPMI 远程命令

IPMI 远程命令的语法如下：

```
<command> [<value>]
```

其中

- <command> - IPMI 命令，没有空格
- <value> - 'on', 'off' 或任何无符号整数。<value> 是可选参数。

示例

示例 1

在一定条件下重启 Windows。

为了在 Zabbix 检测到问题时自动重启 Windows，定义了以下操作：

参数描	
Operation type	'Remote command'
Type	'Custom script'
Command	c:\windows\system32\shutdown.exe -r -f

示例 2

通过 IPMI 控制重启主机。

参数描	
Operation type	'Remote command'
Type	'IPMI'
Command	reset

示例 3

通过 IPMI 控制关闭主机电源。

参数描	
Operation type	'Remote command'
Type	'IPMI'
Command	power off

3 附加操作

概述

在本章中，您可以找到发现/自动注册事件的附加操作 的一些详细信息。

添加主机

添加主机是在发现的过程中完成的，而不是在发现过程结束时。

<note tip> 由于一些主机/服务不可用，网络发现可能会花费一些时间。因此，建议您耐心等待并使用合理的 IP 范围。:::

添加主机时，其名称由标准 **gethostbyname** 函数决定。如果可以解析主机，则使用解析的名称。如果不能，则使用 IP 地址。此外，若必须要用 IPv6 地址作为主机名，则所有 “：”（冒号）将替换为 “_”（下划线），因为主机名中不允许有冒号。

Attention:
如果通过 agent 执行发现，当前主机名查找仍然在 Zabbix server 上进行。

<note important> 如果 Zabbix 配置中已经存在一个主机，且其名称与新发现的主机相同，那么 Zabbix 1.8 之前的版本将添加另一个同名的主机。Zabbix 1.8.1 及更高版本将会在主机名中添加 **_N**，其中 **N** 是从 2 开始的递增数字。:::

4 在信息中使用宏

概述

在消息主题和消息文本中，可以使用宏来更有效地报告问题。

提供了 Zabbix 支持的宏的完整列表，可供参阅。

示例

此处的示例说明了如何在消息中使用宏。

示例 1

消息主题:

```
Problem: {TRIGGER.NAME}
```

当收到消息时，消息主题会被替换为：

```
Problem: Processor load is too high on Zabbix server
```

示例 2

消息内容:

```
Processor load is: {zabbix.zabbix.com:system.cpu.load[,avg1].last()}
```

当收到消息后，消息将被替换为：

```
Processor load is: 1.45
```

示例 3

消息内容:

```
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)}
```

当收到消息时，消息将被替换为：

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

示例 4

消息内容：

```
http://<server_ip_or_name>/zabbix/tr_events.php?triggerid={TRIGGER.ID}&eventid={EVENT.ID}
```

当收到消息时，它将包含一个指向 事件细节页面的链接，该页面提供有关事件、其触发器的信息以及由同一触发器生成的最新事件列表。

示例 5

在一个触发器表达式中通知来自多个主机的值。

消息内容:

```
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})
```

当收到消息时，消息将被替换为:

```
Problem name: Processor load is too high on a local host
Trigger expression: {Myhost:system.cpu.load[percpu,avg1].last()}>5 or {Myotherhost:system.cpu.load[percpu,

1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))
```

示例 6

在恢复 消息中接收问题事件和恢复事件的详细信息:

消息内容:

```
Problem:

Event ID: {EVENT.ID}
Event value: {EVENT.VALUE}
Event status: {EVENT.STATUS}
Event time: {EVENT.TIME}
Event date: {EVENT.DATE}
Event age: {EVENT.AGE}
Event acknowledgment: {EVENT.ACK.STATUS}
Event update history: {EVENT.UPDATE.HISTORY}

Recovery:

Event ID: {EVENT.RECOVERY.ID}
Event value: {EVENT.RECOVERY.VALUE}
Event status: {EVENT.RECOVERY.STATUS}
Event time: {EVENT.RECOVERY.TIME}
Event date: {EVENT.RECOVERY.DATE}
Operational data: {EVENT.OPDATA}
```

当收到消息时，这些宏将被替换为：

```
Problem:

Event ID: 21874
Event value: 1
Event status: PROBLEM
Event time: 13:04:30
Event date: 2018.01.02
Event age: 5m
Event acknowledgment: Yes
Event update history: 2018.01.02 13:05:51 "John Smith (Admin)"
Actions: acknowledged.

Recovery:

Event ID: 21896
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2018.01.02
Operational data: Current value is 0.83
```

Attention:
从 Zabbix 2.2.0 开始，支持把原始问题事件和恢复事件使用的通知宏分离开。

3 恢复操作

概述

恢复操作允许您在问题解决时收到通知。

恢复操作支持消息和远程命令。虽然可以添加一些操作，但不支持升级操作 - 所有操作都被分配到了一个单独的步骤，因此将同时执行。

使用场景

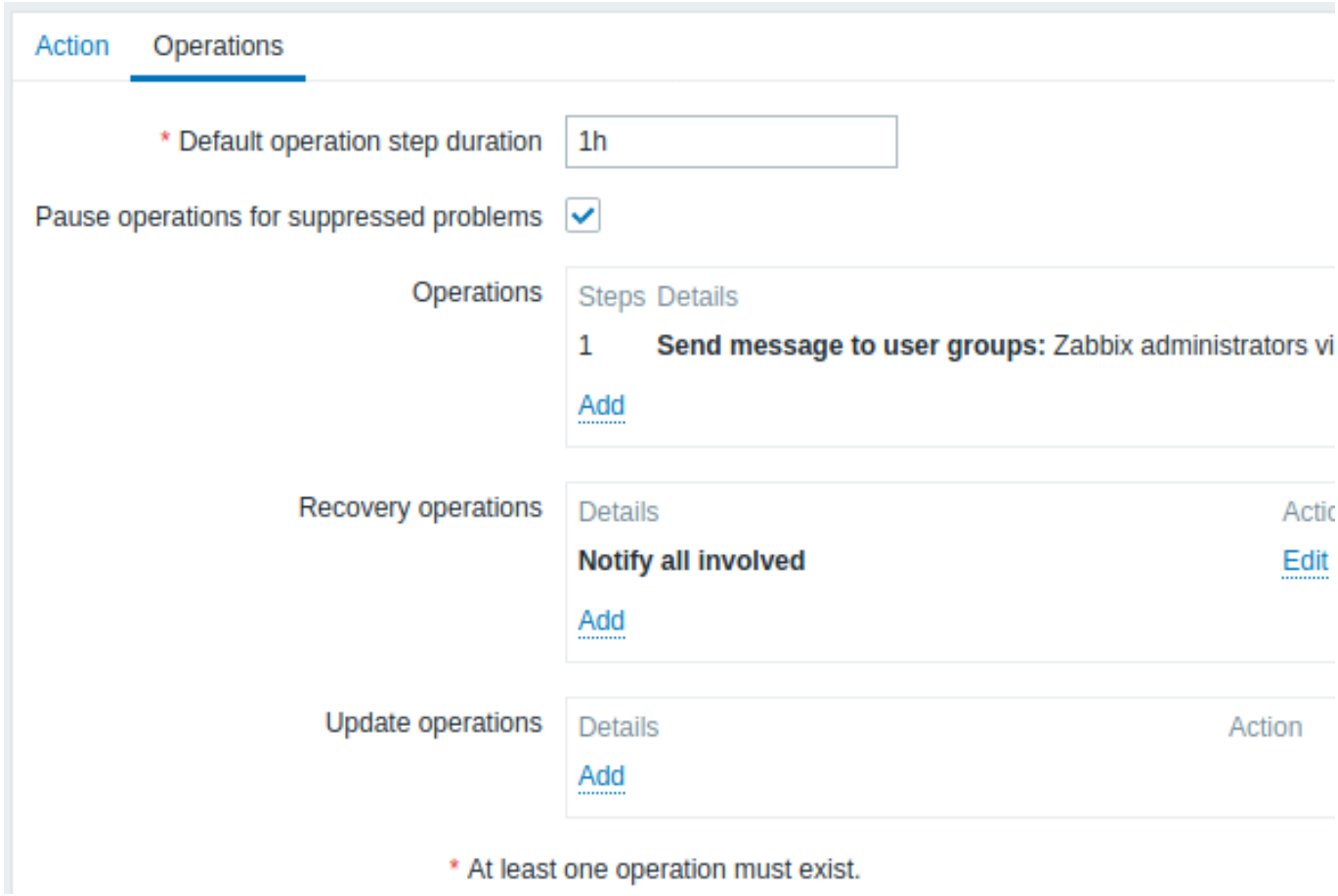
恢复操作的一些用例如下：

1. 通知所有之前已经收到该问题通知的用户

    * 选择'发送恢复消息'作为操作类型
- 恢复时有多个操作：发送通知和执行远程命令
    * 添加发送消息和执行命令的操作类型
- 在外部帮助台/工单系统中建立一个工单，并在问题解决后将其关闭
    * 创建一个与帮助台系统通信的外部脚本
    * 创建一个具有执行此脚本的操作的动作，从而生成一个工单
    * 进行恢复操作，使用其他参数执行此脚本并关闭工单
    * 使用 {EVENT.ID} 宏来引用原始问题

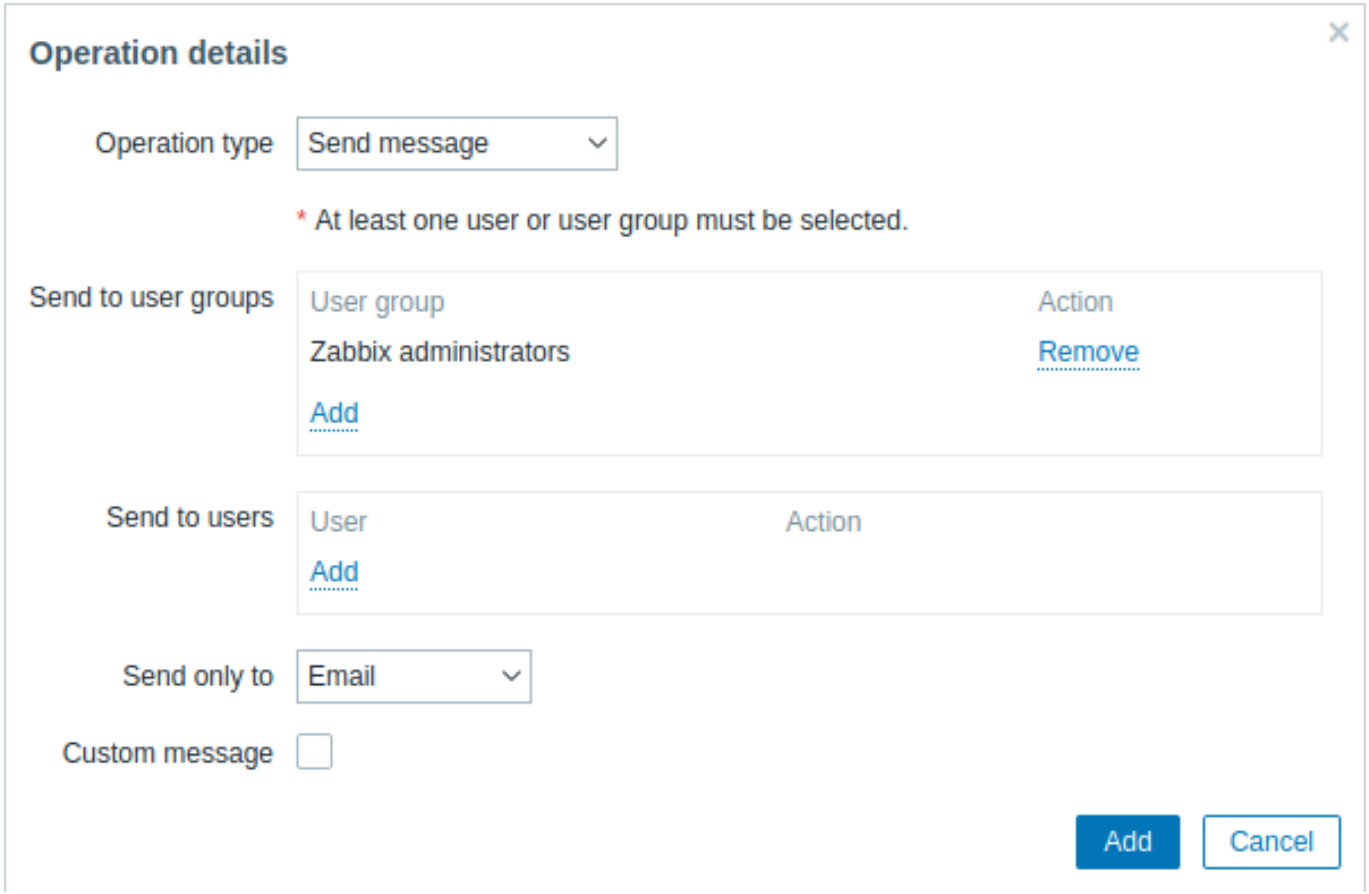
配置恢复操作

配置恢复操作，请前往动作 配置中的 操作选项卡。



要配置新恢复操作的详细信息，请在恢复操作块中点击 Add。要编辑现有的操作，点击操作旁边的 Edit。将会打开一个弹出窗口，您可以在其中编辑操作步骤的详细信息。

恢复操作细节



参数			描
操作类型		复事	有三种操作类型： **Send message** - 发送恢复消息给指定用户 **Remote command** - 执行远程命令 **Notify all involved** - 向所有收到问题事件通知的用户发送恢复消息 注意：如果在多个操作类型中定义了相同的收件人，但没有指定 自定义消息，则不会发送重复的通知。
	操作类型: 发送消息		
	Send to user groups		点击 添加选择要发送恢复消息的用户组。 若要收到通知，用户组至少要对主机具有 “读” 权限。
	Send to users		点击 添加选择要发送恢复消息的用户。 若要收到通知，用户至少要对主机具有 “读” 权限。
	Send only to		将默认恢复消息发送到所有定义的媒介类型或仅发送到选定的媒介类型。
	Custom message		如果选中，则可以配置自定义消息。
	Subject		自定义消息的主题。主题中可以包含宏。

参数		描
	Message	自定义的消息。消息内容中可以包含宏。
	操作类型：远程命令	
	Target list	选择要执行命令的目标： **Current host** - 在导致异常事件的触发器所在的主机上执行命令。如果触发器中有多个主机，则此选项将不起作用。 **Host** - 选择要在其上执行命令的主机。 **Host group** - 选择需要执行该命令的主机组。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。 主机上的命令只执行一次，即使该主机被多次匹配（例如来自多个主机组，单台主机和从主机组中匹配）。 如果在 Zabbix server 上执行命令，那么目标列表是没有意义的。在这种情况下，选择更多目标只会导致命令在服务器上执行更多次。 注意：对于全局脚本，目标选择也取决于全局脚本配置 中 主机组的设置。
	Type	选择命令类型： **IPMI** - 执行IPMI 命令 **Custom script** - 执行自定义命令集 **SSH** - 执行 SSH 命令 **Telnet** - 执行 Telnet 命令 **Global script** - 执行在管理 → 脚本中定义的全局脚本之一。
	Execute on	在以下位置执行自定义脚本： **Zabbix agent** - 该脚本将由主机上的 Zabbix agent 执行 **Zabbix server (proxy)** - 该脚本将由 Zabbix server 或 proxy 执行——这取决于主机是由 server 监控还是由 proxy 监控的 **Zabbix server** - 该脚本仅由 Zabbix server 执行 要在 agent 上执行脚本，必须将 agent 配置 为允许来自服务器的远程命令。 如果 类型是’ 自定义脚本’，则该字段可用。

参数		描
	Commands	输入命令。 所支持的宏将根据导致事件的触发表达式进行解析。例如，主机宏将解析为触发器表达式的主机（而不是目标列表的主机）。
	操作类型: 通知所有参与者	
	Custom message	如果选中，则可以配置自定义消息。
	Subject	自定义消息的主题。主题中可以包含宏。
	Message	自定义的消息。消息内容中可以包含宏。

红色星号标记的为必填字段。完成后，点击 添加将操作添加到 恢复操作列表中。

4 Update operations

Overview

Update operations allow you to be notified when problems are updated by other users, i.e.:

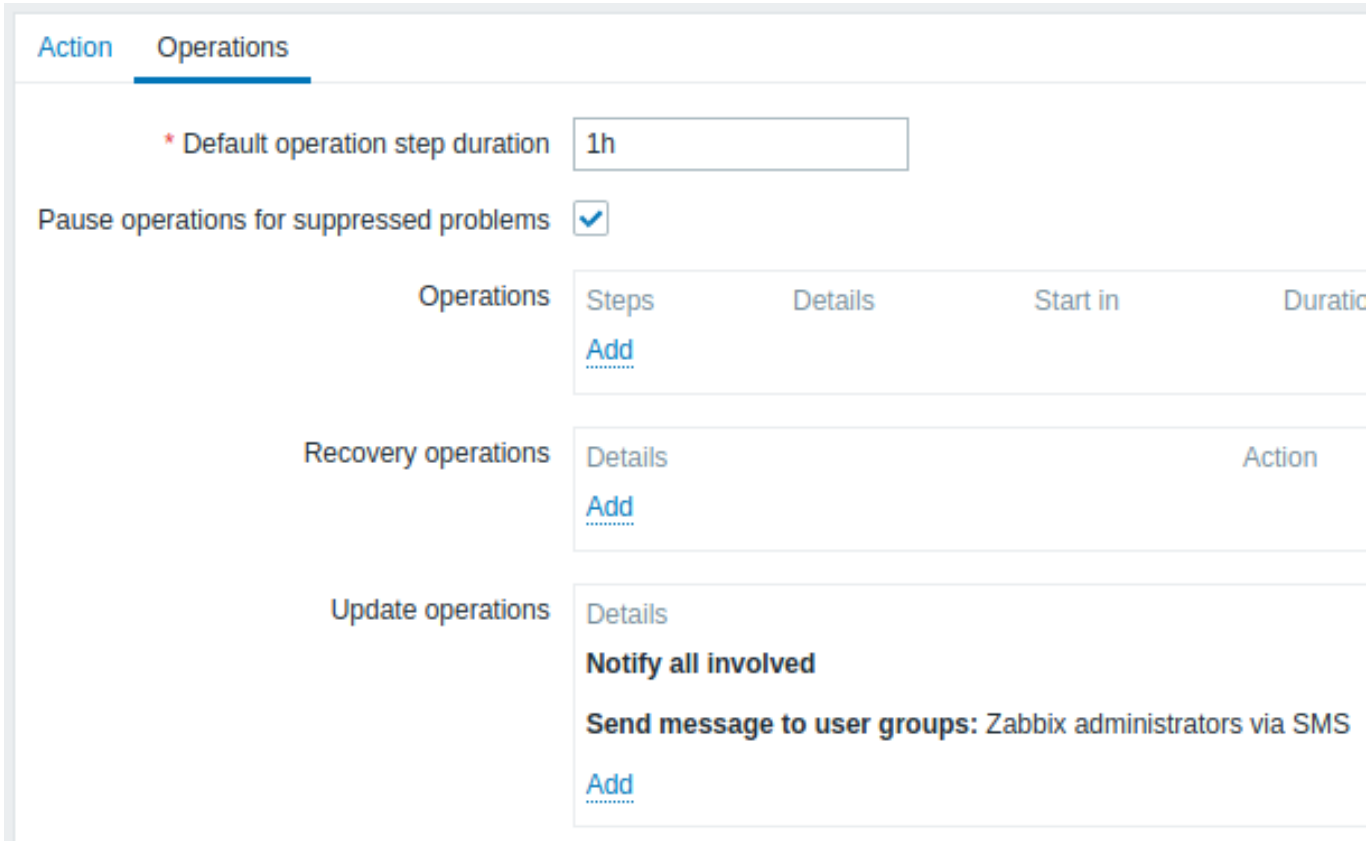
- commented upon
- acknowledged
- severity changed
- closed (manually)

Update operations are available in actions with the event source as *Triggers*.

Both messages and remote commands are supported in update operations. While several operations can be added, escalation is not supported - all operations are assigned to a single step and therefore will be performed simultaneously.

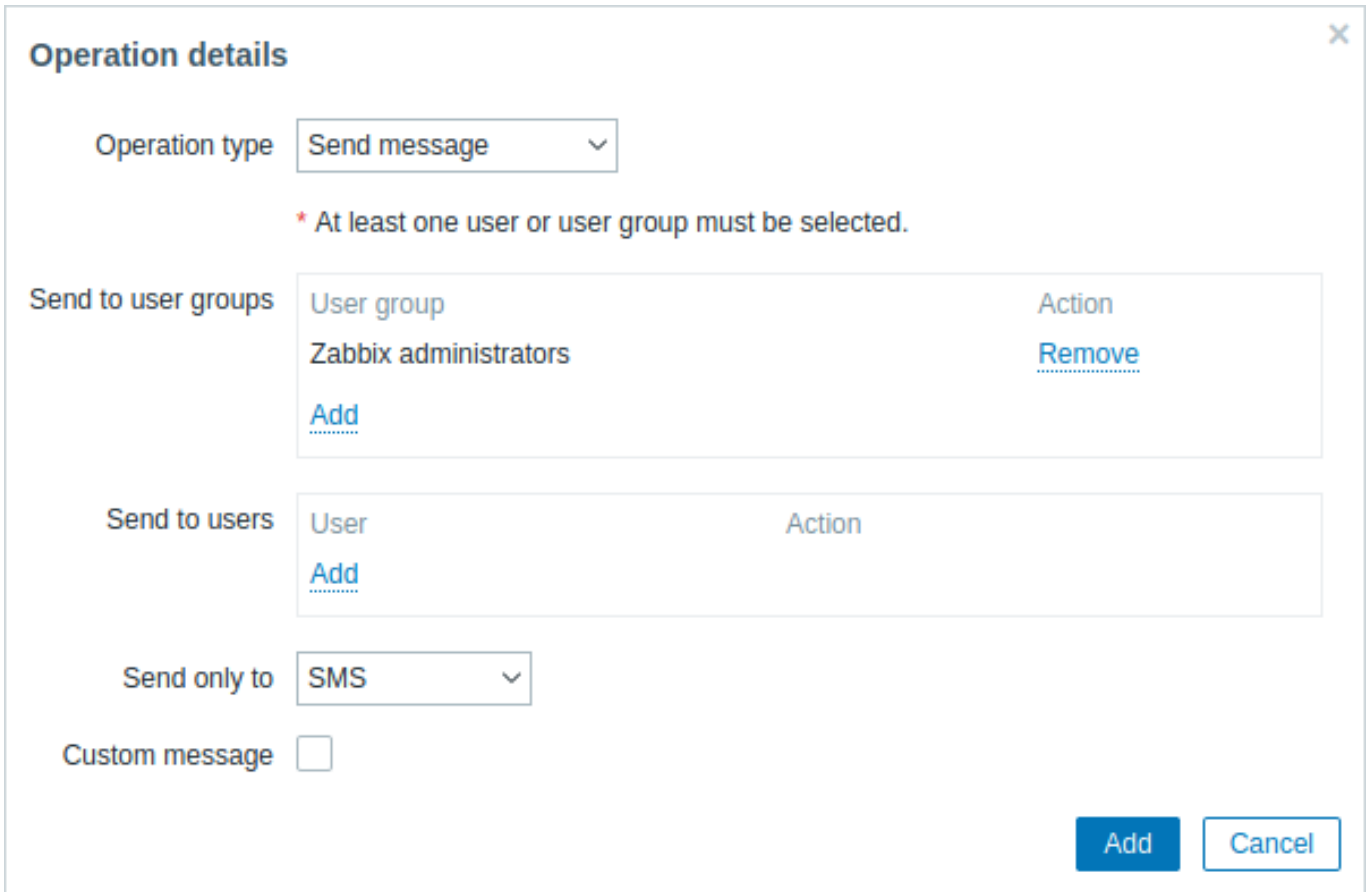
Configuring an update operation

To configure an update operation go to the *Operations* tab in action configuration.



To configure details of a new update operation, click on Add in the Update operations block. To edit an existing operation, click on Edit next to the operation. A popup window will open where you can edit the operation step details.

Update operation details



Operation type	Three operation types are available for update operations: **Send message** - send update message to specified user when event is updated, for example, acknowledged **Remote command** - execute a remote command when event is updated, for example, acknowledged **Notify all involved** - send notification message to all users who received notification about the problem appearing and/or have updated the problem event. If the same recipient with unchanged default subject/message is defined in several operation types, duplicate notifications are not sent. The person who updates a problem does not receive notification about their own update.
Operation type: send message	
Send to user groups	Click on *Add* to select user groups to send the update message to. The user group must have at least ”read” permissions to the host in order to be notified.
Send to users	Click on *Add* to select users to send the update message to. The user must have at least ”read” permissions to the host in order to be notified.
Send only to	Send update message to all defined media types or a selected one only.

Custom message	If selected, the custom message can be defined.
Subject	Subject of the custom message. The subject may contain macros.
Message	The custom message. The message may contain macros.
Operation type: remote command	

Target list	Select targets to execute the command on: **Current host** - command is executed on the host of the trigger that caused the problem event. This option will not work if there are multiple hosts in the trigger. **Host** - select host(s) to execute the command on. **Host group** - select host group(s) to execute the command on. Specifying a parent host group implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups. A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). The target list is meaningless if the command is executed on Zabbix server. Selecting more targets in this case only results in the command being executed on the server more times. Note that for global scripts, the target selection also depends on the *Host group* setting in global script configuration.

Type	Select the command type: **IPMI** - execute an IPMI command **Custom script** - execute a custom set of commands **SSH** - execute an SSH command **Telnet** - execute a Telnet command **Global script** - execute one of the global scripts defined in *Administration→Scripts*.
Execute on	Execute a custom script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as *Type*.
Commands	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
Operation type: notify all involved	

	Default media type	Users who update a problem but have not received notifications about the problem appearing will receive notifications about further updates on the selected default media type - Email or SMS. This field is available since Zabbix 3.4.2.
	Custom message	If selected, the custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.
	Message	The custom message. The message may contain macros.

All mandatory input fields are marked with a red asterisk. When done, click on *Add* to add operation to the list of *Update operations*.

5 通知升级

概述

通过升级，您可以创建发送通知或执行远程命令的自定义场景。

实际应用上，升级可以实现：

- 用户可以立即收到新问题通知
- 可以重复通知，直到问题解决
- 可以延时发送通知
- 通知可以升级到另一个 “更高级别” 的用户组
- 可以立即执行远程命令，或者当问题长时间没有得到解决时

操作会根据 升级步骤进行通知升级。每一步都有一个持续时间。

您可以定义单个步骤的默认持续时间和自定义持续时间。一个升级步骤的最短持续时间为 60 秒。

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。第一步是立即执行动作。如果您想延迟某个动作，可以将其分配给后面的步骤。每个步骤可以定义多个动作。

升级步骤的数量不受限制。

在配置操作 时，可以对升级进行定义。只支持对问题操作进行升级，而不支持恢复操作。

升级的其他方面

让我们考虑一下，如果一个操作包含几个升级步骤，那么在不同的情况下会发生什么。

情景行
在发送初始问题通知之后，有问题的主机将进入维护状态取决于动作配置(/zh/manual/config/notification中暂停操作以制止问题的设置，所有剩余的升级步骤会因维护周期而延迟执行，或者不延迟执行。维护期不会取消操作。

情景行

在 时间段 动作条件中定义的时间段在发送初始通知后结束所有剩余的升级步骤都将执行。时间段条件不能

止操作；它对于何时启动/未启动动作有效果，而不是操作。

情景行	
在维护过程中出现问题，并在维护结束后继续（未解决）取决于动作配置(/zh/manual/config/notifications/action)中暂停操作以制止问题的设置，所有升级步骤可以从维护结束的那一刻开始执行，也可立即执行。	
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，必须等待触发器的触发。	

情景行
不同的升级紧随其后并重叠每个新的升级都将取代之

的升级，但是至少一个升级步骤总是在上一个升级中执行。此行为与触发器的每个问题评估所创建的事件的操作相关。

情景行	
在升级过程中（如正在发送的消息），基于任何类型的事件： 发送进行中的消息，然后再发送另一条关于升级的消息。后- 操作被禁用 <*br*> 基于触发器的事件：<*br*>- 触发器被禁用 <*br*>- 主机或监控项被禁用 <*br*> 基于关于触发器的内部事件：<*br*>- 触发器被禁用 <*br*> 基于关于监控项/低级别发现规则的内部事件：<*br*>- 监控项被禁用 <*br*>- 主机被禁用	消息将在消息正文的开头显示取消文本(注意：升级已取消)并说明原因(例如，注意：取消升级：动作'<动作名称>'已禁用)。通过这种方式，收件人被告知升级被取消，不

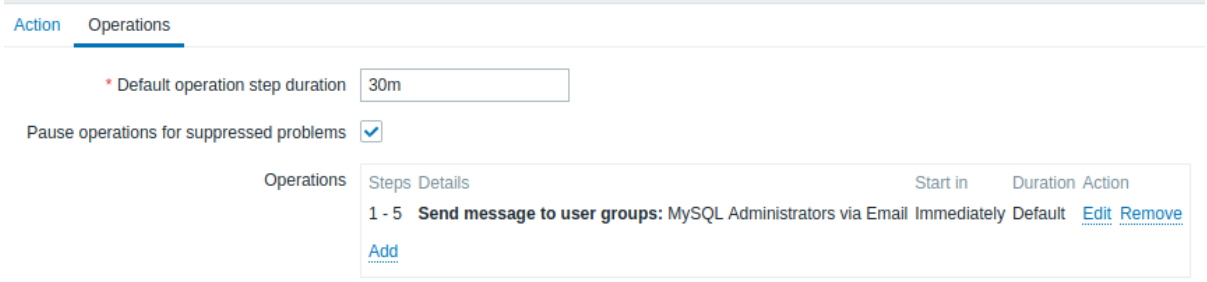
情景行	
在升级过程中（如发送消息），该操作被删除不再发送任何消息。信息将记录到服务器日	文件(从Debug Level 3=Warning)开始，例如：`escalation cancelled: action id:334 deleted`

升级示例

示例 1

每 30 分钟向 ‘MySQL Administrators’ 组发送一次重复的通知（共 5 次）。配置如下：

- 在操作选项卡中，将 *Default operation step duration* 设置为’30m’（即，30 分钟）
- 将升级步骤设置为 从 ‘1’ 到 ‘5’
- 选择’MySQL Administrators’ 组作为消息的接收者



通知将分别在问题发生后的第一时间（0:00）以及 30 分钟 (0:30)、1 小时 (1:00)、1 个半小时 (1:30) 和 2 个小时 (2:00) 后发送 (当然，除非问题提前解决)

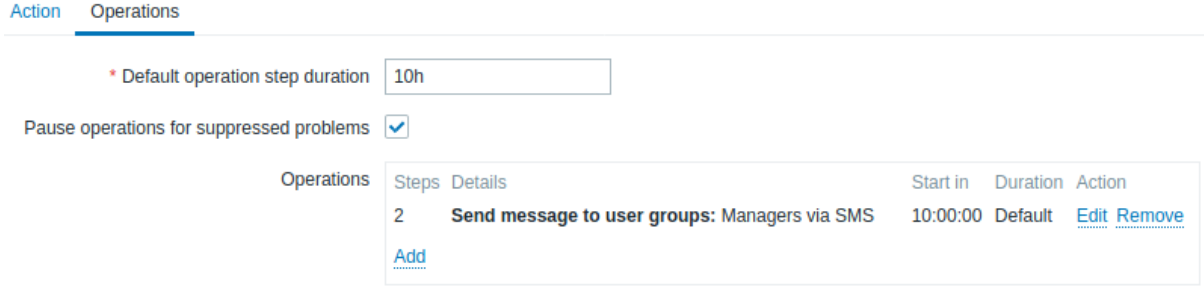
如果问题解决了并且配置了恢复消息，那么恢复消息将发送给之前在此升级场景中至少收到一条问题消息的人。

Note:
如果触发器已生成了一个激活状态的升级，该触发器被禁用了，那么 Zabbix 会向所有已经收到通知的人发送一条关于该升级信息的消息。

示例 2

发送关于一个长期存在的问题的延迟通知：

- 在操作选项卡中，将 *Default operation step duration* 设置为’10h’（即，10 小时）
- 将升级步骤设置为 从 ‘2’ 到 ‘2’



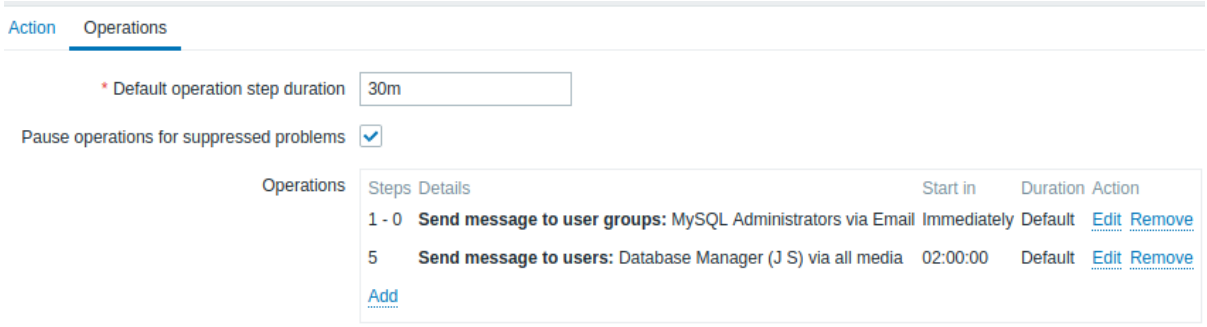
通知只会在升级场景的第 2 步发送，或者在问题发生 10 小时后发送

您可以自定义消息文本，如 “该问题已超过 10 小时了”。

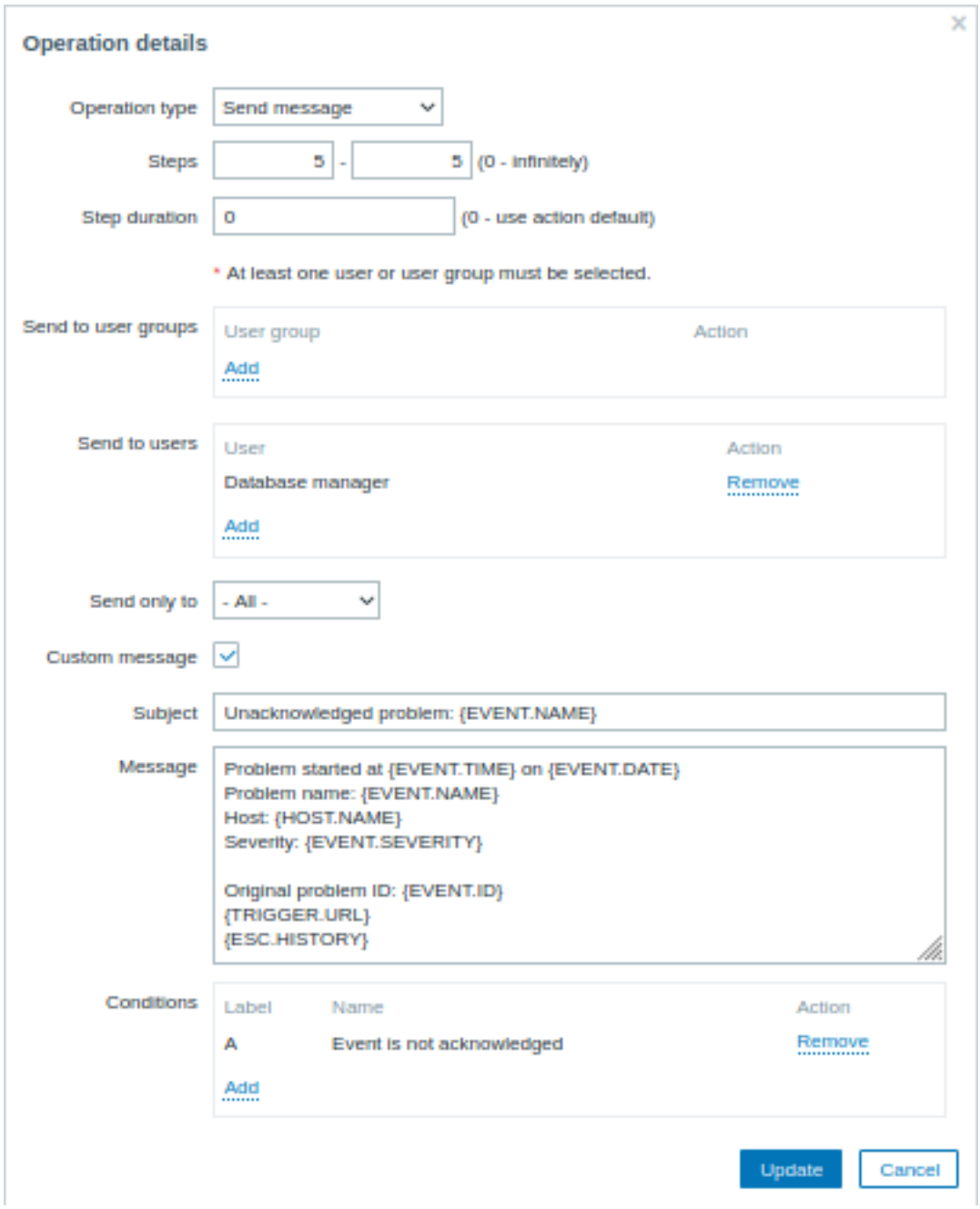
示例 3

将问题升级到老板那里。

在上面的第一个示例中，我们配置了定期向 MySQL 管理员发送消息。在这种情况下，在问题升级到数据库经理那之前，数据库管理员们将收到四条消息。注意：只有在问题尚未被确认的情况下（假设没有人处理这个问题），经理才会收到消息。



该操作中步骤 2 的详细信息:



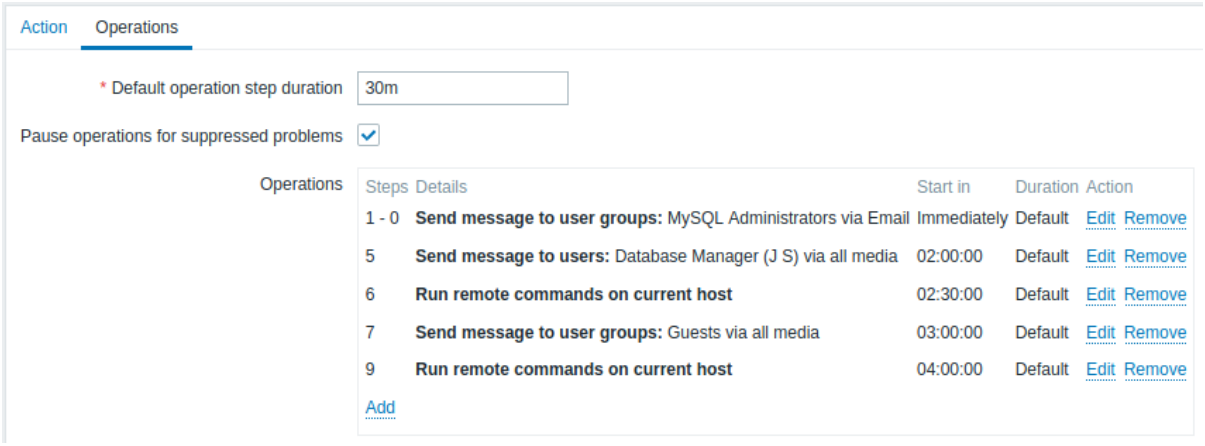
请注意 {ESC.HISTORY} 宏在自定义消息中的使用。该宏将包含关于此升级之前执行的所有步骤的信息，例如发送的通知和执行的命令。

示例 4

本示例展示了一个更为复杂的场景。在向 MySQL 管理员发送了多个消息并升级到经理那之后，Zabbix 将尝试重启 MySQL 数据库。如果该问题持续了两个半小时还没得到确认，就会进行重启的操作。

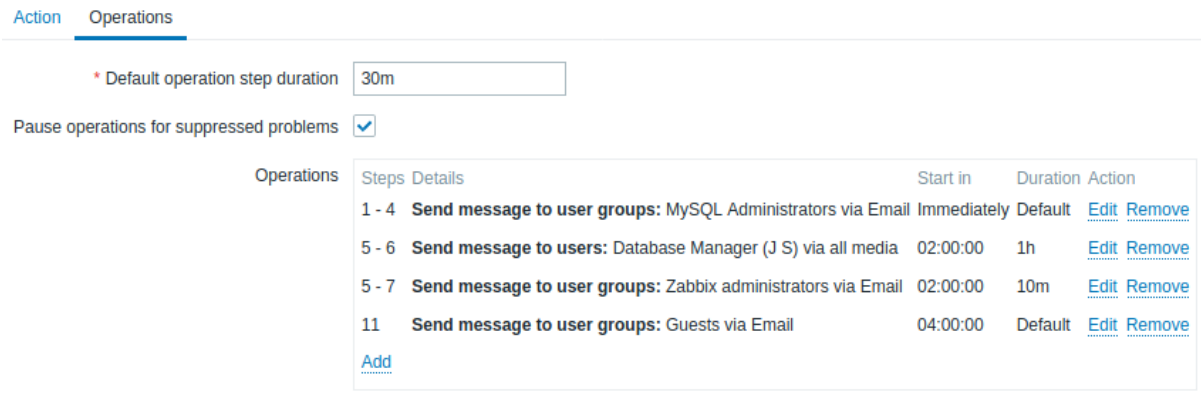
如果问题仍然存在，则再过 30 分钟，Zabbix 将向所有来宾用户发送一条消息。

如果还没有帮助，则再过 1 小时，Zabbix 将使用 IPMI 命令重启 MySQL 数据库服务器。



示例 5

一个步骤分配了具有多个操作的升级，并使用了自定义的时间间隔。默认的操作步骤持续时间为 30 分钟。



通知将按以下方式发出:

- 在问题发生后的第一时间（0:00）以及在发生后的 30 分钟 (0:30)、1 小时 (1:00)、1 个半小时 (1:30) 和 2 个小时 (2:00)，向 MySQL 管理员发送消息
- 在 2 小时 (2:00) 及 2 小时 10 分钟 (2:10) 后，向数据库经理发送消息（而不是 3 小时后；注意步骤 5 和 6 与下一个操作重叠了，下一个操作中较短的自定义步骤持续时间 (10 分钟) 覆盖了试图在此处设置的较长的步骤持续时间 (1 小时)）
- 在问题发生后的 2 小时、2 小时 10 分钟及 2 小时 20 分钟，向 Zabbix 管理员发送消息（自定义的步骤持续时间（10 分钟）生效）
- 在问题发生后的 4 小时，向来宾用户发送消息（步骤 8 到步骤 11，将返回继续使用默认的持续时间 (30 分钟)）

3 接收不支持监控项的通知

概述

从 Zabbix 2.2 开始，支持接收不支持的监控项的通知。

它是 Zabbix 内部事件概念的一部分，允许在这些情况下通知用户。内部事件反映了状态的变化：

- 当监控项从 “正常” 变成 “不支持”（反之亦然，即从 “不支持” 变成 “正常”）
- 当触发器从 “正常” 改为 “未知”（反之亦然，即从 “未知” 改为 “正常”）
- 当低级别发现规则从 “正常” 到 “不支持”（反之亦然，即从 “不支持” 到 “正常”）

本节介绍了如何在监控项变为不支持时 接收通知的操作方法。

配置

总的来说，对于那些以前就在 Zabbix 中设置过告警的人来说，设置通知的过程应该是相当熟悉了。

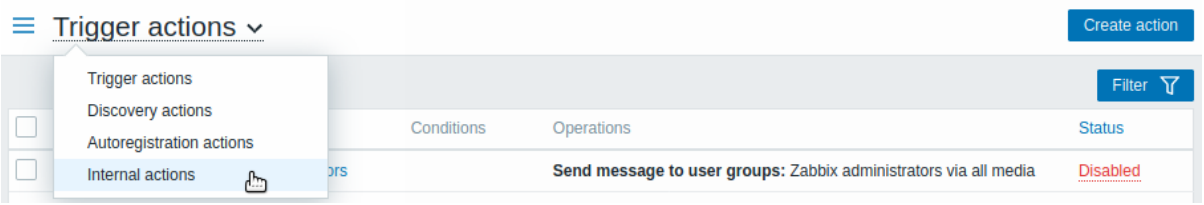
步骤 1

配置用于通知的媒介，如电子邮件、短信或脚本。请参阅手册的相应章节来执行此任务。

<note important> 内部事件通知使用了默认的严重性（' 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 时选中它。:::

步骤 2

转到 配置 → 动作，从标题的下拉菜单中选择 内部动作。

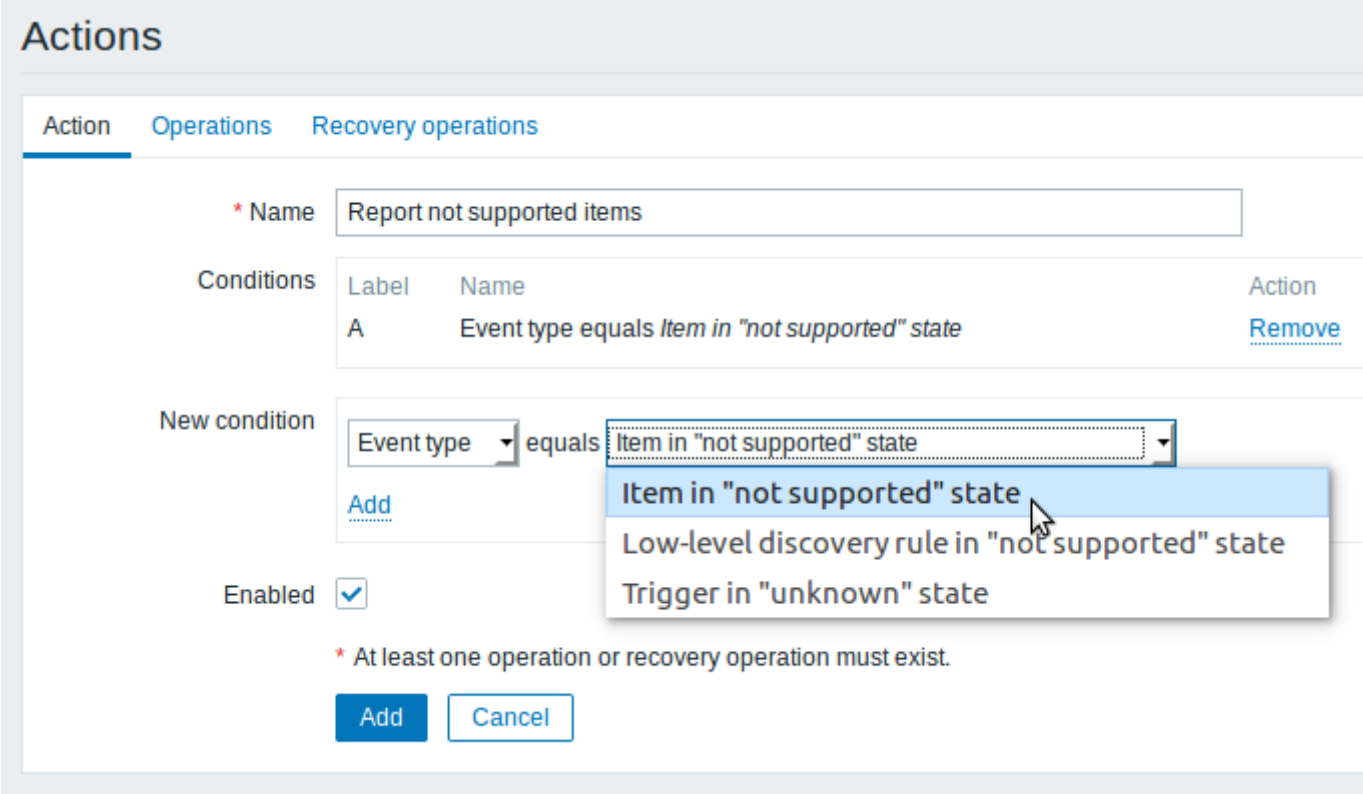


点击右侧的 创建动作打开一个动作配置表单。

步骤 3

在 动作选项卡中输入动作的名称。然后点击条件模块中的 添加来新增一个条件。

在新增条件弹出窗口中，选择 事件类型作为条件类型，然后选择 监控项在 “不支持” 的状态作为事件类型的值。

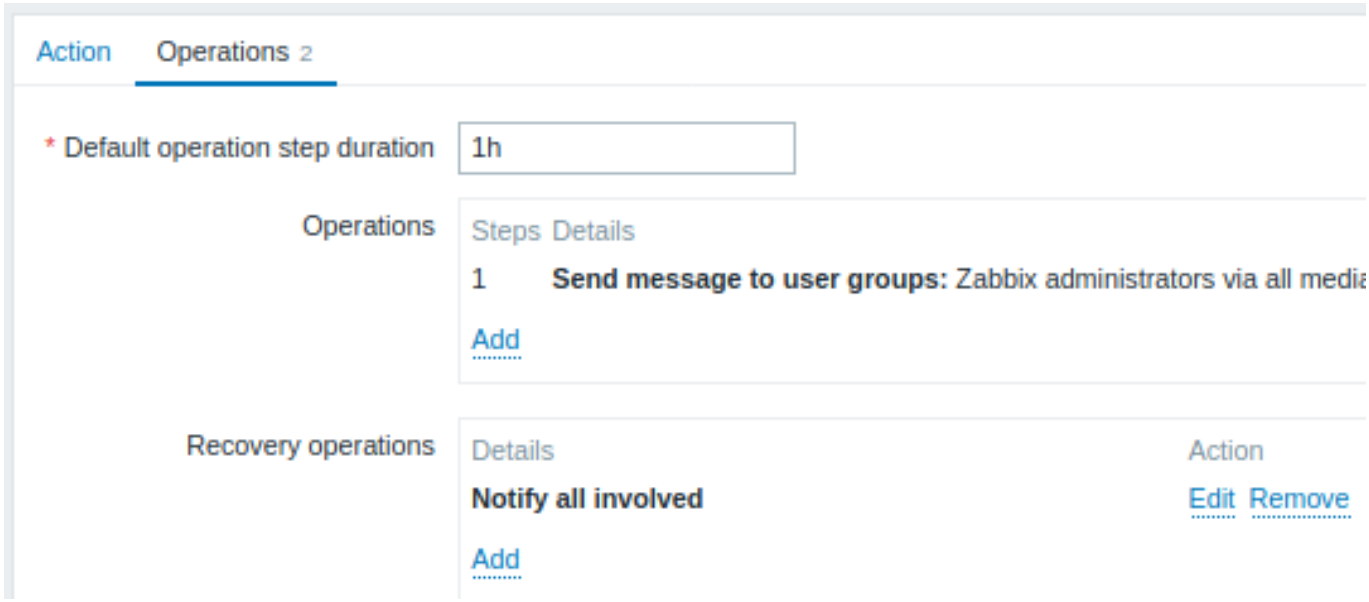


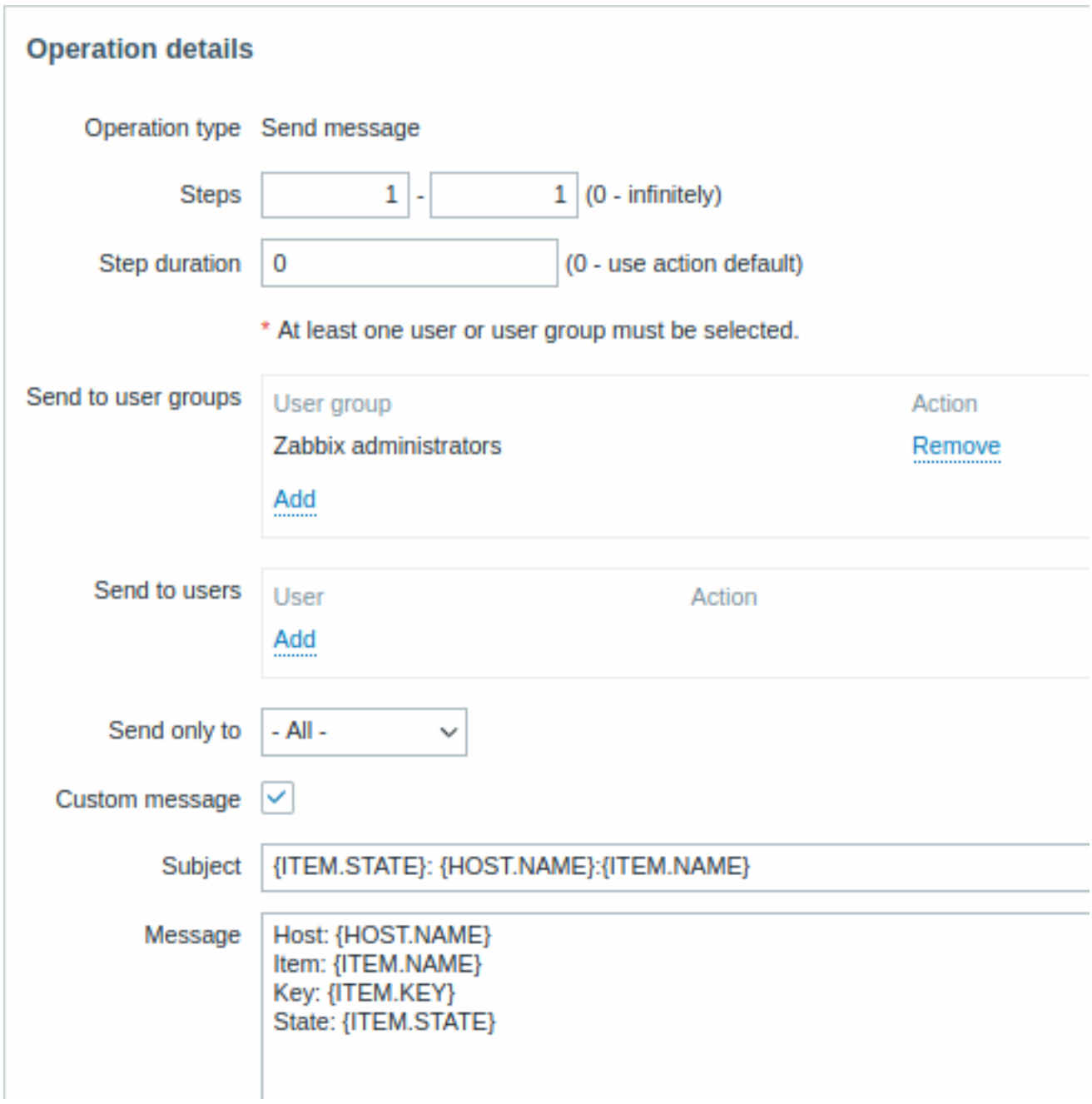
不要忘记点击 添加，来列入 条件模块中的实际所包含的条件。

步骤 4

在 操作选项卡中，在 操作模块中点击 添加，选择消息的接收者（用户组或用户）和用于发送的媒介类型（或选择 “所有”）

如果您希望自定义问题消息的主题/内容，请勾选 自定义消息复选框。





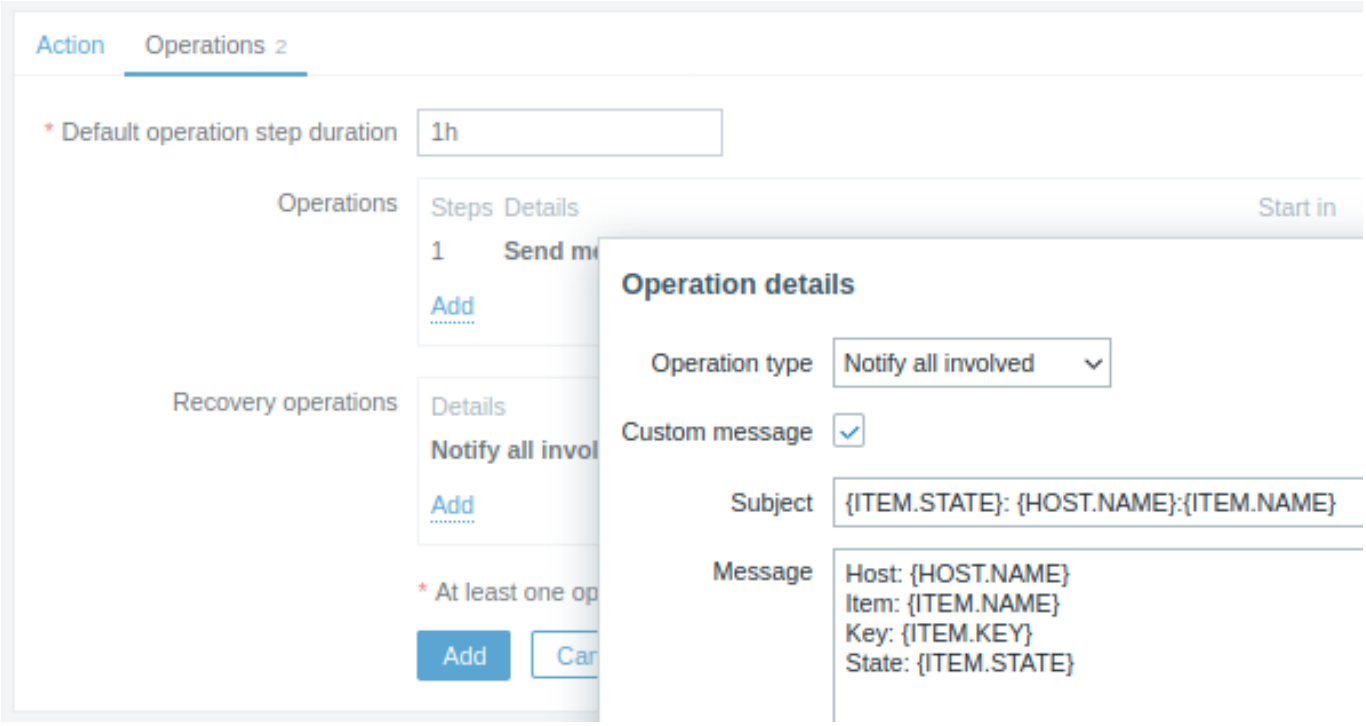
点击 操作模块中的 添加，列入实际所包含的操作。

如果您希望收到多个通知，请设置操作步骤持续时间（消息发送间隔）并添加其他步骤。

步骤 5

恢复操作模块中可配置恢复通知，当监控项恢复到正常状态时，可以收到其恢复消息。点击 恢复操作模块中的 添加，选择操作类型、消息接收者（用户组或用户）以及用于发送的媒介类型（或选择 “所有”）

如果您希望自定义恢复消息的主题/内容，请勾选 自定义消息复选框。



在 操作细节的弹出窗口中点击 添加，列入在 恢复操作模块中实际所包含的操作。

步骤 6

完成后，点击表单下方的 添加按钮。

这样您就完成了！现在，您可以期待收到 Zabbix 发来的第一个通知了（若某些监控项变成了 “不支持” 状态）。

10 宏

概述

Zabbix 支持许多内置的宏，这些宏可以在各种情况下使用。宏是一个变量，由特定的语法标识：

```
{MACRO}
```

根据上下文中，宏解析为一个特定的值。

有效地使用宏可以节省时间，并且使 Zabbix 的配置更为简化易懂。

在模板中使用宏是一种典型的用法。因此，模板的触发器就可能命名为 “Processor load is too high on {HOST.NAME}”。当这个模板应用于主机（如 Zabbix Server）时，当触发器展示在监测页面上时，其名称将解析为 “Processor load is too high on Zabbix server”。

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]` 。没有必要对参数使用双引号，因为 Zabbix 会处理任何有歧义的特殊符号（如果这些符号存在于已解析的宏中）。

除内置宏之外，Zabbix 还支持用户自定义的宏、具有上下文的自定义宏和用于低级别发现的宏。

另请参阅：

- 内置宏 的完整列表
- 宏函数
- 用户宏
- 上下文用户宏
- 低级别发现宏

1 宏函数

概述

宏函数提供了能自定义宏 值的功能。

有时宏可能会解析为一个不一定易于处理的值。它可能很长，或包含你想提取的感兴趣的特定子字符串。那么此时宏函数就发挥了作用。

宏函数的语法为：

```
{<macro>.<func>(<params>)}
```

其中：

- <macro> - 为要自定义的宏（例如 {ITEM.VALUE} 或 {#LLDMACRO}）
- <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)，，这些符号开头，则必须用引号括起来。

例如：

```
{{ITEM.VALUE}.regsub(pattern, output)}
{{#LLDMACRO}.regsub(pattern, output)}
```

可支持的宏函数

函数	描述 *	参数 ** ** 可	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。支持 **\1 - \9** 占位符来匹配组。**\0** 返回匹配的文本。{ITEM.LASTVALUE}	低级别发现宏 （除了在低级别发现规则过滤器中）
iregsub (<pattern>,<output>)	通过正则表达式匹配提取的子字符串（不区分大小写）。**pattern** - 匹配的正则表达式	{ITEM.VALUE} **output** - 输出的选项。支持 **\1 - \9** 占位符来匹配组。**\0** 返回匹配的文本。{ITEM.LASTVALUE}	低级别发现宏 （除了在低级别发现规则过滤器中）

如果在受支持的位置 使用函数，但是应用于不支持宏函数的宏，那么宏的计算结果为' 未知'。

如果 pattern 不是一个正确的正则表达式，那么宏的计算结果为' 未知'(低级别发现宏除外，在这种情况下，函数将被忽略，宏将保持未展开)

如果宏函数应用于不支持宏函数的位置的宏，那么该函数将被忽略。

示例

下面的例子说明了宏函数用于自定义宏值的方法，这些宏值包含 "日志行" 作为接收值:

接收值宏	输出
123Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", Problem)}
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", "Problem")}

接收值宏	输出
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+) Log line$", "Problem ID: \1")}
Log line	{{ITEM.VALUE}.regsub("^Log line$", "Problem ID: \1")}
MySQL crashed errno 123	{{ITEM.VALUE}.regsub("^(\w+) (\d+)", "Problem ID: \1_\2")}
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+) Log line$", "Problem ID: \1")}
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}
customername_1	{$MACRO:"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}"}
customername_1	{$MACRO:"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}"}
customername_1	{$MACRO:"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}"}
customername_1	"{$MACRO:\"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}\"}"
customername_1	"{$MACRO:\"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}\"}"

接收值宏	输出
customername_1	"{$MACRO:\"{{#IF{$MACRO:\"{$MACRO:\"{$ \1)}\"}" \1)}\"}" (无效的正则表达式)

2 User macros

Overview

User macros are supported in Zabbix for greater flexibility, in addition to the macros supported out-of-the-box.

User macros can be defined on global, template and host level. These macros have a special syntax:

`{$MACRO}`

Zabbix resolves macros according to the following precedence:

1. host level macros (checked first)
2. macros defined for first level templates of the host (i.e., templates linked directly to the host), sorted by template ID
3. macros defined for second level templates of the host, sorted by template ID
4. macros defined for third level templates of the host, sorted by template ID, etc.
5. global macros (checked last)

In other words, if a macro does not exist for a host, Zabbix will try to find it in the host templates of increasing depth. If still not found, a global macro will be used, if exists.

> **Warning:**
> If a macro with the **same name** exists on multiple templates of the same level, the macro from the template with the lowest ID will be used. Thus having macros with the same name in multiple templates is a configuration risk.

If Zabbix is unable to find a macro, the macro will not be resolved.

> **Attention:**
> Macros (including user macros) are left unresolved in the Configuration section (for example, in the trigger list) by design to make complex configuration more transparent.

User macros can be used in:

- item name
- item key parameter
- item update intervals and flexible intervals
- trigger name and description
- trigger expression parameters and constants (see examples)
- many other locations - see the full list

Common use cases of global and host macros

- use a global macro in several locations; then change the macro value and apply configuration changes to all locations with one click
- take advantage of templates with host-specific attributes: passwords, port numbers, file names, regular expressions, etc.

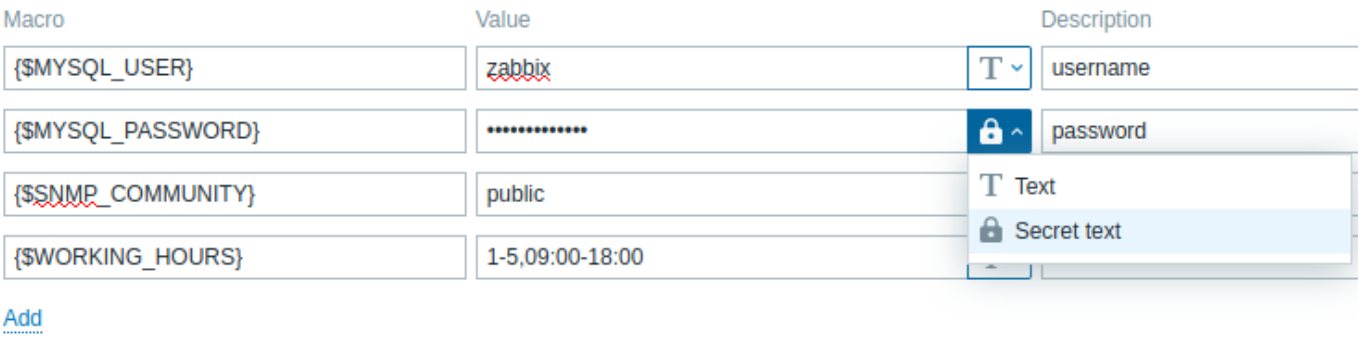
Configuration

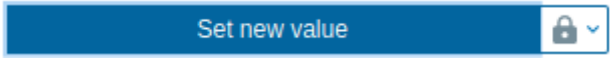
To define user macros, go to the corresponding location in the frontend:

- for global macros, visit *Administration → General → Macros*
- for host and template level macros, open host or template properties and look for the *Macros* tab

Note:
If a user macro is used in items or triggers in a template, it is suggested to add that macro to the template even if it is defined on a global level. That way, if the macro type is *text* exporting the template to XML and importing it in another system will still allow it to work as expected. Values of secret macros are not exported.

User macro has the following attributes:



Parameter	Description
Macro	Name of a macro. Must be wrapped in curly brackets. Text inside the brackets must start with a dollar sign. Example: {$FRONTEND_URL}. The following characters are allowed in the macro names: **A-Z** (uppercase only) , **0-9** , _ , **.**
Value	Content of a macro. Starting from Zabbix 5.0 two types of values are supported in user macros: *Text* (default) and *Secret text*. *Secret text* mode masks the content of a macro with asterisks, which could be useful to protect sensitive information such as passwords or shared keys. *Note* that while the value of a secret macro is hidden from sight, the value can be revealed through the use in items. For example, in an external script an 'echo' statement referencing a secret macro may be used to reveal the macro value to the frontend because Zabbix server has access to the real macro value. To select macro value type click on a button at the right end of the `Value` input field: icon indicates a text macro icon indicates a secret text macro. Upon hovering, the `Value` field transforms into  button, which allows to enter a new value of the macro (to exit without saving a new value, click backwards arrow ( )
Description	Text field used to provide more information about this macro.

Note:
URLs that contain a secret macro will not work as the macro in them will be resolved as "******".

Attention:
In trigger expressions user macros will resolve if referencing a parameter or constant. They will NOT resolve if referencing a host, item key, function, operator or another trigger expression. Secret macros cannot be used in trigger expressions.

Examples

Example 1

Use of host-level macro in the "Status of SSH daemon" item key:

```
net.tcp.service[ssh,,{$SSH_PORT}]
```

This item can be assigned to multiple hosts, providing that the value of **{$SSH_PORT}** is defined on those hosts.

Example 2

Use of host-level macro in the "CPU load is too high" trigger:

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

Such a trigger would be created on the template, not edited in individual hosts.

Note:
If you want to use amount of values as the function parameter (for example, **max(#3)**), include hash mark in the macro definition like this: SOME_PERIOD => #3

Example 3

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

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

Note that a macro can be used as a parameter of trigger function, in this example function **min()**.

Example 4

Synchronize the agent unavailability condition with the item update interval:

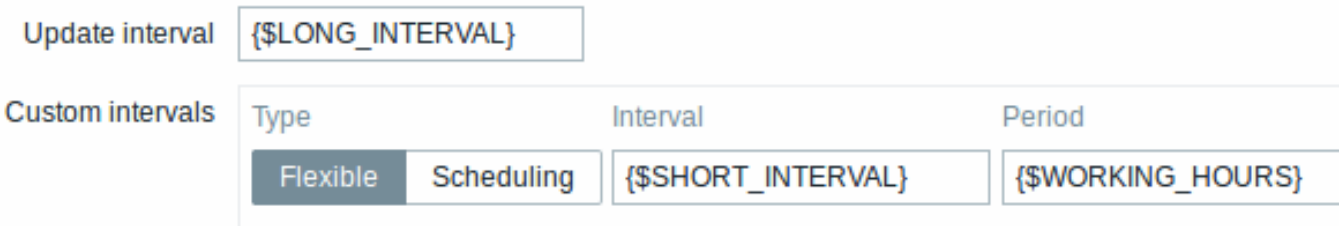
- define {$INTERVAL} macro and use it in the item update interval;
- use {$INTERVAL} as parameter of the agent unavailability trigger:

```
{ca_001:agent.ping.nodata({$INTERVAL})}=1
```

Example 5

Centralize configuration of working hours:

- create a global {$WORKING_HOURS} macro equal to 1-5,09:00-18:00;
- use it in *Administration → General → Working time*;
- use it in *User → Media → When active*;
- use it to set up more frequent item polling during working hours:

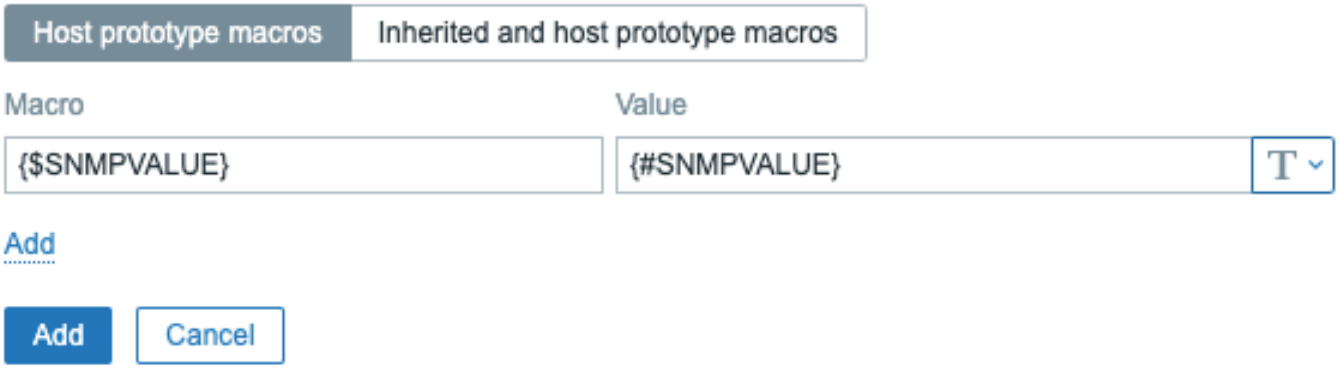


- use it in the *Time period* action condition;
- adjust the working time in *Administration → General → Macros*, if needed.

Example 6

Use host prototype macro to configure items for discovered hosts:

- on a host prototype define user macro {$SNMPVALUE} with {#SNMPVALUE} low-level discovery macro as a value:



- assign *Template Module Generic SNMPv2* to the host prototype;
- use {$SNMPVALUE} in the *SNMP OID* field of *Template Module Generic SNMPv2* items.

User macro context

See user macros with context.

3 上下文用户宏

概述

可选的上下文可以在用户宏 中使用，允许用上下文特定的值覆盖默认值。

上下文是附加到宏名称中的；语法取决于上下文是否是静态文本值：

```
{$MACRO:"static text"}
```

或正则表达式（支持 zabbix 5.0.2 及以上版本）

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

要注意的是：具有正则表达式上下文的宏只能在用户宏配置中定义。若 `regex:` 前缀在其他地方用作用户宏上下文，比如在触发器表达式中，它将被视为静态上下文。

上下文引用是可选的（请参考注意事项）。

宏上下文使用示例：

示例描	
{$LOW_SPACE_LIMIT}	用户宏无上下文。
{$LOW_SPACE_LIMIT:/tmp}	用户宏使用上下文(静态字符串)。

示例描	
{$LOW_SPACE_LIMIT:regex:"^/tmp$"}	用户宏使用上下文(正则表达式)。与{$LOW_SPACE_相同
{$LOW_SPACE_LIMIT:regex:"^/var/log/.*$"}	用户宏使用上下文(正则表达式)。匹配所有以/var/log/开头的字符串。

使用案例

在触发器表达式中，使用上下文用户宏可以实现更为灵活的阈值（基于低级别发现检索到的值）。例如，您可以这样来定义：

- {$LOW_SPACE_LIMIT} = 10
- {$LOW_SPACE_LIMIT:/home} = 20
- {$LOW_SPACE_LIMIT:regex:”^\/[a-z]+$”} = 30

那么，在用于发现已挂载文件系统的触发器原型中，低级别发现的宏将会被用作宏上下文：

```
{host:vfs.fs.size[{#FSNAME},pfree].last()}<{$LOW_SPACE_LIMIT:"{#FSNAME}"}
```

当发现不同的低空间阈值后，根据已发现的挂载点或文件系统类型，宏上下文将应用到触发器中。那么以下情况，将会触发问题事件：

- /home 文件夹的可用磁盘空间不足 20%
- 匹配到正则表达式的文件夹（如/etc,/tmp 或是/var）可用磁盘空间不足 30%
- 未匹配正则表达式并且不是/home 的文件夹可用磁盘空间不足 10%

注意事项

- 如果存在多个有上下文的用户宏，Zabbix 将尝试先匹配简单的上下文宏，然后非顺序地匹配具有正则表达式的上下文宏。

<note warning> 请勿创建不同的上下文宏来匹配相同的字符串，以避免未定义的行为:::

- 如果在主机、关联模板及全局上找不到带有上下文的宏，那么将会搜索不带有上下文的宏。
- 上下文只支持低级别发现的宏。任何其他宏都将被忽略并视其为纯文本。

从技术上来说，宏上下文指定使用的规则类似于监控项键值 的参数。除非如果有个 , 字符，宏上下文不是被解析为几个参数：

- 如果上下文中包含} 字符或者以" 字符开头的，宏上下文必须用" 引起来。引号内的引号必须用\字符进行转义。
- \ 字符本身是不转义, 这就意味着在引用的上下文中是不可能出现以 \ 字符结尾的 —— {$MACRO:”a:\b\c\”} 这个宏是无效的。
- 上下文中的前导空格会被忽略，后面的空格不会：
  - 例如：{$MACRO:A} 和 {$MACRO: A} 是一样的, 但与 {$MACRO:A } 则是两个不同的宏。
- 前导引号之前和后引号之后的所有空格都会被忽略，但引号内的所有空格不会被忽略：
  - {$MACRO:”A”}, {$MACRO: ”A”}, {$MACRO:”A” } 和 {$MACRO: ”A” } 这几个宏都是等价的, 但 {$MACRO:”A”} 和 {$MACRO:” A ”} 则是两个不同的宏。

下面这些宏是等价的，它们都具有相同的上下文：{$MACRO:A}, {$MACRO: A} 和 {$MACRO:”A”}。这与监控项键值相反，’key[a]’, ’key[a]’ 和’key[”a”]’ 在语义上相同，但在独特性用途上是不同的。

4 低级别发现宏

概述

有一种在低级别自动发现 函数中使用的宏:

{#MACRO}

它是一个在低级别发现规则中使用的宏，其返回文件系统名称、网络接口和 SNMP OIDs 等的真实值。

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、网络接口等时，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。

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。

一些低级别发现宏在 Zabbix 中是已经预先内置了的，例如 {#FSNAME}、{#FSTYPE}、{#IFNAME}、{#SNMPINDEX}、{#SNMPVALUE} 这些宏。但是，在创建自定义 低级别发现规则的时候，遵守这些宏名称并不是强制性的。所以，你可以使用任何其他的低级别发现宏名称并引用该名称。

可支持的位置

低级别发现宏可以用在：

- 在低级别规则过滤器中
- 用于监控项原型中：
  - 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
  - 从 Zabbix 4.0 开始，也适用于:
    - 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

- 用于触发器原型中：
  - names
  - operational data
  - expressions(仅用于常量和函数参数中)
  - URLs
  - descriptions
  - event tag names and values
- 用于图形原型中：
  - names
- 用于主机原型中：
  - names
  - visible names
  - host group prototype names
  - host macro value
  - (详细查阅完整列表)

在上述所有位置，低级别发现宏都可以在静态用户宏上下文 中使用。

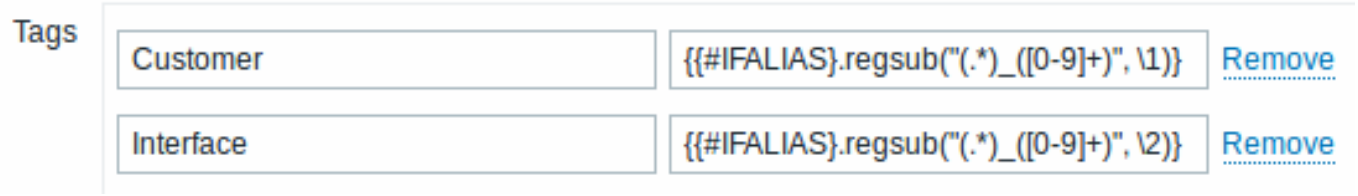
使用宏函数

宏函数支持低级别发现的宏（除了在低级别发现规则过滤器中），允许使用正则表达式提取宏值的特定部分。

比如，您可能希望从以下低级别发现宏中提取客户名称和接口编号，以便进行事件标记：

`{#IFALIAS}=customername_1`

为此，`regsub` 宏函数可以与触发器原型的事件标记值字段中的宏一起使用：



注意, 在未加引号的监控项键值参数 中不允许使用逗号, 因此，包含宏函数的参数必须用引号括起来。反斜杠 (\) 字符用作于转义参数内的双引号。例如:

```
net.if.in["{{#IFALIAS}.regsub(\"(.*)_([0-9]+)\", \1)}",bytes]
```

有关宏函数语法的更多信息，请参考：宏函数

从 Zabbix 4.0 开始，低级别发现宏就支持宏函数了。

脚注

[1] 在标记为单个宏的字段[1] 中必须填满整个字段。字段中不支持多个宏或宏与文本的混合。

11 用户和用户组

概述

Zabbix 中的所有用户都通过 Web 前端去访问 Zabbix 应用程序。每个用户都分配有唯一的登陆名和密码。

所有用户的密码均已加密并储存在 Zabbix 数据库中。用户不能使用其用户 ID 和密码直接登陆到 UNIX 服务器中，除非这些用户也在 UNIX 中进行了相应的设置。可以使用 SSL 来保护 Web 服务器和用户浏览器之间的通讯。

通过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：

- 管理 Zabbix 的前端功能
- 主机组中被监视的主机

1 配置用户

概述

初始安装完 Zabbix 后，已有两个预先定义好的用户：

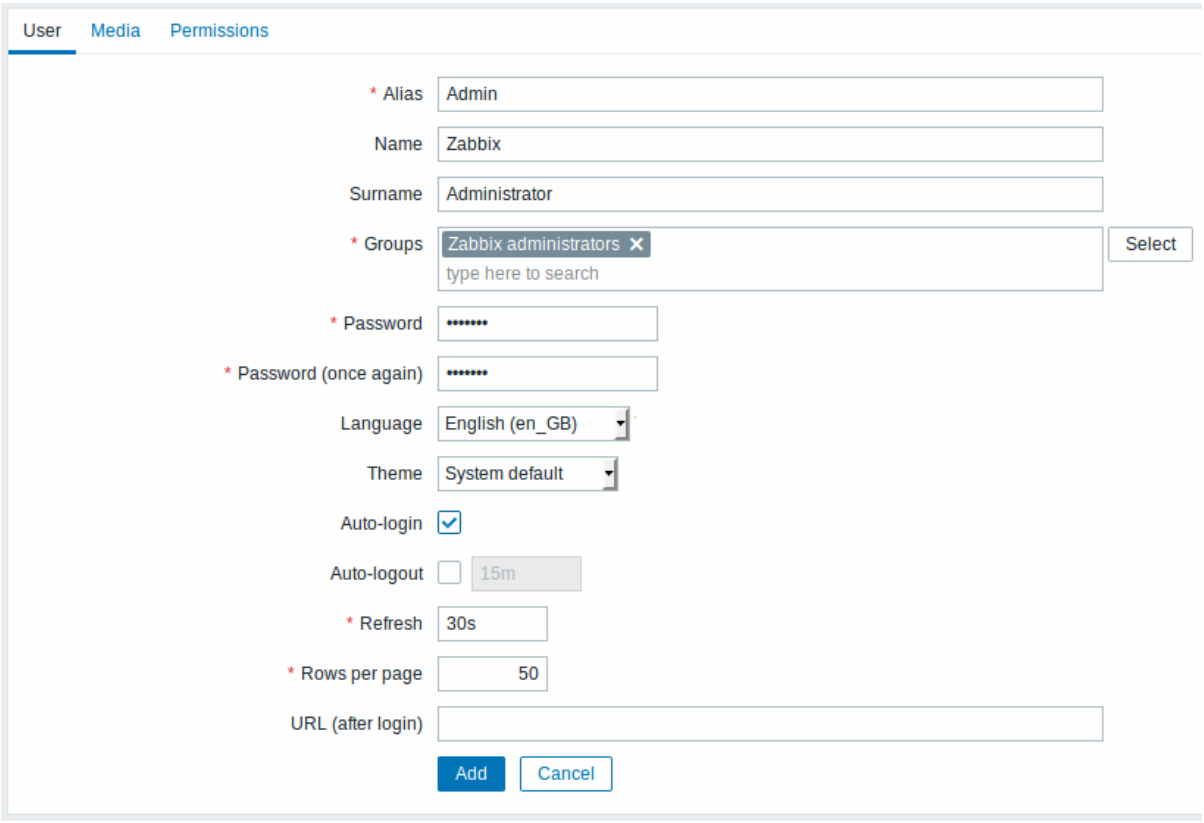
- *Admin* - 具有完全权限的 Zabbix 超级用户 ；
- *guest* - 特殊的 Zabbix 用户。'guest' 用户默认是禁用的。如果将其添加到 Guests 用户组，则可以在没有登录的情况下访问 Zabbix 中的监控页面。注意：默认情况下, 'guest' 对 Zabbix 对象没有权限。

根据以下步骤来配置一个新用户：

- 转到 管理 → 用户
- 点击 创建用户（或单击用户名来编辑现有用户）
- 在表单中编辑用户属性

一般属性

用户选项卡包含了一般的用户属性：



红色星号标记的为必填字段。

参数	描
别名	唯一的用户名，用作登陆名。

参数	描
名字用	的名字(选填)。若此项不为空，则在确认信息和通知收件人信息中可见。
姓氏用	的姓氏(选填)。若此项不为空，则在确认信息和通知收件人信息中可见。

参数	描
群组	选用户所属的用户组。从Zabbix 3.4.3开始，此字段是自动完成的。所以在输入用户组名称时，会提供匹配组的下拉列表，向下滚动来进行选择。或者点击选择来添加组。点击’x’以删除所选的组。所属的用户组决定了用户可以访问哪些主机组和主机。

参数	描
密码有	个字段用于输入用户密码。对于已经存在的密码，只会有一个 密码按钮，单击此按钮则可以打开密码字段。
语言 Z	bbix 前端的语言。进行语言转换，需要 PHP get-text 的扩展插件。

参数	描
主题	设前端的样式：系统默认 - 使用默认的系统设置蓝 - 标准的蓝色主题深色 - 深色主题高对比度亮色 - 高对比度的明亮主题高对比度深色 - 高对比度的深色主题
自动登录	选中此选框，使 Zabbix 在 30 天内记住该用户并自动登录。此项需要使用浏览器的 cookies。

参数	描
自动注销	选中此选框，用户将在已设置的秒数(最短90秒，最长1天)后自动注销。
刷新设	图形、屏幕、文本数据等的刷新频率。设置为0则表示禁用刷新。
每页行数	设置在表中，每个页面显示的行数。
*URL(*登录后*)* 使 Z	bbix在用户登录后，跳转到此URL。例如，登录后转到问题页面。

报警媒介

报警媒介选项卡包含了为用户定义的所有媒体列表。报警媒介用于发送通知。点击 添加将报警媒介分配给用户。

关于配置媒介类型的详细信息，请参见媒介类型 章节。

权限

权限选项卡包含以下信息：

- 用户类型（用户, 管理员和超级管理员）。用户不能更改自己的用户类型。

- 用户可以访问的主机组。默认情况下，“用户” 和 “管理员” 用户无权访问任何的主机组和主机。若要获得访问权限，需要将它们放到能够访问相应主机组和主机的用户组中。

详细信息请参阅用户权限 页面。

2 权限

概述

在 zabbix 中，您可以通过分别定义用户类型，然后将非特权用户包含在具有访问主机组数据权限的用户组中，来区分用户权限。

用户类型

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。

用户类型	描述
Zabbix 用户	可以访问“监测”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”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	可以访问所有内容：监测、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。

主机组权限

只准许主机组级别的用户组 访问 Zabbix 中的任何主机数据。

这就意味着单个用户不能被直接授予主机（或主机组）的访问权限。只能通过其归属的用户组被授予（目标）主机所在的主机组的访问权限，从而访问该主机。

3 用户组

概述

用户组允许根据组织目的和为数据分配权限对用户进行分组。监控主机组数据的权限只能分配给用户组，而不是单个用户。

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。这可以通过对用户进行分组，然后将不同的权限分配给主机组来实现。

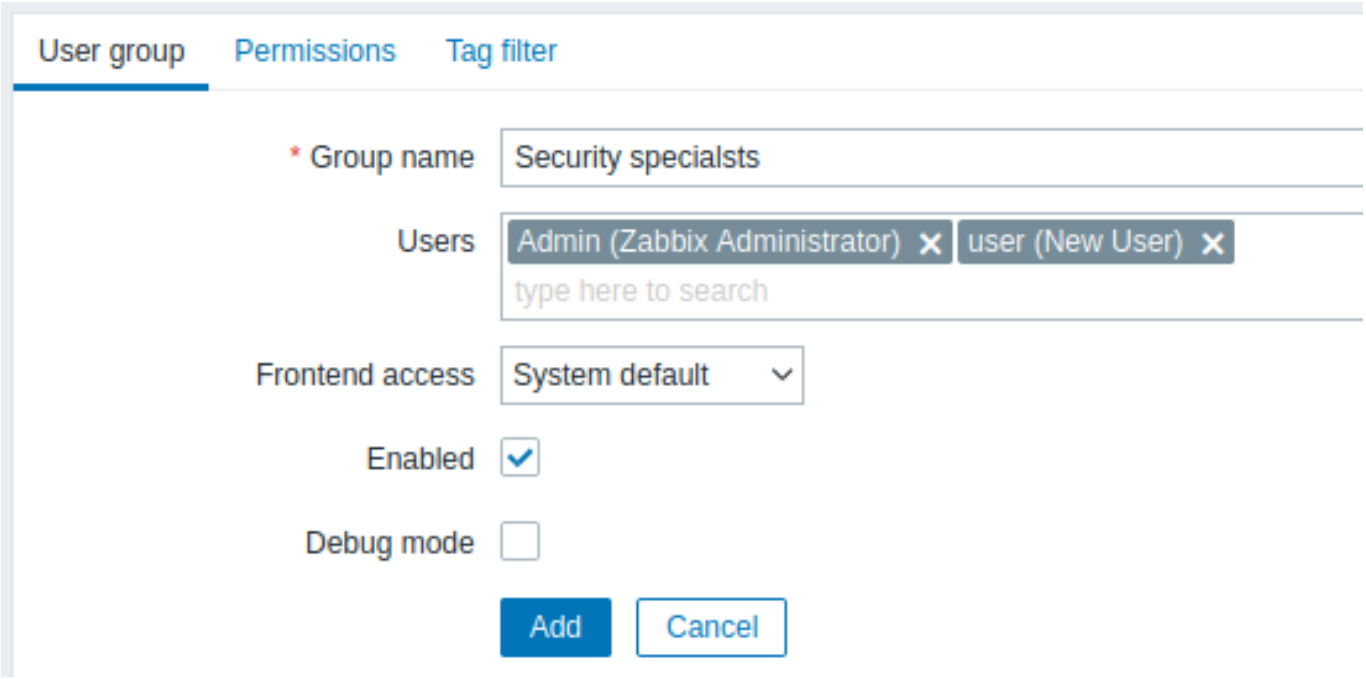
一个用户可以属于任意数量的组。

配置

通过以下步骤配置用户组：

- 转到 //管理 → 用户组 //
- 点击 创建用户组（或者点击组名编辑现有的组）
- 在表单中编辑组的属性。

用户组选项卡包含了以下一般的组属性：



红色星号标记的为必填字段。

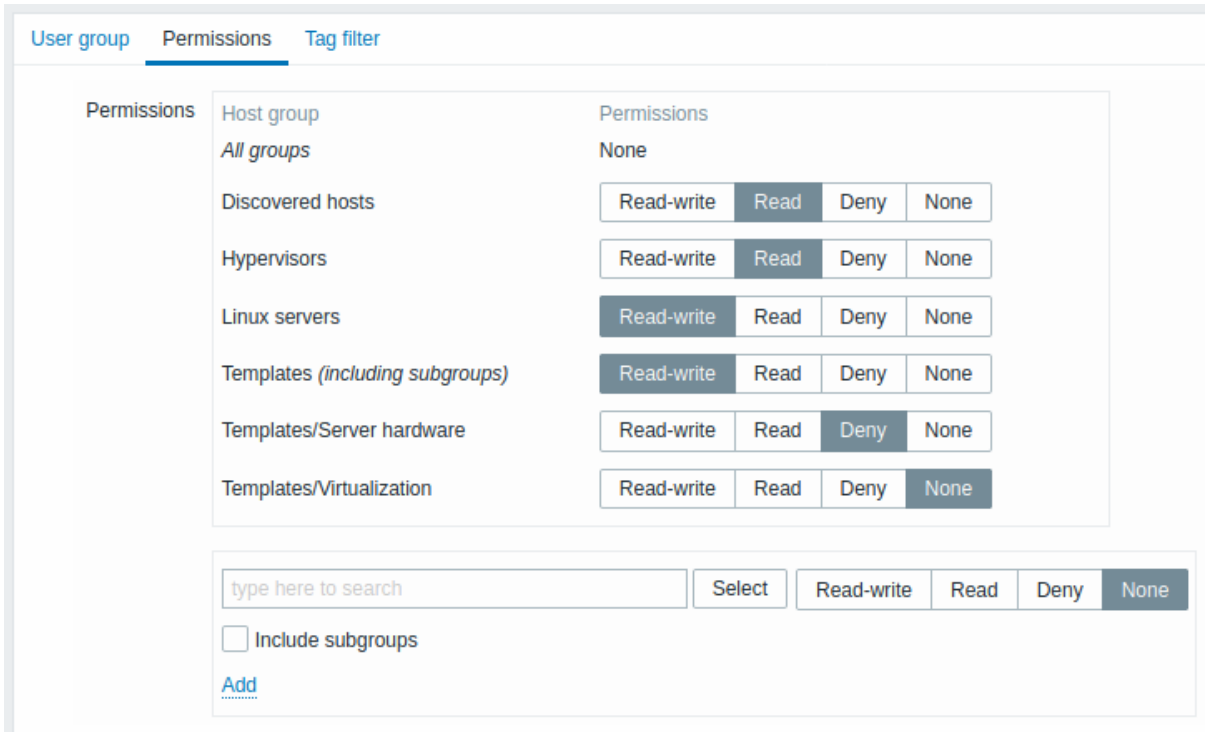
参数描	
组名唯	的组名。

参数	描
用户输	现有用户的名称，将其添加到组中。当下拉菜单中出现了匹配的用户名时，向下滚动来进行选择. 或者您也可以点击选择按钮，在弹出的窗口中来选择用户。

参数	描
前端访问如何验	组的用户。 系统默认 - 使用默认的验证方式 (全局设置) **Internal** - 使用 Zabbix 内部验证 (即使全局使用了 LDAP 验证)。如果 HTTP 认证是全局默认的，则忽略。 **LDAP** - 使用 LDAP 验证 (即使全局使用了内部验证)。如果 HTTP 认证是全局默认的，则忽略。 停用的 - 被禁止访问 Zabbix 前端。
已启用用户	及其成员的状态。 选中 - 用户组和用户被启用。 未选中 - 用户组和用户被禁用。

参数描	
调试模式选中此	选框以激活用户的调试模式。

权限选项卡允许您指定用户组对主机组（和组内主机）数据的访问权限：



当前对主机组的权限显示在 权限块中。

如果所有嵌套的主机组都继承了主机组的当前权限，则主机组名称后括号中的 包括子组文本表示该主机组。

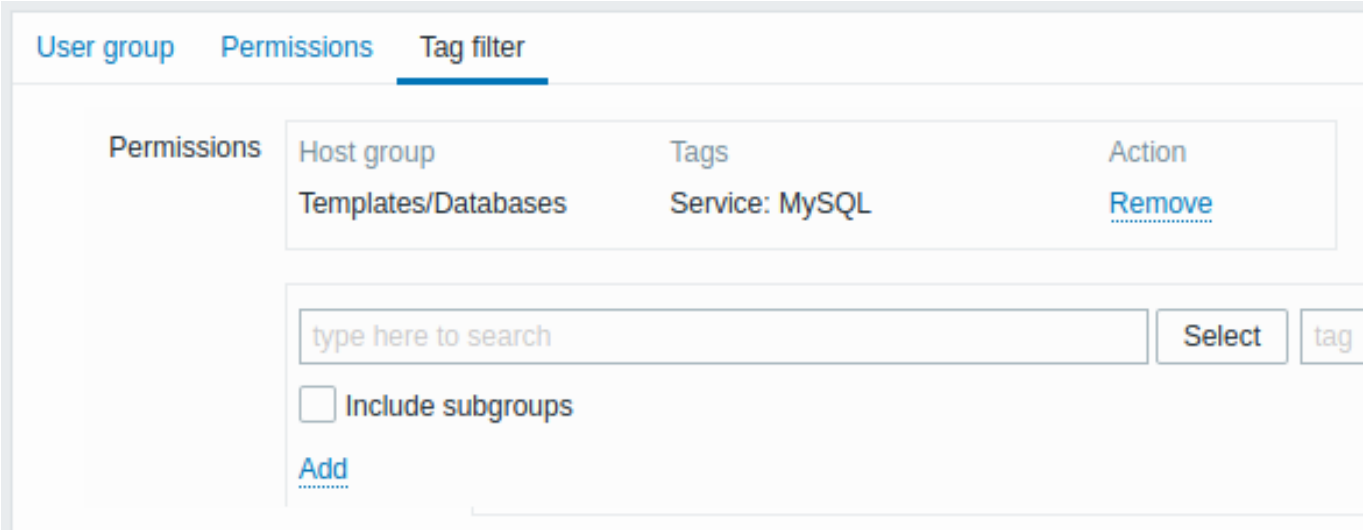
您可以更改对主机组的访问级别：

- 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。

使用下面的选择字段选择主机组和对它们的访问级别（注意：如果主机组已经在列表中，选择 无会将该主机组从列表中移除）。如果要包括嵌套主机组，请选中 包括子组复选框。该字段是自动完成的，因此在开始键入主机组名称时，将会提供一个匹配组的下拉列表。如果你希望查看所有主机组，请单击 选择按钮。

请注意，主机组配置 中的 Zabbix 超级管理员可以对嵌套主机组执行与父主机组相同级别的权限。

标签过滤器选项卡允许您为用户组设置基于标签的权限，来查看使用标签名称及其值过滤的问题：



要选择一个标签过滤器应用于的主机组，点击 选择查看现有主机组的完整列表或输入一个主机组的名称来获取匹配主机组的下拉列表。如果您想将标记过滤器应用于嵌套的主机组，使用内置的主机组标签，请选中 包括子组复选框。

标签过滤器允许将对主机组的访问与发现问题的可能性分开。

例如，如果一个数据库管理员只需查看”MySQL” 数据库的问题，则需要先创建一个数据管理员的用户组，然后配置”Service” 标签名的值为”MySQL”。

如果指定了 “Service” 标签名，value 字段为空，则相应的用户组将看到标签名称为 “Service” 的所选主机组的所有问题。

如果标签名称和值字段均为空，但主机组已选中，则相应的用户组将看到所选主机组的所有问题。

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。

让我们来看一个例子：用户是多个用户组的成员。在这种情况下，过滤器将对标签使用 OR 条件。

用户组 **A**	**	户组 B**	** 两组的	户 (成员) 的可见结果 **
标签过滤器 主机组 * 标 名 * 标签值 Templates/Databases Service	主机组 MySQL	标签名 * 标签 * Templates/Databases Service	Oracle	标签名为 Service，标签值为 MySQL 或 Oracle 的问题可见
Templates/Databases 空	空 * T	mplates/Databases vice O S	acle 所	问题可见
未选择 * 空 空	Temp	ates/Databases ce Orac Serv	e 标签名和	签值为 Service:Oracle 的问题可见

Attention:
添加过滤器（例如，在主机组名”Templates/Databases” 中添加标签）将导致无法看到其他主机组的问题。

来自多个用户组的主机访问

一个用户可以属于任意数量的用户组。这些组可能对主机具有不同的访问权限。

因此，了解非特权用户最终能够访问哪些主机是很重要的。例如，让我们考虑一下对于用户组 A 和 B 中的用户，在不同情况下对主机 **X**(在主机组 1 中) 的访问将受到怎样的影响。

- 如果组 A 仅具有对主机组 1 的 读访问权限，而组 B 具有对主机组 1 的 读写访问权限，那么用户将获得对’X’ 的 读写访问权限。

Attention:
从 Zabbix 2.2 开始，“读写” 权限优先于 “读” 权限。

- 在与上述相同的场景中，如果'X' 同时也在主机组 2 中，并且 拒绝了用户组 A 和 B，那么对'X' 的访问将 不可用，尽管对主机组 1 具有 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'X' 的 读写访问权限。
- 如果用户组 A 具有对主机组 1 的 拒绝访问权限，而用户组 B 具有对主机组 1 的 读写权限，则用户将被拒绝访问'X'。

其他细节

- 具有对主机 读写权限的管理员级别用户，如果没有访问 *templates* 组的权限，就不能链接/取消链接模板。有了对 *Templates* 组的读访问权限，他将能够将模板链接到主机，但是，在模板列表中看不到任何模板，并且不能对其他地方的模板进行操作。
- 具有对主机 读权限的管理员级别用户，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。
- 任何非 zabbix 超级管理员用户（包括 '来宾'）都可以看到网络地图，只要地图为空或只有图像。当主机、主机组或触发器添加到地图中时，将遵守权限。这也同样适用于聚合图形和幻灯片演示。无论权限如何，用户都会看到任何没有直接或间接链接到主机的对象。
- 如果明确拒绝访问相关主机，Zabbix Server 将不会向定义为动作操作接收者的用户发送通知。

8. 服务监控

概述　服务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，现有 IT 基础设施架构的结构，以及更高层面的监控信息。

Zabbix 服务监控就是针对上述问题提出的解决方案。

服务监控是一种监控数据的分层表现。

下面我们来看一个非常简单的服务结构：

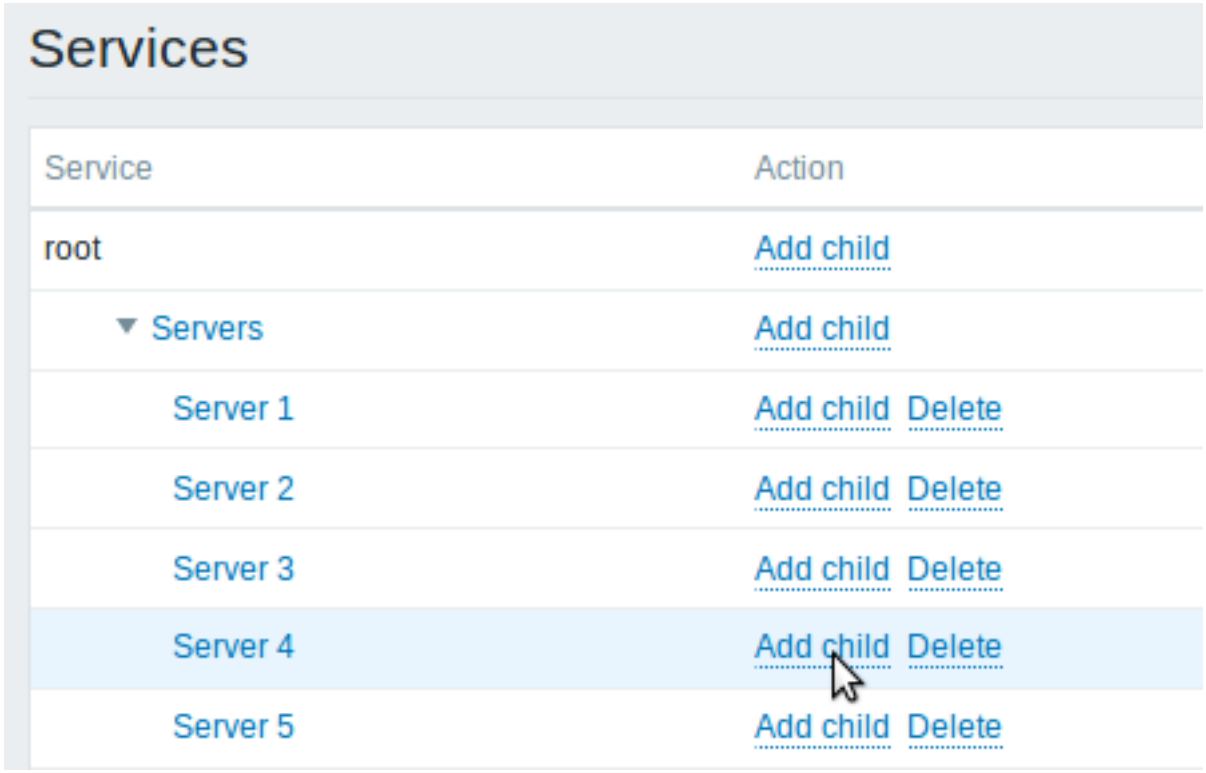
```
服务
|
|-工作站
| |
| |-工作站1
| |
| |-工作站2
|
|-服务器
```

在结构上每个节点都具有监控属性状态。根据所选择的算法，这个状态会被计算并关联到上层状态，服务监控功能最底层是关联的触发器。每个节点状态都是受其触发器状态影响。

> **Note:**
> 触发器的严重等级如：不分类或信息是不影响 SLA 指标计算的。

配置　配置服务监控，请点击：配置 → 服务。

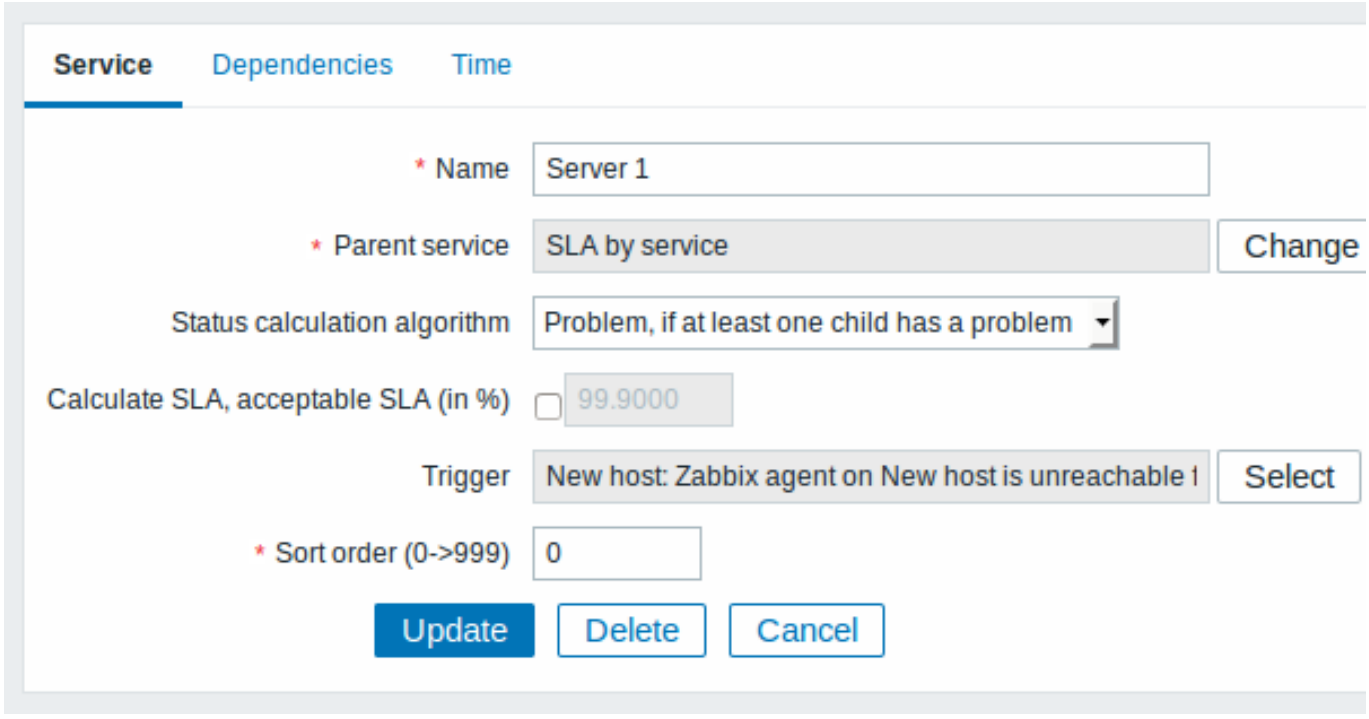
在这个界面上，您可以构建被监视的基础结构的层次结构。最高级的父服务是 "root"。您可以向下构建层次结构，方法是添加低级的父服务，然后向它们添加单个节点。



点击 添加子节点增加服务监控。点击名称可编辑一个已创建的服务监控，您可以通过弹出的界面编辑该服务监控属性。

配置一个服务监控

服务监控选项卡包含通用的服务监控属性



所有必填字段都标有红色星号。

参数说	
//名称 // 服	监控名称。

参数说	
//父服务监控 // 服务监控	属的父服务监控。

//状态计算算法 // 服务监控状

计算方法:服务状态的计算方法:\不计算-不计算服务监控状态。**异常，至少一个子服务出现问题**-只要一个子服务有异常，状态为异常。**异常，所有的子服务都有

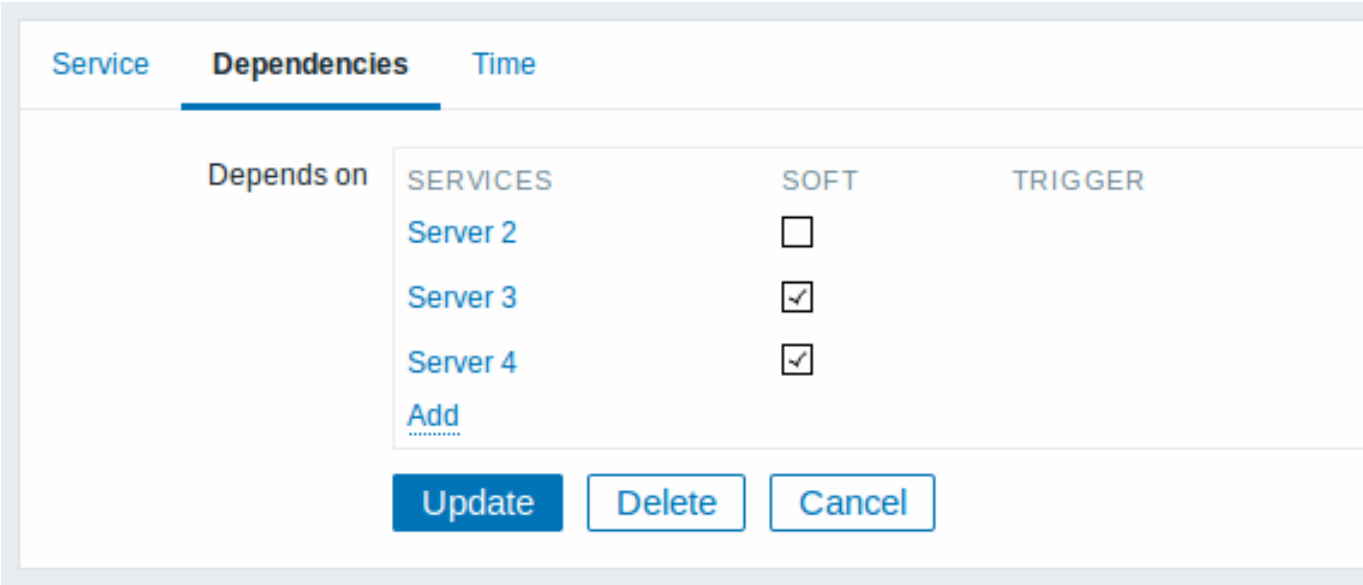
不计算,不计算服务状态**问题,如果至少有一个子节点服务有问题**—一问题状态,如果至少一个子节点服务有问题**问题,如果所有的孩子都有问题**—

参数	说
//计算 SLA // 启	SLA 计算并显示。
//可接受的 SLA(%) // 此服务控	可接受的 SLA 百分比，用于报告。

参数说	
//触发器 // 选择	联的触发器:无 - 没有关联的触发器 ** 触发器名称 ** - 选择关联触发器，因此取决于触发器状态。最底层服务监控必须关联触发器状态。(否则服务监控状态将无法

参数	说
//排序 // 显	排序的顺序，按升序排列。

** 依赖关系 ** 选项卡可以看到该服务监控所有子节点。单击 添加增加一个之前配置过的服务监控节点。

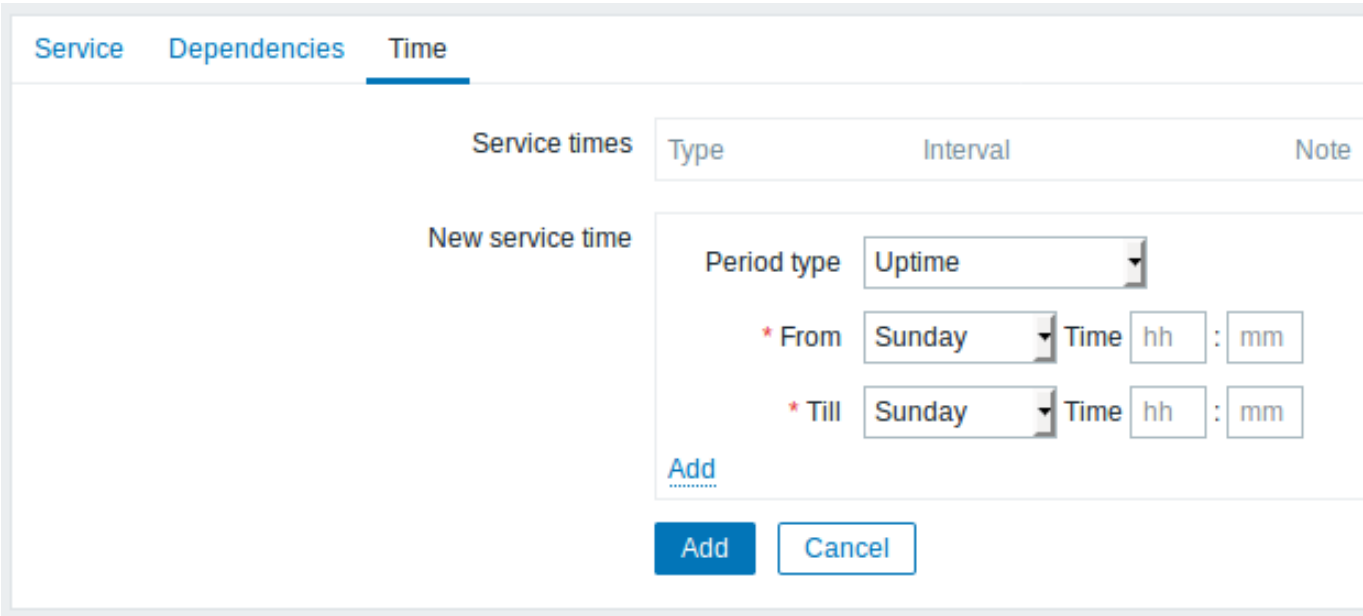


硬依赖和软依赖

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。第一个选项是将所有这些直接添加为子服务监控。

然而，如果有一些服务监控在其他节点已增加过，则不能简单的将其移动到该子节点。那该如何创建服务节点依赖？这个问题的答案是“软链接”。添加服务监控并勾选软连接选项。通过这种方式，服务可以保留节点之前原始位置，也可以绑定依赖到其他服务上。这种 “软连接” 的服务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。

** 时间 ** 选项卡，用于设置服务监控的工作时间。



参数说	
//服务监控时间 // 默认，所有	务监控都是预设24x7x365统计时间，如有特殊需要，请增加新的服务监控时间。

参数说

//新的服务监控时间 // 服务监控时间

** 在线时间 ** - 服务监控正常运行时间。

** 故障停机时间 ** - 故障停机时间周期内不会纳入SLA服务时间统计。

** 单次停机 ** - 单次停机时间，在该时间阶段内不

参数说

展示　前往监控服务，请点击 监控 → 服务。

9. Web 监控

概述　您可以使用 Zabbix 对网站进行多方面可用性监控：

> **Attention:**
> 若要使用 Web 监控，Zabbix Server 必须编译安装时加入 cURL (libcurl) 库支持

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 server 根据预定义的命令周期性的执行这些步骤。如果主机是通过代理监控的话，这些步骤将由代理执行。

从 Zabbix2.2 开始，Web 场景和监控项，触发器等一样，是依附在主机/模版上的。这意味着 web 场景也可以创建到一个模板里，然后应用于多个主机。

任何 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
- 最近的错误信息

对于 web 场景的所有步骤，都会收集下列数据：

- 每秒下载速度
- 响应时间
- 响应码

更多详情，请参见web 监控项。

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。

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。

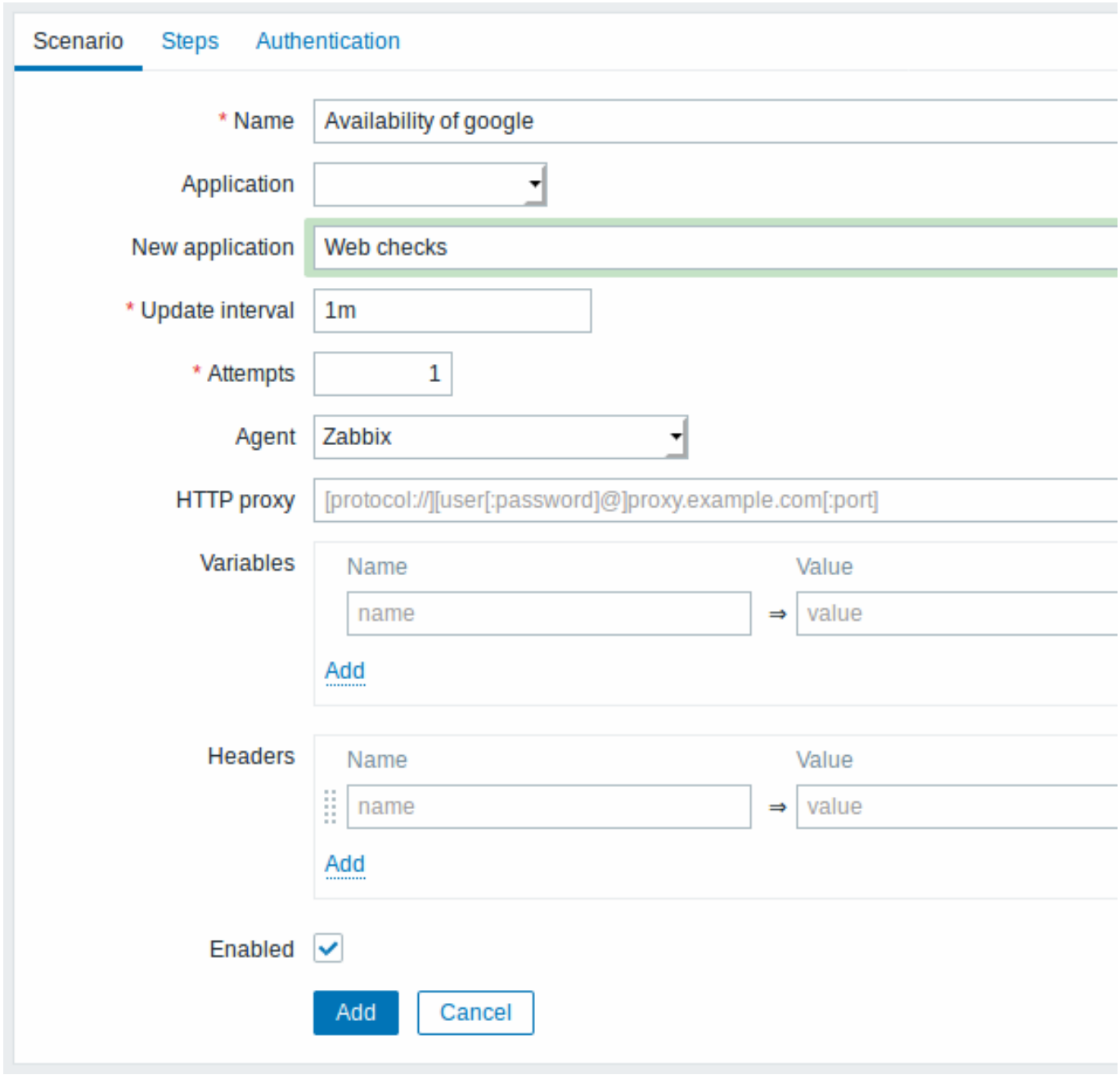
Zabbix web 监控同时支持 HTTP 和 HTTPS。当运行 web 场景时，Zabbix 将选择跟踪重定向（请参见下面的选择跟踪重定向）。重定向硬编码的最大数量为 10 （使用 cURL 选项 CURLOPT_MAXREDIRS)。在执行 web 场景时，所有 Cookie 都会保存。

web 监控使用 HTTPS 协议请参阅已知问题。

配置 **Web** 场景　配置 web 场景：

- 转到：配置 --> 主机（或者 模板）
- 点击主机/模板行中的 *Web*
- 点击右上角 创建场景（或点击场景名字进行编辑现有的场景）
- 在场景的表单中输入参数

场景选项卡允许您配置此 Web 场景的通用参数。



所有必填字段都用红色星号标注。

Note:
许多 Web 场景参数都支持用户宏，但不应在 URL 中使用秘密宏，因为它们会解析为”******” 。

场景参数：

参数	说
主机	场所属的主机名或模板的名字。

参数	说
名称	唯的场景名称。Zabbix 2.2 开始，这个名字支持用户宏和 {HOST.*} 宏 。
应用	选一个场景属于的应用。Web 场景监控项在 监控 → 最新数据 栏中将会分组在选择的应用中。
新的应用	对场景建个新的应用。

参数	说
更新间隔	执行场时间间隔。自 Zabbix 3.4.0 起，支持时间的后缀，例如 30s,1m,2h,1d。自从 Zabbix 3.4.0 开始。支持用户宏。注意，如果使用用户宏变量来改变值（如 5m → 30s），将在下一个执行周期执行更新（在之后的示例中进行更多演示）。

参数	说
重试次数	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)Zabbix 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，默认值为 1。注意:Zabbix 不会因为一个错误的响应代码或者期望的字符串没有出现就会触发这个重试。*Zabbix 2.2* 开始支持此参数。

参数	说
代理	选一个客户端代理。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	您以指定要使用一个 HTTP 代理，使用格式 [protocol://][u 这将设置 CURLOPT_PROXY cURL 选项。可选 protocol:// 前缀可用于指定备用代理协议（协议前缀支持已在 cURL7.21.7 中添加）。如果未指定协议，则该代理将被视为 HTTP 代理。。默认情况下，将使用 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。如果未指定，那么代理将不会覆盖代理相关的环境变量。输入的值是通过”as is”, 不需要进行完整

参数	说
变量	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}**=value1 **{macro2}**=value2 **{macro3}**=regex:<正则表达式>例如：{username}=Alexei {password}=kj3h5k {hostid}=regex:hostis ([0-9]+)然后可以在 {username}，{password} 和 {hostid} 的步骤中引用宏。Zabbix 将自动将其替换为实际值。请注意，使用 `regex:` 的变量：需要一个步骤来获取正则表达式的值，因此提取的值只能应用于后续步骤。如果值部分以 `regex:` 开头，那么它之后的部分将被视为正则表

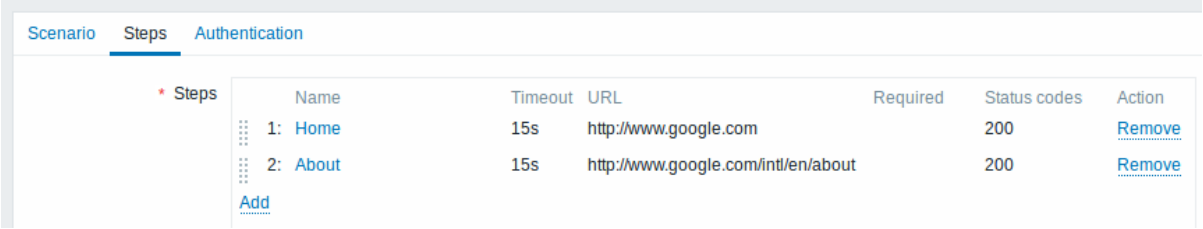
参数	说
头部	请求时将发送的自定义的 HTTP 头部 头部应使用与在 HTTP 协议中出现的语法相同的语法列出，还可选地使用 CURLOPT_HTTPHEADER cURL 选项支持的一些其他功能。 例如：Accept-Charset=utf-8 Accept-Language=en-US Content-Type=application/xml; charset=utf-8 用户宏和 {HOST.*} 宏 和可以在此字段中使用。 从 *Zabbix 2.4* 开始支持指定自定义头。
启用	如选中此复选框，则此场景处于启用状态，否则禁用。

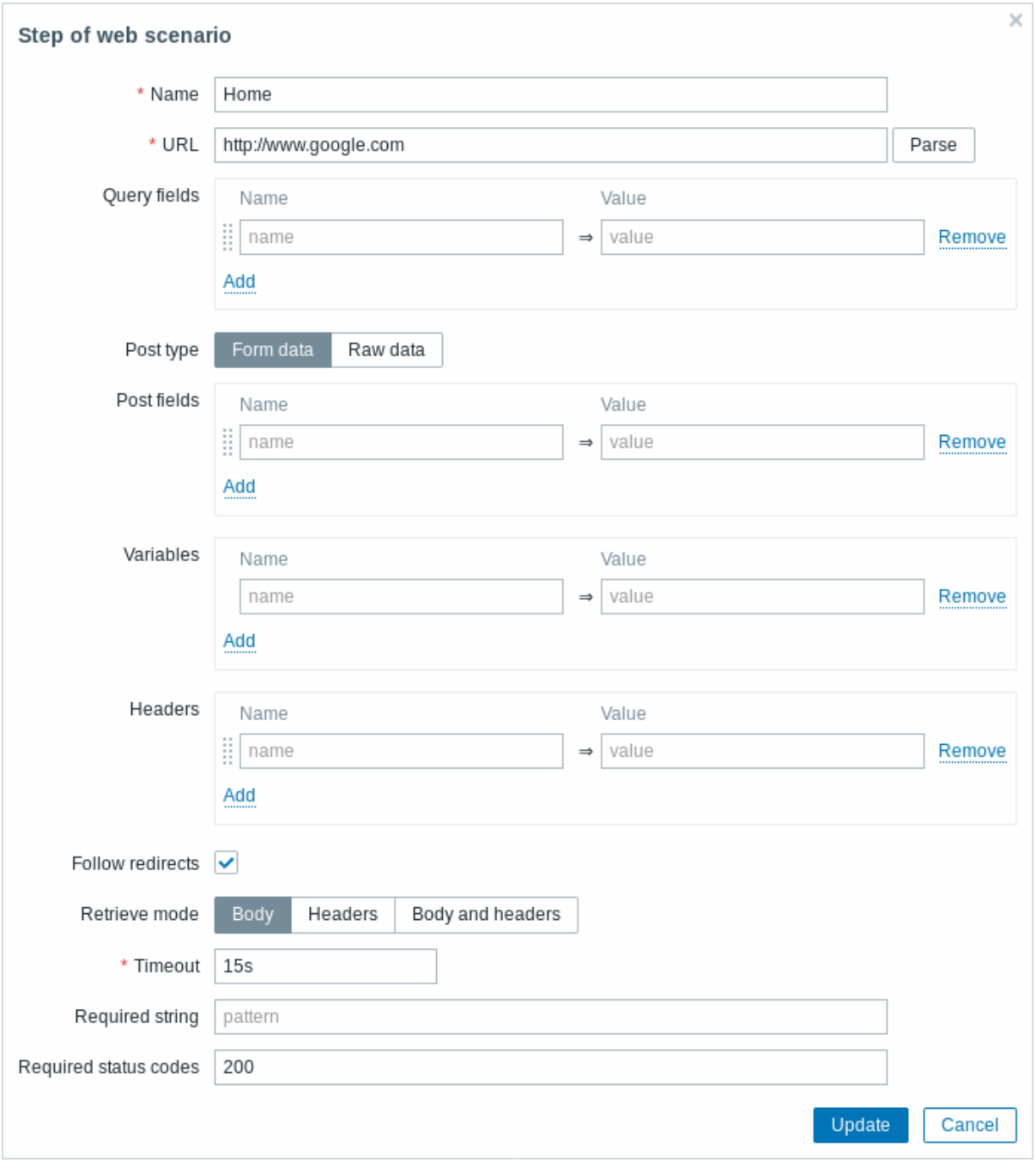
注意，当编辑一个现有的场景时，会出现两个额外的按钮：

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。

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 server 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//。

“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 步骤中单击 添加。





配置步骤

步骤参数：

参数	说
名称	唯

的步骤名称。*Zabbix 2.2* 开始，该名称可以支持用户宏和 {HOST.*} 宏。

参数	说
URL	用于连接和检索数据的网址。例如：%%https://www.google.com%%http://www.zabbix.Zabbix 3.4 以后，可以以 Unicode 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 中分离可选的查询字段（例如？name=Admin&password=mypassword），将属性和值放到查询字段以

参数说	
查询字段 URL	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}.urlencod 语法将对其进行双重 URL 编码。从 *Zabbix 2.2* 开始开始支持用户宏和 {HOST.*} 宏 。

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/值显示在一条线上，并与 & 符号连接。Raw 方式的值可以使用 {{macro}.urlencoc 或 {{macro}.urldecoc 手动进行 URL 编码/解码。例如：id=2345&userid= 如果 {user}

参数	说
变量	可于 GET 和 POST 方法的步骤级变量。指定属性和值。步骤级变量覆盖之前的场景变量或上一步中的变量。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)。它们具有以下格式：**{**宏**}**=值 **{**宏**}**=regex:<正则表达式 > 有关更多信息，请参阅场景 级别上的变量描述。*Zabbix 2.2* 开始支持步骤级变量。

参数	说
报文头执行	求时将发送的自定义 HTTP 头部。指定属性和值步骤级别上的报文头将覆盖为该场景指定的报文头。例如，设置“User-Agent：”为空时，将覆盖在场景上设置的 User-Agent 名称。支持用户宏和 {HOST.*} 宏。这将设置 CURLOPT_HTTPHEADER cURL 选项。*Zabbix 2.4* 开始，支持指定自定义 HTTP 头部。

参数	说
// 跟踪重定向// 选中该复	框以跟踪 HTTP 重定向。将会设置 CURLOPT_FOLLOWLOC cURL 选项。*Zabbix 2.4* 开始支持此选项。
检索模式选择检	模式: 报文体 - 只检索来自 HTTP 响应的报文体 报文头 - 只检索来自 HTTP 响应的报文头 报文体和报文头 - 从 HTTP 响应中检索报文体和报文头 自 *Zabbix 4.2* 开始支持此选项。

参数	说
// 超时时间// Zab	ix 在处理 URL 上所花费的时间不会超过此设置的时间(从一秒钟到 1 小时)。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。支持时间后缀，例如 30s, 1m, 1h。支持用户宏。

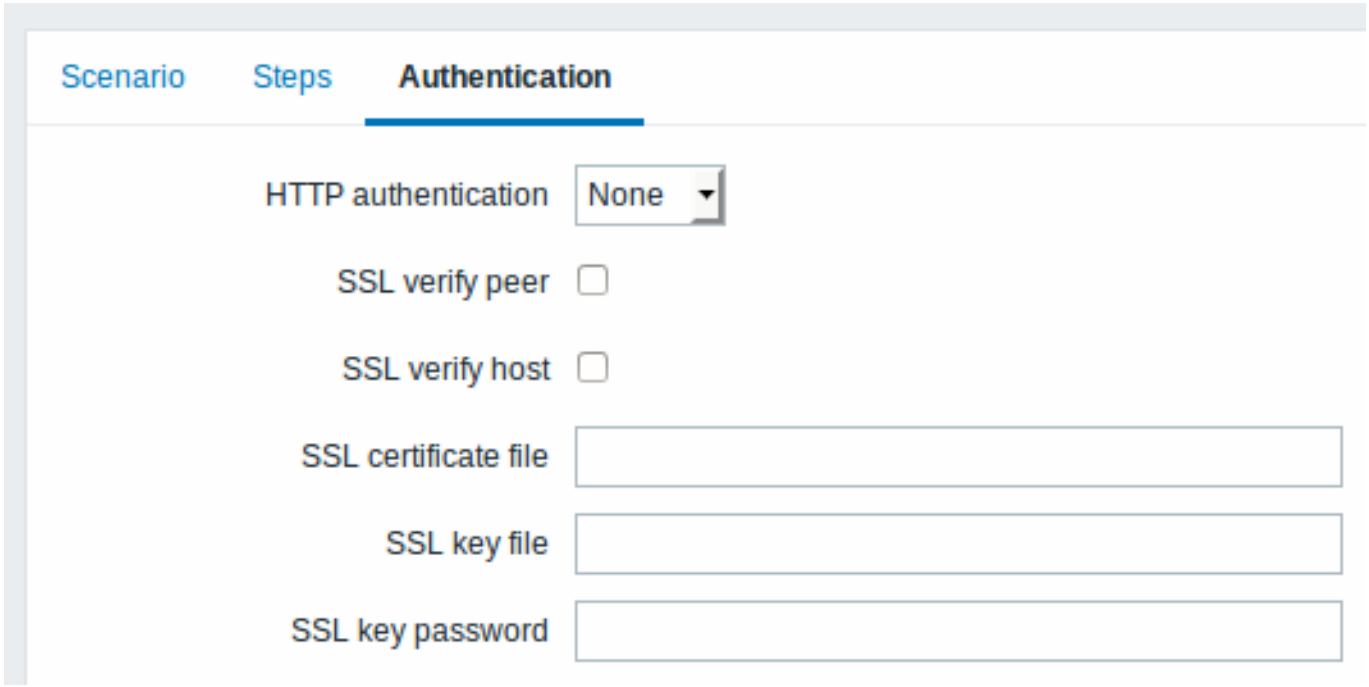
参数	说
// 必需的字符串// 必需的正则	达式。除非检索到的内容(HTML)匹配所需的模式，否则步骤将失败。如果为空，则不执行检查 所需字符串。例如：Zabbix的主页 Welcome.*admin 注意：在此字段中不支持引用 在Zab-bix前端中创建的正则表达式。 *Zabbix 2.2*开始，支持用户宏和{HOST.*}宏

参数	说
必需的状态码	可以设置预的 HTTP 状态代码列表。如果 Zabbix 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，则不执行检查状态码。例如：200,201,210-299 *Zabbix 2.2* 开始，支持用户宏。

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。

另请参见如何配置 Web 监控步骤的实际示例。

配置身份验证　身份验证选项卡允许您配置场景身份验证选项。


Scenario
Steps
Authentication
HTTP authentication
None
SSL verify peer
SSL verify host
SSL certificate file
SSL key file
SSL key password

认证参数：

参数	说
认证认	选项。 **None** - 未使用身份验证。 **Basic** - 使用基本认证。 **NTLM** - 使用 NTLM (Windows NT LAN Manager) 身份验证。 **Kerberos** - 使用 Kerberos 认证。可参考: Zabbix Kerberos 配置。 选择身份认证方法将提供两个附加字段，用于输入用户名和密码。从 *Zabbix 2.2* 开始，用户宏可以在用户和密码字段中使用。

参数	说
SSL 验证对等方	选中复选以验证 Web 服务器的 SSL 证书。服务器证书将自动从系统的证书颁发机构 (CA) 位置获取。您可以使用 Zabbix server 或代理配置参数SSLCALocation覆盖 CA 文件的位置。这将设置 CURLOPT_SSL_VERIFYP cURL 选项。*Zabbix 2.4* 开始支持此选项。

参数	说
SSL 验证主机选中复	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_VERIFYH cURL 选项。*Zabbix 2.4* 开始支持此选项。

参数	说
//SSL 证书文件 // 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* 密钥文件字段留空。如果密钥加密，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 或代理配置参数SSLCertLocation指定。`HOST.*` 宏和用户宏可以在此字段中使用。这将会设置 CURLOPT_SSLCERT cURL 选项。

参数	说
SSL 密钥文件用于客	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 Zabbix server 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 CURLOPT_SSLKEY cURL 选项。*Zabbix 2.4* 开始支持此选项。

参数说	
SSL 密钥密码 SSL	钥文件密码。用户宏可以在此字段中使用。这将设置 CURLOPT_KEYPASSWD cURL 选项。*Zabbix 2.4* 开始支持此选项。

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
```

Note:
Zabbix server 对证书的更改无需重启。

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
```

展示　要查看为主机配置的 Web 场景，请转至 监控 → 主机，在列表中找到主机，然后单击最后一列中的 *Web* 超链接。单击场景名称以获取详细信息。

Web 场景的概览也可以通过一个 Web 监控小部件显示在监控 → 仪表盘中。

Web 场景执行的最新结果在 监控 → 最新数据中可用。

扩展监控　有时需要记录接收的 HTML 页面内容。如果某些 Web 场景步骤失败时，这将非常有用。调试级别 5（跟踪）用于此目的。此级别可以在服务器 和代理 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）来设置此级别。以下示例说明如果调试级别 4 已设置，扩展监控如何启动：

```
提高所有http轮询器的日志级别：
shell> zabbix_server -R log_level_increase="http poller"

提高第二个http轮询器的日志级别：
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` 选项来停止。

1 Web 监控项

概述

在创建 Web 场景时，会自动添加一些新监控项以进行监控。

场景监控项

创建场景后，Zabbix 会自动添加以下监控项用以监控，并将它们链接到所选的应用上。

监控项描述	
// 场景的下载速度 // 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：数值型(浮点数)

监控项描述	
// 场景的失败步骤 // 此监控项将显	场景上失败步骤的编号。如果所有步骤成功执行，则返回0。监控项key: web.test.fail[Sce 类型：数值型*(无符号)*

监控项描述	
// 场景的最近错误消息 // 此监控项返回场景	最近一个错误消息文本。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[Sc 类型：字符型

将使用实际场景名称代替 “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:
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 key 中正确引用。在其他情况下，不会执行额外的引用。

这些监控项可用于创建触发器和定义通知条件。

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

要创建一个 “Web 场景失败” 触发器，并在触发器名称中提供有用的问题描述，可以定义一个名称为：

```
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 应用运行慢” 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。

场景步骤监控项

创建场景后，Zabbix 会自动添加以下监控项用以监控，并将它们链接到所选的应用上。

监控项	描述
// 场景 <Scenario> 的步骤 <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：数值型(浮点数)

监控项描述
// 场景 <Scenario> 的步骤 <Step> 的响应时间 // 此监控项将收集有关

骤响应时间的信息(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：数值型(浮点数)

监控项描述	
// 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：数值型 *(无符号)*

将分别使用实际场景和步骤名称而不是 “Scenario” 和 “Step”。

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:
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 key 中正确引用。在其他情况下，不会执行额外的引用。

这些监控项可用于创建触发器和定义通知条件。例如，创建一个 “Zabbix GUI 登录太慢” 触发器，你可以定义一个触发器表达式：

```
{zabbix:web.test.time[ZABBIX GUI,Login,resp].last()}>3
```

2 真实场景监控

概述

本节提供了有关如何使用 Web 监控的每一步实际示例。

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、是否正常工作以及其响应速度。为此，我们还必须使用我们的用户名和密码登录。

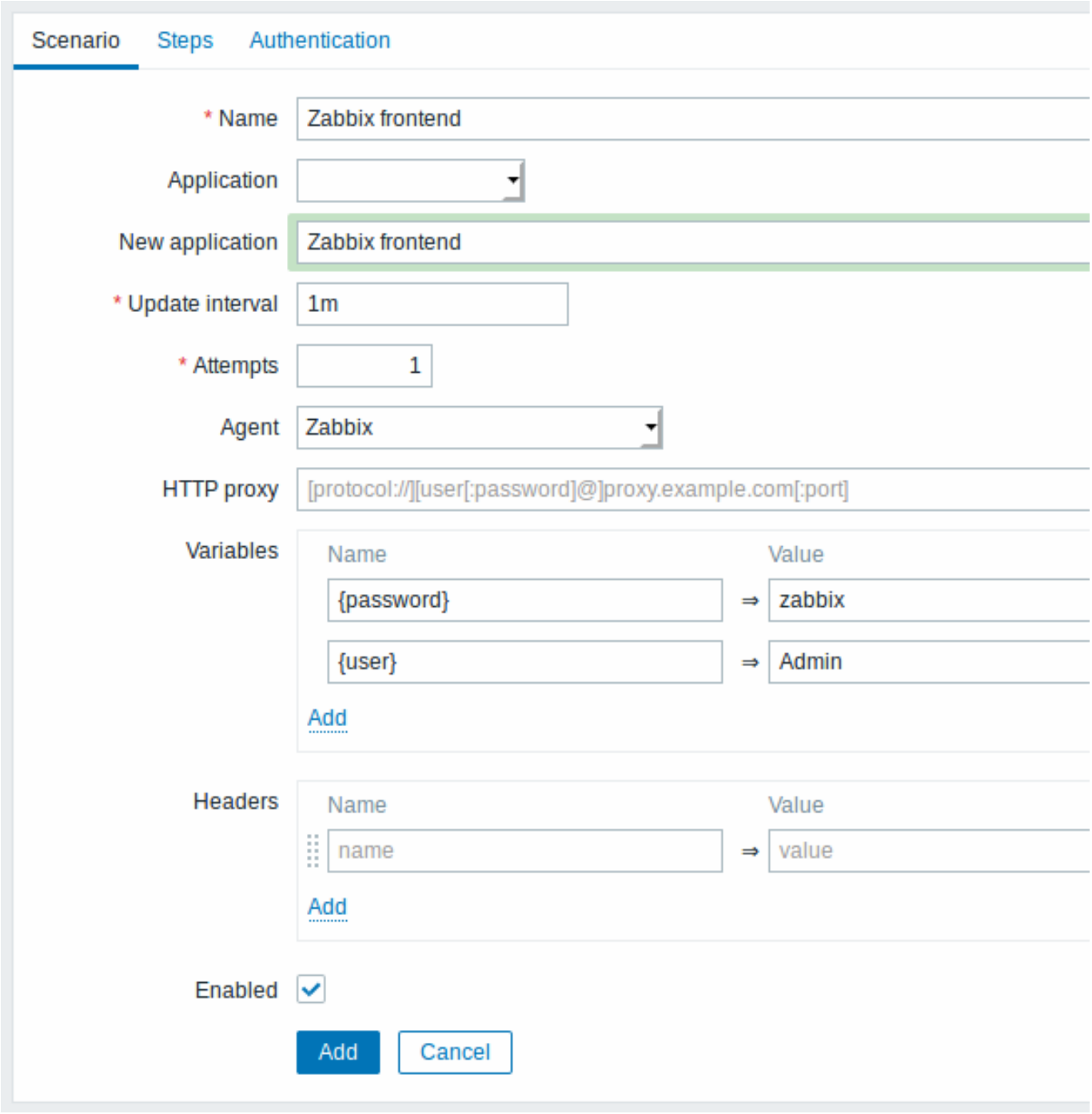
场景

第 1 步

创建新的 Web 场景。

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。

点击 // 配置 → 主机//，选择一个主机，然后在该主机行中单击 *Web*。然后单击 创建 *Web* 场景。



所有必填字段均有红色星号标记。

在新的场景中，我们将场景命名为 *Zabbix* 前端，并为其创建一个新的 *Zabbix* 前端应用。

注意，我们还需要创建两个变量：{user} 和 {password}。

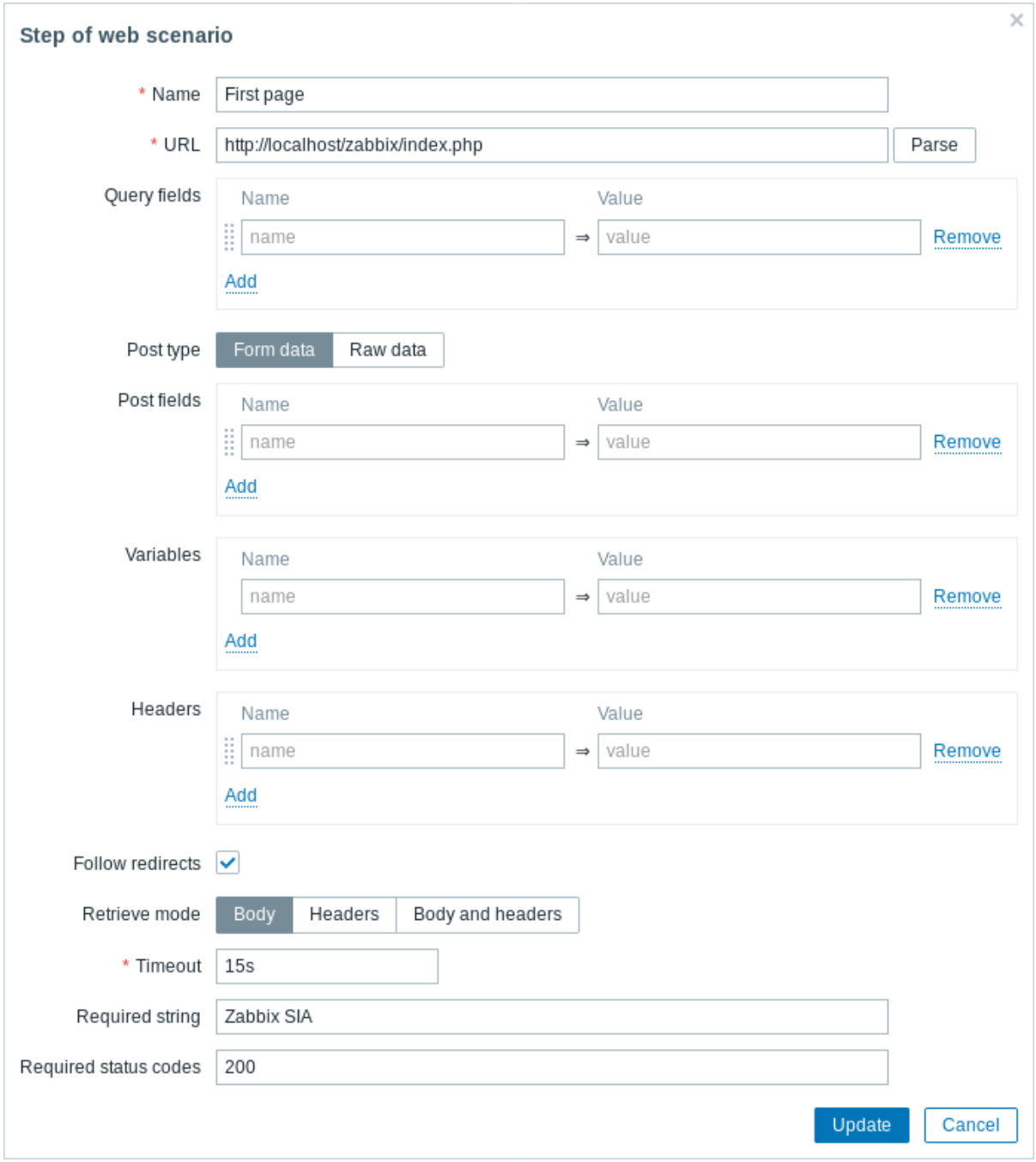
第 2 步

定义场景的步骤

单击 //步骤 //选项卡中的 // 添加 // 按钮添加各个步骤。

Web 场景步骤 *1*

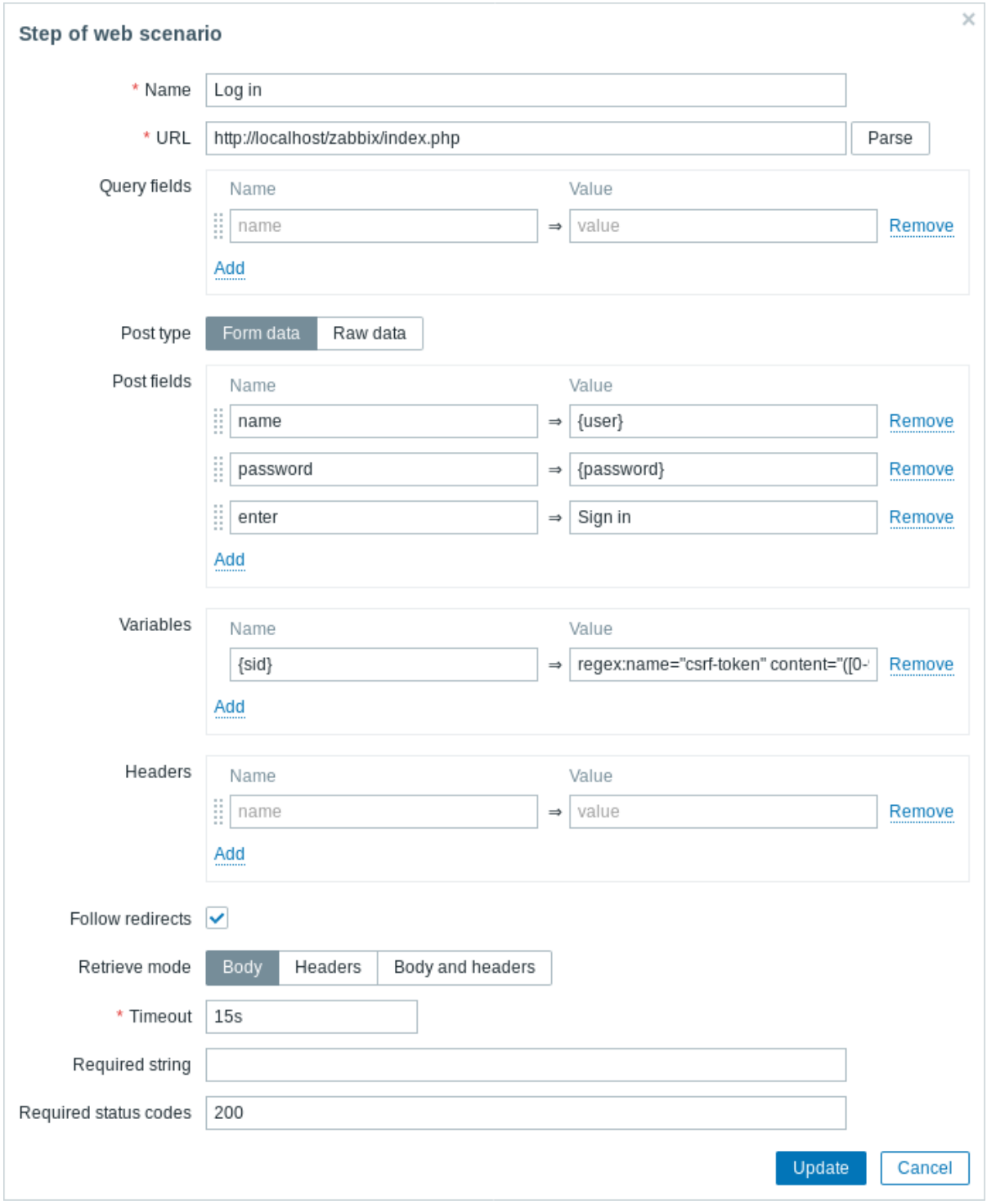
我们首先检查第一页响应是否正确响应，返回 HTTP 响应代码 200，并包含文本 “Zabbix SIA”。



完成配置步骤后，单击 // 添加 //。

Web 场景步骤 *2*

我们继续登录 Zabbix 前端，并通过重用在场景级别定义的宏（变量）—{user} 和 {password} 来进行操作。

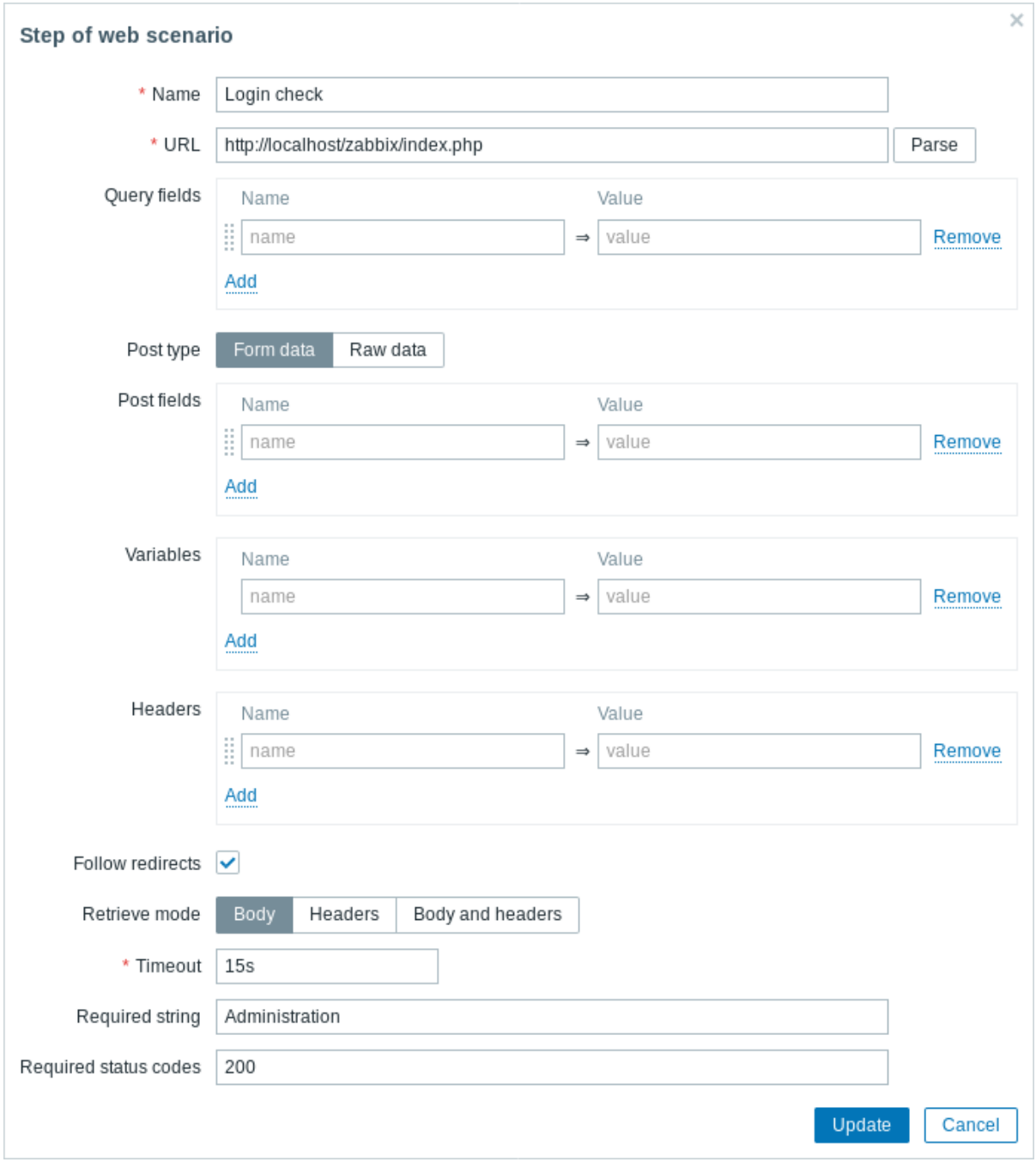


Attention:
注意，Zabbix 前端在登录时使用 JavaScript 重定向，因此首先我们必须登录，只有在下一步的步骤中，我们才能检查登录功能。此外，登录步骤必须使用完整的 URL 以获取 **index.php** 文件

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。

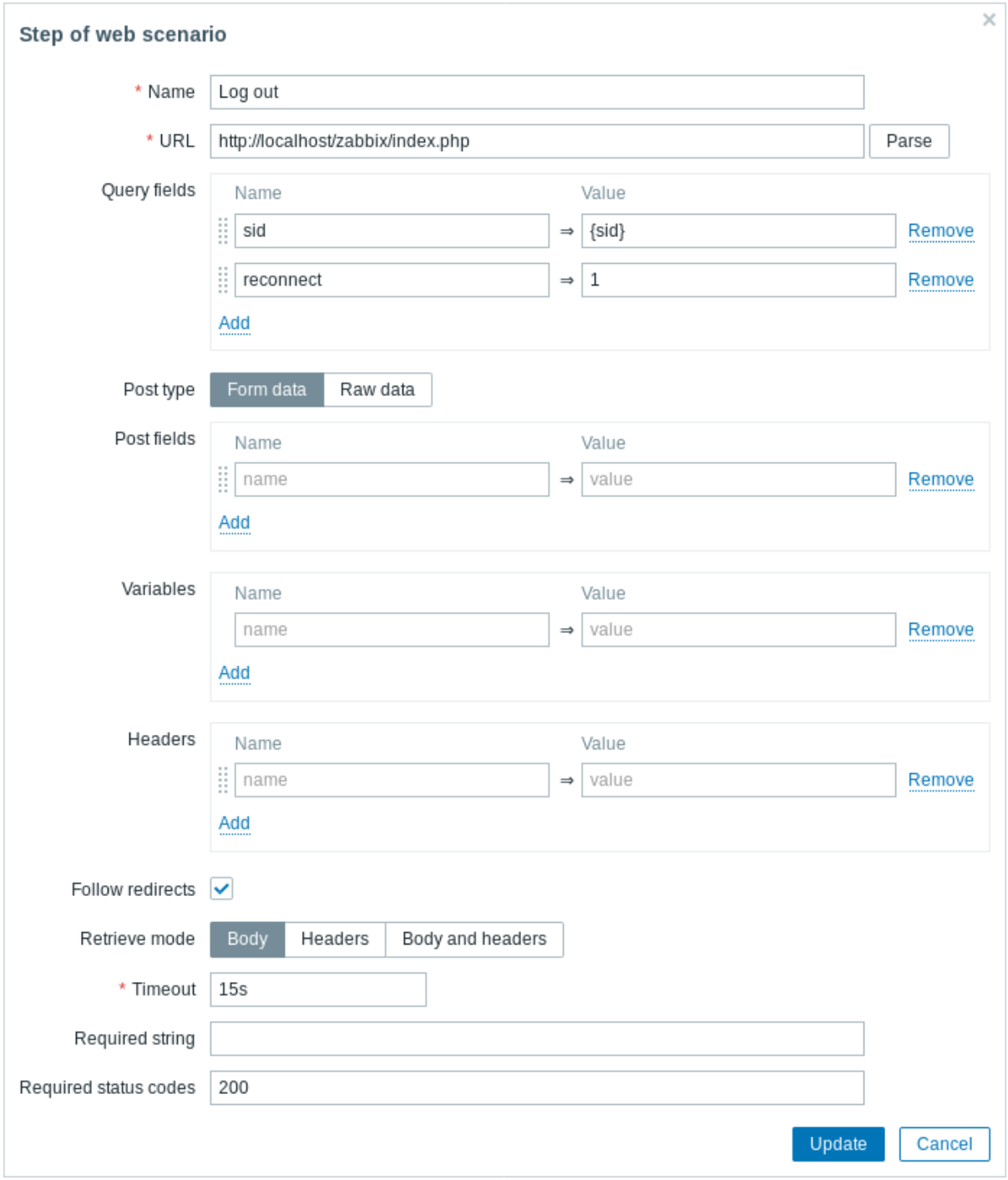
Web 场景步骤 *3*

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**。



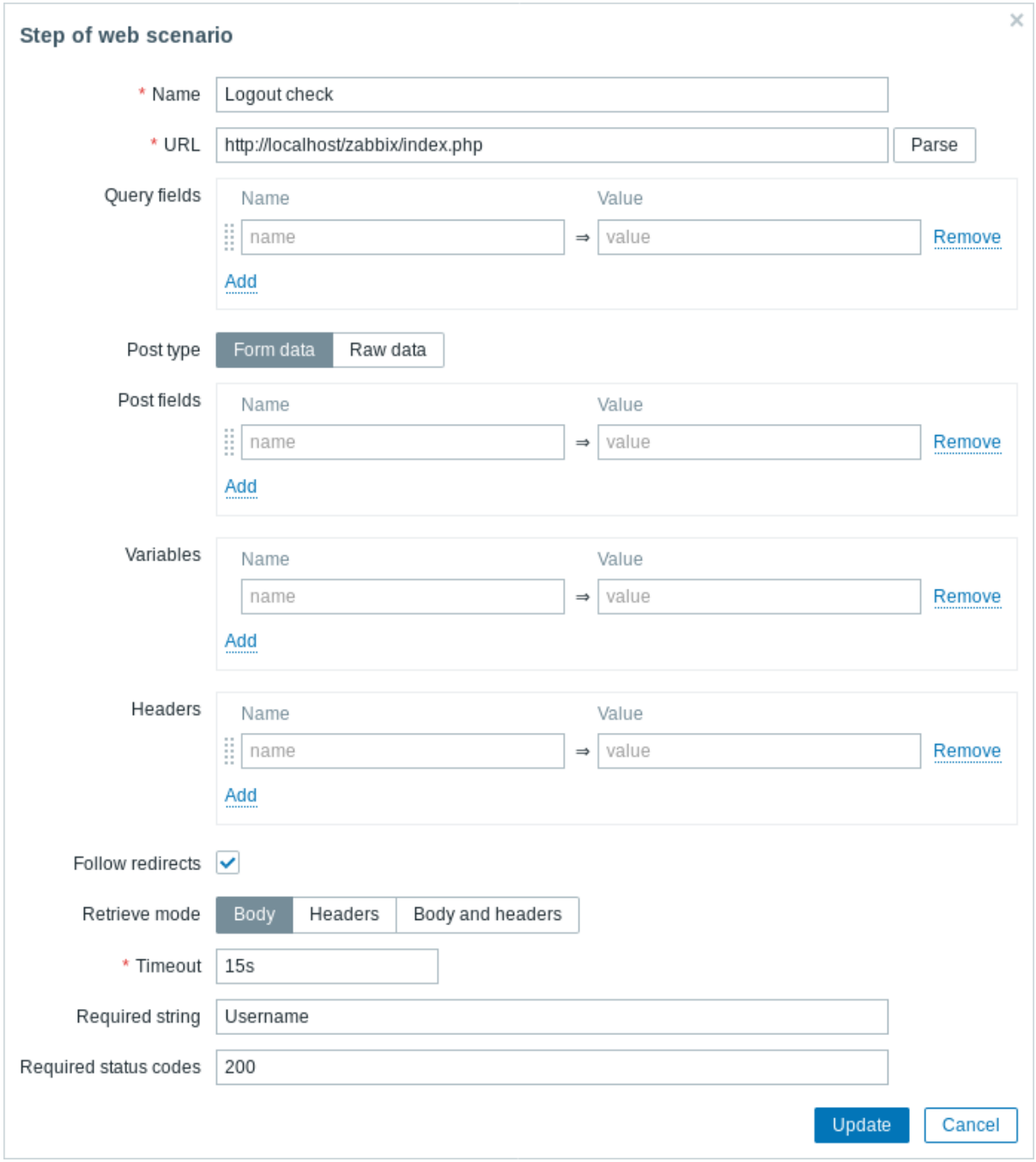
Web 场景步骤 *4*

现在我们已经验证了前端是可访问的，并且我们可以登录并检索登录的内容，我们也应该注销 - 否则 Zabbix 数据库将将被大量打开的会话记录占用资源。



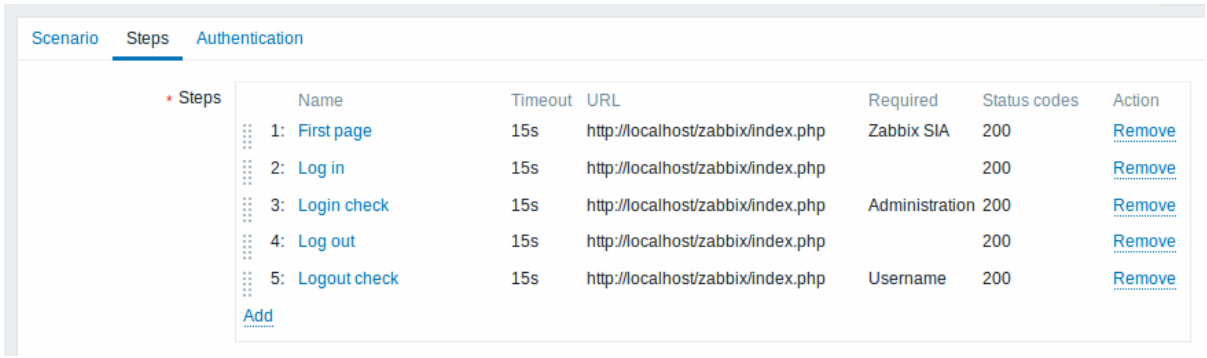
Web 场景步骤 *5*

我们可以通过查找 **Username** 字符串来检查是否已经注销。



// 完成步骤配置 //

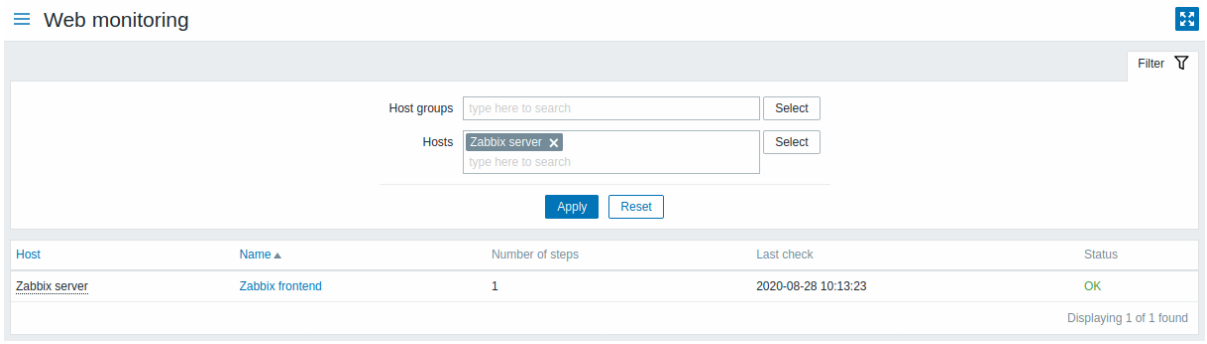
Web 场景步骤的完整配置应如下所示：



第 3 步

保存配置完成的 Web 监控场景。

该场景将被添加到主机。要查看 Web 场景信息，请转至监控 → 主机，在列表中找到主机，然后单击最后一列中的 Web 超链接。



单击场景名称以查看更详细的统计信息：

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 可以根据事先定义的主机原型使用低级别自动发现发现 Vmware 宿主机和虚拟机，并为其创建主机并添加监控。

Zabbix 中默认提供了几个开箱即用模板，可以直接用来监控 VMware vCenter 或 ESX hypervisor。

支持 VMware vCenter 或 vSphere 版本最低为 5.1。

详情　监控虚拟机分两个步骤完成。首先，Zabbix 是通过 *vmware* 收集器进程来获取虚拟机数据。这些进程通过 SOAP 协议从 VMwareWeb 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 轮询器通过 zabbix 简单检查VMware 密钥来检索这些数据。

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 性能计数器数据。这两种类型都由 *vmware* 收集器进程独立收集。因此，建议启用比受监控的 VMware 服务更多的收集器。否则，VMware 性能计数器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 性能计数器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，应该使用 --with-libxml2 和 --with-libcurl 编译选项来编译Zabbix。

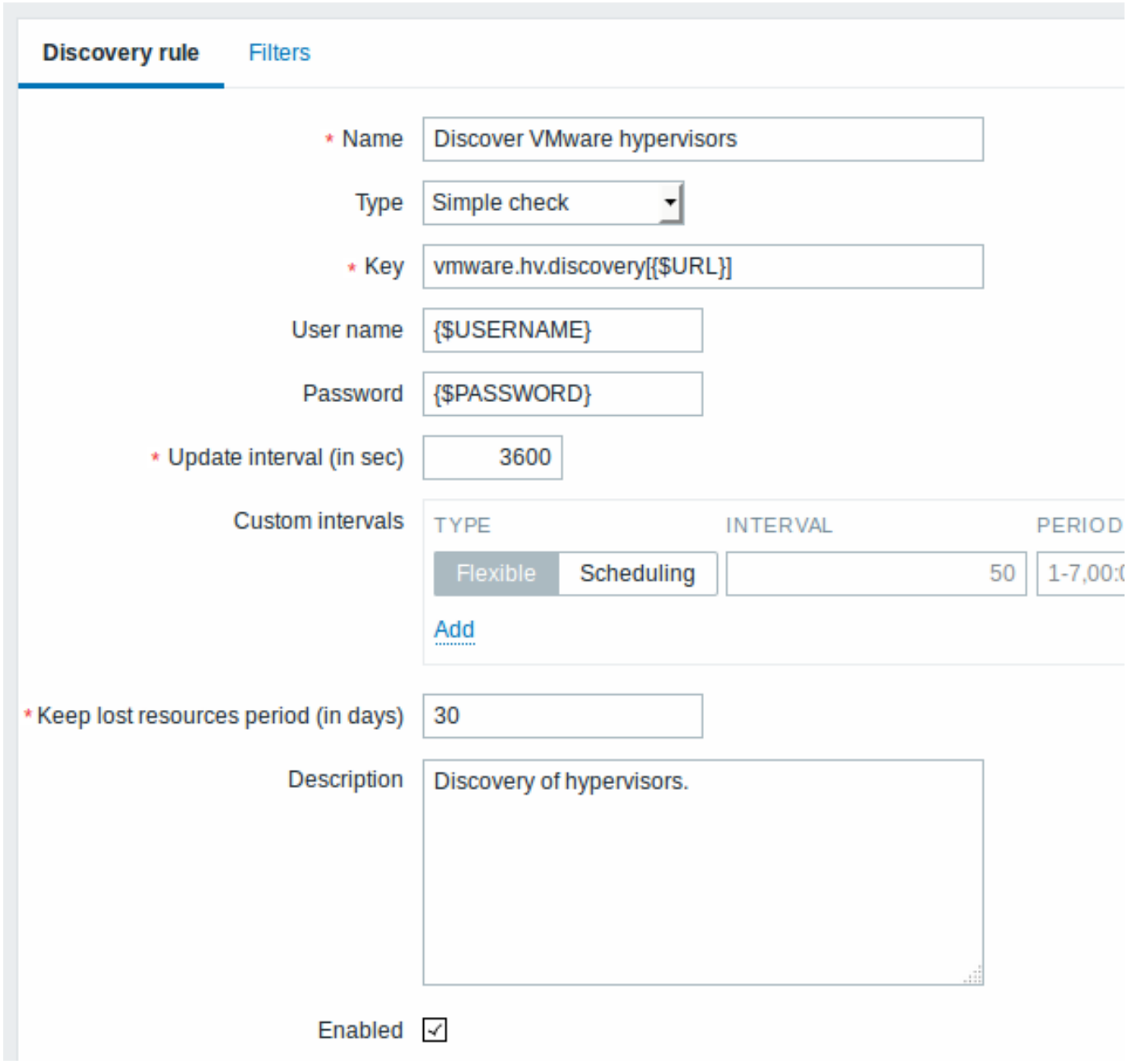
以下配置文件参数可用于调整虚拟机监控：

- **StartVMwareCollectors** - 预先启动 Vmware 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，并且不应大于您监控的 VMware 服务数量的 2 倍。还要记住，此值还取决于 VMware 环境大小及 *VMwareFrequency* 和 *VMwarePerfFrequency* 配置参数（请参阅下文）。
- **VMwareCacheSize**
- **VMwareFrequency**
- **VMwarePerfFrequency**
- **VMwareTimeout**

有关更多详细信息，请参阅 zabbix 服务器 和代理的配置文件页面。

<note important> 为了支持数据存储容量指标，Zabbix 要求 VMware 配置 vpxd.stats.maxQueryMetrics 参数至少为 64。另请参见 VMware 知识库文章 :::

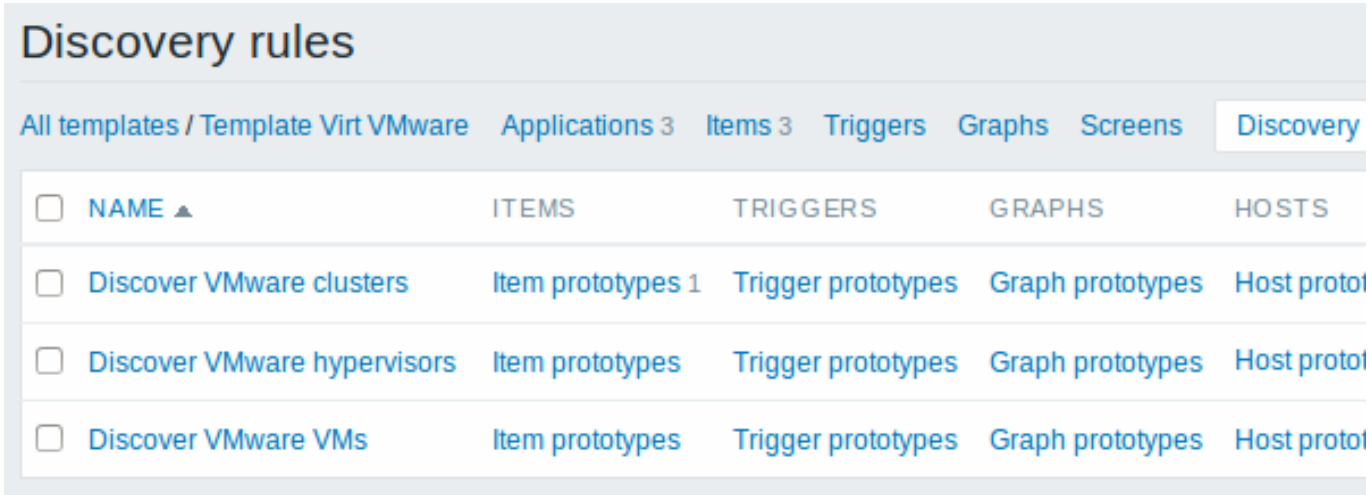
发现　Zabbix 可以使用低级别发现规则自动发现 VMware 宿主机和虚拟机。



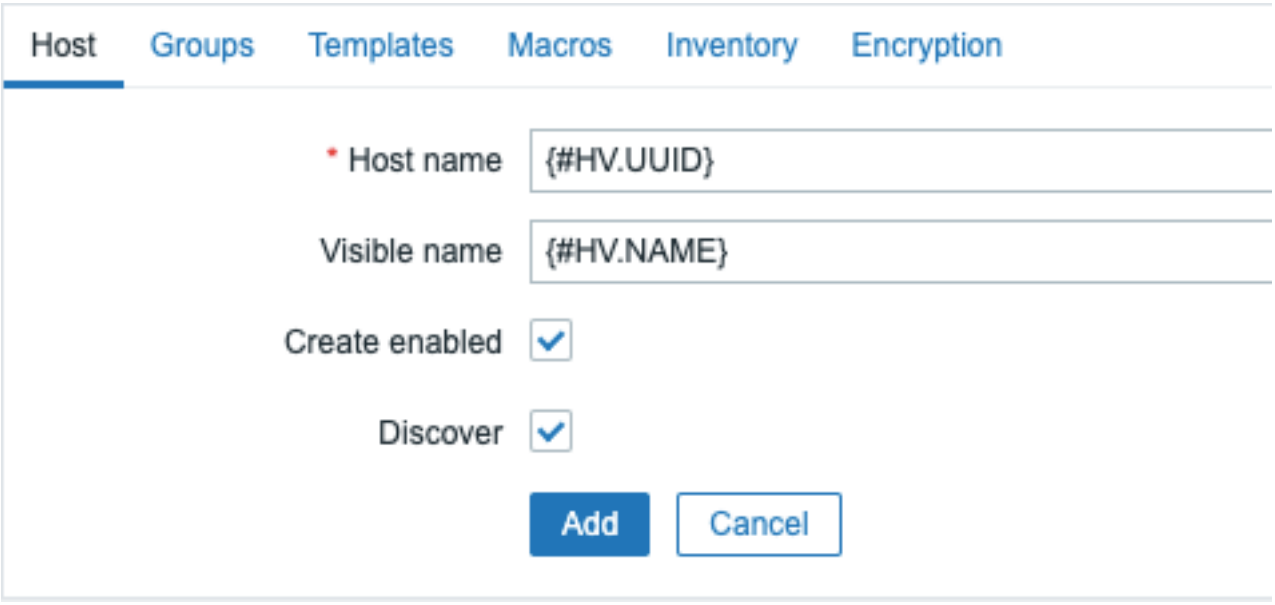
所有必填项输入字段均标有红色星号。

以上截图中的发现规则 key 是 *vmware.hv.discovery[{$URL}]*。

主机原型　可以使用低级别发现规则来创建主机原型。当发现虚拟机时，这些原型会成为真实主机。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。发现的主机将属于现有主机，并将根据获取的已有主机 IP 进行主机配置。



在主机原型配置中，低级别发现宏用于主机名，显示名称和主机组原型字段。也可以使用低级别发现宏将用户宏定义为值。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 创建启用，则主机将添加为启用状态。如果未选中，将添加主机，但处于禁用状态。

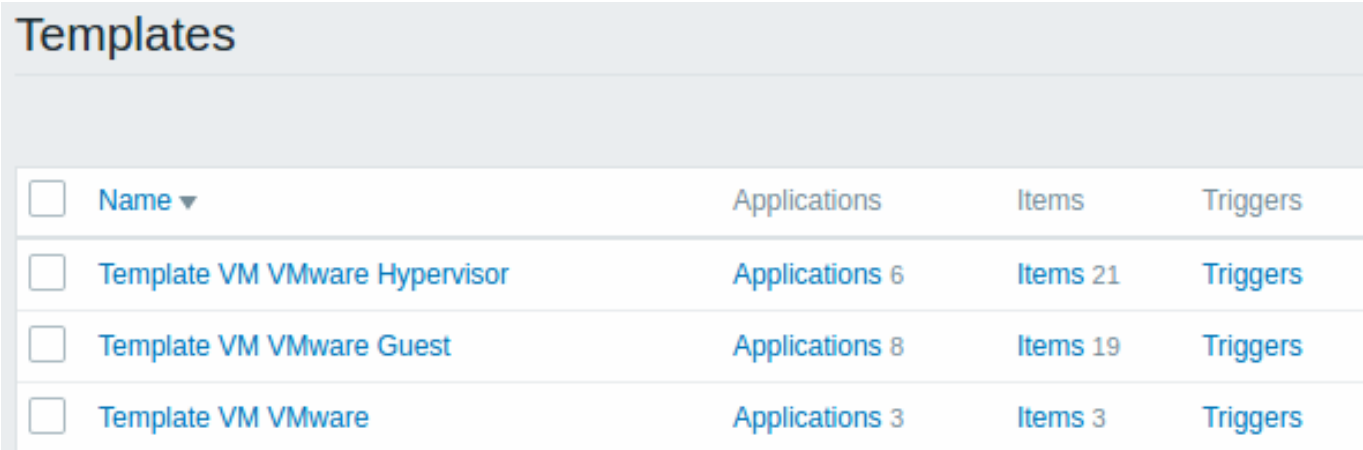
如果选中“发现”（默认），将创建主机。如果未选中，则除非在发现规则中覆盖此设置，否则不会创建主机。创建发现规则时，此功能提供了更多的灵活性。

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 // 保留丢失资源期限（以天为单位）// 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低级别发现规则以及用于监视虚拟安装的内置检查。注意：

- ” 模板 *VM VMware*” 模板应用于 VMware vCenter 和 ESX hypervisor 监控；
- 发现使用” 模版 *VM VMware Hypervisor*” 和” 模版 *VM VMware Guest*”，通常不应手动将其链接到主机。



Note:
如果您的服务器从 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。然而这些模板依赖于 *VMware VirtualMachinePowerState* 和 //VMware status // 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 或更大
- 创建一个新主机
- 设置监控 VMware 服务所需的身份验证相关的主机宏：

```
    {{..:..:assets:zh:manual:vm_monitoring:vm_host_macros.png|}}
* 将主机链接到VMware服务模板：
    {{..:..:assets:zh:manual:vm_monitoring:vm_host_templates.png|}}
* 单击 //添加// 按钮保存主机
```

扩展日志　可以使用调试级别 5 记录由 VMware 收集器收集的数据，以进行详细调试。此级别可以在服务器 和代理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）设置此级别。以下示例说明如果将调试级别设置为 4，如何启动扩展日志记录：

```
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

- 如果指标不可用，请确保在最新的 VMware vSphere 版本中默认情况下它们是否不可用或未关闭，或者性能指标数据库查询未设置某些限制。有关其他详细信息，请参见ZBX-12094
- 如果*'config.vpxd.stats.maxQueryMetrics'* 无效或超过允许的最大字符数 ** 错误，请在 vCenter 服务器设置中添加一个 `config.vpxd.stats.maxQueryMetrics` 参数。此参数的值应与 VMware's *web.xml* 中 `maxQuerysize` 的值相同。有关详细信息，请参见此 VMware 知识库文章。

1 虚拟机发现 **key** 字段

下表列出了虚拟机发现 key 返回的内容。

监控项 **key**		
描述 *	字段 ** ** 检	内容 **
vmware.cluster.discovery 执行集群发现。{#CLU	TER.ID} 集群 ID。	{#CLUSTER.NAME} 集群名称。
vmware.datastore.discovery 执行数据存储发现。{#DATAS	ORE} 数据存储名称。	
vmware.dc.discovery 执行数据中心发现（自 Zabbix5.0.3 起）。{#DATACENTE	} 数据中心名称。	{#DATACENTER.ID} 数据中心 ID。
vmware.hv.discovery 执行宿主机发现。{#HV.U	ID} 唯一的宿主机 I	

监控项 **key**	
	{#HV.ID} 宿主机的 ID (由 Host-System 管理对象名称)
	{#HV.NAME} 宿主机的名字
	{#CLUSTER.NAME} 群集名称，可能为空。
	{#DATACENTER.NAME} 数据中心名称。
	{#PARENT.NAME} 在宿主机上存储的容器名称 自 Zabbix4.0.3 开始支持
	{#PARENT.TYPE} 在宿主机上存储的容器类型。此值可能为 Datacenter, Folder, ClusterComputeRVmware，当值为 “VMware” 时，代表未知容器类型。自 Zabbix4.0.3 开始支持
vmware.hv.datastore.discovery 执行宿主机数据存储发现。请注意，多个宿主机可以使用相同的数据存储。{#DATASTORE} 数据存储名称。	
vmware.vm.discovery 执行虚拟机发现。{#VM.U	ID} 唯一虚拟机 ID

监控项 **key**		
	{#VM.ID}	虚拟机ID（由Vir-tual-Ma-chine管理对象名称）。
	{#VM.NAME}	虚拟机名称。
	{#HV.NAME}	宿主机名称。
	{#CLUSTER.NAME}	集群名称，可能为空。
	{#DATACENTER.NAME}	数据中心名称。
vmware.vm.net.if.discovery 执行虚拟机网络接口发现。{#IFNAME}	网络接口名称。	
vmware.vm.vfs.dev.discovery 执行虚拟机磁盘设备发现。{#DISKNAME	磁盘设备名称。	
vmware.vm.vfs.fs.discovery 执行虚拟机文件系统发现。{#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周期。

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。

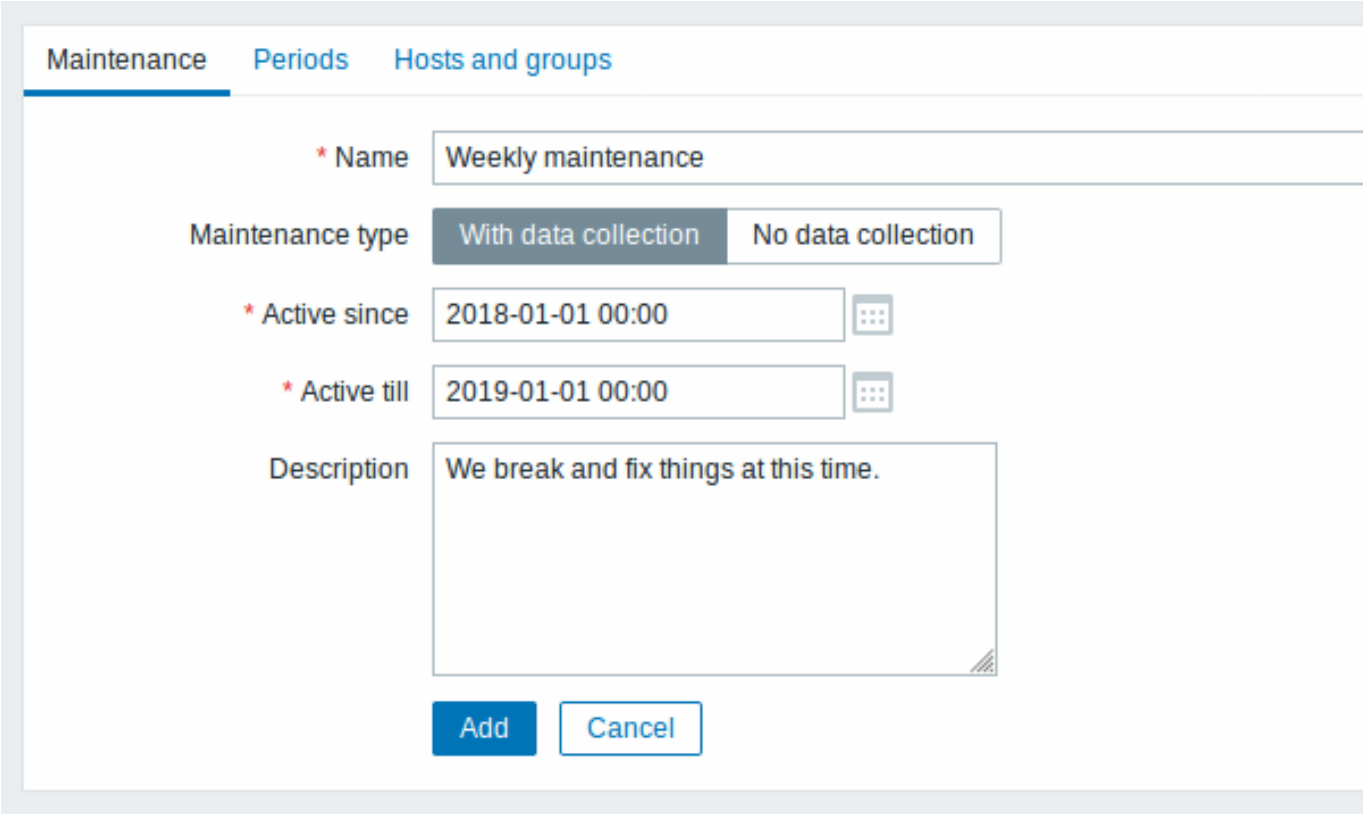
如果主机处于“无数据收集”维护类型期间，此时给它发送一个带时间戳的值（例如，使用Zabbix sender），那么这个值会被丢弃。然而，如果主机的维护期过期了，此时给它发送一个带有时间戳的值，是会被接收的。

<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。

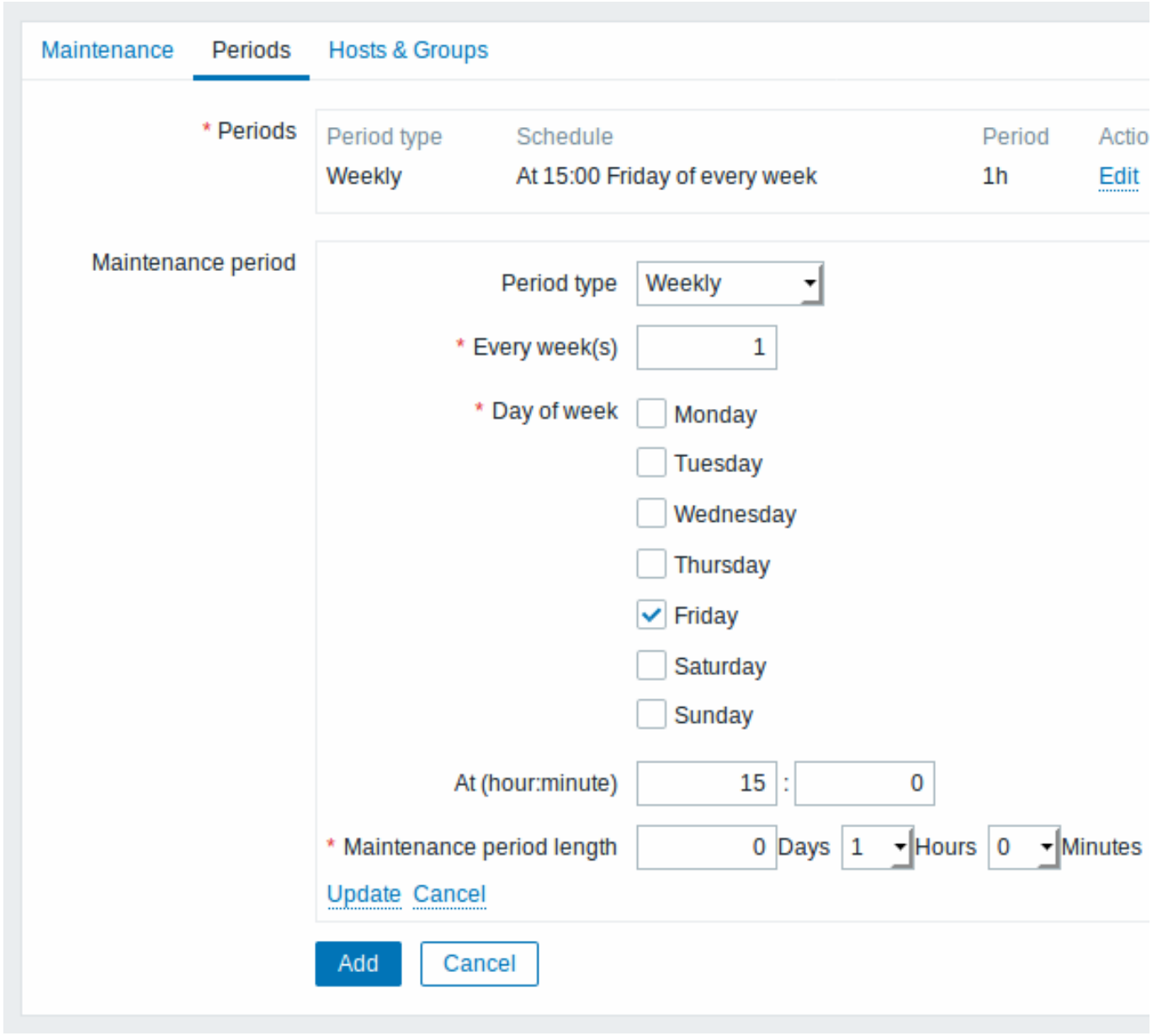
参数说	
名称（*Name*）维护期	的名称。

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。

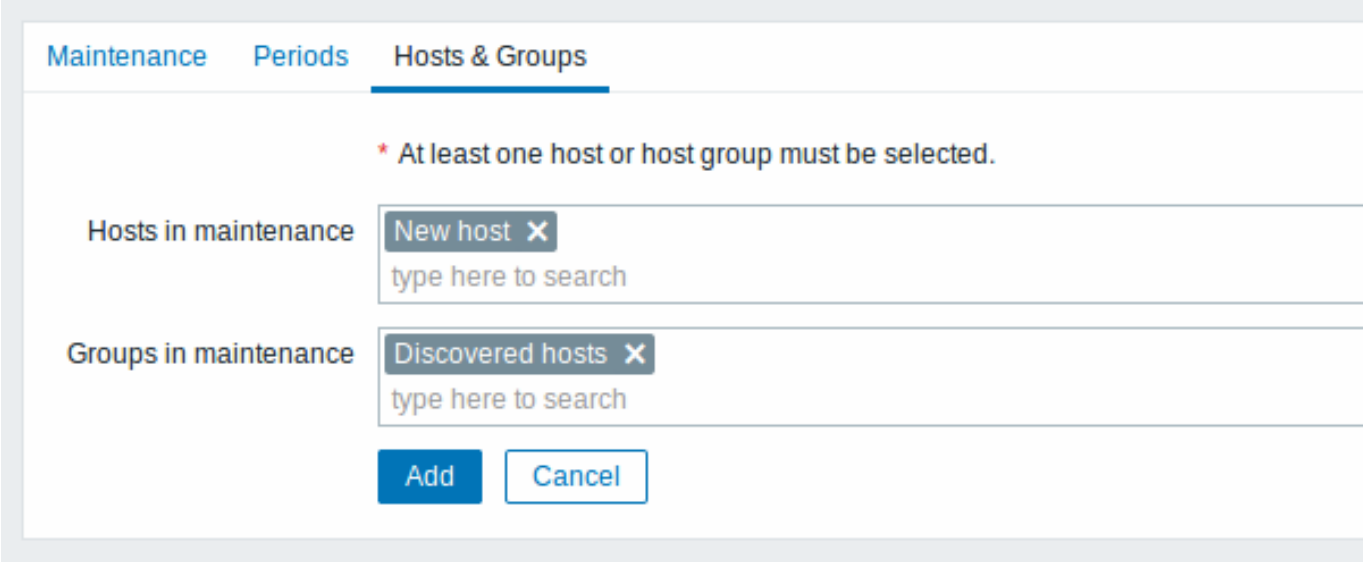
周期（**Periods**）选项卡允许您定义维护发生的确切天数和小时数。单击 新建（*New*）会打开一个 维护周期（*Maintenance period*）表单，可灵活配置维护期间的时间段 - 每天、每周、每月或者仅一次。



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 自启用时间起作用时的日期或星期。

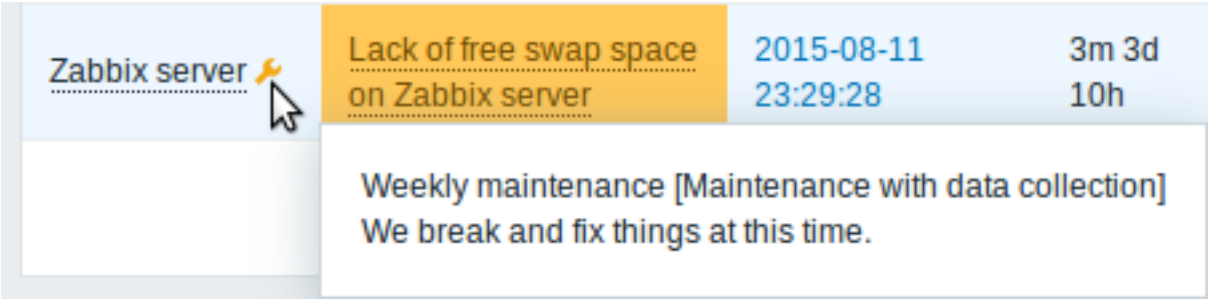
比如，启用自从（*Active since*）设置为 2013-09-06 12:00，如果有一个在 23:00 开始的为期一个小时的维护期间，每两天执行一次，那么第一次维护期间将会开始于 2013-09-06 23:00，第二次维护期间开始于 2013-09-08 23:00。或者，再举个例子，如果还是那个相同的启用自从（*Active since*），每两天执行一次，每次一小时，开始时间设定为 01:00，那么，第一次维护期间将开始于 2013-09-08 01:00，第二次开始于 2013-09-10 01:00。

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。



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。

显示 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。



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。

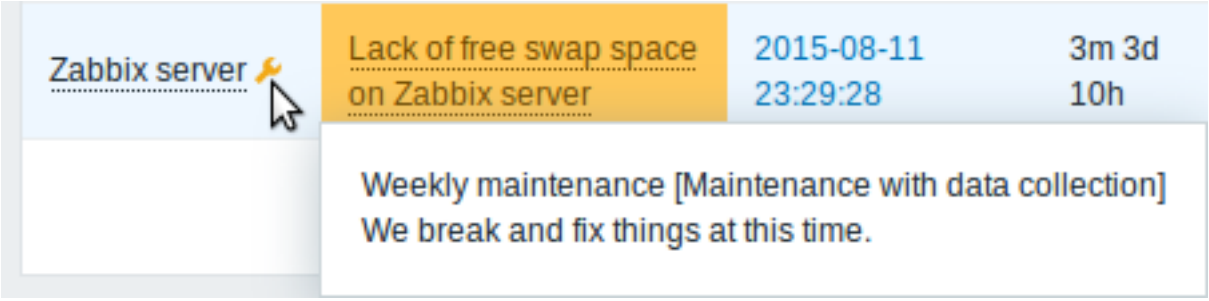
Note:
可以使用仪表板过滤功能完全取消显示仪表板中处于维护状态的主机。

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）”。

Displaying hosts in maintenance

An orange wrench icon next to the host name indicates that this host is in maintenance in:

- *Monitoring* → *Dashboard*
- *Monitoring* → *Problems*
- *Inventory* → *Hosts* → *Host inventory details*
- *Configuration* → *Hosts* (See 'Status' column)



Maintenance details are displayed when the mouse pointer is positioned over the icon.

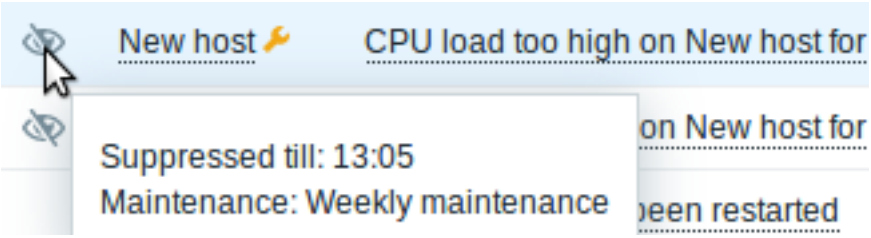
Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps*.

Displaying suppressed problems

Normally problems for hosts in maintenance are suppressed, i.e. not displayed in the frontend. However, it is also possible to configure that suppressed problems are shown, by selecting the *Show suppressed problems* option in these locations:

- *Monitoring* → *Dashboard* (in *Problem hosts*, *Problems*, *Problems by severity*, *Trigger overview* widget configuration)
- *Monitoring* → *Problems* (in the filter)
- *Monitoring* → *Overview* (in the filter; with 'Triggers' as *Type*)
- *Monitoring* → *Maps* (in map configuration)
- Global notifications (in user profile configuration)

When suppressed problems are displayed, the following icon is displayed: . Rolling a mouse over the icon displays more details:



12. 正则表达式

概述　Zabbix 支持 Perl Compatible Regular Expressions (PCRE)。

在 Zabbix 中有两种使用正则表达式的方法：

- 手动输入正则表达式
- 使用在 Zabbix 中创建的全局正则表达式

正则表达式　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。

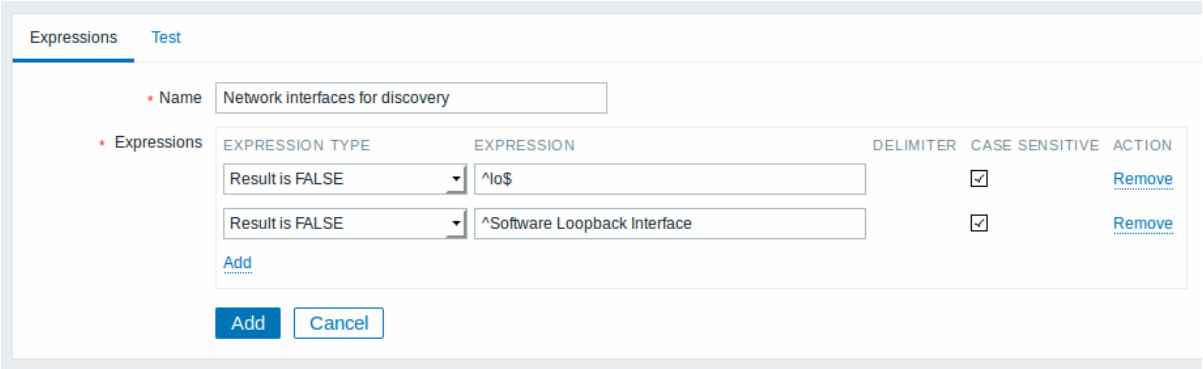
全局正则表达式　在 Zabbix 前端，有一个高级的编辑器用于创建和测试复杂的正则表达式。

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*。

要创建全局正则表达式：

- 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*）

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。



所有必填输入字段都标有红色星号。

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。

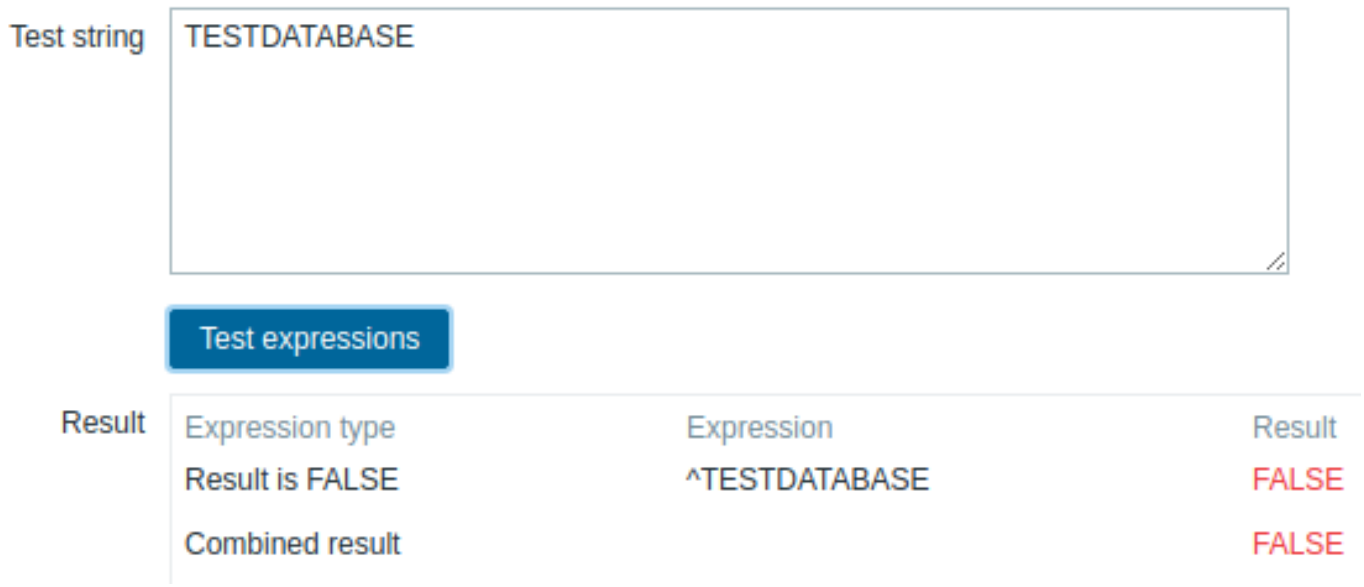
参数	说	
区分大小写（*Case sensitive*）	此复选框用于	定正则表达式是否对字母大小写敏感。

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。

<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）。:::

举例　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：

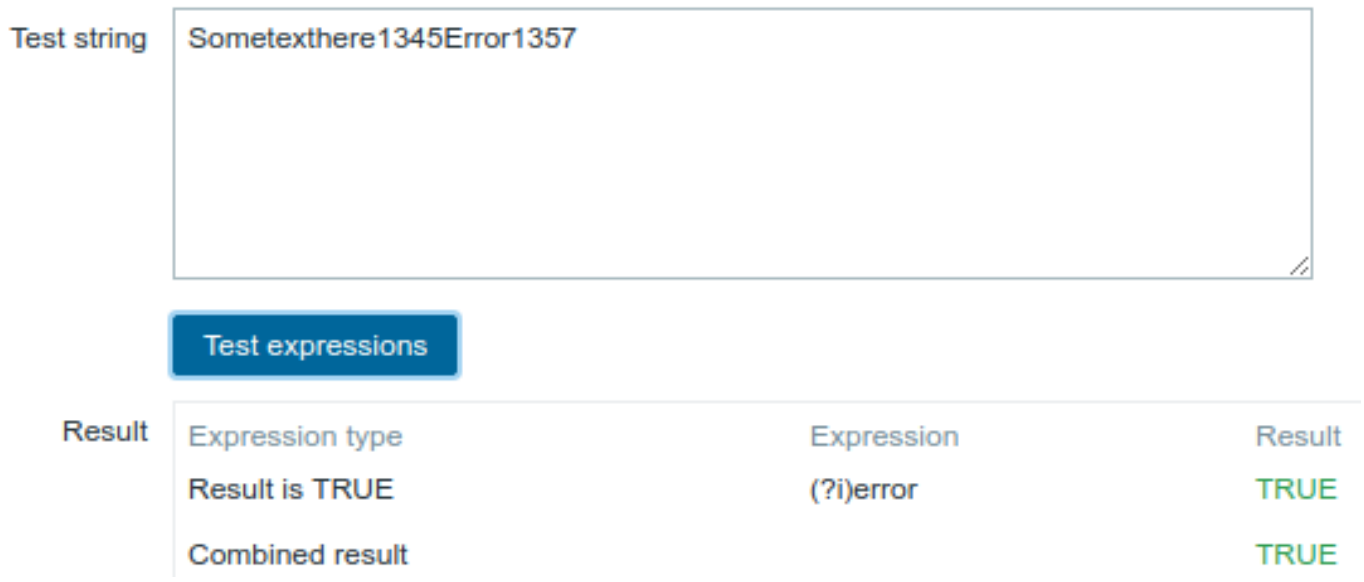
```
^TESTDATABASE$
```



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*”。

内联正则表达式修饰符的示例　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
```



选择 表达式类型（*Expression type*）：” 结果为真（Result is TRUE）”。“error” 字符被匹配到。

内联正则表达式修饰符的另一个示例　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：

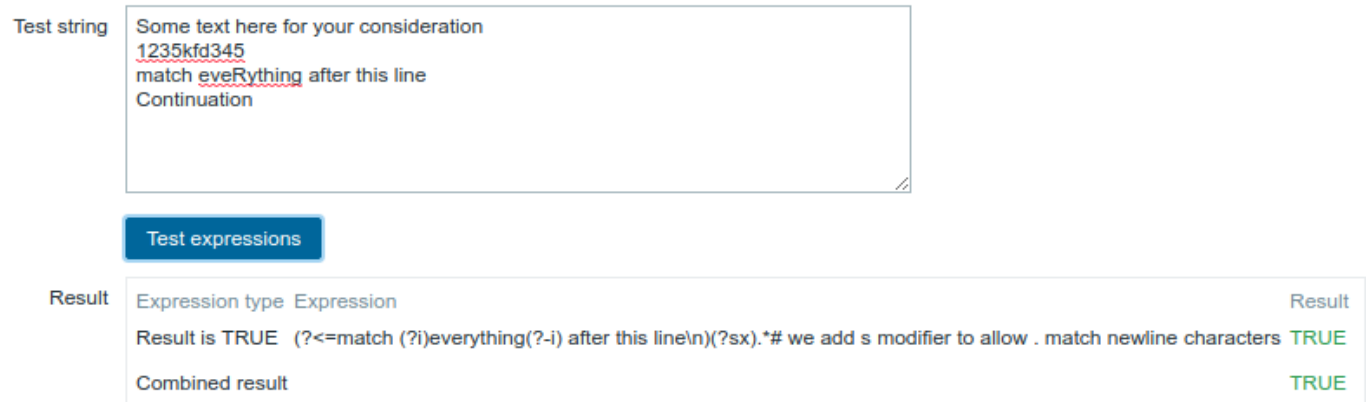
```
(?<=match (?i)everything(?-i) after this line\n)(?sx).*    #我们增加了一个修饰符(?s)来使点号（.）具备匹配
```

(?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。

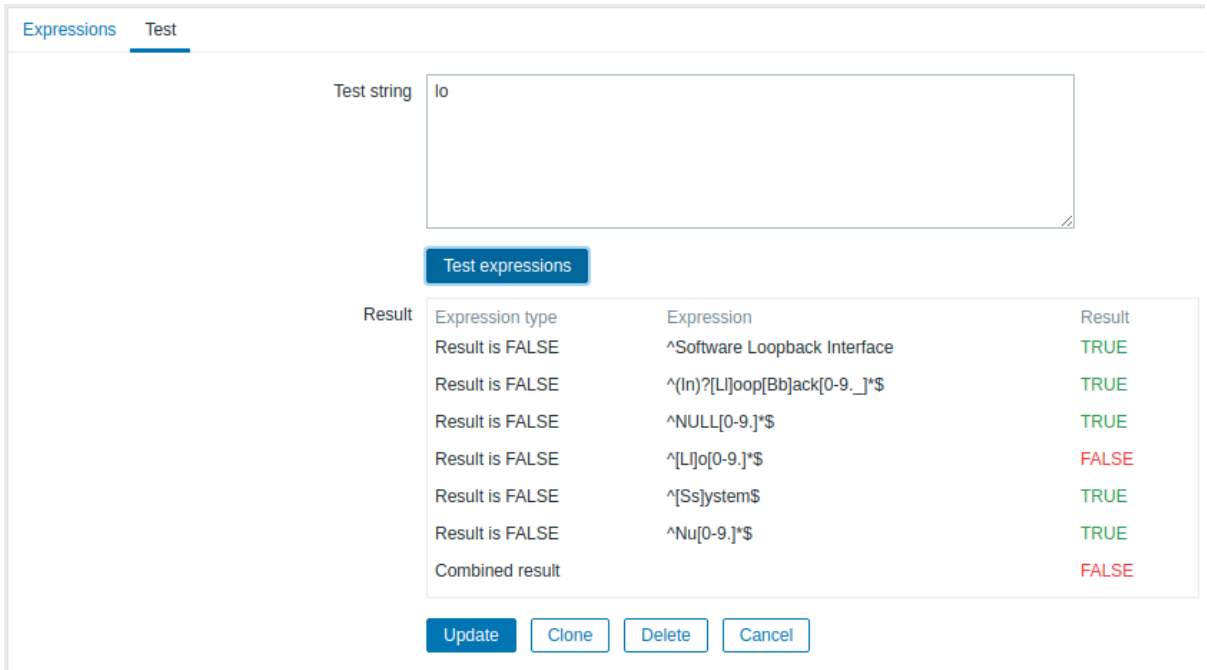
Note:
所以，上面这个例子告诉我们，匹配 match everything after this line\n 后面的字符串，且只有 everything 不区分大小写，而且开启了 (?sx) 模式。



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 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。

更复杂的例子　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。



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。

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。

全局正则表达式的说明

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)$`	” 或 “ext2” 或 “btrf 匹配 “btrf 或 “ext3” 或 “ext4” 或 “jfs” 或 “reiser” 或 “xfs” 或 “ffs” 或 “ufs” 或 “jfs” 或 “jfs2” 或 “vxfs” 或 “hfs“或”refs“或”ntfs“或”f
发现网络接口（*Network interfaces for discovery*） `^Softw|re Loopback Interface` 匹配以”Sof	ware Loop-back Inter-face” 开头的字符串 `^lo$` 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 delayed”。

支持正则表达式的位置

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
	log.count[]			

位置	正	表达式全局正则表达	注释	
	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）](/zh/manual/appendix/trig|<|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）](/zh/manual/discovery/low_level_ 字段 | |**[Web 监测（Web monitoring）](/zh/manual/web_monitoring#co 带有 **regex:** 前缀 *Required string* 字段 | |**[宏函数（Macro functions）](/zh//manual/config/macros/macr |<|reg-sub()|Yes|No|pattern‘ 参数
	iregsub()		
图标映射（**Icon mapping**）	Yes	Yes	*Expr ssion* 字段

Regular expression support by location

Location	Regular expression	Global regular expression	Comments
Agent items			
	eventlog[]	Yes	regexp, severity, source, eventid parameters
	log[]		regexp parameter
	log.count[]		

Location	Regular expression	Global regular expression	Comments
	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
	system.sw.packages[]		`package` parameter
	vfs.dir.count[]		`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
	vfs.dir.size[]		`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
	vfs.file.regexp[]		`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
	logsource()		
	iregexp()		
	regexp()		
Low-level discovery			
	Filters	Yes	*Regular expression* field

Location	Regular expression	Global regular expression	Comments
	Overrides	No	In *matches*, *does not match* options for *Operation* conditions
Action conditions	Yes	No	In *matches*, *does not match* options for *Host name* and *Host metadata* autoregistration conditions
Web monitoring	Yes	No	*Variables* with a **regex:** prefix *Required string* field
User macro context	Yes	No	In macro context with a **regex:** prefix Supported since Zabbix 5.0.2.
Macro functions			
	regsub(), iregsub()	No	`pattern` parameter
Icon mapping	Yes	Yes	*Expression* field

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：

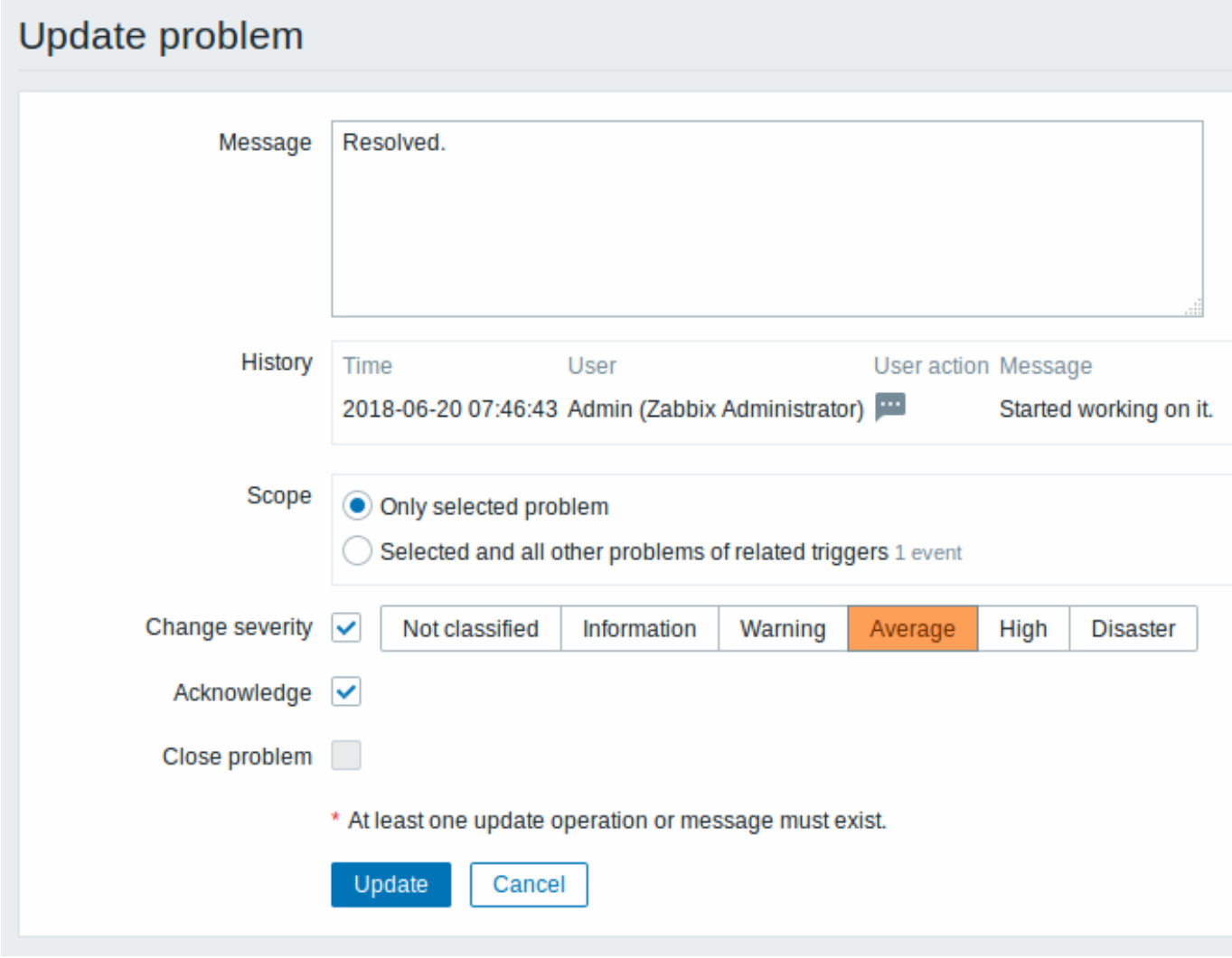
- 监测中（*Monitoring*）→ 仪表板（*Dashboard*） (数据概览（*Data overview*）和 触发器概览（*Trigger overview*）小部件)

- 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。

参数描
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 detail）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：

- 分享模板或者网络拓扑图 - 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
- 聚合图形
- 媒介类型
- 图像
- 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>5.0</version>
    <date>2020-04-22T06:20:11Z</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>5.0</version>

导出的版本。

<date>2020-04-22T06:20:11Z</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。

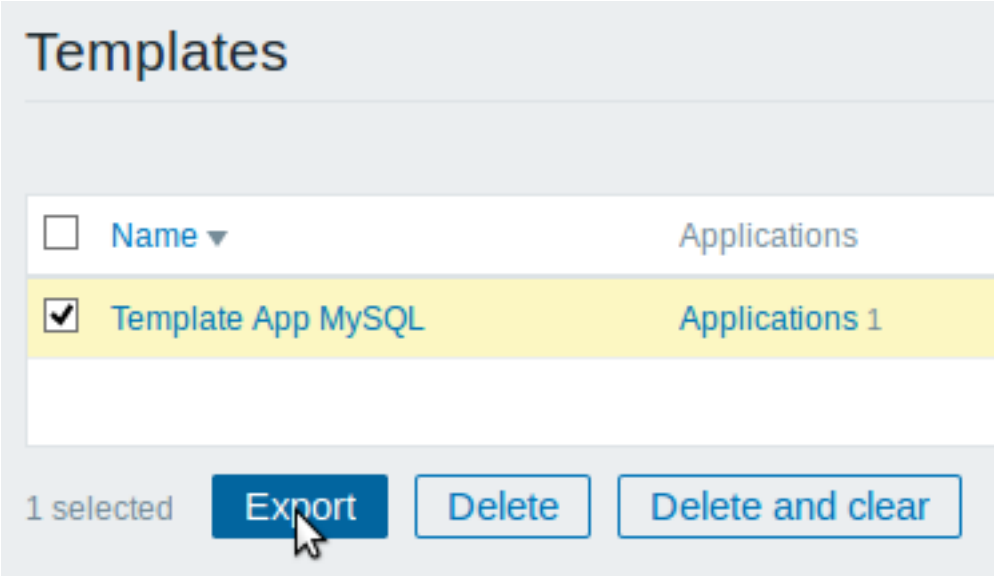
模板导出包含的内容：

- 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

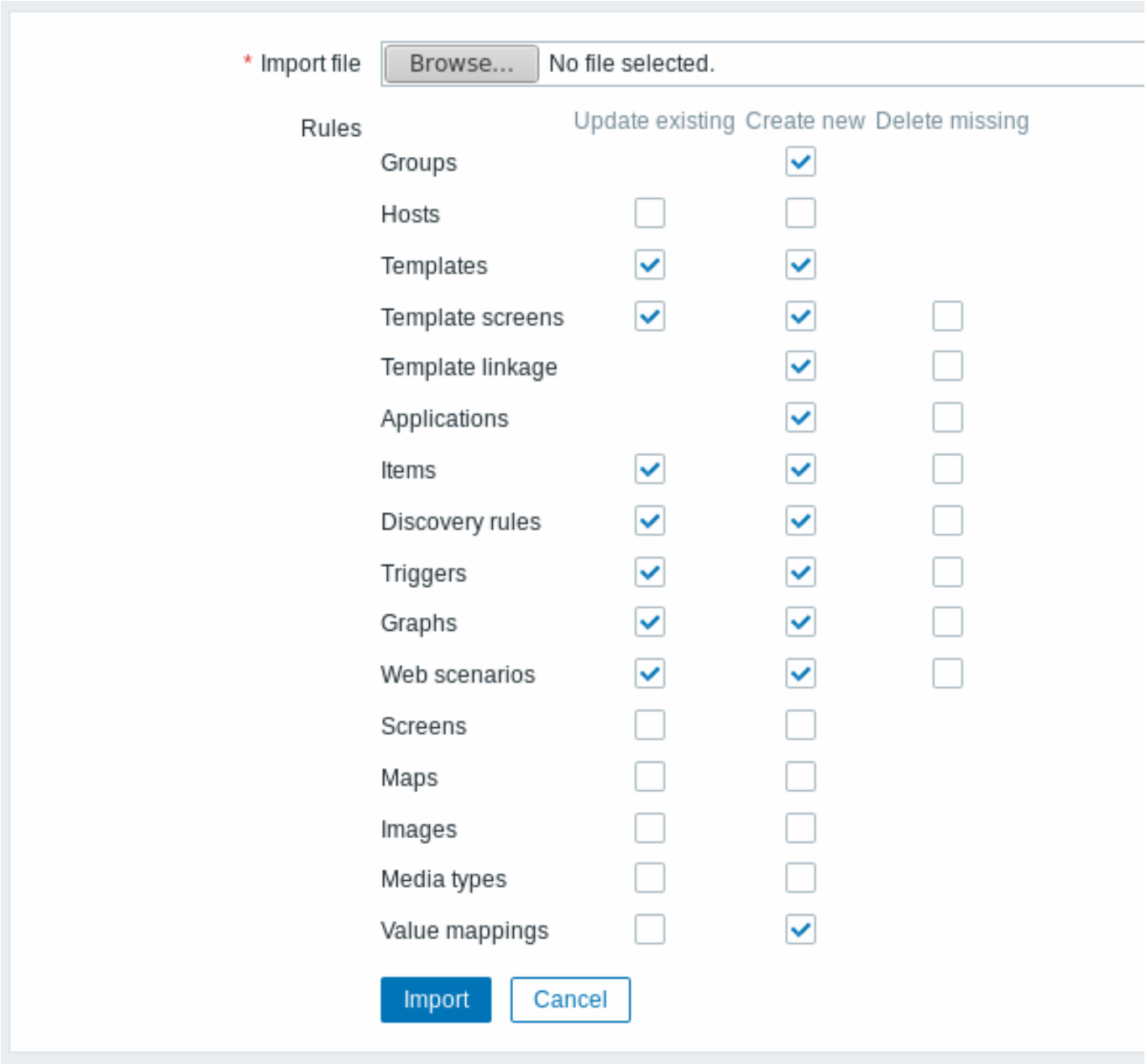


选中的模板被导出到本地的 XML 文件里，默认的名称是 *zabbix_export_templates.xml*。

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>5.0</version>
    <date>2020-04-22T10:03:11Z</date>
    <groups>
        <group>
            <name>Templates/Modules</name>
        </group>
    </groups>
    <templates>
```

```
        <template>
            <template>Template Module Linux filesystems by Zabbix agent</template>
            <name>Template Module Linux filesystems by Zabbix agent</name>
            <description>Template tooling version used: 0.30</description>
            <groups>
                <group>
                    <name>Templates/Modules</name>
                </group>
            </groups>
            <applications>
                <application>
                    <name>Filesystems</name>
                </application>
            </applications>
            <discovery_rules>
                <discovery_rule>
                    <name>Mounted filesystem discovery</name>
                    <key>vfs.fs.discovery</key>
                    <delay>1h</delay>
                    <filter>
                        <evaltype>AND</evaltype>
                        <conditions>
                            <condition>
                                <macro>{#FSTYPE}</macro>
                                <value>{$VFS.FS.FSTYPE.MATCHES}</value>
                                <formulaid>C</formulaid>
                            </condition>
                            <condition>
                                <macro>{#FSTYPE}</macro>
                                <value>{$VFS.FS.FSTYPE.NOT_MATCHES}</value>
                                <operator>NOT_MATCHES_REGEX</operator>
                                <formulaid>D</formulaid>
                            </condition>
                            <condition>
                                <macro>{#FSNAME}</macro>
                                <value>{$VFS.FS.FSNAME.MATCHES}</value>
                                <formulaid>A</formulaid>
                            </condition>
                            <condition>
                                <macro>{#FSNAME}</macro>
                                <value>{$VFS.FS.FSNAME.NOT_MATCHES}</value>
                                <operator>NOT_MATCHES_REGEX</operator>
                                <formulaid>B</formulaid>
                            </condition>
                        </conditions>
                    </filter>
                    <description>Discovery of file systems of different types.</description>
                    <item_prototypes>
                        <item_prototype>
                            <name>{#FSNAME}: Free inodes in %</name>
                            <key>vfs.fs.inode[{#FSNAME},pfree]</key>
                            <history>7d</history>
                            <value_type>FLOAT</value_type>
                            <units>%</units>
                            <application_prototypes>
                                <application_prototype>
                                    <name>Filesystem {#FSNAME}</name>
                                </application_prototype>
                            </application_prototypes>
                            <trigger_prototypes>
                                <trigger_prototype>
                                    <expression>{min(5m)}<{$VFS.FS.INODE.PFREE.MIN.CRIT:"{#FSNAME}"}</expr
```

```
                                <name>{#FSNAME}: Running out of free inodes (free < {$VFS.FS.INODE.PFR
                                <priority>AVERAGE</priority>
                                <description>Last value: {ITEM.LASTVALUE1}.
It may become impossible to write to disk if there are no index nodes left.
As symptoms, 'No space left on device' or 'Disk is full' errors may be seen even though free space is avai
                            </trigger_prototype>
                            <trigger_prototype>
                                <expression>{min(5m)}<{$VFS.FS.INODE.PFREE.MIN.WARN:"{#FSNAME}"}</expr
                                <name>{#FSNAME}: Running out of free inodes (free < {$VFS.FS.INODE.PFR
                                <priority>WARNING</priority>
                                <description>Last value: {ITEM.LASTVALUE1}.
It may become impossible to write to disk if there are no index nodes left.
As symptoms, 'No space left on device' or 'Disk is full' errors may be seen even though free space is avai
                                <dependencies>
                                    <dependency>
                                        <name>{#FSNAME}: Running out of free inodes (free < {$VFS.FS.I
                                        <expression>{Template Module Linux filesystems by Zabbix agent
                                    </dependency>
                                </dependencies>
                            </trigger_prototype>
                        </trigger_prototypes>
                    </item_prototype>
                    <item_prototype>
                        <name>{#FSNAME}: Space utilization</name>
                        <key>vfs.fs.size[{#FSNAME},pused]</key>
                        <history>7d</history>
                        <value_type>FLOAT</value_type>
                        <units>%</units>
                        <description>Space utilization in % for {#FSNAME}</description>
                        <application_prototypes>
                            <application_prototype>
                                <name>Filesystem {#FSNAME}</name>
                            </application_prototype>
                        </application_prototypes>
                    </item_prototype>
                    <item_prototype>
                        <name>{#FSNAME}: Total space</name>
                        <key>vfs.fs.size[{#FSNAME},total]</key>
                        <history>7d</history>
                        <units>B</units>
                        <description>Total space in Bytes</description>
                        <application_prototypes>
                            <application_prototype>
                                <name>Filesystem {#FSNAME}</name>
                            </application_prototype>
                        </application_prototypes>
                    </item_prototype>
                    <item_prototype>
                        <name>{#FSNAME}: Used space</name>
                        <key>vfs.fs.size[{#FSNAME},used]</key>
                        <history>7d</history>
                        <units>B</units>
                        <description>Used storage in Bytes</description>
                        <application_prototypes>
                            <application_prototype>
                                <name>Filesystem {#FSNAME}</name>
                            </application_prototype>
                        </application_prototypes>
                    </item_prototype>
                </item_prototypes>
                <trigger_prototypes>
                    <trigger_prototype>
```

```
                            <expression>{Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#F
(({Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#FSNAME},total].last()}-{Template Module
                            <name>{#FSNAME}: Disk space is critically low (used > {$VFS.FS.PUSED.MAX.CRIT:
                            <priority>AVERAGE</priority>
                            <description>Last value: {ITEM.LASTVALUE1}.
Space used: {ITEM.VALUE3} of {ITEM.VALUE2} ({ITEM.VALUE1}), time left till full: < 24h.
Two conditions should match: First, space utilization should be above {$VFS.FS.PUSED.MAX.CRIT:"{#FSNAME}"}
 Second condition should be one of the following:
 - The disk free space is less than 5G.
 - The disk will be full in less than 24hours.</description>
                            <manual_close>YES</manual_close>
                        </trigger_prototype>
                        <trigger_prototype>
                            <expression>{Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#F
(({Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#FSNAME},total].last()}-{Template Module
                            <name>{#FSNAME}: Disk space is low (used > {$VFS.FS.PUSED.MAX.WARN:"{#FSNAME}"
                            <priority>WARNING</priority>
                            <description>Last value: {ITEM.LASTVALUE1}.
Space used: {ITEM.VALUE3} of {ITEM.VALUE2} ({ITEM.VALUE1}), time left till full: < 24h.
Two conditions should match: First, space utilization should be above {$VFS.FS.PUSED.MAX.CRIT:"{#FSNAME}"}
 Second condition should be one of the following:
 - The disk free space is less than 10G.
 - The disk will be full in less than 24hours.</description>
                            <manual_close>YES</manual_close>
                            <dependencies>
                                <dependency>
                                    <name>{#FSNAME}: Disk space is critically low (used > {$VFS.FS.PUSED.M
                                    <expression>{Template Module Linux filesystems by Zabbix agent:vfs.fs.
(({Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#FSNAME},total].last()}-{Template Module
                                </dependency>
                            </dependencies>
                        </trigger_prototype>
                    </trigger_prototypes>
                    <graph_prototypes>
                        <graph_prototype>
                            <name>{#FSNAME}: Disk space usage</name>
                            <width>600</width>
                            <height>340</height>
                            <type>PIE</type>
                            <show_3d>YES</show_3d>
                            <graph_items>
                                <graph_item>
                                    <color>969696</color>
                                    <calc_fnc>LAST</calc_fnc>
                                    <type>GRAPH_SUM</type>
                                    <item>
                                        <host>Template Module Linux filesystems by Zabbix agent</host>
                                        <key>vfs.fs.size[{#FSNAME},total]</key>
                                    </item>
                                </graph_item>
                                <graph_item>
                                    <sortorder>1</sortorder>
                                    <color>C80000</color>
                                    <calc_fnc>LAST</calc_fnc>
                                    <item>
                                        <host>Template Module Linux filesystems by Zabbix agent</host>
                                        <key>vfs.fs.size[{#FSNAME},used]</key>
                                    </item>
                                </graph_item>
                            </graph_items>
                        </graph_prototype>
                    </graph_prototypes>
```

```
                </discovery_rule>
            </discovery_rules>
            <macros>
                <macro>
                    <macro>{$VFS.FS.FSNAME.MATCHES}</macro>
                    <value>.+</value>
                </macro>
                <macro>
                    <macro>{$VFS.FS.FSNAME.NOT_MATCHES}</macro>
                    <value>^(/dev|/sys|/run|/proc|.+/shm$)</value>
                </macro>
                <macro>
                    <macro>{$VFS.FS.FSTYPE.MATCHES}</macro>
                    <value>^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat3
                </macro>
                <macro>
                    <macro>{$VFS.FS.FSTYPE.NOT_MATCHES}</macro>
                    <value>^\s$</value>
                </macro>
                <macro>
                    <macro>{$VFS.FS.INODE.PFREE.MIN.CRIT}</macro>
                    <value>10</value>
                </macro>
                <macro>
                    <macro>{$VFS.FS.INODE.PFREE.MIN.WARN}</macro>
                    <value>20</value>
                </macro>
                <macro>
                    <macro>{$VFS.FS.PUSED.MAX.CRIT}</macro>
                    <value>90</value>
                </macro>
                <macro>
                    <macro>{$VFS.FS.PUSED.MAX.WARN}</macro>
                    <value>80</value>
                </macro>
            </macros>
        </template>
    </templates>
</zabbix_export>
```

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	属性必需	型	围	说明	
templates		-			模板的根元素。
template		-			单独的模板。
	template	x	字符	唯	模板名称。
	name	-	字符	显	模板名称。
	description	-	文本	模	描述。
groups		x			主机组根元素。
group		x			单独的主机组。
	name	x	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	x	字符	应	集名称。
macros		-			模板用户宏的根元素。
macro		-			单独的模板用户宏。
	macro	x	字符	用	宏名称。
	type	-	字符 0	- TEXT (default) 用 1 - SECRET_TEXT	宏的类型。
	value	-	字符	用	宏的值。

元素元	属性必需	型	围	说明	
	description	-	字符	用	宏描述。
tags		-			模板用户标签的根元素。
tag		-			单独的模板用户标签。
	tag	x	字符	标	名称。
	value	-	字符	标	的值。
templates		-			链接模板的根元素。
template		-			单独的模板。
	name	x	字符	模	名称。

模板监控项标签

元素元	属性必需	型	围	说明	
items		-			监控项的根元素。
item		-			单独的监控项。
	name	x	字符	监	项名称。

元素元	属性必需	型	围	说明
	type	-	字符0	- 项类型。监Zabbix agent 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 monitoring scenario step) 10 - external 11 - database monitor 12 - IPMI agent 13 - SSH agent 14 -

元素元	属性必需	型	围	说明	
	snmp_oid	-	字符	S	MP 对象 ID。SNMP 监控项必需。
	key	x	字符	监	项的 key。

元素元	属性必需	型	围	说明	
	delay	-	字符缺	: 1m 监控项	更新间隔。 秒或以 (30s, 1m, 2h, 1d) 为基准的时间单位。 补充说明，可以将一个或多个自定义间隔设定为灵活间隔或调度计划。多个间隔用分号分隔。可以使用用户宏

元素元	属性必需	型	围	说明	
	history	-	字符缺	:90d 历史数	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。
	trends	-	字符缺	:365d 趋势数	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	属性必需	型	围	说明
	status	-	字符0	- 态。EN-ABLED(缺省) 监控项 1 - DIS-ABLED
	value_type	-	字符0	- 值的FLOAT 数1 - 类型。CHAR 2 - LOG 3 - UN-SIGNED (缺省) 4 - TEXT
	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。

元素元	属性必需	型	围	说明	
	units	-	字符	返	值的单位(bps, B)。
	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。

元素元	属性必需	型	围	说明	
	ipmi_sensor	-	字符	如	‘type’是12，这是IPMI传感器ID。
	authtype	-	字符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。

元素元	属性必需	型	围	说明	
	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。
	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。

元素元	属性必需	型	围	说明
	description	-	文本	监项描述。
	inventory_link	-	文本0	-None 由大写的主机清单字段名。支持的主机资产字段和例如：4 - ALIAS 6 - OS_FULL 14 - HARD-WARE D请参照主机资产。
	logtimefmt	-	字符	日条目的时间格式。只有日志监控项使用。
	jmx_endpoint	-	字符	如‘type’是16，这是JMX端点。

元素元	属性必需	型	围	说明	
	url	-	字符	如	’type’是19，这是URL字符。
	allow_traps	-	字符0	- 不允许trap-ping。如果’ty1 - 允许trap-ping。	e’是19，属性允许发送数据给监控项。
	follow_redirects	-	字符0	- 不跟随重定向。如果’type’1 - 跟随重定向。	19，跟随HTTP重定向。

元素元	属性必需	型	围	说明
headers		-		如果’type’是19，这是带有HTTP(S)请求头的根元素，请求头名字作为key，请求头的值作为value。
header		-		单独的请求头。
	name	x	字符请	头的名字。
	value	x	字符请	头的值。
	http_proxy	-	字符	如’type’是19，这是HTTP(S)代理连接字符。

元素元	属性必需	型	围	说明
	output_format	-	字符0	- 保持原样存储。如果’type’19，怎样处理响应。1 - 转换为JSON。
	post_type	-	字符0	- 原始数据。如果’type’19，这是请求体的类型。2 - JSON数据。3 - XML数据。
	posts	-	字符	如’type’是19，这是请求体。
query_fields		-		如果’type’是19，请求查询参数的根元素。

元素元	属性必需	型	围	说明	
query_field		-		单独的请求查询参数的根元素。	
	name	x	字符参	的名字。	
	value	-	字符参	的值。	
	request_method	-	字符0	- GET 1 - POST 2 - PUT 3 - HEAD	’type’ 如是19，这是请求方法。
	retrieve_mode	-	字符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。

元素元	属性必需	型	围	说明	
	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。
	ssl_key_password	-	字符	如	‘type’是19，这是SSL密钥文件的密码。

元素元	属性必需	型	围	说明	
	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	属性必需	型	围	说明	
	verify_host	-	字符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	-	字符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	属性必需	型	围	说明	
	name	x	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	属性必需	型	围	说明
	type	x	字符1	- 项值预处理步骤的类型 MULTIPLIER 2 - RTRIM 3 - LTRIM 4 - TRIM 5 - REGEX 6 - BOOL_TO_DECIMAL 7 - OCTAL_TO_DECIMAL 8 - HEX_TO_DECIMAL 9 - SIMPLE_CHANGE (calculated as (received value-previous value)) 10 - CHANGE_PER_SECON (calculated as (received value-previous value)/(time now-time of last check)) 11 - XMLPATH 12 - JSONPATH 13 - IN_RANGE 14 - MATCHES_REGEX 15 - NOT_MATCHES_REGE 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 18 -

元素元	属性必需	型	围	说明
	params	x	字符	监项值预处理步骤的参数。
	error_handler	-	字符0	-ORIG-I-NAL_ERROR(de-fault) 预理失败时的动作类型。1 - DIS-CARD_VALUE2 - CUS-TOM_VALUE3 - CUS-TOM_ERROR
	error_handler_params	-	字符	错处理的参数。
master_item				单个监控项主监控项数据。
	key	x	字符	从监控项的主监控项值。

元素元	属性必需	型	围	说明
triggers		-		简单触发器的根元素。
trigger		-		单独的触发器。
	更多简单触发器的标签值, 请参照模板触发器标签.			

模板低级别发现规则标签

元素元	属性必需	型	围	说明
discovery_rules		-		低级别发现规则的根元素。
discovery_rule		-		单独的低级别发现规则。
	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。			

元素元	属性必需	型	围	说明
	type	-	字符0	- 项类型监ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 16 - JMX 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT

元素元	属性必需	型	围	说明	
	lifetime	-	字符	时	周期，监控项超过此时间不再被发现的话将被删除。秒，带有后缀的时间单位或者用户宏。
filter				单独的过滤条件。	

元素元	属性必需	型	围	说明
	evaltype	-	字符0	- 与/或逻辑检查低级别1 - 与逻辑2 - 或逻辑3 - 自定义公式过现规则过滤条件的逻辑。
	formula	-	字符	条件的自定义计算公式。
conditions		-		过滤条件的根元素。
condition		-		单独的过滤器条件。
	macro	x	字符	低别发现宏变量名称。

元素元	属性必需	型	围	说明
	value	-	字符	过器值：正则表达式或者全局正则表达式。
	operator	-	字符8	- MATCHES_REGEX (de-fault) 9 - NOT_MATCHES_REGE运算符条

元素元	属性必需	型	围	说明	
	formulaid	x	字符	用	从自定义表达式中索引条件的任意唯一ID。只能包含大写字母。修改过滤条件时必须由用户定义ID，但在以后请求时将重新生成ID。

元素元	属性必需	型	围	说明	
lld_macro_paths		-			低级别发现宏路径的根元素。
lld_macro_path		-			单独的低级别发现宏路径。
	lld_macro	x	字符	低	别发现宏的名字。
	path	x	字符	给	应宏赋值的选择器。
preprocessing		-			低级别发现规则处理过程。

元素元	属性必需	型	围	说明
step		-		单独的低级别发现规则处理步骤。
	更多元素标签值，请参考监控项值处理模板中的元素标签值。如下仅描述指定给低级别发现值处理模板的标签值。			
	type	x	字符5	- 项监理步骤类型。 REGEX 11 - XML-PATH 12 - JSON-PATH 15 - NOT_MATCHES_REGE 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 20 - DIS-CARD_UNCHANGED_ 21 - JAVASCRIPT 23 - PROMETHEUS_TO_JS 24 - CSV_TO_JSON 25 - STR_REPLACE
trigger_prototypes		-		触发器原型根元素。
trigger_prototype		-		单独的触发器原型。
	对于大部分元素标签值来说，请查阅触发器模板 标签.			

元素元	属性必需	型	围	说明
graph_prototypes		-		图形原型的根元素。
graph_prototype		-		单独的图形原型。
	对于大部分元素标签值来说，请查阅图形模板 标签。			
host_prototypes		-		主机原型的根元素。
host_prototype		-		单独的主机原型。
	对于大部分元素标签值来说，请查阅主机 标签。			
item_prototypes		-		监控项原型的根元素。
item_prototype		-		单独的监控项原型。
	对于大部分元素标签值来说，请查阅监控项模板 标签。			

元素元	属性必需	型	围	说明	
application_prototypes		-			应用集原型的根元素。
application_prototype		-			单独的应用集原型。
	name	x			应用集原型名称。
master_item_prototype		-			单独的监控项原型主监控项原型数据。
	key	x	字符	单	的监控项原型主监控项原型键值。

模板触发器标签

元素元	属性必需	型	围	说明	
triggers		-			触发器的根元素。
trigger		-			单独的触发器。
	expression	x	字符	触	器表达式。
	recovery_mode	-	字符 0	- 表达式 生成 OK1 - 恢复表达式 2 - none	件的基础。
	recovery_expression	-	字符	触	器恢复表达式。
	name	x	字符	触	器名称。
	correlation_mode	-	字符 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	-	字符 0	- enabled 触 1 - disabled	器状态。

元素元	属性必需	型	围	说明
	priority	-	字符 0	- 未分类 触发器严 1 - 信息 2 - 警告 3 - 一般严重 4 - 严重 5 - 灾难 性。
	description	-	文本	触 器描述。
	type	-	字符 0	- 单个问题事件 事件生成类型。 1 - 多个问题事件
	manual_close	-	字符 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。
	name	x	字符	依 触发的名称。
	expression	x	字符	依 触发器的表达式。
	recovery_expression	-	字符	依 触发器的恢复表达式。

元素元	属性必需	型	围	说明	
tags		-			事件标签的根元素。
tag		-			单独的事件标签。
	tag	x	字符	标	名称。
	value	-	字符	标	值。

模板图形标签

元素元	属性必需	型	范围	说明	
graphs		-			图形的根元素。
graph		-			单独的图形。
	name	x	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。

元素元	属性必需	型	范围	说明	
	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。
	yaxismax	-	双精度	如果	ymax_type_1’是1，那么这是Y轴的最大值。
	show_work_period	-	字符 0	- no 如 1 - yes	‘type’是0、1，突显非工作日。

元素元	属性必需	型	范围	说明	
	show_triggers	-	字符 0	- no 如 1 - yes	‘type’ 是 0、1，以线条方式显示简单的触发器值。
	type	-	字符 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	-	字符 0	- no 显 1 - yes	图形图例。
	show_3d	-	字符 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。

元素元	属性必需	型	范围	说明	
	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。
ymin_item_1		-			单独的监控项明细。 要求’ymin_type_1是2.
	host	x	字符	监	项主机。
	key	x	字符	监	项键。

元素元	属性必需	型	范围	说明
ymax_item_1		-		单独监控项明细。 要求’ymax_type_1是2。
	host	x	字符	监项主机。
	key	x	字符	监项key。
graph_items		x		图形监控项的根元素。
graph_item		x		单独的图形监控项。

元素元	属性必需	型	范围	说明	
	sortorder	-	字符	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	-	字符 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	属性必需	型	范围	说明	
	yaxisside	-	字符 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	-	字符 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	-	字符 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		x			单独的监控项。
	host	x	字符	监	项的主机。

元素元	属性必需	型	范围	说明
	key	x	字符	监项的键。

模板 web 场景标签

元素元	属性必需	型	围	说明
httptests		-		web场景的根元素。
httptest		-		单独的web场景。
	name	x	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。

元素元	属性必需	型	围	说明	
	attempts	-	整型 1	10 执	Web 场景步骤的尝试次数。
	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。

元素元	属性必需	型	围	说明
	http_proxy	-	字符	指要使用的HTTP代理，使用这个格式：http://[user
variables		-		场景列表-用在场景步骤中场景级别的变量(宏)。
variable		-		具体的变量。
	name	x	文本	变名称。
	value	x	文本	变值。

元素元	属性必需	型	围	说明
headers		-		请求头列表 - 当执行请求的时候，发送的HTTP头部。因请求头可能直接出现在HTTP协议中，请求头列表中的请求头需使用相同的语法规则。

元素元	属性必需	型	围	说明	
header		-			具体的请求头。
	name	x	文本	请	头名称。
	value	x	文本	请	头值。
	status	-	字符 0	- en- abled w1 - disabled	b 场景状态。
	authentication	-	字符 0	- none 认 1 - basic 2 - NTLM	方法。
	http_user	-	字符	认	用户名。
	http_password	-	字符	指	用户名的认证密码。
	verify_peer	-	字符 0	- no 校 1 - yes	web 服务器的 SSL 证书。

元素元	属性必需	型	围	说明
	verify_host	-	字符 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。
	ssl_key_file	-	字符	客 端认证用到的 SSL 私钥文件的名称。

元素元	属性必需	型	围	说明	
	ssl_key_password	-	字符	S	L 私钥文件密码。
steps		x		web 场景步骤的根元素。	
step		x		具体的 web 场景步骤。	
	name	x	字符	w	b 场景步骤名称。
	url	x	字符	要	控的 URL。
query_fields		-			查询字段列表的根元素
					- 当发送 HTTP 请求时会加到 URL 中的 HTTP 字段列表。

元素元	属性必需	型	围	说明	
query_field		-			具体的查询字段。
	name	x	字符	查	字段的名称。
	value	-	字符	查	字段的值。
posts		-			字符串(post原始数据)型或者列表(数据字段表单)型HTTP POST变量。
post_field		-			具体的post字段。
	name	x	文本	P	st字段名称。
	value	x	文本	P	st字段值。

元素元	属性必需	型	围	说明
variables		-		步骤列表-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	属性必需	型	围	说明	
variable		-			具体的变量。
	name	x	文本	变	名称。
	value	x	文本	变	值。

元素元	属性必需	型	围	说明
headers		-		请求头列表 - 当执行请求的时候，发送的HTTP头部。因请求头可能直接出现在HTTP协议中，请求头列表中的请求头需使用相同的语法规则。

元素元	属性必需	型	围	说明	
header		-			具体的请求头。
	name	x	文本	请	头名称。
	value	x	文本	请	头的值。
	follow_redirects	-	字符 0	- no 跟 1 - yes	HTTP 跳转。
	retrieve_mode	-	字符 0	- 内容 HTT1 - 仅 HTTP 头部	响应检索模式。
	timeout	-	字符缺	：15s 执行步	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	属性必需	型	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。

聚合图形模板标签

元素元	属性必需	型范围	**[1](脚注)**^ 描述
screens		-	聚合图形模板集列表。
screen		-	具体的聚合图形模板。
screen_items		-	局和图形监控项模板列表。
screen_item		-	具体的聚合图形监控项模板。

脚注

[1] 对于字符串值，只导出字符串（例如 “ZABBIX_ACTIVE”），而不使用此表中使用的编号。此表中范围值（对应于 API 值）的数字仅用于排序。

3 主机

概述

导出（exported） 的主机具有许多相关对象和对象关系。

主机导出的内容包含：

- 链接的主机组
- 主机数据
- 模板链接
- 主机组链接
- 主机接口
- 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
- 直接链接的具有所有原型的发现规则
- 直接链接的 web 场景
- 主机宏

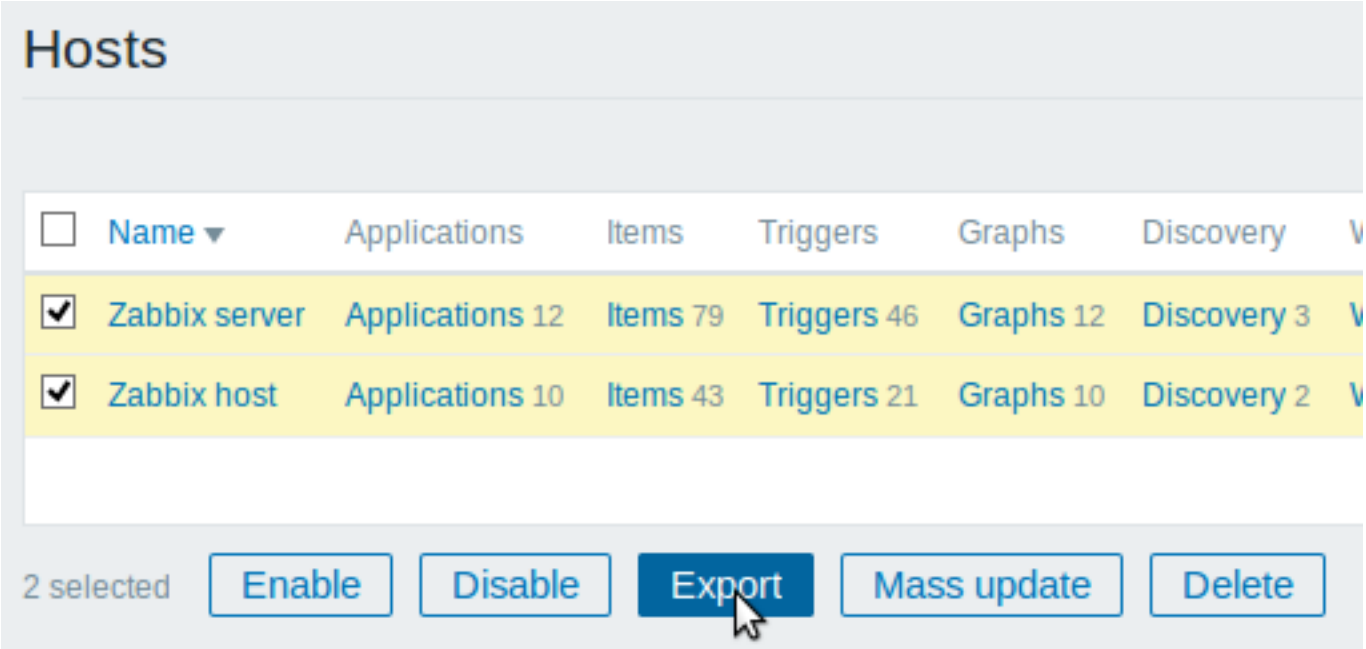
- 主机资产清单数据
- 值映射

导入和更新主机时，它只能链接到附加的模板，并且不会取消链接。

导出

要导出主机，按照如下操作：

- 切换到：配置（*Configuration*） → 主机（*Hosts*）
- 选中要导出主机的复选框
- 单击列表下方的导出（*Export*）按钮

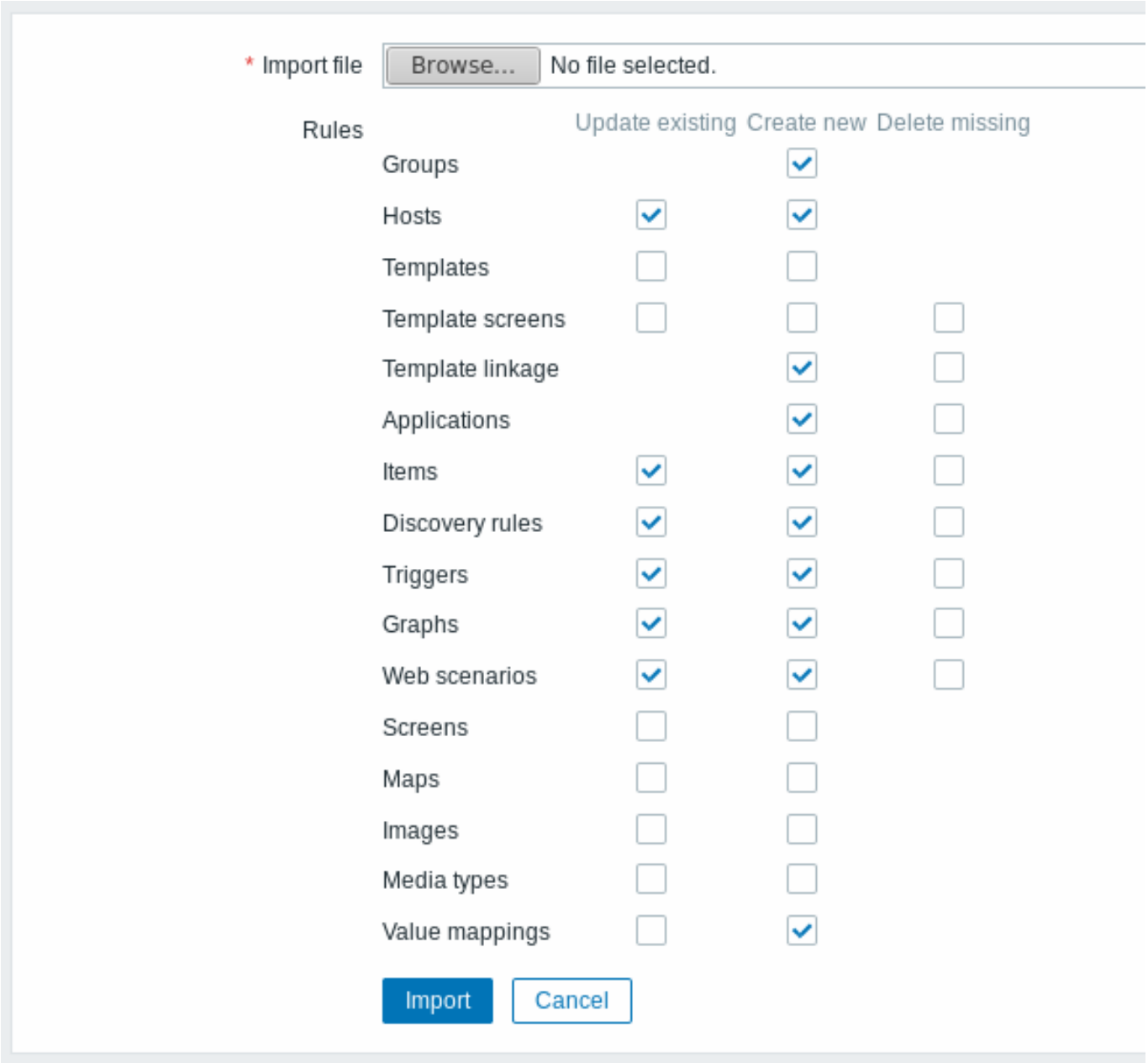


选中的主机会以默认名称 *zabbix_export_hosts.xml* 导出到本地的 XML 文件里。

导入

导入主机，按照如下操作：

- 切换到：配置（*Configuration*） → 主机（*Hosts*）
- 单击右侧的导入（*Import*）按钮
- 选择导入文件
- 标记导入规则里的必选项
- 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*）现有元素将使	从导入文件中获取的数据进行更新。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元素。否则它不会添加它们。
删除不存在（*Delete missing*）导入将删除导	文件中不存在的现有元素。否则它不会删除它们。

导出格式

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2020-04-22T07:47:33Z</date>
    <groups>
        <group>
            <name>Discovered hosts</name>
        </group>
        <group>
            <name>Zabbix servers</name>
```

```
        </group>
    </groups>
    <hosts>
        <host>
            <host>Zabbix server 1</host>
            <name>Main Zabbix server</name>
            <proxy>
                <name>Remote proxy</name>
            </proxy>
            <tls_connect>TLS_PSK</tls_connect>
            <tls_accept>
                <option>NO_ENCRYPTION</option>
                <option>TLS_PSK</option>
            </tls_accept>
            <tls_psk_identity>z112</tls_psk_identity>
            <tls_psk>1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952</tls_psk>
            <templates>
                <template>
                    <name>Template App Zabbix Server</name>
                </template>
                <template>
                    <name>Template OS Linux</name>
                </template>
            </templates>
            <groups>
                <group>
                    <name>Discovered hosts</name>
                </group>
                <group>
                    <name>Zabbix servers</name>
                </group>
            </groups>
            <interfaces>
                <interface>
                    <ip>192.168.1.1</ip>
                    <interface_ref>if1</interface_ref>
                </interface>
            </interfaces>
            <items>
                <item>
                    <name>Zabbix trap</name>
                    <type>TRAP</type>
                    <key>trap</key>
                    <delay>0</delay>
                    <history>1w</history>
                    <applications>
                        <application>
                            <name>Zabbix server</name>
                        </application>
                    </applications>
                    <preprocessing>
                        <step>
                            <type>MULTIPLIER</type>
                            <params>8</params>
                        </step>
                    </preprocessing>
                    <triggers>
                        <trigger>
                            <expression>{last()}=0</expression>
                            <name>Last value is zero</name>
                            <priority>WARNING</priority>
                            <tags>
```

```
                            <tag>
                                <tag>Process</tag>
                                <value>Internal test</value>
                            </tag>
                        </tags>
                    </trigger>
                </triggers>
            </item>
        </items>
        <tags>
            <tag>
                <tag>Process</tag>
                <value>Zabbix</value>
            </tag>
        </tags>
        <macros>
            <macro>
                <macro>{$HOST.MACRO}</macro>
                <value>123</value>
            </macro>
            <macro>
                <macro>{$PASSWORD1}</macro>
                <type>SECRET_TEXT</type>
            </macro>
        </macros>
        <inventory>
            <type>Zabbix server</type>
            <name>yyyyyy-HP-Pro-3010-Small-Form-Factor-PC</name>
            <os>Linux yyyyyy-HP-Pro-3010-Small-Form-Factor-PC 4.4.0-165-generic #193-Ubuntu SMP Tue Se
        </inventory>
        <inventory_mode>AUTOMATIC</inventory_mode>
    </host>
</hosts>
<graphs>
    <graph>
        <name>CPU utilization server</name>
        <show_work_period>NO</show_work_period>
        <show_triggers>NO</show_triggers>
        <graph_items>
            <graph_item>
                <drawtype>FILLED_REGION</drawtype>
                <color>FF5555</color>
                <item>
                    <host>Zabbix server 1</host>
                    <key>system.cpu.util[,steal]</key>
                </item>
            </graph_item>
            <graph_item>
                <sortorder>1</sortorder>
                <drawtype>FILLED_REGION</drawtype>
                <color>55FF55</color>
                <item>
                    <host>Zabbix server 1</host>
                    <key>system.cpu.util[,softirq]</key>
                </item>
            </graph_item>
            <graph_item>
                <sortorder>2</sortorder>
                <drawtype>FILLED_REGION</drawtype>
                <color>009999</color>
                <item>
                    <host>Zabbix server 1</host>
                    <key>system.cpu.util[,interrupt]</key>
```

```
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>3</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>990099</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,nice]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>4</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>999900</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,iowait]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>5</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>990000</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,system]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>6</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>000099</color>
                    <calc_fnc>MIN</calc_fnc>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,user]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>7</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>009900</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,idle]</key>
                    </item>
                </graph_item>
            </graph_items>
        </graph>
    </graphs>
</zabbix_export>
```

元素标签

元素标签值在下表中说明。

主机标签

元素元	属性必需	型	围	说明	
groups		x			主机组的根元素。
group		x			单独的主机组。
	name	x	字符	唯	组名。
hosts		-			主机根元素。
host		-			单独的主机。
	host	x	字符	唯	主机名。
	name	-	字符	可	主机名。
	description	-	文本	主	说明。
	status	-	字符 0	- 监控 主机状 1 - 不监控	。
	ipmi_authtype	-	字符 -	- 默认 IPM0 - none 1 - MD2 2 - MD5 4 - straight 5 - OEM 6 - RMCP+	会话认证类型。

元素元	属性必需	型	围	说明	
	ipmi_privilege	-	字符1	- call-back I2 - user 3 - oper-ator 4 - ad-min 5 - OEM	MI会话权限级别。
	ipmi_username	-	字符	I	MI检查的用户名。
	ipmi_password	-	字符	I	MI检查的密码。
	tls_connect	-	字符1	- 不加密 出口连接 2 - TLS with PSK 4 - TLS with cer-tifi-cate	类型。
tls_accept		-			入口连接类型的根元素。

元素元	属性必需	型	围	说明
	option	-	字符 1	- NO_ENCRYPTION (缺省) 入口连 2 - TLS_PSK 4 - TLS_CERTIFICATE 的类型。 如果同时允许未加密和加密的连接，则 <option> 属性将被使用两次，一次使用不加密，另一次使用加密选项 (参见上面的示例)。
	tls_issuer	-	字符	允 的 agent/proxy 证书颁发者。

元素元	属性必需	型	围	说明
	tls_subject	-	字符	允许的agent/proxy证书主题。
	tls_psk_indentity	-	字符	PSK身份字符串。
	tls_psk	-	字符	PSK值字符串。
proxy		-		代理。
	name	x	字符	监主机的proxy节点(如果有的话)名称。
templates		-		链接模板的根元素。
template		-		单独的模板。
	name	x	字符	模名称。

元素元	属性必需	型	围	说明	
interfaces		-			主机接口的根元素。
interface		-			单独的接口。
	default	-	字符0	- 备用接口状1 - 主用(默认的)主机上只	有一种类型的主用接口。
	type	-	字符0	- 未知接口类1 - Zabbix agent2 - SNMP3 - IPMI4 - JMX	。
	useip	-	字符0	- 使用DNS名称连接主机的1 - 使用IP地址	口。
	ip	-	字符	I	地址，IPv4或者IPv6都可以。

元素元	属性必需	型	围	说明	
	dns	-	字符	D	S 名称。
	port	-	字符	P	rt 号。
	bulk	-	字符 0	- dis- able 1 - en- able	使 SNMP 的批量请求。
	interface_ref	x	字符	要	监控项中使用的接口引用名称。
details		-			接口细节信息的根元素。
	community	-	字符	S	MP com- mu- nity. SNMPv1 and SN- MPv2 监控项需要用。
	bulk	-	字符 0	- NO 为 1 - YES (缺省)	SNMP 使用块请求。

元素元	属性必需	型	围	说明	
	snmpv3_contextname	-	字符	S	MPv3 con-text 名称。 仅 SN-MPv3 监控项使用。
	snmpv3_securityname	-	字符	S	MPv3 se-cu-rity 名称。 仅 SN-MPv3 监控项使用。
	snmpv3_securitylevel	-	字符 0	- NOAU-THNO-PRIV (缺省) SNM1 - AU-THNO-PRIV 2 - AU-TH-PRIV	v3 安全等级。 仅 SN-MPv3 监控项使用。
	snmpv3_authprotocol	-	字符 0	- MD5 (缺省) SNM1 - SHA	v3 认证协议。 仅 SN-MPv3 监控项使用。

元素元	属性必需	型	围	说明
	snmpv3_authpassphrase	-	字符	S MPv3 认证密码。 仅 SN-MPv3 监控项使用。
	snmpv3_privprotocol	-	字符 0	- DES (缺省) SNM - AES v3 私有协议。 仅 SN-MPv3 监控项使用。
	snmpv3_privpassphrase	-	字符	S MPv3 私有密码。 仅 SN-MPv3 监控项使用。
items		-		监控项的根元素。
item		-		具体的监控项。

针对监控项元素标签值，查阅主机监控项 标签。

元素元	属性必需	型	围	说明	
tags		-			主机标签的根元素。
tag		-			具体的主机标签。
	tag	x	字符	标	名称。
	value	-	字符	标	值。
macros		-			宏的根元素。
macro		-			具体的宏。
	macro	x			用户宏名称。
	type	-	字符 0	- TEXT (缺省) 宏的类 1 - SE-CRET_TEXT	。
	value	-	字符	用	宏值。
	description	-	字符	用	宏的描述。

元素元	属性必需	型	围	说明
inventory		-		主机资产的根元素。
	<inventory_property>	-		具体的资产属性。 所有可用的资产属性都列在相应的标签下，例如<type>、<name>、<os>(参见上面的示例)。
	inventory_mode	-	字符 -	- DISABLED 0 - MANUAL (缺省) 1 - AUTOMATIC	模式。

主机监控项标签

元素元	属性必需	型	围	说明
items		-		监控项的根元素。
item		-		单独的监控项。
	name	x	字符	监项名称。

元素元	属性必需	型	围	说明
	type	-	字符0	- 项类型。监Zabbix agent 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监控场景步骤) 10 - external 11 - database monitor 12 - IPMI agent 13 - SSH agent 14 -

元素元	属性必需	型	围	说明
	snmp_oid	-	字符	SNMP对象ID。
	key	x	字符	监项键。
	delay	-	字符	更监控项的间隔。秒，带有后缀的时间单位，自定义间隔或用户宏。
	history	-	字符	历数据应存储多长时间的时间单位。带后缀或用户宏的时间单位。

元素元	属性必需	型	围	说明	
	trends	-	字符	趋	数据应存储多长时间的时间单位。带后缀或用户宏的时间单位。
	status	-	字符 0	- enabled 1 - disabled	项监态。
	value_type	-	字符 0	- float 1 - character 2 - log 3 - unsigned integer 4 - text	值得类型。

元素元	属性必需	型	围	说明	
	allowed_hosts	-	字符	如	’type’是2或者19，这是允许给监控项发送数据的IP地址(逗号分隔)列表。
	units	-	字符	返	单位(bps，B)

元素元	属性必需	型	围	说明	
	params	-	文本	如	’type’ 是 13、14，这是 “执行脚本” 的名称 如果’type’ 是 11，这是 “SQL query” 字段 如果’type’ 是 15，这是 “For-mula” 字段。
	ipmi_sensor	-	字符	如	’type’ 是 12，这是 IPMI 传感器 ID。

元素元	属性必需	型	围	说明	
	authtype	-	字符S	H客户端监控项认证类型：如果’type’是13或0 - 密码1 - 键 HTTP客户端监控项认证类型：0 - none1 - basic2 - NTLM	19，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。

元素元	属性必需	型	围	说明	
	publickey	-	字符	如	’type’ 是 13，这是公共密钥文件的名称。
	privatekey	-	字符	如	’type’ 是 13，这是私有密钥文件的名称。
	description	-	文本	监	项说明。
	inventory_link	-	字符 0	- no link 使 *number* - ’host_inventory’ 表里的字段数。	监控项值来填充这个资产记录字段。

元素元	属性必需	型	围	说明	
	logtimefmt	-	字符	日	条目中的时间格式。仅由日志监控项使用。
	interface_ref	-	字符	引	主机接口。
	jmx_endpoint	-	字符	如	’type’是16，这是JMX端点。
	url	-	字符	如	’type’是19，这是URL字符串。
	allow_traps	-	字符0	- 不允许trap-ping. 果’t1 - 允许trap-ping.	pe’是19，属性如允许发送数据给监控项。

元素元	属性必需	型	围	说明	
	follow_redirects	-	字符0	- 不跟随重定向。如果’type’19，跟随HTTP重定向。1 - 跟随重定向。	
headers		-			HTTP(S)请求头的根元素列表，请求头名称作为键，请求头值作为值。仅用于HTTP agent监控项。
header		-			单独的请求头。
	name	x	字符	请	头名称。

元素元	属性必需	型	围	说明	
	value	x	字符	请	头的值。
	http_proxy	-	字符	如	’type’是19，这是HTTP(S)代理连接字符串。
	output_format	-	字符0	- 原样存储。如果’typ1- 转换成JSON。	’是19，怎样处理响应。
	post_type	-	字符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。

元素元	属性必需	型	围	说明
query_fields		-		查询参数的根元素列表。 仅用于HTTP agent监控项。
query_field		-		具体的查询参数。
	name	x	字符	参 名称。
	value	-	字符	参 值。
	request_method	-	字符 0	- GET 1 - POST 2 - PUT 3 - HEAD 'type' 如是 19，这是请求方法。
	retrieve_mode	-	字符 0	- Body. 1 - 请求头。 2 - 请求体和请求头都被存储。 'type' 是 19，响应的什么部分将被存储。

元素元	属性必需	型	围	说明	
	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SSL私钥文件的路径。
	ssl_key_password	-	字符	如	’type’是19，这是SSL密钥文件的密码。

元素元	属性必需	型	围	说明	
	status_codes	-	字符	如	’type’是19，这是以逗号分隔的所要求的HTTP状态码的范围。
	timeout	-	字符	如	’type’是19，这是监控项数据轮询请求超时。

元素元	属性必需	型	围	说明	
	verify_host	-	字符0	- 不校验。如果’ty1 - 校验。	e’为19，则在URL中校验主机名是Common Name字段或主机证书的Subject Alternate Name字段。
	verify_peer	-	字符0	- 不校验。如果’ty1 - 校验。	e’为19，则校验主机证书是否可信。
value map		-			值映射。

元素元	属性必需	型	围	说明	
	name	x	字符	用	监控项的值映射的名称。
applications		-			应用集的根元素。
application		-			单独的应用集。
	name	x			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	属性必需	型	围	说明
	type	x	字符1	- 项值预处理步骤的类型。 MULTIPLIER 2 - RTRIM 3 - LTRIM 4 - TRIM 5 - REGEX 6 - BOOL_TO_DECIMAL 7 - OCTAL_TO_DECIMAL 8 - HEX_TO_DECIMAL 9 - SIMPLE_CHANGE (计算为(收到的值 - 先前值)) 10 - CHANGE_PER_SECOND (计算为(收到的值 - 先前值)/(当前时间- 上次检查的时间)) 11 - XMLPATH 12 - JSONPATH 13 - IN_RANGE 14 - MATCHES_REGEX 15

元素元	属性必需	型	围	说明
	params	x	字符	监项值预处理步骤的参数。
	error_handler	-	字符0	- ORIG-I-NAL_ERROR (de-fault) 预1 - DIS-CARD_VALUE 2 - CUS-TOM_VALUE 3 - CUS-TOM_ERROR 理步骤失败时的动作类型。
	error_handler_params	-	字符	错处理参数。
master_item		-		单个监控项主监控项数据。

元素元	属性必需	型	围	说明	
	key	x	字符	独	监控项主监控项键值。 允许递归最多3个依赖项，最大依赖项计数等于2999。
triggers		-			简单触发器的根元素。
trigger		-			单独的简单触发器。
	针对触发器元素标签的值，查阅主机触发器标签。				

主机低级别发现规则标签

元素元	属性必需	型	围	说明
discovery_rules		-		低级别发现规则的根元素。
discovery_rule		-		单独的低级别发现规则。
	对于大多数元素标签值，请参阅常规监控项的元素标签值。下面仅说明特定于低级别发现规则的标签。			

元素元	属性必需	型	围	说明
	type	-	字符0	- 型。 ZABBIX_PASSIVE (缺省) 监控项 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 16 - JMX 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT

元素元	属性必需	型	围	说明	
	lifetime	-	字符缺	:30d 将删除	再发现的监控项的时间段。秒，带后缀或用户宏的时间单位。
filter				单独的过滤器。	
	evaltype	-	字符0	- 和/或逻辑 用于检查低1 - 与逻辑2 - 或逻辑3 - 自定义公式	别发现规则过滤条件的逻辑。

元素元	属性必需	型	围	说明	
	formula	-	字符	过	条件的自定义计算公式。
	conditions	-			过滤条件的根元素。
conditions		-			过滤条件的根元素。
condition		-			单独的过滤条件。
	macro	x	字符	低	别发现宏名称。
	value	-	字符	过	值：正则表达式或全局正则表达式。

元素元	属性必需	型	围	说明	
	operator	-	字符	条	运算符。

元素元	属性必需	型	围	说明	
	formulaid	x	字符	过	条件ID。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。修改过滤条件时必须由用户定义ID，但在以后请求时将重新生成ID。

元素元	属性必需	型	围	说明	
lld_macro_paths		-			低级别发现宏路径的根元素。
lld_macro_path		-			单独的低级别发现宏路径。
	lld_macro	x	字符	低	别发现宏名称。
	path	x	字符	赋	给对应宏的值选择器。
preprocessing		-			低级别发现规则值预处理。

元素元	属性必需	型	围	说明
step		-		单独的低级别发现规则值预处理步骤。
	针对大部分元素的标签值，查阅主机监控项值预处理的元素标签值。如下仅描述指定给低级别发现值预处理的标签。			
	type	x	字符5	- 项值监预处理步骤类型。 REGEX 11 - XML-PATH 12 - JSON-PATH 15 - NOT_MATCHES_REGEX 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 20 - DIS-CARD_UNCHANGED_H 21 - JAVASCRIPT 23 - PROMETHEUS_TO_JSO 24 - CSV_TO_JSON 25 - STR_REPLACE
trigger_prototypes		-		触发器原型的根元素。

元素元	属性必需	型	围	说明
trigger_prototype		-		单独的触发器原型。
	针对触发器原型元素标签值，查阅主机触发器 标签。			
graph_prototypes		-		图形原型的根元素。
graph_prototype		-		单独的图形原型。
	针对图形原型元素标签值，查阅主机图形 标签。			
host_prototypes		-		主机原型的根元素。
host_prototype		-		单独的主机原型。
	针对主机原型的元素标签值，查阅主机 标签。			
item_prototypes		-		监控项原型的根元素。

元素元	属性必需	型	围	说明
item_prototype		-		单独的监控项原型。
	对于大多数元素标签值，请参阅常规监控项的元素标签值。下面仅说明了监控项原型特有的标签。			
application_prototypes		-		应用程序原型的根元素。
application_prototype		-		单独的应用程序原型。
	name	x		应用程序原型名称。
master_item_prototype		-		单个监控项原型主监控项原型数据。

元素元	属性必需	型	围	说明	
	key	x	字符	从	监控项原型主监控项原型键值。

主机触发器标签

元素元	属性必需	类型	围	说明	
triggers		-			触发器的根元素。
trigger		-			单独的触发器。
	expression	x	字符	触	器表达式。
	recovery_mode	-	字符 0	- 表达式 生成 OK1 - 恢复表达式。 2 - none	件的基础。
	recovery_expression	-	字符	触	器恢复表达式。
	name	x	字符	触	器名称。

元素元	属性必需	类型	围	说明	
	correlation_mode	-	字符 0	- 没有事件关联 关联模式。1 - 按照标签的事件关联	
	correlation_tag	-	字符	用	事件关联的标签名称。
	url	字符	触	器 URL。	
	status	-	字符 0	- enabled 触 1 - disabled	器状态。
	priority	-	字符 0	- 未分类 触发器严 1 - 信息 2 - 告警 3 - 一般严重 4 - 严重 5 - 灾难	性。
	description	-	文本	触	器说明。
	type	-	字符 0	- 单个问题事件。 事件生成类型。1 - 多个问题事件。	
	manual_close	-	字符 0	- 不允许 手工关闭 1 - 允许	题事件。
dependencies		-			依赖项的根元素。
dependency		-			单独的依赖。

元素元	属性必需	类型	围	说明
	name	x	字符	依 关系触发器名称。
	expression	x	字符	依 关系触发表达式。
	recovery_expression	-	字符	依 关系触发恢复表达式。
tags		-		事件标签的根元素。
tag		-		单独的事件标签。
	tag	x	字符	标 名称。
	value	-	字符	标 值。

主机图形标签

元素元	属性必需	型	范围	说明
graphs		-		图形的根元素。

元素元	属性必需	型	范围	说明	
graph		-			单独的图形。
	name	x	字符	图	名称。
	width	-	整型	图	宽度，以像素为单位。用于预览和饼图/爆炸图。
	height	-	整型	图	高度，以像素为单位。用于预览和饼图/爆炸图。
	yaxismin	-	双精度	如果	ymin_type_1’为1，则是Y轴的值最小。

元素元	属性必需	型	范围	说明	
	yaxismax	-	双精度	如果	ymax_type_1’ 为 1，则是 Y 轴的最大值。
	show_work_period	-	字符 0	- no 如 1 - yes	‘type’ 为 0,1，则突出显示非工作时间。
	show_triggers	-	字符 0	- no 如 1 - yes	‘type’ 为 0,1，则将简单触发值显示为一行。
	type	-	字符 0	- 正常 图形类 1 - 柱状图 2 - 饼图 3 - 爆炸图 4 - 3D 饼图 5 - 3D 爆炸图	。
	show_legend	-	字符 0	- no 显 1 - yes	图形图例。

元素元	属性必需	型	范围	说明	
	show_3d	-	字符 0	- 2D 如 1 - 3D	‘type’ 为 2,3，则启用 3D 样式。
	percent_left	-	双精度	如果	type’ 为 0，则显示左轴的百分位线。
	percent_right	-	双精度	如果	type’ 为 0，则显示右轴的百分位线。
	ymin_type_1	-	字符 0	- 计算值 如果’t1 - 固定值 2 - 所选监控项的最后一个值	pe’ 为 0,1，则为 Y 轴的最小值。
	ymax_type_1	-	字符 0	- 计算值 如果’t1 - 固定值 2 - 所选监控项的最后一个值	pe’ 为 0,1，则 Y 轴的最大值。

元素元	属性必需	型	范围	说明
ymin_item_1		-		监控项详细信息。 要求’ymax_type_1为ITEM.
	host	x	字符	监项主机。
	key	x	字符	监项键。
ymax_item_1		-		监控项详细信息。 要求’ymax_type_1为ITEM.
	host	x	字符	监项主机。
	key	x	字符	监项键。
graph_items		x		图形监控项的根元素。
graph_item		x		单独的图形监控项。

元素元	属性必需	型	范围	说明	
	sortorder	-	整型	绘	顺序。首先绘制较小的值。可用于在另一个后面(或前面)绘制线条或区域。
	drawtype	-	字符 0	- 单线 如果图 1 - 填充区域 2 - 粗线 3 - 虚线 4 - 中划线	‘type’为 0，则绘制样式。
	color	-	字符	元	颜色(6 个符号，十六进制)。

元素元	属性必需	型	范围	说明	
	yaxisside	-	字符 0	- 左轴 如果图 1 - 右轴	‘type’为 0,1，则元素所属的 Y 轴位置(左或右)。
	calc_fnc	-	字符 1	- 最小值 如果监控 2 - 平均值 4 - 最大值 7 - 全部(最小值，平均值和最大值，如果图形’类型’为 0) 9 - 最后一个值（如果图形’类型’不是 0,1）	存在多个值，则绘制数据。
	type	-	整型 1	- 该监控项的值按比例表示在饼图上 绘制饼图/爆炸图的类型。2 - 监控项的值表示整个饼图（图形总和）	
item		x			单个监控项。
	host	x	字符	监	项主机。
	key	x	字符	监	项键。

主机 web 场景标签

元素元	属性必需	型	围	说明	
httptests		-			Web场景的根元素。
httptest		-			单独的web场景。
	name	x	字符	w	b场景名称。
	delay	-	字符	执	Web方案的频率。秒，带后缀或用户宏的时间单位。
	attempts	-	整型 1	10 执	Web场景步骤的尝试次数。

元素元	属性必需	型	围	说明
	agent	-	字符	客端代理。Zabbix将假装成为选定的浏览器。当网站为不同的浏览器返回不同的内容时，这非常有用。
	http_proxy	-	字符	指要使用的HTTP代理，使用以下的格式：http://[user

元素元	属性必需	型	围	说明
variables		-		场景步骤中用到的场景级别变量(宏)的列表。
variable		-		单独的变量。
	name	x	文本	变名称。
	value	x	文本	变值。

元素元	属性必需	型	围	说明
headers		-		HTTP请求头列表-发起HTTP请求时要发送的。由于请求头会出现在HTTP协议中，请求头列表中的请求头格式要相同。
header		-		单独的请求头。
	name	x	文本	请 头名称。
	value	x	文本	请 头的值。

元素元	属性必需	型	围	说明	
	status	-	字符 0	- en- abled w1 - disabled	b 场景状态。
	authentication	-	字符 0	- none 认 1 - basic 2 - NTLM	方法。
	http_user	-	字符	认	用户名。
	http_password	-	字符	指	用户名的认证密码。
	verify_peer	-	字符 0	- no 验 1 - yes	Web 服务器的 SSL 证书。
	verify_host	-	字符 0	- no 验 1 - yes	Web 服务器证书的 Com- mon Name 字段或 Sub- ject Al- ter- nate Name 字段是否匹配。

元素元	属性必需	型	围	说明
	ssl_cert_file	-	字符	用客户端身份验证的SSL证书文件的名称。
	ssl_key_file	-	字符	用客户端身份验证的SSL私钥文件的名称。
	ssl_key_password	-	字符	S L私钥文件密码。
steps		x		Web场景步骤的根元素。
step		x		单独的web场景步骤。

元素元	属性必需	型	围	说明
	name	x	字符	w b 场景步骤名称。
	url	x	字符	用 监控的URL。
query_fields		-		查询字段的列表 - 在发送HTTP请求时会被加到URL的HTTP字段。
query_field		-		单独的查询列表字段。
	name	x	字符	查 字段名称。
	value	-	字符	查 字段的值。

元素元	属性必需	型	围	说明	
posts		-			’Post’ 变量字符串 (原始 post 数据) 或者’Post’ 变量列表 (字段数据表单)。
post_field		-			单独的 post 字段。
	name	x	字符	P	st 字段名称。
	value	x	字符	P	st 字段值。

元素元	属性必需	型	围	说明
variables		-		应在此步骤之后应用的步骤级变量(宏)列表。 如果变量值具有’regex：’前缀，则根据’regex：’前缀后面的正则表达式模式从该步骤返回的数据中提取其值。

元素元	属性必需	型	围	说明
variable		-		单独的变量。
	name	x	字符	变 名称。
	value	x	字符	变 值。
headers		-		HTTP请求头列表-发起HTTP请求时要发送的。由于请求头会出现在HTTP协议中，请求头列表中的请求头格式要相同。

元素元	属性必需	型	围	说明	
header		-			单独的请求头。
	name	x	字符	请	头名称。
	value	x	字符	请	头的值。
	follow_redirects	-	字符 0	- NO H1 - YES (缺省)	TP 跟随重定向。
	retrieve_mode	-	字符 0	- BODY (缺省) HTT1 - HEADERS 2 - BOTH	相应返回模式。
	timeout	-	字符 缺	: 15s 步骤 执	超时时间。秒，带有后缀或用户宏的时间单位。
	required	-	字符	所	求的字符串。如果为空则忽略。

元素元	属性必需	型	围	说明
	status_codes	-	字符	以号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

脚注

[1] 对于字符串值，只导出字符串（例如 “ZABBIX_ACTIVE”），而不使用此表中使用的编号。此表中范围值（对应于 API 值）的数字仅用于排序.

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

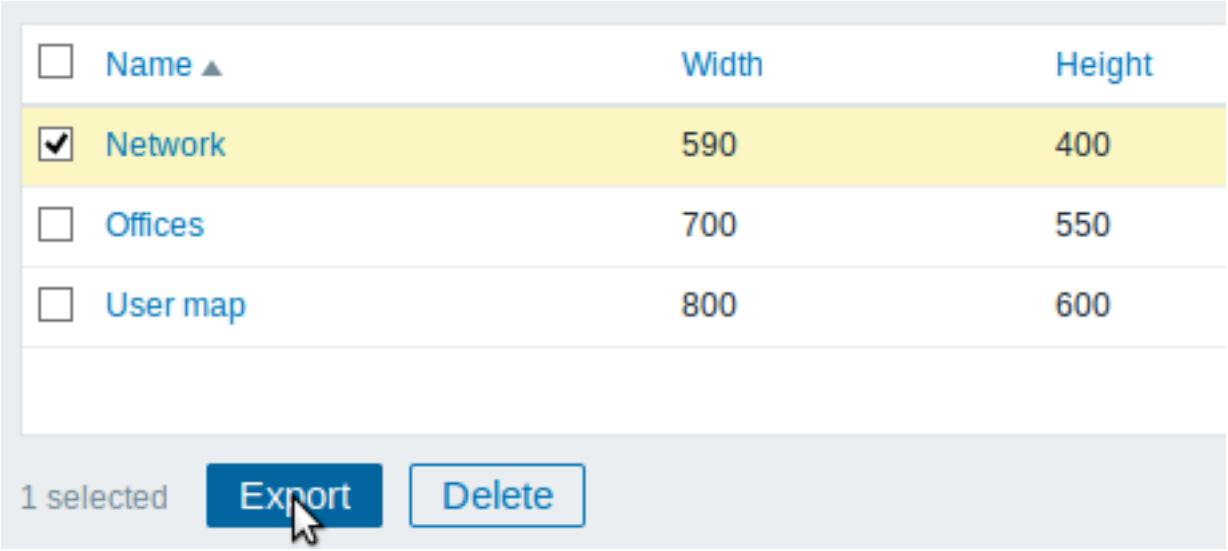
未导出的是主机组，主机，触发器，其他拓扑图或可能与导出的拓扑图相关的任何其他元素。因此，如果缺少拓扑图所引用的元素中的任何一个，导入将失败。

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

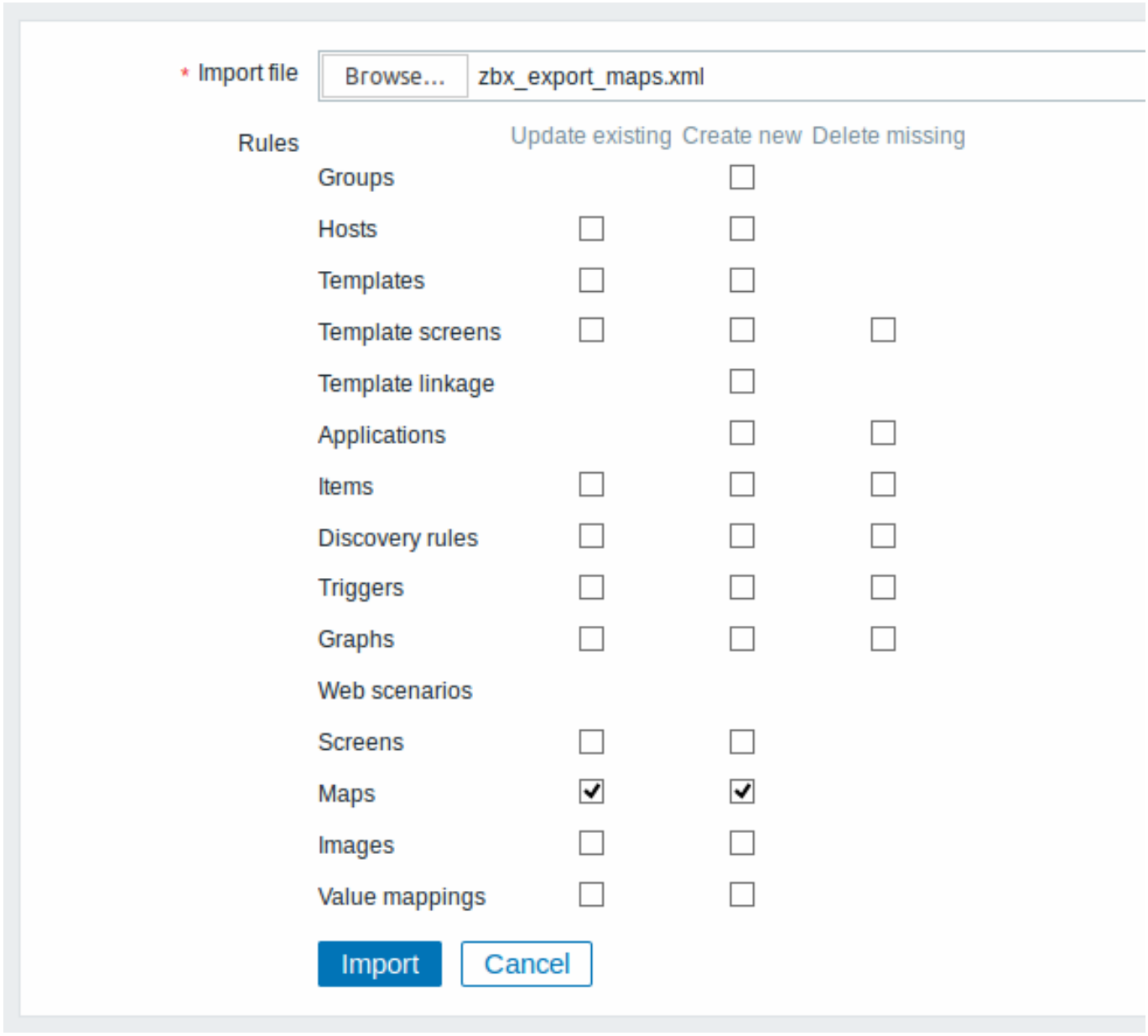


选中的拓扑图以默认名称 *zabbix_export_maps.xml* 导出到本地的 XML 文件里。

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>5.0</version>
    <date>2020-04-22T09:22:17Z</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>
            <show_suppressed>0</show_suppressed>
            <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。
	show_suppressed	整型 0	- no 显 1 - yes 由于主机维护而被抑制(未显示)的问题。

元素元	属性类型	范围	说明	
	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。
	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。

元素元	属性类型	范围	说明	
	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
	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”

元素元	属性类型	范围	说明
	label_string_host	字符	如“label_type_host=5”，这是主机元素的自定义标签。
	label_string_hostgroup	字符	如“label_type_host=5”，这是主机组元素的自定义标签。
	label_string_trigger	字符	如“label_type_trig=5”，这是触发元素的自定义标签。

元素元	属性类型	范围	说明	
	label_string_map	字符	如	“label_type_map = 5”，则是拓扑图元素的自定义标签
	label_string_image	字符	如	“label_type_image = 5”，则是图像元素的自定义标签
	expand_macros	整型 0	- no 在 1 - yes	扑图配置中展开标签中的宏。
	background	id		如果“im-agetype = 2”，则是背景图像的ID（如果有）

元素元	属性类型	范围	说明	
	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
	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。

元素元	属性类型	范围	说明	
	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
	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

元素元	属性类型	范围	说明	
	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	应用集名称过滤器。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

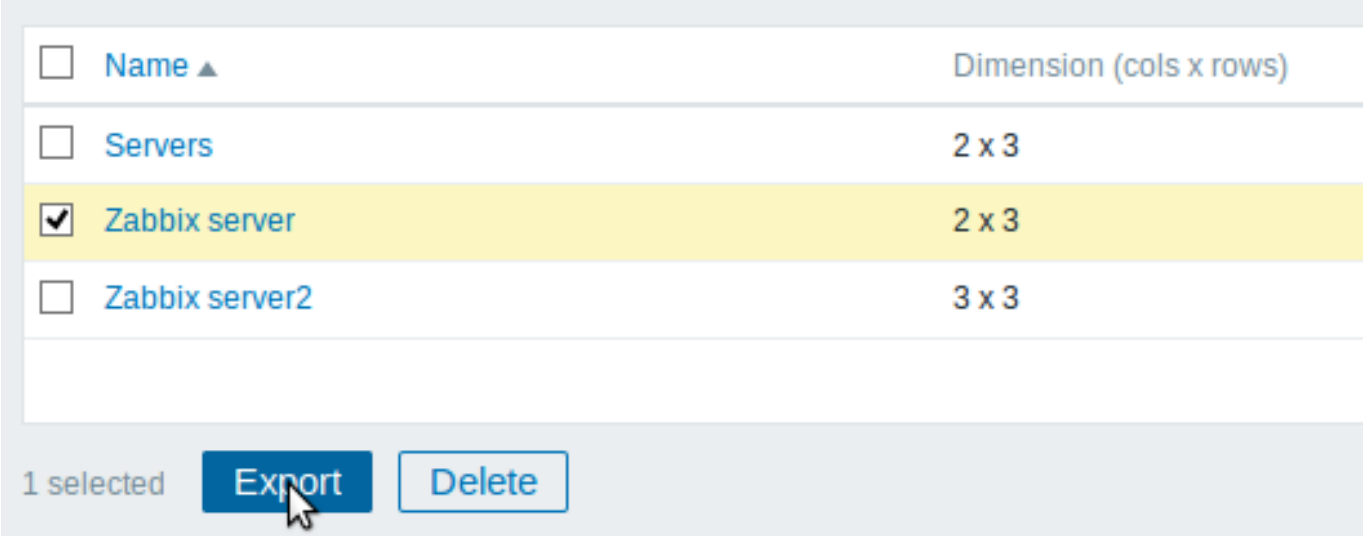
聚合图形导出（export） 包含聚合图形的结构 - 所有聚合图形设置和所有聚合图形元素及其配置。

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

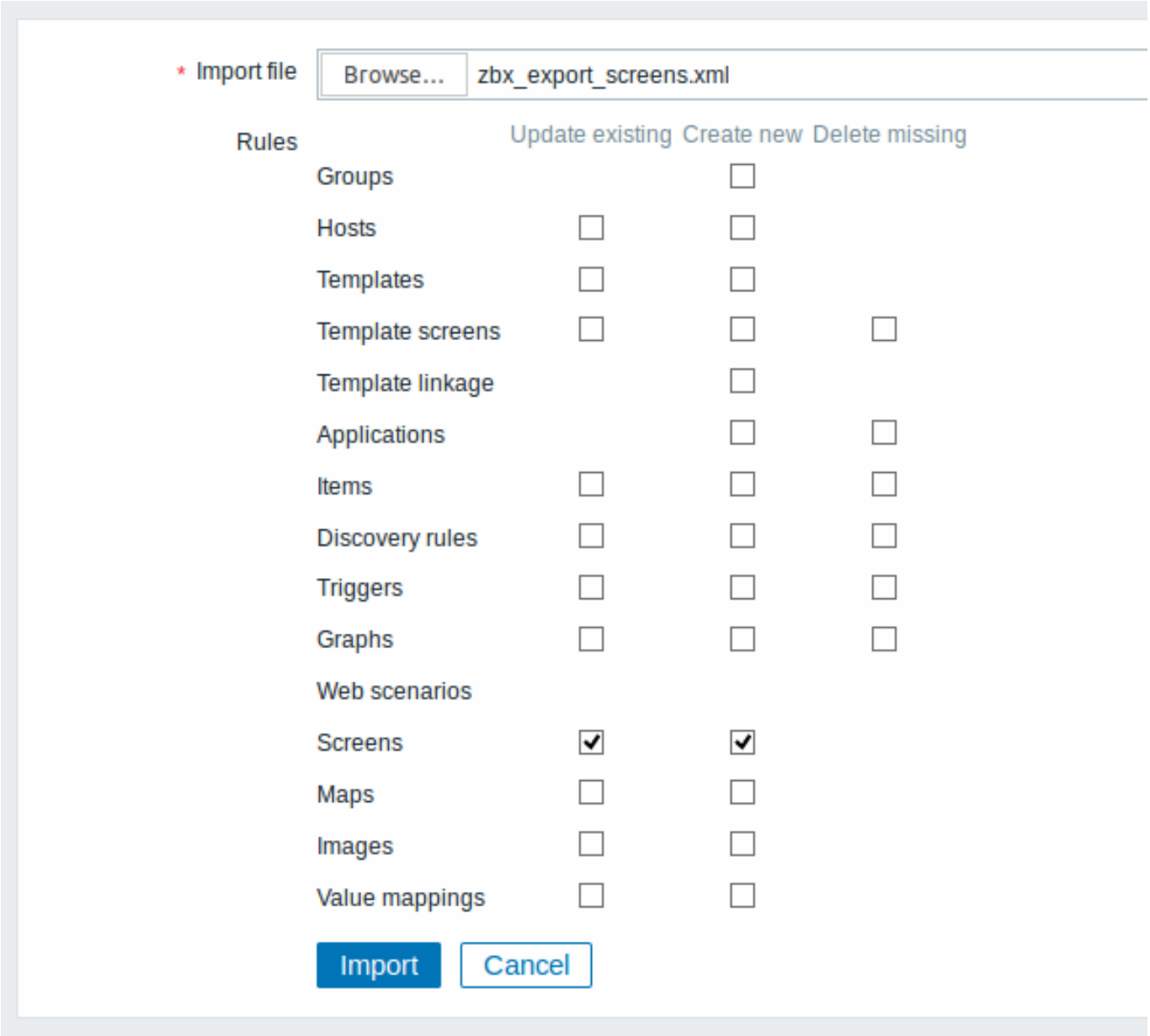


选定的聚合图形将导出到本地 XML 文件，默认名称为 *zabbix_export_screens.xml*。

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>5.0</version>
    <date>2020-04-22T09:22:17Z</date>
    <screens>
        <screen>
            <name>Zabbix server</name>
            <hsize>2</hsize>
            <vsize>3</vsize>
            <screen_items>
                <screen_item>
```

```
                    <resourcetype>0</resourcetype>
                    <width>300</width>
                    <height>80</height>
                    <x>0</x>
                    <y>0</y>
                    <colspan>1</colspan>
                    <rowspan>1</rowspan>
                    <elements>0</elements>
                    <valign>0</valign>
                    <halign>0</halign>
                    <style>0</style>
                    <url/>
                    <dynamic>1</dynamic>
                    <sort_triggers>0</sort_triggers>
                    <resource>
                        <name>CPU load</name>
                        <host>Zabbix host</host>
                    </resource>
                    <max_columns>3</max_columns>
                    <application/>
                </screen_item>
                <screen_item>
                    <resourcetype>0</resourcetype>
                    <width>300</width>
                    <height>80</height>
                    <x>1</x>
                    <y>0</y>
                    <colspan>1</colspan>
                    <rowspan>1</rowspan>
                    <elements>0</elements>
                    <valign>0</valign>
                    <halign>0</halign>
                    <style>0</style>
                    <url/>
                    <dynamic>1</dynamic>
                    <sort_triggers>0</sort_triggers>
                    <resource>
                        <name>CPU utilization</name>
                        <host>Zabbix host</host>
                    </resource>
                    <max_columns>3</max_columns>
                    <application/>
                </screen_item>
            </screen_items>
        </screen>
    </screens>
</zabbix_export>
```

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。

6 媒介类型

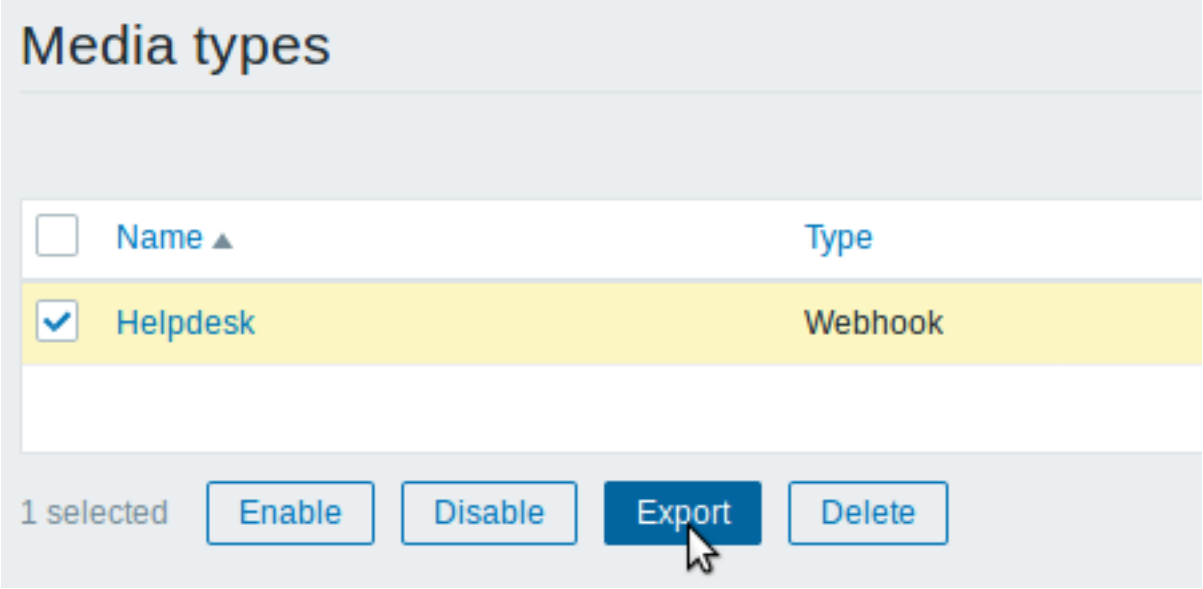
概述

媒介类型是与所有相关对象和对象关系一起导出的

导出

导出媒介类型步骤如下：

- 进入: 管理 → 媒介类型
- 标记要导出的媒介类型的复选框
- 点击列表下面的 导出

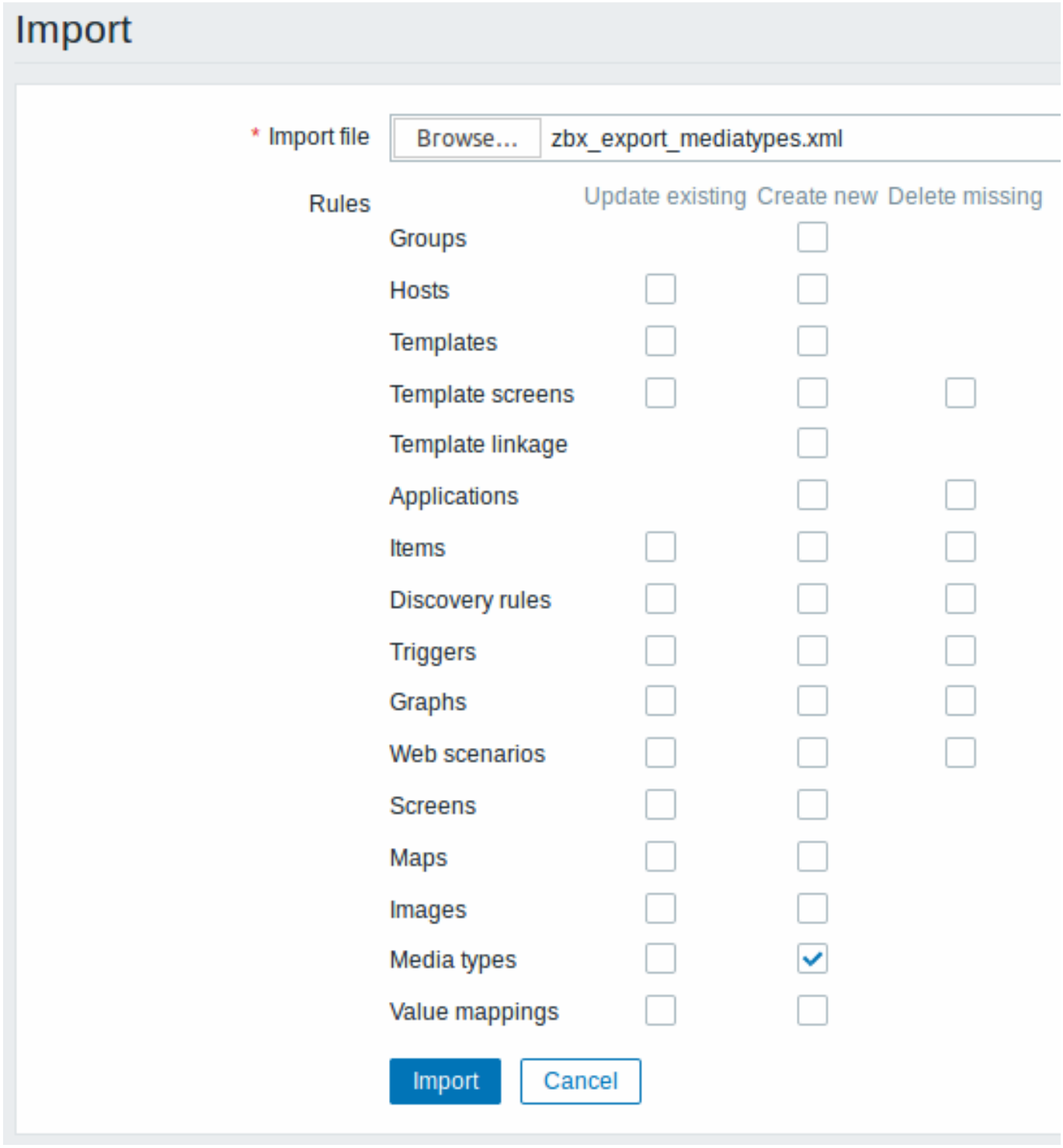


选择的媒介类型将会导出到本地 XML 文件，默认名为 *zbx_export_mediatypes.xml*

导入

导入媒介类型步骤如下：

- 进入: 管理 → 媒介类型
- 点击右边的导入
- 选择要导入的文件
- 在导入规则中标记所需的选项
- 点击导入



导入成功或失败的消息将在前端页面上显示。

导入规则：

规则	说
更新现有元素	现有元素将从导入文件中获取的数据更新。否则它们将不会被更新。
新建元素	使用导文件中的数据添加新元素。否则将不会添加它们。
删除缺失元素	现有元素若在导入的文件中，该元素将会被删除。否则不会删除它们。

导入格式

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2020-01-24T06:44:38Z</date>
    <media_types>
```

```
        <media_type>
            <name>Slack chat</name>
            <type>WEBHOOK</type>
            <parameters>
                <parameter>
                    <name>channel</name>
                    <value>{ALERT.SENDTO}</value>
                </parameter>
                <parameter>
                    <name>text</name>
                    <value>{ALERT.MESSAGE}</value>
                </parameter>
                <parameter>
                    <name>username</name>
                    <value>bot</value>
                </parameter>
            </parameters>
            <script>var req = new CurlHttpRequest();
req.AddHeader('Content-Type: application/x-www-form-urlencoded');

Zabbix.Log(127, 'webhook requqest value='+value);

req.Post('https://hooks.slack.com/services/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',
  'payload='+value
);

Zabbix.Log(127, 'response code: '+req.Status());

return JSON.stringify({
  'tags': {
    'delivered': 'slack'
  }
});</script>
            <process_tags>YES</process_tags>
            <show_event_menu>YES</show_event_menu>
            <event_menu_url>https://www.zabbix.com</event_menu_url>
            <event_menu_name>Slack message</event_menu_name>
            <description>Slack chat messages.</description>
        </media_type>
    </media_types>
</zabbix_export>
```

元素标签

元素标签值详解

元素元	属性必须	型	范围 ^**[1](脚注)**^	说明
media_types		-		media_types 的根元素。
media_type		-		单独的 me-dia_type。

元素元	属性必须	型	范围 ^**[1](	脚注)**^ 说明	
	name	x	字符串	媒介	型名称。
	type	x	字符串 0	电子邮件 媒体类型使用 1 - 短消息 2 - 脚本 4 - WEBHOOK	传输。
	status	-	字符串 0	启用 (默认) 媒介类型是否 1 - 禁用	用。
	max_sessions	-	整型 S	S 可能的值: 1 - (默认) 可以并行处理的 其他媒介类型的可能值: 0-100, 0 - 无限制	大警报数。
	attempts	-	整型 1	10 (默认: 3) 发送警	的最大尝试次数。
	attempt_interval	-	字符串 0-	0s (默认: 10s) 重试的时	间隔。 接受秒和带后缀的时间单位。
	description	-	字符串	媒介	型说明。
message_templates		-			媒介类型消息模板的根元素。

元素元	属性必须	型	范围 ^**[1](	脚注)**^ 说明	
message_template		-			单独的消息模板。
	event_source	x	字符串 0	触发器 事件来源。1 - 自动发现 2 - 自动注册 3 - 内部	
	operation_mode	x	字符串 0	问题 操作模式 1 - 恢复 2 - 更新	
	subject	-	字符串	信息	题。
	message	-	字符串	信息	体。
Used only by e-mail media type					
	smtp_server	x	字符串	SM	P 服务器。
	smtp_port	-	整型默	: 25 SMT	服务器连接端口。
	smtp_helo	x	字符串	SM	P helo 信息。
	smtp_email	x	字符串	发送	知的电子邮件地址。
	smtp_security	-	字符串 0	NONE (默认) SMTP1 - STARTTLS 2 - SSL_OR_TLS	连接安全级别。

元素元	属性必须	型	范围 ^**[1](	脚注)**^ 说明	
	smtp_verify_host	-	字符串 0	否 (默认) SSL 验证 1 - 是	MTP 的主机。当 smtp_security 的值为 START-TLS 或 SSL_OR_TLS 时，为可选项。
	smtp_verify_peer	-	字符串 0	否 (默认) SSL 验证 1 - 是	MTP 的对等体。当 smtp_security 的值为 START-TLS 或 SSL_OR_TLS 时，为可选项。
	smtp_authentication	-	字符串 0	无 (默认) SMTP 身 1 - 密码	验证方法。
	username	-	字符串	用户	。
	password	-	字符串	密码	证。
	content_type	-	字符串 0	TEXT 消息 1 - HTML (默认)	式。
Used only by SMS media type					
	gsm_modem	x	字符串	Se	ial GSM mo-dem 的设备名称。

元素元	属性必须	型	范围 ^**[1](	脚注)**^ 说明	
Used only by script media type					
	script name	x	字符串	脚本	称。
parameters		-			脚本参数的根元素。
parameter		-			单独的脚本参数。
Used only by webhook media type					
	script	x	字符串	脚本	
	timeout	-	字符串 1-	0s (默认: 30s) Java	cript 脚本 HTTP 请求超时时间的
	process_tags	-	字符串 0	否 (默认) 是否处理返 1 - 是	标记。

元素元	属性必须	型	范围 ^**[1](	脚注)**^ 说明	
	show_event_menu	-	字符串 0	否 (默认) 如果 {E1 - 是	ENT.TAGS.*} 在字段 event_menu_u 和 event_menu_n 成功解析, 该字段表示事件菜单中存在条目。
	event_menu_url	-	字符串	事件	单项的 URL。支持 {EVENT.TAGS.* 宏。
	event_menu_name	-	字符串	事件	单项的名称。支持 {EVENT.TAGS.* 宏。
parameters		-			webhook 媒介类型参数的根元素。

元素元	属性必须	型	范围 ^**[1](	脚注)**^ 说明	
parameter		-			单独的 web-hook 媒介类型参数。
	name	x	字符串	We	hook 参数名称。
	value	-	字符串	We	hook 参数值。

脚注

[1] 对于字符串值，仅将导出字符串（例如 EMAIL），而无需使用此表中的编号。该表中范围值 (对应于 API 值) 的数字仅用于排序。

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”。

Interface creation when adding hosts

When hosts are added as a result of network discovery, they get interfaces created according to these rules:

- the services detected - for example, if an SNMP check succeeded, an SNMP interface will be created
- if a host responded both to Zabbix agent and SNMP requests, both types of interfaces will be created
- if uniqueness criteria are Zabbix agent or SNMP-returned data, the first interface found for a host will be created as the default one. Other IP addresses will be added as additional interfaces.
- if a host responded to agent checks only, it will be created with an agent interface only. If it would start responding to SNMP later, additional SNMP interfaces would be added.
- if 3 separate hosts were initially created, having been discovered by the ”IP” uniqueness criteria, and then the discovery rule is modified so that hosts A, B and C have identical uniqueness criteria result, B and C are created as additional interfaces for A, the first host. The individual hosts B and C remain. In *Monitoring* → *Discovery* the added interfaces will be displayed in the ”Discovered device” column, in black font and indented, but the ”Monitored host” column will only display A, the first created host. ”Uptime/Downtime” is not measured for IPs that are considered to be additional interfaces.

Changing proxy setting

The hosts discovered by different proxies are always treated as different hosts. While this allows to perform discovery on matching IP ranges used by different subnets, changing proxy for an already monitored subnet is complicated because the proxy changes must be also applied to all discovered hosts.

For example the steps to replace proxy in a discovery rule:

1. disable discovery rule
2. sync proxy configuration
3. replace the proxy in the discovery rule
4. replace the proxy for all hosts discovered by this rule
5. enable discovery rule

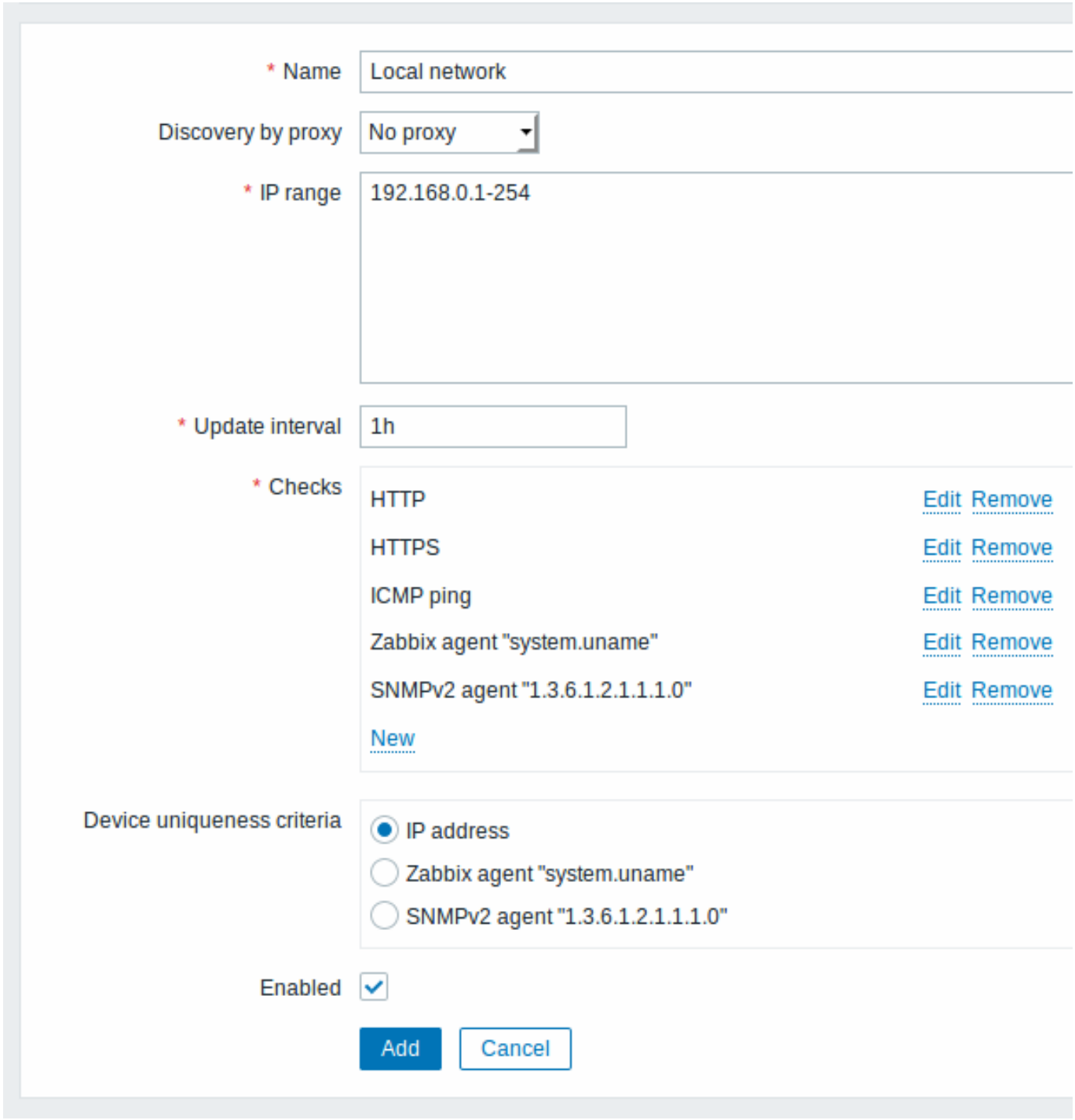
1 配置网络发现规则

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（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



所有红色的星号为必填字段.

参数描	
Name	规则的惟一名称. 例如：“Local network”。

参数	描
Discovery by proxy	执行自动发现的方式: **no proxy** - Zabbix server 执行自动发现 **<proxy name>** - proxy 执行自动发现

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	这参数定义Zab-bix多久执行一次规则.该间隔是在前一个发现实例的执行结束后进程测量的,因此不存在重叠.自从Zab-bix 3.4.0起支持时间后缀列如:30s,1m,2h,1d,自从Zab-

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*）使用复	框标记规则是激活的，将由 Zabbix server 执行. 如果未标记，则规则未激活，它不会被执行.

真实使用场景

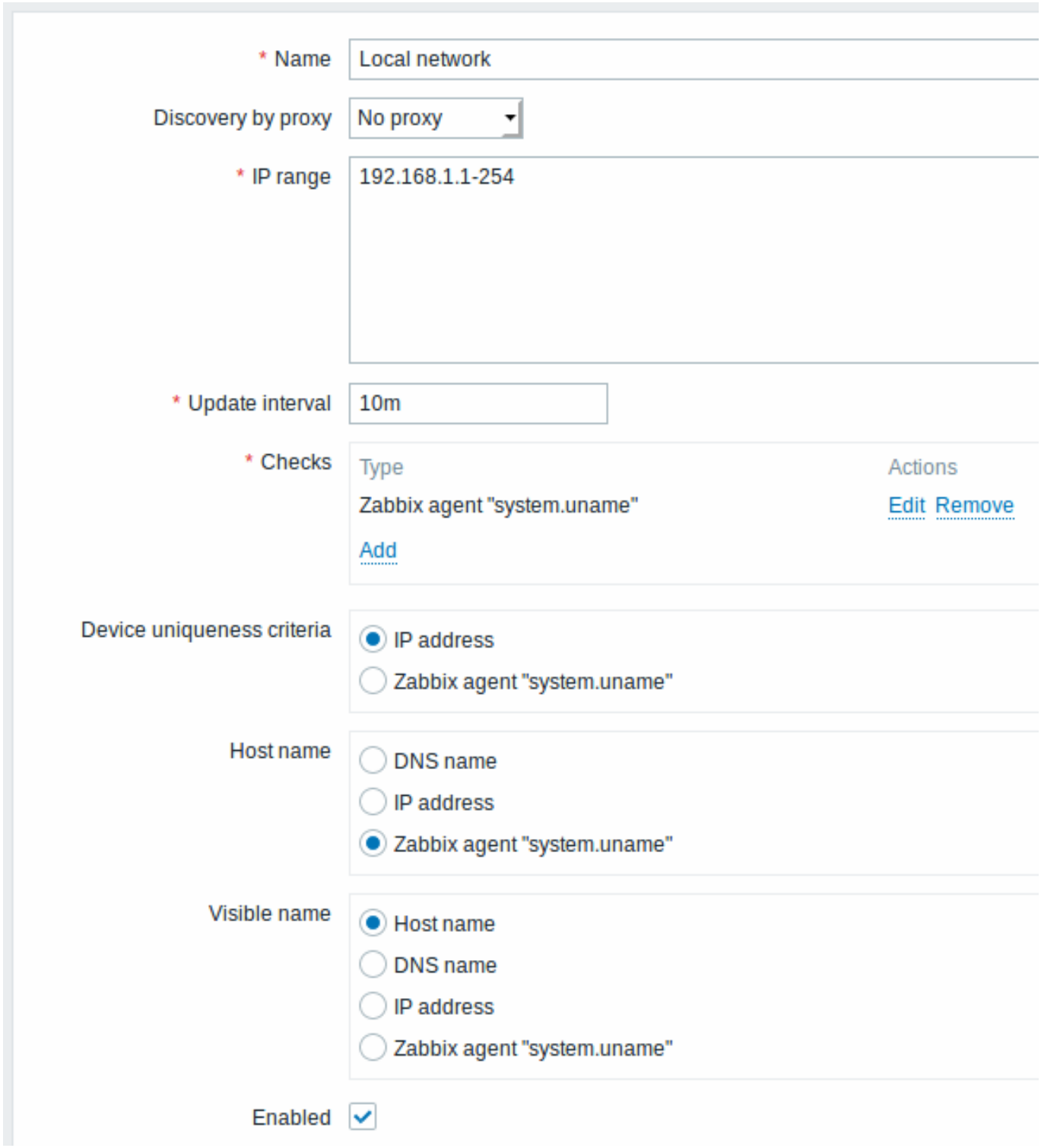
例如我们设置 IP 段为 192.168.1.1-192.168.1.254 的网络发现规则。

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。



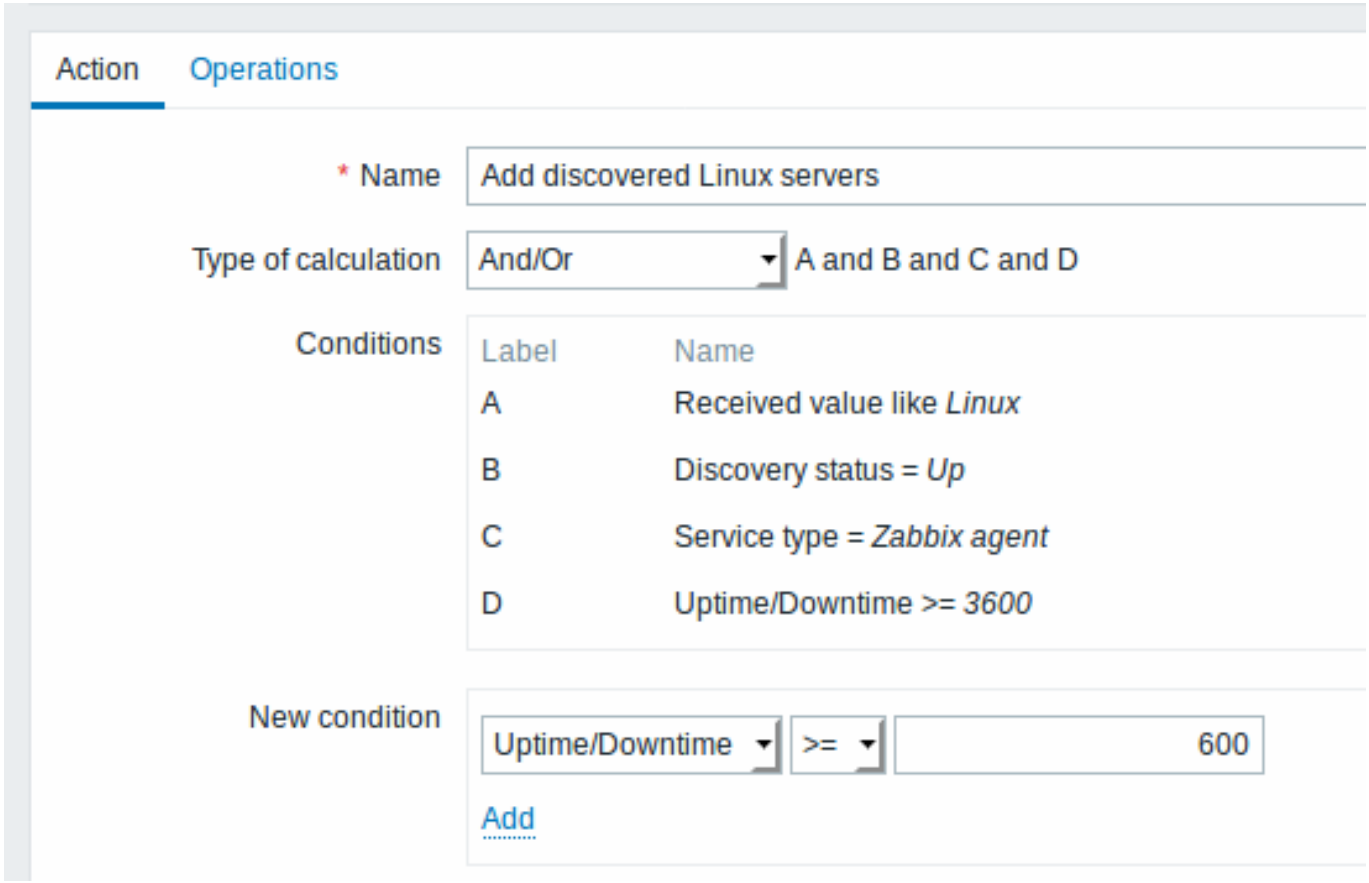
Zabbix 试图通过连接 Zabbix agents 并获取 **system.uname** 键值来发现 IP 段为 192.168.1.1-192.168.1.254 中的主机。根据不同键值来对应不同的操作系统的不同操作。根据不同键值来对应不同的操作系统的不同操作。例如将 Windows 服务器链接到 Template OS Windows，将 Linux 服务器链接到 Template OS Linux。

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。

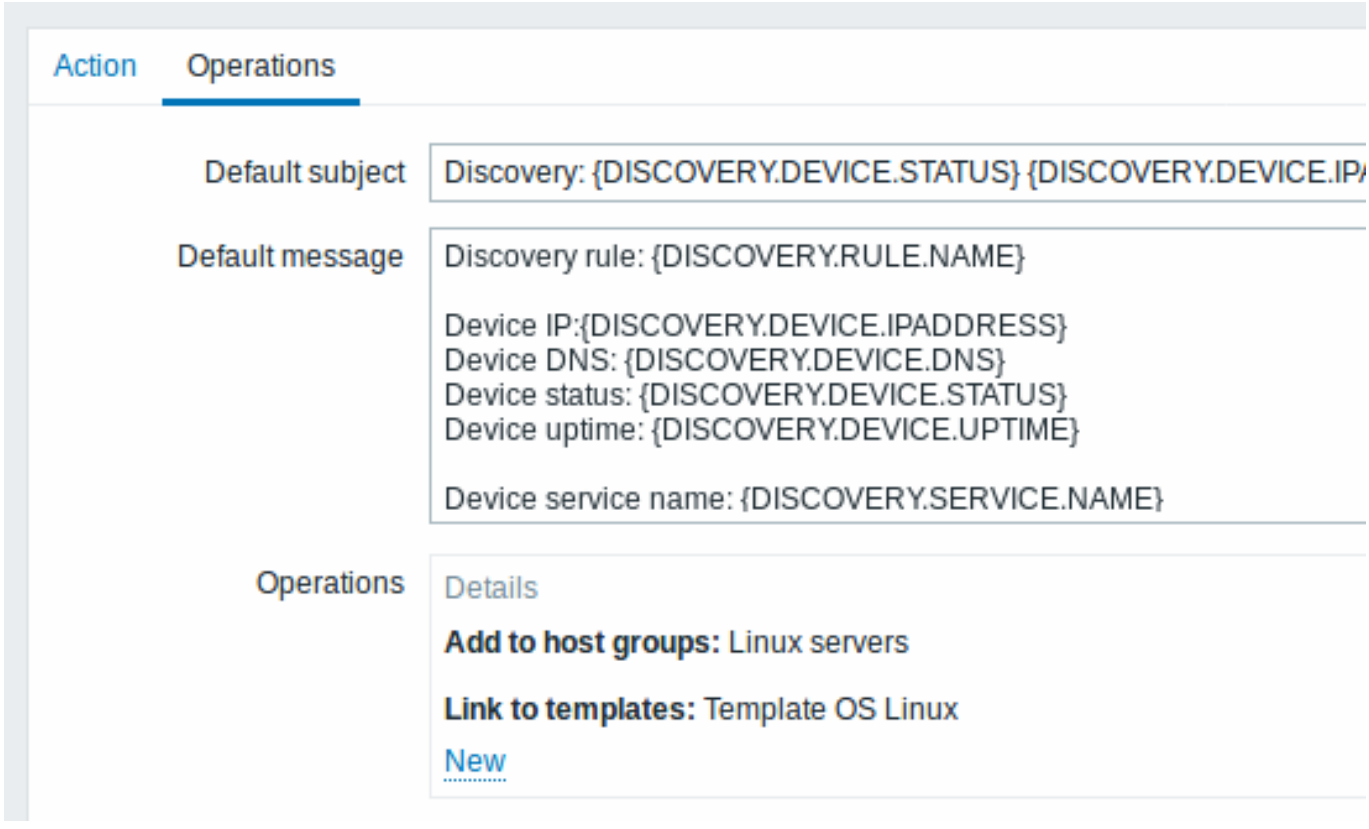
步骤 2

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

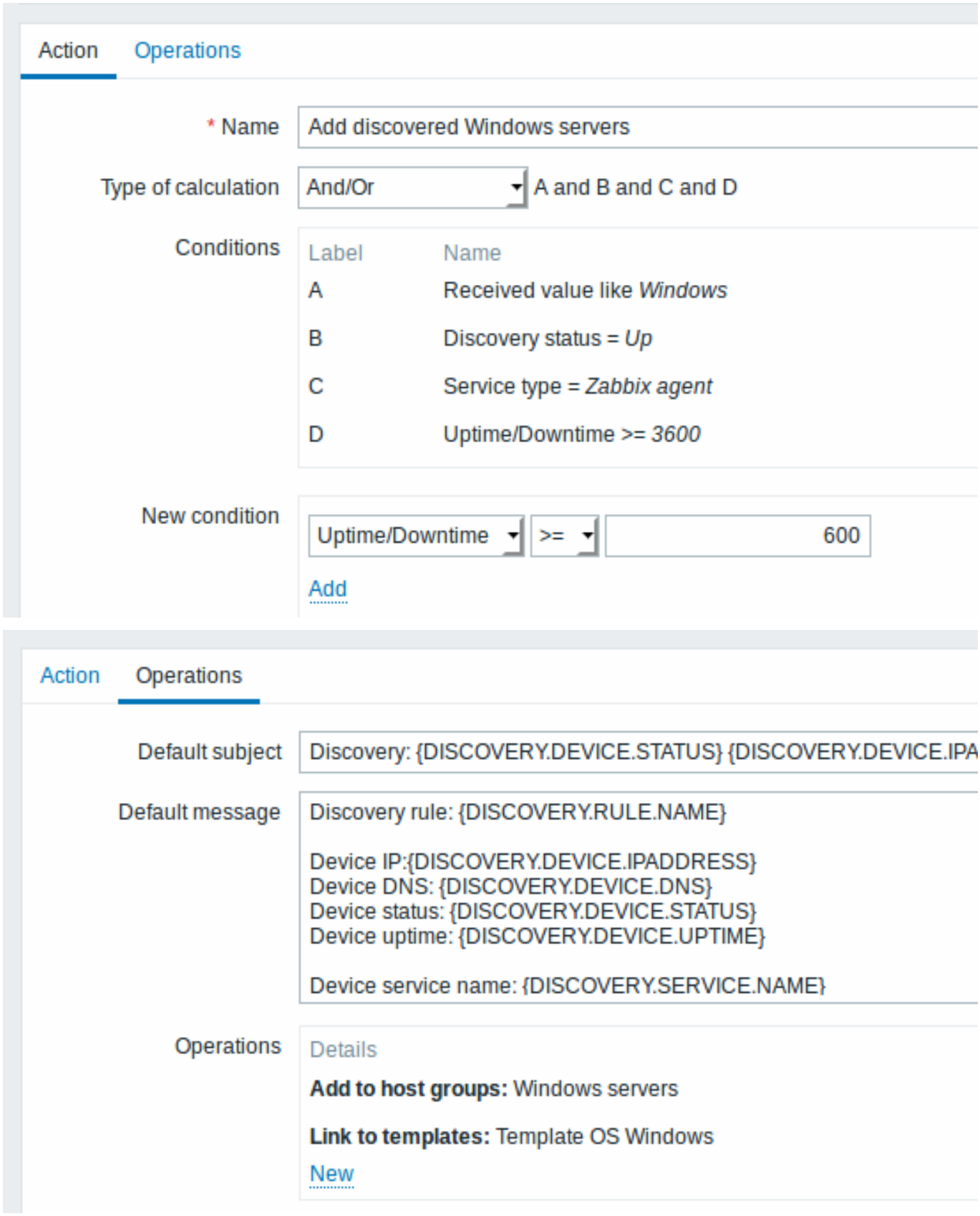


该动作 (action) 将执行以下操作：

- 将发现的主机添加到 “Linux servers” 组（如果以前未添加主机，则自动添加主机）
- 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。

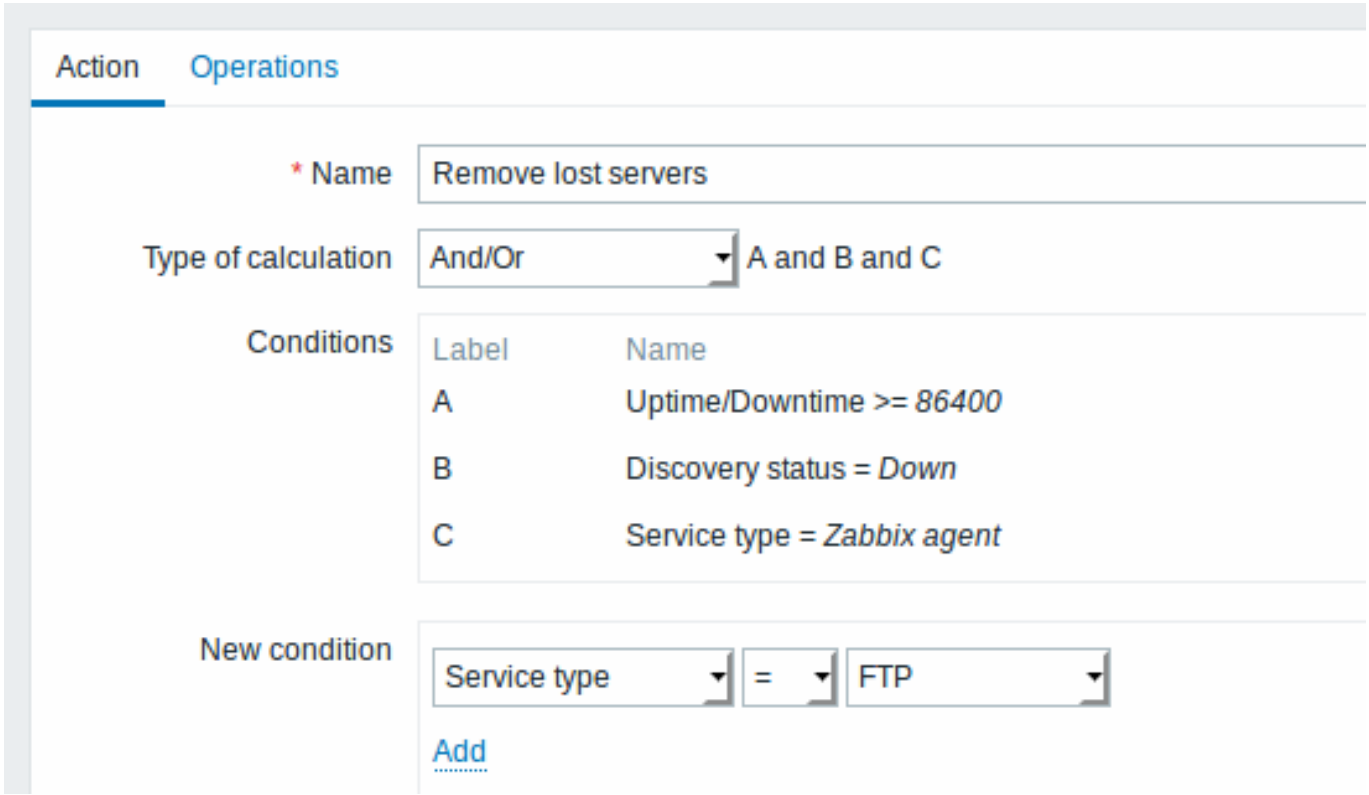
步骤 3

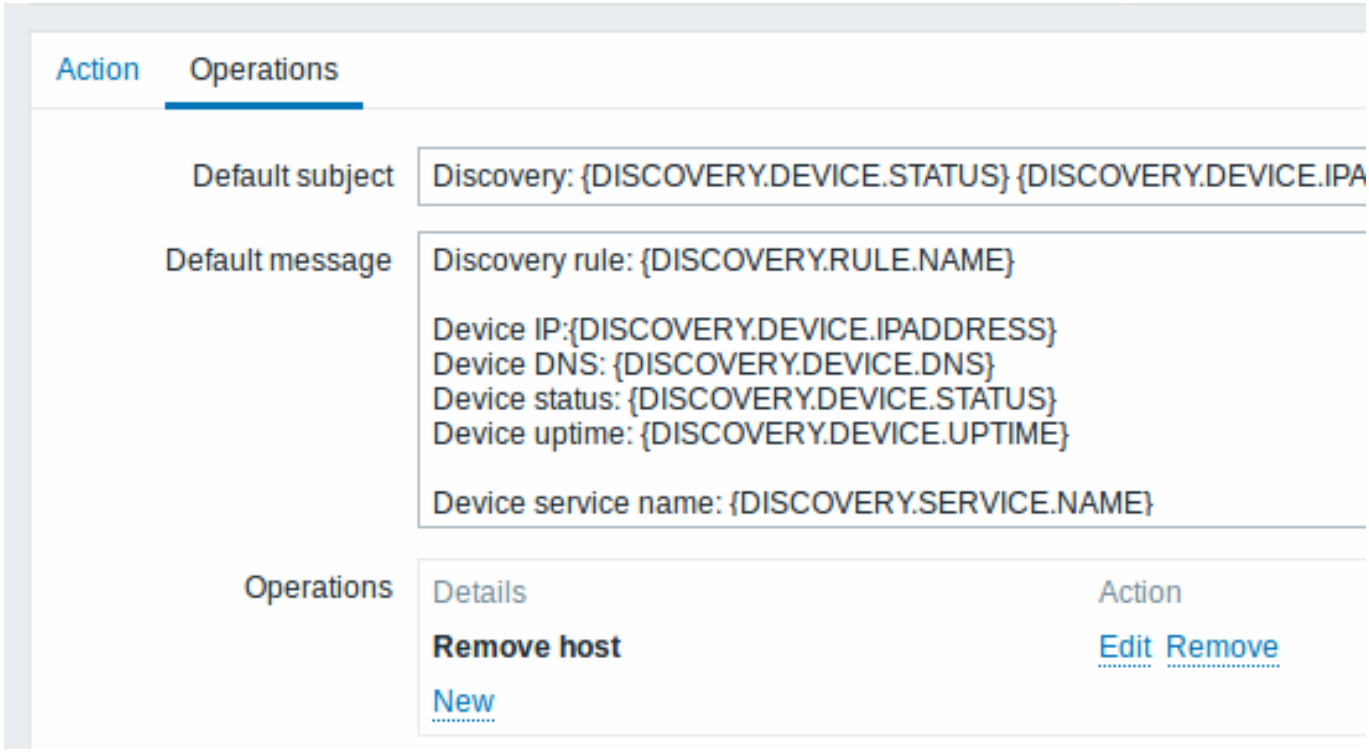
定义动作 (action) 将所发现的 Windows 服务器添加到相应的组/模板



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. 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:**
> 设置网络发现并不需要激活 agent 来实现自动注册

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。:::

创建的主机被添加到发现的主机组中 (默认情况下，可在 “管理 → 通用 → 其他” 中配置)。如果需要将主机添加到其他主机组中，需要添加 “从主机组中移除”(指定 “已发现的主机”) 和 “添加到主机组”(指定其他主机组)，因为主机必须属于某个主机组。d

安全的自动注册

通过使用加密连接配置基于 psk 的身份验证，可以实现自动注册的安全方式。

加密级别的全局配置方式 *Administration* → 一般, 可以通过右边的下拉菜单访问” 自动注册” 部分。可以选择不加密\TLS 加密与 PSK 认证方式，或两者都选择 (这样一些主机可以注册不加密，而其他人通过加密)。

PSK 认证在添加主机前需要通过 Zabbix 服务器验证。如果成功，链接/添加主机，从/到主机的连接将被设置为’PSK’，仅使用与全局自动注册设置相同的身份/预共享密钥。

> **Attention:**
> 为了确保使用代理的安装自动注册的安全性，Zabbix server 和 Zabbix proxy 之间应该启用加密。

使用 DNS 作为默认接口

HostInterface 和 HostInterfaceItem配置参数允许在自动注册期间为主机接口指定自定义值。

更具体地说，如果主机应该以 DNS 名称 (而不是 IP 地址) 作为默认代理接口进行自动注册，则它们是有用的。在这种情况下，应该指定 DNS 名称，或者作为 HostInterface 或 HostInterfaceItem 参数的值返回。注意，如果两个参数之一的值发生变化，自动注册的主机接口就会更新。因此，可以将缺省接口更新为另一个 DNS 名称或更新为一个 IP 地址。但是，为了使更改生效，必须重新启动代理。

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 只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
```

通过这种方式，您可以确保主机元数据将包含 “Linux” 或 “Windows”，这取决于代理运行在哪个主机上。本例中的主机元数据示例:

```
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
```

不要忘记在对配置文件进行任何更改后重新启动代理。

示例 2

第一步

使用主机元数据允许一些基本的保护，以防止不受欢迎的主机注册。

前置配置

在前端创建一个动作，使用一些难以猜测的秘密代码禁止不需要的主机注册:

- 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
```

请注意，由于数据是以明文形式传输的，仅此方法不能提供强大的保护。为了使更改立即生效，需要重新加载配置缓存。

Agent 配置

在配置文件中增加下面一行代码:

```
HostMetadata=Linux      21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

其中”Linux” 是一个平台，字符串的其余部分是难以猜测的秘密文本。不要忘记在对配置文件进行任何更改后重新启动代理。

第二步

可以为已经注册的主机添加额外的监控项。

前端配置

在前端更新操作:

- 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 配置

在配置文件中增加下面一行代码:

```
HostMetadata=MySQL on Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

不要忘记在对配置文件进行任何更改后重新启动代理。

3 自动发现（**LLD**）

概述

自动发现（LLD）提供了一种在计算机上为不同实体自动创建监控项，触发器和图形的方法。例如，Zabbix 可以在你的机器上自动开始监控文件系统或网络接口，而无需为每个文件系统或网络接口手动创建监控项。此外，可以配置 Zabbix 根据定期执行发现后的得到实际结果，来移除不需要的监控。

用户可以自己定义发现类型，只要它们遵循特定的 JSON 协议。

发现过程的一般架构如下。

首先，用户在 “配置”→“模板”→“发现” 列中创建一个发现规则。发现规则包括（1）发现必要实体（例如，文件系统或网络接口）的项和（2）应该根据该项的值创建的监控项，触发器和图形的原型

发现所需实体的项就像其他地方所看到的常规项一样：服务器（server）向 Zabbix agent（或者对应该项的其他类型的设置）查询该项的值，agent 以文本值进行响应。区别在于 agent 响应的值应该包含特定 JSON 格式的已发现实体的列表。虽然这个列表的详细信息仅对自定义发现检查来说很重要，但有必要知道返回的值包含宏 - > 值对的列表。例如，项目 “net.if.discovery” 可能会返回两对：“{#IFNAME}” - >“lo” 和 “{#IFNAME}” - >“eth0”。

这些宏用于名称，键值和其他原型字段中，然后用接收到的值为每个发现的实体创建实际的监控项，触发器，图形甚至主机。请参阅使用 LLD 宏选项的完整列表。。

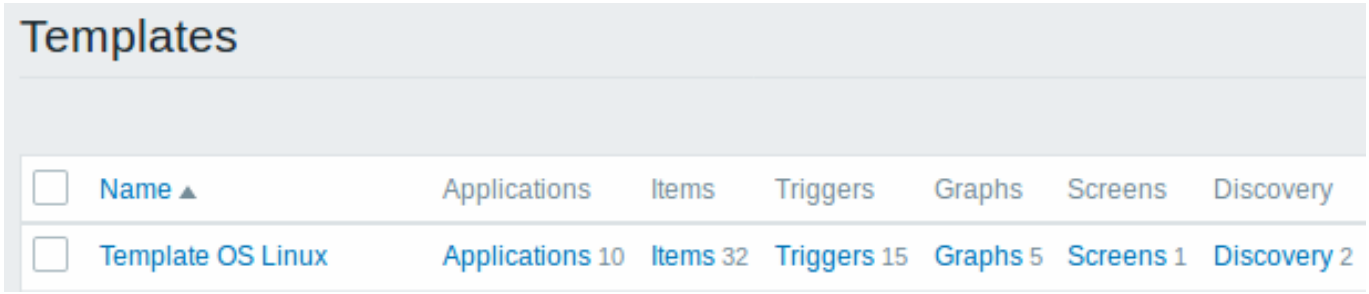
当服务器接收到已发现项的值时，它会查看宏 → 值对，每对都根据原型生成实际监控项，触发器和图形。在上面的 “net.if.discovery” 示例中，服务器将生成环路接口 “lo” 的一组监控项，触发器和图表，另一组用于界面 “eth0”。

配置低级别发现（LLD）

我们来用文件系统发现为例子说明低级别发现（LLD）。

请执行以下操作，配置发现：

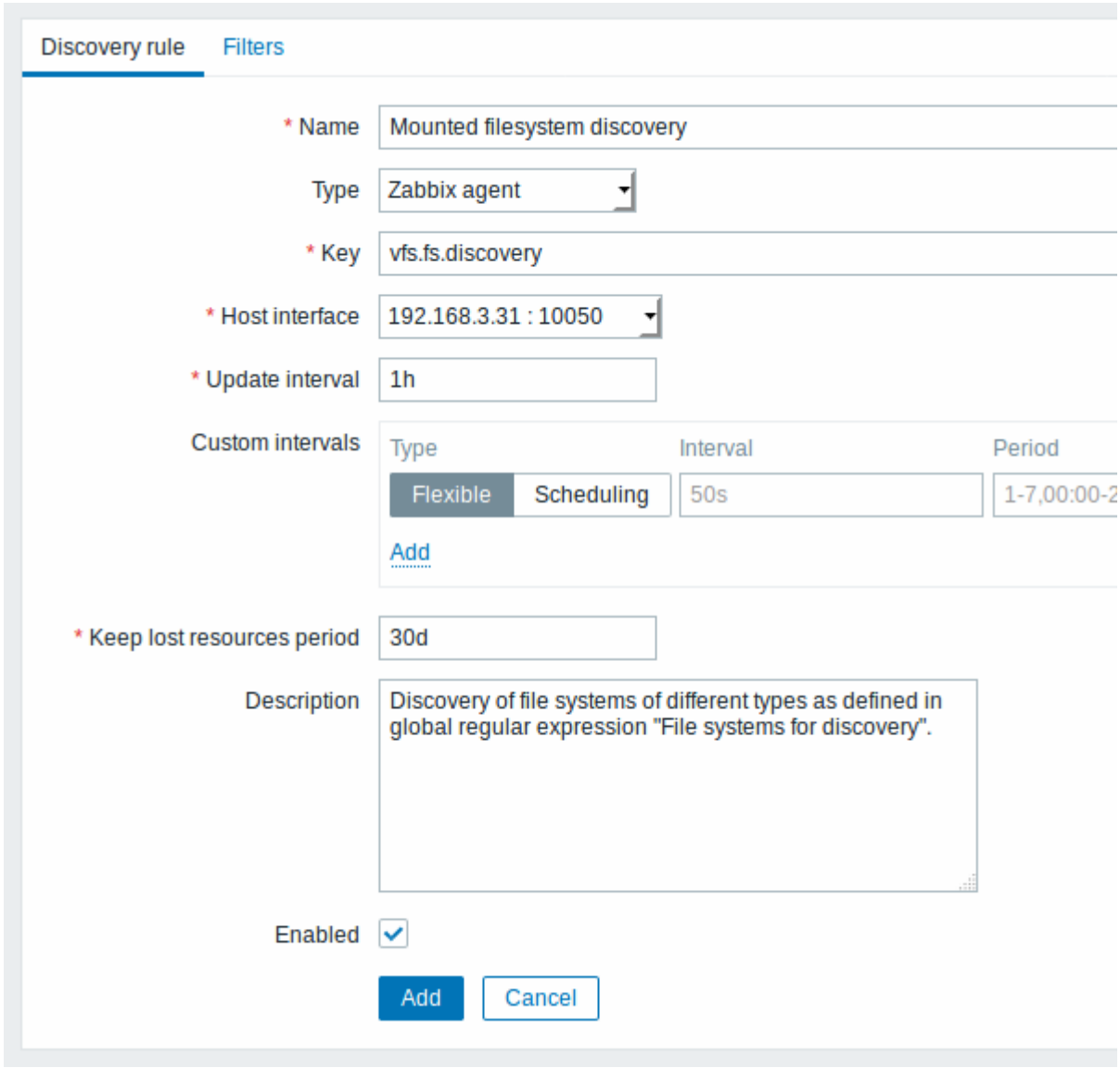
- 进入: 配置 → 模板
- 选择一个合适的模板的行点击发现



- 单击屏幕右上角的 创建发现规则
- 填写需要的详细信息

发现规则

发现规则选项卡包含常规发现规则属性：



所有必填输入字段都标有红色星号。

参数描	
名称发	规则名称。

参数	描
类型执	发现的检查类型;可以是 *Zabbix agent* 或 *Zabbix agent* (主动文件系统发现。

参数	描
键值自	.0 版以来，许多平台上 Zab-bix agent 中都内置了一个带有“vfs.fs.discovery”键的项目,(详情参见支持项目键列表)，将返回一个 JSON，包含计算机上存在的文件系统列表及其类型详情。

参数	描
数据更新间隔 *(秒)* 此字段指定 Z	bbix 执行发现的频率。在开始刚刚设置文件系统发现时，可以将其设置为一个小间隔，但是一旦知道它可以工作之后，可以将其设置为 30 分钟或更长时间，因为文

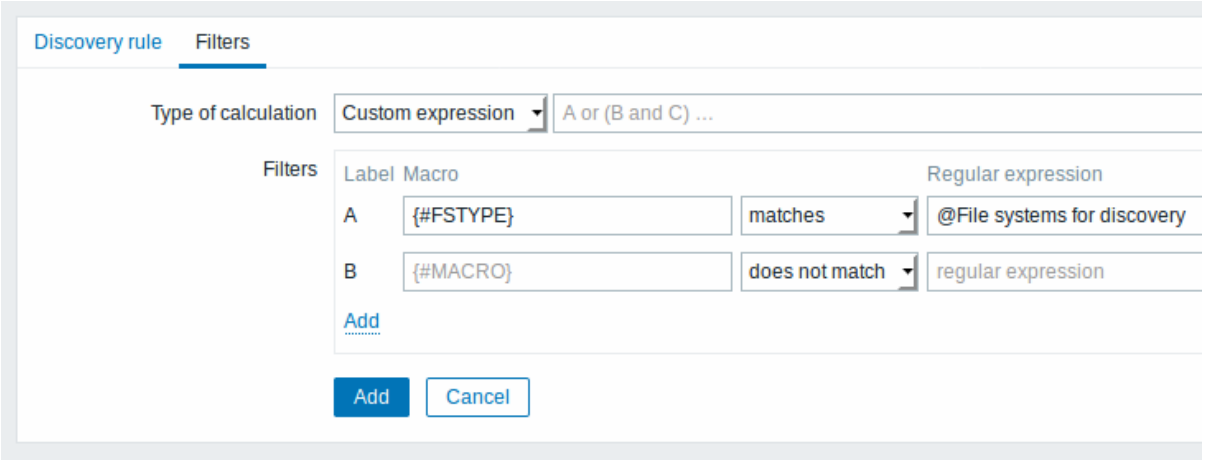
参数	描
隔	可以创建用于检查项目的自定义规则： **灵活** - 创建例外的更新间隔(不同频次的间隔) **调度** - 创建自定义轮询调度。 有关详细信息，请参阅自定义时间间隔. 从Zabix 3.0.0

参数描
资源周期不足 *(天)* 该字段允许你置发现的实体将被发现状态变为“不再支持”(最多3650天)后将被保留(不会被删除)的天数。自Zab-bix 3.4.0起，支持时间后缀例如2h，1d。自Zab-bix 3.4.0起，支持用户宏注意*:*如果设置为

参数描	
Description	输入一段描述。
Enabled	如果选中，表示规则正常启用。

发现规则过滤器

过滤器选项卡包括发现规则过滤器定义:



Parameter	Description
参数描 计算类型可以使	以下计算过滤器的选项： **And** - 所有过滤器必须通过; **Or** - 只需要一个过滤器通过就足够了; **And/Or** - 使用具有不同的宏名称的 *And* ，具有相同宏名称的 *Or* ; **Custom expression** - 提供定义过滤器的自定义计算的可能性。公式必须包含列表中的所有过滤器。限 255 个符号。

Parameter	Description
过滤器过滤	可以用来生成真实监控项，触发器，但是图表仅用于特定的文件系统。 它期待一个 Perl Compatible Regular Expression (PCRE)。例如，如果你只对 C:, D:, 和 E: 文件系统有想法，你可以把 {#FSNAME} 放进”宏” 中并将”^C|^D|^E” 正则表达式放入” 正则表达式” 文本字段。使用 {#FSTYPE} 宏（例如”^ext|^reiserfs”）的文件系统类型和使用 {#FSDRIVETYPE} 宏（例如，“fixed”）的驱动器类型（仅由 Windows agent 支持）也可以进行过滤。 您可以在 “正则表达式” 字段中输入正则表达式或引用全局正则表达式 你可以用”grep -E” 来测试一个正则表达式，例如： `for f in ext2 nfs reiserfs smbfs; do echo $f | grep -E '^ext` 宏从 Zabbix **3.0.0** 在 Windows 上开始支持。 从 Zabbix **2.4.0** 开始支持定义多个过滤器。 注意，如果一些来自过滤器在响应中丢失，那发现的实体将会被忽略。 \\“过滤器” 下拉列表提供两个值，用于指定宏是与正则表达式匹配还是不匹配。

<note important> 如果要正确发现仅按大小写不同的文件系统名称，则必须将 MySQL 中的 Zabbix 数据库创建为区分大小写。:::

<note important> 在正则表达式中的错误或错字，应用在 LLD 规则中时可能会导致删除删除数以千计的配置元素，历史值和许多主机的活动。例如，错误的 “用于发现的文件系统” 正则表达式可能会导致删除数以千计的监控项，触发器，历史值和活动。:::

Note:
不保留发现规则历史记录。

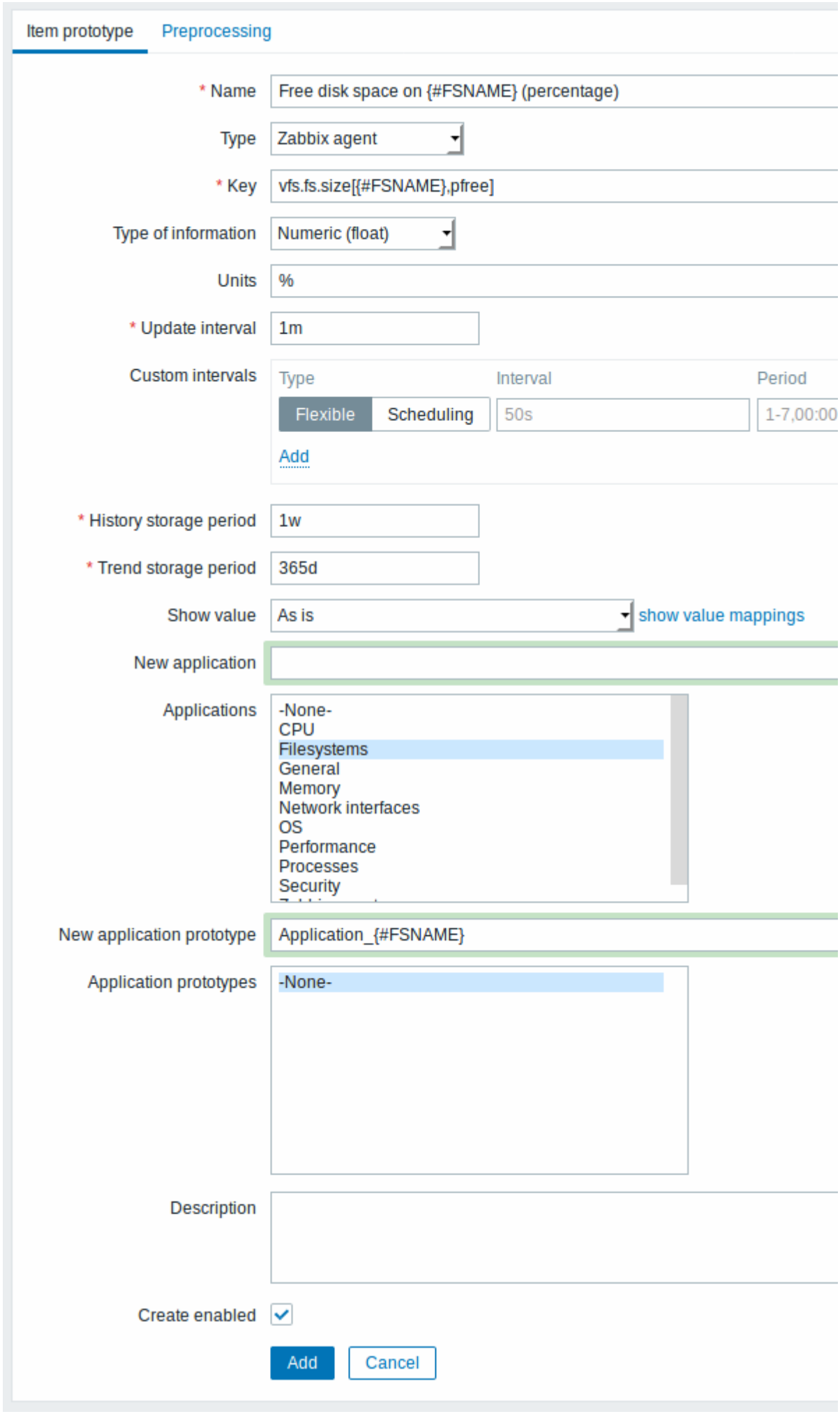
表格按钮

在表格底部的按钮会显示许多可用的操作。

Update	更新发现规则的属性。此按钮仅适用于现有发现规则。
Clone	在现有的发现规则的属性基础上创建另一个发现规则。
Check now	立即根据发现规则执行发现。发现规则必须已存在。查看详情. 注意当立即执行发现时，配置缓存不会更新，因此结果并不代表发现规则配置中最新的改变。
Delete	删除发现规则。
Cancel	取消编辑发现规则属性。

监控项原型

一旦规则创建完成了，找到那条规则下的监控项，并点击 “创建原型” 来创建一个监控项原型。请注意在需要文件系统名称的情况下如何使用宏 {#FSNAME} 。处理发现规则时，此宏将替换为发现的文件系统。


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
1w
* Trend storage period
365d
Show value
As is
show value mappings
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
Add
Cancel

低级别发现宏 and 用户宏 可能会被用在监控项原型配置和监控项值预处理参数.

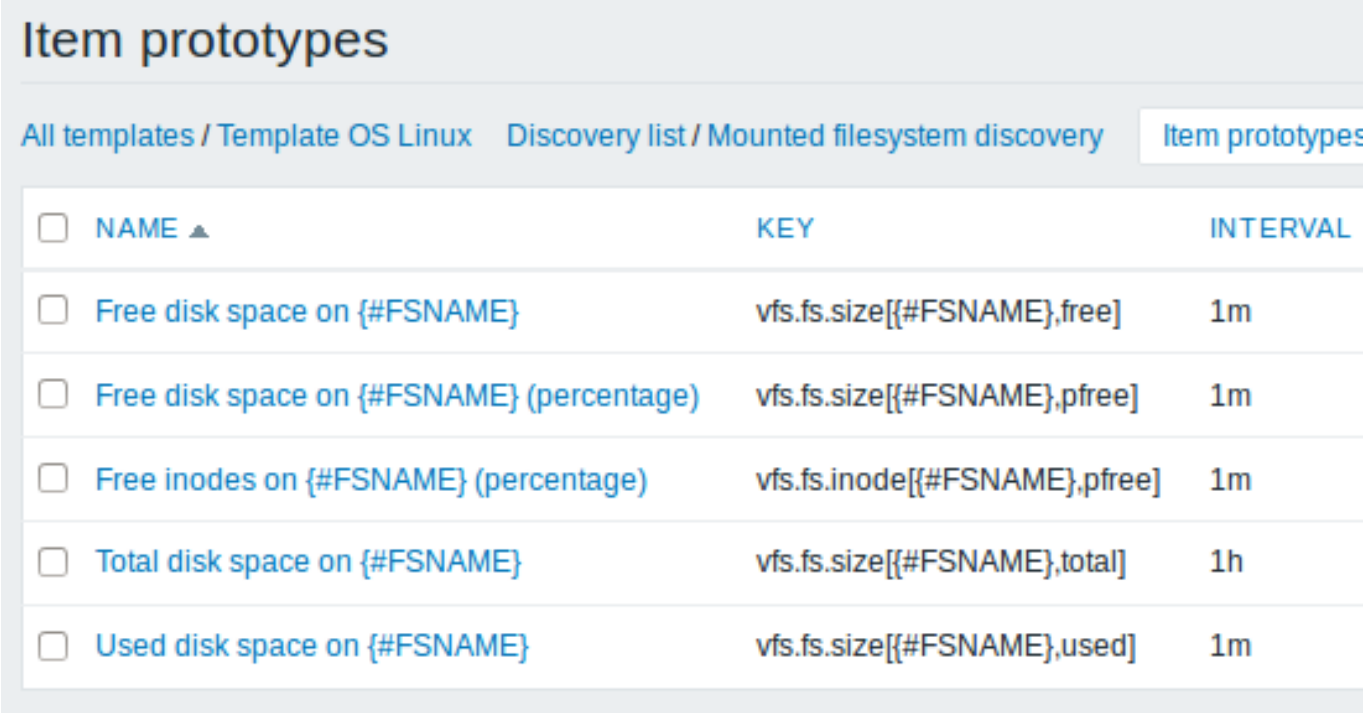
> **Note:**
> 执行特定于上下文的低级别发现宏的转义，以便在正则表达式和 XPath 预处理参数中安全使用。

Attributes that are specific for item prototypes: 特定于监控项原型的属性:

参数	描	
全新的应用原型你可能定义一		新的应用原型。在应用原型中，你可以使用低级别发现宏，在发现后，将替换为实际值以创建特定于已发现实体的应用程序. 在应用发现注意事项 查看更详细的信息。
应用程序原型从现有的应		程序原型选择。

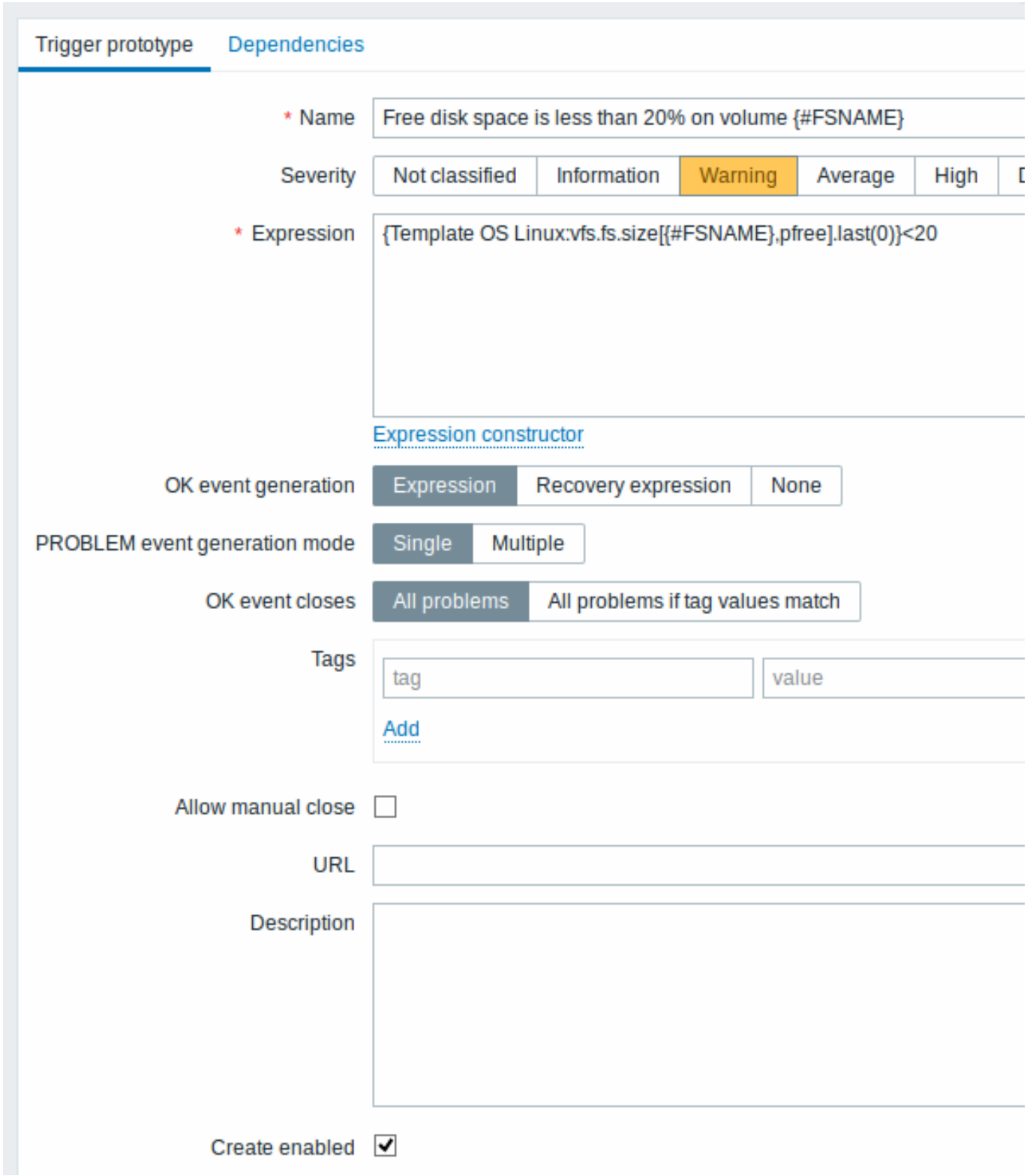
参数描	
创建已启用如果选中	则监控项将以启用状态添加。如果未选中，则该监控项将添加到已发现的实体，但处于禁用状态。

我们可以根据需求为每个文件系统指标创建许多监控项原型:



触发器原型

我们创建触发器原型的方式和创建监控项原型的方式相似：

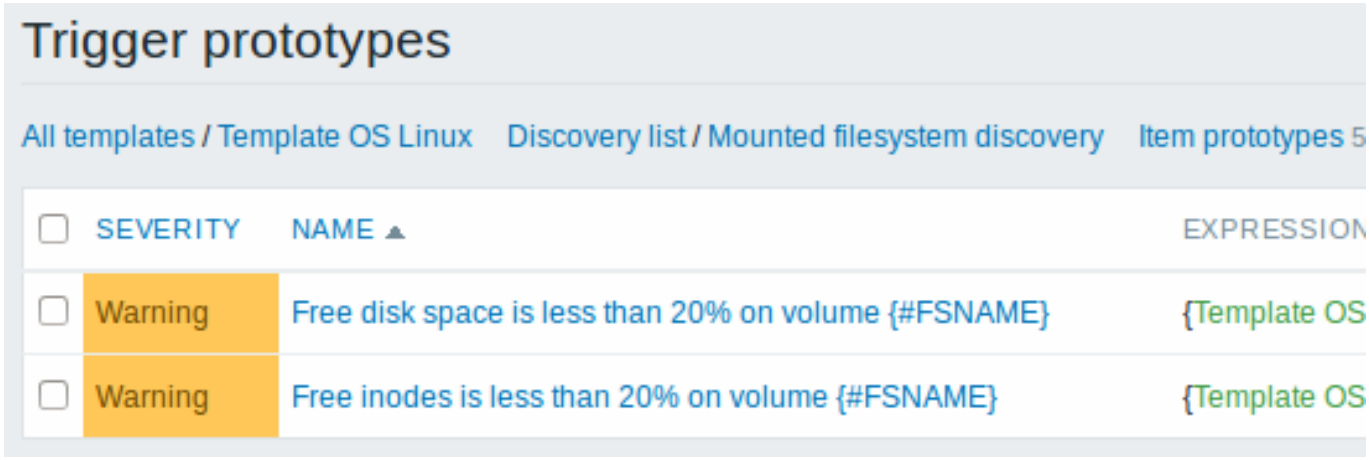


特定于触发器原型的属性：

参数	描
创建已启用	如果选中，则触发器将以启用状态添加。 如果未选中，则该触发器将添加到已发现的实体，但处于禁用状态。

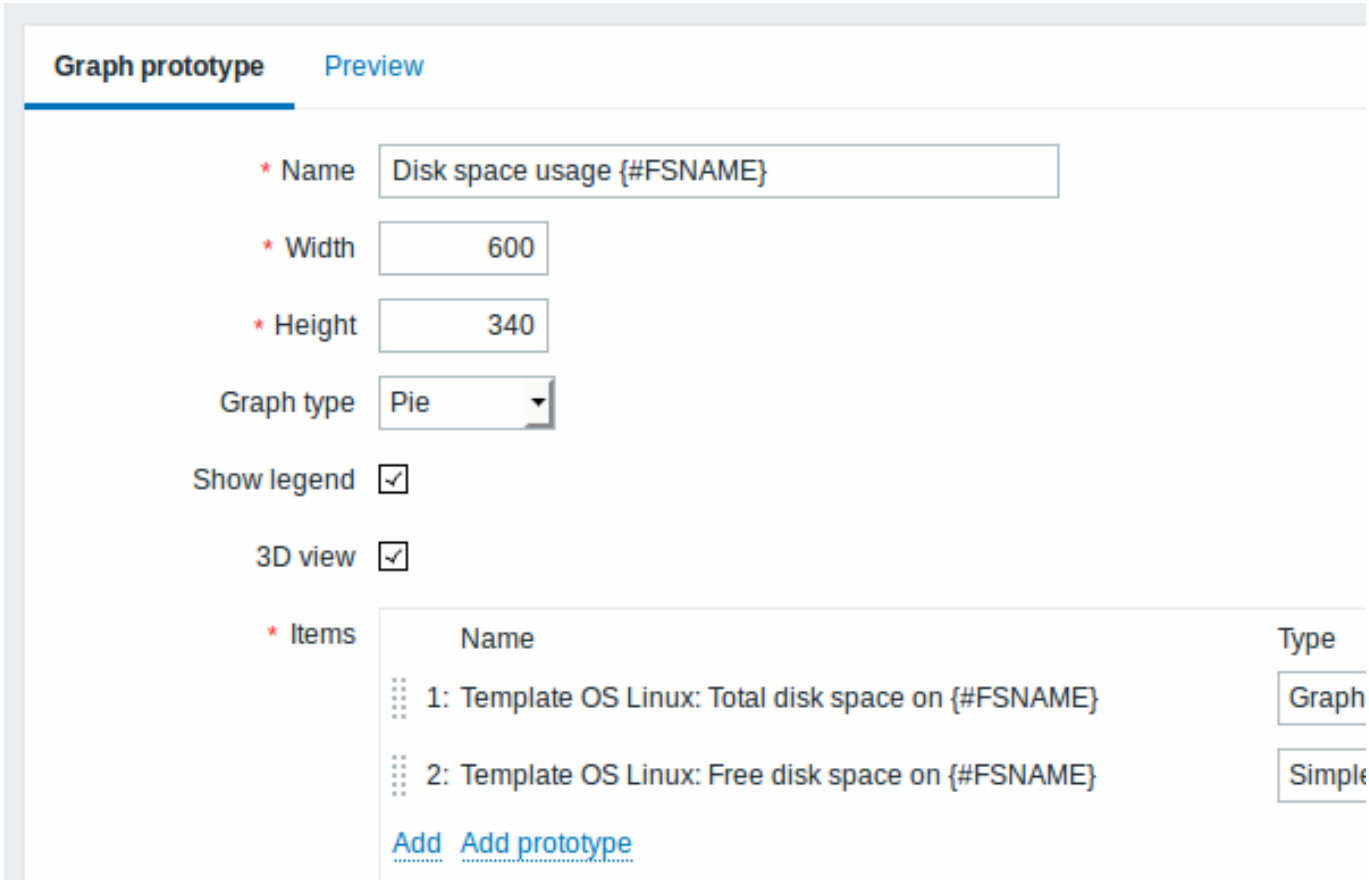
当真实的触发器从原型创建，可能需要灵活地确定在表达式中用于比较的常量（在我们的示例中为'20'）。了解基于环境的用户宏 如何有助于实现这种灵活性。

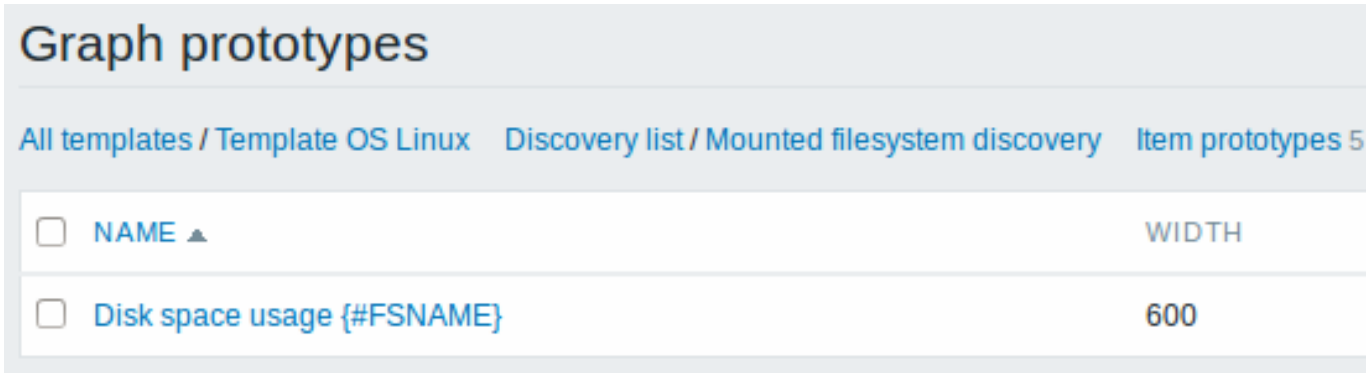
您也可以定义触发器原型之间的（从 Zabbix 3.0 开始支持）。You can define 依赖关系（从 Zabbix 3.0 开始支持）。为此，转到 依赖关系 选项卡。触发器原型可能依赖于来自相同低级别发现（LLD）规则的另一个触发器原型或常规触发器。触发器原型可能不依赖于来自不同 LLD 规则的触发器原型或依赖于触发器原型创建的触发器。主机触发器原型不能依赖于来自模板的触发器。



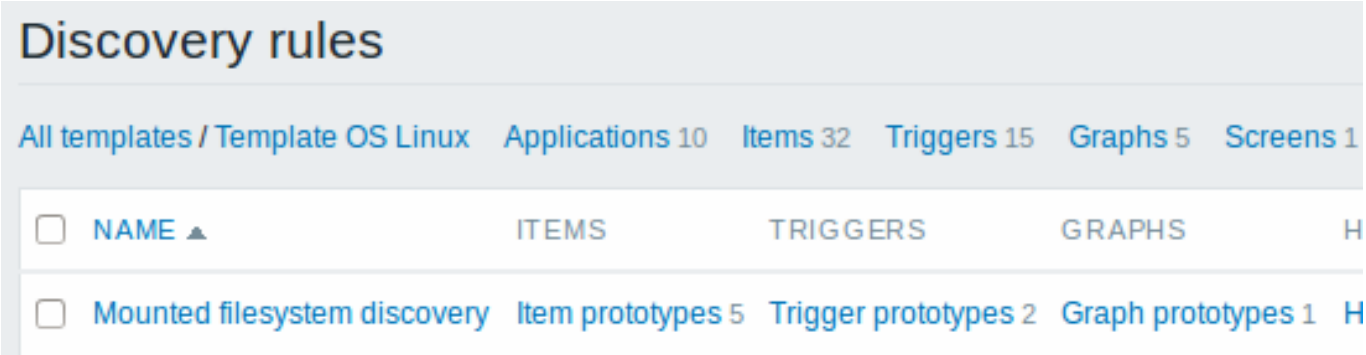
图表原型

我们也能创建图表原型:





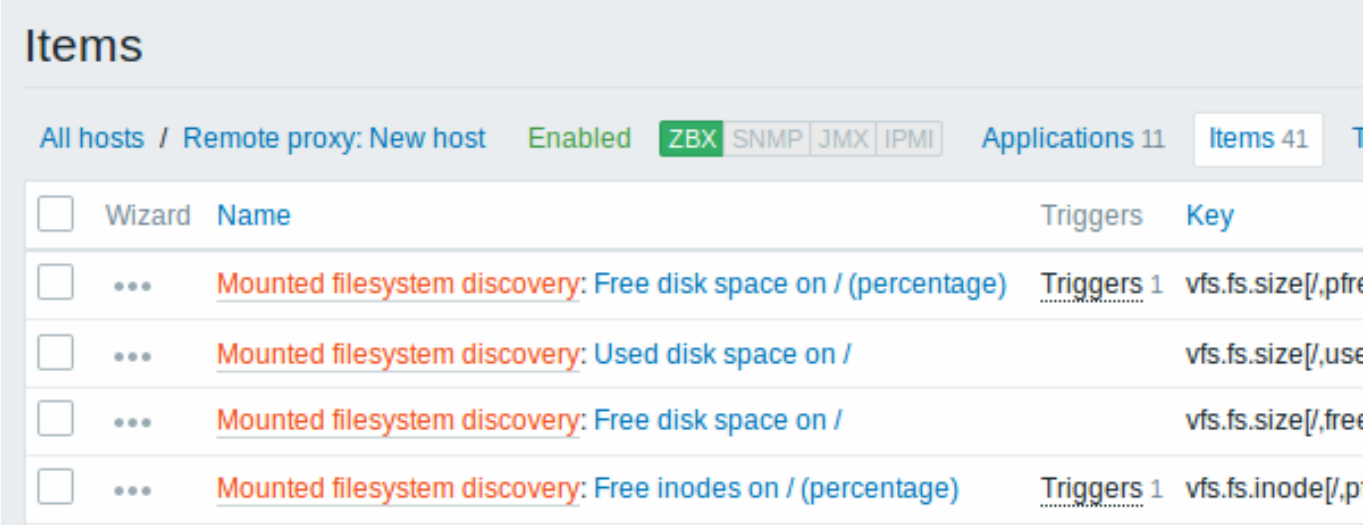
最后，我们像下面的展示的一样创建一个发现规则。有五个监控项原型，两个触发器原型，和一个图表原型。



注意: 有关配置主机原型的信息，请参阅虚拟机监视中有关主机原型配置的部分。

发现的实体

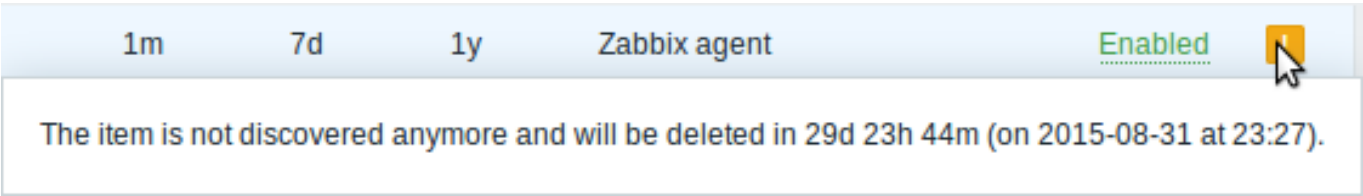
下面的屏幕截图说明了主机配置中发现的项目，触发器和图形的外观。发现的实体的前缀是橙色链接，指向它们来自的发现规则。



请注意，如果已存在具有相同唯一性条件的实体，则不会创建已发现的实体，例如，具有相同 key 或具有相同名称的图形的监控项。

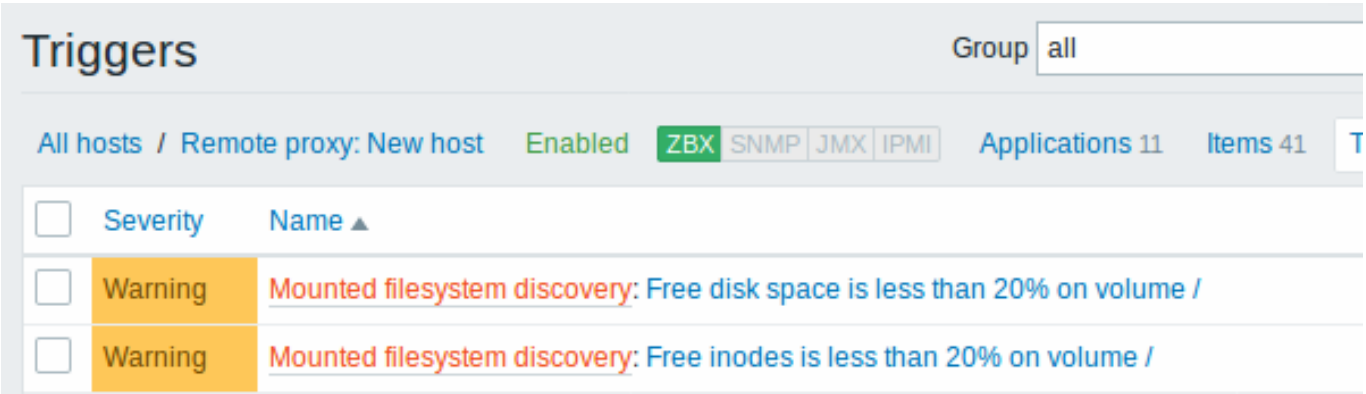
如果一个发现的实体（文件系统，接口等）停止被发现（或是再也不通过过滤器），使用低级别发现规则创建的监控项（触发器和图表也相似）将会自动被删除。在这种情况下，在 Keep lost resources period 字段传递中定义的日期之后，将删除监控项，触发器和图表。

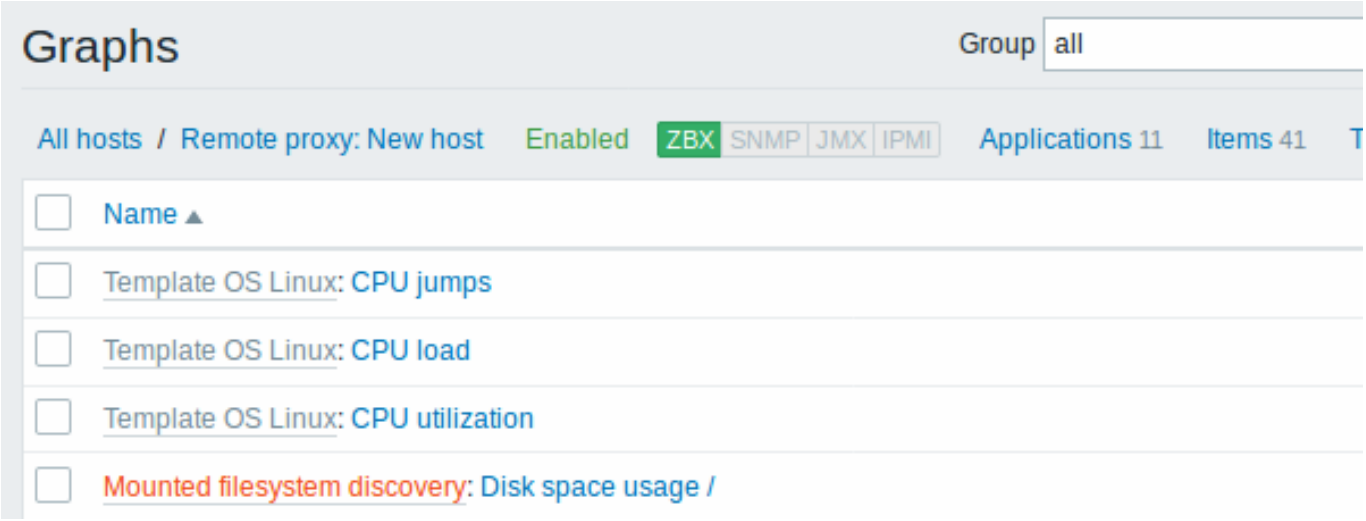
当发现实体转变成 “不再发现” 状态，生命周期指示符显示在监控项列表中。将鼠标指针移到它上面，将显示一条消息，表示在删除项目之前剩余的天数。



如果实体已标记为删除但未在预期时间删除（已禁用的发现规则或项目主机），则下次处理发现规则时将删除这些实体。

如果在发现规则级别更改，则包含标记为删除的其他实体的实体将不会更新。例如，如果基于 LLD 的触发器包含标记为删除的监控项，则不会更新。





其他类型的发现

以下部分提供了有关其他类型的开箱即用发现的更多详细信息和方法：

- 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.
- 网络接口发现;
- CPUs 和 CPU 核心发现;
- SNMP OIDs发现;
- JMX 对象发现;
- 使用ODBC SQL 查询发现;
- Windows services发现;
- 在 Zabbix 中的主机接口发现

有关发现项的 JSON 格式的更多详细信息以及如何将自己的文件系统发现者实现为 Perl 脚本的示例，请参阅创建自定义 LLD 规则。

反馈值的数据限制

如果是直接来自 Zabbix server，那对低级别发现规则 JSON 数据没有限制，因为反馈值是在没有存储在数据库中的情况下处理的。对于定制的低级别发现规则也没有限制，但是，如果如果要使用用户参数获取自定义 LLD 数据，则应用用户参数反馈值会有限制（512 KB）。

如果数据必须通过 Zabbix proxy，则必须将此数据存储在数据库中，以便应用数据库限制，例如，在运行 IBM DB2 数据库的 Zabbix proxy 上应用 2048 字节。

同一监控项的多个 LLD 规则

自从 Zabbix agent 3.2 版本开始，就可以使用相同的发现监控项定义许多低级别发现规则了。

为此，你需要定义 Alias agent 参数，在不同的发现规则中允许使用更改发现监控项密钥。例如，`vfs.fs.discovery[foo]`，`vfs.fs.discovery[bar]` 等。

创建自定义 LLD 规则

也可以创建一个完整的自定义 LLD 规则，同时发现任何类型的实体——例如，在 database server 上的数据库。

为此，应创建一个反馈 JSON 的自定义项，指定找到的对象并可选——他们的一些属性。每个实体宏不受限制——虽然内置的发现规则反馈一个或两个宏（例如，两个用于文件系统发现），单反馈更多宏也是可能的。

这里有个例子可以最好地证明需要的 JSON 格式。假设我们正在运行一个接的 Zabbix 1.8 agent（不支持"vfs.fs.discovery"），但我们仍旧需要发现文件系统。这有一个简单的 Linux Perl 脚本，可以发现挂载的文件系统，并输出 JSON，其中包括文件系统的名称和类型。一种使用它的方式是使用键"vfs.fs.discovery_perl" 作为 UserParameter：

```
###!/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:
LLD 宏名称的允许符号为 **0-9**，**A-Z**，_，**.**

名称中不支持小写字母。

其输出示例（为清晰起见重新格式化）如下所示。自定义发现检查的 JSON 必须遵循相同的格式。

```
{
  "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"  }

  ]
}
```

然后，在发现规则中的 “过滤器” 部分，我们可以指定”{#FSTYPE}” 作为一个宏，”rootfs|ext3” 作为一个正则表达式。

Note:
你不必要使用自定义 LLD 规则的宏名称 FSNAME/FSTYPE ，你可以使用任何你喜欢的名称。

需要注意的是，如果使用一个用户参数，反馈值限制到 512 KB。更多信息，请参考LLD 反馈值的数据限制.

在用户宏环境中使用 LLD 宏

环境中的用户宏可用于在触发器表达式中实现更灵活的阈值。不同的阈值可以在用户宏级别上定义不同的阈值，然后根据发现的环境将其用于触发器常量。当宏中使用的低级别发现宏 被解析为实际值时，将显示已发现的环境。

为了证明我们可以使用上述的例子中的数据，假设将发现以下文件系统：/, /home, /tmp, /usr, /var.

我们可以为主机定义一个自由磁盘空间触发器原型，其中阈值由具有发现环境的用户宏表示：

```
{host:vfs.fs.size[{#FSNAME},pfree].last()}<{$LOW_SPACE_LIMIT:"{#FSNAME}"}
```

然后新增用户宏:

- {$LOW_SPACE_LIMIT} **10**
- {$LOW_SPACE_LIMIT:/home} **20**
- {$LOW_SPACE_LIMIT:/tmp} **50**

现在，一旦文件系统被发现了，如果/, /usr and /var 文件系统有少于 **10**% 的自由磁盘空间，/home 文件系统——少于 **20**% 的自由磁盘空间或/tmp 文件系统——少于 **50**% 的自由磁盘空间，将生成事件。

Multiple LLD rules for same item

Since Zabbix agent version 3.2 it is possible to define several low-level discovery rules with the same discovery item.

To do that you need to define the Alias agent parameter, allowing to use altered discovery item keys in different discovery rules, for example `vfs.fs.discovery[foo]`, `vfs.fs.discovery[bar]`, etc.

Creating custom LLD rules

It is also possible to create a completely custom LLD rule, discovering any type of entities - for example, databases on a database server.

To do so, a custom item should be created that returns JSON, specifying found objects and optionally - some properties of them. The amount of macros per entity is not limited - while the built-in discovery rules return either one or two macros (for example, two for filesystem discovery), it is possible to return more.

The required JSON format is best illustrated with an example. Suppose we are running an old Zabbix 1.8 agent (one that does not support "vfs.fs.discovery"), but we still need to discover file systems. Here is a simple Perl script for Linux that discovers mounted file systems and outputs JSON, which includes both file system name and type. One way to use it would be as a UserParameter with key "vfs.fs.discovery_perl":

```
###!/usr/bin/perl

$first = 1;

print "[\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"{#FSNAME}\":\"$fsname\",\n";
    print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

Attention:
Allowed symbols for LLD macro names are **0-9** , **A-Z** , _ , **.**

Lowercase letters are not supported in the names.

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
[
    { "{#FSNAME}":"/",                          "{#FSTYPE}":"rootfs"   },
    { "{#FSNAME}":"/sys",                       "{#FSTYPE}":"sysfs"    },
    { "{#FSNAME}":"/proc",                      "{#FSTYPE}":"proc"     },
    { "{#FSNAME}":"/dev",                       "{#FSTYPE}":"devtmpfs" },
    { "{#FSNAME}":"/dev/pts",                   "{#FSTYPE}":"devpts"   },
    { "{#FSNAME}":"/lib/init/rw",               "{#FSTYPE}":"tmpfs"    },
    { "{#FSNAME}":"/dev/shm",                   "{#FSTYPE}":"tmpfs"    },
    { "{#FSNAME}":"/home",                      "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/tmp",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/usr",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/var",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/sys/fs/fuse/connections",   "{#FSTYPE}":"fusectl"  }
]
```

In previous example it is required that the keys match the LLD macro names used in prototypes, the alternative is to extract LLD macro values using JSONPath {#FSNAME} → $.fsname and {#FSTYPE} → $.fstype, thus making such script possible:

```
###!/usr/bin/perl

$first = 1;

print "[\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"fsname\":\"$fsname\",\n";
    print "\t\t\"fstype\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
[
    { "fsname":"/",                          "fstype":"rootfs"   },
    { "fsname":"/sys",                       "fstype":"sysfs"    },
    { "fsname":"/proc",                      "fstype":"proc"     },
    { "fsname":"/dev",                       "fstype":"devtmpfs" },
    { "fsname":"/dev/pts",                   "fstype":"devpts"   },
    { "fsname":"/lib/init/rw",               "fstype":"tmpfs"    },
    { "fsname":"/dev/shm",                   "fstype":"tmpfs"    },
    { "fsname":"/home",                      "fstype":"ext3"     },
    { "fsname":"/tmp",                       "fstype":"ext3"     },
    { "fsname":"/usr",                       "fstype":"ext3"     },
    { "fsname":"/var",                       "fstype":"ext3"     },
    { "fsname":"/sys/fs/fuse/connections",   "fstype":"fusectl"  }
]
```

Then, in the discovery rule's "Filter" field, we could specify "{#FSTYPE}" as a macro and "rootfs|ext3" as a regular expression.

> **Note:**
> You don't have to use macro names FSNAME/FSTYPE with custom LLD rules, you are free to use whatever names you like. In case JSONPath is used then LLD row will be an array element that can be an object, but it can be also another array or a value.

Note that, if using a user parameter, the return value is limited to 512 KB. For more details, see data limits for LLD return values.

Using LLD macros in user macro contexts

LLD macros may be used inside user macro context, for example, in trigger prototypes.

1 发现已挂载的文件系统

概述

可以发现已挂载的文件系统及其属性 (挂载点名称、挂载点类型、文件系统大小和索引节点统计信息).

要做到这一点, 你可以结合使用:

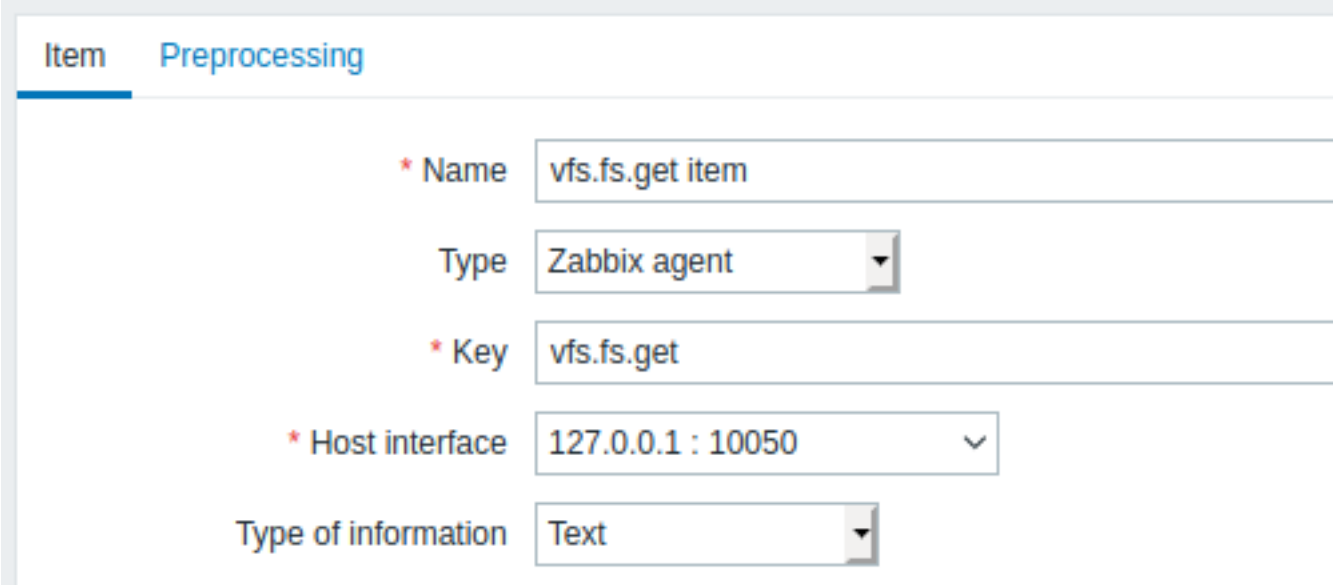
- 在 master 上使用 agent 的 `vfs.fs.get` 监控项
- 依赖低级别发现规则和监控项原型

配置

服务端的监控项

创建一个 Zabbix Agent 的监控项可以使用以下 key:

`vfs.fs.get`



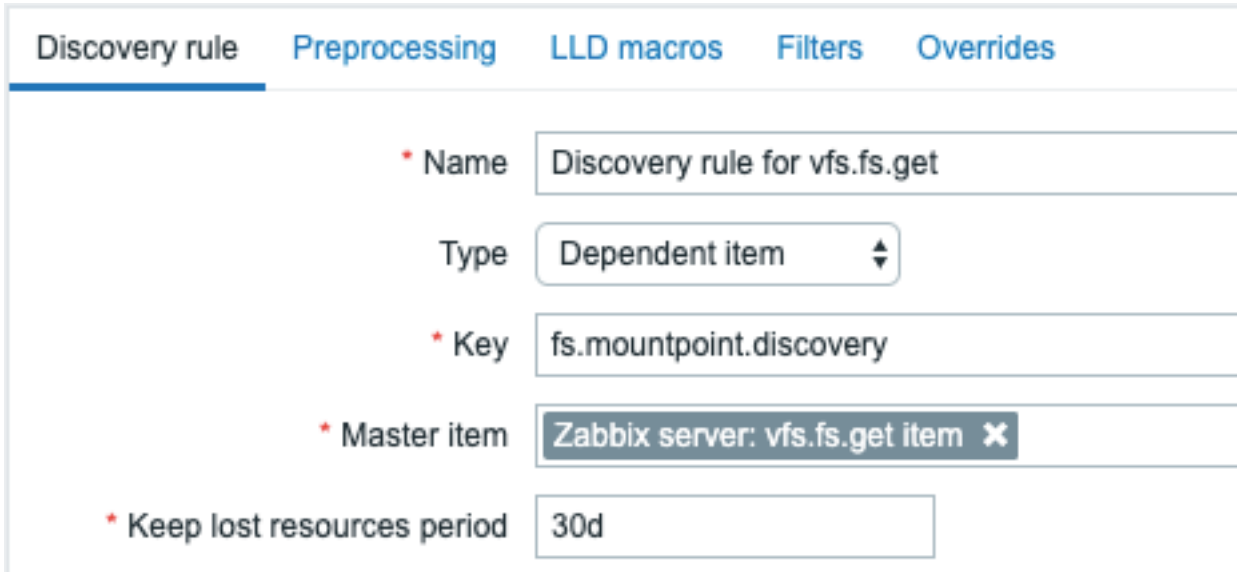
可能较大的 JSON 数据，设置信息类型为”Text”.

已挂载的文件系统，此项返回的数据将包含如下内容:

```
{
  "fsname": "/",
  "fstype": "rootfs",
  "bytes": {
    "total": 1000,
    "free": 500,
    "used": 500,
    "pfree": 50.00,
    "pused": 50.00
  },
  "inodes": {
    "total": 1000,
    "free": 500,
    "used": 500,
    "pfree": 50.00,
    "pused": 50.00
  }
}
```

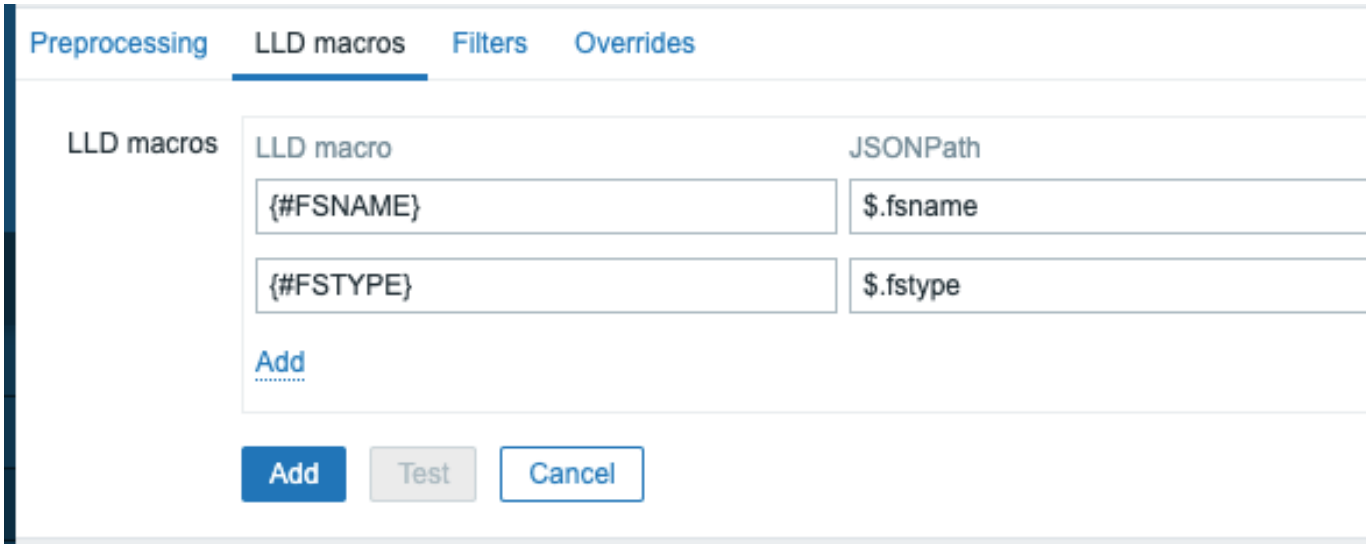
依赖 LLD 规则

创建一个低级别发现规则作为 “依赖项” 类型:



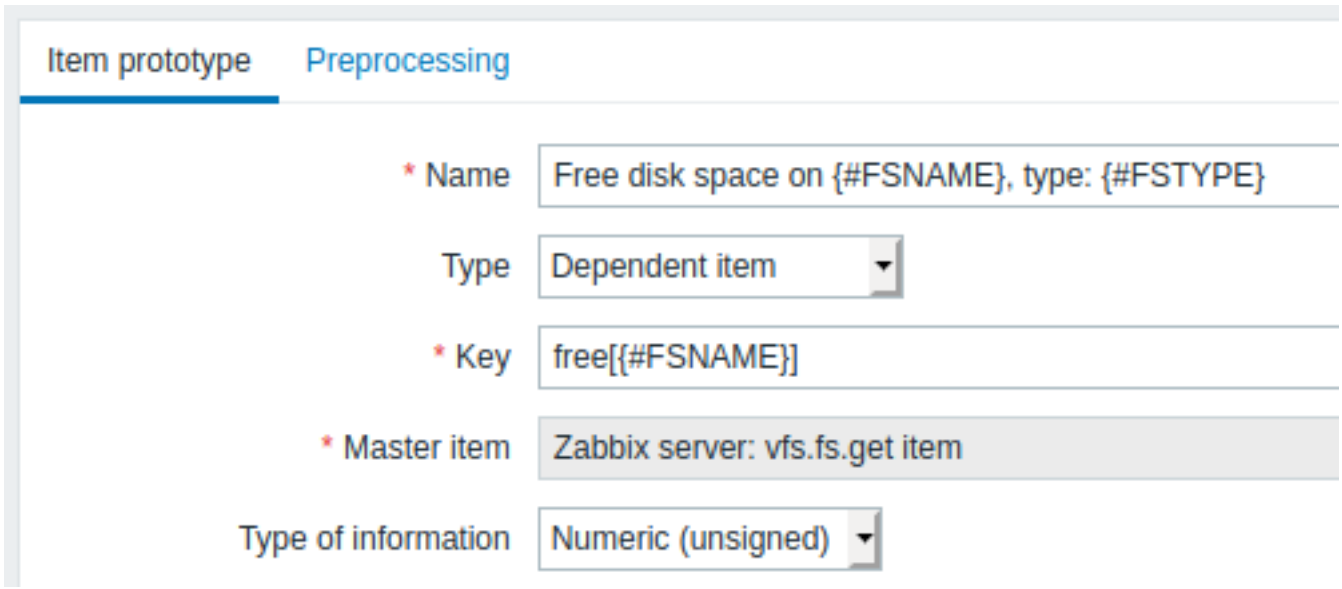
主要项选择 `vfs.fs.get` 作为创建的监控项

在"LLD 宏" 选项中，用相应的 JSONPath 定义自定义宏:



依赖监控项原型

在自动发现规则中创建监控项原型，类型选择" 依赖项". 主要项选择我们创建的 `vfs.fs.get`.



注意在监控项原型名称和键值中使用了自定义宏:

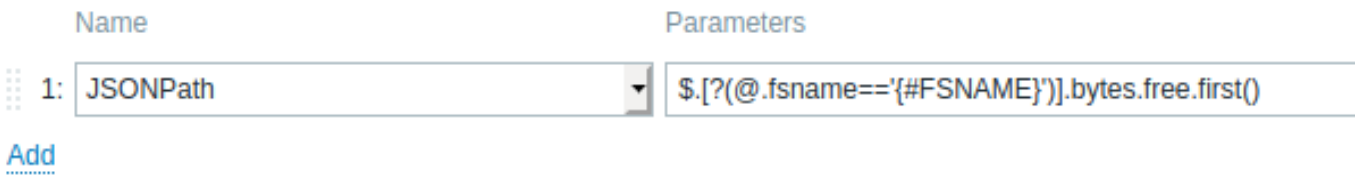
- 名称: Free disk space on {#FSNAME}, type: {#FSTYPE}
- 键值: Free[{#FSNAME}]

信息类型的使用:

- 数字 (正负数) 可用于这类指标'free', 'total', 'used'
- 浮点数可用于这类指标'pfree', 'pused' (percentage)

在监控项原型 "预处理" 选项卡中选择 JSONPath 并使用以下 JSONPath 表达式作为参数:

```
$.[?(@.fsname=='{#FSNAME}')].bytes.free.first()
```



当自动发现启动，将为每个挂载点创建一个项目。该项将返回给定挂载点的空闲字节数.

2 发现网络接口

与发现文件系统 的方式相似，如此也可以发现网络接口。

键值

在发现规则 中的键值应用是：

```
net.if.discovery
```

此监控项从 Zabbix 2.0 开始支持。

支持宏

你可以在过滤器，监控项，触发器 和图表的原型的发现规则中使用 {#IFNAME} 宏。

举例：你想在"net.if.discovery" 的基础上创建监控项原型。

- "net.if.in[{#IFNAME},bytes]",
- "net.if.out[{#IFNAME},bytes]".

3 发现 **CPU** 和 **CPU** 核心

与发现文件系统的方式相似，如此也可以发现 CPUs 和 CPU 核心。

键值

在发现规则中的键值应用是：

```
system.cpu.discovery
```

此监控项从 Zabbix 2.4 开始支持。

支持宏

此发现键值反馈两个宏——{#CPU.NUMBER} 和 {#CPU.STATUS}，分别识别 CPU 编号和状态。请注意，在实际的，物理的处理器，内核和超线程之间无法做出明确的区分。Linux，UNIX 和 BSD 系统上的 {#CPU.STATUS} 可以反馈处理器的状态，"在线状态" 或 "离线状态"。在 Windows 系统中，这个相同的宏可能代表第三个值——"未知状态"——代表已检测到处理器，但尚未收集任何信息。

CPU 发现依赖于代理的收集器进程来保持与收集器提供的数据一致，并节省获取数据的资源。这样会产生有此键值无法使用代理二进制文件的 test（-t）命令行标志的效果，从而反馈一个 NOT_SUPPORTED 状态以及一条伴随的信息表示收集器进程尚未启动。

可以基于 CPU 发现创建监控项原型包括，例如：

- "system.cpu.util[{#CPU.NUMBER}, <type>, <mode>]"
- "system.hw.cpu[{#CPU.NUMBER}, <info>]".

4 发现 **SNMP OIDs**

概述

在这个部分，我们将会在交换机上展示discoverySNMP。

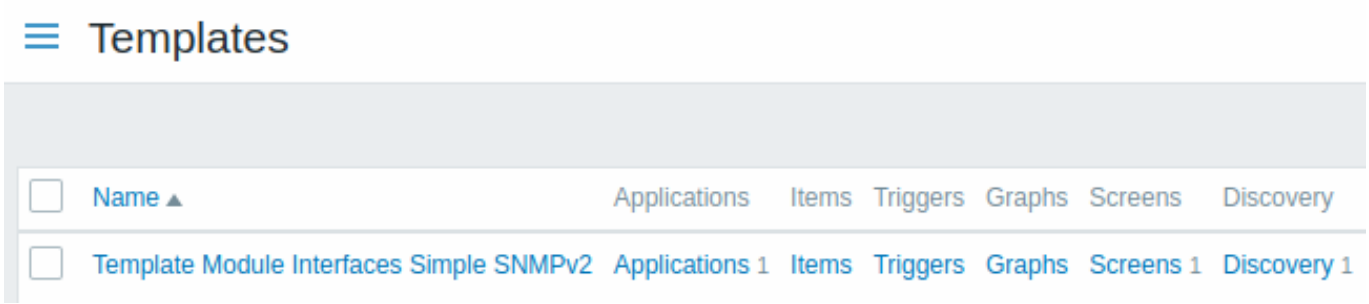
键值

和文件系统和网络接口发现不同，此监控项无需有”snmp.discovery” 密钥 - 监控项类型的 SNMP agent 就足够了。

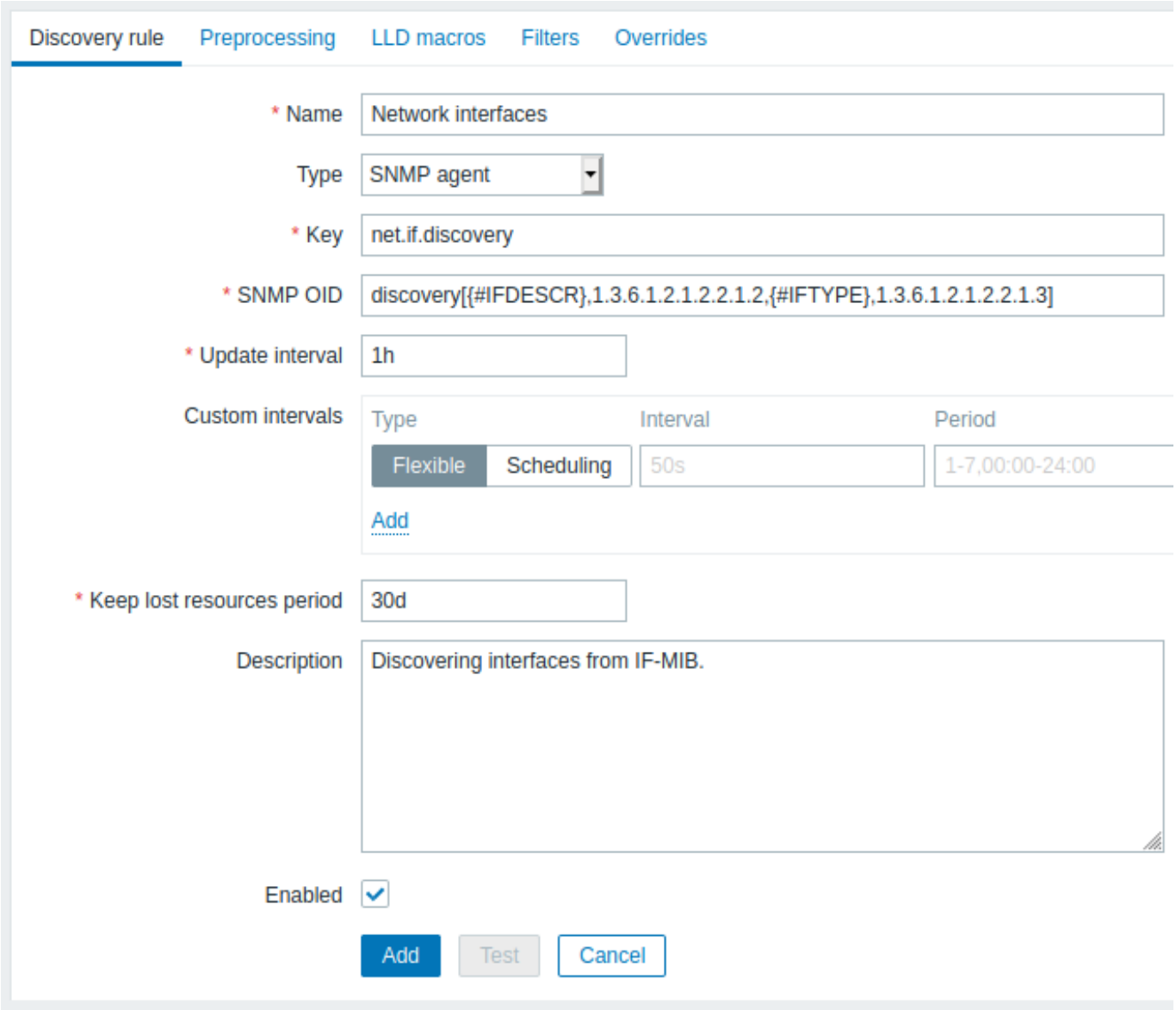
从 Zabbix server/proxy 2.0 开始支持发现 SNMP OIDs。

根据以下操作来设置发现规则:

- 前往: 配置 → 模板
- 点击相应模板中的发现



- 点击屏幕右上角的 创建发现规则
- 填写发现规则表单，如下面的屏幕截图所示



所有必填输入字段都标有红色星号。

要发现的 OID 在 SNMP OID 字段中以以下格式定义: `discovery[{#MACRO1}, oid1, {#MACRO2}, oid2, …,]`

其中 *{#MACRO1}*, *{#MACRO2}* ... 是有效的 LLD 宏名称，*oid1*, *oid2*... 是能够为这些宏生成有意义值的 OID。包含已发现 OID 索引的内置宏 *{#SNMPINDEX}* 将应用于已发现的实体。发现的实体按 *{#SNMPINDEX}* 宏的值分组。

为了理解我们的意思，让我们在我们的交换机上展示一些 snmpwalks：

```
$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: WAN
IF-MIB::ifDescr.2 = STRING: LAN1
IF-MIB::ifDescr.3 = STRING: LAN2

$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifPhysAddress
IF-MIB::ifPhysAddress.1 = STRING: 8:0:27:90:7a:75
IF-MIB::ifPhysAddress.2 = STRING: 8:0:27:90:7a:76
IF-MIB::ifPhysAddress.3 = STRING: 8:0:27:2b:af:9e
```

然后设置 SNMP OID 到: `discovery[{#IFDESCR}, ifDescr, {#IFPHYSADDRESS}, ifPhysAddress]`

现在，此规则将发现将 {#IFDESCR} 宏设置为 **WAN**, **LAN1** and **LAN2**, {#IFPHYSADDRESS} 宏设置为 **8:0:27:90:7a:75**, **8:0:27:90:7a:76**, 和 **8:0:27:2b:af:9e**, {#SNMPINDEX} 宏设置为发现的 OID 索引 **1**, **2** and **3**:

```
{
    "data": [
        {
            "{#SNMPINDEX}": "1",
            "{#IFDESCR}": "WAN",
```

```
            "{#IFPHYSADDRESS}": "8:0:27:90:7a:75"
        },
        {
            "{#SNMPINDEX}": "2",
            "{#IFDESCR}": "LAN1",
            "{#IFPHYSADDRESS}": "8:0:27:90:7a:76"
        },
        {
            "{#SNMPINDEX}": "3",
            "{#IFDESCR}": "LAN2",
            "{#IFPHYSADDRESS}": "8:0:27:2b:af:9e"
        }
    ]
}
```

如果一个实体没有一个具体的 OID，则该实体将省略相应的宏。例如我们有以下数据：

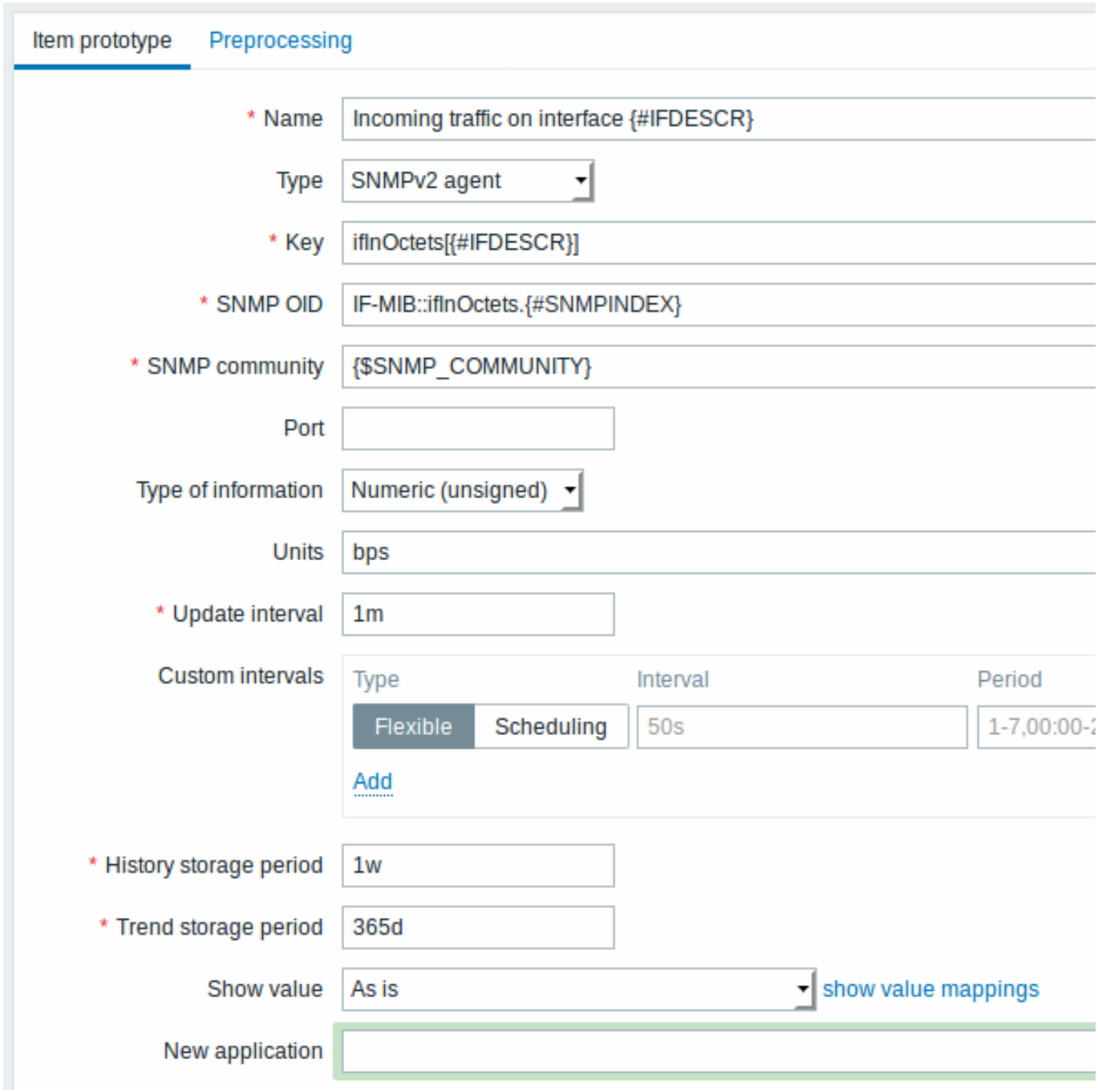
```
ifDescr.1 "Interface #1"
ifDescr.2 "Interface #2"
ifDescr.4 "Interface #4"

ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"
```

然后在在 SNMP 发现 `discovery[{#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias]` 将会反馈以下结构：

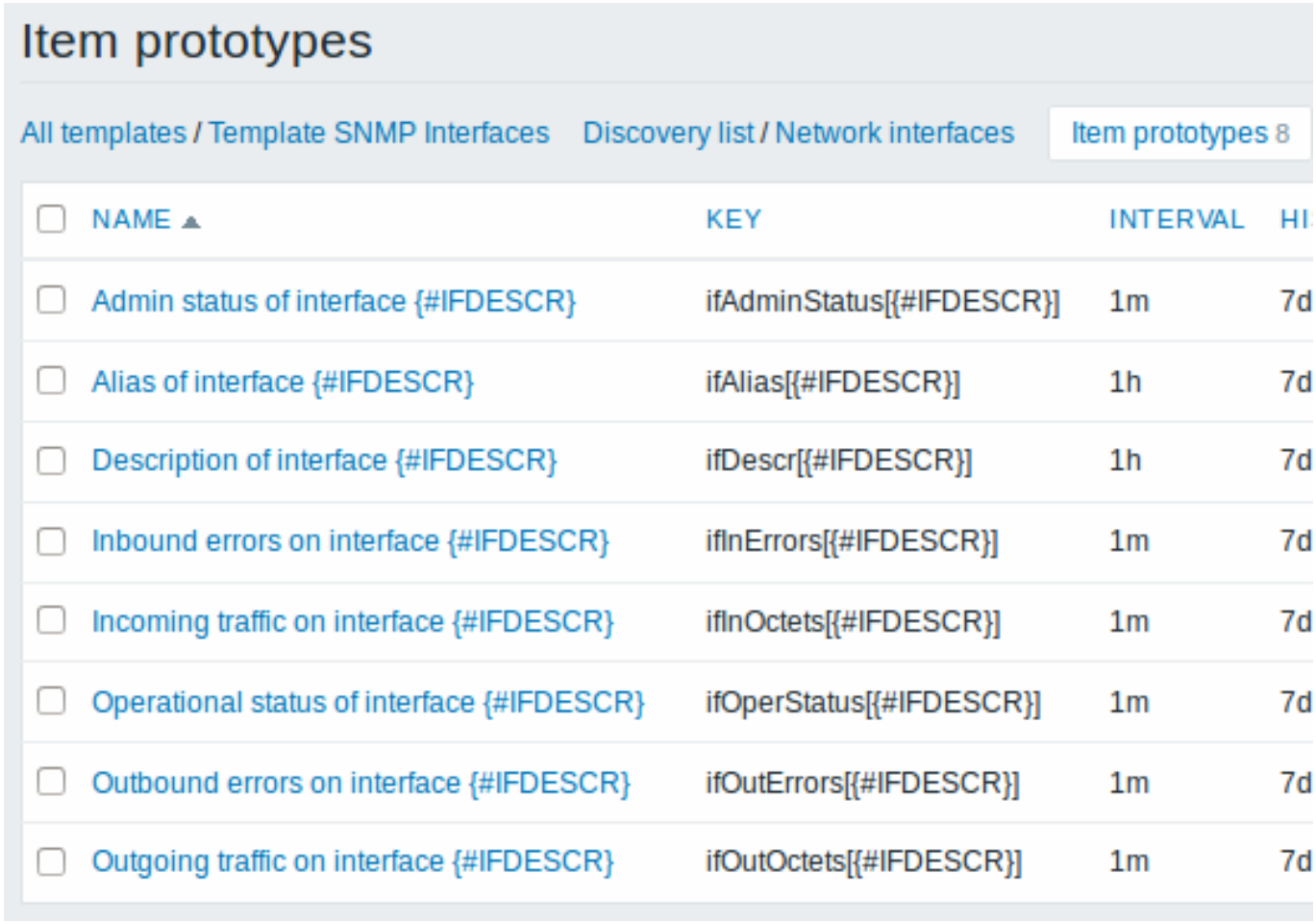
```
{
    "data": [
        {
            "{#SNMPINDEX}": 1,
            "{#IFDESCR}": "Interface #1",
            "{#IFALIAS}": "eth0"
        },
        {
            "{#SNMPINDEX}": 2,
            "{#IFDESCR}": "Interface #2",
            "{#IFALIAS}": "eth1"
        },
        {
            "{#SNMPINDEX}": 3,
            "{#IFALIAS}": "eth2"
        },
        {
            "{#SNMPINDEX}": 4,
            "{#IFDESCR}": "Interface #4"
        },
        {
            "{#SNMPINDEX}": 5,
            "{#IFALIAS}": "eth4"
        }
    ]
}
```

监控项原型

以下截屏说明了我们如何在监控项原型中使用这些宏：

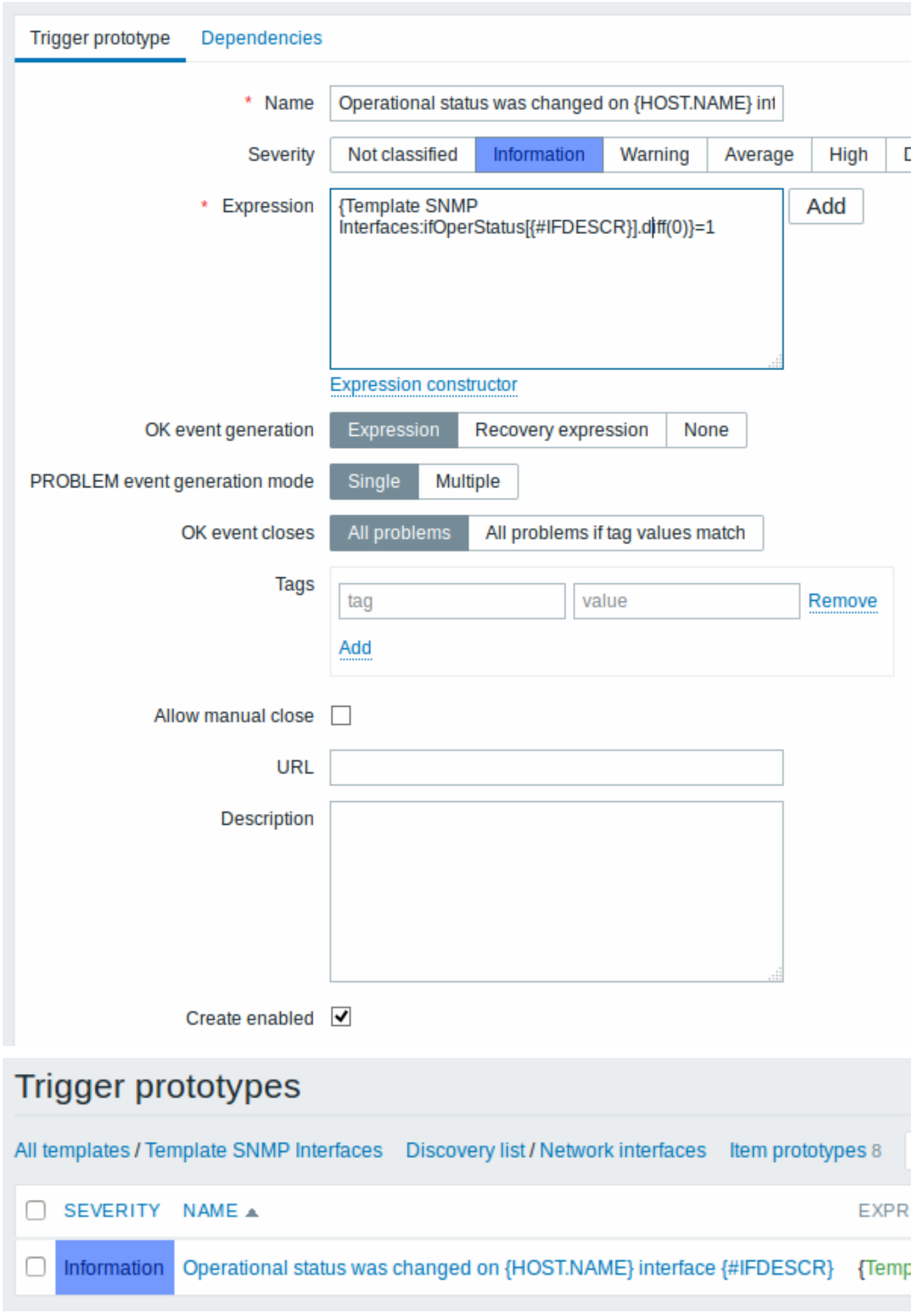


重复一下，根据需求数量创建监控项原型：



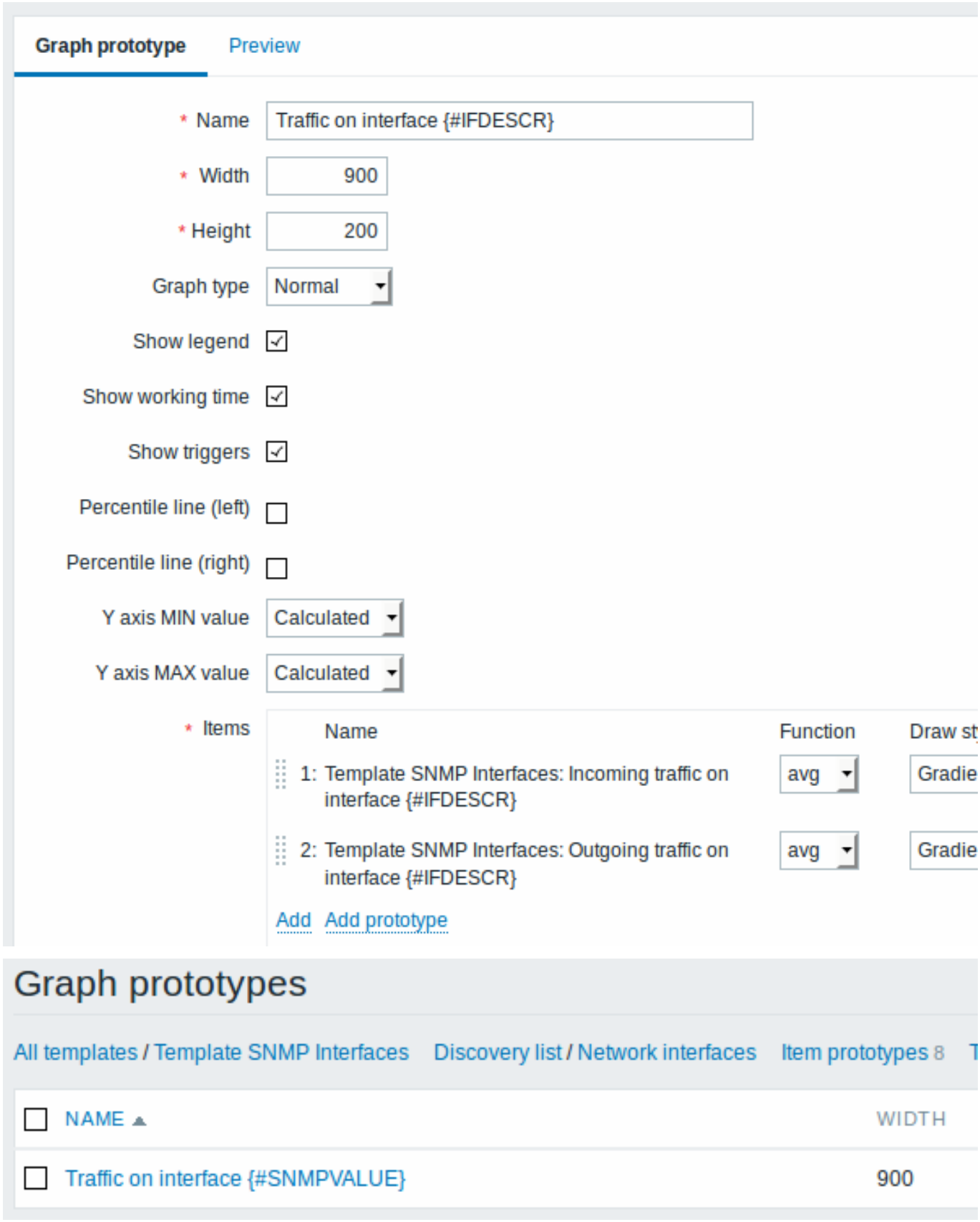
触发器原型

以下截屏说明了我们如何在触发器原型中使用这些宏：

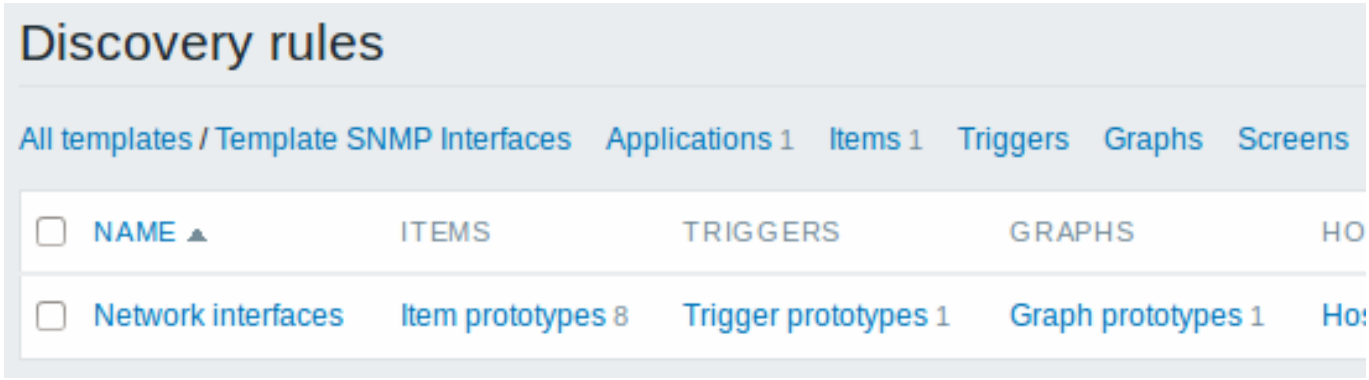

Trigger prototype
Dependencies
* Name
Operational status was changed on {HOST.NAME} int
Severity
Not classified
Information
Warning
Average
High
* Expression
{Template SNMP Interfaces:ifOperStatus[{#IFDESCR}].diff(0)}=1
Add
Expression constructor
OK event generation
Expression
Recovery expression
None
PROBLEM event generation mode
Single
Multiple
OK event closes
All problems
All problems if tag values match
Tags
tag
value
Remove
Add
Allow manual close
URL
Description
Create enabled
Trigger prototypes
All templates / Template SNMP Interfaces
Discovery list / Network interfaces
Item prototypes 8
SEVERITY
NAME
EXPR
Information
Operational status was changed on {HOST.NAME} interface {#IFDESCR}
{Temp

图表原型

以下截屏说明了我们如何在图表原型中使用这些宏：

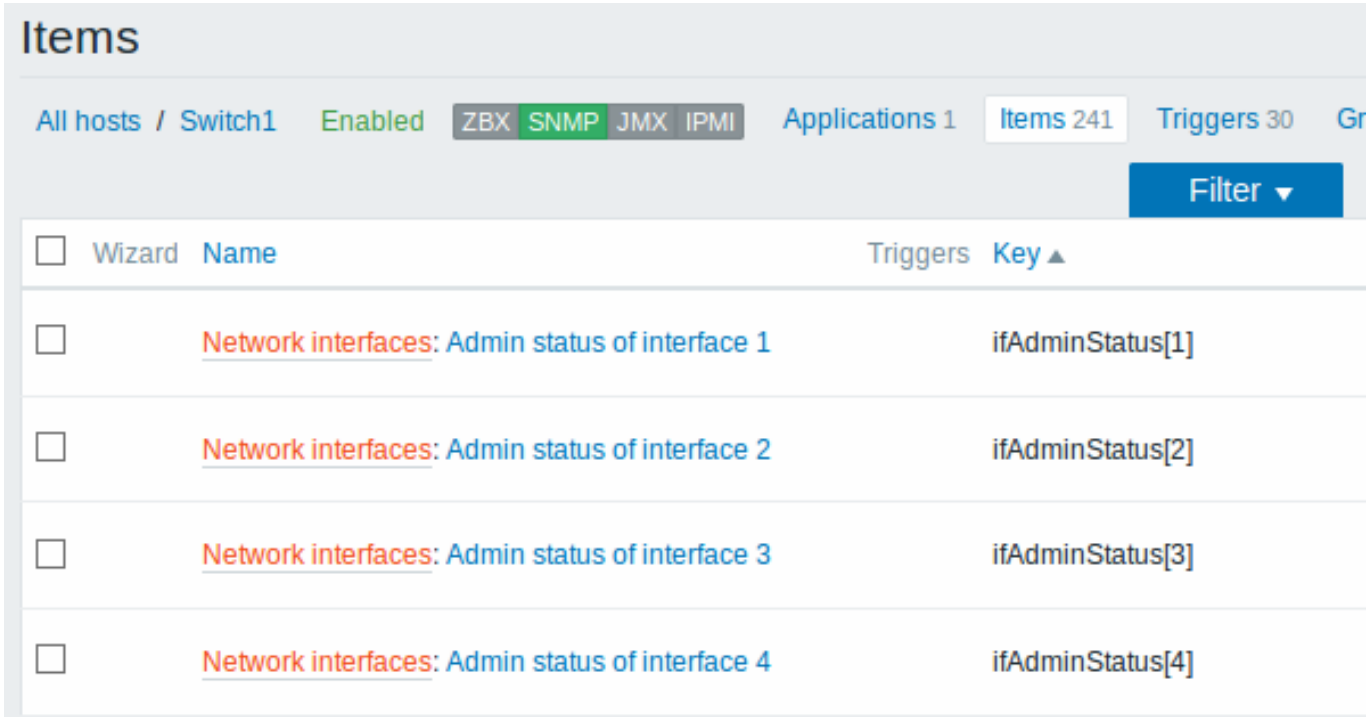


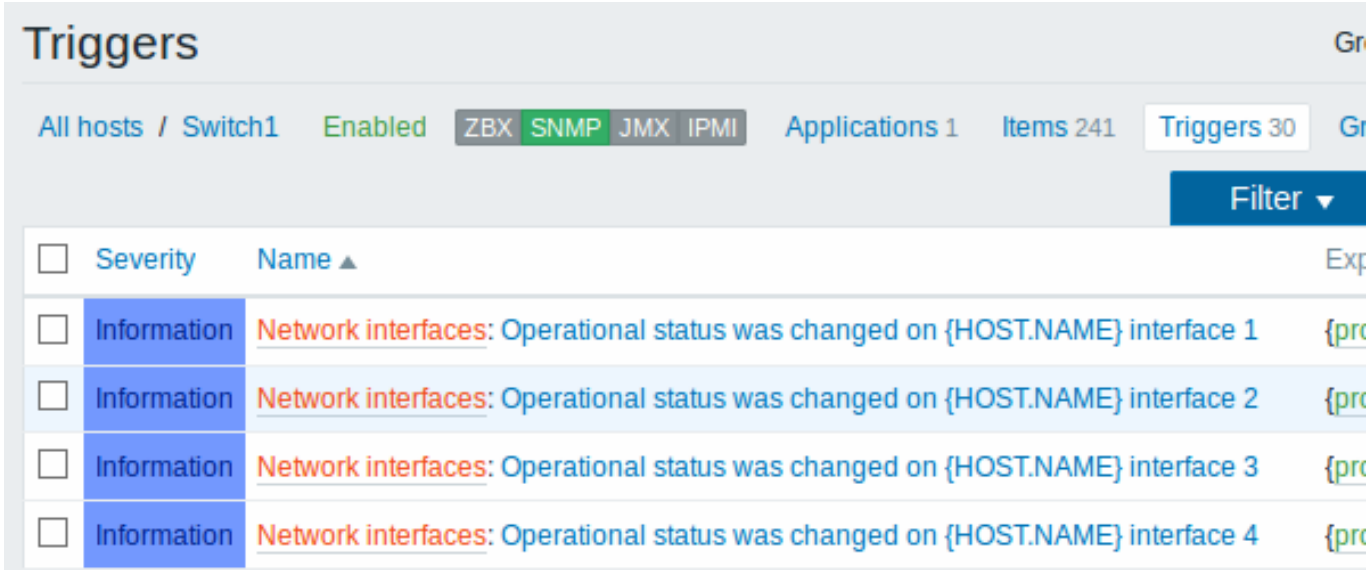
我们发现规则的总结：

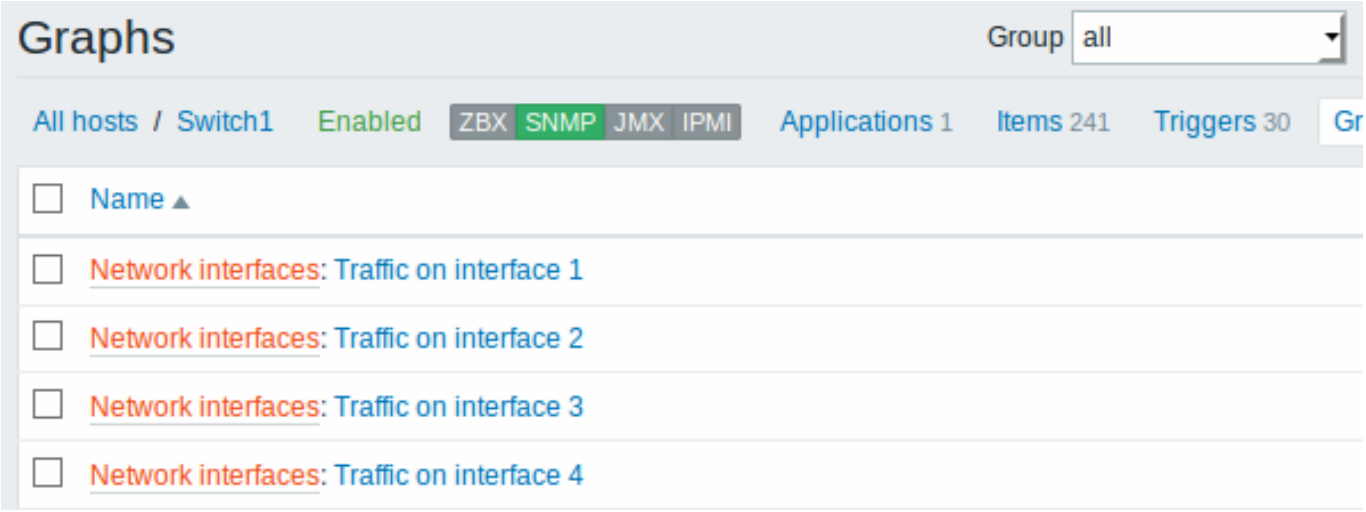


发现实体

当 server 运行时，它会基于 SNMP 发现规则的反馈的价值，创建真实的监控项，触发器和图表。在主机配置中，它们的前缀是橙色链接，指向它们来自的发现规则。







5 发现 **JMX** 对象

概述

这能够发现 全部 JMX MBeans 或 MBean 属性，或指定用于发现这些对象的模式。

必须了解发现规则配置的 Mbean 和 Mbean 属性之间的区别。MBean 是一个对象，可以表示设备，应用程序或需要管理的任何资源。例如一个代表 web-server 的 Mbean。它的属性是连接数，线程数，请求超时，http 文件缓存，内存使用等。用普通人的语言理解这个想法的话，我们可以将咖啡机定义为 Mbean，它具有以下被监控的点：每杯水量，一段时间内的平均水消耗量，每杯所需的咖啡豆数量，咖啡豆和补水时间等。

键值

在发现规则 配置中, 在类型区域选择 **JMX agent**。

该键值为：

```
jmx.discovery[<discovery mode>,<object name>]
```

- 发现模式 - 其中之一：属性（检索 JMX MBean 属性，默认值）或 *beans*（检索 JMX MBeans）
- 对象名称 - 辨别要检索的 MBean 名称的对象名称模式（默认为空，检索所有已注册的 beans）。

你可以参考 使用手册里的 ObjectName 以获取指定对象名称模式的选项。

如果未传递任何参数，则意味着请求 JMX 中的所有 MBean 属性。

> **Attention:**
> 不指定 JMX 发现的参数或尝试接收范围广泛的所有属性 `*:type=*,name=*` 可能会导致潜在的性能问题。

此监控项从 Zabbix Java gateway 3.4 开始支持。

键值举例:

```
jmx.discovery  #检索所有JMX MBean属性
jmx.discovery[beans]  #检索所有 JMX MBeans
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]  #检索所有垃圾收集器属性
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]  #检索所有垃圾收集器
```

此监控项反馈一个 JSON 对象。例如，在发现 MBean 属性（为清楚起见重新格式化）：

```
{
   "data":[
      {
         "{#JMXVALUE}":"0",
         "{#JMXTYPE}":"java.lang.Long",
         "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,CollectionCount",
         "{#JMXATTR}":"CollectionCount"
      },
      {
         "{#JMXVALUE}":"0",
```

```
            "{#JMXTYPE}":"java.lang.Long",
            "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,CollectionTime",
            "{#JMXATTR}":"CollectionTime"
        },
        {
            "{#JMXVALUE}":"true",
            "{#JMXTYPE}":"java.lang.Boolean",
            "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,Valid",
            "{#JMXATTR}":"Valid"
        },
        {
            "{#JMXVALUE}":"PS Scavenge",
            "{#JMXTYPE}":"java.lang.String",
            "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,Name",
            "{#JMXATTR}":"Name"
        },
        {
            "{#JMXVALUE}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXTYPE}":"javax.management.ObjectName",
            "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,ObjectName",
            "{#JMXATTR}":"ObjectName"
        }
    ]
}
```

在发现 MBean 属性（为清楚起见重新格式化）：

```
{
    "data":[
        {
            "{#JMXDOMAIN}":"java.lang",
            "{#JMXTYPE}":"GarbageCollector",
            "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXNAME}":"PS Scavenge"
        }
    ]
}
```

支持宏

以下宏支持在发现规则中的过滤器，监控项，触发器和图表的原型中的应用：

宏	述
发现 MBean 属性	
{#JMXVALUE}	属性值。
{#JMXTYPE}	属性类型。
{#JMXOBJ}	对象名称。
{#JMXDESC}	对象名称，包括属性名称。
{#JMXATTR}	属性名称。
发现 MBeans	
{#JMXDOMAIN}	MBean domain. (*Zabbix* 保留名称)
{#JMXOBJ}	Object name. (*Zabbix* 保留名称)

宏	述
{#JMX<key property>}	MBean properties (like {#JMXTYPE}, {#JMXNAME}). 定义由以下算法从 MBean 属性名创建的 MBean 属性名时需要注意的一些重要事项: 属性名大小写改为大写; 属性名大小写被忽略 (不生成 LLD 宏)，如果它包含不支持的字符。支持的字符可以用以下正则表达式来描述:"A-Z0-9_\."; 如果一个属性名被称为"*obj*" 或"*domain*"，它将被 Zabbix 属性 {#JMXOBJ} 和 {#JMXDOMAIN} 的值所替换 (自 Zabbix 3.4.3 以来支持)。

请考虑 jmx.discovery (以"beans" 模式) 的例子. MBean 定义了以下属性：

```
name=test
  =Type
attributes []=1,2,3
Name=NameOfTheTest
domAin=some
```

作为 JMX 发现的结果，将生成以下 LLD 宏：

- {#JMXDOMAIN} - Zabbix 内部，描述了 MBean 领域
- {#JMXOBJ} - Zabbix 内部，描述了 MBean 对象
- {#JMXNAME} - 从"name" 属性创建

忽略的属性是:

- тип：它的名字内包含无法识别的字母 (non-ASCII)
- attributes[]：它的名字内包含无法识别的字母 (不支持方括号)
- Name：它已经被定义了 (name=test)
- domAin：这是 Zabbix 的保留名称

让我们回顾两个使用 Mbean 创建 LLD 规则的实际示例。要了解收集 Mbeans 的 LLD 规则与收集 Mbean 属性的 LLD 规则之间的区别，请查看下表：

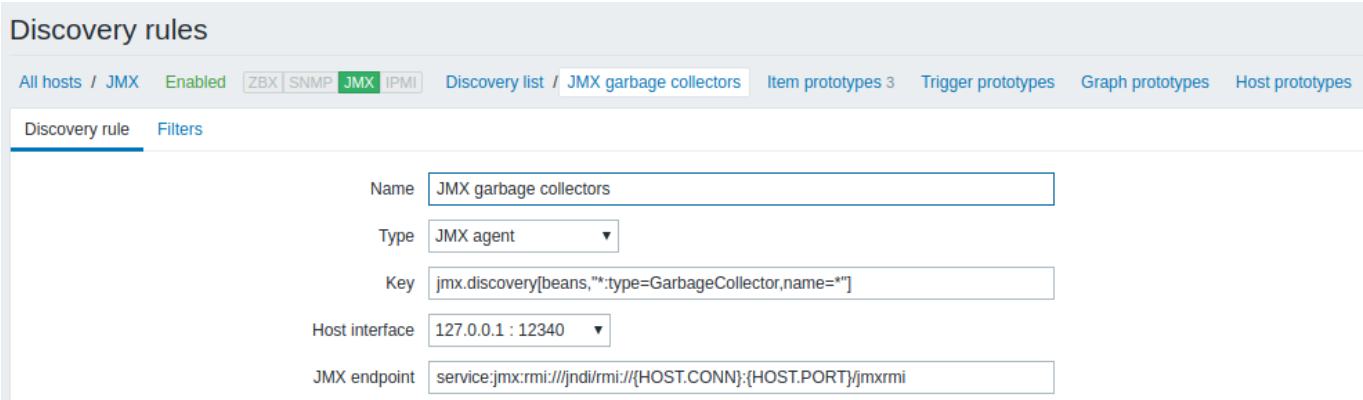
MBean1	**MBean2**	**MBean3**
MBean1Attribute1	MBean2Attribute1	MBean3Attribute1
MBean1Attribute2	MBean2Attribute2	MBean3Attribute2
MBean1Attribute3	MBean2Attribute3	MBean3Attribute3

以 LLD 规则收集 Mbeans

规则将会反馈三个对象: 该列的顶行：MBean1, MBean2, MBean3.

有关对象的更多信息，请参阅支持宏 表格, 发现 *MBean* 部分。

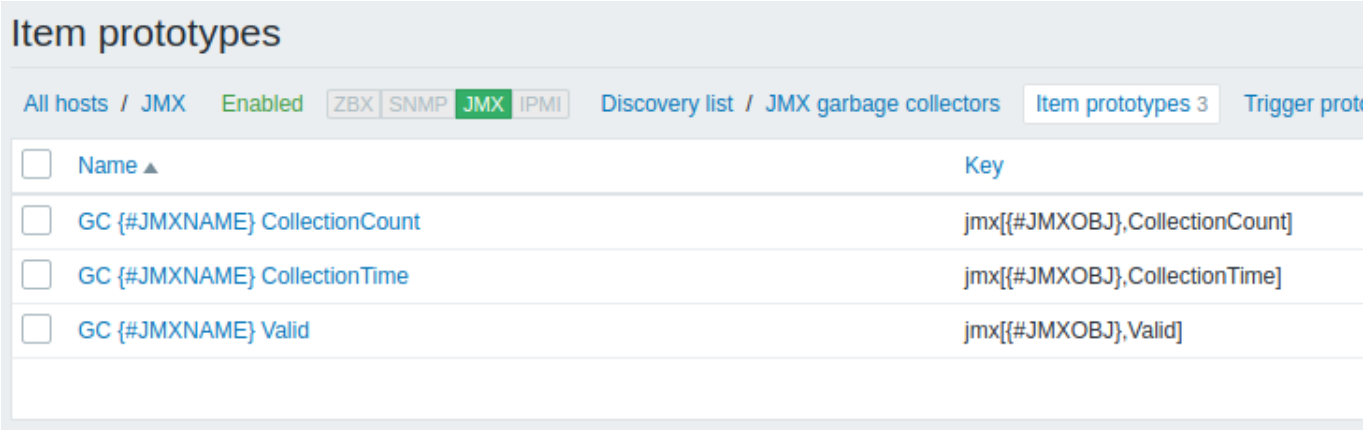
收集 Mbeans（无属性）的发现规则配置如下所示：



使用键值:

```
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]
```

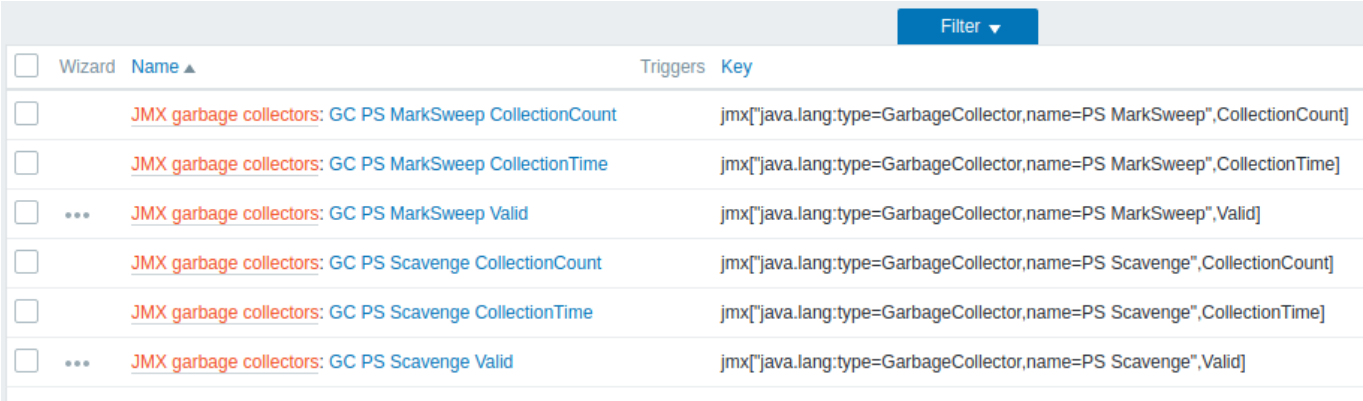
能发现所有没有属性的垃圾收集器。由于垃圾收集器具有相同的属性集，我们可以通过以下方式在项原型中使用所需的属性：



使用键值:

```
jmx[{#JMXOBJ},CollectionCount]
jmx[{#JMXOBJ},CollectionTime]
jmx[{#JMXOBJ},Valid]
```

LLD 发现规则将导致与此接近的内容（为两个垃圾收集器发现的监控项）：

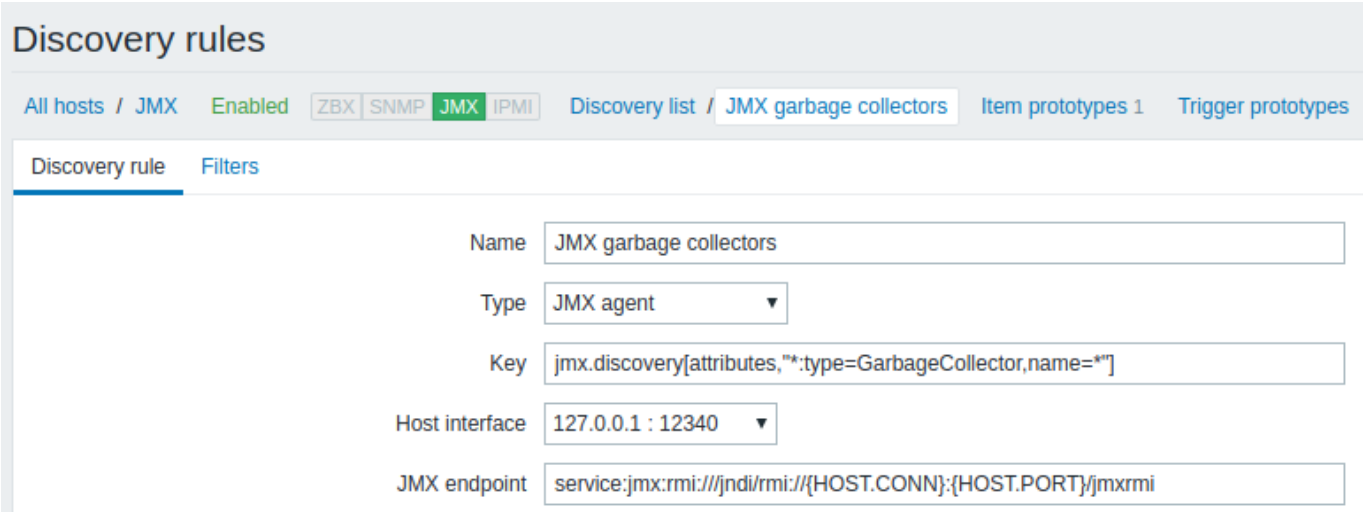


LLD 规则收集 MBean 属性

这条规则将会反馈 9 个对象：MBean1Attribute1，MBean2Attribute1，Mbean3Attribute1,MBean1Attribute2,MBean2Attribute2, Mbean3Attribute2, MBean1Attribute3, MBean2Attribute3, Mbean3Attribute3.

更多有关于对象的信息，请参考支持宏 表格, 发现 *MBean* 属性部分.

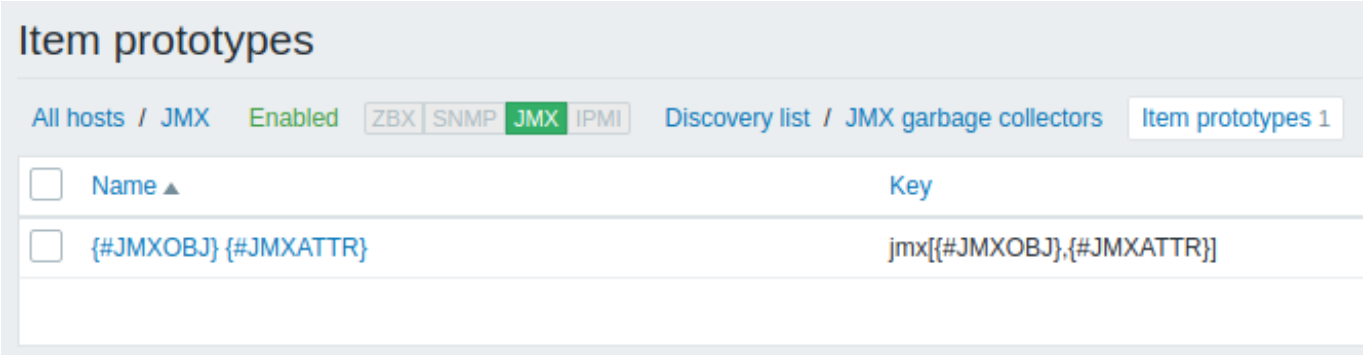
收集 MBean 属性的发现规则配置如以下所示：



使用键值:

```
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]
```

将发现具有单个项属性的所有垃圾收集器。



在这种特殊情况下，将从原型为每个 MBean 属性创建一个监控项。这种配置的主要缺点是从触发器原型的触发器创建是不可能的，因为所有属性只有一个监控项原型。因此，此设置可用于数据收集，但不建议用于自动监控。

Examples

Let's review two more practical examples of a LLD rule creation with the use of Mbean. To understand the difference between a LLD rule collecting Mbeans and a LLD rule collecting Mbean attributes better please take a look at following table:

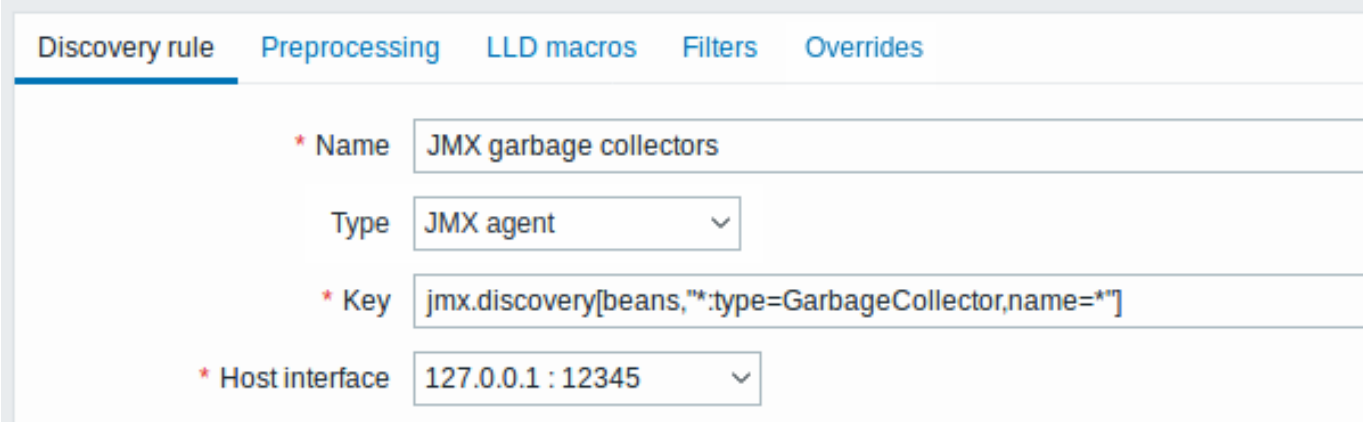
MBean1	**MBean2**	**MBean3**
MBean1Attribute1	MBean2Attribute1	MBean3Attribute1
MBean1Attribute2	MBean2Attribute2	MBean3Attribute2
MBean1Attribute3	MBean2Attribute3	MBean3Attribute3

Example 1: Discovering Mbeans

This rule will return 3 objects: the top row of the column: MBean1, MBean2, MBean3.

For more information about objects please refer to supported macros table, *Discovery of MBeans* section.

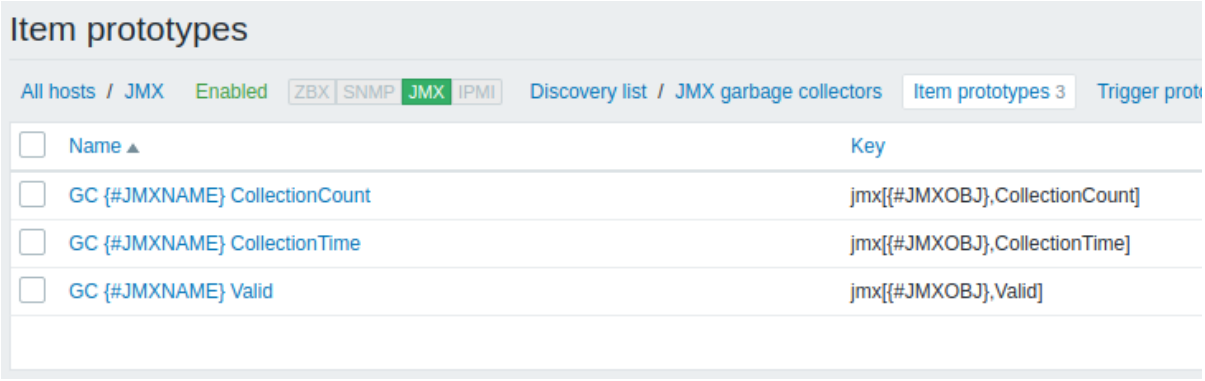
Discovery rule configuration collecting Mbeans (without the attributes) looks like the following:



Key used:

```
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]
```

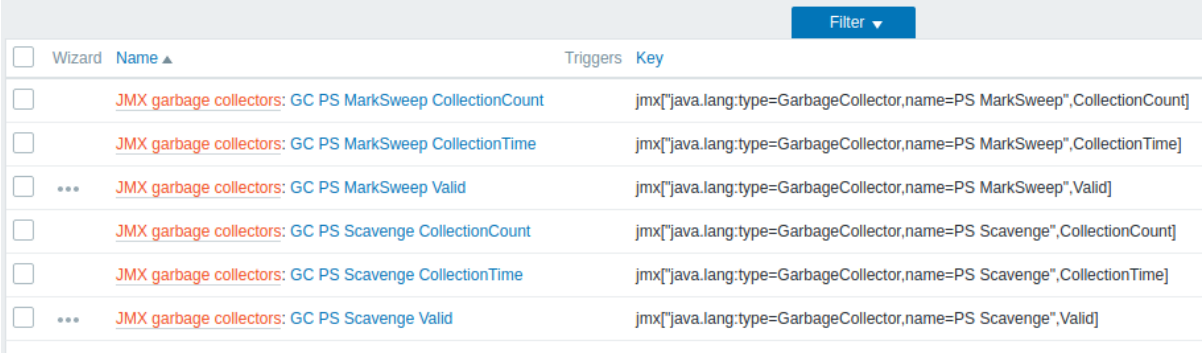
All the garbage collectors without attributes will be discovered. As Garbage collectors have the same attribute set, we can use desired attributes in item prototypes the following way:



Keys used:

```
jmx[{#JMXOBJ},CollectionCount]
jmx[{#JMXOBJ},CollectionTime]
jmx[{#JMXOBJ},Valid]
```

LLD discovery rule will result in something close to this (items are discovered for two Garbage collectors):

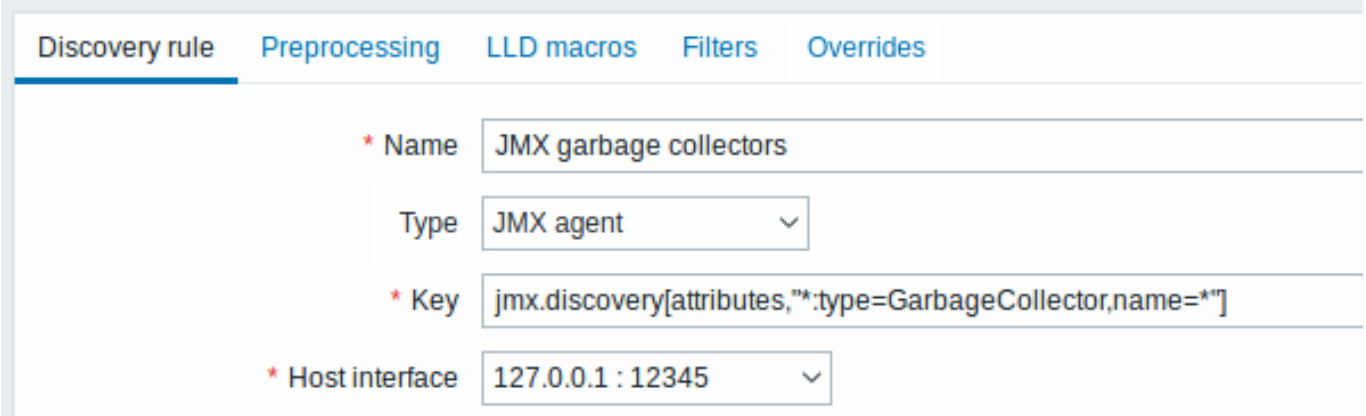


Example 2: Discovering Mbean attributes

This rule will return 9 objects with the following fields: MBean1Attribute1, MBean2Attribute1, Mbean3Attribute1,MBean1Attribute2,MBean2Attr Mbean3Attribute2, MBean1Attribute3, MBean2Attribute3, Mbean3Attribute3.

For more information about objects please refer to supported macros table, *Discovery of MBean attributes* section.

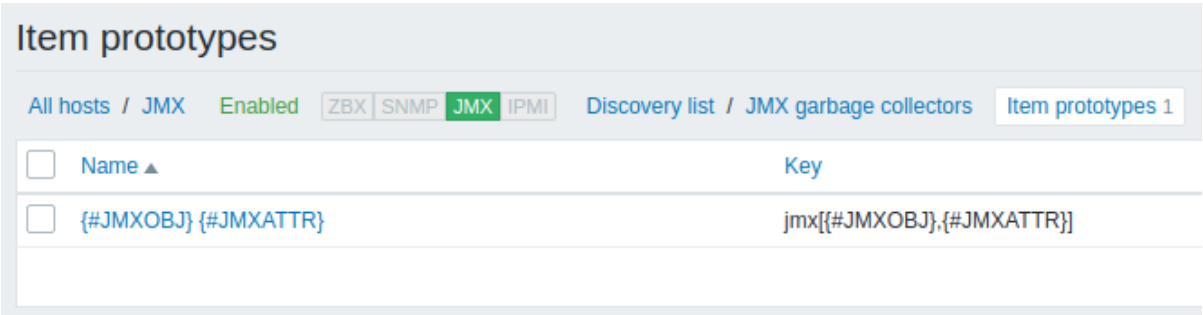
Discovery rule configuration collecting Mbean attributes looks like the following:



Key used:

```
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]
```

All the garbage collectors with a single item attribute will be discovered.



In this particular case an item will be created from prototype for every MBean attribute. The main drawback of this configuration is that trigger creation from trigger prototypes is impossible as there is only one item prototype for all attributes. So this setup can be used for data collection, but is not recommended for automatic monitoring.

Using jmx.get

`jmx.get[]` is similar to the `jmx.discovery[]` item, but is does not turn Java object properties into low-level discovery macro names and therefore can return values without limitations that are associated with LLD macro name generation such as hyphens or non-ASCII characters.

When using `jmx.get[]` for discovery, low-level discovery macros can be defined separately in the custom LLD macro tab of the discovery rule configuration, using JSONPath to point to the required values.

Discovering MBeans

Discovery item: `jmx.get[beans,"com.example:type=*,*"]`

Response:

```
[
    {
        "object": "com.example:type=Hello,data-src=data-base,  =    ",
        "domain": "com.example",
        "properties": {
            "data-src": "data-base",
            "  ": "    ",
            "type": "Hello"
        }
    },
    {
        "object": "com.example:type=Atomic",
        "domain": "com.example",
        "properties": {
            "type": "Atomic"
        }
    }
]
```

Discovering MBean attributes

Discovery item: `jmx.get[attributes,"com.example:type=*,*"]`

Response:

```
[
    {
        "object": "com.example:type=*",
        "domain": "com.example",
        "properties": {
            "type": "Simple"
        }
    },
    {
        "object": "com.zabbix:type=yes,domain=zabbix.com,data-source=/dev/rand,  =    ,obj=true",
        "domain": "com.zabbix",
        "properties": {
            "type": "Hello",
            "domain": "com.example",
            "data-source": "/dev/rand",
            "  ": "    ",
            "obj": true
        }
    }
]
```

6 发现 **IPMI** 传感器

概述

可以自动发现 IPMI 传感器。

要做到这一点，你可以结合以下方式实现:

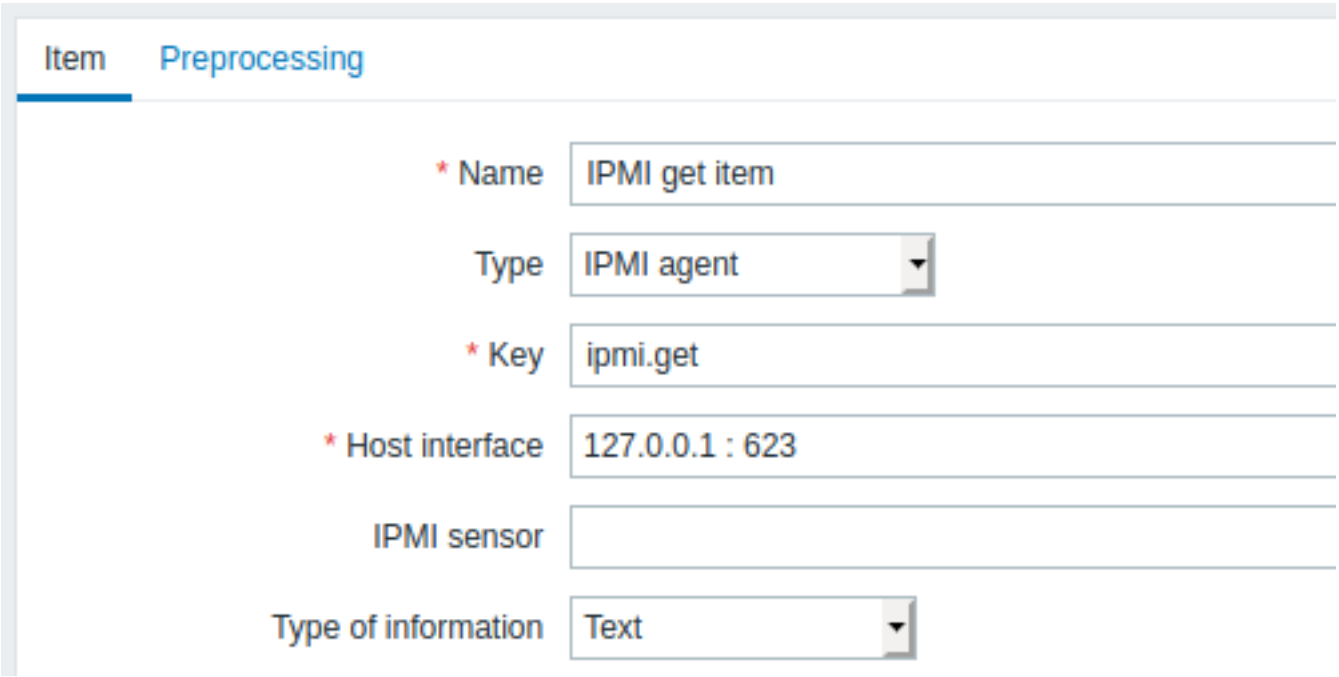
- 主选项选择 `ipmi.get` IPMI 监控项 (Zabbix **5.0.0** 及以上版本支持)
- 依赖低级别发现规则和监控项原型

配置

主要项

使用以下键值创建 IPMI 监控项:

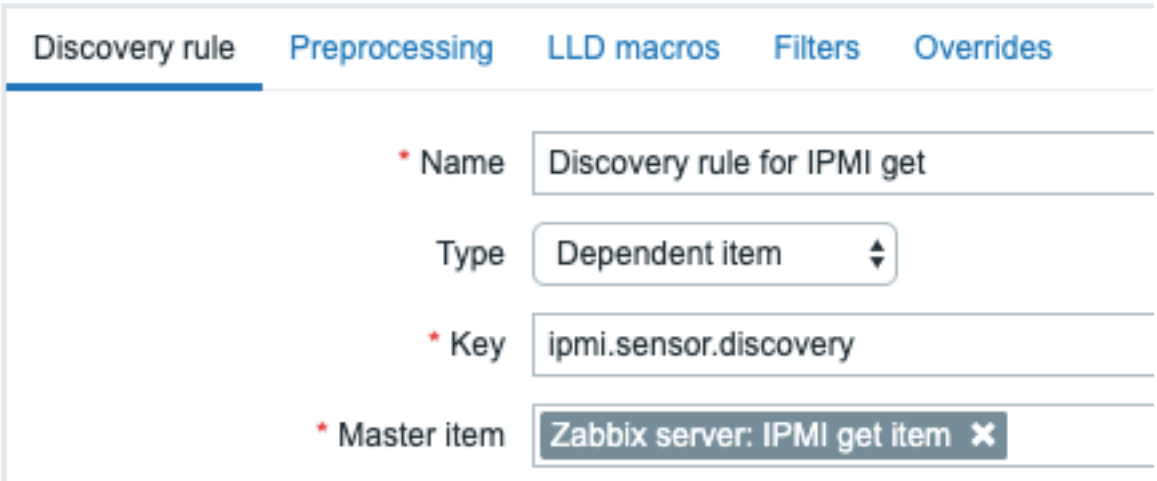
ipmi.get



对于可能较大的 JSON 数据，设置信息类型为”Text”.

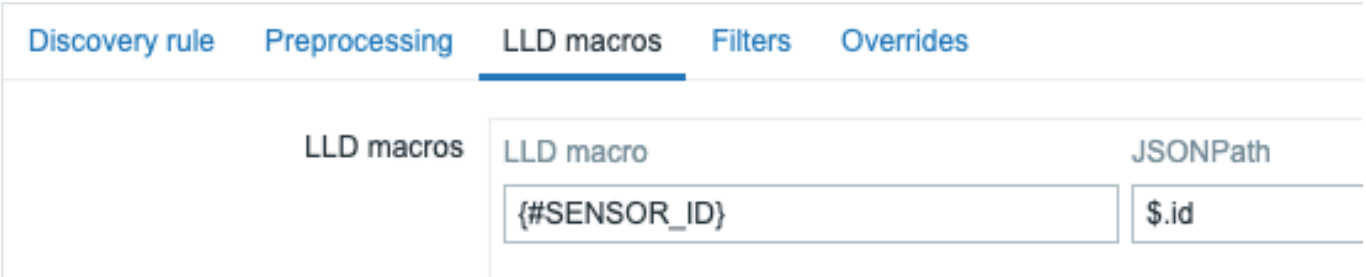
依赖 LLD 规则

创建低级别发现规则为 “依赖项” 类型:



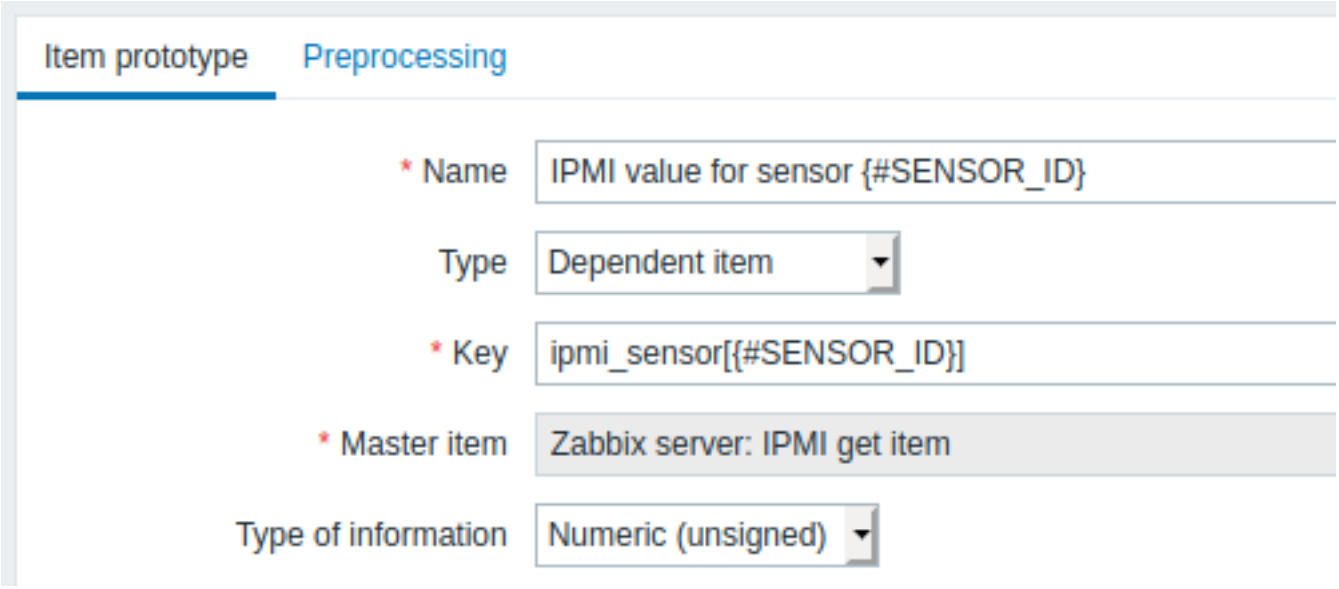
主要项选择 ipmi.get 创建的我们的监控项.

在”LLD 宏” 标签中定义了一个自定义宏，其对应的 JSONPath:



依赖项的原型

在这个 LLD 规则中创建一个带有 “依赖项” 类型的监控项原型. 作为这个原型的主要项，选择 ipmi.get 创建我们的监控项.



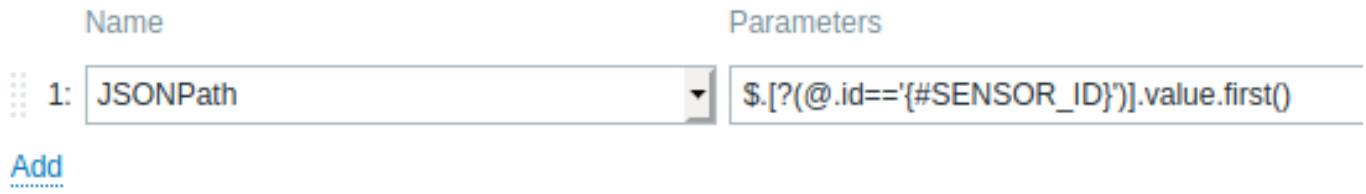
注意在监控项原型名和键中使用了 {#SENSOR_ID} 宏:

- 名称: IPMI value for sensor {#SENSOR_ID}
- 键: ipmi_sensor[{#SENSOR_ID}]

状态类型, 数字 (无正负).

在监控项原型" 预处理" 选项卡中选择 JSONPath 并使用以下 JSONPath 表达式作为参数:

```
$.[?(@.id=='{#SENSOR_ID}')].value.first()
```



当自动发现启动时, 将为每个 IPMI 传感器创建一个项目. 该项将返回给定传感器的整数值。

7 自动发现 **systemd** 服务

概述

zabbix 可以通过自动发现 发现 systemd 服务 (默认情况下是系统服务)

监控项键

可以在自动发现规则 使用的监控项包含以下:

```
systemd.unit.discovery
```

Attention:
监控项 这些键只支持 Zabbix agent 2.

该项返回一个带有 systemd 单元信息的 JSON，例如:

```
[{
    "{#UNIT.NAME}": "mysqld.service",
    "{#UNIT.DESCRIPTION}": "MySQL Server",
    "{#UNIT.LOADSTATE}": "loaded",
    "{#UNIT.ACTIVESTATE}": "active",
    "{#UNIT.SUBSTATE}": "running",
    "{#UNIT.FOLLOWED}": "",
    "{#UNIT.PATH}": "/org/freedesktop/systemd1/unit/mysqld_2eservice",
    "{#UNIT.JOBID}": 0,
    "{#UNIT.JOBTYPE}": ""
```

```
    "{#UNIT.JOBPATH}": "/",
    "{#UNIT.UNITFILESTATE}": "enabled"
}, {
    "{#UNIT.NAME}": "systemd-journald.socket",
    "{#UNIT.DESCRIPTION}": "Journal Socket",
    "{#UNIT.LOADSTATE}": "loaded",
    "{#UNIT.ACTIVESTATE}": "active",
    "{#UNIT.SUBSTATE}": "running",
    "{#UNIT.FOLLOWED}": "",
    "{#UNIT.PATH}": "/org/freedesktop/systemd1/unit/systemd_2djournald_2esocket",
    "{#UNIT.JOBID}": 0,
    "{#UNIT.JOBTYPE}": "",
    "{#UNIT.JOBPATH}": "/"
    "{#UNIT.UNITFILESTATE}": "enabled"
}]
```

支持的宏

在自动发现规则过滤、监控项、触发器、图形的原型中支持使用以下宏：

宏	述
{#UNIT.NAME}	单元名称.
{#UNIT.DESCRIPTION}	单元描述.
{#UNIT.LOADSTATE}	加载状态 (单元文件是否已成功加载)
{#UNIT.ACTIVESTATE}	活动状态 (单元文件当前是否启动)
{#UNIT.SUBSTATE}	子状态 (活动状态的更细粒度版本，它特定于单元类型，而活动状态不是)
{#UNIT.FOLLOWED}	在其状态下被该单元 (如果有的话) 跟随的单元; 否则为空字符串.
{#UNIT.PATH}	单元文件路径.
{#UNIT.JOBID}	如果作业单元有作业排队，则作业 ID 为数字;0, 否则.
{#UNIT.JOBTYPE}	工作单元状态.
{#UNIT.JOBPATH}	工作单元路径.
{#UNIT.UNITFILESTATE}	单元文件的安装状态 (从 5.0.6 开始支持).

监控项原型

可以基于 systemd 服务发现创建的监控项原型, 列如：

- 监控项名称: {#UNIT.DESCRIPTION}; 监控项键值: systemd.unit.info["{#UNIT.NAME}"]
- 监控项名称: {#UNIT.DESCRIPTION}; 监控项键值: systemd.unit.info["{#UNIT.NAME}",LoadState]

systemd.unit.info agent 监控项 Zabbix 4.4 以上的支持.

8 Windows 发现服务

概述

与发现文件系统的方式相似，同样可以以此 Windows 发现服务。

键值

在发现规则 中使用的监控项是

service.discovery

此监控项从 Zabbix Windows agent 3.0 开始支持。

支持宏

支持在发现规则过滤器，以及监控项，触发器和图表原型中使用以下宏：

宏	述
{#SERVICE.NAME}	服务名称
{#SERVICE.DISPLAYNAME}	展示中的服务名称
{#SERVICE.DESCRIPTION}	服务描述

宏	述
{#SERVICE.STATE}	服务状态数值： 0 - 运行中 1 - 暂停 2 - 开始待定 3 - 暂停待定 4 - 继续待定 5 - 停止待定 6 - 停止 7 - 未知
{#SERVICE.STATENAME}	服务状态名称 (运行, 暂停, 开始待定, 暂停待定, 继续待定, 停止待定, 停止或 未知).
{#SERVICE.PATH}	服务路径
{#SERVICE.USER}	服务用户
{#SERVICE.STARTUP}	服务启动类型数值： 0 - 自动 1 - 自动延迟 2 - 手动 3 - 禁用 4 - 未知
{#SERVICE.STARTUPNAME}	服务启动类型名称 (自动, 自动延迟, 手动, 禁用, 未知).
{#SERVICE.STARTUPTRIGGER}	用于指示服务启动类型是否具有以下内容的数值： 0 - 没有启动触发器 1 - 已启动触发器 此宏从 Zabbix 3.4.4 开始支持。发现诸如自动（触发器启动），自动延迟（触发器启动）和手动（触发器启动）等服务启动类型非常有用。

基于 Windows 发现服务，你可以创建一个监控项原型，如：

```
service.info[{#SERVICE.NAME},<param>]
```

其中 `param` 接受以下值: *state*, *displayname*, *path*, *user*, *startup* 或 *description*.

例如，要获取服务的显示名称，你可以使用”service.info[{#SERVICE.NAME},displayname]” 的监控项。如果未指定 `param` 值 (”service.info[{#SERVICE.NAME}]”)，则使用默认 *state* 参数。

9 发现 **Windows** 性能计数器实例

概述

可以发现Windows 性能计数器的对象实例。这对于多实例性能计数器很有用。

监控项值

在发现规则中使用的项

```
perf_instance.discovery[object]
```

或者，能够只提供英文对象名，独立于操作系统本地化:

```
perf_instance_en.discovery[object]
```

示例:

```
perf_instance.discovery[Processador]
perf_instance_en.discovery[Processor]
```

Zabbix Windows agent 5.0.1 开始支持。

支持宏

发现将返回 {#INSTANCE} 宏中指定对象的所有实例，这些实例可以用于 perf_count 和 perf_count_en 项的原型中。

```
[
    {"{#INSTANCE}":"0"},
    {"{#INSTANCE}":"1"},
```

```
    {"{#INSTANCE}":"_Total"}
]
```

例如，如果发现规则中使用的项目键为:

```
perf_instance.discovery[Processor]
```

你可以创建一个项目原型:

```
perf_counter["\Processor({#INSTANCE})\% Processor Time"]
```

注意:

*如果指定的对象不被发现或者不支持变量实例，那么发现项将不被支持。

*如果指定的对象支持可变实例，但目前没有任何实例，则返回一个空JSON数组。

*如果有重复的实例，它们将被跳过。

10 使用 **WMI** 查询发现

概述

WMI 是 Windows 中的一个功能强大的界面，可用于检索有关 Windows 组件、服务、状态和安装的软件的各种信息。

它可以用于物理磁盘发现及其性能数据收集、网络接口发现、Hyper-V 客户发现、监视 Windows 服务和 Windows 操作系统中的许多其他事情。这种低级的发现使用 WQL 查询完成，查询的结果会自动转换为适合低级发现的 JSON 对象。

监控项值

发现规则中使用的项是:

```
wmi.getall[<namespace>,<query>]
```

监控项将查询结果转换成一个 JSON 数组。例如:

```
select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'
```

返回的内容:

```
[
    {
        "DeviceID" : "\\.\PHYSICALDRIVE0",
        "BytesPerSector" : 512,
        "Capabilities" : [
            3,
            4
        ],
        "CapabilityDescriptions" : [
            "Random Access",
            "Supports Writing"
        ],
        "Caption" : "VBOX HARDDISK ATA Device",
        "ConfigManagerErrorCode" : "0",
        "ConfigManagerUserConfig" : "false",
        "CreationClassName" : "Win32_DiskDrive",
        "Description" : "Disk drive",
        "FirmwareRevision" : "1.0",
        "Index" : 0,
        "InterfaceType" : "IDE"
    },
    {
        "DeviceID" : "\\.\PHYSICALDRIVE1",
        "BytesPerSector" : 512,
        "Capabilities" : [
            3,
            4
        ],
        "CapabilityDescriptions" : [
```

```
            "Random Access",
            "Supports Writing"
        ],
        "Caption" : "VBOX HARDDISK ATA Device",
        "ConfigManagerErrorCode" : "0",
        "ConfigManagerUserConfig" : "false",
        "CreationClassName" : "Win32_DiskDrive",
        "Description" : "Disk drive",
        "FirmwareRevision" : "1.0",
        "Index" : 1,
        "InterfaceType" : "IDE"
    }
]
```

Zabbix Windows agent 4.4 开始支持

低级别发现宏

即使在返回的 JSON 中没有创建低级发现宏，用户也可以通过使用自定义 LLD 宏功能定义这些宏，并使用 JSONPath 指向返回的 JSON 中发现的值。这些宏可以用来创建项目、触发器等原型。

11 使用 ODBC SQL 查询发现

概述

这种低级的发现是通过 SQL 查询完成的，查询结果会自动转换为适合低级发现的 JSON 对象。

监控项键值

SQL 查询使用" 数据库监视" 项类型执行。因此，在ODBC 监控页上的大多数说明适用于获得一个工作的 "数据库监视器" 发现规则，唯一的区别是:

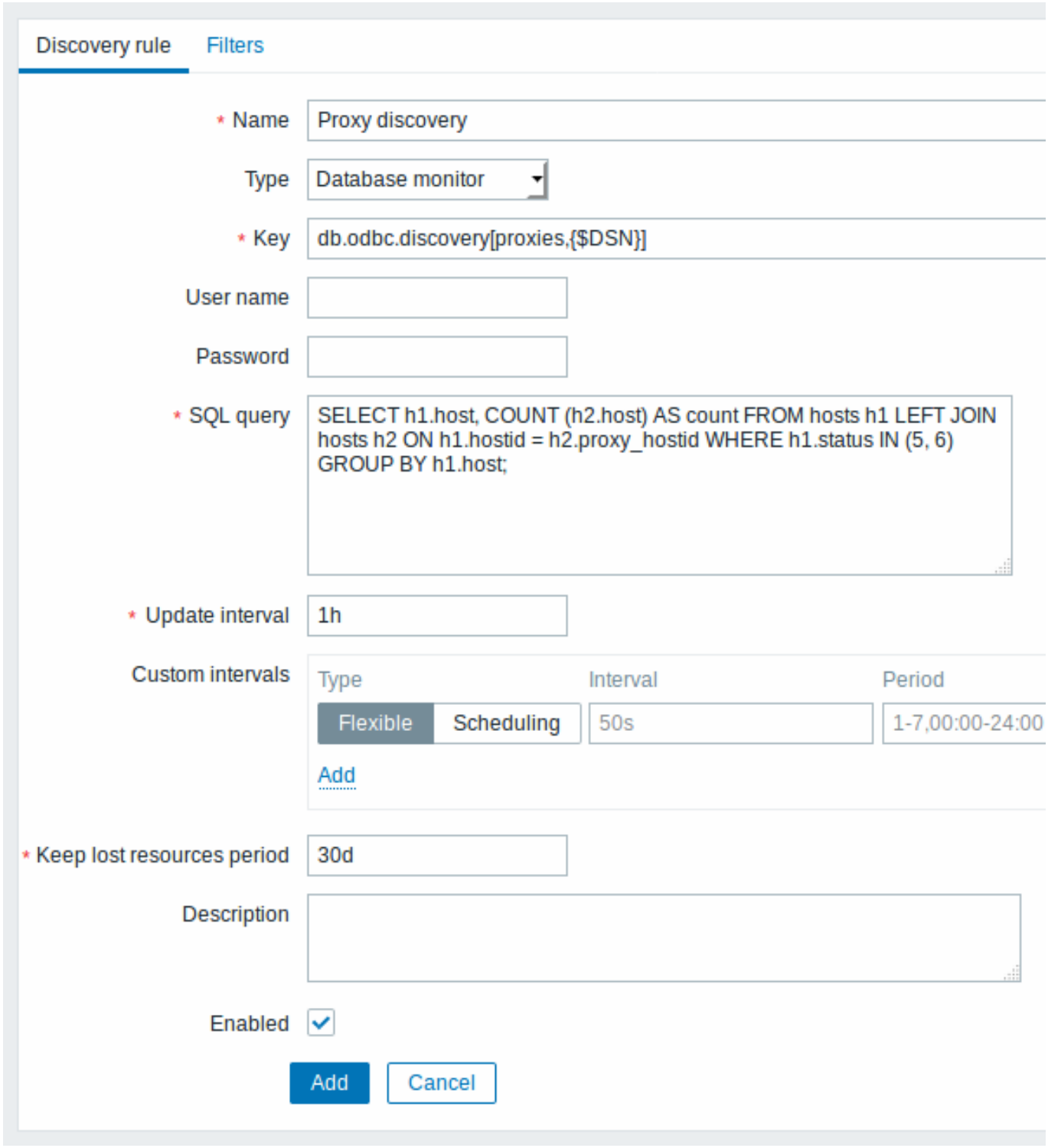
`db.odbc.discovery[<description>,<dsn>]`

键值可以被替换"db.odbc.select[<description>,<dsn>]".

Zabbix server/proxy 3.0 开始支持使用 SQL 查询进行发现。

作为一个演示如何将 SQL 查询转换为 JSON 的实际示例，让我们考虑通过在 Zabbix 数据库上执行 ODBC 查询来发现 Zabbix 代理。这对于自动创建"zabbix[proxy,<name>,lastaccess]" 内部监控项 来监视哪些代理是活的很有用。

让我们从发现规则配置开始:



所有强制输入字段都用红色星号标记。

Here, the following direct query on Zabbix database is used to select all Zabbix proxies, together with the number of hosts they are monitoring. The number of hosts can be used, for instance, to filter out empty proxies: 这里，使用以下对 Zabbix database 的直接查询来选择所有 Zabbix proxies，以及它们正在监视的主机数量。例如，可以使用主机的数量来过滤空代理:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_ho
+---------+-------+
| host    | count |
+---------+-------+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+---------+-------+
3 rows in set (0.01 sec)
```

通过"db.odbc.discovery[]" 的内部监控项，这个查询的结果自动转换成 JSON:

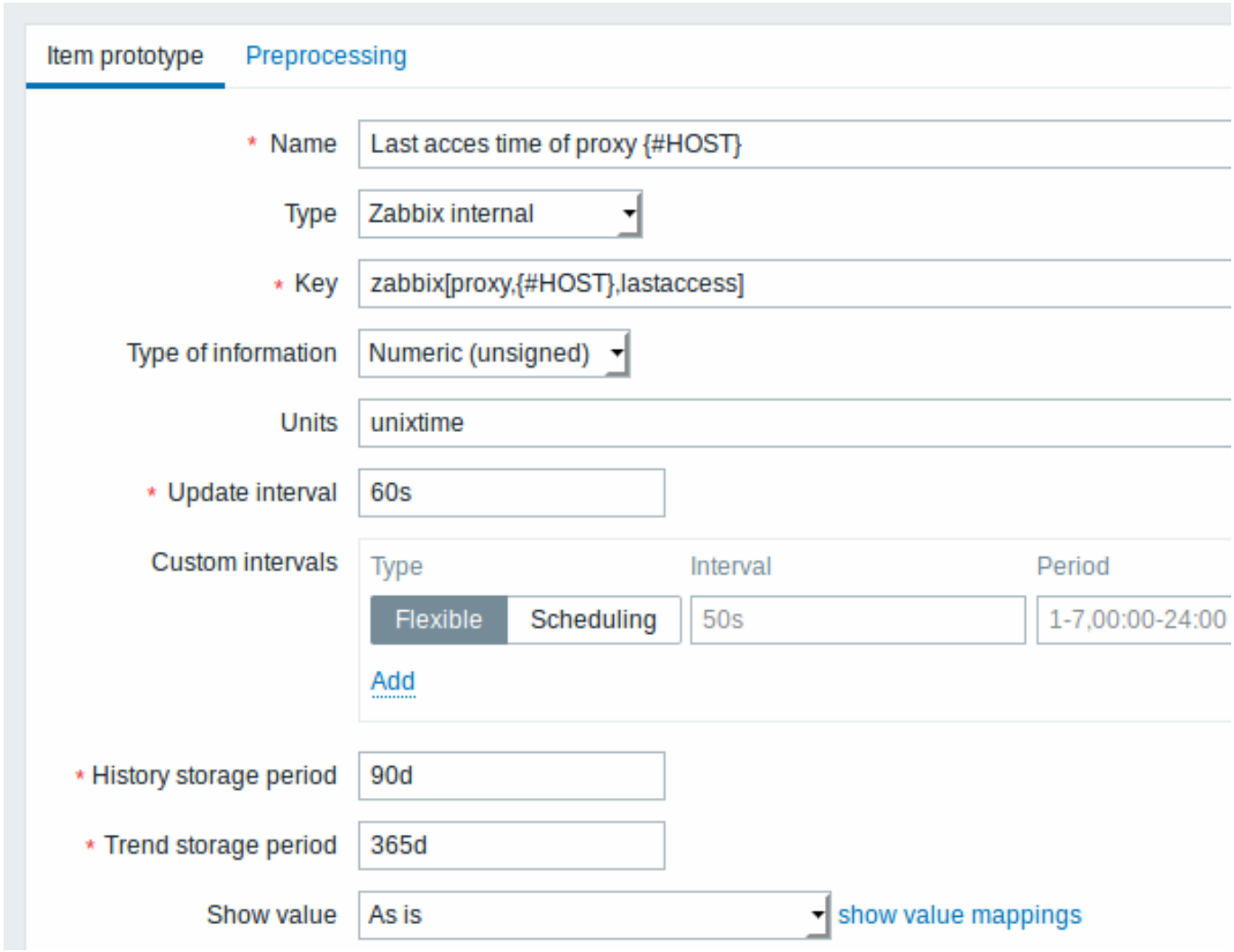
```
{
    "data": [
        {
            "{#HOST}": "Japan 1",
            "{#COUNT}": "5"
        },
        {
            "{#HOST}": "Japan 2",
            "{#COUNT}": "12"
        },
        {
            "{#HOST}": "Latvia",
            "{#COUNT}": "3"
        }
    ]
}
```

可以看到，列名变成了宏名，所选的行变成了这些宏的值。

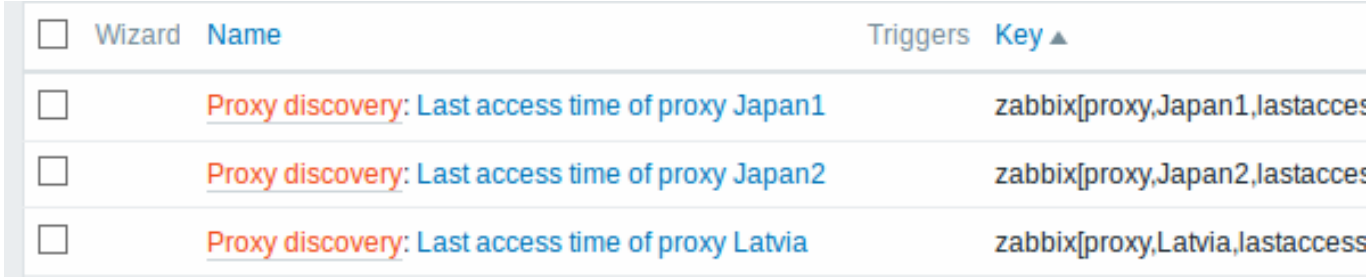
Note:
如果列名转换成宏名的方式不明显，建议使用列名别名，如上面示例中的"COUNT(h2.host) AS COUNT"。
如果无法将列名转换为有效的宏名，则不支持发现规则，错误消息将详细说明出错的列号。如果需要额外的帮助，在 Zabbix Server 日志文件的 DebugLevel=4 下提供获得的列名:

```
$ grep db.odbc.discovery /tmp/zabbix_server.log
 ...
 23876:20150114:153410.856 In db_odbc_discovery() query:'SELECT h1.host, COUNT(h2.host) FROM hosts h1 I
 23876:20150114:153410.860 db_odbc_discovery() column[1]:'host'
 23876:20150114:153410.860 db_odbc_discovery() column[2]:'COUNT(h2.host)'
 23876:20150114:153410.860 End of db_odbc_discovery():NOTSUPPORTED
 23876:20150114:153410.860 Item [Zabbix server:db.odbc.discovery[proxies,{$DSN}]] error: Cannot convert
```

现在我们了解了 SQL 查询如何转换为 JSON 对象，我们可以在 item 原型中使用 {#HOST} 宏:



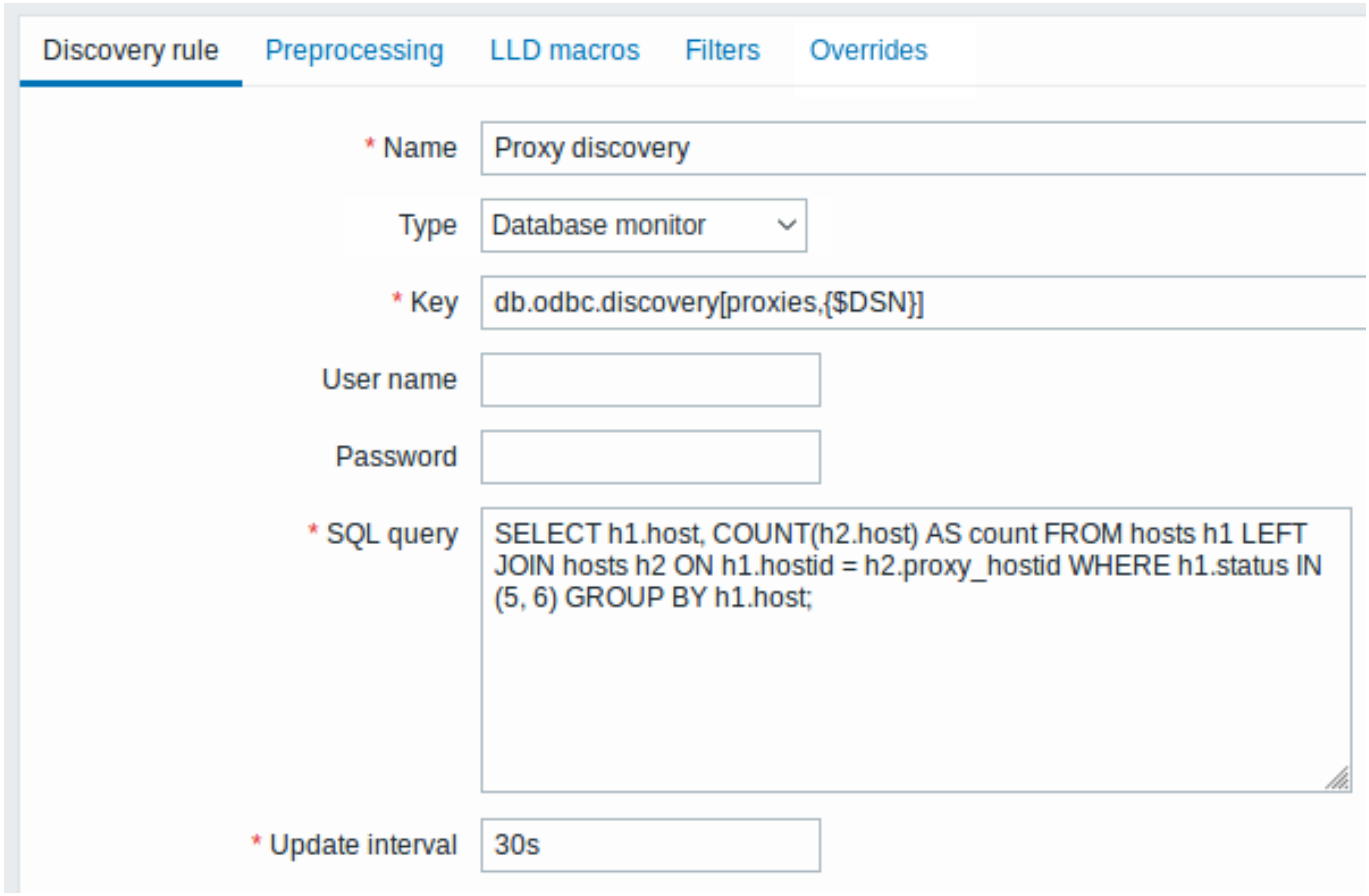
一旦发现 zabbix proxy 被执行，将为每个代理创建一个监控项:



Using db.odbc.discovery

As a practical example to illustrate how the SQL query is transformed into JSON, let us consider low-level discovery of Zabbix proxies by performing an ODBC query on Zabbix database. This is useful for automatic creation of "zabbix[proxy,<name>,lastaccess]" internal items to monitor which proxies are alive.

Let us start with discovery rule configuration:



All mandatory input fields are marked with a red asterisk.

Here, the following direct query on Zabbix database is used to select all Zabbix proxies, together with the number of hosts they are monitoring. The number of hosts can be used, for instance, to filter out empty proxies:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_ho
+---------+-------+
| host    | count |
+---------+-------+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+---------+-------+
3 rows in set (0.01 sec)
```

By the internal workings of "db.odbc.discovery[,{$DSN}]" item, the result of this query gets automatically transformed into the following JSON:

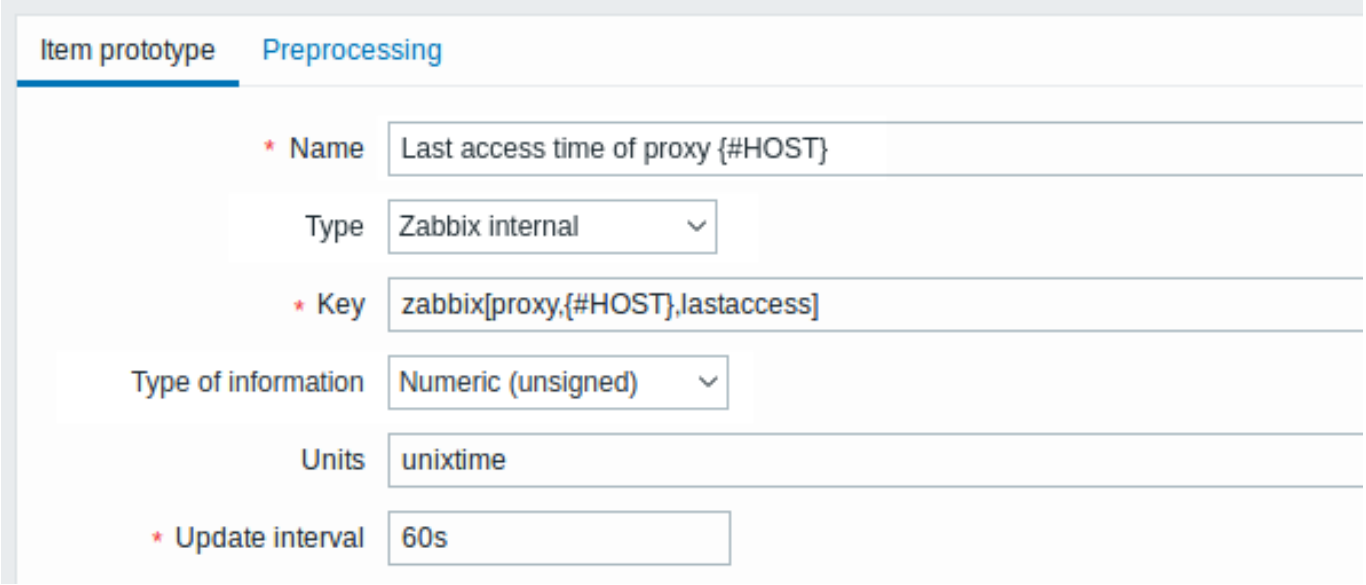
```
[
    {
        "{#HOST}": "Japan 1",
        "{#COUNT}": "5"
    },
    {
        "{#HOST}": "Japan 2",
        "{#COUNT}": "12"
    },
    {
        "{#HOST}": "Latvia",
        "{#COUNT}": "3"
    }
]
```

It can be seen that column names become macro names and selected rows become the values of these macros.

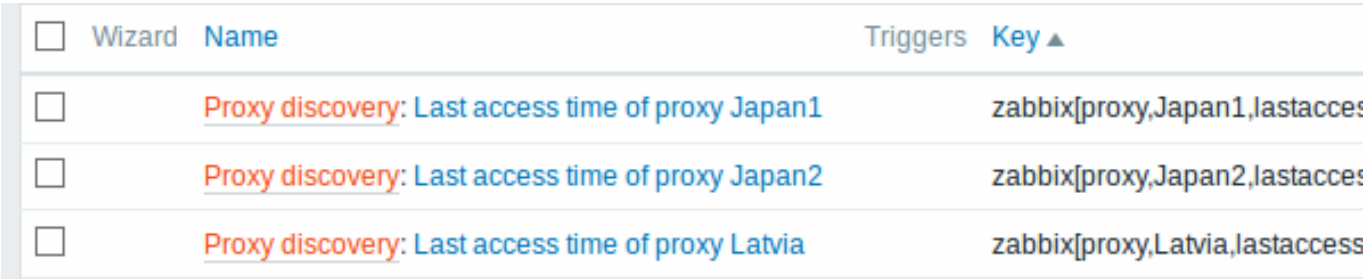
Note:
If it is not obvious how a column name would be transformed into a macro name, it is suggested to use column aliases like "COUNT(h2.host) AS count" in the example above.
In case a column name cannot be converted into a valid macro name, the discovery rule becomes not supported, with the error message detailing the offending column number. If additional help is desired, the obtained column names are provided under DebugLevel=4 in Zabbix server log file:

```
$ grep db.odbc.discovery /tmp/zabbix_server.log
 ...
 23876:20150114:153410.856 In db_odbc_discovery() query:'SELECT h1.host, COUNT(h2.host) FROM hosts h1 L
 23876:20150114:153410.860 db_odbc_discovery() column[1]:'host'
 23876:20150114:153410.860 db_odbc_discovery() column[2]:'COUNT(h2.host)'
 23876:20150114:153410.860 End of db_odbc_discovery():NOTSUPPORTED
 23876:20150114:153410.860 Item [Zabbix server:db.odbc.discovery[proxies,{$DSN}]] error: Cannot convert
```

Now that we understand how a SQL query is transformed into a JSON object, we can use {#HOST} macro in item prototypes:



Once discovery is performed, an item will be created for each proxy:



Using db.odbc.get

Using `db.odbc.get[,{$DSN}]` and the following SQL example:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_ho
+---------+-------+
| host    | count |
+---------+-------+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+---------+-------+
3 rows in set (0.01 sec)
```

this JSON will be returned:

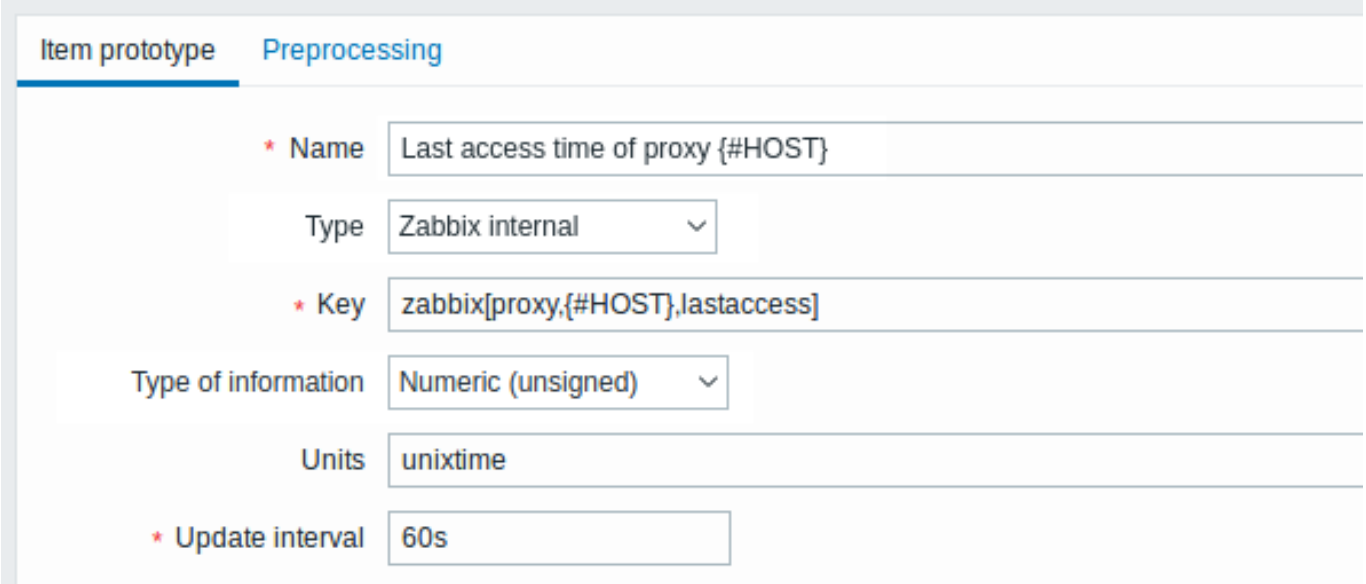
```
[
    {
        "host": "Japan 1",
```

```
        "count": "5"
    },
    {
        "host": "Japan 2",
        "count": "12"
    },
    {
        "host": "Latvia",
        "count": "3"
    }
]
```

As you can see, there are no low-level discovery macros there. However, custom low-level discovery macros can be created in the LLD macros tab of a discovery rule using JSONPath, for example:

`{#HOST}` → `$.host`

Now this `{#HOST}` macro may be used in item prototypes:



12 使用 **Prometheus** 数据发现

概述

Prometheus 提供的数据格式可以低级别发现使用

查看Prometheus 检查的详细数据是如何在 Zabbix 中实现的。

配置

低级发现规则应该作为监控项依赖创建到 Prometheus 数据的 HTTP 监控项中。

Prometheus 转 JSON

在发现规则中，进入 Preprocessing 选项卡，并选择 *Prometheus to JSON* Preprocessing 选项。发现需要 JSON 格式的数据，而 *Prometheus to JSON* 预处理选项将会返回正确的数据，包含以下属性:

- metric name
- metric value
- help (if present)
- type (if present)
- labels (if present)
- raw line

例如, 查询 `wmi_logical_disk_free_bytes`:

从 Prometheus 获取数据:

```
# HELP wmi_logical_disk_free_bytes Free space in bytes (LogicalDisk.PercentFreeSpace)
# TYPE wmi_logical_disk_free_bytes gauge
```

```
wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11
wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09
wmi_logical_disk_free_bytes{volume="HarddiskVolume4"} 4.59276288e+08
```

返回:


```
[
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "C:"
         },
        "value": "3.5180249088e+11",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"C:\"} 3.5180249088e+11"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "D:"
         },
        "value": "2.627731456e+09",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"D:\"} 2.627731456e+09"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "HarddiskVolume4"
         },
        "value": "4.59276288e+08",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"HarddiskVolume4\"} 4.59276288e+08"
    }
]
```

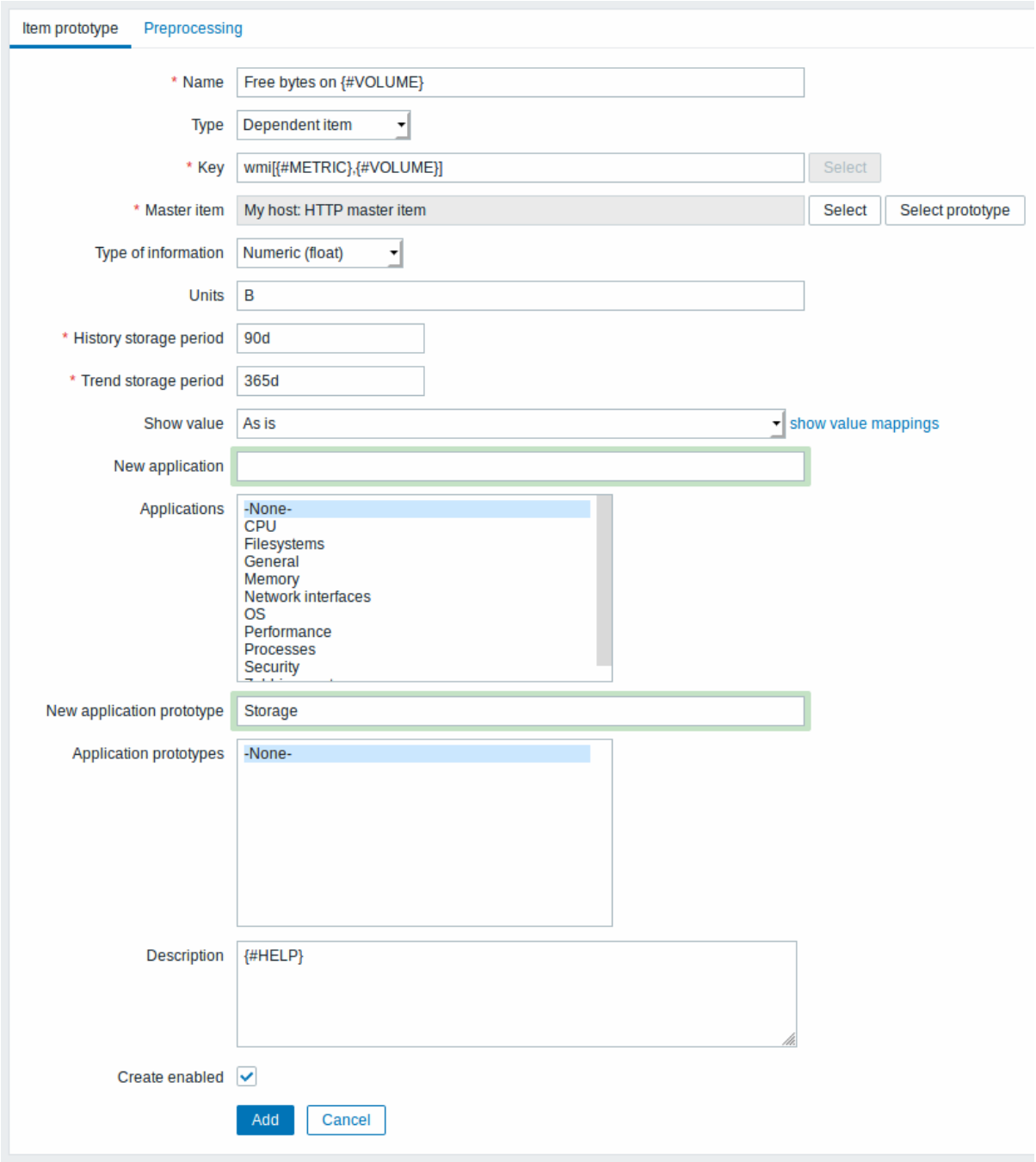

映射 LLD 宏

接下来，你必须进入 LLD 宏标签，并创建以下映射:

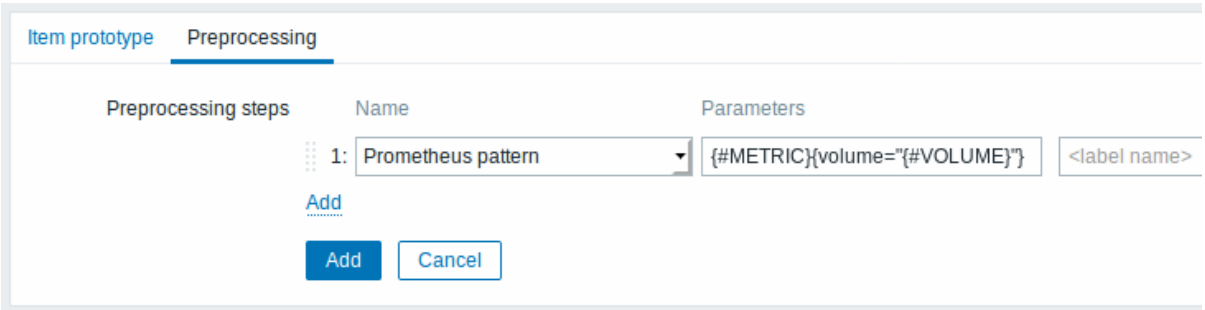
```
{#VOLUME}=$.labels['volume']
{#METRIC}=$['name']
{#HELP}=$['help']
```

监控项原型

你可能想要这样创建一个监控项原型:



与预处理选项:



13 发现块设备

与发现文件系统的方式相似，也可发现块设备及其设备类型。

监控项键值

在发现规则中监控项使用键值

vfs.dev.discovery

此监控项仅支持 Linux 平台，始于 Zabbix Agent 4.4。

可在发现监控项中创建如下的过滤规则：

- filter: **{#DEVNAME} matches** sd[\D]$ - 用于仅发现设备名如”sd0”, ”sd1”, ”sd2”, ...
- filter: **{#DEVTYPE} matches** disk **AND {#DEVNAME} does not match** ^loop.* - 用于发现类型名称为 ‘disk’ 且开头不是”loop”

支持的宏

此发现键值返回两个宏 - {#DEVNAME} 和 {#DEVTYPE} 分别用于标识块设备名及设备类型，例如:

```
[
   {
      "{#DEVNAME}":"loop1",
      "{#DEVTYPE}":"disk"
   },
   {
      "{#DEVNAME}":"dm-0",
      "{#DEVTYPE}":"disk"
   },
   {
      "{#DEVNAME}":"sda",
      "{#DEVTYPE}":"disk"
   },
   {
      "{#DEVNAME}":"sda1",
      "{#DEVTYPE}":"partition"
   }
]
```

发现块设备在创建监控项原型时，当键值为 vfs.dev.read[] 和 vfs.dev.write[] 时允许使用 {#DEVNAME} 宏, 例如:

- ”vfs.dev.read[{#DEVNAME},sps]”
- ”vfs.dev.write[{#DEVNAME},sps]”

{#DEVTYPE} 用于设备类型过滤。

14 发现 **Zabbix** 主机的接口

概述

可以发现 Zabbix 前端中为主机配置的所有接口。

监控项键值

在发现规则下监控项中使用

zabbix[host,discovery,interfaces]

进行内部监控. 此监控项从 Zabbix server 3.4 后支持.

监控项返回含接口描述的 JSON, 包括:

- IP 地址/DNS 域名 (取决于 “连接” 主机的设置)
- 端口号
- 接口类型 (Zabbix agent, SNMP, JMX, IPMI)
- 是否默认接口
- 是否支持批量请求 - 仅 SNMP 接口.

示例:

```
{"data":[{"{#IF.CONN}":"192.168.3.1","{#IF.IP}":"192.168.3.1","{#IF.DNS}":"","{#IF.PORT}":"10050","{#IF.TY
```

多个接口情况下 JSON 中接口记录排序方式为:

- 接口类型,
- 默认 - 默认接口在其他接口之前,
- 接口 ID (由小到大顺序).

支持的宏

下面的宏支持在发现规则下过滤 和监控原型, 触发器及图形中使用:

宏	述
{#IF.CONN}	接口的 IP 地址/DNS 主机名
{#IF.IP}	接口的 IP 地址
{#IF.DNS}	接口的 DNS 主机名
{#IF.PORT}	接口的端口号
{#IF.TYPE}	接口的类型 (”AGENT”, ”SNMP”, ”JMX”, or ”IPMI”).
{#IF.DEFAULT}	是否默认接口: 0 - 非默认接口 1 - 默认接口
{#IF.SNMP.BULK}	SNMP 接口是否允许批量请求: 0 - 禁止 1 - 支持 仅当接口类型为 “SNMP”.

自动发现（**LLD**）事项

发现应用集

应用集原型支持使用自动发现（LLD）宏。

应用集原型可以被同一发现规则下多个监控项型使用。

如果创建的应用集原型未由任何监控项原型使用，将自动从 “应用集原型” 列表中删除。

像其他发现实体一样，应用集遵循发现规则中定义的生存期（“保持丢失的资源期限” 设置）-在指定天数内未发现它们将被删除。

如果某个应用集所有发现的项目中不再发现，将自动从中删除，即使该应用集由于设置 “资源失效时间” 尚未删除。

由一个发现规则定义的应用集原型无法发现相同的应用集。在这种情况下，只有第一个原型发现将成功，其余的将报自动发现（LLD）错误。只有在不同发现规则中定义的应用程序原型才能导致发现同一应用集。

16. 分布式监控

概述　Zabbix 通过 Zabbix 代理为 IT 基础设施提供有效和可用的分布式监控

代理 (proxies) 可用于代替 Zabbix server 在本地收集数据，然后将数据报告给服务器。

Proxy 特性

当选择使用或不使用 proxy 时，必须考虑以下几个注意事项：

	代理 (Proxy)
轻量级（*Lightweight*） **Ye	**
图形界面（*GUI*） No	
独立工作（*Works independently*） **Yes	*
易于维护（*Easy maintenance*） **Yes	*
自动创建数据库（*Automatic DB creation*）[1] **Yes**	
本地管理（*Local administration*） No	
适用嵌入式硬件（*Ready for embedded hardware*） **Yes**	
单向 *TCP* 连接（*One way TCP connections*） **Yes	*
集中配置（*Centralised configuration*） **Yes	*
生成通知（*Generates notifications*） No	

Note:
[1] 自动创建数据库功能仅适用于 SQLite。其他数据库需要手动设置.

1 代理

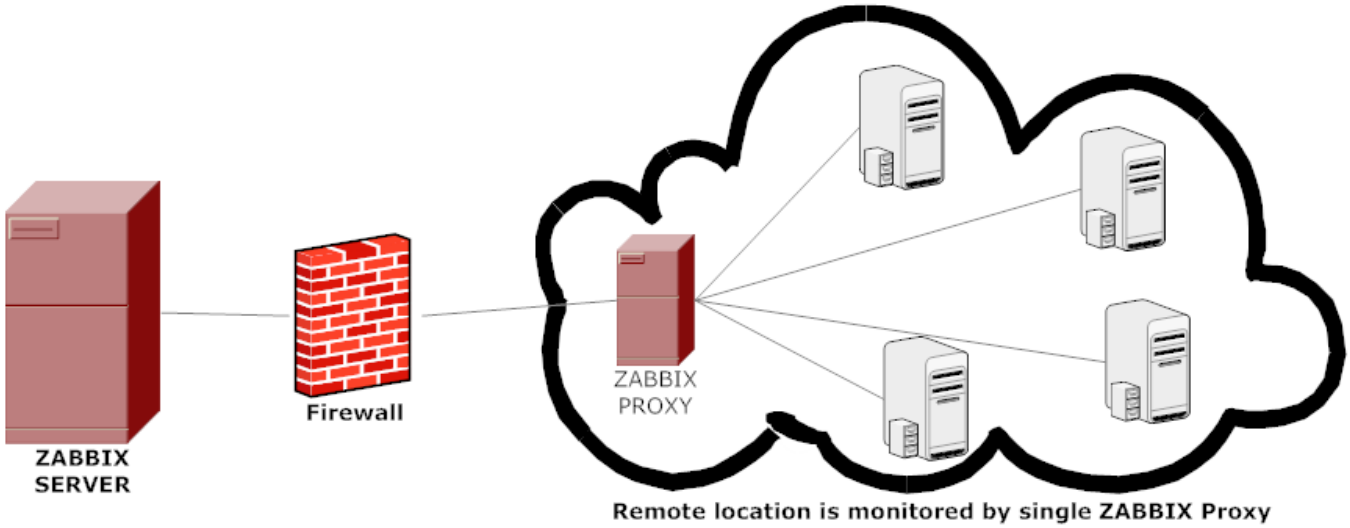
概述

Zabbix proxy 可以代替 Zabbix server 收集性能和可用性数据，承担一些收集数据的负担，分担了 Zabbix server 的负荷。

此外，使用 proxy 是实现集中式和分布式监控的最简单方法，所有 agents 和 proxies 发送给一个 Zabbix server，从而集中收集所有数据。

Zabbix proxy 使用场景:

- 监控远程区域设备
- 监控本地网络不稳定区域
- 监控上千设备时, 减轻 zabbix server 的负荷
- 简化分布式监控的维护



Zabbix proxy 到 Zabbix server 只需要一条 tcp 连接，仅在防火墙上配置一条规则即可。

Attention:
Zabbix proxy 数据库必须和 server 数据库分开，否则 Zabbix server 数据库会被破坏。

proxy 收集到数据都先存储在本地，然后在一定时间后传给 Zabbix server，这样就不会因为暂时无法连接 zabbix server 而丢失数据。本地保留时间由proxy 配置文件中参数 *ProxyLocalBuffer* 和 *ProxyOfflineBuffer* 决定。

Attention:
注意从 zabbix server 数据库直接更新最新配置的 proxy 可能会比 Zabbix server 更快生效。当 Zabbix server 由于设置缓存更新周期 的原因而无法快速更新时，proxy 收集发送到 Zabbix server 的数据可能会被忽略。

Zabbix proxy 只是一个数据收集器，不运行触发器、不处理事件、不发送报警。有关 proxy 功能详情，如下表：

功能	p	oxy 支持状态
监控项 (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**
	日志文件监控 *(Log file monitoring)* **Yes	*
	内部检查 *(Internal checks)* **Y	s**
	SSH 检查 *(SSH checks)* *	Yes**
	Telnet 检查 *(Telnet checks)* *	Yes**
	外部检查 *(External checks)* **Y	s**

功能	p	oxy 支持状态
	从属监控项 *(Dependent items)* **Ye	** [2]
内置 web 监控 (Built-in web monitoring)	**Y	s**
网络发现 (Network discovery)	**Y	s**
自动发现 (Low-level discovery)	**Y	s**
远程命令 (Remote commands)	**Y	s**
触发器计算 (Calculating triggers)	*No*	
处理事件 (Processing events)	*No	
事件关联 (Event correlation)	*No*	
发送报警 (Sending alerts)	*No	
监控项值的预处理 (Item value preprocessing)	*No*	

Note:
[1] 使用 agent 主动模式, 一定要记住在 agent 的配置文件参数 **ServerActive** 加上 proxy 的 IP 地址。

Note:
[2] Zabbix Server 对监控项值预处理时，需要先从主监控项获取到所需的数据。

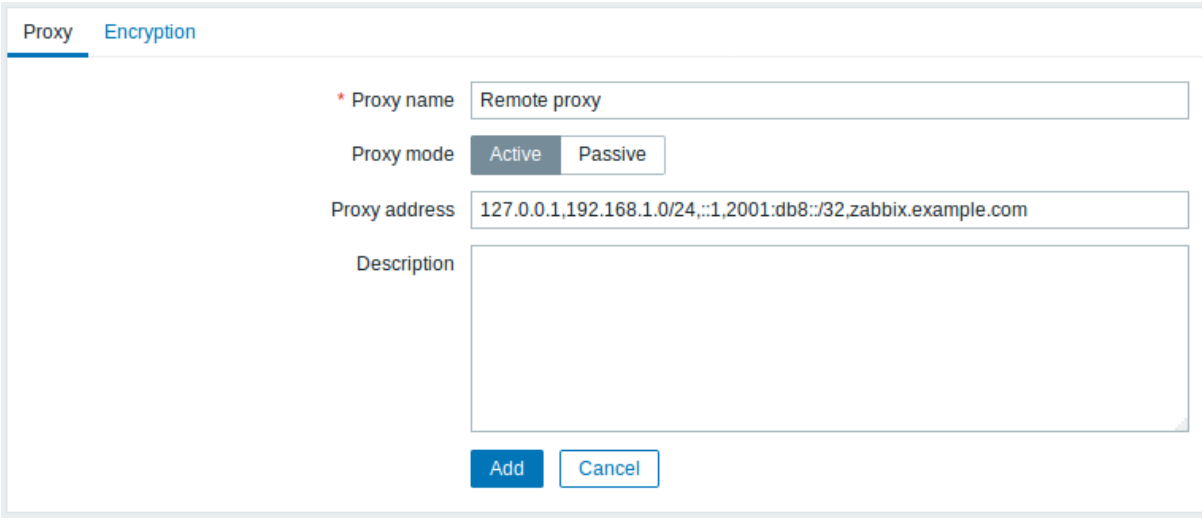
配置

安装并配置了一个 proxy 后，可在 Zabbix 前端进行设置。

添加代理

要在 Zabbix 前端配置代理：

- 转到：管理 → *agent* 代理程序
- 单击创建代理



参数	描	
代理名称 *(Proxy Name)*	pro	y 名称。它必须与 proxy 配置文件中的 Hostname 参数中的名称相同。

参数	描	
代理模式 *(Proxy mode)*	选择	roxy 运行模式 主动模式 **(Active)** - proxy 将连接到 Zabbix server 并请求配置数据 被动模式 **(Passive)** - Zabbix server 连接到 proxy 注意，当 proxy 使用主动模式时，访问 Zabbix server 的 trapper 端口可获取到未加密通信(敏感) proxy 配置数据。如果不进行身份验证或在代理地址限制 IP 范围，任何人都可以伪装成主动模式 proxy 并请求配置数据。
代理地址 *(Proxy address)*	设置仅	受指定 IP 地址、地址段、DNS 名的 proxy 主动发起的网络请求。\\仅在代理模式选择主动模式下生效，不支持宏(Macros)。此选项始于 Zabbix 4.0.0.
接口 *(Interface)*	被	模式 proxy 的接口详情。仅在代理模式选择被动模式下生效。
	IP 地址 *(IP Address)* 被	模式 proxy 的 IP (可选)。
	DNS 名 *(DNS Name)*	动模式 proxy 的 DNS 名 (可选)。

参数	描	
	连接到 *(Connect to)* 单选	钮，确定 Zabbix server 从 proxy 获取数据的途径: **IP** - 连接到指定 IP 的 proxy (推荐) **DNS** - 连接到指定 DNS 名的 proxy
	端口 *(Port)* 被	模式 proxy 使用 TCP/UDP 端口号 (默认 10051)。
描述	p	oxy 描述。

该 加密选项卡用于 proxy 的加密连接。

参数描	
连接代理服务器	接到被动代理的加密方式：非加密(默认)，共享秘钥(PSK)或证书。

参数	描
从代理连接主动模式	proxy 连接服务器的加密方式。可以同时选择几种连接类型(用于测试和切换到其他连接类型)。默认为“无加密”。

参数描	
发行者 *(Issuer)* 允许	发证书。证书首先通过CA(认证机构)验证。如果CA有效，则由CA签名，这时可以使用发行者字段来进一步限制允许的CA。该字段是可选的，如果Zab-bix安装使用多个CA

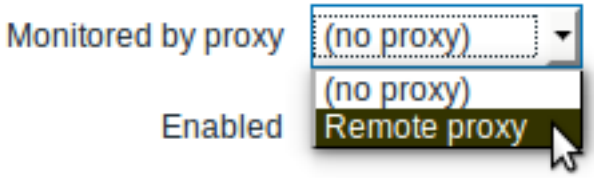
参数描
主体 *(Subject)* 允

的证书。证书首先通过CA验证。如果它有效，由CA签名，这时主体字段可以用于仅允许一个主体字符串值。如果此字段为空，则接受CA签名的任何有效证书。

参数	描
共享密钥一致性 *(PSK identity)* 共享密钥身份	符串
共享密钥（*PSK*）共享密钥 (	6 进 制)。 如 果 Zab- bix 使 用 mbed TLS (Po- larSSL) 库， 最 大 长 度 为 64 位 十 六 进 制 (32 字 节 PSK)； 如 果 Zab- bix 使 用 GnuTLS 或 OpenSSL 库， 最 大 长 度 为 512 位 十 六 进 制 数 (256 字 节 PSK)。 示 例： 1f87b595725ac

主机配置

您可以使用由 *agent* 代理程序监测字段指定主机配置表单中的 proxy 监控单个主机。



另一种配置指定主机由 proxy 监控的方式是选择主机批量更新。

17. 加密

概述　Zabbix 支持使用传输层安全（TLS）协议 v.1.2 在 Zabbix server，Zabbix proxy，Zabbix agent，zabbix_sender 和 zabbix_get 程序之间的加密通信。从 Zabbix 3.0 开始支持加密，支持基于证书和共享秘钥加密。

加密是可选的，可以针对各个组件分别配置（例如，一些 proxies 和 agents 可以配置为与服务器一起使用基于证书的加密，而其他可以使用共享秘钥加密，剩下的可以像以前一样继续使用未加密的通信）。

Zabbix server（proxy）可以为不同的主机使用不同的加密配置。

Zabbix 守护程序使用一个监听端口进行加密和未加密的传入连接。添加加密不需要在防火墙上开放新的端口。

限制

- 私钥以明文形式存储在 Zabbix 组件启动期间可读的文件中。
- 共享密钥在 Zabbix 前端输入，并以纯文本形式存储在 Zabbix 数据库中。
- 内置加密不保护如下通讯：

```
* 在运行Zabbix前端的Web服务器和用户Web浏览器之间
* 在Zabbix前端和Zabbix服务器之间，
* Zabbix服务器（代理）和Zabbix数据库之间。
* 目前每个加密的连接都会打开一个完整的TLS握手，没有实现会话缓存和凭据。
* 根据网络延迟，添加加密会增加检查和操作的时间。\\ 例如，如果分组延迟为100ms，则打开TCP连接并发送未加密的
* [[zh:manual/discovery/network_discovery|网络发现]]不支持加密。通过网络发现执行的Zabbix agent检查将是未加
```

编译 **Zabbix** 启用加密支持　为了支持加密 Zabbix 必须编译并链接到三个加密库之一：

- *mbed TLS*（以前的 *PolarSSL*）（版本 1.3.9 及更高版本 1.3.x）。*mbed TLS* 2.x 当前不支持，它不是 1.3 分支的替代替代，Zabbix 将不会使用 *mbed TLS* 2.x 进行编译。
- *GnuTLS*（3.1.18 版）
- *OpenSSL*（1.0.1 版）

通过指定 “configure” 脚本的选项来选择库：

- `--with-mbedtls[=DIR]`
- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]`

例如，要使用 *OpenSSL* 配置服务器和 agent 代理的源，可以使用以下内容：
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2 --with-openssl

可以使用不同的加密库（例如具有 *OpenSSL* 的服务器，具有 *GnuTLS* 的 agent 代理）来编译不同的 Zabbix 组件。

Attention:
如果您计划使用共享密钥（PSK），请考虑使用 PSKs 在 Zabbix 组件中使用 *GnuTLS* 或 *mbed TLS* 库。*GnuTLS* 和 *mbed TLS* 库支持具有Perfect Forward Secrecy的 PSK 密码。*OpenSSL* 库（版本 1.0.1，1.0.2c）支持 PSK，但可用的 PSK 密码套件不确保转发绝对保密。

加密连接管理　Zabbix 中的连接可以使用：

- 非加密（默认）
- 基于 RSA 证书的加密
- 基于 PSK 的加密

有两个重要参数用于为 Zabbix 组件之间的连接指定加密：

- TLSConnect
- TLSAccept

TLSConnect 指定要使用什么加密传出连接和可以采取 3 个中的一个（unencrypted，PSK，certificate）。TLSConnect 用于 Zabbix proxy 的配置文件（在主动模式下，仅指定与服务器的连接）和 Zabbix agentd（用于主动检查）。在的 zabbix 前端的 TLSConnect 等效物是连接主机在字段配置 → 主机 →< 一些主机 >→ 加密选项卡和连接代理字段中管理 → 代理 →< 一些代理 >→ 加密选项卡。如果配置的连接加密类型失败，则不会尝试其他加密类型。

TLSAccept 指定允许进入连接的连接类型。连接类型：unencrypted，PSK，certificate。可以指定一个或多个值。TLSAccept 用于 Zabbix proxy 的配置文件（在被动模式下，仅指定来自服务器的连接）和 Zabbix agentd（用于被动检查）。在的 zabbix 前端的 TLSAccept 等效物是从主机连接在字段配置 → 主机 →< 一些主机 >→ 加密选项卡和从连接代理在字段管理 → 代理 →< 一些代理 >→ 加密选项卡。

通常，您仅为传入加密配置一种类型的加密。但您可能希望切换加密类型，例如从加密到基于证书的最小停机时间和回滚可能性。
要实现这一点，您可以 TLSAccept=unencrypted,cert 在 agentd 配置文件中设置并重新启动 Zabbix agent。
然后，您可以 zabbix_get 使用证书测试与 agent 的连接。如果一切正常，你可以重新配置加密中的 zabbix 前端，agent 配置 → 主机 →< 某些主机 >→ 加密设置选项卡连接主机到 “证书”。
当服务器配置缓存被更新（如果主机正在通过 proxy 进行监视时，proxy 配置被更新），则与该 agent 的连接将被加密。
如果一切正常工作，您可以 TLSAccept=cert 在 agent 配置文件中设置并重新启动 Zabbix agent。
现在 agent 将只接受加密的基于证书的连接。未加密和基于 PSK 的连接将被拒绝

以类似的方式，它可以在服务器和 proxy 上运行。如果在 Zabbix 前端主机配置中连接设置为 “证书”，则只能从 agent（主动检查）和 zabbix_sender（trapper 项目）接受基于证书的加密连接。

很可能您将配置传入和传出连接使用相同的加密类型或根本不加密。但从技术上讲，可以非对称地进行配置，例如基于传入和基于 PSK 的出口连接的基于证书的加密。

有关概述，每个主机的加密配置将显示在 Zabbix 前端配置 → 主机右侧的代理加密列中。配置显示示例：

例	接到主机允许从主机	接拒绝从主机连接	
NONE	未加密未加	加密证书和	于 PSK 的证书
CERT NONE PSK CERT	加密，基于证书基于加密证书	未加密和基于 PSK 的	
PSK NONE PSK CERT	加密，基于 PSK 的加密 PSK	主未加密和基于证书	
PSK NONE PSK CERT	加密，基于 PSK 的未加密和基	PSK 的加密以证书为基础	
CERT NONE PSK CERT	加密，基于证书未加密，PS	或基于证书的加密 -	

Attention:
默认是未加密的连接。必须单独为每个主机和代理配置加密。

zabbix_get 和 zabbix_sender 使用加密　请参阅zabbix_get和zabbix_sender中使用加密的内容.

密码套件　在 Zabbix 启动期间内部配置了密码套件，并且依赖于加密库，目前用户不可配置。

按照从高到低顺序的库类型配置密码：

密码库证书	码 PSK 密码	
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_PS TLS_ECDHE_PS TLS_PSK_AES_1 TLS_PSK_AES_1 TLS_PSK_AES_1
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

密码库证书	码 PSK 密码
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

密码套件使用证书：

	TLS 服务端		
TLS 客户端 *m	ed TLS (PolarSSL)* *G	uTLS* *O	enSSL 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

密码套件使用 PSK:

	TLS 服务端		
TLS 客户端 *m	ed TLS (PolarSSL)* *G	uTLS* *O	enSSL 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

User-configured ciphersuites The built-in ciphersuite selection criteria can be overridden with user-configured ciphersuites.

> **Attention:**
> User-configured ciphersuites is a feature intended for advanced users who understand TLS ciphersuites, their security and consequences of mistakes, and who are comfortable with TLS troubleshooting.

The built-in ciphersuite selection criteria can be overridden using the following parameters:

Override scope	Parameter	Value	Description
Ciphersuite selection for certificates	TLSCipherCert13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	Certificate-based ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherCert	Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.	Certificate-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)
Ciphersuite selection for PSK	TLSCipherPSK13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	PSK-based ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherPSK	Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.	PSK-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)
Combined ciphersuite list for certificate and PSK	TLSCipherAll13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	Ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.

Override scope	Parameter	Value	Description
	TLSCipherAll	Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.	Ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)

To override the ciphersuite selection in zabbix_get and zabbix_sender utilities - use the command-line parameters:

- `--tls-cipher13`
- `--tls-cipher`

The new parameters are optional. If a parameter is not specified, the internal default value is used. If a parameter is defined it cannot be empty.

If the setting of a TLSCipher* value in the crypto library fails then the server, proxy or agent will not start and an error is logged.

It is important to understand when each parameter is applicable.

Outgoing connections

The simplest case is outgoing connections:

- For outgoing connections with certificate - use TLSCipherCert13 or TLSCipherCert
- For outgoing connections with PSK - use TLSCipherPSK13 and TLSCipherPSK
- In case of zabbix_get and zabbix_sender utilities the command-line parameters `--tls-cipher13` and `--tls-cipher` can be used (encryption is unambiguously specified with a `--tls-connect` parameter)

Incoming connections

It is a bit more complicated with incoming connections because rules are specific for components and configuration.

For Zabbix **agent**:

Agent connection setup	Cipher configuration
TLSConnect=cert	TLSCipherCert, TLSCipherCert13
TLSConnect=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert	TLSCipherCert, TLSCipherCert13
TLSAccept=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert,psk	TLSCipherAll, TLSCipherAll13

For Zabbix **server** and ** proxy**:

Connection setup	Cipher configuration
Outgoing connections using PSK	TLSCipherPSK, TLSCipherPSK13
Incoming connections using certificates	TLSCipherAll, TLSCipherAll13
Incoming connections using PSK if server has no certificate	TLSCipherPSK, TLSCipherPSK13
Incoming connections using PSK if server has certificate	TLSCipherAll, TLSCipherAll13

Some pattern can be seen in the two tables above:

- TLSCipherAll and TLSCipherAll13 can be specified only if a combined list of certificate- **and** PSK-based ciphersuites is used. There are two cases when it takes place: server (proxy) with a configured certificate (PSK ciphersuites are always configured on server, proxy if crypto library supports PSK), agent configured to accept both certificate- and PSK-based incoming connections
- in other cases TLSCipherCert* and/or TLSCipherPSK* are sufficient

The following tables show the `TLSCipher*` built-in default values. They could be a good starting point for your own custom values.

Parameter	GnuTLS 3.6.12
TLSCipherCert	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509
TLSCipherPSK	NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL
TLSCipherAll	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509

Parameter	OpenSSL 1.1.1d [1]
TLSCipherCert13	
TLSCipherCert	EECDH+aRSA+AES128:RSA+aRSA+AES128
TLSCipherPSK13	TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
TLSCipherPSK	kECDHEPSK+AES128:kPSK+AES128
TLSCipherAll13	
TLSCipherAll	EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128

[1] Default values are different for older OpenSSL versions (1.0.1, 1.0.2, 1.1.0), for LibreSSL and if OpenSSL is compiled without PSK support.

** Examples of user-configured ciphersuites **

See below the following examples of user-configured ciphersuites:

- Testing cipher strings and allowing only PFS ciphersuites
- Switching from AES128 to AES256

Testing cipher strings and allowing only PFS ciphersuites

To see which ciphersuites have been selected you need to set 'DebugLevel=4' in the configuration file, or use the `-vv` option for zabbix_sender.

Some experimenting with `TLSCipher*` parameters might be necessary before you get the desired ciphersuites. It is inconvenient to restart Zabbix server, proxy or agent multiple times just to tweak `TLSCipher*` parameters. More convenient options are using zabbix_sender or the `openssl` command. Let's show both.

1. Using zabbix_sender.

Let's make a test configuration file, for example /home/zabbix/test.conf, with the syntax of a zabbix_agentd.conf file:

```
Hostname=nonexisting
ServerActive=nonexisting

TLSConnect=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agent.psk
```

You need valid CA and agent certificates and PSK for this example. Adjust certificate and PSK file paths and names for your environment.

If you are not using certificates, but only PSK, you can make a simpler test file:

```
Hostname=nonexisting
ServerActive=nonexisting
```

```
TLSConnect=psk
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agentd.psk
```

The selected ciphersuites can be seen by running zabbix_sender (example compiled with OpenSSL 1.1.d):

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

Here you see the ciphersuites selected by default. These default values are chosen to ensure interoperability with Zabbix agents running on systems with older OpenSSL versions (from 1.0.1).

With newer systems you can choose to tighten security by allowing only a few ciphersuites, e.g. only ciphersuites with PFS (Perfect Forward Secrecy). Let's try to allow only ciphersuites with PFS using `TLSCipher*` parameters.

Attention:
The result will not be interoperable with systems using OpenSSL 1.0.1 and 1.0.2, if PSK is used. Certificate-based encryption should work.

Add two lines to the `test.conf` configuration file:

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
```

and test again:

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

The "certificate ciphersuites" and "PSK ciphersuites" lists have changed - they are shorter than before, only containing TLS 1.3 ciphersuites and TLS 1.2 ECDHE-* ciphersuites as expected.

2. TLSCipherAll and TLSCipherAll13 cannot be tested with zabbix_sender; they do not affect "certificate and PSK ciphersuites" value shown in the example above. To tweak TLSCipherAll and TLSCipherAll13 you need to experiment with the agent, proxy or server.

So, to allow only PFS ciphersuites you may need to add up to three parameters

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
TLSCipherAll=EECDH+aRSA+AES128:kECDHEPSK+AES128
```

to zabbix_agentd.conf, zabbix_proxy.conf and zabbix_server_conf if each of them has a configured certificate and agent has also PSK.

If your Zabbix environment uses only PSK-based encryption and no certificates, then only one:

```
TLSCipherPSK=kECDHEPSK+AES128
```

Now that you understand how it works you can test the ciphersuite selection even outside of Zabbix, with the `openssl` command. Let's test all three `TLSCipher*` parameter values:

```
$ openssl ciphers EECDH+aRSA+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
$ openssl ciphers kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 E
$ openssl ciphers EECDH+aRSA+AES128:kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
```

You may prefer `openssl ciphers` with option `-V` for a more verbose output:

```
$ openssl ciphers -V EECDH+aRSA+AES128:kECDHEPSK+AES128
           0x13,0x02 - TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
           0x13,0x03 - TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256
           0x13,0x01 - TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
           0xC0,0x2F - ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(128) Mac=AEAD
```

```
0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA256
0xC0,0x13 - ECDHE-RSA-AES128-SHA    TLSv1 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA1
0xC0,0x37 - ECDHE-PSK-AES128-CBC-SHA256 TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA256
0xC0,0x35 - ECDHE-PSK-AES128-CBC-SHA TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA1
```

Similarly, you can test the priority strings for GnuTLS:

```
$ gnutls-cli -l --priority=NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-A
Cipher suites for NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-ALL:+COMP-
TLS_ECDHE_RSA_AES_128_GCM_SHA256                        0xc0, 0x2f      TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA256                        0xc0, 0x27      TLS1.2

Protocols: VERS-TLS1.2
Ciphers: AES-128-GCM, AES-128-CBC
MACs: AEAD, SHA256
Key Exchange Algorithms: ECDHE-RSA
Groups: GROUP-SECP256R1, GROUP-SECP384R1, GROUP-SECP521R1, GROUP-X25519, GROUP-X448, GROUP-FFDHE2048, GR
PK-signatures: SIGN-RSA-SHA256, SIGN-RSA-PSS-SHA256, SIGN-RSA-PSS-RSAE-SHA256, SIGN-ECDSA-SHA256, SIGN-E
```

Switching from AES128 to AES256

Zabbix uses AES128 as the built-in default for data. Let's assume you are using certificates and want to switch to AES256, on OpenSSL 1.1.1.

This can be achieved by adding the respective parameters in `zabbix_server.conf`:

```
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/server.crt
TLSKeyFile=/home/zabbix/server.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
TLSCipherPSK13=TLS_CHACHA20_POLY1305_SHA256
TLSCipherPSK=kECDHEPSK+AES256:-SHA1
TLSCipherAll13=TLS_AES_256_GCM_SHA384
TLSCipherAll=EECDH+aRSA+AES256:-SHA1:-SHA384
```

Attention:
Although only certificate-related ciphersuites will be used, `TLSCipherPSK*` parameters are defined as well to avoid their default values which include less secure ciphers for wider interoperability. PSK ciphersuites cannot be completely disabled on server/proxy.

And in `zabbix_agentd.conf`:

```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
```

1 使用证书

Overview

Zabbix 可以使用 PEM 格式的 RSA 证书，由公共或内部认证机构（CA）签名。根据预先配置的 CA 证书进行证书验证。不支持自签名证书。可以选择使用证书撤销列表（CRL）。每个 Zabbix 组件只能配置一个证书。

有关如何设置和操作内部 CA 的更多信息，如何生成证书请求并签名，如何撤销证书，您可以找到许多在线操作，例如OpenSSL PKI Tutorial v1.1。

仔细考虑和测试证书扩展 - 请参阅使用 X.509 v3 证书扩展的限制.

证书配置参数

参数必	描述
TLSCAFile	* 包含用于对等证书验证的顶级CA证书的文件的完整路径名。在具有多个成员的证书链的情况下，它们必须被排序：较低级别的CA证书，然后是较高级别的CA证

参数必	描述
TLSCRLFile	包含证书吊销列表的文件的完整路径名。见证书吊销清单(CRL)中的注释。Certificate Revocation Lists (CRL).

参数必	描述
TLSCertFile	* 包含证书(证书链)的文件的完整路径名。设置此文件的访问权限 - 它必须只能由 Zabbix 用户读取。在具有多个成员的证书链的情况下，必须首先对其进行排序：服

参数必	描述
TLSKeyFile	* 包含私钥的文件的完整路径名。设置此文件的访问权限 - 它必须只能由 Zabbix 用户读取。
TLSServerCertIssuer	允许的服务器证书发行者 (issuer)。

参数必	描述
TLSServerCertSubject	允许的服务器证书主体 (subject)。

在 Zabbix server 上配置证书

1. 为了验证对等证书，Zabbix server 必须具有使用其顶级自签名根 CA 证书的文件访问权限。例如，如果我们期望来自两个独立根 CA 的证书，我们可以将其证书放入文件中 /home/zabbix/zabbix_ca_file：

```
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. 将 Zabbix 服务器证书链放入文件中，例如/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-----
```

先放入 Zabbix server 证书，随后是中间 CA 的证书。

3. 将 Zabbix server 私钥放入文件中，例如/home/zabbix/zabbix_server.key：

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXDl
...
IJLkhbybBYEf47MLhffWa7XvZTY=
-----END PRIVATE KEY-----
```

4. 在 Zabbix server 配置文件中编辑 TLS 参数，如下所示：

```
TLSCAFile=/home/zabbix/zabbix_ca_file
```

```
TLSCertFile=/home/zabbix/zabbix_server.crt
TLSKeyFile=/home/zabbix/zabbix_server.key
```

Zabbix proxy 配置基于证书的加密

1. 使用顶级 CA 证书，proxy 证书(链)和私钥准备文件，如在Zabbix server 上配置证书中所述。编辑参数 `TLSCAFile`，`TLSCertFile`，`TLSKeyFile` 在 proxy 配置相应。

2. 对于 proxy 代理编辑 `TLSConnect` 参数：

```
TLSConnect=cert
```

对于被动 proxy 编辑 `TLSAccept` 参数：

```
TLSAccept=cert
```

3. 现在你有一个基于证书的最小 proxy 配置。您可能希望通过设置 `TLSServerCertIssuer` 和 `TLSServerCertSubject` 参数来提高 proxy 安全性（请参阅限制允许的证书发行者和主体）。

4. 在最终的 proxy 配置文件中，TLS 参数可能如下所示：

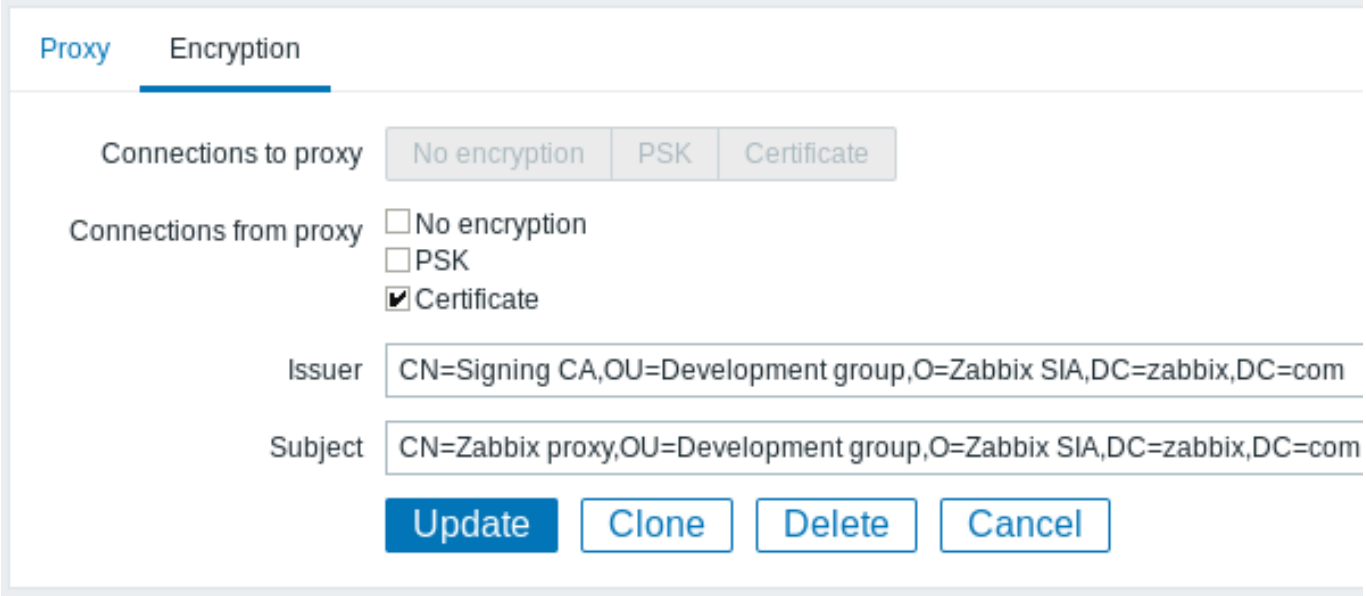
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_proxy.crt
TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

5. 在 Zabbix 前端配置此 proxy 的加密：

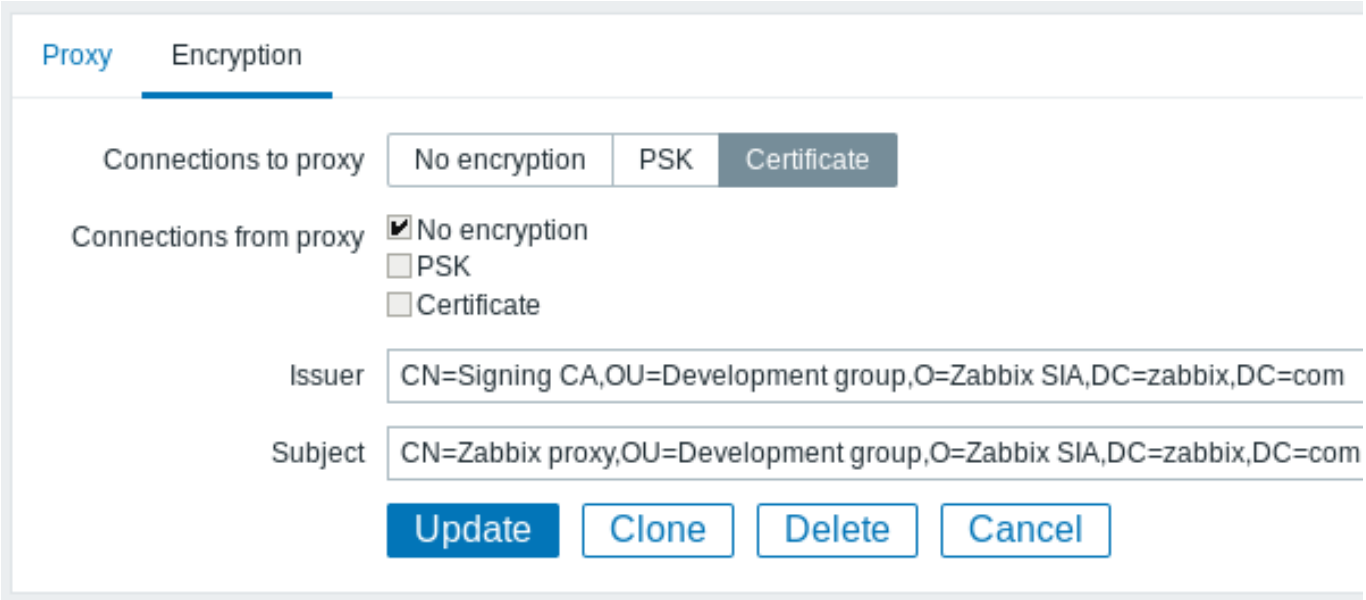
- 转到：管理 →*agent* 代理程序 *(proxies)*
- 选择代理，然后单击加密选项卡

在下面的示例中，发行者 (Issuer) 和主体 (fields) 字段填写 - 请参阅 [zh:manual:encryption/using_certificates#restricting_allowed_certificate_issu
限制允许的证书发行者和主体]] 为什么以及如何使用这些字段。

对于主动 proxy



对于被动 proxy



Zabbix agent 配置基于证书的加密

1. 使用顶级 CA 证书，代理证书（链）和私钥准备文件，如在Zabbix server 配置证书中所述。编辑参数 `TLSCAFile`，`TLSCertFile`，`TLSKeyFile` 在 agent 配置相应。

2. 对于主动检查编辑 `TLSConnect` 参数：

```
TLSConnect=cert
```

对于被动检查编辑 `TLSAccept` 参数：

```
TLSAccept=cert
```

3. 现在，您有一个基于证书的最小 agent 配置。您可能希望通过设置 `TLSServerCertIssuer` 和 `TLSServerCertSubject` 参数提高 agent 安全性。（请参阅限制允许的证书发行者和主体）。

4. 在最终 agent 配置文件中，TLS 参数可能如下所示：

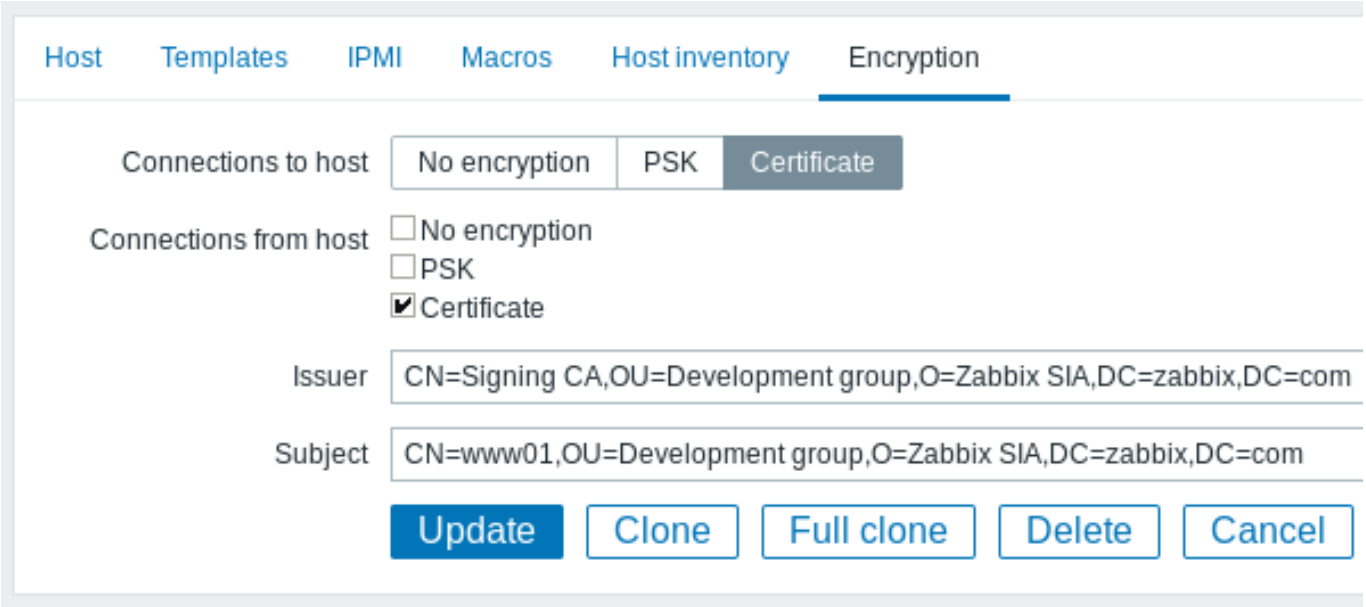
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_agentd.crt
TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

（例如，假设主机是通过 proxy 监视的，因此是 proxy 证书主体。）

5. 在 Zabbix 前端为此 agent 配置加密：

- Go to: *Configuration* → *Hosts*
- 转到：配置 → 主机
- Select host and click on **Encryption** tab
- 选择主机，然后单击加密选项卡

在下面的示例中，发行者和主体字段填写 - 请参阅限制允许的证书发行者和主体为什么以及如何使用这些字段。



限制允许的证书发行者和主体

当两个 Zabbix 组件（例如服务端和 agent）建立 TLS 连接时，他们会检查对方的证书。如果对等证书由受信任的 CA（具有预先配置的顶级证书 TLSCAFile）签名有效，尚未过期且通过其他检查项，则可以进行通信。在最简单的情况下，不会检查证书发行者和主体。

这存在一个风险 - 任何拥有有效证书的人都可以冒充任何人（例如，主机证书可以用来模拟服务器）。在内部 CA 签发证书的小型环境中，这种风险可能是可以接受的，冒充的风险较低。

如果您的顶级 CA 用于签发其他证书而不应被 Zabbix 接受，或者你想降低冒充风险，您可以通过指定其发行者 (Issuer) 和主体 (Subject) 字符串来限制允许的证书。

例如，您可以在 Zabbix proxy 配置文件中写：

```
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
```

通过这些设置，主动 proxy 将不会与证书中具有不同发行者或主体字符串的 Zabbix server 通信，被动 proxy 将不接受来自此类服务器的请求。

有关发行者或主体字符串匹配的说明：

1. 独立检查发行者和主体字符串。两者都是可选的。
2. 允许使用 UTF-8 字符。
3. 未指定的字符串等同于任何字符串都被接受。
4. 字符串按 “原样” 比较，它们必须完全一致才能匹配。
5. 不支持通配符和正则表达式。
6. 只有RFC 4514 轻量级目录访问协议（LDAP）的一些要求：实现了可分辨名称的字符串表示：
   - 转义字符'"' (U+0022), '+' U+002B, ',' U+002C, ';' U+003B, '<' U+003C, '>' U+003E, '\' U+005C 在字符串
   - 字符串开头处的转义字符空格(' ' U+0020) 或数字符号 ('#' U+0023)。
   - 字符串末尾的转义字符空间(' ' U+0020) 。
- 如果遇到空字符(U+0000)（[[http://tools.ietf.org/html/rfc4514|RFC 4514]]允许），则匹配失败。
- 要求[[http://tools.ietf.org/html/rfc4517|RFC 4517轻量级目录访问协议（LDAP）：句法和匹配规则]]和 [[http:/

发行者和主体的内容格式及字段顺序很重要！Zabbix 遵循 RFC 4514建议，并使用 “反向” 字段顺序。

反向顺序可以举例说明：

```
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
```

请注意，它以低位（CN）开始，进入中级（OU，O）并以顶级（DC）字段结束。

默认情况下，*OpenSSL* 将以 “正常” 的顺序显示证书发行者和主体字段，具体取决于使用的其他选项：

```
$ 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
```

这里发行者和主体字符串从顶级（DC）开始，以低级（CN）字段结尾，空格和字段分隔符取决于所使用的选项。这些值都不会匹配 Zabbix 发行者（Issuer）和主体（Subject）字段！

Attention:
要获得适当的发行者和主体字符串可用于 Zabbix 使用特殊选项调用\\OpenSSL
`-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
```

现在字段是反序，字段是逗号分隔的，可以在 Zabbix 配置文件和前端使用。

使用 X.509 v3 证书扩展的限制

- 主体备用名称（***subjectAltName***）扩展名。
  Zabbix 不支持来自 *subjectAltName* 扩展名的替代主体名称（如 IP 地址，电子邮件地址）。只能在 Zabbix 中检查 “主体” 字段的值（请参阅限制允许的证书发行者和主体）。
  如果证书使用 *subjectAltName* 扩展名，那么结果取决于加密工具包的特定组合。Zabbix 组件被编译（可能工作或不工作，Zabbix 可能拒绝接受来自对等体的证书）
- 扩展密钥使用扩展。
  如果使用，则通常需要 *clientAuth*（TLS WWW 客户端身份验证）和 *serverAuth*（TLS WWW 服务器身份验证）。
  例如，被动检查的 zabbix agent 是作为 TLS 服务器，所以 *serverAuth* 必须在 agent 证书设置。对于主动检查 agent 证书需要 *clientAuth* 进行设置。
  GnuTLS 在违规使用情况下发出警告，但允许通信进行。
- 名称限制扩展。
  并不是所有的加密工具包都支持它。此扩展可能会阻止 Zabbix 加载 CA 证书，此部分被标记为关键 *(critical)*（取决于特定的加密工具包）。

证书撤销清单（CRL）

如果证书受到威胁，CA 可以通过在 CRL 中包含来撤销证书。可以在 server 器，proxy 和 agent 的配置文件中配置 CRL TLSCRLFile。例如：

```
TLSCRLFile=/home/zabbix/zabbix_crl_file
```

where `zabbix_crl_file` may contain CRLs from several CAs and look like:

`zabbix_crl_file` 可能包含几个 CA 的 CRL，如下所示

```
-----BEGIN X509 CRL-----
MIIB/DCB5QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv
...
treZeUPjb7LSmZ3K2hpbZN7SoOZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t
...
CAEebS2CND3ShBedZ8YSil59O6JvaDP61lR5lNs=
-----END X509 CRL-----
```

CRL 文件仅在 Zabbix 启动时加载。CRL 更新需要重新启动。

Attention:
如果使用 *OpenSSL* 编译 Zabbix 组件，要使用 CRL，则证书链中的每个顶级和中级 CA 都必须具有相应的 CRL（可以为空）`TLSCRLFile`。

使用 CRL 扩展的限制

- 权限密钥标识符扩展。
  具有相同名称的 CA 的 CRL 在 *mbedTLS*（*PolarSSL*）的情况下可能不起作用，即使使用 “权限密钥标识符” 扩展。

2 使用共享密钥

概述

Zabbix 中的每个共享密钥（PSK）实际上是一对：

- 非私密 PSK identity（共享密钥一致性）字符串，
- 私密 PSK 字符串值。

PSK identity（共享密钥一致性）字符串是非空的 UTF-8 字符串。

例如，“PSK ID 001 Zabbix agentd”。这是一个独特的名称，由 Zabbix 组件引用该特定的 PSK。不要将敏感信息放在 PSK identity（共享密钥一致性）字符串中 - 它通过未加密网络传输。

PSK 值通常是很难猜出十六进制数字的字符串，例如 “e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9”。

长度限制

在 Zabbix 中有 PSK identity（共享密钥一致性）和 PSK 值的长度限制，在某些情况下，加密库可以有下限：

组件 P	K identity 长度上限 PSK 值长	下限 PSK 值长度上限	
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)
Zabbix	128 个 UTF-8 字符 12	位（16 字节 PSK，输入 32 位十六进制数字）2048 位（256 字节 PSK，输入	12 个十六进制数字）
GnuTLS	128 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)

组件 P	K identity 长度上限 PSK 值长	下限 PSK 值长度上限	
GnuTLS	128 字节（可能包括 UTF-8 字符）-	2048 位（256	节 PSK，输入 512 个十六进制数字）
mbed TLS (PolarSSL)	128 UTF-8 characters	-	256-bit (default limit) (32-byte PSK, entered as 64 hexadecimal digits)
mbed TLS (PolarSSL)	128 个 UTF-8 字符 -	25	位 (默认限制)(32 字节 PSK，以 64 位十六进制数字输入）

组件 P	K identity 长度上限 PSK 值长	下限 PSK 值长度上限	
OpenSSL	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
OpenSSL	127 字节（可能包括 UTF-8 字符）-	2048 位（256	节 PSK，输入 512 个十六进制数字）

Attention:
Zabbix 前端允许配置多达 128 个字符的长的 PSK identity（共享密钥一致性）字符串和 2048 位长的 PSK，而不管使用的加密库。
如果某些 Zabbix 组件支持较低限制，则用户有责任为这些组件配置 PSK identity（共享密钥一致性）和 PSK 值。
超出长度限制会导致 Zabbix 组件之间的通信故障。

在 Zabbix server 使用 PSK 连接到 agent 之前，服务器将查找数据库中为该 agent 配置的 PSK identity（共享密钥一致性）和 PSK 值（实际上在配置缓存中）。agent 在收到连接后，从其配置文件中读取 PSK identity（共享密钥一致性）和 PSK 值。如果双方具有相同的 PSK identity（共享密钥一致性）字符串和 PSK 值，则连接才可能会成功。

Attention:
用户有责任确保每个 PSK identity（共享密钥一致性）字符串只对应唯一的 PSK。否则可能会导致使用 PSK 和 PSK identity（共享密钥一致性）字符串的 Zabbix 组件之间的通信被中断。

生成 PSK

例如，可以使用以下命令生成 256 位（32 字节）PSK：

- with *OpenSSL*:
- 使用 *OpenSSL*：

```
$ openssl rand -hex 32
af8ced32dfe8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
```

- 使用 *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
```

请注意，上面"psktool" 命令产生 PSK 值的数据库文件。Zabbix 只需要 PSK 文件中的 PSK，因此应该从文件中删除'psk_identity:'。

配置 PSK 进行服务器和 agent 通信（示例）

在 agent 主机上，将 PSK 值写入文件，例如/home/zabbix/zabbix_agentd.psk。该文件必须在第一个文本字符串中包含 PSK，例如：

```
1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

设置 PSK 文件的访问权限 - 它必须只能由 Zabbix 用户读取。

在 agent 配置文件 zabbix_agentd.conf 中编辑 TLS 参数，例如 set：

```
TLSConnect=psk
TLSAccept=psk
TLSPSKFile=/home/zabbix/zabbix_agentd.psk
TLSPSKIdentity=PSK 001
```

agent 将连接到服务器（主动检查）并接受来自服务器和 `zabbix_get` 使用 PSK 连接。PSK 身份将是 "PSK 001"。

重新启动 agent。现在可以使用 zabbix_get 例如：

```
$ 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
```

(为了最大限度地减少停机时间看看如何改变连接方式加密连接管理).

在 Zabbix 前端为此 agen 配置 PSK 加密：

- 转到：配置 → 主机
- 选择主机，然后单击加密选项卡

例如：

带红色星号标识的字段均为必填项。

当配置缓存与数据库同步时，新连接将使用 PSK。检查服务器和 agent 日志文件可查看错误消息。

配置 PSK 服务 - proxy 主动模式 (示例)

在 proxy 上，将 PSK 值写入文件，例如/home/zabbix/zabbix_proxy.psk。该文件必须在第一个文本字符串中包含 PSK，例如：

```
e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9
```

设置 PSK 文件的访问权限 - 仅 Zabbix 用户可读取。

在 proxy 配置文件 zabbix_proxy.conf 中编辑 TLS 参数，设置示例：

```
TLSConnect=psk
TLSPSKFile=/home/zabbix/zabbix_proxy.psk
TLSPSKIdentity=PSK 002
```

oxy 将使用 PSK 连接到服务器。PSK identity（共享密钥一致性）将是 "PSK 002"。

(为了最大限度地减少停机时间看看如何改变连接方式加密连接管理).

在 Zabbix 前端配置此 proxy 的 PSK。转到管理 →*agent* 代理程序，选择代理，转到 “加密” 选项卡。在 “从代理连接” 勾选 PSK。将 “PSK identity(共享密钥一致性)” 字段填上 “PSK 002”，“共享密钥(PSK)“字段填上”e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327b
点击 “更新”。

重新启动 proxy。它将开始使用基于 PSK 的加密连接到服务器。检查服务器和 proxy 日志文件查看错误消息。

对于被动 proxy，该过程非常相似。唯一的区别 - TLSAccept=psk 在 proxy 配置文件中设置并在 Zabbix 前端设置 “连接代理” PSK。

3 排错

一般建议

- 从理解开始，哪个组件充当 TLS 客户端，哪个组件充当问题的 TLS 服务器。
  Zabbix server, proxies and agents，取决于它们之间的交互，都可以作为 TLS 服务器和客户端。
  例如，Zabbix server 连接到 agent 进行被动检查，充当 TLS 客户端。该 agent 是 TLS 服务器的角色。
  Zabbix agent，请求从 proxy 的主动检查列表，充当 TLS 客户端。prox 服务器是 TLS 服务器。
  `zabbix_get` 并且 `zabbix_sender` 程序始终作为 TLS 客户端。

* Zabbix 使用相互验证。
每一方验证对方，并可拒绝连接。
例如，如果 agent 的证书无效，则连接到 agent 的 Zabbix server 可以立即关闭连接。反之亦然 - 如果服务器不被 agent 信任，Zabbix agent 可关闭来自服务器的连接。

- 检查双方的日志文件 - 在 TLS 客户端和 TLS 服务器中。
  拒绝连接的一方可能会记录为什么被拒绝的准确理由。其他方面经常报告相当普遍的错误（例如”Connection closed by peer”, ”connection was non-properly terminated”）。
- 有时配置错误的加密会导致混淆的错误消息，而不会指向真正的原因。
  在下面的小节中，我们尝试提供一个（简单的）的消息收集和可能有助于故障排除的可能原因。
  请注意，不同的加密工具包（OpenSSL，GnuTLS，mbed TLS（PolarSSL））在相同的问题情况下经常产生不同的错误消息。
  有时错误消息甚至依赖于两端的密码工具包的特定组合。

1 连接类型或权限问题

服务器配置为与 agent 程序连接，但 agent 仅接受未加密的连接

在服务器或 proxy 日志（带有 *mbed TLS*（*PolarSSL*）1.3.11）

```
Get value from agent failed: ssl_handshake(): SSL - The connection indicated an EOF
```

在服务器或 proxy 日志中（使用 *GnuTLS* 3.3.16）

```
Get value from agent failed: zbx_tls_connect(): gnutls_handshake() failed: \
    -110 The TLS connection was non-properly terminated.
```

在服务器或 proxy 日志中（使用 *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
```

一方连接证书，但另一方只接受 PSK，反之亦然

在任意日志中（使用 *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
```

在任意日志中（使用 *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.
```

在任意日志中（使用 *OpenSSL* 1.0.2c）：

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake returned error code 1:\
    file .\ssl\s3_srvr.c line 1411: error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher:\
    TLS write fatal alert "handshake failure"
```

Attempting to use Zabbix sender compiled with TLS support to send data to Zabbix server/proxy compiled without TLS

In connecting-side log:

Linux:

```
...In zbx_tls_init_child()
...OpenSSL library (version OpenSSL 1.1.1  11 Sep 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...End of zbx_tls_connect():FAIL error:'connection closed by peer'
...send value error: TCP successful, cannot establish TLS to [[localhost]:10051]: connection closed by pee
```

Windows:

```
...OpenSSL library (version OpenSSL 1.1.1a  20 Nov 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...zbx_psk_client_cb() requested PSK identity "PSK test sender"
...End of zbx_tls_connect():FAIL error:'SSL_connect() I/O error: [0x00000000] The operation completed succ
...send value error: TCP successful, cannot establish TLS to [[192.168.1.2]:10051]: SSL_connect() I/O erro
```

In accepting-side log:

```
...failed to accept an incoming connection: from 127.0.0.1: support for TLS was not compiled in
```

One side connects with PSK but other side uses LibreSSL or has been compiled without encryption support

LibreSSL does not support PSK.

In connecting-side log:

```
...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() I/O error: [0] Success
```

In accepting-side log:

```
...failed to accept an incoming connection: from 192.168.1.2: support for PSK was not compiled in
```

In Zabbix frontend:

```
Get value from agent failed: TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect()
```

One side connects with PSK but other side uses OpenSSL with PSK support disabled

In connecting-side log:

```
...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() set result code to SSL_ERR
```

In accepting-side log:

```
...failed to accept an incoming connection: from 192.168.1.2: TLS handshake set result code to 1: file ssl
```

2 证书问题

OpenSSL 与 CRL 一起使用，对于证书链中的某些 CA，其 CRL 不在 “TLSCRLFile” 中

在 *mbed TLS (PolarSSL)* 和 *OpenSSL* 对等（peers）情形 TLS 服务器日志:

```
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"
```

在 *GnuTLS* 对等（peer）情形 TLS 服务器日志:

```
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
```

服务器运行期间 CRL 过期或到期

OpenSSL 在服务器端日志:

- 过期前:

```
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"
```

- 过期后:

```
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"
```

需要指出的是，有效的 CRL 撤销证书时，被告知为 “证书撤销”。当 CRL 到期时，错误消息将更改为 “证书已过期”，这是非常容易误导的。

GnuTLS 在服务器日志:

- 过期前或过期后:

```
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:

- 过期前:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    invalid peer certificate: revoked
```

- 过期后:

```
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
```

Self-signed certificate, unknown CA

OpenSSL, in log:

```
error:'self signed certificate: SSL_connect() set result code to SSL_ERROR_SSL: file ../ssl/statem/statem_
      line 1924: error:1416F086:SSL routines:tls_process_server_certificate:certificate verify failed:\
      TLS write fatal alert "unknown CA"'
```

This was observed when server certificate by mistake had the same Issuer and Subject string, although it was signed by CA. Issuer and Subject are equal in top-level CA certificate, but they cannot be equal in server certificate. (The same applies to proxy and agent certificates.)

3 PSK 问题

PSK 对应的 16 进制字符数为奇数

Proxy 或 agent 未启动，在 proxy 或 agent 日志中有:

```
invalid PSK in file "/home/zabbix/zabbix_proxy.psk"
```

超过 128 字节的 PSK identity（共享密钥一致性）字符串传递到 GnuTLS

在 TLS 客户端日志:

```
gnutls_handshake() failed: -110 The TLS connection was non-properly terminated.
```

在 TLS 服务端日志:

```
gnutls_handshake() failed: -90 The SRP username supplied is illegal.
```

超过 32 个字节的 PSK 传递到 mbed TLS (PolarSSL)

在任何 Zabbix 日志:

```
ssl_set_psk(): SSL - Bad input parameters to function
```

18. Web 界面

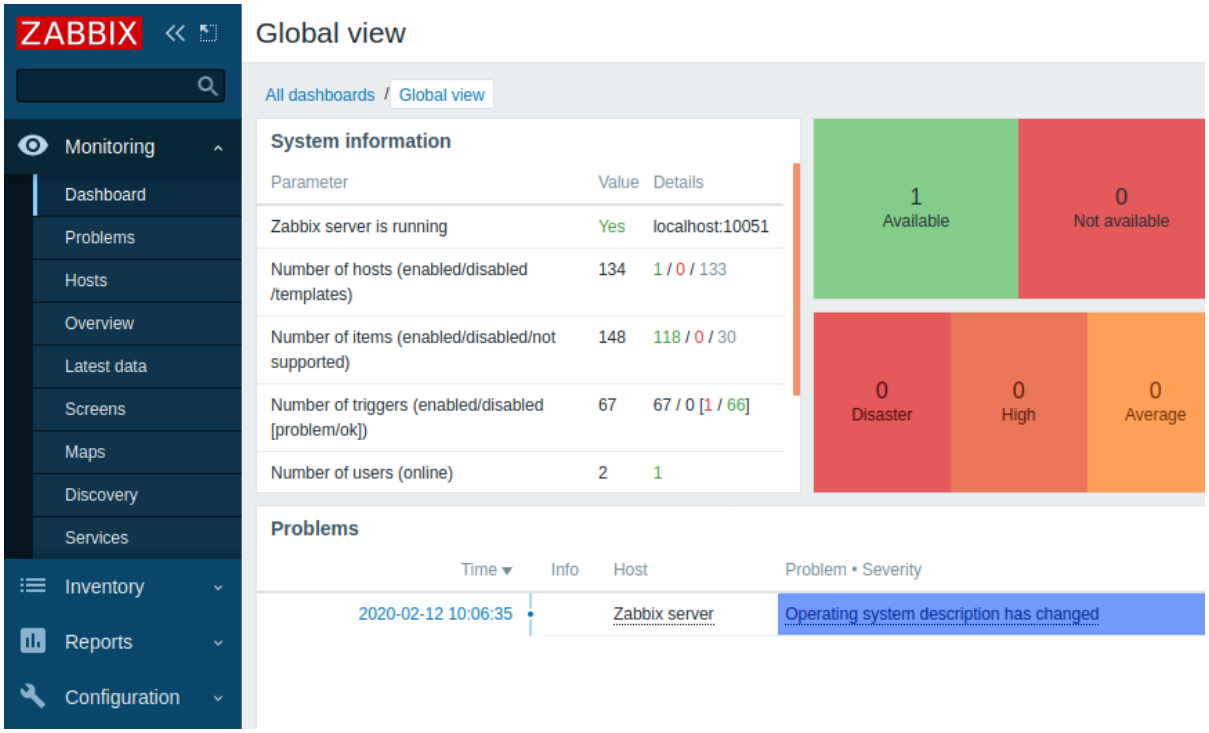
概览　为了能从任何地方和任何平台轻松访问 Zabbix，Zabbix 提供了基于 Web 的界面。

Note:
尝试同时访问安装在同一个主机上，不同端口的两个 Zabbix 前端会失败。登录第二个会话将终止第一个会话——除非在前端定义中，为第二个前端调整了默认的前端会话名称 (see ZBX_SESSION_NAME)。

1 菜单

概览

侧边栏的垂直菜单可访问 Zabbix 前端各个部分。菜单默认使用深蓝主题。

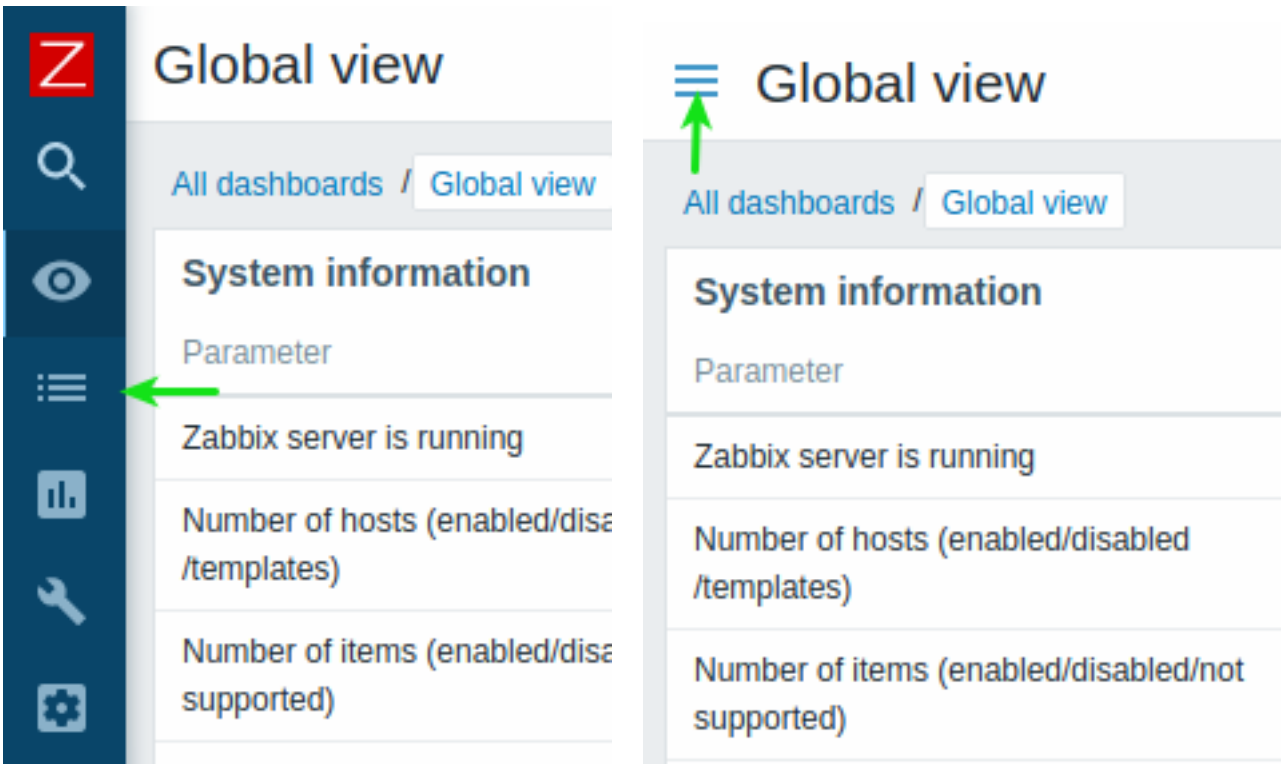


使用菜单

全局搜索 在 Zabbix 标识下方。

菜单可以整个折叠或隐藏：

- 折叠，单击 Zabbix logo 旁边的
- 隐藏，单击 Zabbix logo 旁边的



菜单折叠后仅显示图标. 隐藏菜单.

折叠菜单

当菜单折叠为图标时，将鼠标光标放在菜单上，就会重新显示完整菜单。需要注意菜单只是重新浮在页面内容上；要将页面内容移至右侧，您必须单击展开按钮。如果再次将鼠标光标置于整个菜单之外，则该菜单将在两秒钟后再次折叠。

你还可以通过按 Tab 键使完全折叠的菜单重新出现，反复按 Tab 键可切换到下一个菜单元素上。

隐藏菜单

即使菜单被完全隐藏，只需要通过鼠标单击汉堡图标即可获得完整的菜单。需要注意的是它只是重新浮现在页面内容上；要将页面内容移至右侧，您必须通过单击显示侧边栏按钮来取消隐藏菜单。

2 前端

1 监测

概览

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。

View mode buttons

The following buttons located in the top right corner are common for every section:

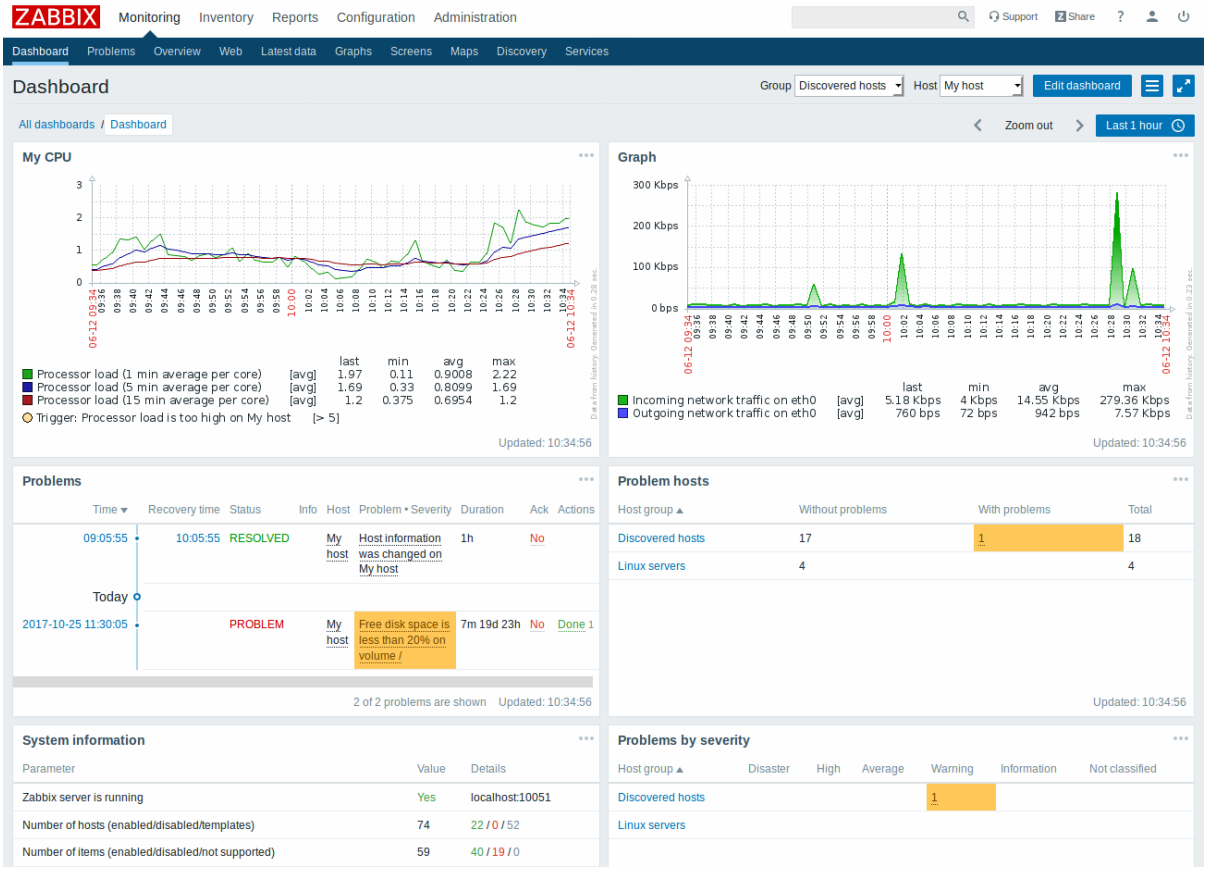
	Display page in kiosk mode. In this mode only page content is displayed. To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

1 仪表板

概览

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。

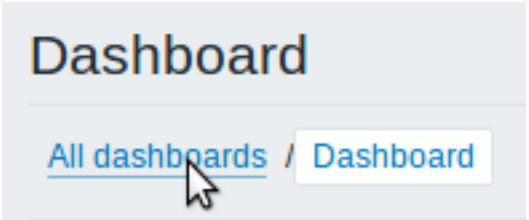
在定义单个仪表板中，您可以对来自各种来源的模块进行分组以便快速浏览，同时你还可以创建包含不同模块、内容的多个仪表板并在它们之间切换。

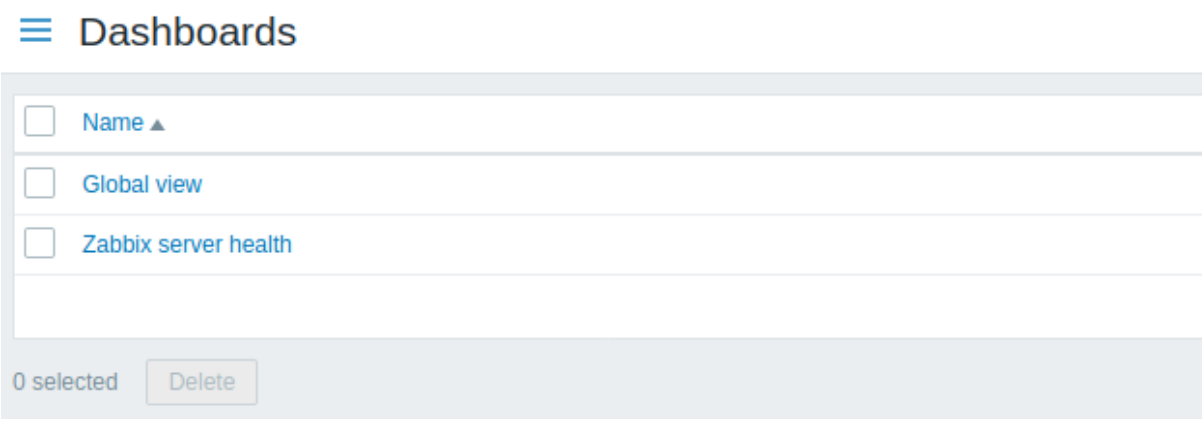
仪表板展示的时间范围你可以通过右上角的时间控制器来进行选择time period selector 单机时间控制器，来选择要查询的历史时间或者最近一段时间的数据。

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



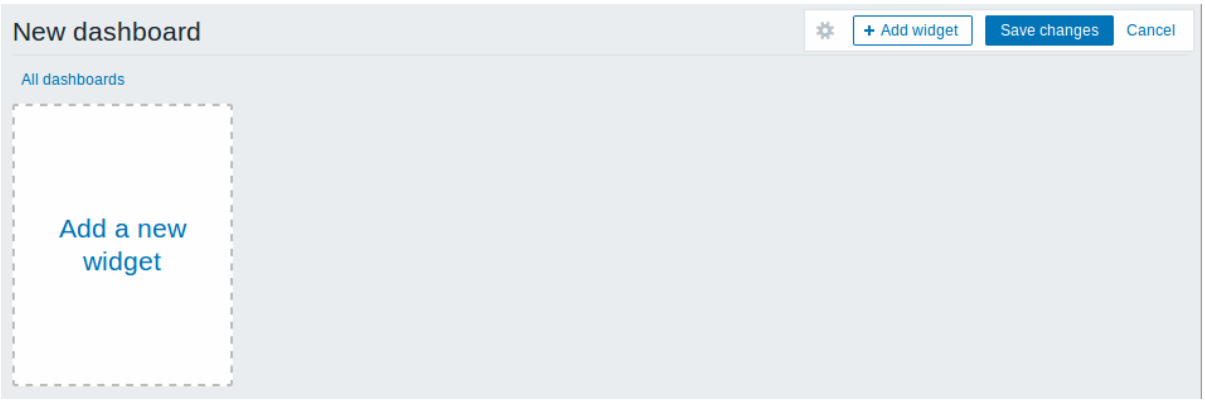


在仪表板列表中选择对应的仪表板连接。

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:

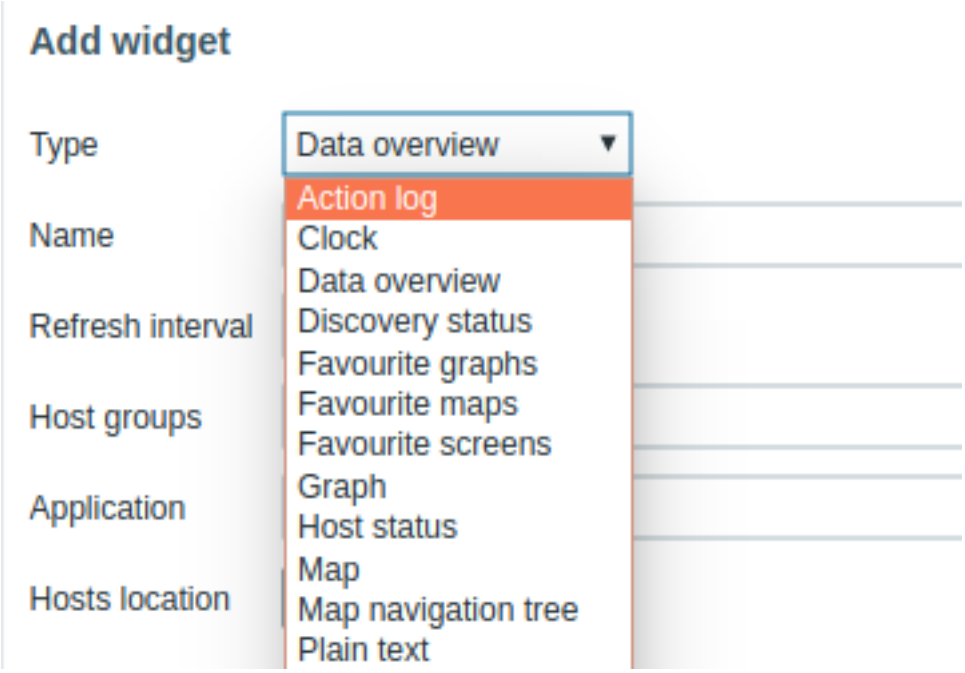
- 单击右上角 添加新构件按钮
- 单击页面左边 添加新构件的连接

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。

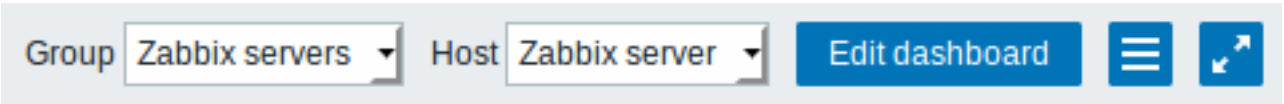
动态小构件

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

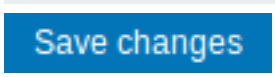
你可以通过以下选项查看单个仪表板:

		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

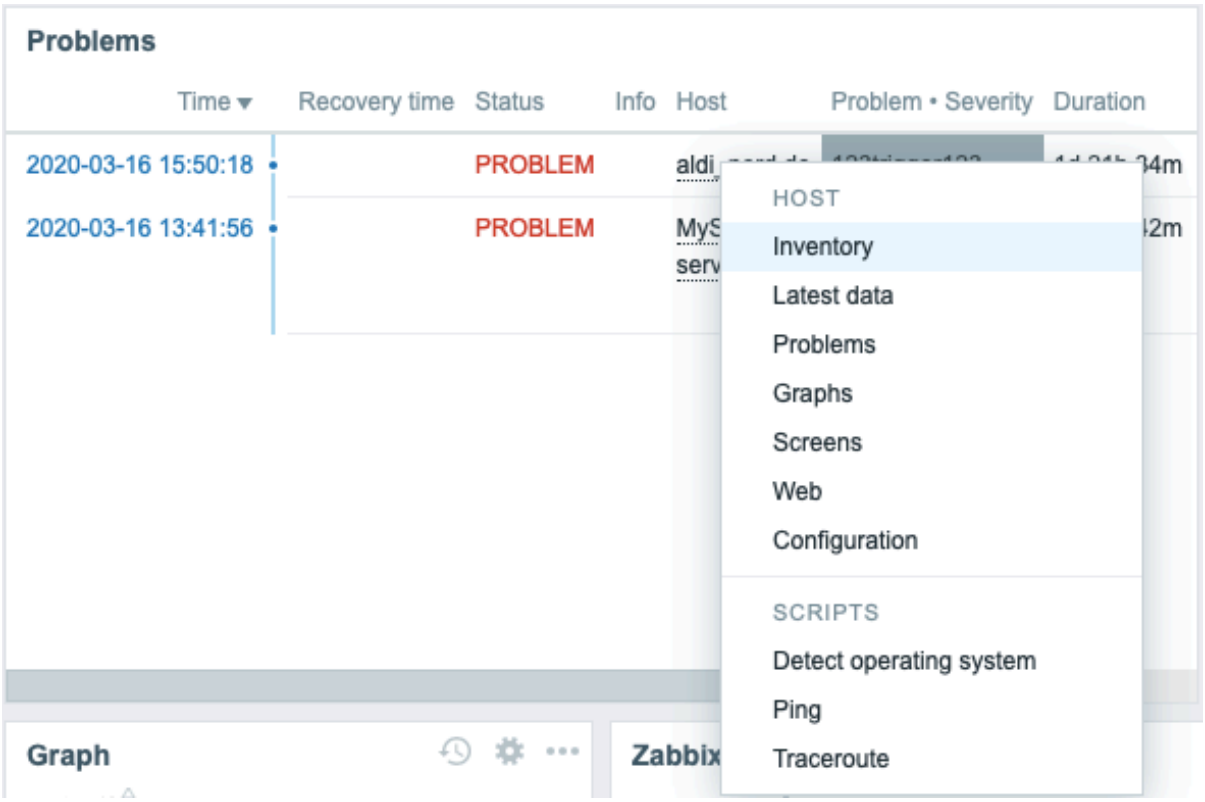
从 Zabbix 3.4.2 以后，普通和 Zabbix Admin 用户的仪表板权限受到以下限制：:

- 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。

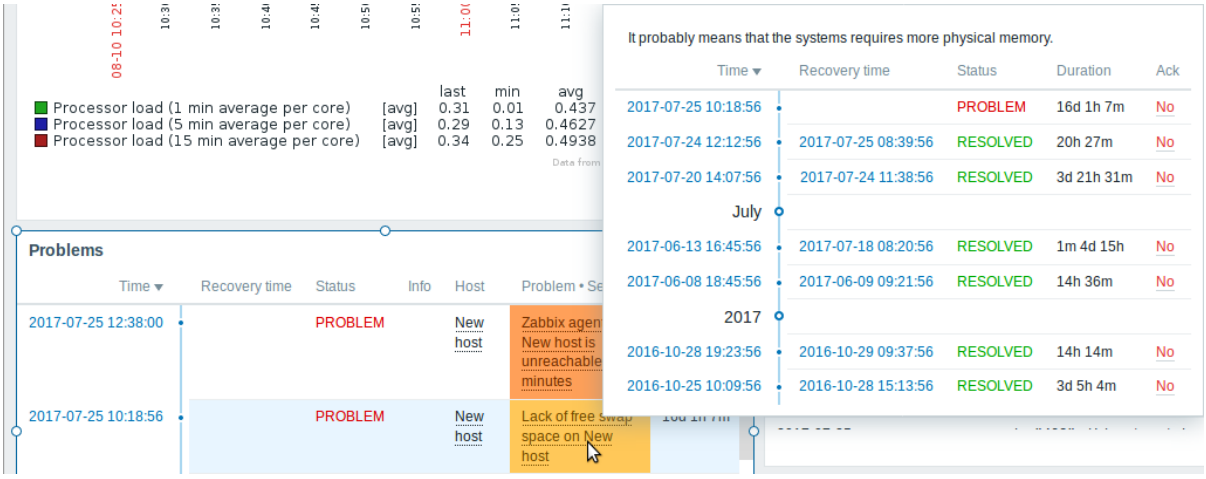


单击 WEB 前端中其他位置中的主机名，也可以访问此主机菜单。

- 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)

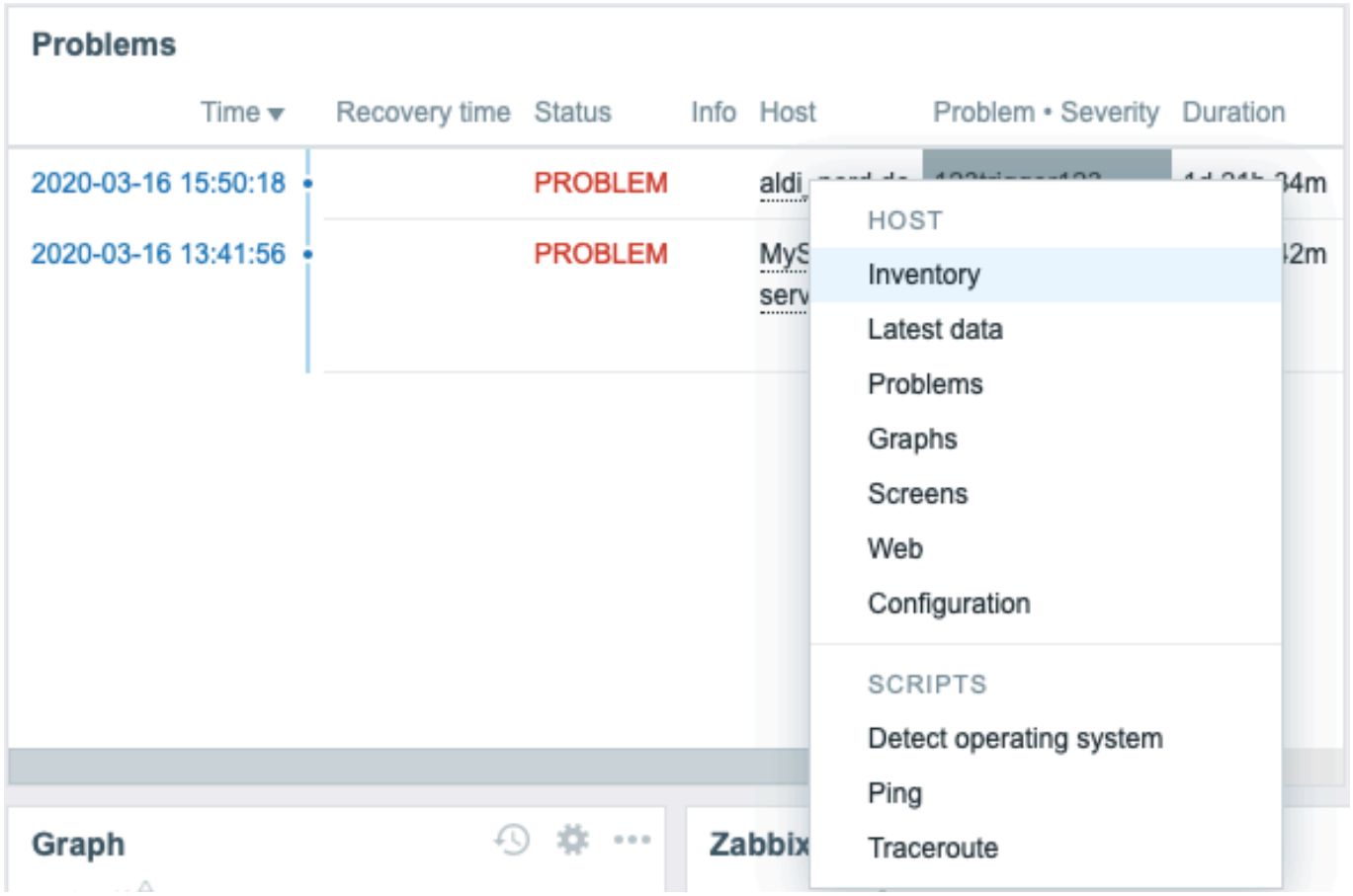
触发器弹出窗口

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。



Host menu

Clicking on a host in the *Problems* widget brings up the host menu. It includes links to custom scripts, inventory, latest data, problems, graphs, screens, web scenarios and configuration for the host. Note that host configuration is available for Admin and Superadmin users only.

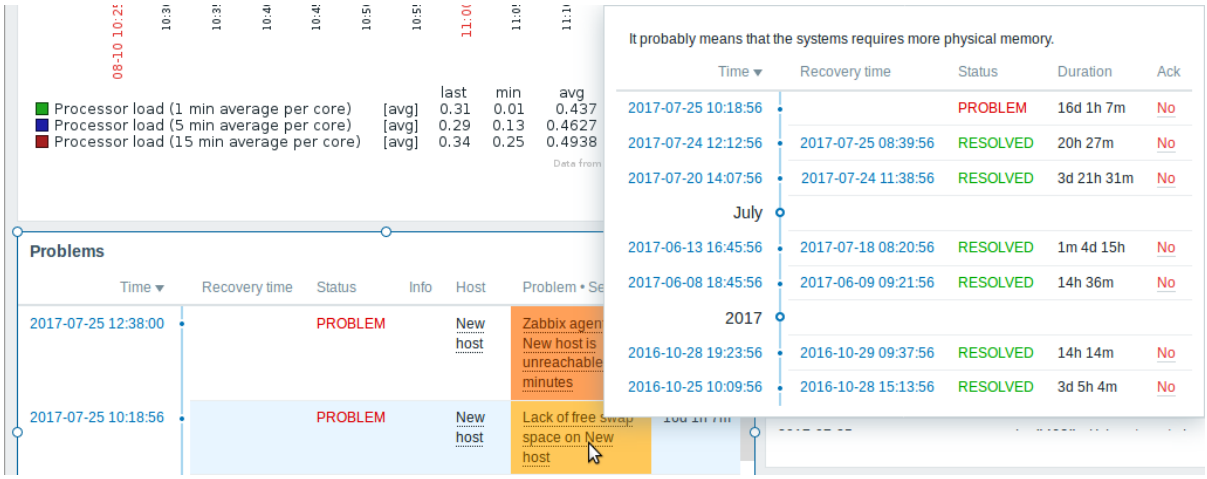


The host menu is accessible by clicking on a host in several other frontend sections:

- Monitoring → Problems
- Monitoring → Problems → Event details
- Monitoring → Hosts
- Monitoring → Hosts → Web Monitoring (since 5.0.4)
- Monitoring → Overview (on *Hosts: left*)
- Monitoring → Latest data
- Monitoring → Screens (in *Host issues* and *Host group issues* widgets)
- Monitoring → Maps
- Reports → Triggers top 100

Problem event popup

The problem event popup includes the list of problem events for this trigger and, if defined, the trigger description and a clickable URL.



To bring up the problem event popup:

- roll a mouse over the problem name in the *Problem-Severity* column of the *Problems* widget. The popup disappears once you remove the mouse from the problem name.

- click on the problem name in the *Problem-Severity* column of the *Problems* widget. The popup disappears only if you click on the problem name again.

Attention:
Resolved values of {ITEM.VALUE} and {ITEM.LASTVALUE} macros in trigger descriptions are truncated to 20 characters. To see the entire values you may use macro functions with these macros, e.g. `{{ITEM.VALUE}.regsub("(.*)", \1)}`, `{{ITEM.LASTVALUE}.regsub("(.*)", \1)}` as a workaround.

1 仪表板小构件

概览

本文列出了一些可用的仪表板 小构件，并且提供了一些配置方法。

以下参数对于每个小构件都是通用的:

Name	输入小构件名称。	
刷新间隔 *(Refresh interval)* 配置默	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。| | 显示标题 (Show header)// 选中复选框则永久显示标题。取消选中时，标题将被隐藏以节省空间，并且仅在视图和编辑模式下，将鼠标置于窗口小部件上时，标题才会向上滑动并再次可见。将小部件拖动到新位置时，	也是半可见的。

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:

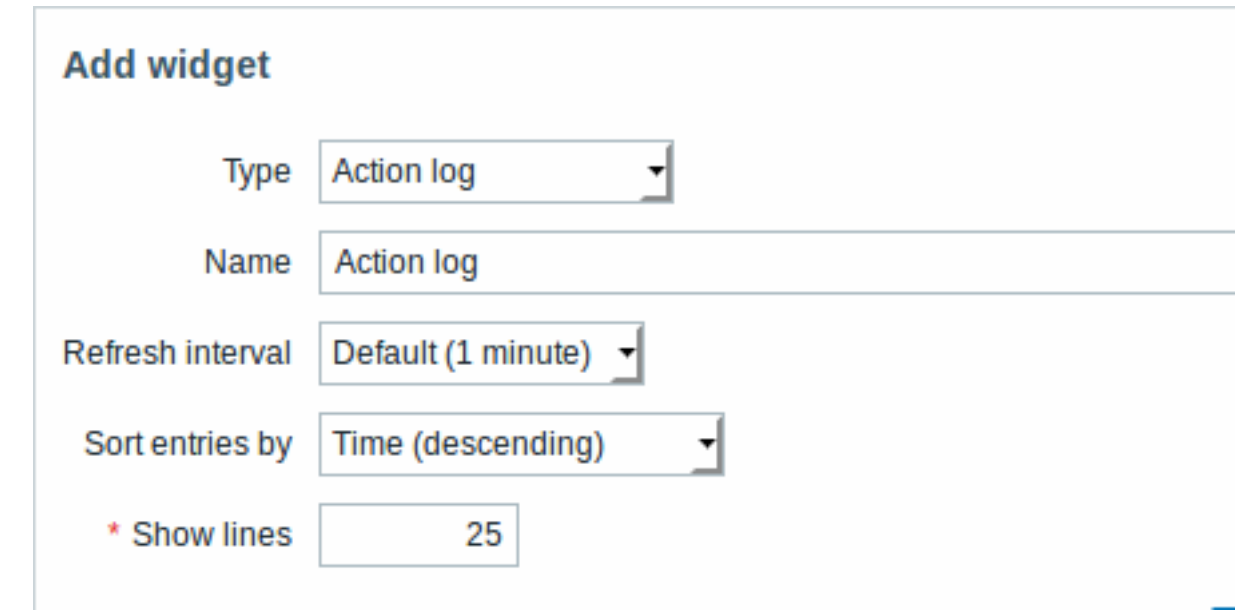
- 如果要设置全局用户的默认刷新值，请切换到编辑模式（单击 “编辑仪表板” 按钮，找到要设置的小构件，单击 “编辑” 按钮，编辑小构件表单是现在打开）并从下拉列表中选择所需的刷新间隔。
- 通过单击某个窗口小构件的 按钮，可以在视图模式下单独为某个用户设置唯一的刷新间隔。

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

你可以在动作日志小构件中显示关于动作操作的详细信息（通知，远程命令）。它的信息是从 *Administration* → *Audit* 这里复制的。

通过选择 *Action log* 类型来配置:



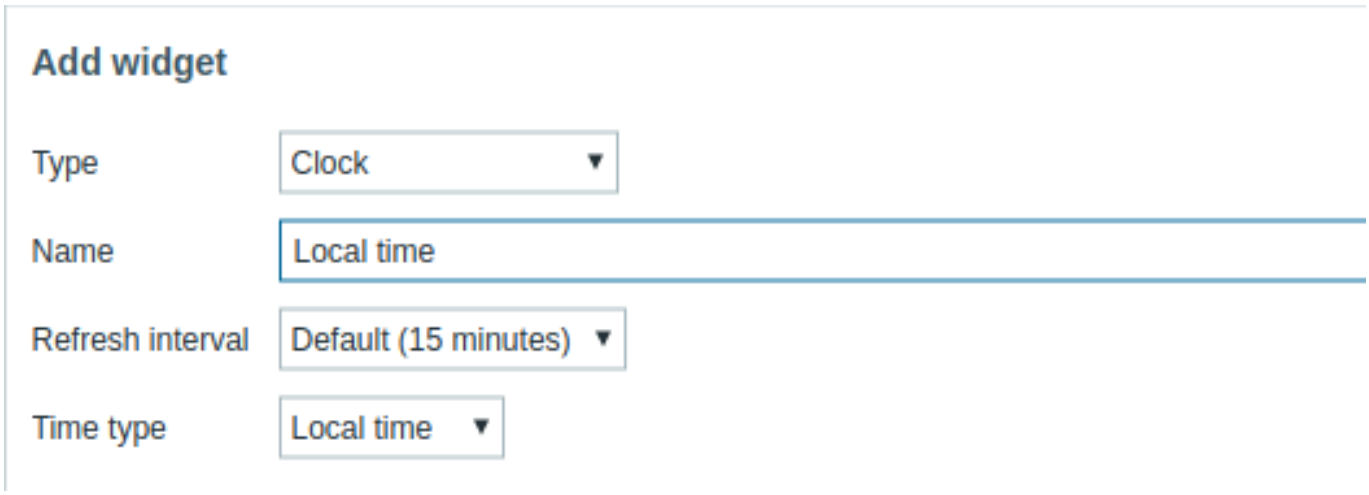
你可以设置以下特定选项:

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：



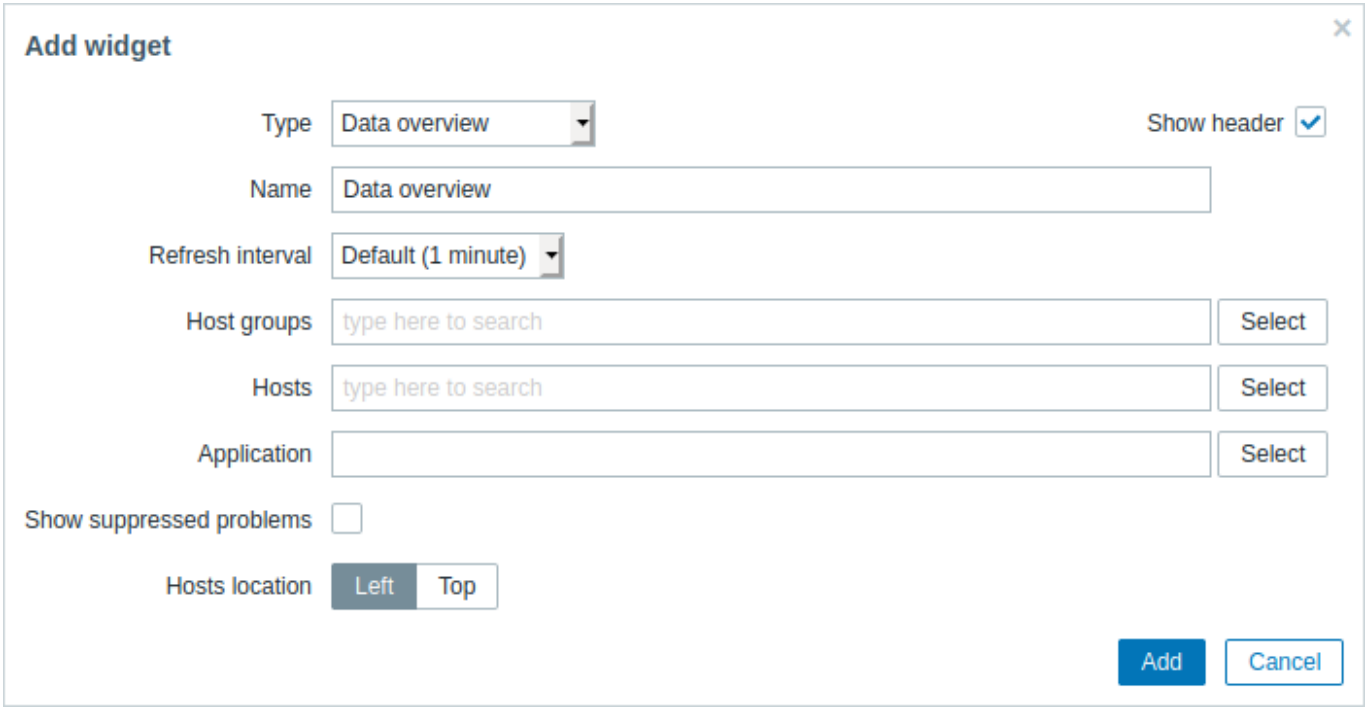
你可以设置以下特定选项:

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*）).

如果要配置，请选择//数据概述 // 类型:

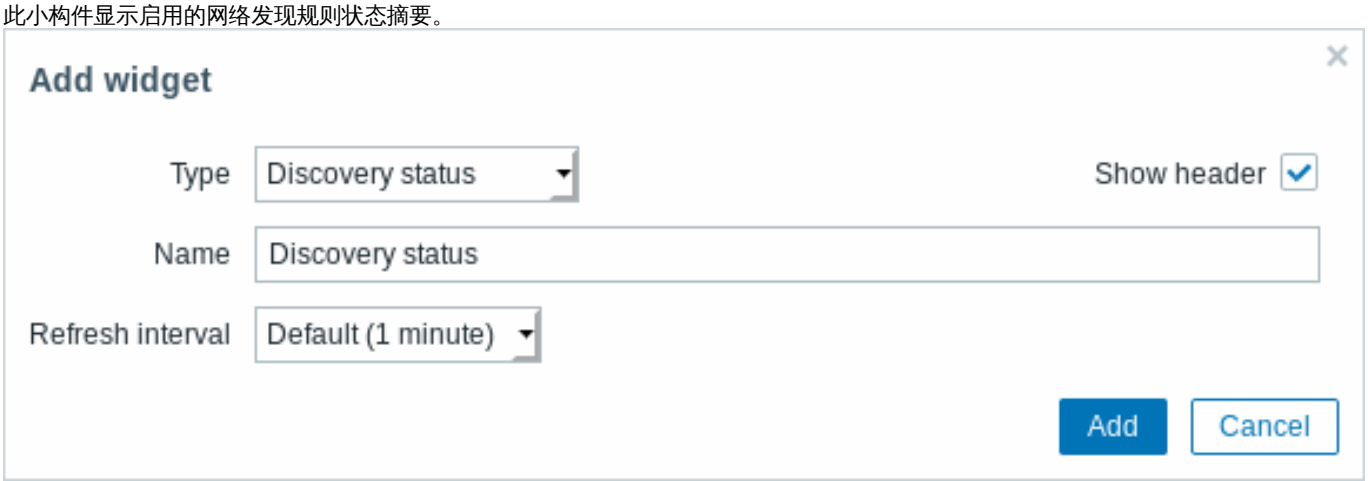


你可以使用以下特定配置选项:

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。
显示被抑制的问题选中该复选框以	示由于主机维护而被抑制（未显示）的问题。
主机地址选择主	位置，左侧或者顶部。

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，图表名称按字母顺序排序。

当您查看图形时单击 添加到收藏夹按钮时，将图形添加到快捷方式列表。

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 添加到收藏夹按钮时，将图形添加到快捷方式列表。

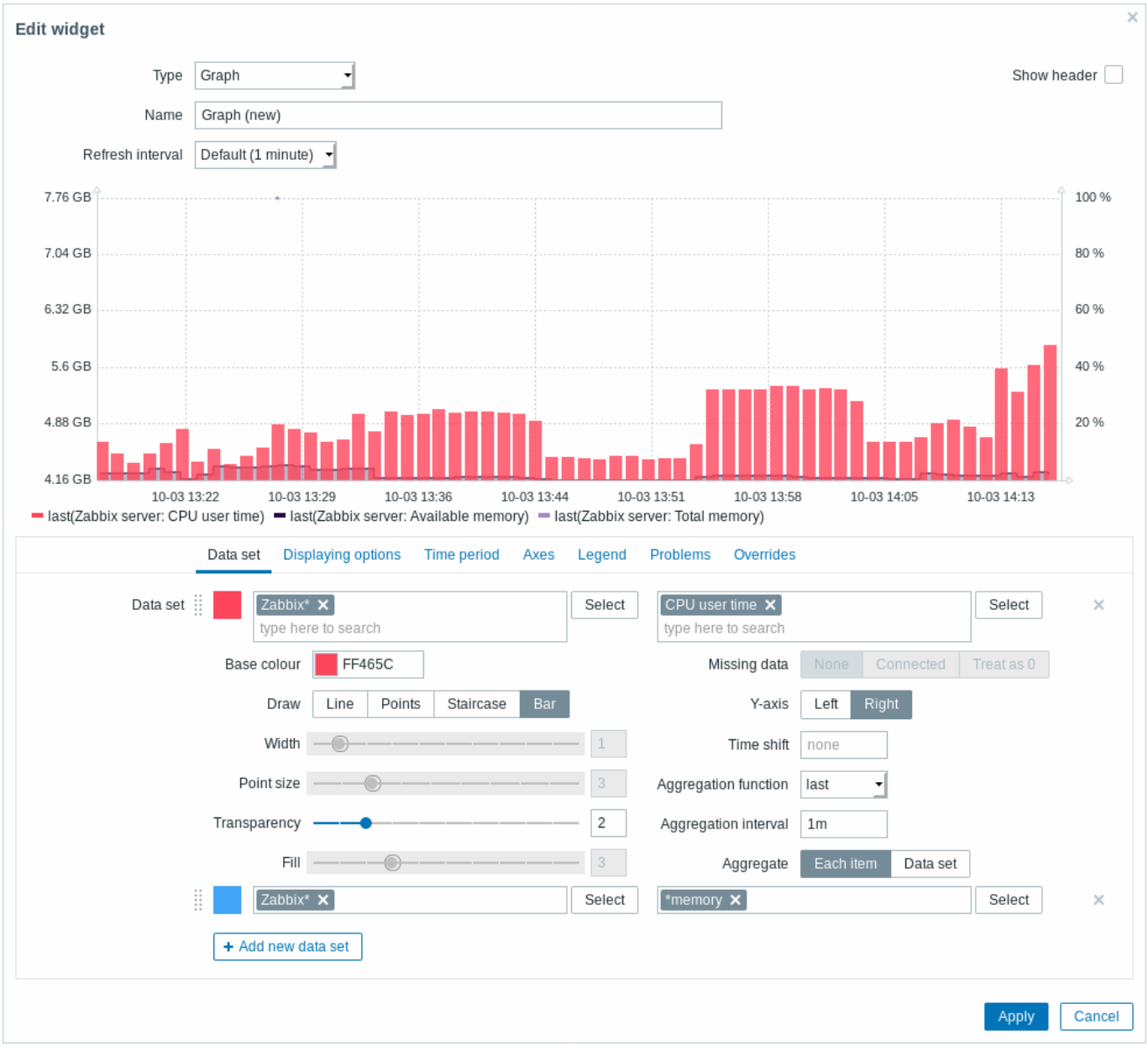
喜欢的聚合图形

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 添加到收藏夹按钮时，将图形添加快捷方式列表。

图表

图形小构件提供了一种现代且通用的方式，可以使用矢量图绘制技术来可视化 Zabbix 收集的数据。此图形小构件从 Zabbix 4.0 开始支持。请注意，Zabbix 4.0 之前支持的图形小构件仍可以用作图形 (经典的)。

如果要配置, 请选择 图表 *(Graph)* 类型：



数据集选项卡允许添加数据集并定义其视觉效果:

数据集 *(Data set)* 选择	在图表上显示的主机和监控项。或者，您可以输入主机和监控项的模糊匹配。可以使用通配符模式（例如：* 将返回匹配到零个或多个字符的结果）。指定通配符模式，只需手动输入字符串，然后按 Enter。在键入时，请注意下拉列表中是如何显示所有匹配的主机。 图表中最多可以显示 50 个项目。 主机模式和监控项模式字段是必填字段。 请注意，在此字段中不解析包含不推荐使用的位置宏（$1- $9）的监控项名称。因此，不会将名称如 `CPU $2 time` 的监控项，以解析后的名称 (如 `CPU user time`) 单独添加到图形中; 而使用其未解析的名称 `CPU $2 time`, 将所有对应监控项添加到图形中（例如:`CPU user time`,`CPU system time`,`CPU idle time`, 等。） 通配符总是被转换，因此，如果还有其他匹配项（例如 item2，item3），则不可能单独添加例如名为 “ item *” 的项目。
底色 *(Base color)* 通	颜色选择器或手动调整基本颜色。底色用于为数据集的每个项目设置不同的颜色。
绘制 *(Draw)* 选	指标的绘图类型。可能的绘制类型为 “线”（默认设置），“点”，“阶梯” 和 “条形”。 请注意，如果折线/楼梯图中只有一个数据点，则无论绘制类型如何，都将其绘制为点。 点大小是根据线宽计算得出的，即使线宽再小，也不能小于 3 个像素。
宽度 *(Width)* 设	线宽。选择 “线条” 或 “阶梯” 绘制类型时，此选项可用。
点大小 *(Point size)* 设置	大小。选择 “点” 绘制类型时，此选项可用。
透明度 *(Transparency)* 设置	明度级别。
填充 *(Fill)* 设	填充级别。选择 “线条” 或 “楼梯” 绘制类型时，此选项可用。
缺失数据 *(Missing data)* 设置显	缺失数据的选项： **None** - 缺失留空 **Connected** - 连接两个边界值 **Treat as 0** - 丢失的数据显示为 0 值 不适用于 “点” 和 “线” 绘制类型。
Y 轴 *(Y-axis)*	择设置图表侧边 Y 轴。
时间偏移 *(Time shift)* 根据需	定义时间偏移。要展示自定义图表时，你可以在该字段使用时间后缀。可以是负值。
聚合功能 *(Aggregation function)* 定义要	用的聚合函数： **min** - 显示最小值 **max** - 显示最大值 **avg** - 显示平均值 **sum** - 显示值的总和 **count** - 显示统计值 **first** - 显示第一个值 **last** - 显示最新值 **none** - 显示所有值 (不聚合) 聚合允许显示所选时间间隔（5 分钟，一小时，一天）的聚合值，而不是所有值。参照:聚合图形 从 Zabbix 4.4 开始支持此选项。
聚合间隔 *(Aggregation interval)* 定义值	合间隔。你可以在该字段使用时间后缀。不带后缀的数值将被视为秒。 从 Zabbix 4.4 开始支持此选项。
聚合 *(Aggregation)* 定	是否聚合： **Each time** - 数据集中的每个项目都将聚合并单独显示。 **Data set** - 所有数据集项将被聚合并显示为一个值。 从 Zabbix 4.4 开始支持此选项。

现有数据集显示在列表中。您可以这样:

-

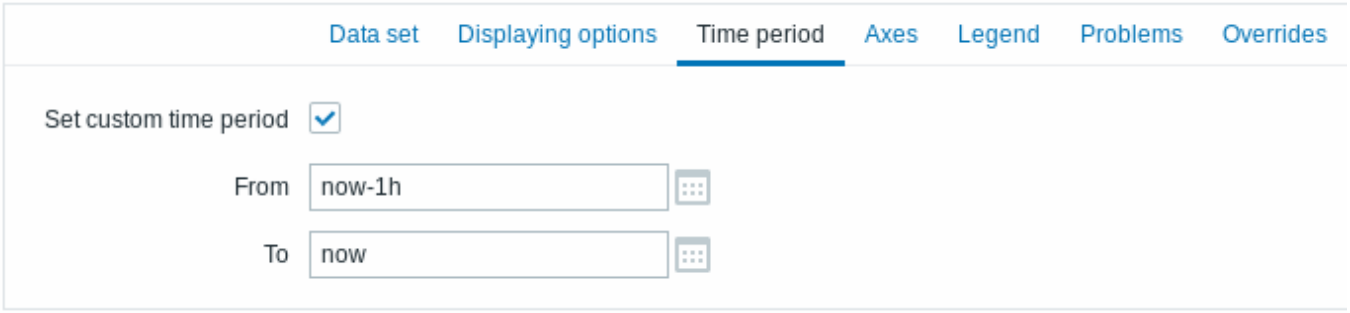
- 单击此按钮添加新的数据集
- - 单击颜色图标以展开/折叠数据集详细信息

- - 点击移动图标，然后将数据集拖到列表中的新位置

显示选项选项卡允许定义历史数据选择：
graph_displaying_options2.png

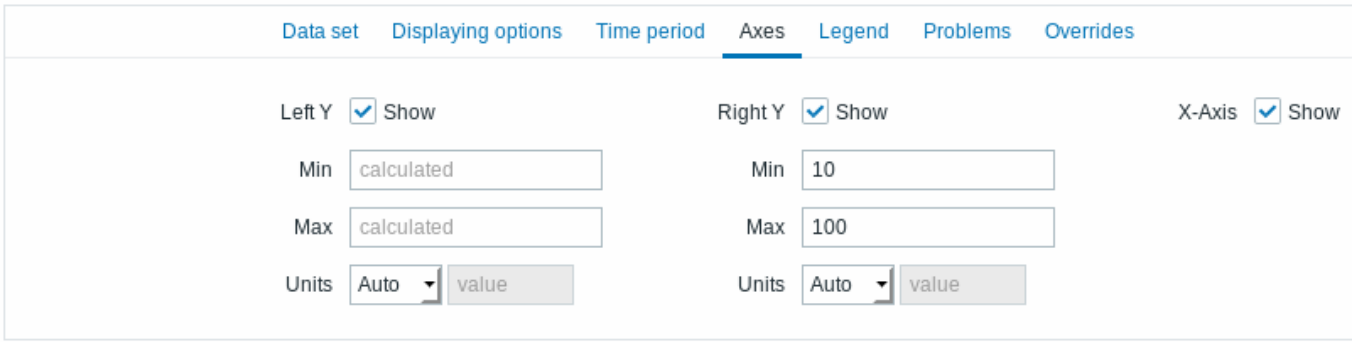
历史数据选择设置图形数	的来源： **Auto** - 数据根据经典图算法获得（默认）。 **History** - 历史数据。**Trends** - 趋势数据。

时间段选项卡允许设置自定义时间段：



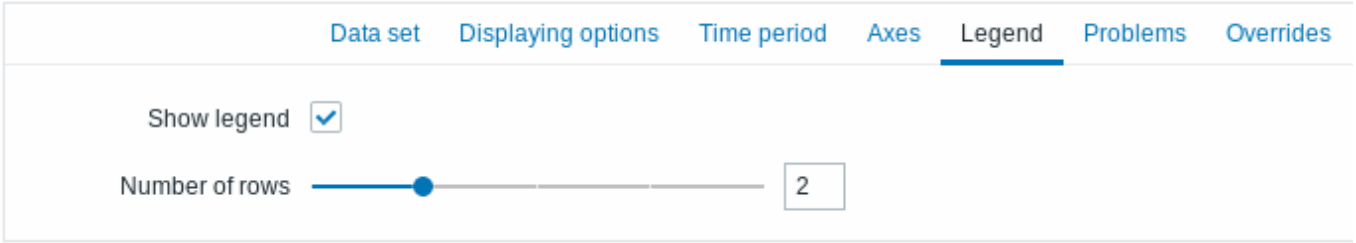
设置自定义时间段选中此复选框可	置图表的自定义时间段（默认情况下未选中）。
From	设置图表的自定义时间段的开始时间。
To	设置图表的自定义时间段的结束时间。

轴选项卡允许自定义轴的显示方式：



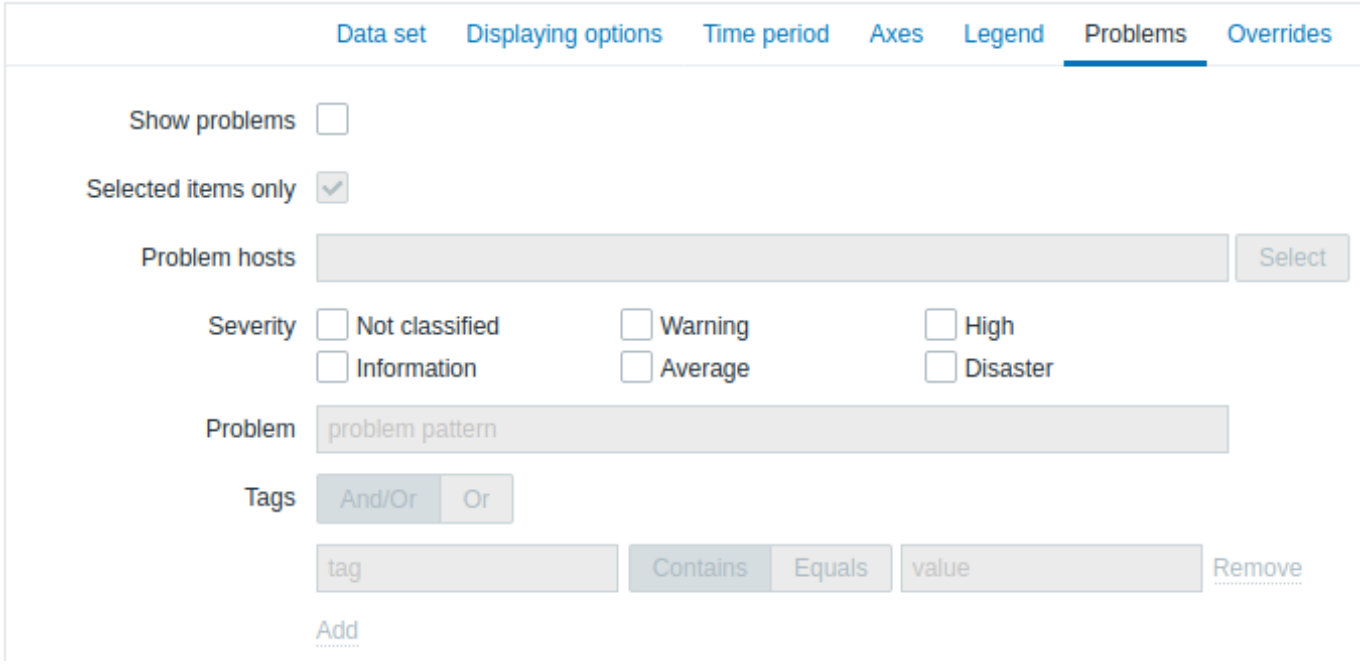
Left Y	选中此复选框以使左 Y 轴可见。如果在“数据集”或“覆盖”选项中未选中该复选框，则会被禁用。
Right Y	选中此复选框以使右 Y 轴可见。如果在“数据集”或“覆盖”选项中未选中该复选框，则会被禁用。
X-Axis	取消选中此复选框可隐藏 X 轴（默认选中）。
Min	设定对应坐标轴的最小值。设置 Y 轴可见范围的最小值。
Max	设定对应坐标轴的最大值。设置 Y 轴可见范围的最大值。
Units	从下拉列表中选择图形坐标轴的单位。如果选择“Auto”选项，则以第一个监控项的单位作为坐标轴单位。“Static”选项用来自定义坐标轴单位。如果选择了“Static”选项，并且值输入字段保留为空白，则相应的坐标轴单位名由数字值组成。

图例选项卡允许自定义图例：



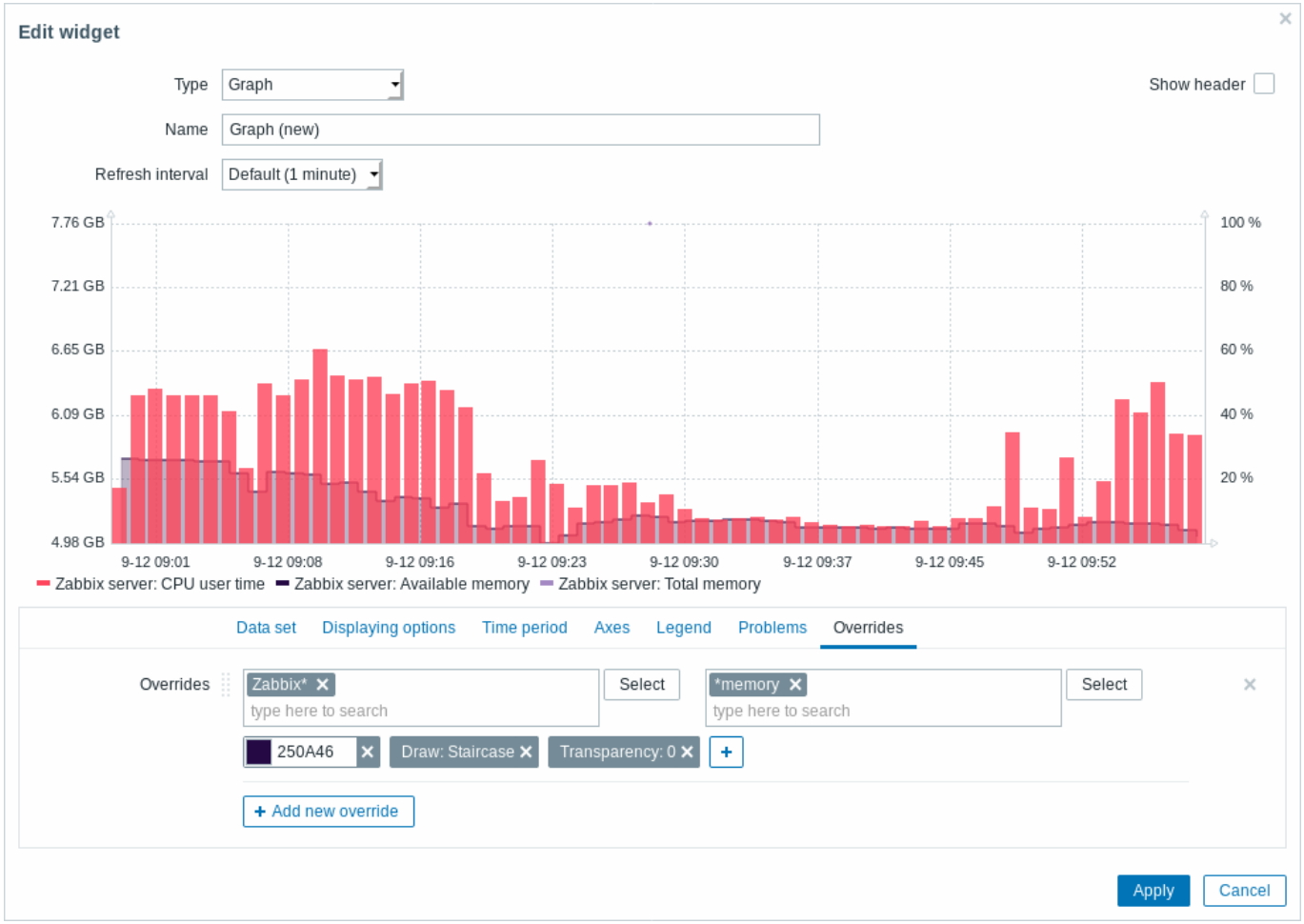
显示图例 *(Show legend)* 取消选	此复选框可在图形上隐藏图例（默认选中）。
行数 *(Number of rows)* 设	要在图形上显示的行数。

问题选项卡允许自定义显示问题：



显示问题 *(Show Problems)* 选中此	选框启用在图形上显示问题（未选中，即默认情况下处于禁用状态）。
仅所选项目 *(Selected items only)* 选中此复	框，在图表上仅显示包含所选项目的问题。
异常主机 *(Problem Hosts)* 选择要	图表上显示的异常主机。可使用通配符（例如：* 返回匹配零个或多个字符的结果）。要指定通配符模式，只需手动输入字符串，然后按 Enter。在键入时，请注意下拉列表中如何显示所有匹配的主机。
严重性 *(Severity)* 标记	在图表上显示的问题严重性。
问题 *(Proble)* 指	要在图表上显示的问题名称
标签 *(Tags)* 指	标签名称和值以限制图形上显示的问题数量。要添加更多标签名称和值，请单击 “添加”。 有几种条件的两种计算类型: **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 “或” 条件分组 **Or** - 如果满足一个条件就足够了 匹配标记值的方法有两种： **Contains** - 区分大小写的子字符串匹配（标签值包含输入的字符串） **Equals** - 区分大小写的字符串匹配（标记值等于输入的字符串）。

“覆盖” 选项卡允许为数据集添加自定义覆盖：

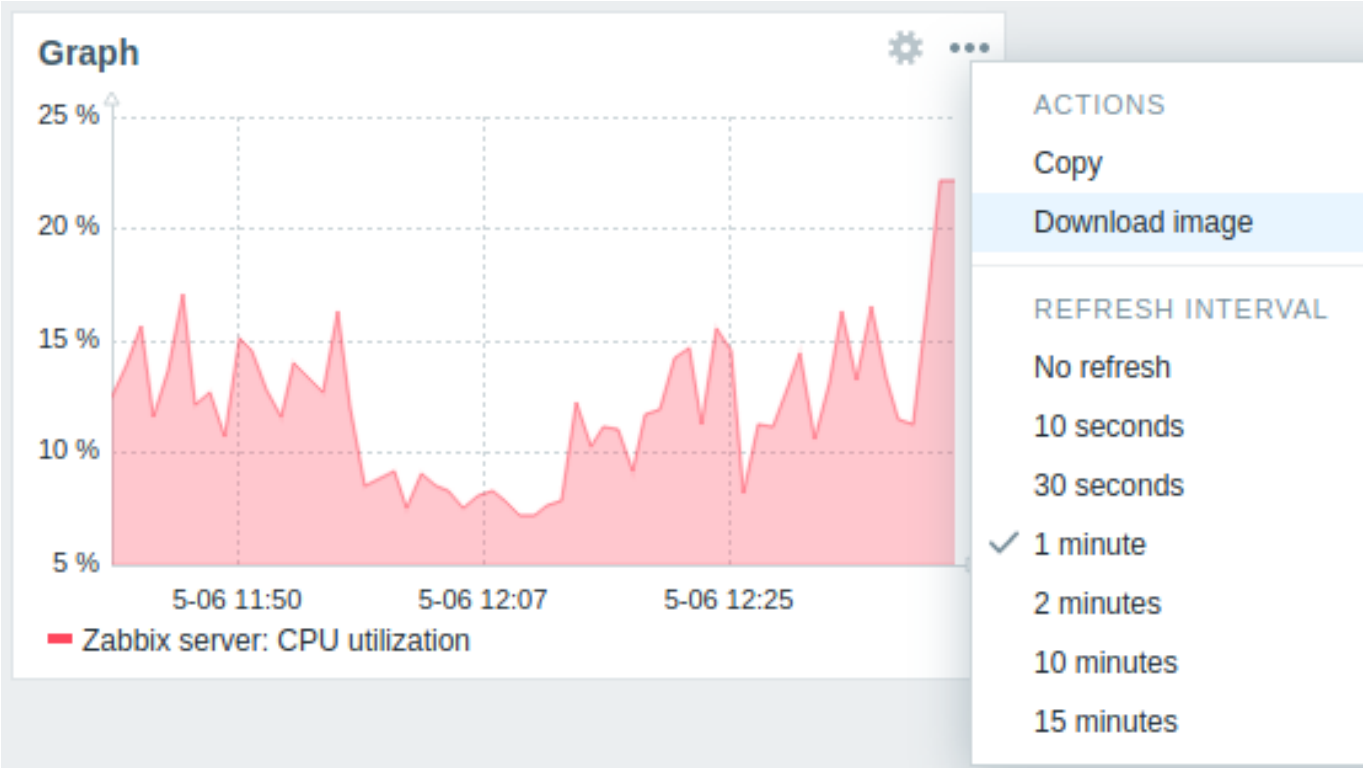


“覆盖”非常有用，当使用 * 通配符为数据集选择了多个项目，并且您想更改默认情况下监控项的显示方式（例如，默认基色或任何其他属性）。

现有的“覆盖”（如果有）会显示在列表中。要添加新的覆盖：

- 点击该 按钮
- 选择主机和监控项作为覆盖。或者，您可以输入主机和监控项模糊匹配。可以使用通配符（例如：* 返回匹配零个或多个字符的结果）。要指定通配符模式，只需手动输入字符串，然后按 Enter。在键入时，请注意下拉列表中如何显示所有匹配的主机。通配符始终被解释，因此，如果还有其他匹配项（例如 item2，item3），则无法分别添加名为“ item *”的监控项。主机模式和监控项模式字段是必填字段。
- 点击 ，选中覆盖参数。至少选择一个覆盖参数。有关参数的说明，请参见上面的“数据集”选项卡。

图形小构件显示的信息可以使用小部件菜单下载为.png 图像：



小构件的屏幕快照将保存到 “下载” 文件夹中。

图表（经典）

在经典图表小构件中，您可以展示单个自定义图表或简单图表。

请选择 Graph(classic) 作为配置类型：

您可以设置以下特定选项：

来源 *(Source)* 选	图形类型: **Graph** - 自定义图形 **Simple graph** - 简单图。
图表 *(Graph)* 选	要显示的自定义图形。 如果来源 *(Source)* 选择 “图表”，则此选项可用。
监控项 *(Item)* 选择	在简单图中显示的监控项。 如果来源 *(Source)* 选择 “简单图”，则此选项可用。
显示图例 *(Show legend)* 不选中	复选框会隐藏图表上的图例（默认选中）。
动态监控项 *(Dynamic item)* 设置图表	据所选主机显示不同的数据。

经典图表窗口小构件展示的信息，可以使用窗口小构件件菜单下载为.png 图像：

小构件的截图将保存到 “下载” 文件夹中。

图表原型

在图表原型小构件中，您可以展示通过低级发现从图原型或监控项原型创建的网格图。

请选择图表原型 *(Graph prototype)* 作为窗口小构件配置类型：

您可以设置以下特定选项：

来源 *(Source)* 选 图表原型 *(Graph prototype)* 选择图	源：图表原型或简单图原型。 原型来展示已发现图表原型。 如果来源 *(Source)* 选择 “图表原型”，则此选项可用。
监控项原型 *(Item prototype)* 选择监控	原型来展示基于监控项原型发现的监控项简单图。 如果来源 *(Source)* 选择 “简单图原型”，则此选项可用。
显示图例 *(Show legend)* 不选中	复选框会隐藏图表上的图例（默认选中）。
动态监控项 *(Dynamic item)* 设置图表	据所选主机显示不同的数据。
列 *(Columns)*	入要在图表原型小构件中展示的图表列数。
行 *(Rows)*	入要在图表原型小构件中展示的图表行数。

主机可用性

在主机可用性小构件中，您可以展示有关主机可用性的高级统计信息。

请选择主机可用性作为配置类型：

您可以设置以下特定选项：

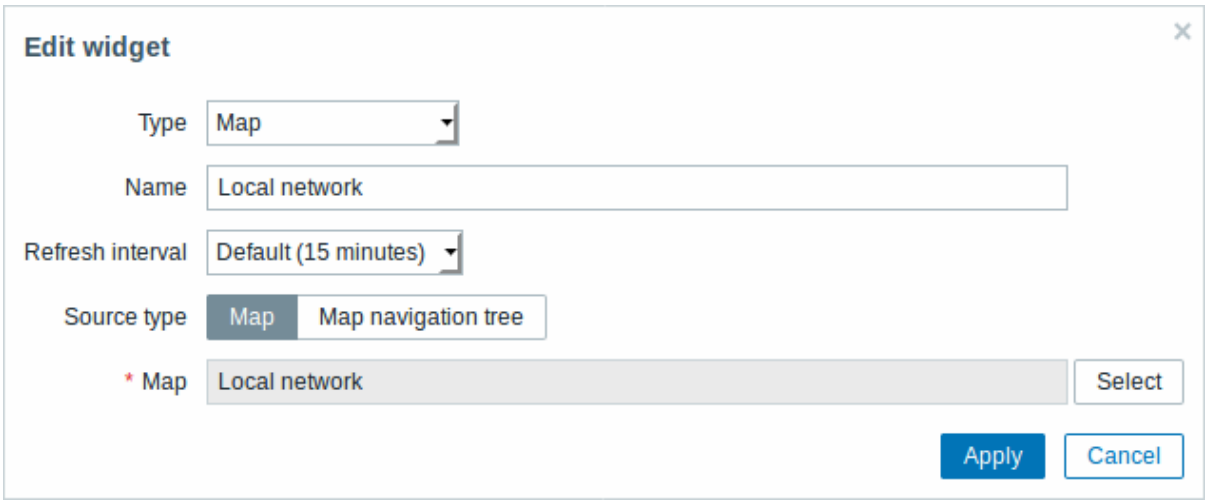
主机组 *(Host Groups)* 选择	机组。该字段是自动填写的，输入组名时会出现匹配组的下拉列表。向下滚动选择。
接口类型 *(Interface type)* 选择要	看可用性数据的主机接口。 如果不选择，则默认显示所有接口的可用性。
布局 *(Layout)* 选	垂直或水平显示。
显示维护期主机 *(Show hosts in maintenance)* 在统计信息中	括维护中的主机。

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:

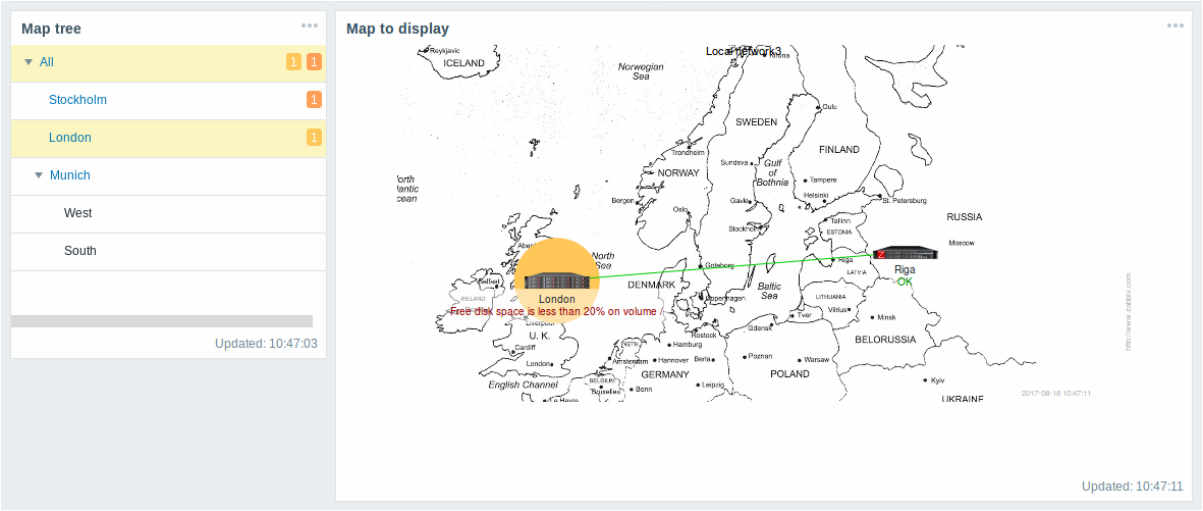
来源类型 *(Source type)* 选择显	： 拓扑图 - 网络拓扑图 拓扑图导航树 - 所选拓扑图导航树中的一个拓扑图
地图 *(Map)* 选	要显示的拓扑图。当选择 “拓扑图” 类型作为源类型 *(Source type)* 时，此选项可用。
过滤 *(Filter)* 选	拓扑图导航树显示。当选择 “拓扑图导航树” 作为源类型 *(Source type)* 时，此选项可用。

参照：IE11 的已知问题

拓扑图导航树

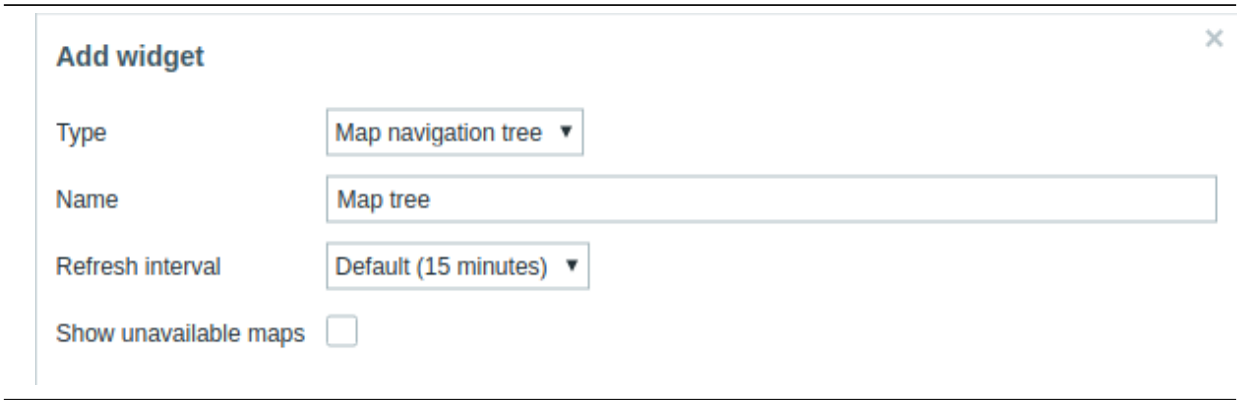
此窗口小构件允许构建现有拓扑图的层次结构，同时还显示每个包含的拓扑图和拓扑图组的问题统计信息。

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：



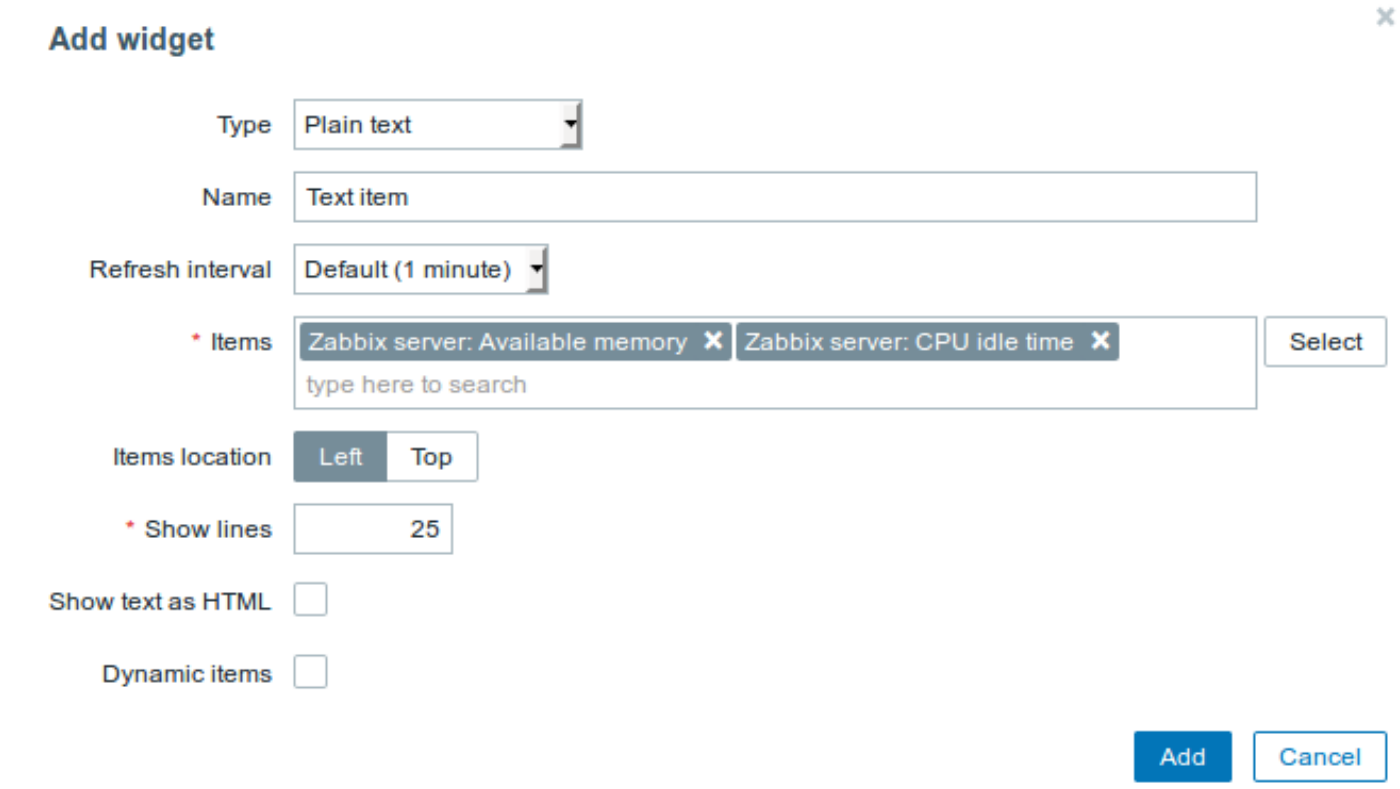
你还需要配置以下选项:

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:



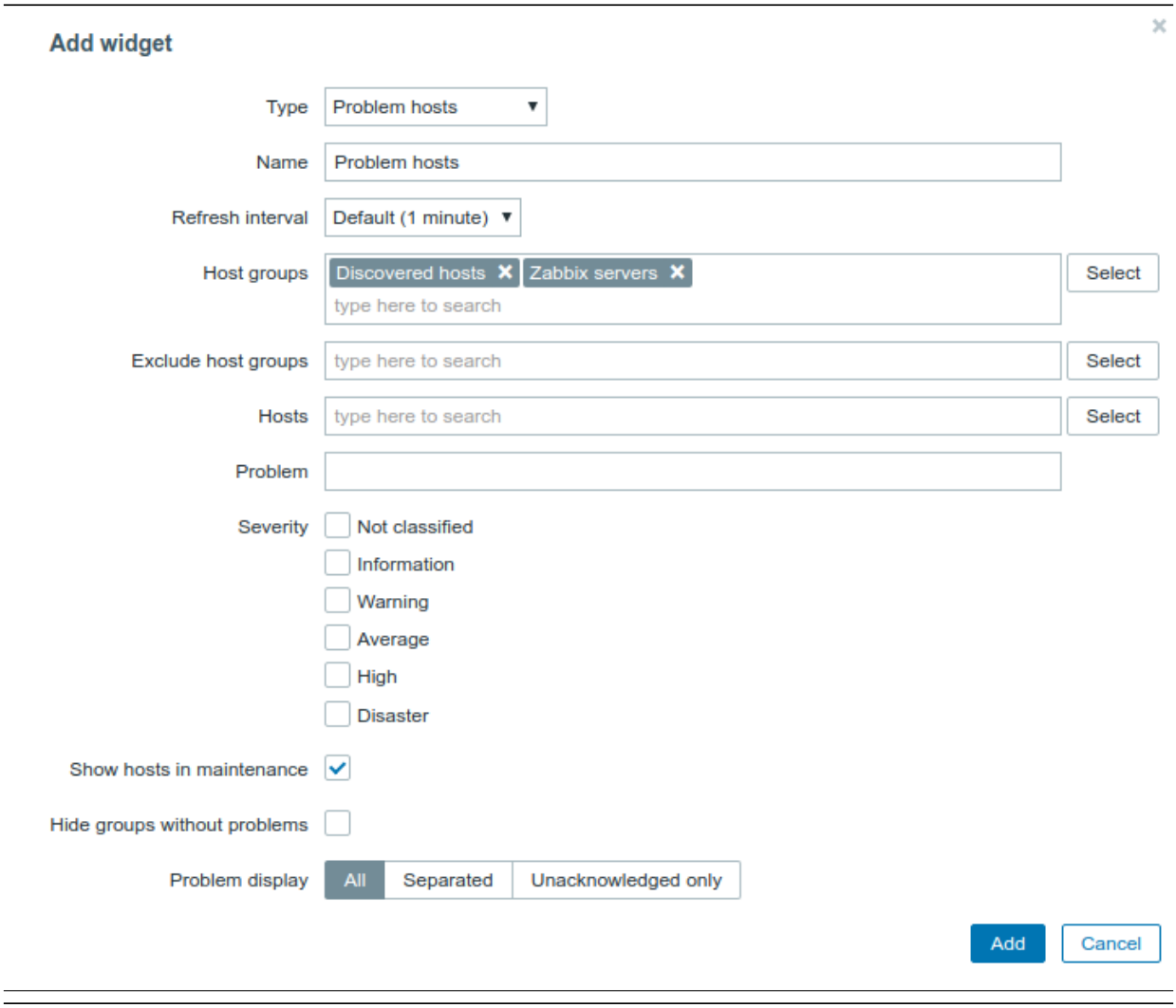
你还需要设置以下选项:

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。

问题主机

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
标签通	设置标签名称和值，来限制图表中要展示问题主机的数量。要添加更多标签名称和值，请单击添加。几个条件之间有两种计算类型：*且/或(And/Or)*-和/或-必须满足所有条

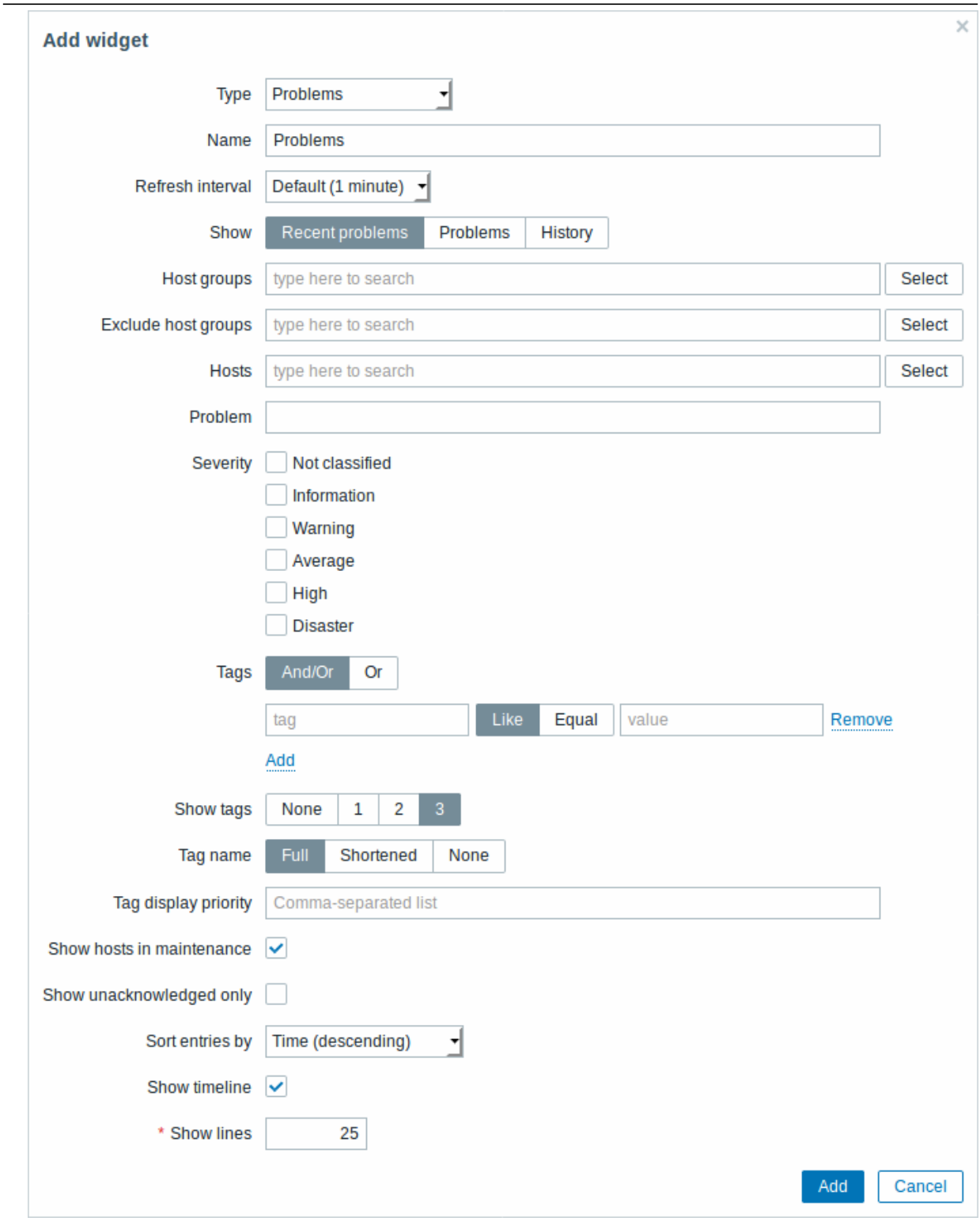
参数类型功能说	
隐藏没有问题的组选中 * 隐藏没	问题的组 * 选项，在该构件中隐藏没有问题的组。

参数	类型	功能说
显示异常	显示异	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。

问题

在小构件中展示问题信息. 此构件中内容类似 监测中（*Monitoring*） → 问题（*Problems*）.

配置方法，选择 问题类型:



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
主机在	构件中输入主机来展示异常。输入后会自动出现与之匹配的下拉菜单。如果不输入主机，则显示所有异常的主机。

参数类型	功能说
问题你	以根据名称来限制要展示异常的数量。如果你在此输入字符串，仅显示名称中包含输入字符串的那些异常。不能使用宏变量。

参数类型	功能说
严重性标记	在小构件中展示的异常严重性。

参数	类型	功能说
标签指		事件标签和值来限制要展示的问题数量。要添加更多事件标签和值，请单击添加。多个条件之间有以下两种运算类型：且/或 **(And/Or)** - 必须满足所有条件，具有相同标签名

参数类型功能说	
展示标签选择展	的标签数量：无 **(None)**- 在异常 *(Problems)* 小构件中不展示标签 *(Tags)* 列 **1**- 在标签 *(Tags)* 列中展示一个标签 **2**- 在标签 *(Tags)* 列中展示两个标签 **3**- 在标签 *(Tags)* 列中展示三个标签要查看

参数类型	功能说
标签名称选择标	名称显示模式：完整 **(Full)** - 标签名称和值完整显示。缩短 **(Shortened)** - 标签名称缩短为 3 个符号；标签值完整显示。无 **(None)** - 只显示标签值；不显示标签名称。

参数类型功能说	
标签显示优先级输入异常的标	显示优先级，标签列表以逗号分隔(例如：Services,Appl只能使用标签名称，不能标签值。该列表中的标签始终优先展示，不是按字母自然排序。

参数类型	功能说
展示运营数据	选择展示运数据的模式：无**(None)** - 不展示运营数据单独**(Separately)** - 在单独的列中展示运营数据和异常名称一起**(With problem name)** - 将运营数据用括号引起了附加到异常名称后面

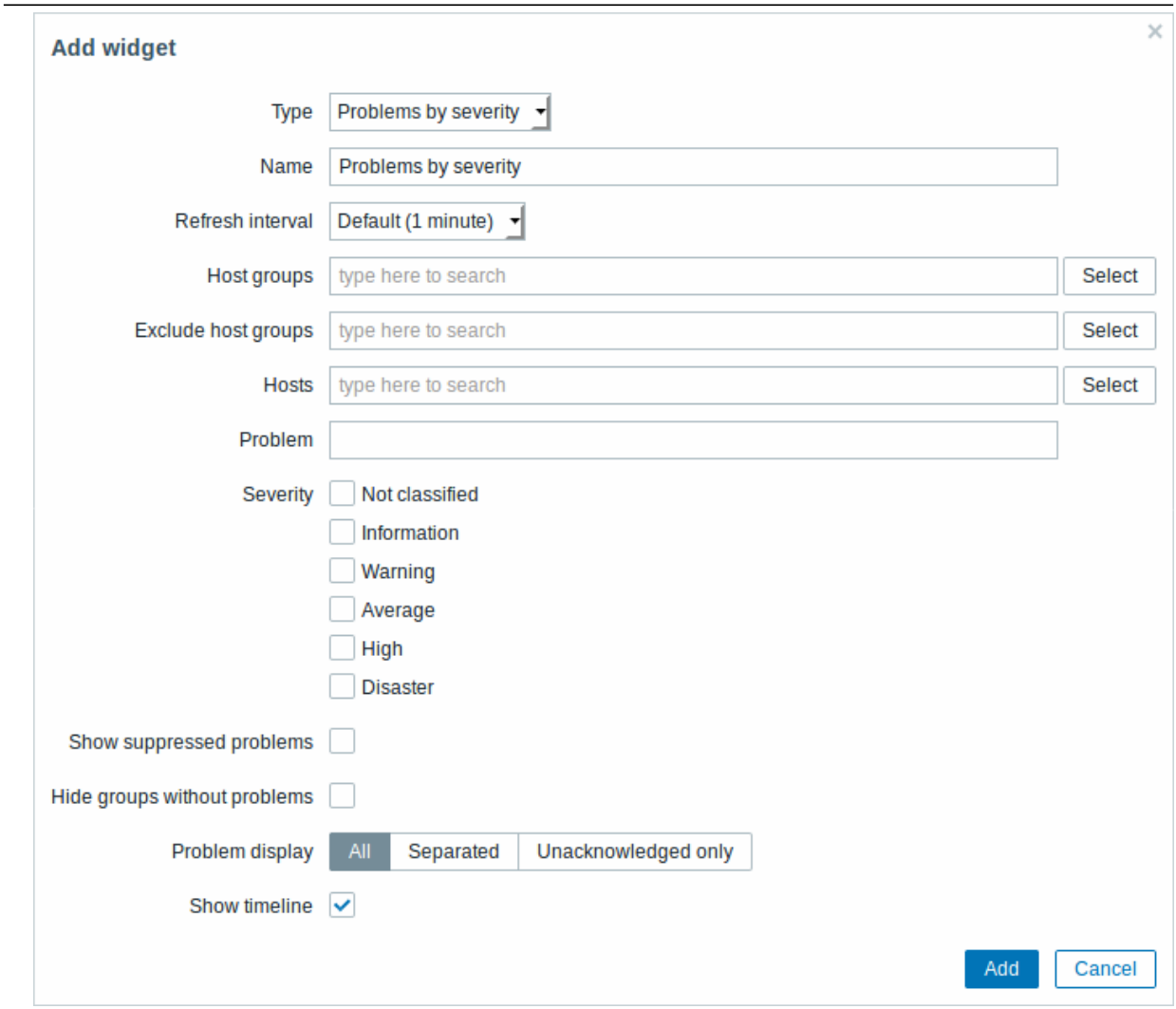
参数类型	功能说
展示维护期异常选中该复选框	展示由于主机在维护期而被抑制(未显示)的问题。
只展示未确认的异常选中该复选框，则	展示未确认的异常。

参数类型功能说	
排序依据排序依	：时间**(Time)**(降序或升序)严重性**(Severity)**(降序或升序)异常名称**(Problem name)**(降序或升序)主机**(Host)**(降序或升序)。
显示时间轴选中该复	框以显示可视时间轴。
显示行数设置要	示的异常行数。

异常严重性

你可以在这个小构件中根据严重性来展示异常。你可以限制哪些主机和触发器能够在这里展示以及定义异常计数的展示方式。

选择按严重性的问题作为配置类型:



你可以设置以下特定选项:

参数	描
主机组输入	显示的主机组。输入组名称时会自动出现与之匹配的主机组下拉菜单。指定父主机组将隐式选择所有嵌套主机组。这些主机组中的主机数据将被显示。如果没

参数描	
排除主机组输入要隐	的主机组。输入组名称时会自动出现与之匹配的主机组下拉菜单。指定父主机组将隐式选择所有嵌套主机组。这些主机组中的主机数据不会被展示。例如：主机

参数	描
主机输	要在下构件中显示的主机。输入名称时会自动出现与之匹配的主机下拉菜单.如果没有输入主机，将展示所有主机。

参数	描
异常	你以通过异常名称来限制异常主机的展示数量。如果你输入一个字符串，则只会显示名称与输入字符串匹配的异常主机。宏变量不可用。

参数	描
严重性标记	在下构件中显示的问题严重性。

参数	描
标签指	标签名称和值来限制图表上显示的异常数量。要添加更多标签名称和值，请单击添加。几个条件之间有两种计算类型：*且/或(And/Or)*-和/或-必须满足所有条件，具有相同

参数	描
展示选	展示选项： 主机组 **(Host groups)** - 显示每个主机组的异常。 总计 **(Total)** - 在与异常严重性相对应的彩色块中展示所有选定主机组的异常总数。

参数	描
布局选	布局选项：水平 **(Horizontal)** - 水平展示总计彩色块。垂直 **(Vertical)** - 垂直展示总计彩色块。如果选择“总计”作为“展示”选项，则此字段可编辑。

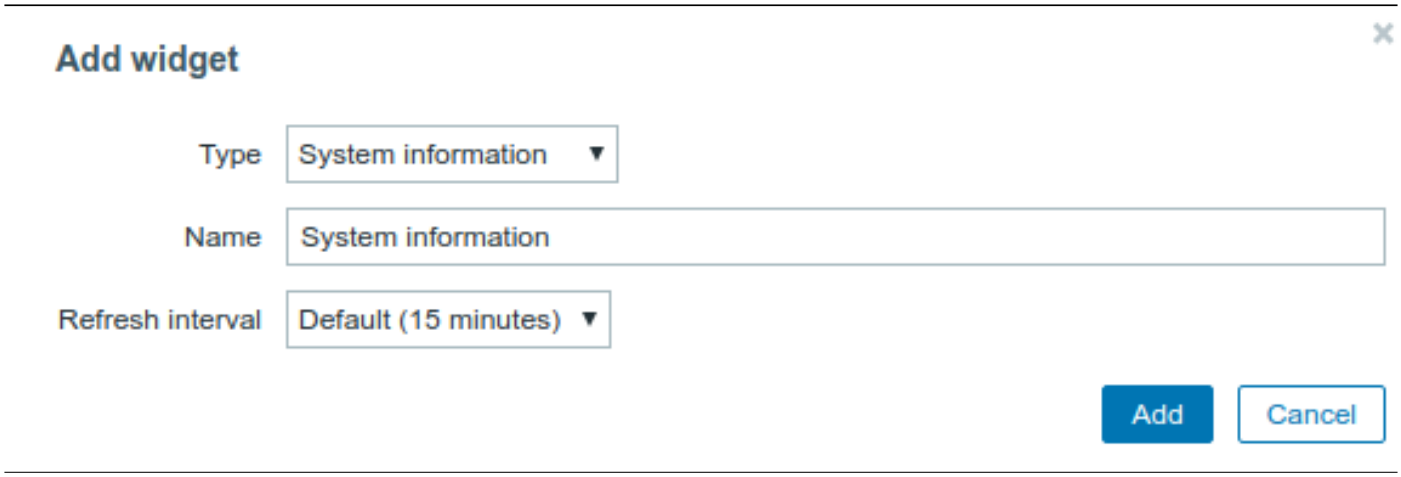
参数	描
显示被抑制的问题	选中该复选框来显示由于主机在维护期而被抑制(未显示)的问题。
隐藏没有异常的组	选中 * 隐藏没有问题的组 * 选项，在该构件中隐藏没有问题的组。
展示运营数据	选中该复选来显示运营数据(请参阅监视→问题中的运营数据说明)。

参数	描
显示异常	显示异常统计： - 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。
显示时间轴	选中该复框以显示可视时间轴。

系统信息

在“系统信息”小构件中，您可以显示高级 Zabbix 和 Zabbix 服务器信息。

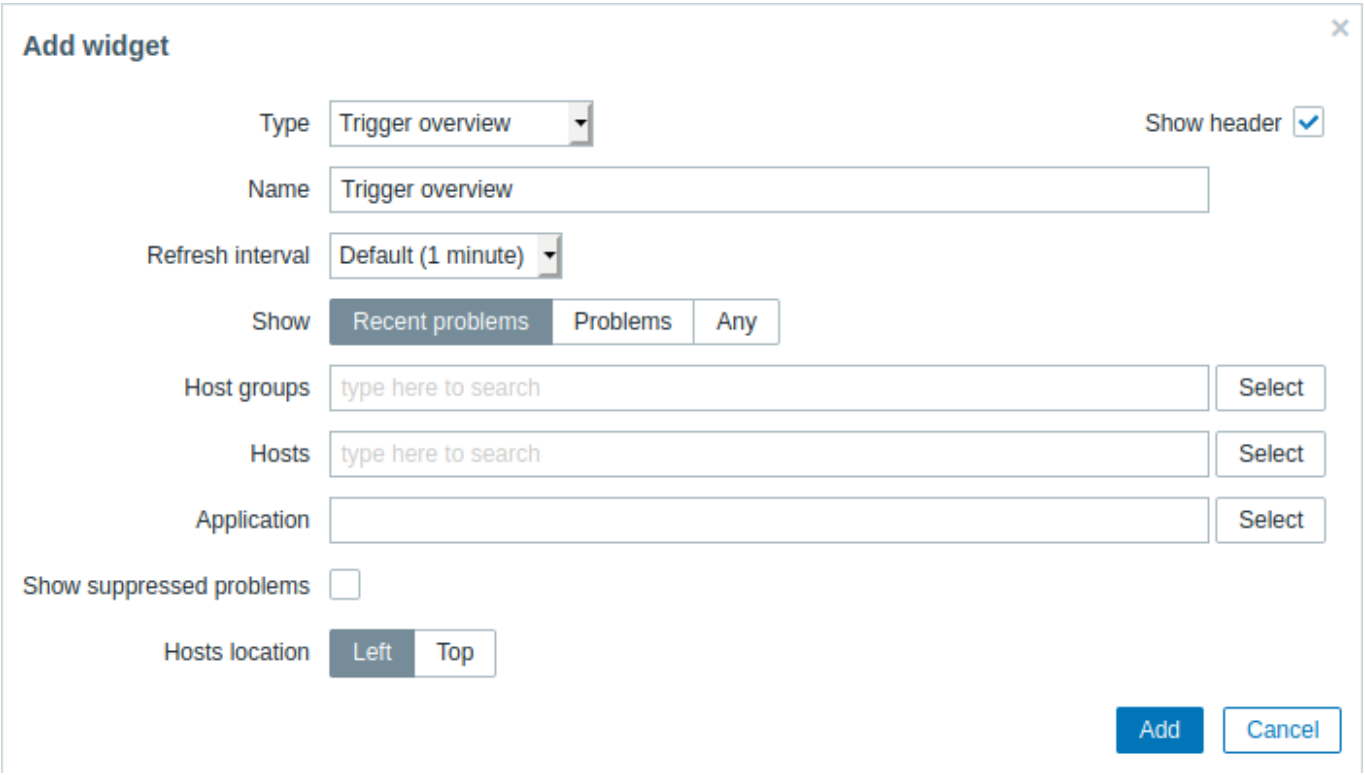
选择系统信息作为配置类型:



触发器概述

在触发器概述窗口小部件中，您可以显示一组主机的触发器状态。它的信息是从 监测中 → 概览复制过来的（当触发被选为 Type 时）。

选择触发器概览作为配置类型:



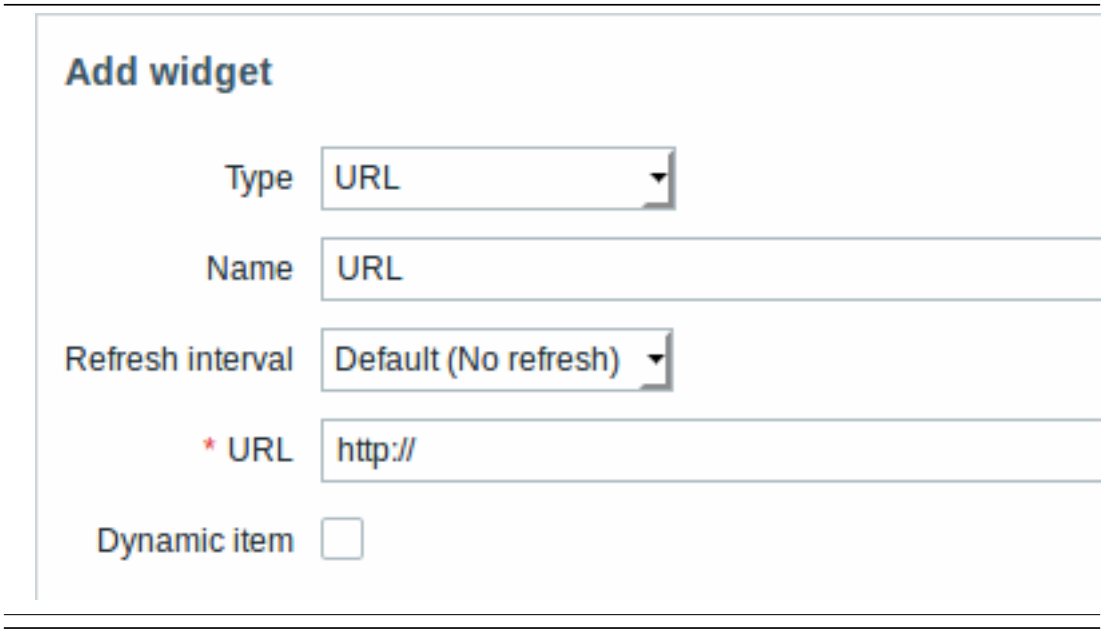
您可以设置以下特定选项：

查看按	题状态过滤: 最近的问题 - 显示未解决和最近解决的问题（默认） 问题 - 显示未解决的问题 任何 - 显示所有历史事件
主机组选择	机组。输入组名称时会自动出现与之匹配的主机组下拉菜单。
主机选	主机。输入名称时会自动出现与之匹配的主机下拉菜单。向下滚动选择。点击 'x' 移除选择。
应用输	应用名。
显示被抑制的问题选中该复选框来	示由于主机在维护期而被抑制（未显示）的问题。
主机位置选择主	位置 - 左侧或顶部。

URL

该小构件中，您可以展示来自外部资源的 URL 内容。

选择 *URL* 作为配置类型:



您可以设置以下特定选项：

URL	输入要显示的 URL。 URL 必须是 `http://`开头。 支持 {HOST.*} 宏。
动态监控项 根据所选	机显示不同的 URL 内容 如果在 URL 中使用了 {HOST.*} 宏，则可以使用此功能。

<note important> 如果是通过 HTTPS 访问 Zabbix 前端，浏览器可能无法加载 HTTP 页面。:::

Web 检测

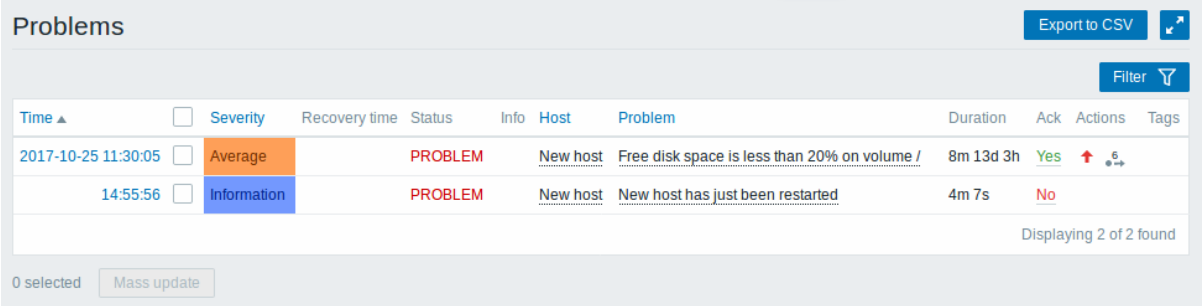
该构件用来展示监控 active web 场景的状态摘要。

Note:
如果用户无权访问某些小构件元素，则在小构件的配置过程中，该元素的名称将显示为// 无法访问。这会导致出现不可访问的项目，不可访问的主机，不可访问的组，不可访问的地图和不可访问的图形，而不是元素的“真实”名称。

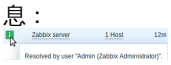
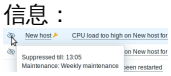
2 问题

概览

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	开始时间。

参数功	说明
//严重等级 (Severity) // 显示异	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格背景色。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关闭，则会显示绿色信息图标。将鼠标移动到该图标会显示更多详细信息： 
	如果显示抑制的问题（请参阅过滤器中的 “显示抑制的问题” 选项），则会显示以下图标。将鼠标移动到该图标会显示更多详细信息： 
主机（*Host*）显示异	的主机。

参数功	说明
问题（*Problem*）显示问	名称。 问题名称取决于其触发器的问题名称。 发生问题时，会解析触发器名称中的宏，并且解析的值不再更新。 注意，可以在问题名称后附加显示一些监控项最新值的操作数据。 单击问题名称将打开事件菜单。 将鼠标悬停在问题名称之后的 图标上，将显示触发器说明（针对存在问题的触发器）。 (需要注意，触发器描述中的宏{ITEM.VALUE}和{ITEM.LASTVALUE}解析值被截断为20 个字符。要查看整个值，你可以将宏函数与这些宏配合使用，例如：{ITEM.VALUE}.regsub("(.*)" \1)}, {ITEM.LASTVALUE}.regsub("(\1)} 作为解决办法。）
操作数据（*Operational data*）显示包含监	项最新值的操作数据 如果在触发级别上配置，则操作数据可以是文本和监控项值宏的组合。如果在触发级别上未配置任何操作数据，则显示表达式中所有监控项的最新值。 只有 “在过滤器中显示运行数据” 选择为 *Separately* 时，才显示此列。
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。 如果你点击链接将跳转到问题确认 可以对显示的问题进行简单的处置，包括注释和确认问题。

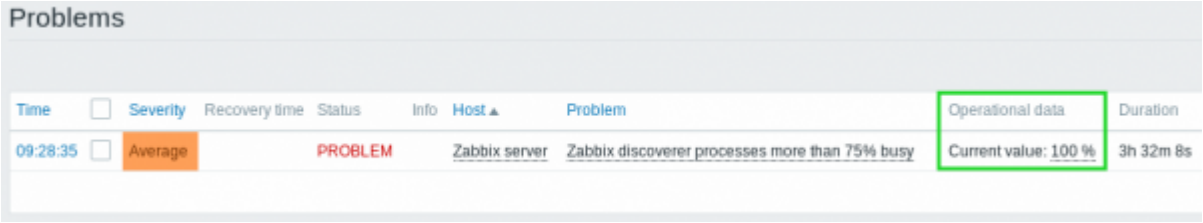
参数功	说明
动作（*Actions*）使用符	标记有关问题的活动的历史记录： - 显示已经更新的描述数量信息。 -问题的告警级别提高 (例如：信息级别 → 告警级别) - 问题严重级别下降 (例如：警告 → 信息) - 问题的严重级别发生过变化，但是目前回归到初始问题级别。(例如：警告 → 信息 → 警告) - 已经触发动作，并且显示当前触发的动作数。 - 动作操作正在进行中，显示当前操作数量进度。 - 动作进行过程中至少有 1 次的动作发生失败。 当鼠标移动到图标时会显示当前的动作信息，更多内容请参见查看详情
标记（*Tags*） [时间	签](/zh/manual/config/triggers/ever 显示时间标签 (如果存在)。 此外，还可以显示来自外部票务系统的标签 (配置Webhooks时，请参阅 “处理标签” 选项)。

问题的操作数据

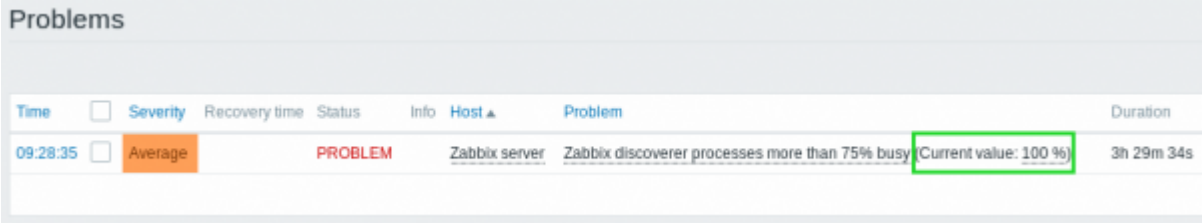
可以显示当前问题的操作数据，即最新的项目值，而不是出现问题时的项目值。

在监视 → 问题过滤器中或者在相应的仪表板小部件的配置中，通过选择以下三个选项之一来配置操作数据显示：

- None - 不显示操作数据
- Separately - 操作数据显示在单独的列中



- With problem name - 操作数据将附加到问题名称和括号中。仅当触发器配置中的 “操作数据” 字段为非空时，才会将操作数据附加到问题名称中。



可以在 “操作数据” 字段中为每个触发器配置操作数据的内容。该字段接受带有宏的任意字符串，最重要的是宏 `{ITEM.LASTVALUE <1-9>}`。

此字段中的 `{ITEM.LASTVALUE <1-9>}` 将始终解析为触发器表达式中各项的最新值，此字段中的 `{ITEM.VALUE <1-9>}` 将在触发状态更改时解析为监控项值 (即: 变成 Problem，变成 OK，被用户手动关闭或被关联关闭)。

消极的问题持续时间

在某些情况下，可能会出现具有消极的持续时间，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。并且会出现问题持续时间为负数。；
- 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，还将显示消极问题持续时间。

Note:
消极问题持续时间不以任何方式影响SLA 计算或特定触发器的可用性报告；它既不会减少也不会延长问题时间。

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。

按钮

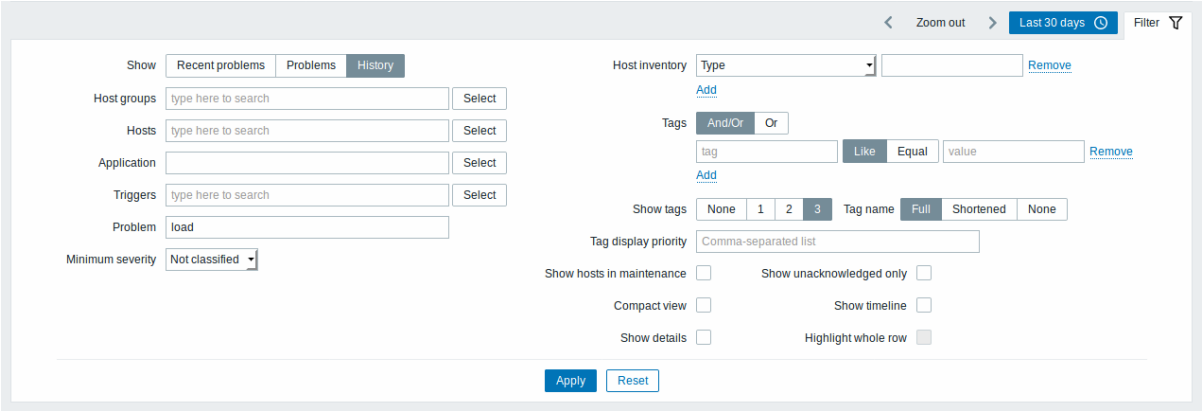
右侧的按钮提供以下选项：

Export to CSV	将所有页面的内容导出到 CSV 文件。

在监视页面上介绍了所有部分共有的查看模式按钮。

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。
//严重等级 (Severity) // 按触发	(问题）严重性过滤。
//年龄小于 (Age less than) // 按问题	年龄过滤。
//主机资产记录 (Host inventory) // 按资产记录	型和值进行筛选。

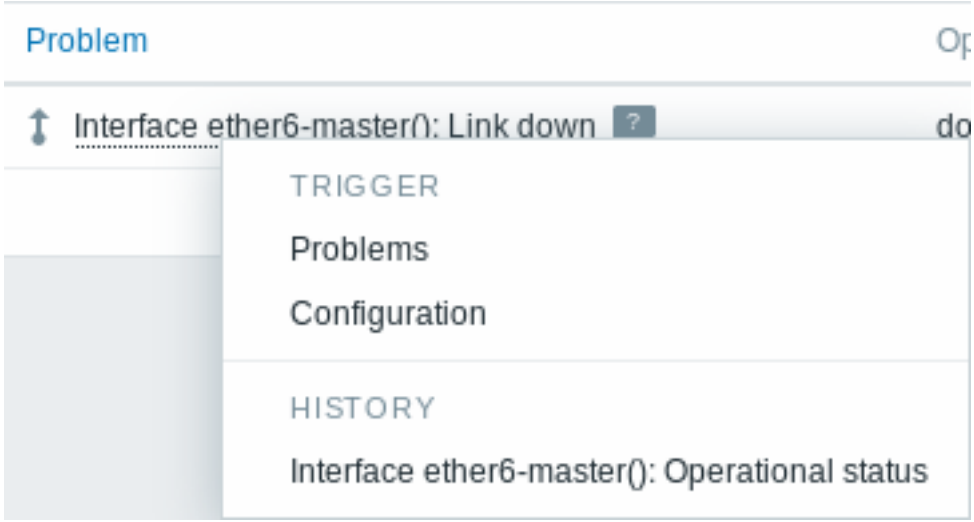
参数功	说明
//标签 (Tags) // 按	件标签名称和值进行筛选。 可以设置多个条件，条件中可以增加判断。 和（**And**）/或者（**Or**）- 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 或者（**Or**）- 满足其中一条即可。 匹配表标记值的方法有两种： (类似）**Like** - 模糊类型的字段匹配 等于（**Equal**） - 精确匹配
显示标签（*Show tags*）选择显示的	签数量： 无或空（**None**）- 没有 标签的监控问题 监测 → 问题 **1**- 标签列包含一个标签 **2**- 标签列包含两个标签 **3**- 标签栏包含三个标签 要查看问题的所有标记，请将鼠标悬停在三个点图标上。
//标签名称 (Tag name) // 选择标	名称显示模式： **Full** - 完整显示标签名称和值 **Shortened** - 标签名称缩短为 3 个符号；标签值完整显示 **None** - 仅显示标签值；没有名字

参数功	说明
//标签显示优先级 (Tag display priority) // 输入问题的标	显示优先级，以逗号分隔的标签列表形式 (例如：`Services,Applicatio` 只能使用标签名称，不能使用任何值。该列表的标签将始终被首先显示，而不是按字母自然排序。
//显示操作数据 (Show operational data) // 选择显示 [	作数据](#operational_data_of 的模式: **None** - 不显示操作数据 **Separately** - 在单独的列中显示操作数据 **With problem name** - 使用括号将操作数据附加到问题名称
//显示抑制的问题 (Show suppressed problems) // 选中该复选框	显示由于主机维护期而被一直的问题（未显示）。
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。
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*)。

参数功	说明
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。

事件菜单

单击问题名称将打开事件菜单：

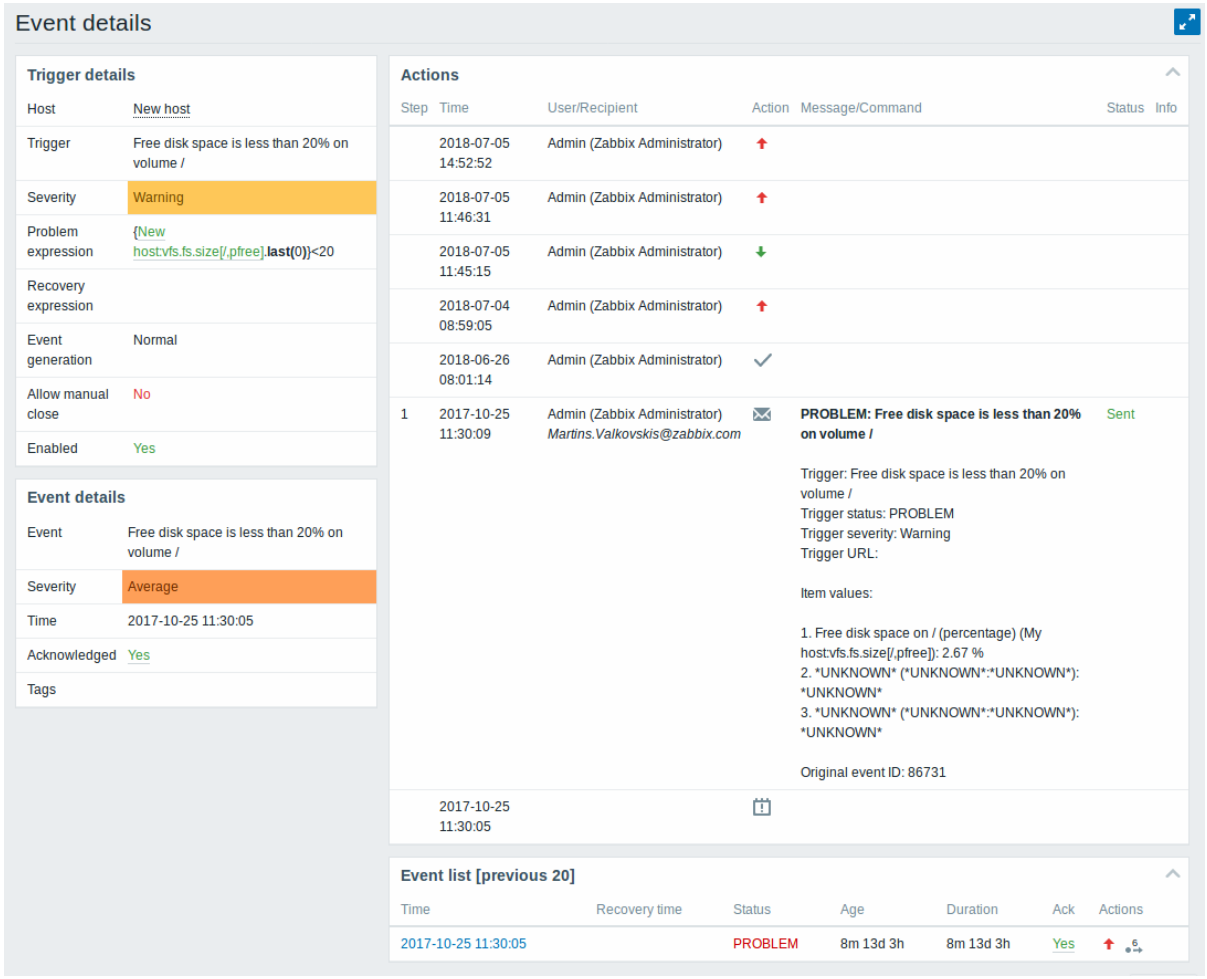


事件菜单允许：

- 过滤问题触发器
- 访问触发器配置
- 访问基础监控项的简单图形/监控项历史记录
- 访问问题的外部票据（如果配置了票据的话，请在配置webhook 时查看//Include event menu entry //菜单项选项）

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

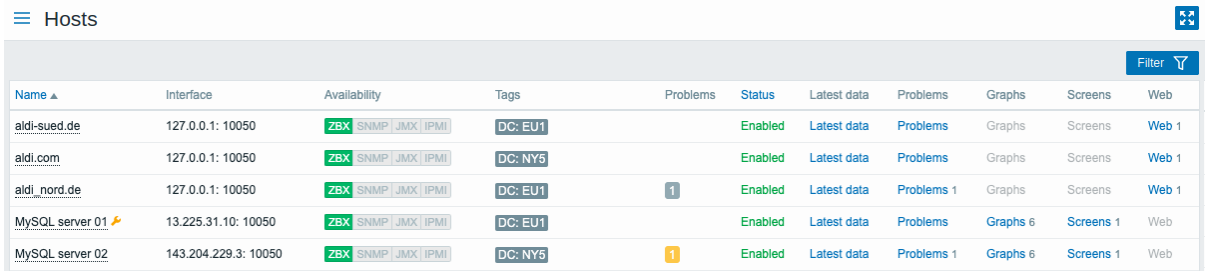
在操作列表中，以下图标用于表示活动类型:

- - 生成问题事件
- - 信息已发送
- - 已确认问题事件
- - 未确认问题事件
- - 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 主机

概览

检测中 → 主机部分是一张展示了被监控主机关于接口、可用性、标签、当前问题、状态（启用/停用）详细信息的完整表单，同时有一些链接可以方便的导航到对应主机的最新数据、历史问题、图表、聚合图形以及 WEB 场景。



列名	描
主机名主机	可见名称。点击这个名称会打开主机菜单。 一个主机名后的橘色的扳手图标表示此主机正在维护中。
接口展	单击此列的标题行将按主机名的升序（默认）或降序对主机进行排序。 主机的主要接口

列名	描
可用性展示	主机的可用性。四个图标分别代表一个受支持的接口(Zabbix agent、SNMP、IPMI、JMX)。使用对应的颜色表示接口状态：绿色 - 可用红色 - 不可用(鼠标悬停时，将显示无法访问该接口的详细信息)灰色未知或接口未配置请注意，主动模式的Zabbix agent监控项，不会影响主机接口的可用性及所有关联模板的的标签，标签中的宏无法被解析
标签主	

列名	描
问题展	当前开放问题数量的方形图标。图标的颜色表示了问题的严重性。数字代表了对应的严重性当前有多少问题。使用过滤器选择包含被抑制的问题，相关数据将会包含被抑制的问题数量(默认不包含)。
状态展	主机状态 - 启用或停用。单击此列的标题行将按状态的升序或降序对主机进行排序。
最新数据点击这	按钮将打开检测中 - 最新数据页，展示自该主机收集上来的最新数据的。

列名	描
问题点	这个按钮将打开 检测中 - 问题 页，展示只包含指定主机的相关信息。
图表点	这个链接将会展示配置在这台主机的图表。图表的数量以灰色显示。如果主机没有配置图表，则该列的链接将被禁用(灰色文本显示)，并且不会显示任何数字。
聚合图形点击这	链接将会展示配置在这台主机的聚合图形。聚合图形的数量以灰色显示。如果主机没有配置聚合图形，则该列的链接将被禁用(灰色文本显示)，并且不会显示任何数字。

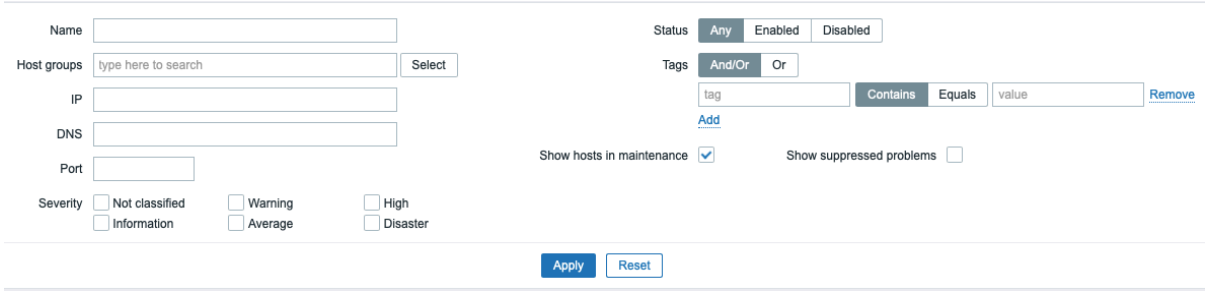
列名	描
WEB 场景点	这个链接将会展示配置在这台主机的 WEB 场景。WEB 场景的数量以灰色显示。如果主机没有配置 WEB 场景，则该列的链接将被禁用（灰色文本显示），并且不会显示任何数字。

按钮

文档检测中展示了通用的显示模式按钮的介绍

使用过滤器

你可以使用过滤器来筛选出你感兴趣的主机。过滤器位于表单的顶部。你可以通过主机名、主机组、IP 与 DNS 名、接口端口、标签、问题严重性、状态（启用/停用/任何）来过滤主机；你也可以选择是否展示被抑制的告警和主机是否处于维护中。



注意:

- 名称 and 主机组字段支持一次填写多个值。
- 指定父主机组将隐式选择所有嵌套主机组。
- 默认情况下，显示所有严重性的问题（如果未被抑制）。
- 默认情况下，显示维护中的主机，但不显示这些主机上受抑制的问题。
- 如果使用 严重性过滤器，并且 展示被抑制的问题前的复选框未选中，则不会显示具有指定严重性的抑制问题的主机。
- 主机可以通过主机级标签以及所有链接模板（包括父模板）中的标签进行过滤。

1 图表

概览

主机图表（host graphs）可以从// 检测中（Monitoring）→ 主机（Hosts）// 点击相应主机的图表（graphs）链接。

任何自定义图表 被配置在主机，就会被显示。但一次不会展示超过 20 张图表。

// 展示为 // 下拉菜单选项可以选择让数据以图形或数值形式展示。对于处于禁用状态的主机也可以访问其图表

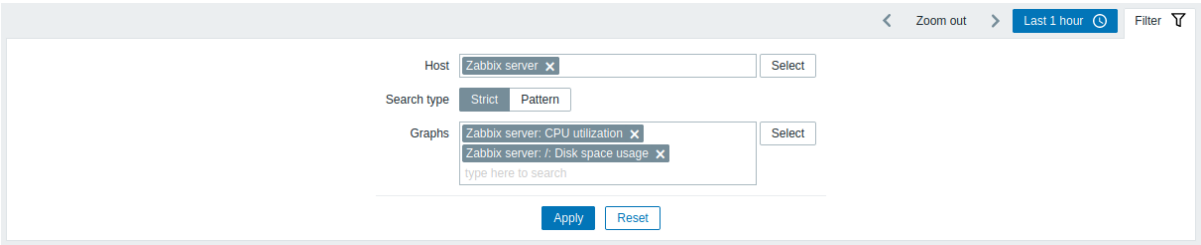
时间段选择器

注意图表上方的时间段选择器。通过单击鼠标，可以快速选择常用的的时间段。

查看更多介绍：时间段选择器

使用过滤器

为了查询特定的图像，通过使用过滤起来选中它。该过滤器允许一次指定一个主机（主机是必需的），然后通过从列表中选择或通过按图表名称模式的方法来快速搜索指定主机图表。



按钮

在顶部右侧有以下按钮：

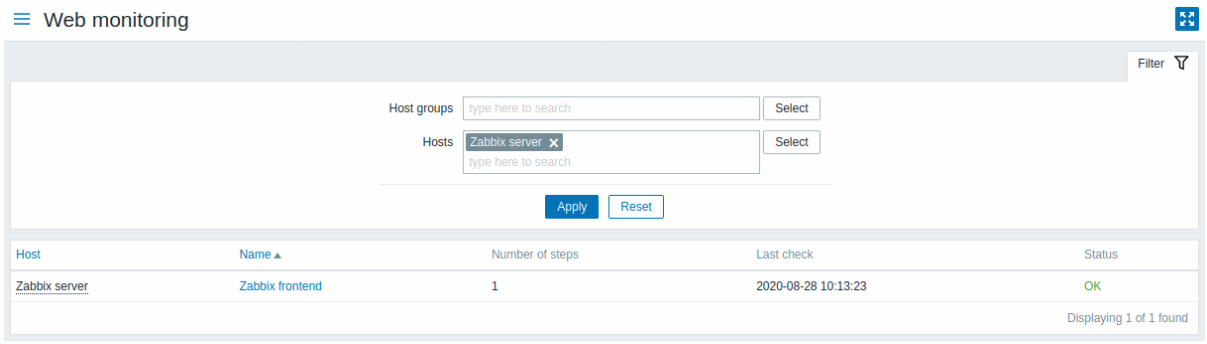
	将该图表添加到仪表板的收藏的图表构件中。
	该图表已经在仪表板的收藏的图表构件中。点击即可将其移除。

文档检测中展示了通用的显示模式按钮的介绍。

2 WEB 场景

概览

主机的WEB 场景 信息可以通过 检测中 *(Monitoring)* → 主机 *(Hosts)* 页面点击对应主机记录的 *Web* 场景链接按钮。



这个页面展示了全部的所选主机配置的 WEB 场景的列表。要查看另一台主机或主机组的 WEB 场景，又不想回到 检测中 *(Monitoring)* → 主机 *(Hosts)* ，你可以通过主机或主机组的过滤器进行筛选。

对于处于禁用状态的主机，其 WEB 场景的监控数据依旧可以访问。但请注意，这类主机的名称将会是红色字体。

每页所能展示的最多场景的数量，靠用户选项设定中，每页行数参数控制。

默认情况下，仅显示过去 24 小时内的值。引入此限制的目的是为了缩短 WEB 监控大页面的初始加载时间。也可以通过更改 *include/defines.inc.php* 文件中的限制常量ZBX_HISTORY_PERIOD

WEB 场景名称可以链接到展示其更详细状态的页面：

按钮

在文档检测中介绍了通用的显示模式按钮。

4 概览

概览

在监视 → 概览部分可能显示以下任一内容：

- 触发器概述 - 触发器状态的概述
- 数据概述 - 一次比较多种主机数据

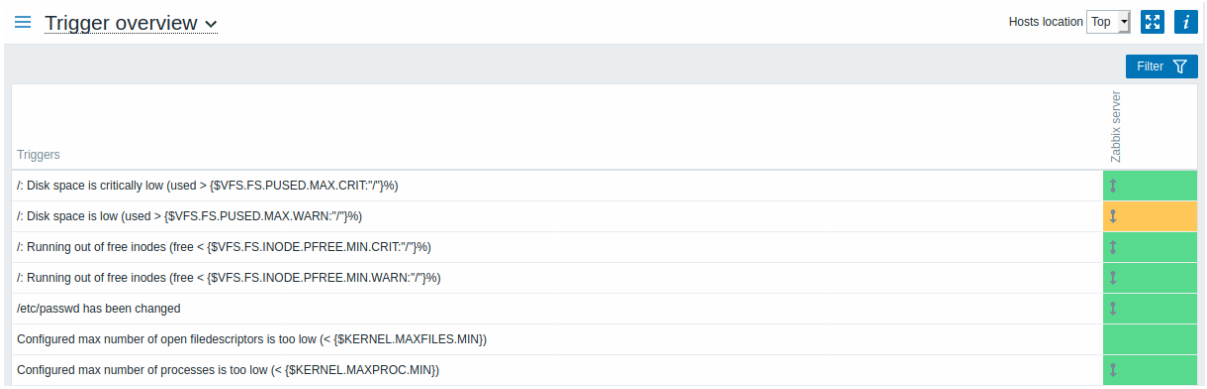
可用的附加显示选项：

- 在主机位置下拉菜单中选择主机名在表格顶端或表格左侧显示

请注意，显示记录固定限制为 50 条。没有分页。如果存在很多记录，表格底部会显示一条消息，要求提供更具体的过滤条件。

触发器概述

在下一个屏幕截图中，选择触发器概述。结果是，本地主机的触发状态以彩色块显示（问题触发器的颜色取决于问题严重性颜色，可以在问题更新 屏幕中调整):

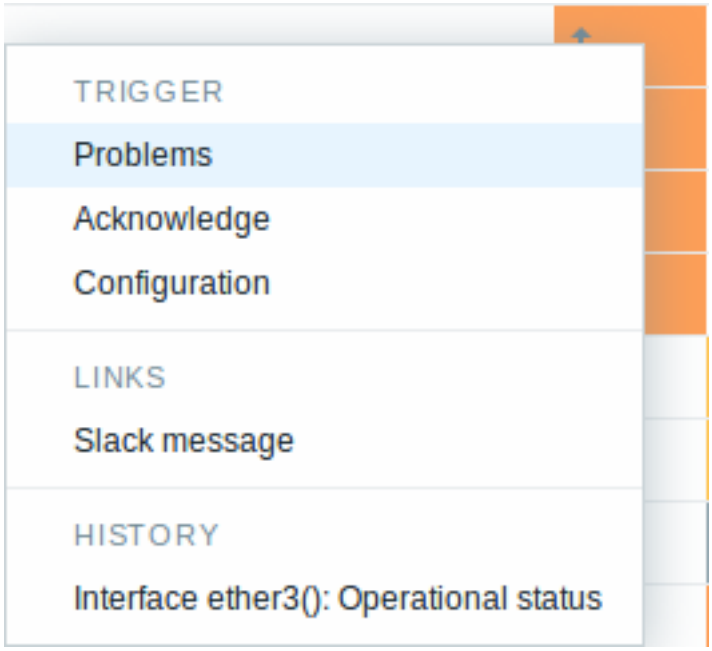


请注意，最近变化的触发器（最近 2 分钟内）会显示为闪烁的块。

蓝色的向上和向下箭头指示具有依赖性的触发器。鼠标悬停时，将显示依赖项详细信息。

复选框图标表示已确认的问题。所有问题或已解决的问题的触发器必须确认，才能显示此图标。

单击触发器块会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/最新值列表相关的具体情况链接。



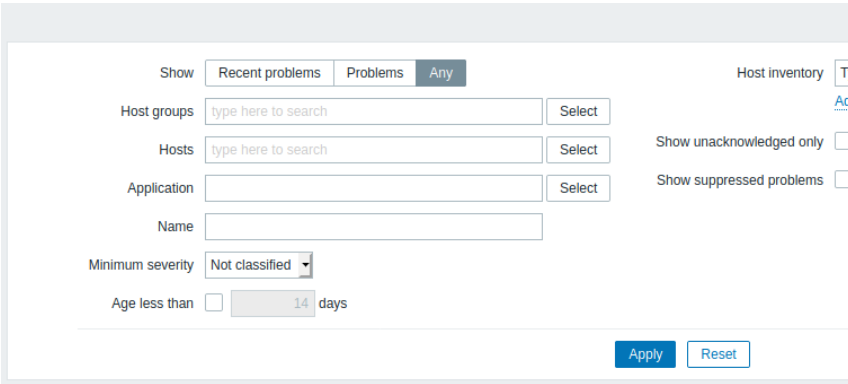
按钮

右侧的按钮有以下选项：

如果将鼠标悬停在此按钮上，则会显示显示页面内容的其他信息。

在文档检测中介绍了通用的显示模式按钮。

使用筛选器



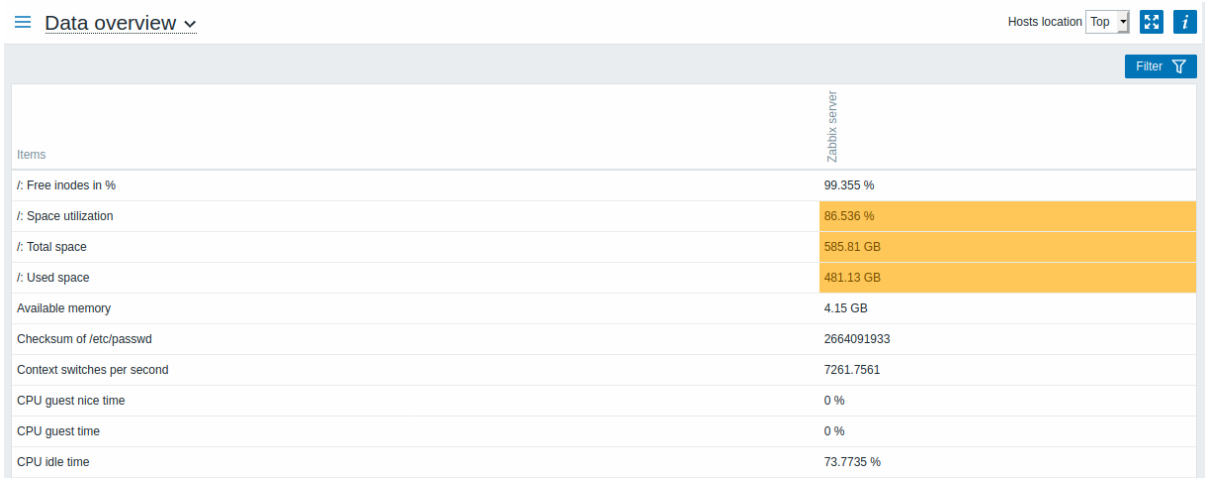
您可以使用筛选器仅显示您感兴趣的问题。筛选器位于表格上方。

参数	描
查看按	题状态过滤：最近的问题 - 显示未解决和最近解决的问题（默认）问题 - 显示未解决的问题任何 - 显示所有事件的历史记录
主机组按主	组过滤。
主机按	机过滤。
应用集按应	集过滤。
名称按	题名称过滤。
最低严重性按最低问	严重性过滤。

参数描	
持续时间小于勾选复选框	问题持续时间过滤。
主机资产按资产	型和值过滤。
仅显示未确认勾选该复选	则仅显示未确认的问题。
显示处理的问题选中该复选框	展示由于主机在维护期而被抑制(未显示)的问题。

数据概述

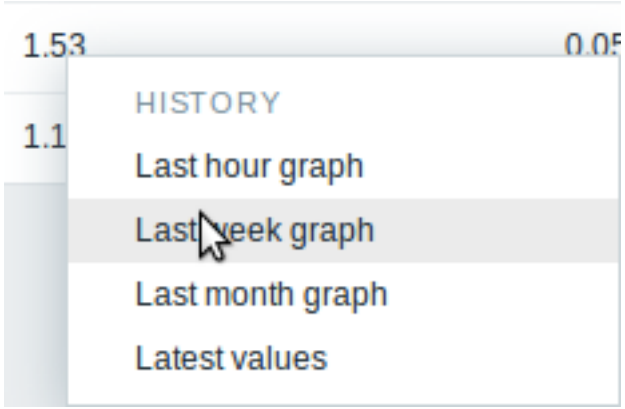
在下一个屏幕截图中，选择数据概述。结果是，显示本地主机的监控项数据。



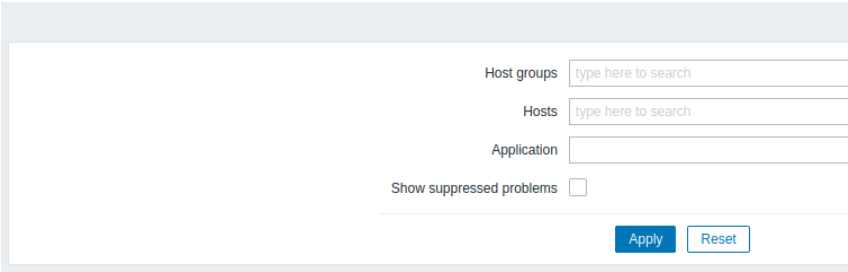
问题项的颜色基于问题严重性颜色，可以在问题更新 中调整.

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。可以通过更改 *include/defines.inc.php* 中 ZBX_HISTORY_PERIOD常量的值来更改此限制。

单击一条数据会出现指向某些预定义图形或最新值的链接。



使用筛选器

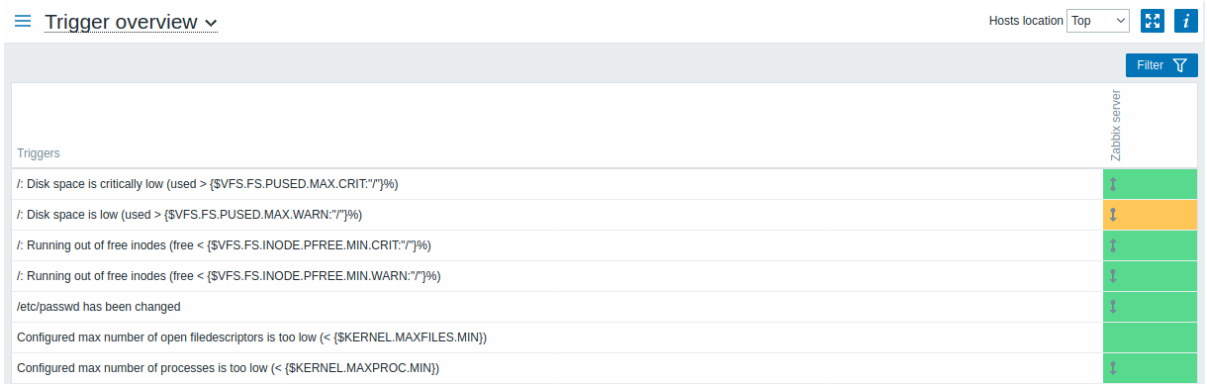


您可以使用筛选器仅显示您感兴趣的问题。筛选器位于表格上方。

参数描	
主机组按主	组过滤。
主机按	机过滤。
应用集按应	集过滤。
显示处理的问题选中该复选框	展示由于主机在维护期而被抑制（未显示）的问题。

Overview of triggers

In the next screenshot *Trigger overview* is selected. As a result, the trigger states of a local host are displayed as colored blocks (the color of problem triggers depends on the problem severity color, which can be adjusted in the problem update screen):

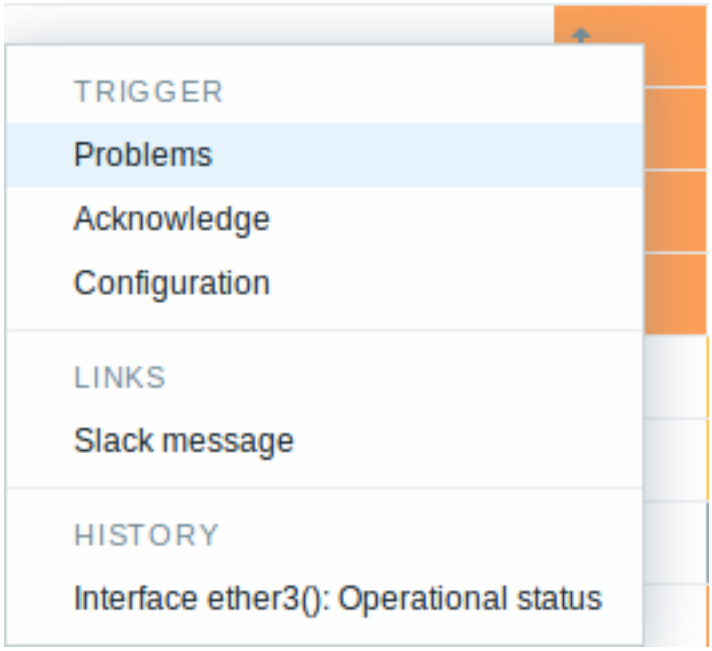


Note that recent trigger changes (within the last 2 minutes) will be displayed as blinking blocks.

Blue up and down arrows indicate triggers that have dependencies. On mouseover, dependency details are revealed.

A checkbox icon indicates acknowledged problems. All problems or resolved problems of the trigger must be acknowledged for this icon to be displayed.

Clicking on a trigger block provides context-dependent links to problem events of the trigger, the problem acknowledgment screen, trigger configuration, trigger URL or a simple graph/latest values list.



Buttons

Button to the right offers the following option:

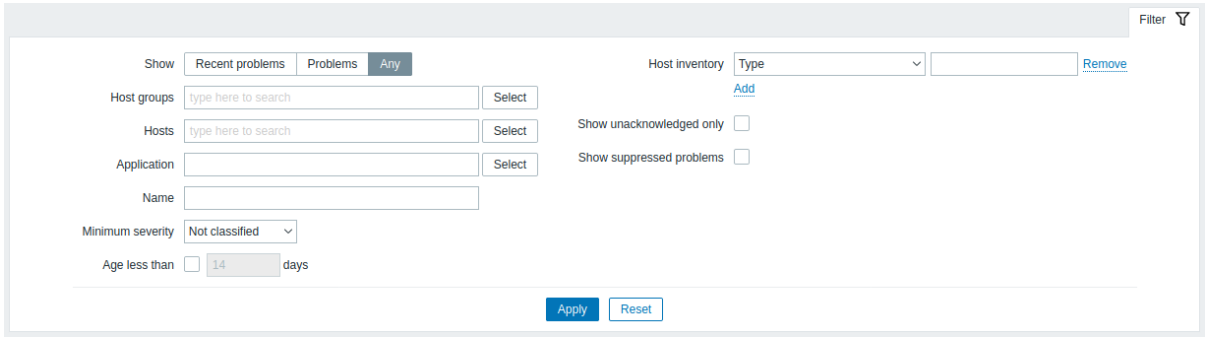
	Additional information on the page content is displayed if you roll the mouse over this button.

View mode buttons being common for all sections are described on the Monitoring page.

Using filter

You can use the filter to display only the problems you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table.

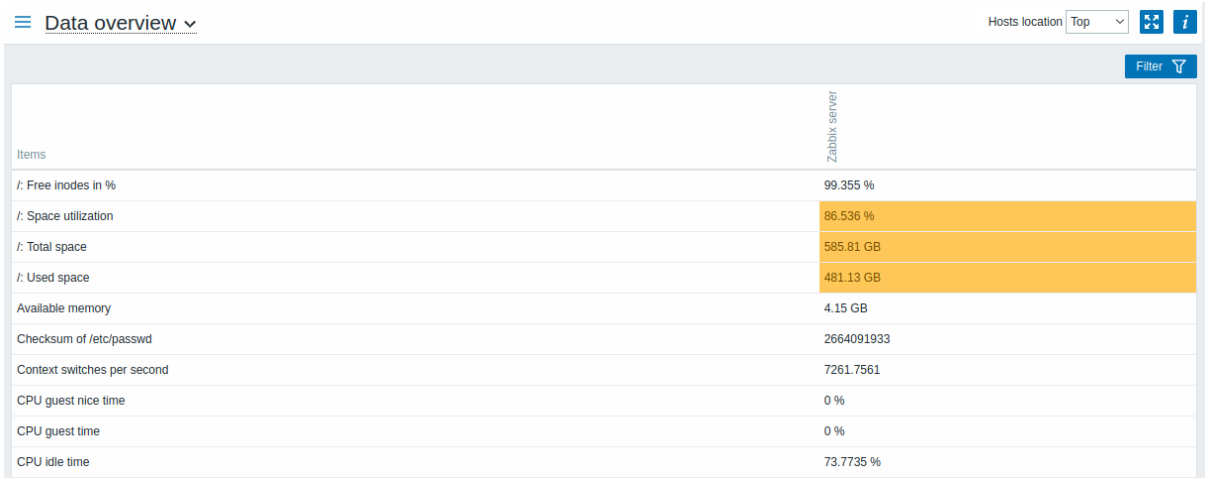


Parameter	Description
Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **Any** - history of all events is displayed
Host groups	Filter by host group.
Hosts	Filter by host.
Application	Filter by application.
Name	Filter by problem name.
Minimum severity	Filter by minimum problem severity.
Age (less than)	Mark the checkbox to filter by problem age.
Host inventory	Filter by inventory type and value.

Parameter	Description
Show unacknowledged only	Mark the checkbox to only display problems which are unacknowledged.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

Overview of data

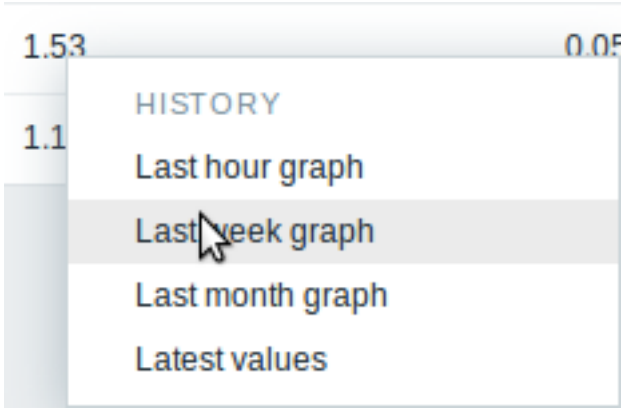
In the next screenshot *Data overview* is selected. As a result, item data of a local host is displayed.



The color of problem items is based on the problem severity color, which can be adjusted in the problem update screen.

Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. It is also possible to change this limitation by changing the value of ZBX_HISTORY_PERIOD constant in *include/defines.inc.php*.

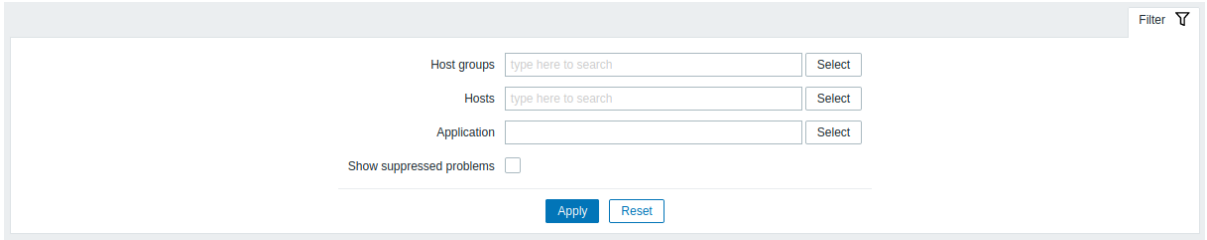
Clicking on a piece of data offers links to some predefined graphs or latest values.



Using filter

You can use the filter to display only the data you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table.



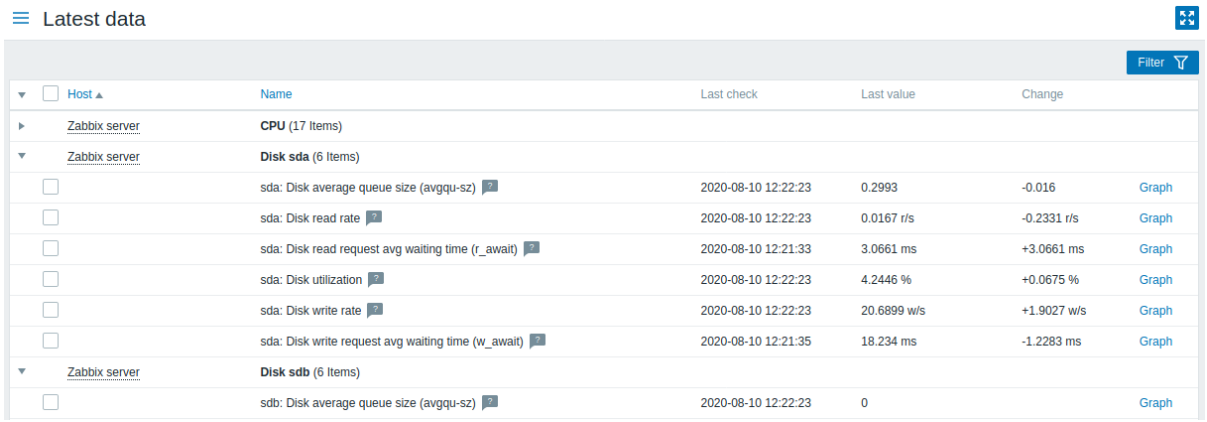
Parameter	Description
Host groups	Filter by host group.

Parameter	Description
Hosts	Filter by host.
Application	Filter by application.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

5 最新数据

概览

监测 → 最新数据可以用来查看监控项收集的最新值，以及访问监控项的各种图表。



在显示的列表中，单击主机和应用集前的 来展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集，进而，通过单击在行头部的 来展示所有监控项（请注意，Zabbix 5.0.2 中不支持应用集扩展/折叠。）

监控项按主机和应用集分组。显示监控项名称，上次检查时间，最新值，变化量以及指向监控项值的简单图形/历史记录链接。

所有具有描述说明的监控项名称旁边都会显示带有问号的图标。如果将鼠标光标放在该图标上，在工具提示中展示监控项说明。

注意：禁用主机的名称显示为红色。在最新数据中也可以访问禁用的主机的数据，包括图形和监控项值列表。

默认情况下，仅显示过去 24 小时内的值。这样设置是为了优化页面加载数据的时间。可以通过更改 *include/defines.inc.php* 中 ZBX_HISTORY_PERIOD常量的值来更改此限制。

<note important> 对于采集间隔为 1 天或以上的监控项，永远不会显示变化量:::

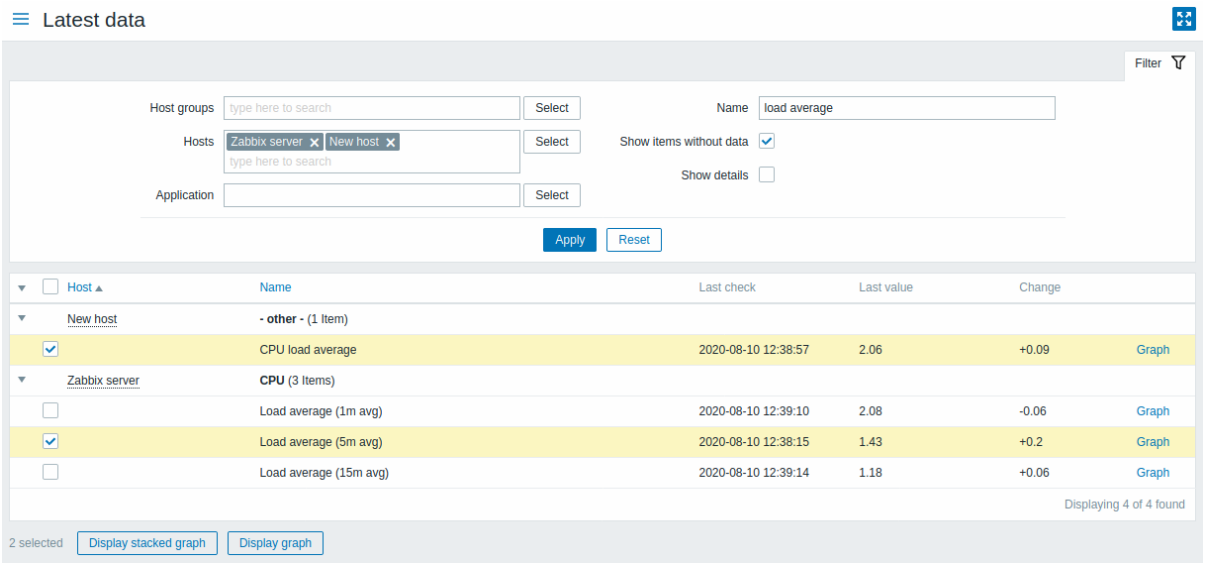
按钮在文档检测中介绍了通用的显示模式按钮。

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右上方。您可以使用它按主机组，主机，应用集，监控项名称中的字符串过滤监控项；还可以选择是否显示没有收集到数据的监控项。为了提高性能，无法使用宏进行过滤数据。

指定一个父主机组，隐式选择所有嵌套的主机组。

显示详细信息允许显示监控项上的扩展信息。诸如：刷新间隔，历史记录和趋势设置，监控项类型和监控项错误（良好/不支持）之类的详细信息。还提供指向监控项配置的链接。



默认情况下，会显示没有数据的项目，但不显示详细内容。为了更好的页面性能，从 Zabbix 5.0.3 开始，如果未在筛选器中选择主机，则显示没有数据的监控项选项被已勾选并禁用。

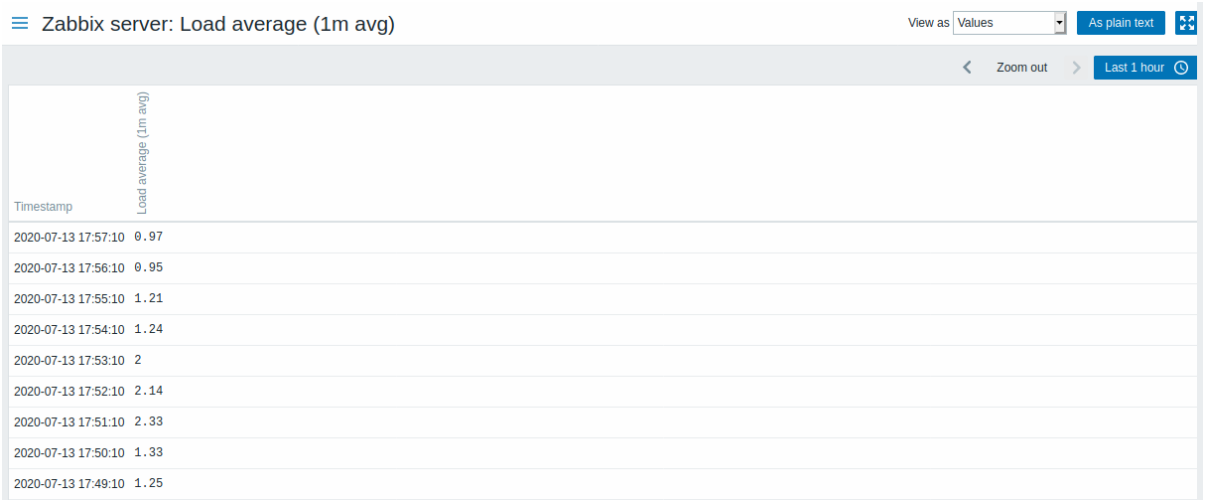
临时比较监控项图表

您可以使用第一列中的复选框选择多个项目，然后用简单或堆叠的临时图表比较它们的数据。要实现这样的效果，请选择感兴趣的监控项，然后单击表格下方所需的图表按钮。

链接到值的历史/简单图形

最新值在列表中的最后一列：

- ** 历史记录链接 ** （用于所有文本项）-链接到显示监控项历史记录的列表（值/最近 500 个值）。
- 图表链接（用于所有数字项）-链接到一个简单图。虽然已图表展示，还是可以通过右上角的下拉菜单切换到值/最近 *500* 个值。



此列表中显示的是“原始的”值，即指未经处理的值。

Note:
显示的记录总数由“搜索限制”和“过滤结果”参数值定义，在管理 → 一般/通用中设置。

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

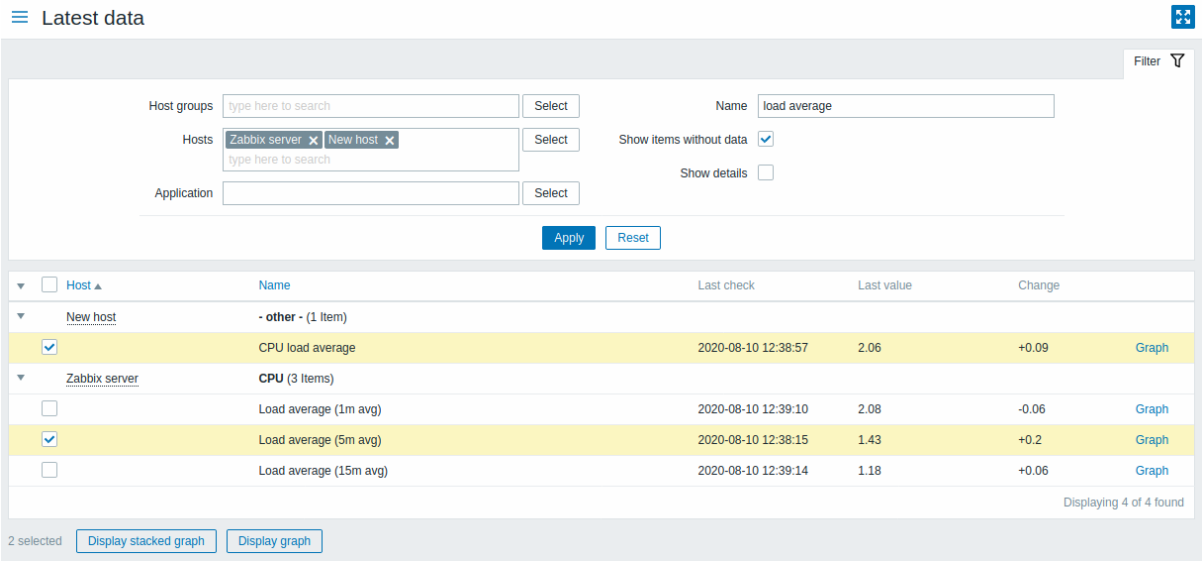
Using filter

You can use the filter to display only the items you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is located above the table to the right. You can use it to filter items by host group, host, application, a string in the item name; you can also select to display items that have no data gathered.

Specifying a parent host group implicitly selects all nested host groups.

Show details allows to extend displayable information on the items. Such details as refresh interval, history and trends settings, item type and item errors (fine/unsupported) are displayed. A link to item configuration is also available.



By default, items without data are shown but details are not displayed. For better page performance, since Zabbix 5.0.3, the *Show items without data* option is checked and disabled if no host is selected in the filter.

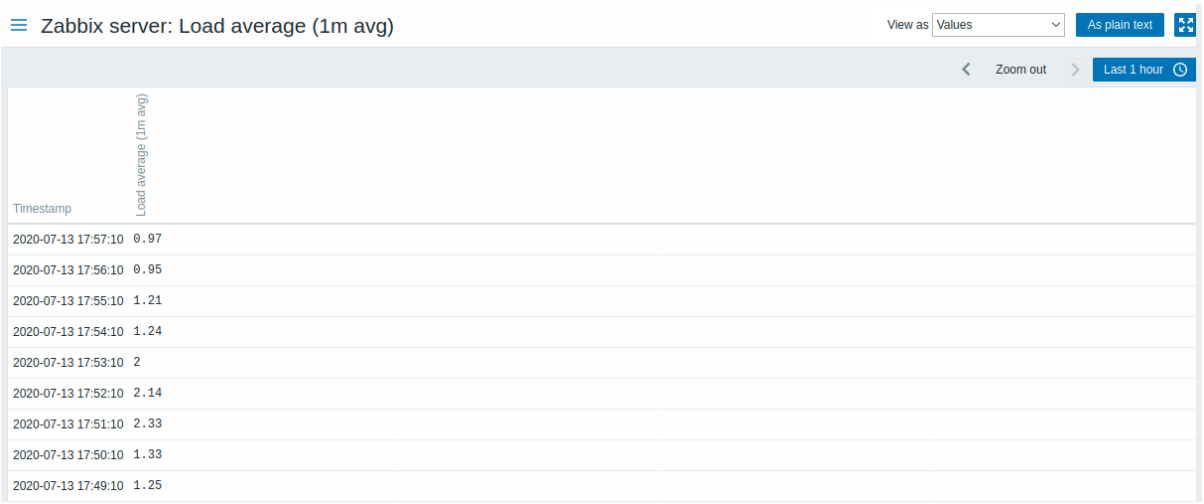
Ad-hoc graphs for comparing items

You may use the checkbox in the first column to select several items and then compare their data in a simple or stacked ad-hoc graph. To do that, select items of interest, then click on the required graph button below the table.

Links to value history/simple graph

The last column in the latest value list offers:

- a **History** link (for all textual items) - leading to listings (*Values/500 latest values*) displaying the history of previous item values.
- a **Graph** link (for all numeric items) - leading to a simple graph. However, once the graph is displayed, a dropdown on the upper right offers a possibility to switch to *Values/500 latest values* as well.



The values displayed in this list are ”raw”, that is, no postprocessing is applied.

Note:
The total amount of values displayed is defined by the value of *Limit for search and filter results* parameter, set in Administration → General.

6 聚合图形

概览

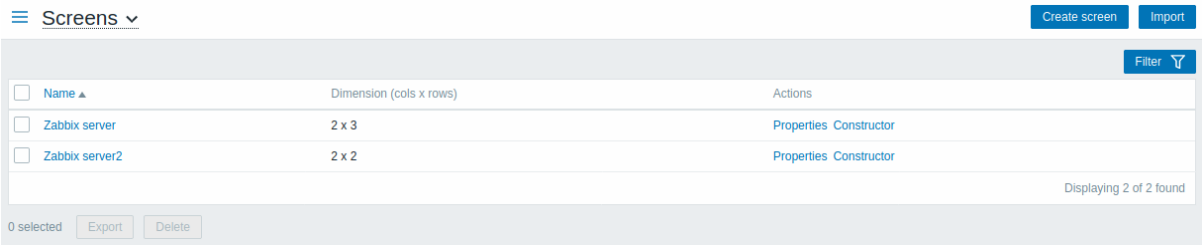
在 监测 → 聚合图形部分，您可以配置，管理和查看 Zabbix 聚合图形 (screens) 以及幻灯片演示 (slide shows)。

当您打开此部分时，您看到的是最后一个聚合图形/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。

所有聚合图形/幻灯片演示可以是公开的也可以是私有的。公开的可供所有用户使用，而私有仅可用于其所有者和共享的用户。

可以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

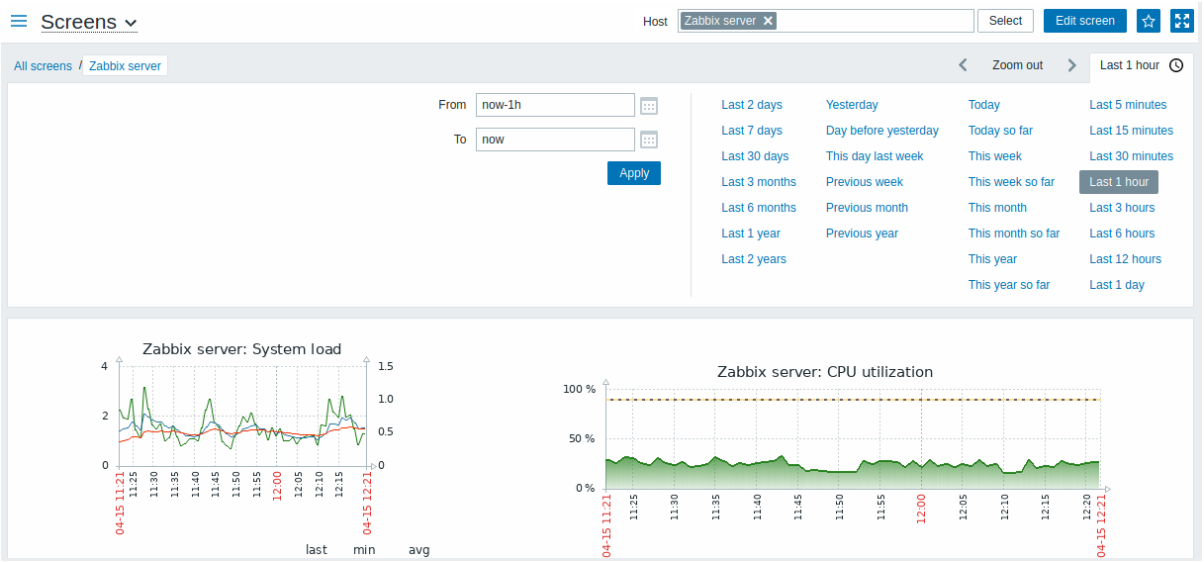
列表下方的按钮提供了一些批量编辑选项：

- 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选中复选框，然后单击所需的按钮。

查看聚合图形

单击聚合图形列表中的名称，即可查看聚合图形。



当聚合图形元素具有与主机相关的动态内容时，将提供一个用于选择主机的字段。

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。它允许您通过单击鼠标选择经常需要的时间段，从而影响图形中显示的数据。更多参见: 时间选择器

按钮

右侧的按钮有以下选项：

Edit screen	跳转到聚合图形构造器来编辑聚合图形
	把聚合图形添加到仪表板中的 “收藏夹” 小构件中
	在仪表板“收藏夹” 小构件中的聚合图形。单击会从 “收藏夹” 中移除聚合图形。

在文档检测中介绍了通用的显示模式按钮。

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：

列	述
名称（*Name*）幻灯片	示的名称. 点击名称查看幻灯片演示。
延迟（*Delay*）每个幻	片展示 & 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

要创建一个新的幻灯片, 请点击右上角的 创建幻灯片按钮。

批量操作选项

列表下方的按钮有一个批量编辑选项：

- 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选中复选框，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。

当聚合图形元素具有与主机相关的动态内容时，将提供一个用于选择主机的字段。

按钮

右侧的按钮提供以下选项：

右侧的按钮有以下选项：

Edit screen	跳转到聚合图形构造器来编辑聚合图形
	把聚合图形添加到仪表板中的 “收藏夹” 小构件中
	在仪表板“收藏夹” 小构件中的聚合图形。单击会从 “收藏夹” 中移除聚合图形。

	放慢或加快幻灯片放映。

在文档检测中介绍了通用的显示模式按钮。

引用聚合图形

可以通过 GET 的 `elementid` 和 `screenname` 参数来引用聚合图形。例如，

```
http://zabbix/zabbix/screens.php?screenname=Zabbix%20server
```

将打开名称为 Zabbix 服务器聚合图形。

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）优先级高。

Viewing slide shows

To view a slide show, click on its name in the list of all slide shows.

A field for selecting the host is available when there are screen elements with dynamic, host-dependent content.

Buttons

Buttons to the right offer the following options:

	Go to the slide show properties.
	Add slide show to the favorites widget in the Dashboard.
	The slide show is in the favorites widget in the Dashboard. Click to remove slide show from the favorites widget.
	Slow down or speed up a slide show.

View mode buttons being common for all sections are described on the Monitoring page.

Referencing a screen

Screens can be referenced by both `elementid` and `screenname` GET parameters. For example,

```
http://zabbix/zabbix/screens.php?screenname=Zabbix%20server
```

will open the screen with that name (Zabbix server).

If both `elementid` (screen ID) and `screenname` (screen name) are specified, `screenname` has higher priority.

7 拓扑图

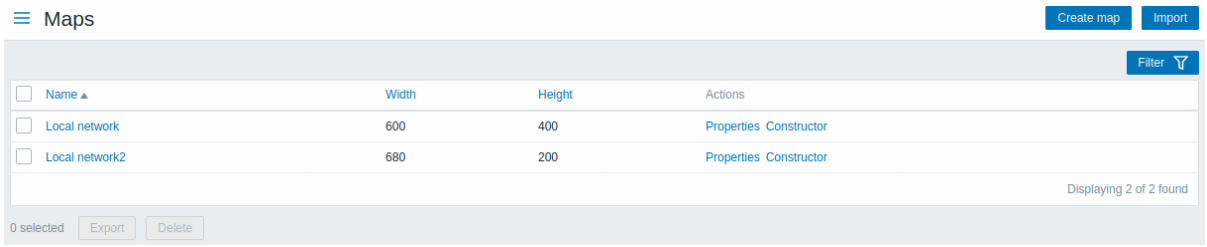
概览

在监测 → 拓扑图部分，您可以配置，管理和查看网络拓扑图。

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。

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，而私有拓扑图只能由其所有者和对其共享的用户访问。

拓扑图列表



显示的数据：

列	述
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*）两个可	操作: 属性 - 编辑拓扑图通用属性 构造函数 - 通过网格化来添加拓扑图元素

要配置新的拓扑图, 请点击右上角的// 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。

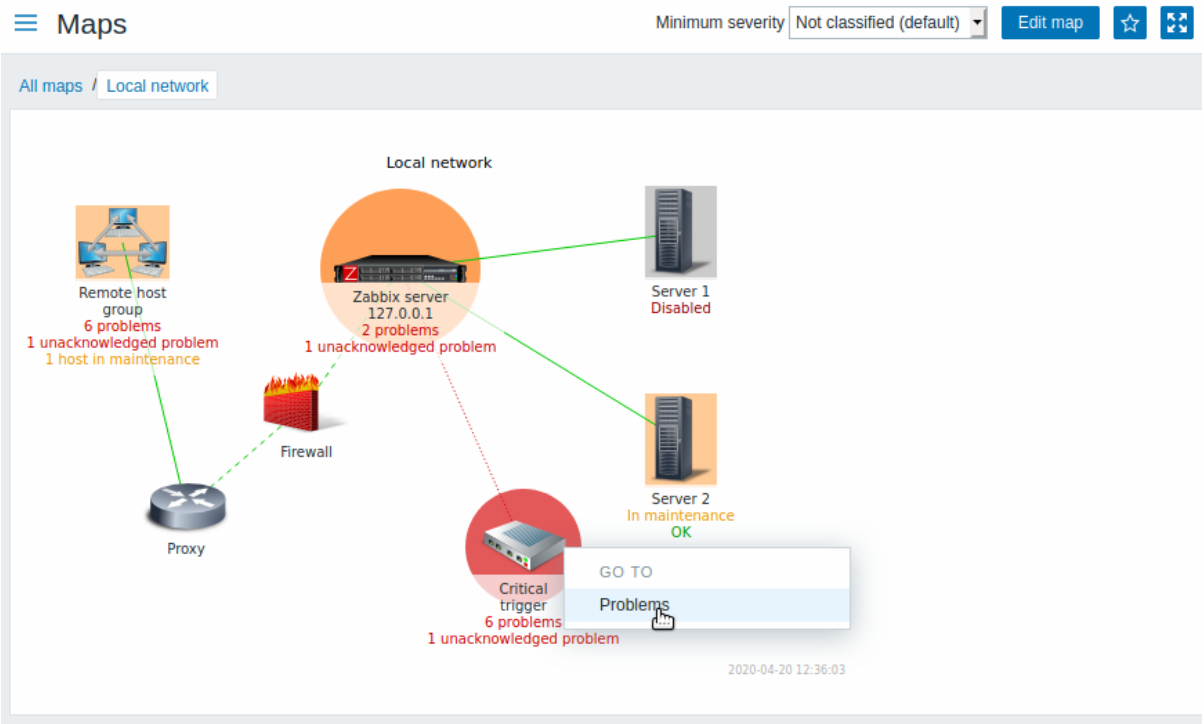
列表下方的两个按钮有一些批量编辑选项：

- 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，请选中各个拓扑图之前的复选框，然后单击所需的按钮。

查看拓扑图

要查看某个拓扑图，单击所有拓扑图列表中对应的名称。



您可以使用拓扑图标题栏中的下拉列表来选择要显示问题触发器的最低严重性级别。默认严重性是在拓扑图配置中设置的级别。如果拓扑图包含子拓扑图，则导航到子拓扑图将保留上层级别拓扑图严重性。

图标高亮显示

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。该元素仅展示选定严重性级别或更高级别的问题。如果所有问题都得到确认，圆形周围会显示一个加粗的绿色边框。

另外：

* 如果一个主机在维护状态，则以橙色背景块高亮显示。请注意，维护期高亮显示的优先级高于问题严重性高亮显示。* 禁用（未监视）主机以灰色背景块高亮显示。

如果在拓扑图配置中选中了图标高亮复选框，则图标会高亮显示。

最近更改的标记

元素周围向内指向的红色三角形表示最近的触发状态变化 - 最近 30 分钟内触发状态发生更改。如果在拓扑图配置中选择了 “触发器状态上的标记组件改变” 复选框，则会显示这些三角形。

链接

点击拓扑图元素会打开一个包含一些可用链接的菜单。

按钮

右侧的按钮有以下选项：

右侧的按钮有以下选项：

Edit screen	跳转到聚合图形构造器来编辑聚合图形
☆	把聚合图形添加到仪表板中的 “收藏夹” 小构件中
★	在仪表板“收藏夹” 小构件中的聚合图形。单击会从 “收藏夹” 中移除聚合图形。

在文档检测中介绍了通用的显示模式按钮。

在拓扑图中读取摘要

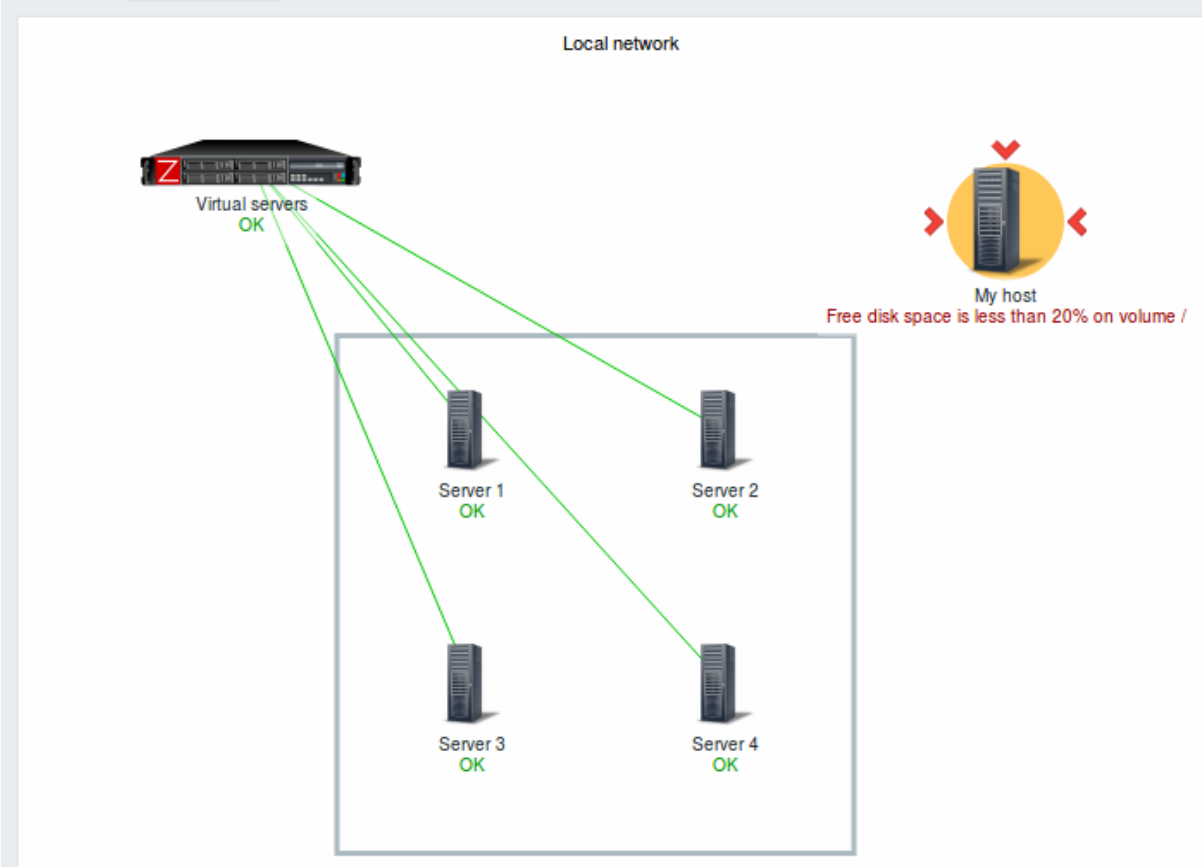
一个隐藏的可用属性 “气泡 (aria-label)”，允许使用屏幕阅读器阅读拓扑图信息。通用拓扑图描述和单个元素描述都可以用，格式如下：

- 拓扑图描述: `<Map name>, <* of * items in problem state>, <* problems in total>.`
- 描述某个元素的某个问题: `<Element type>, Status <Element status>, <Element name>, <Problem description>.`
- 描述某个元素的多个问题: `<Element type>, Status <Element status>, <Element name>, <* problems>.`
- 描述某个没有问题的元素: `<Element type>, Status <Element status>, <Element name>.`

例如，可使用这个描述：

```
'Local network, 1 of 6 elements in problem state, 1 problem in total. Host, Status problem, My host, Free
```

以下地图：:



引用网络拓扑图

可以通过 GET 的 `sysmapid` 和 `mapname` 参数来引用网络映射。例如，

```
http://zabbix/zabbix/maps.php?mapname=Local%20network
```

将打开名称是本地网络的拓扑图。

如果同时指定了 `sysmapid`（映射 ID）和 `mapname`（映射名称），则 `mapname` 具有更高的优先级。

Referencing a network map

Network maps can be referenced by both `sysmapid` and `mapname` GET parameters. For example,

`http://zabbix/zabbix/zabbix.php?action=map.view&mapname=Local%20network`

will open the map with that name (Local network).

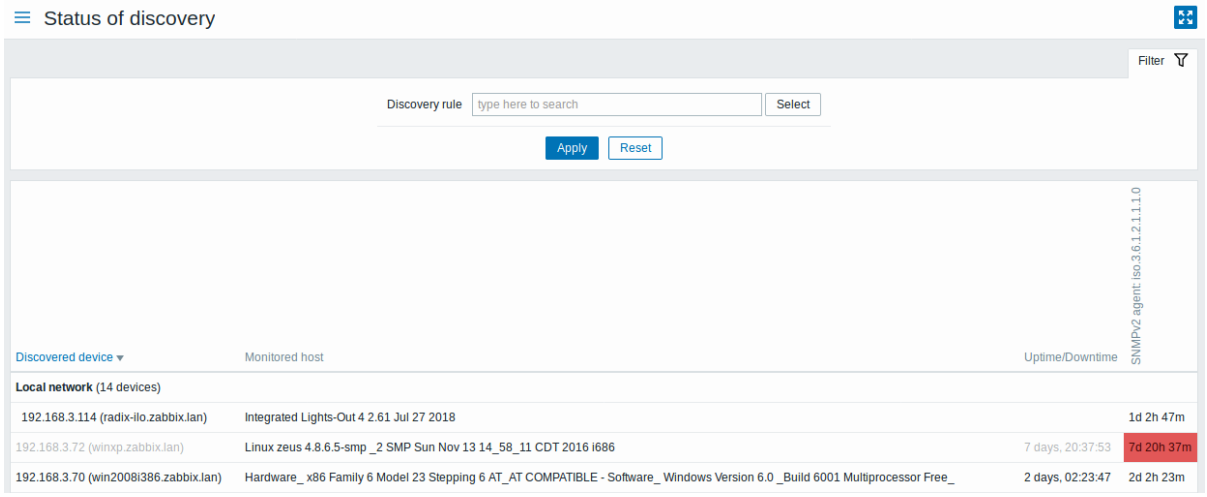
If both `sysmapid` (map ID) and `mapname` (map name) are specified, `mapname` has higher priority.

8 自动发现

概览

在 检测 → 自动发现这部分，显示网络发现的结果。发现的设备按发现规则排序。

在过滤器中未选择任何内容的话，将显示所有启用的发现规则。在过滤器中键入名称，然后选择要具体要显示的发现规则。将列出所有匹配到的已启用现规则供您选择。可以选择多个发现规则。



如果设备已被监控，主机名将列在监控的主机列中，以及在正常运行/停机时间列中显示发现设备的持续时间和发现后的丢失时间。

之后，请按照列显示每个发现的设备的各个服务的状态 (红色单元格显示已关闭的服务)。

服务正常运行时间或停机时间都包含在单元中。

<note important> 这些服务中至少有一个在设备中被发现，才会有显示其状态的列。:::

按钮

在文档检测中介绍了通用的显示模式按钮。

Using filter

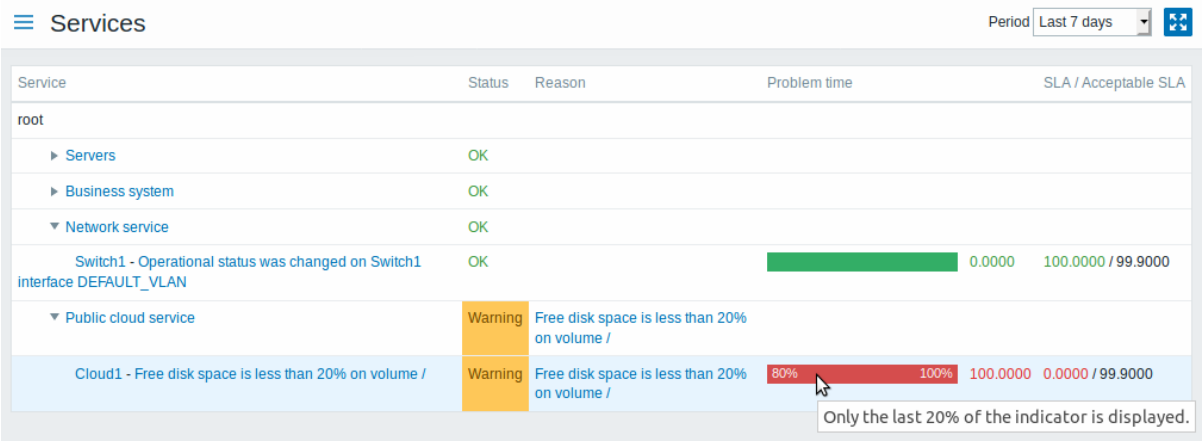
You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

With nothing selected in the filter, all enabled discovery rules are displayed. To select a specific discovery rule for display, start typing its name in the filter. All matching enabled discovery rules will be listed for selection. More than one discovery rule can be selected.

9 服务

概览

在检测中 → *IT* 服务这部分，显示 IT 基础结构或业务服务服务的状态。



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。

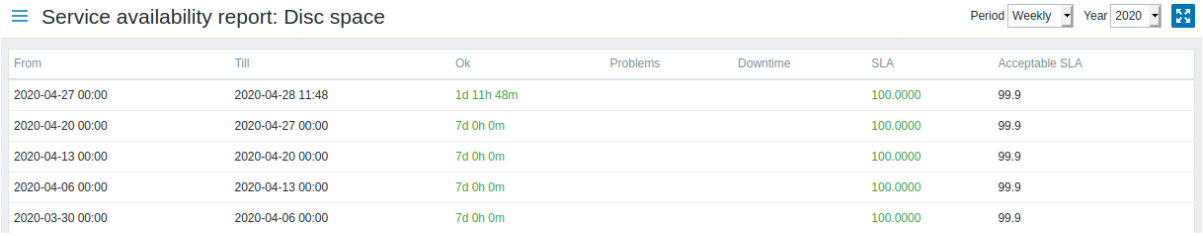
显示数据：

参数描	
服务 *(Service)* 服	名称。
状态 *(Status)* 服	状态: **OK** - 正常 **(**触发颜色和严重性**)** - 表示一个问题及其严重性问题的原因(如果有)。
理由 *(Reason)* 指	

参数	描
问题时间 *(Problem time)* 显示 S	A 柱状条。绿色/红色比例表示可用性/问题的比例。该柱状条显示 SLA 的最后 20 %(从 80 % 到 100 %)。该柱状条包含可用性数据图表的链接。

参数描	
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来访问 *IT* 服务可用性报告。



在这里，您可以长期（按日/每周/每月/每年）评估服务可用性数据。

按钮

在文档检测中介绍了通用的显示模式按钮。

2 资产记录

概览

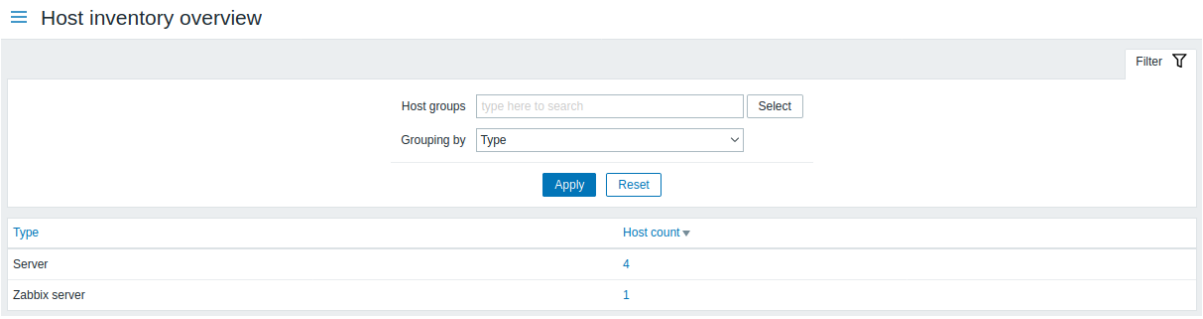
“资产记录” 菜单具有部分，根据所选参数概述主机资产数据，以及查看主机资产明细的功能。

1 概览

概览

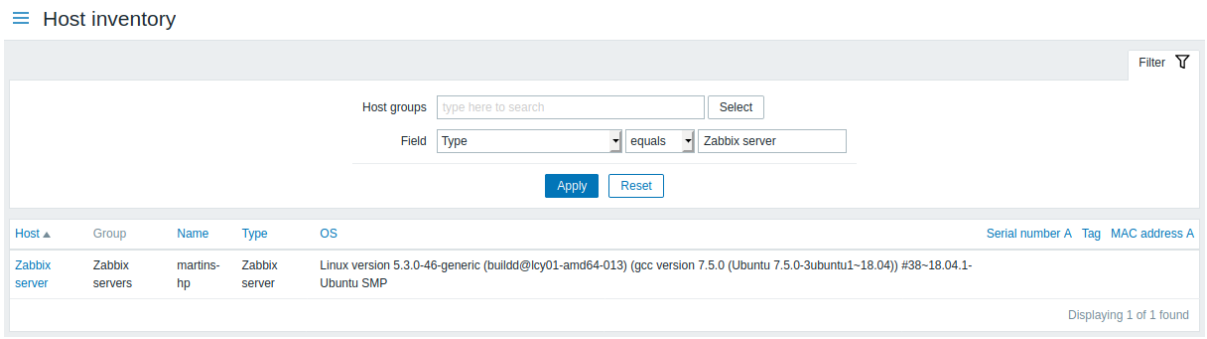
资产记录 **-** > 概览部分提供了有关主机资产数据概览的方法。

要显示的概览，请选择一个主机组（或所有组）和显示数据的资产字段。将显示与所选字段的每个条目相对应的主机数量。



概览的完整性取决于维护了多少主机资产信息。

主机计数列中的数字是链接; 它们导致这些主机在主机资产表中被过滤掉。

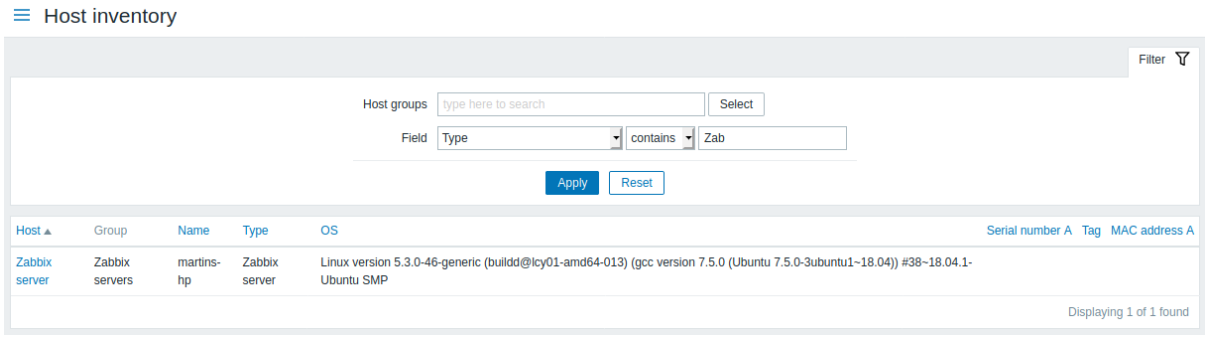


2 主机

概览

进入方法 资产记录 → 主机，资产记录信息 。

您可以按主机组和任何资产字段过滤主机，来显示您感兴趣的主机。

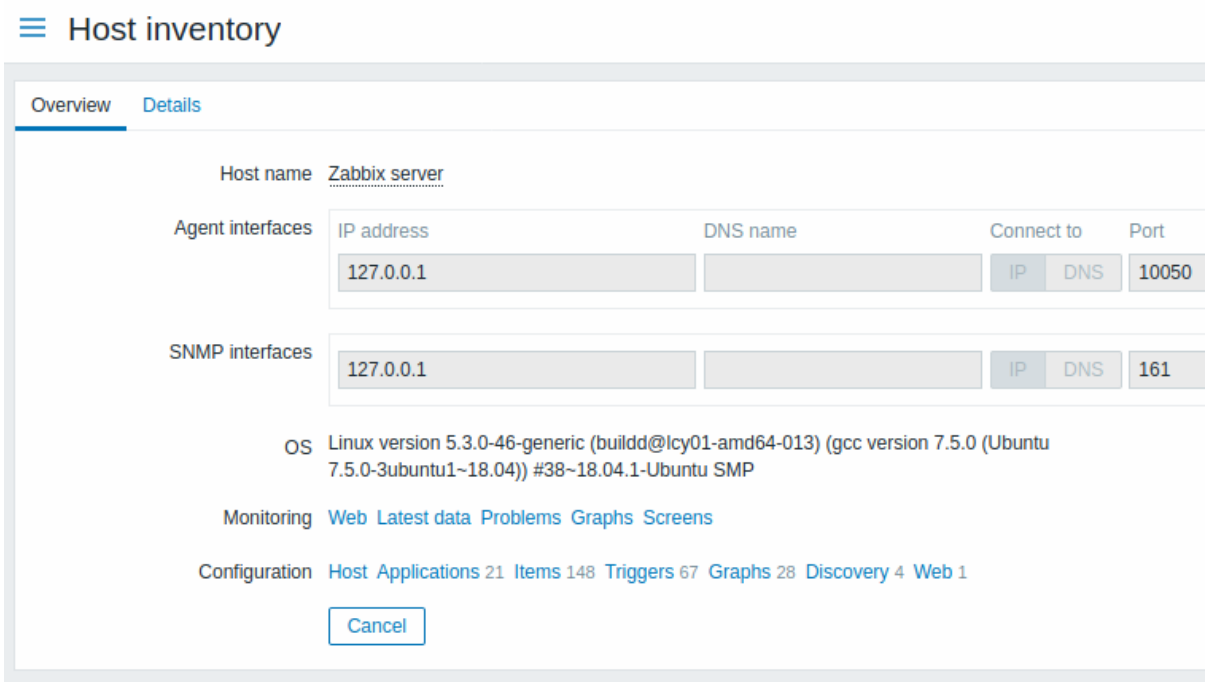


要显示所有主机清单，在组下拉列表中不选择任何主机组，清除过滤器中的比较字段，然后按 “过滤器”。

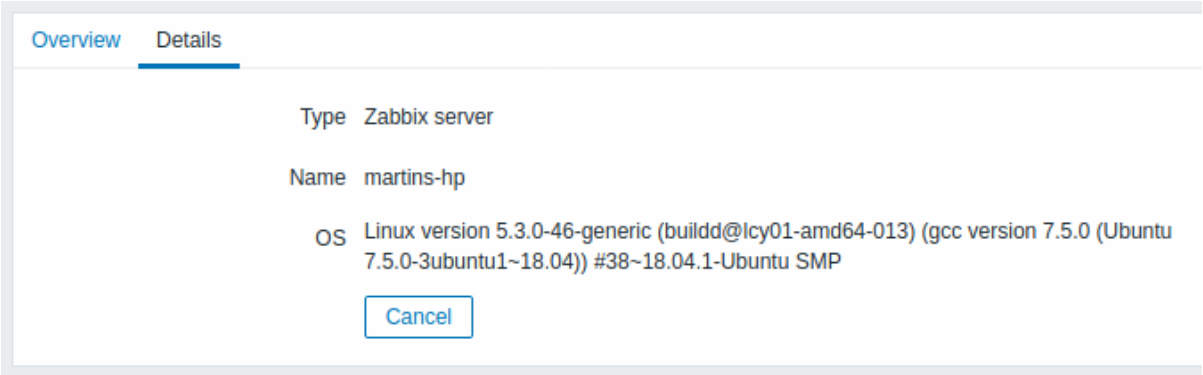
虽然表中只显示了一些关键的资产记录字段，但您也可以查看该主机的所有可用资产信息。如果想这么查看请单击列表中第一列主机名。

资产详情

在 概览选项卡中，包含有关主机的一些通用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 细节选项卡包含主机的所有可用资产记录明细：



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，则细节选项卡禁用。

3 报表

概览

“报表” 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

概览

在报表 **-** > 系统信息中，显示关键系统数据的摘要。

System information

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (enabled/disabled)	8	2 / 6
Number of templates	137	
Number of items (enabled/disabled/not supported)	394	126 / 259 / 9
Number of triggers (enabled/disabled [problem/ok])	128	55 / 73 [1 / 54]
Number of users (online)	2	1
Required server performance, new values per second	1.67	
Database history tables upgraded	No	

此报表也可以显示在仪表板小构件中。

显示数据

参数值	详细	息
Zabbix 服务器端运行中 Zabbix	务器的状态：Zabbix 服务器的位置和 **Yes** - 服务器正在运行 **No** - 服务器没有运行 提示: 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 参数大于 0）	口。

参数值	详细	息
主机数量显示配	的主机总数。受监视主机/不受监视主机的	量。
模板数量显示模	总数。	
监控项数量显示监控	总数。受监控/禁用/不受支持的	控项数量。被禁用主机上的监控项也会被统计。
触发器数量显示触发	数。启用/禁用触发器的数量	[触发器处于问题/正常状态]分配给禁用主机或依赖于禁用监控项的触发器也被统计。
用户数显示	置的用户总数。在线用户数。	

参数值	详细	息
所需的服务器性能，每秒新的值显示每秒由 Zabbix 服务	处理最新值的预计数量。所需服务器性能是一个估计值，可以作为参考。要获取准确数量的	，请使用 zabbix[wcache，values，all]内置监控项。\\计算中包含受监控主机的已启用监控项。日志监控项作为每个监控项更新间隔的一个值被统计。定期间隔值也被统计;弹性和调度间隔值不统计。在“无数据”维护期间计算不做调整。trapper 监控项也不计算在内。

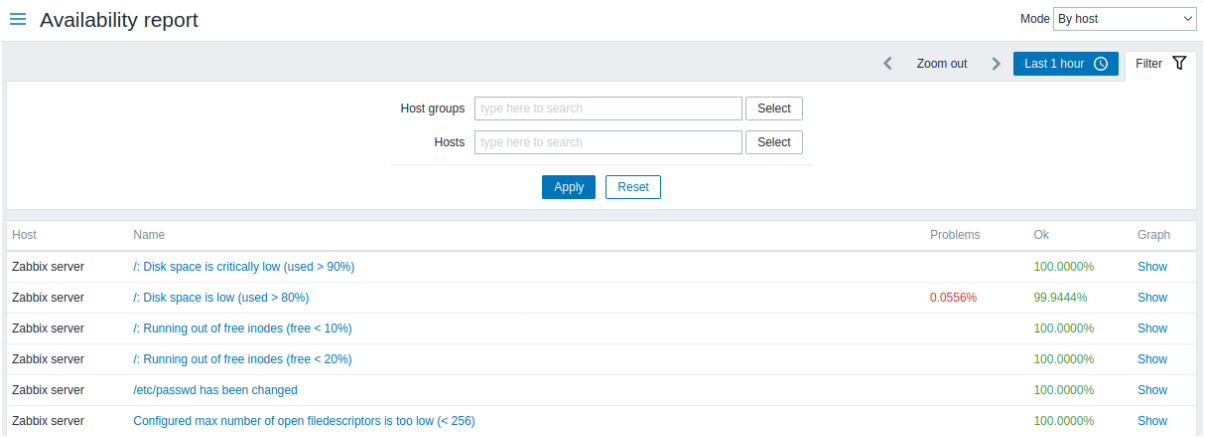
如果数据库没有使用必需的字符集或排序规则（UTF-8），系统信息还将显示一条错误消息。

2 可用性报表

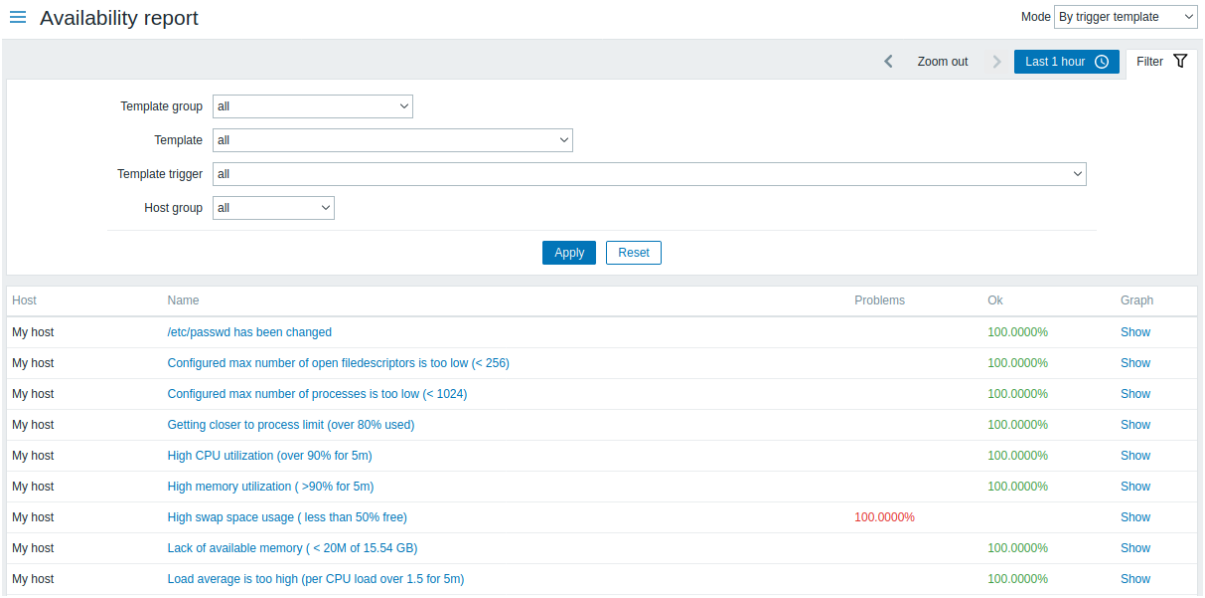
概览

在报表 **-** > 可用性报表中，您可以查看每个触发器处于问题/正常状态的时间比例。显示每种状态的时间百分比。

因此，很容易确定系统上各种元素的可用性情况。



从右上角的下拉列表中，可以选择选择模式-是按主机显示触发还是按模板触发显示。



触发器的名称是指向该触发器的最新事件的链接。

使用过滤器

过滤器可以帮助缩小显示的主机和/或触发器的数量。过滤器位于可用性报表栏下方。单击左侧的“过滤器”选项卡可以将其打开和折叠。

按触发器模板过滤

在触发器模板模式下，可以通过下面列出的一个或几个参数进行过滤结果。

参数描	
模板组从属	该组的模板中选择所有带有触发器的主机。选择至少包含一个模板的任何主机组。
模板从	选模板和所有嵌套模板中选择带有触发器的主机。仅显示从所选模板继承的触发器。不显示嵌套模板中具有的其他的触发器。
模板触发器选择具有	定触发器的主机。不会显示所选主机的其他触发器。
主机组选择	于该组的主机。

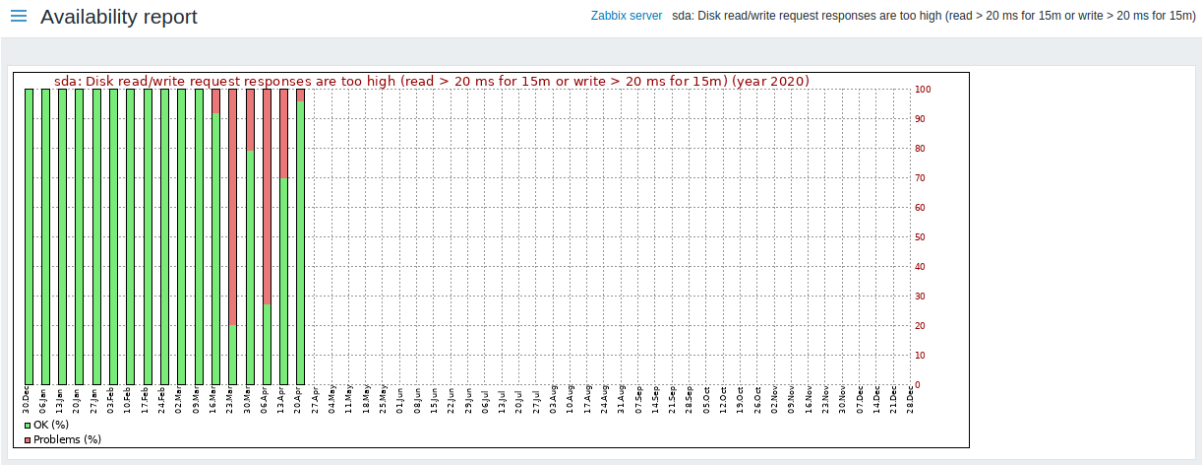
按主机过滤

在按主机模式下，可以按主机或主机组过滤结果。指定父主机组将隐式选择所有嵌套主机组。

时间周期选择器

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”列中的查看显示一个条形图，其中可用性信息以条形图显示，每个条形图表示当年过去一周的数据。



绿色部分代表 OK 时间，红色表示异常时间。

Using filter

Filter can help narrow down the number of hosts and/or triggers displayed. For better search performance, data is searched with macros unresolved.

The filter is located below the *Availability report* bar. It can be opened and collapsed by clicking on the *Filter* tab on the left.

Filtering by trigger template

In the *by trigger template* mode results can be filtered by one or several parameters listed below.

Parameter	Description
Template group	Select all hosts with triggers from templates belonging to that group. Any host group that includes at least one template can be selected.
Template	Select hosts with triggers from chosen template and all nested templates. Only triggers inherited from the selected template will be displayed. If a nested template has additional own triggers, those triggers will not be displayed.
//Template trigger //	Select hosts with chosen trigger. Other triggers of the selected hosts will not be displayed.
Host group	Select hosts belonging to the group.

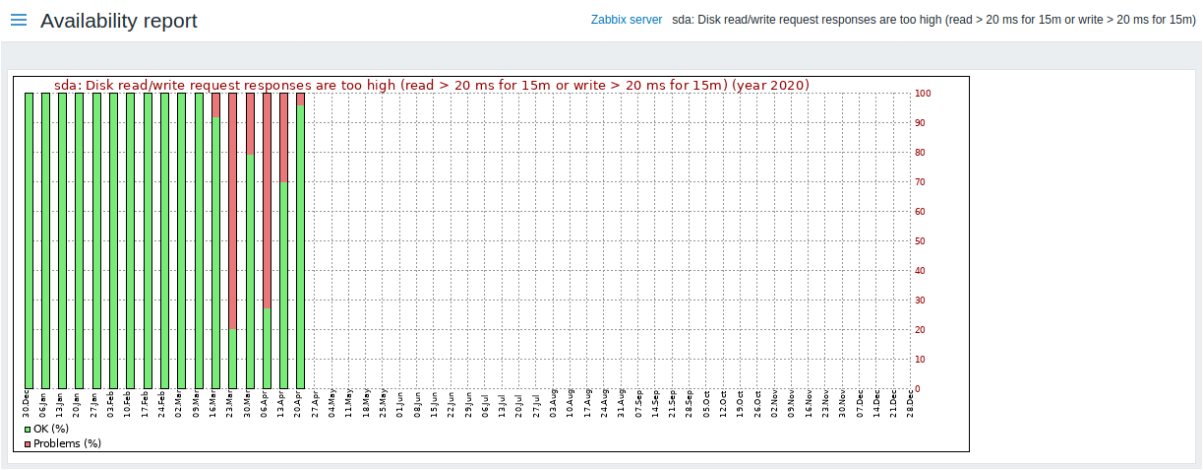
Filtering by host

In the *by host* mode results can be filtered by host or by host group. Specifying a parent host group implicitly selects all nested host groups.

Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

Clicking on *Show* in the Graph column displays a bar graph where availability information is displayed in bar format each bar representing a past week of the current year.

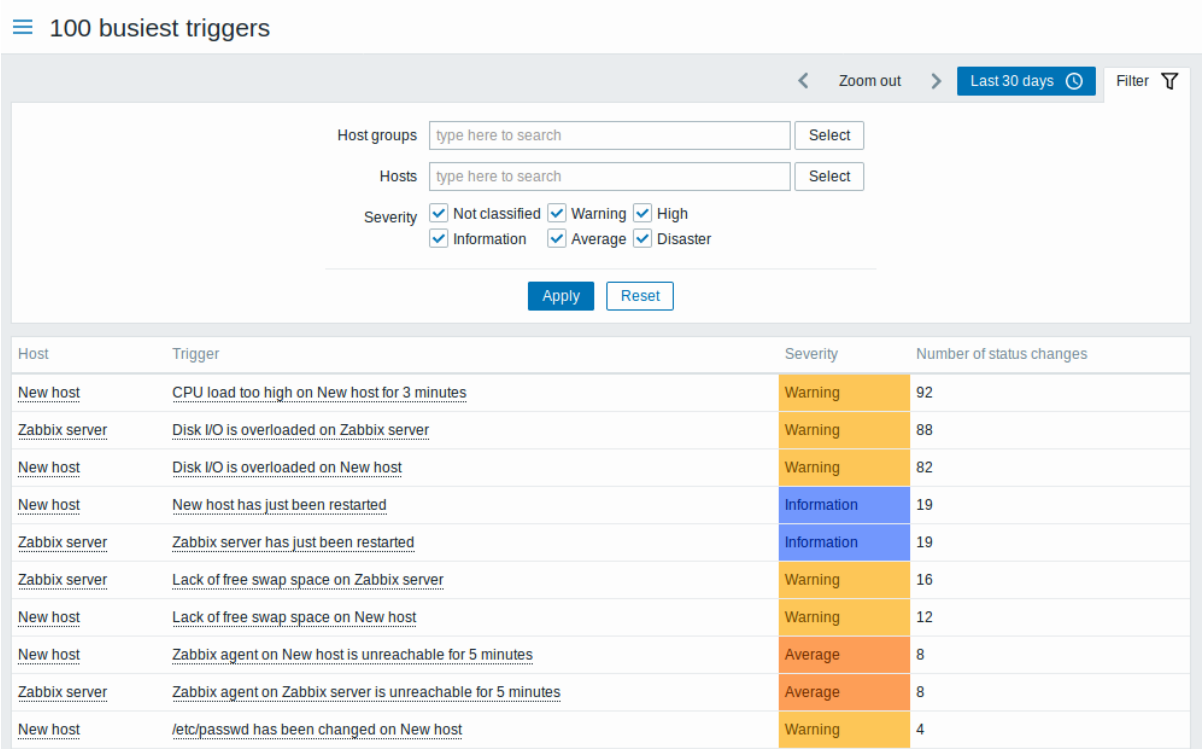


The green part of a bar stands for OK time and red for problem time.

3 触发器前 **100**

概览

在报表 → 触发器 *top100* 部分，您可以看到在评估期内最常发送状态变化的触发器，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

主机和触发器列中的实例都提供一些有用选项的链接：

- 主机 - 链接到用户定义的脚本，最新数据，资产记录，图形和聚合图形
- 触发器 - 链接到最新事件，触发器配置表单和简单图

使用过滤器

您可以使用过滤器按主机组，主机或触发器严重性显示触发器。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。

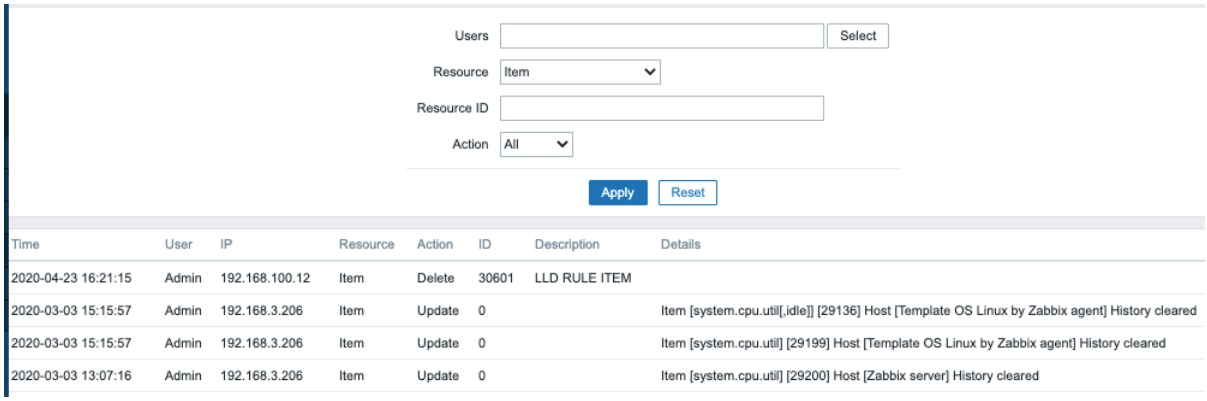
4 审计

概览

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。



显示数据：

参数功	介绍
时间戳审计	录的时间戳.
用户活	用户。
IP	在活动中使用的 IP。
资源将	示受影响的资源。
动作活	类型显示 - 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。
描述显	资源的描述。
细节显	执行活动的详细信息。

使用过滤器

您可以通过滤器按用户，动作类型和受影响的资源来缩小记录范围。

筛选器位于 “审核日志” 下方。单击左侧的 “过滤器” 选项卡可以将其打开和折叠。

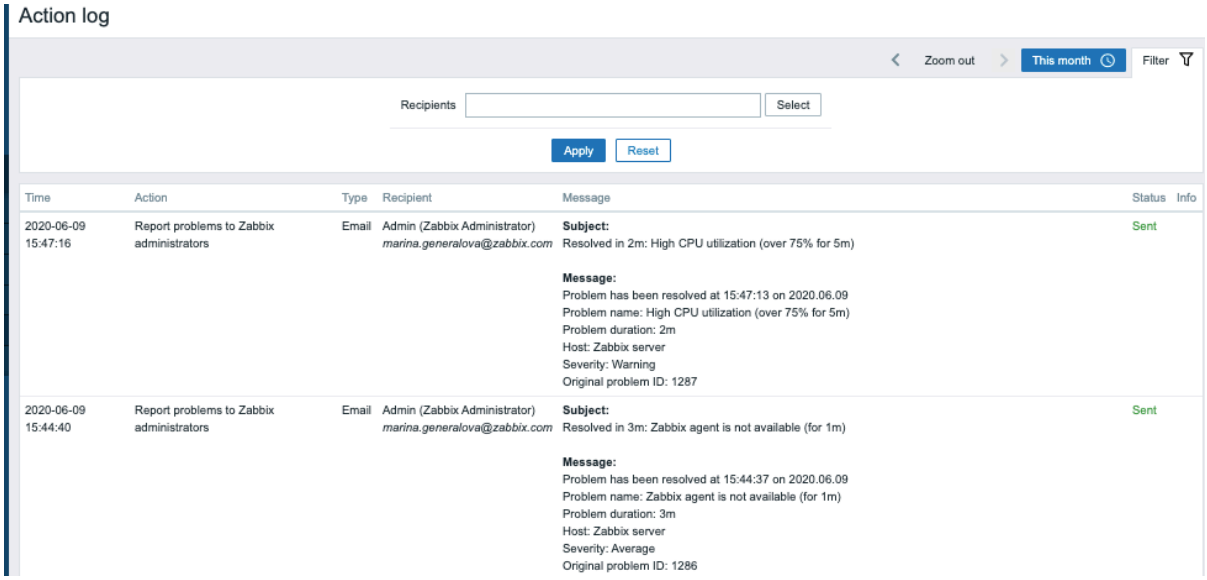
时间选择器

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

概览

在 “报告”→“操作日志” 部分，用户可以查看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
时间戳操作	间戳
动作显	引起操作的动作名称。
类型显	操作类型 - 邮件或者 命令。
接收者显示	知收件人的用户别名，名称和姓氏（用括号括起来）和电子邮件地址。
消息将	示消息/远程命令的内容。 远程命令与目标主机之间用冒号隔开:<host>:<command>。如果在 Zabbix 服务器上执行了远程命令，则该信息具有以下格式：Zabbix server:<command>
状态显	操作状态： 进行中 - 动作正在进行中 对于正在进行的动作，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 - 动作尚未完成。
信息如	有错误，则显示有关动作执行的错误信息。

使用过滤器

您可以通过过滤邮件接收者来缩小记录范围。

过滤器位于动作日志下方。单击左侧的 “过滤器” 选项卡可以将其打开和折叠。

时间选择器

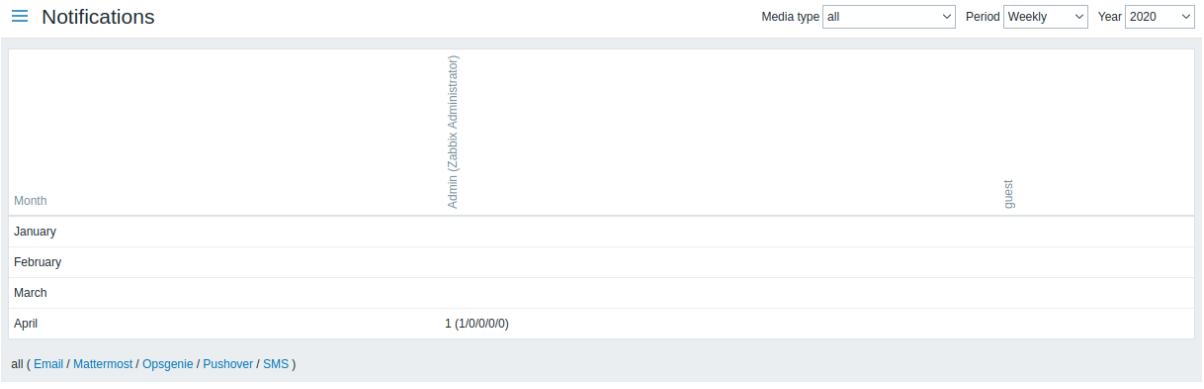
时间选择器 允许通过单击鼠标选择经常需要的时间段。单击过滤器旁边的时间段选项卡可以打开时间段选择器。

6 警报

概览

在 报表 **-** > 警报部分中，将显示发送给每个用户告警数量的报表。

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）和发送警报的年份。



每列显示发送给每个系统用户的总数。

4 配置

概述

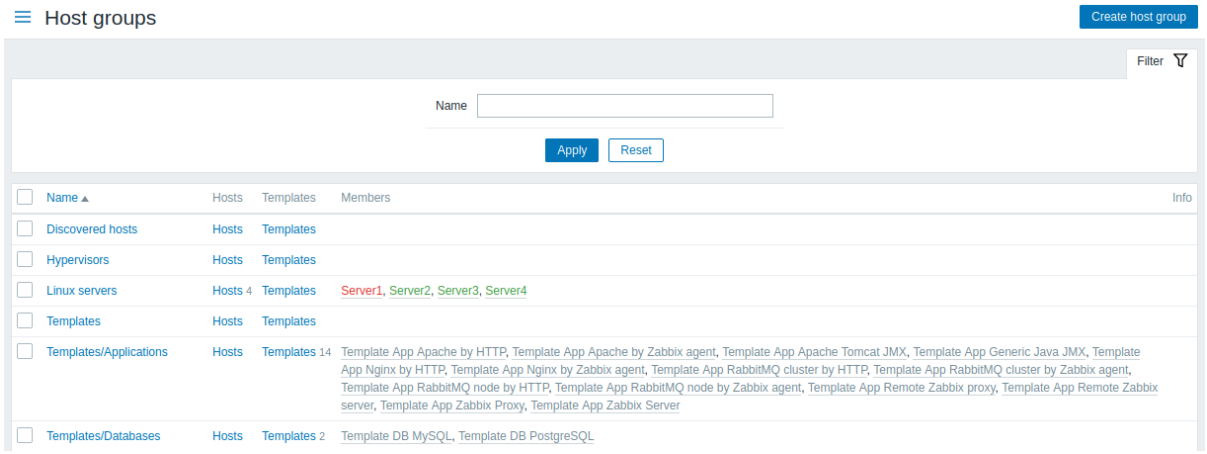
“配置” 菜单包含用于设置主要 Zabbix 功能的部分，例如主机和主机组，数据收集，数据阈值，发送问题通知，创建数据可视化等。

1 主机组

概述

用户可以在 *Configuration→Host groups* 部分中配置和维护主机组。主机组可以同时包含模板和主机。

系统显示当前存在的主机组及其详细信息。您可以通过“名称”搜索和过滤主机组。



数据展示：

列名	描
Name	主机组名称. 点击组名称将打开主机组配置表格.
Hosts	主机组中的主机个数(显示灰色)。单击“主机”将呈现属于该组的主机列表。
Templates	主机组中的模板个数(显示灰色)。单击“模板”将呈现属于该组的模板列表。

列名	描
Members	主机。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 tem-plate/host 配置信息.
Info	显示有关主机组的错误信息（如果有）。

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 required 按钮。

Using filter

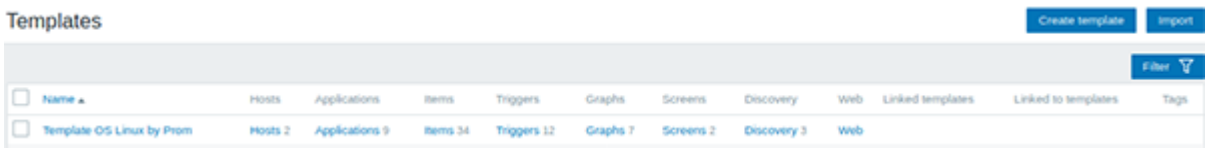
You can use the filter to display only the host groups you are interested in. For better search performance, data is searched with macros unresolved.

2 模板

概述

在 *Configuration* → *Templates(*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：

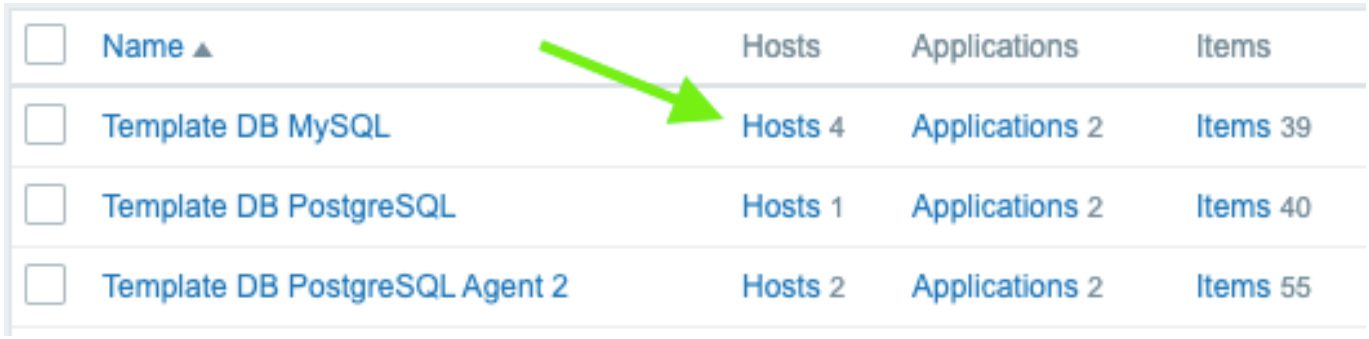
参数说	
模板（*Templates*）模板名	，单击模板名称打开模板配置表单.
主机（*Hosts*）可编辑	机的模板链接数量；不包含只读主机。自 Zabbix 5.0.3 开始，“主机”列可用。

参数说	
*Entities (Applications, Items, Triggers, Graphs, Screens, Discovery, Web) * 实体（应用集，监控项，触发器，图形，屏幕，自动发现，*Web* 监测）	模板中各个实体的数量(以灰色显示)。单击实体名称将在该实体的整个列表中过滤掉属于该模板的那些实体。

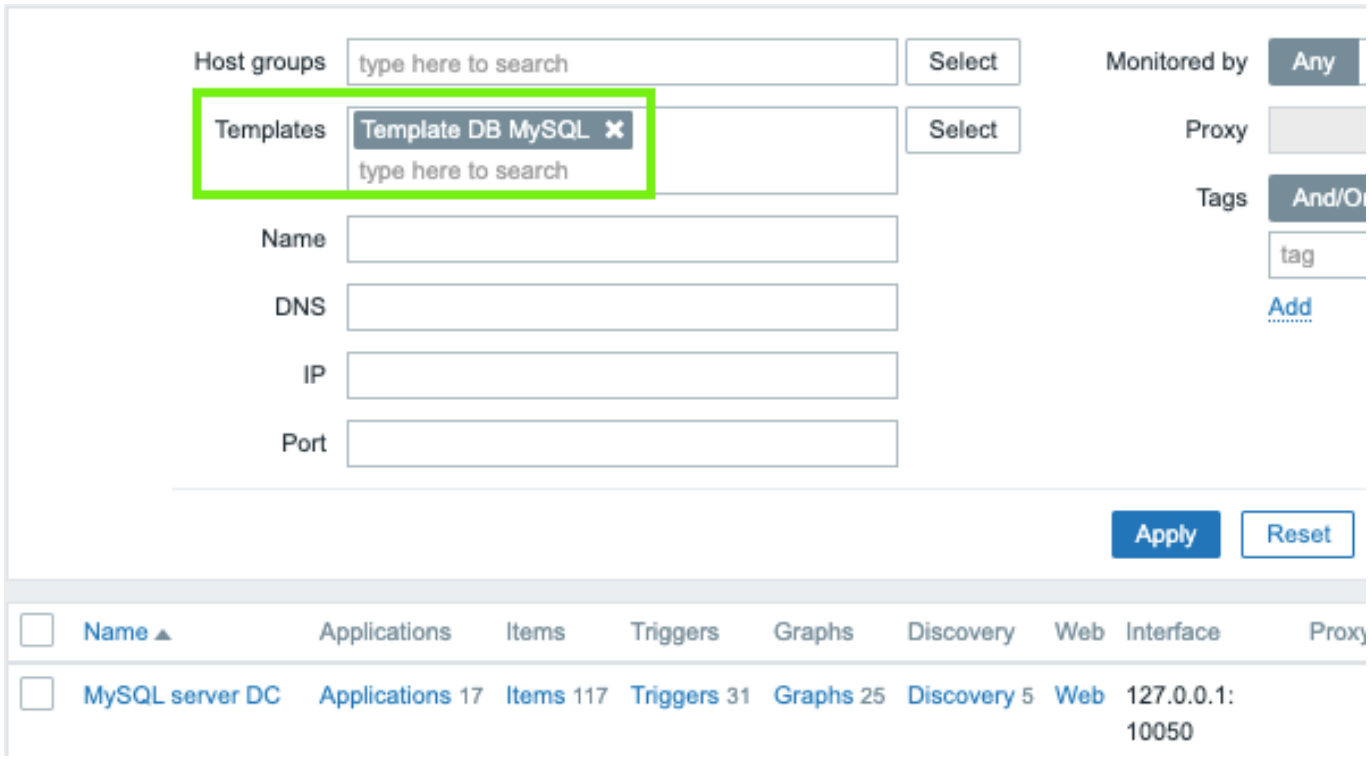
参数	说
*Linked templates(*链接模板*)*	在嵌套置中链接到模板的模板，其中模板将继承所链接模板的所有实体。
*Linked to templates(*链接到模板*)*	模板链接的模板(从该模板继承所有实体的子模板)。从 Zabbix 5.0.3 开始，此列不在包含主机.

参数说	
Tags（标签）模板的	标签](/zh/manual 带有未解析的宏

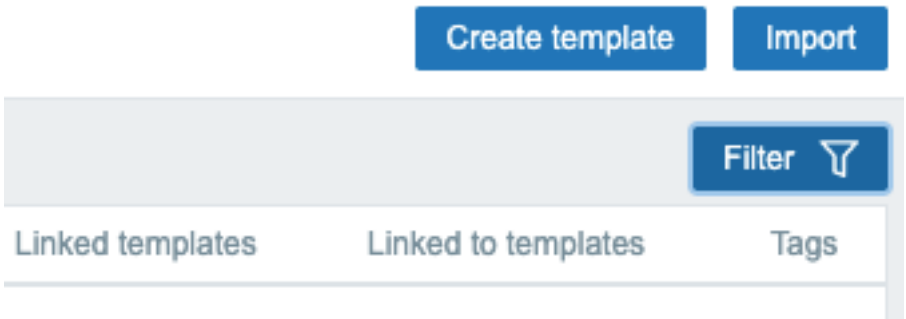
要查看连接到模板的所有主机，请单击 *"Hosts(主机)"* 超链接。



按模板名称打开过滤掉的主机配置部分（因此，将仅显示模板链接到的那些主机）：



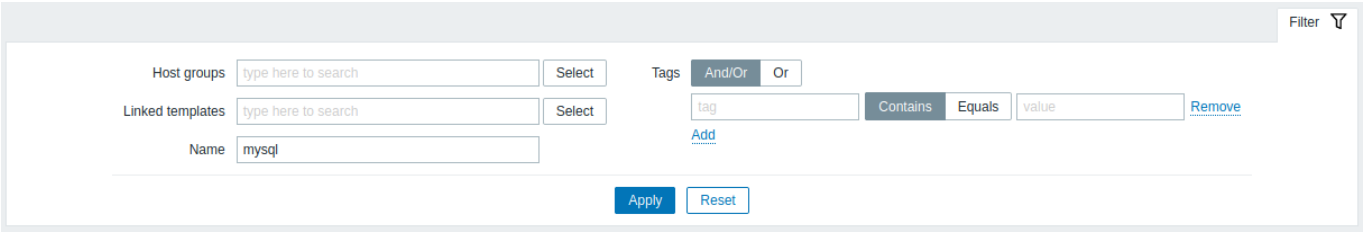
要配置新模板，请单击右上角的 *"Create template(创建模板)"* 按钮。要从 XML 文件导入模板，请单击右上角的 *"import(导入)"* 按钮。



过滤器

由于列表可能包含很多模板，因此可能需要过滤掉您真正需要的模板。

"Filter(过滤器)" 链接位于 *"Create template(创建模板)"* 和 *"import(导入)"* 按钮下方。如果单击它，将提供一个过滤器，您可以在其中按主机组，直接链接的模板和名称来过滤模板。



只能通过模板级标签（不能继承标签）进行过滤。

批量编辑选项

列表下方按钮提供了如下批量编辑选项：

- *Export(导出)* - 将模板导出到 XML 文件
- *Mass update(批量更新)* - 一次更新多个模板的多个属性
- *Delete(删除)* - 删除模板，同时将其链接的实体（包括：监控项、触发器等）保留在主机中
- *Delete and clear(删除并清除)* - 从主机删除模板及其链接的实体

要使用这些选项，请在各个模板之前选择复选框，然后单击所需按钮。

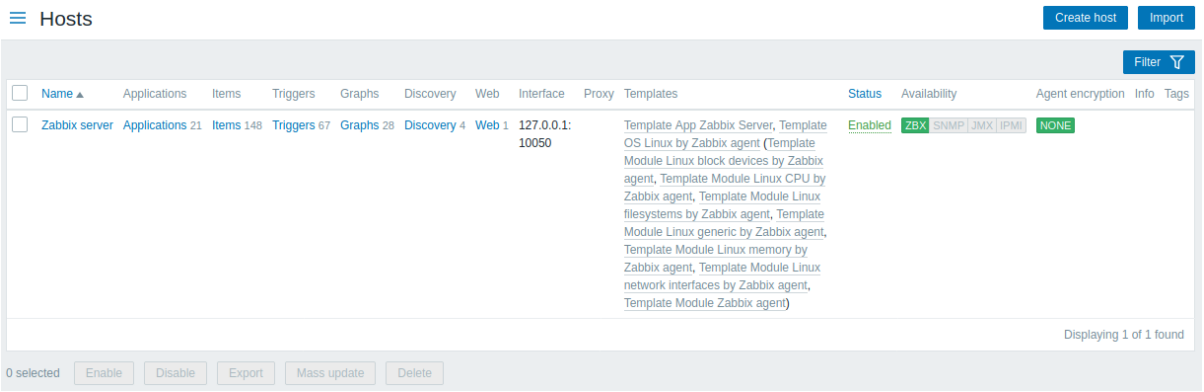
3 主机

概览

在 *Configuration → Hosts (*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 *Hosts(*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：

参数描	说明
*Name(*名称*)* 主	名称，单击主机名打开主机配置表.
*Elements (Applications, Items, Triggers, Graphs, Discovery, Web) * 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。各个元素的数量以灰色显示。

参数描	说明
Interface(界面) 显	主机的主界面。
//Proxy //	如果主机被代理监控，则显示 “代理名称”。 只有当 “monitoring by filter” 选项设置为 “Any” 或 “Proxy” 时，才会显示此列。
Templates(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。
Status(状态) 显	主机状态-启用或禁用。 通过单击状态可以更改它。 主机状态前的橙色扳手图标表示该主机正在维护中。 当鼠标指针位于图标上时，将显示维护详细信息。

参数描	说明
*Availability(*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 请注意，活动的 zabbix agent 监控项不会影响主机的可用性。
Agent encryption(agent 加密*)* 显	与主机连接的加密状态: **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）。
*Tags(*标签*)* 主	的标签，其中有未解析宏。

要配置更新主机，请单击右上角的 *"Create host(*创建主机*)"*。要从 XML 文件导入主机，请单击右上角 *import(*导入*)* 按钮.

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(*启用*)* - 将主机状态更改为 已监控

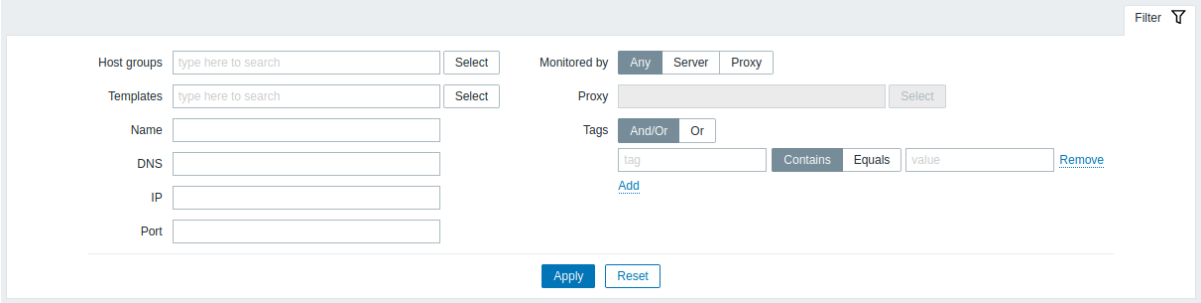
- *Disable(*禁用*)* - 将主机状态更改为 未被监控
- *Export(*导出*)* - 将主机导出为 XML 文件
- *Mass update(*批量更新*)* - 更新多个主机的多个属性。
- *Delete(*删除*)* - 删除主机

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）；
- 只有活跃的 zabbix agent 监控项；
- 该类型的接口没有轮训器（例如：StartPollers = 0）；
- 主机被禁用;
- 主机被设置为由代理监控，一个不同的代理或由服务器监控，如果它被代理监控;
- 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）。

在服务器配置缓存同步之后，将主机可用性设置为未知。在代理配置高速缓存同步之后，在受代理监视的主机上还原主机可用性（可用/不可用）。

请参阅有关主机的更多细节unreachability.

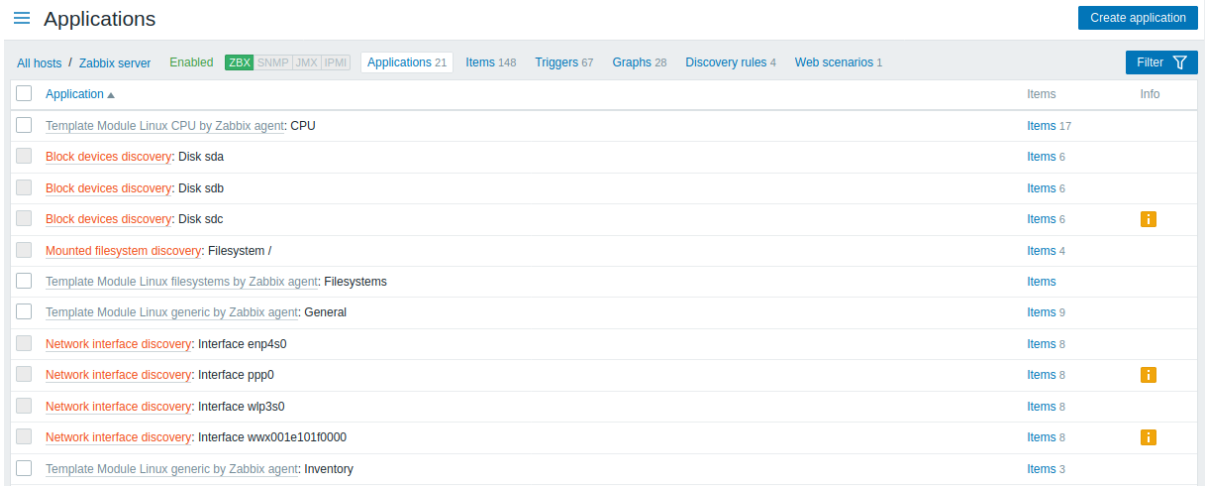
1 应用集

概览

可以从 *Configuration* → *Templates(*配置 → 模板*)* 中访问模板的应用集列表，然后单击相应模板的应用集。

可以在 *Configuration* → *Hosts(*配置 → 主机*)* 中访问主机的应用集列表，然后单击相应主机的应用集。

显示现有应用集的列表：



显示数据：

参数描	说明
*Application(*应用集*)* 应用	名称, 显示为直接创建的应用集的蓝色链接。 单击应用程序名称链接将打开该应用集配置表. 如果主机应用集属于模板，则模板名称将以灰色链接的形式显示在应用集名称之前。单击模板链接将打开模板级别的应用集列表。
*Items(*监控项*)* 点击	控项可以查看应用集中包含的监控项目。项目数量显示为灰色。
Info	显示有关应用集的错误信息（如果有）。

要配置新应用集，请单击右上角的 *"Create application(*创建应用集*)"* 按钮。

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* - 将应用集状态更改为 启用
- *Disable* - 将应用集状态更改为 禁用
- *Delete* - 删除应用集

要使用这些选项，请在各个应用集之前标记复选框，然后单击所需的按钮。

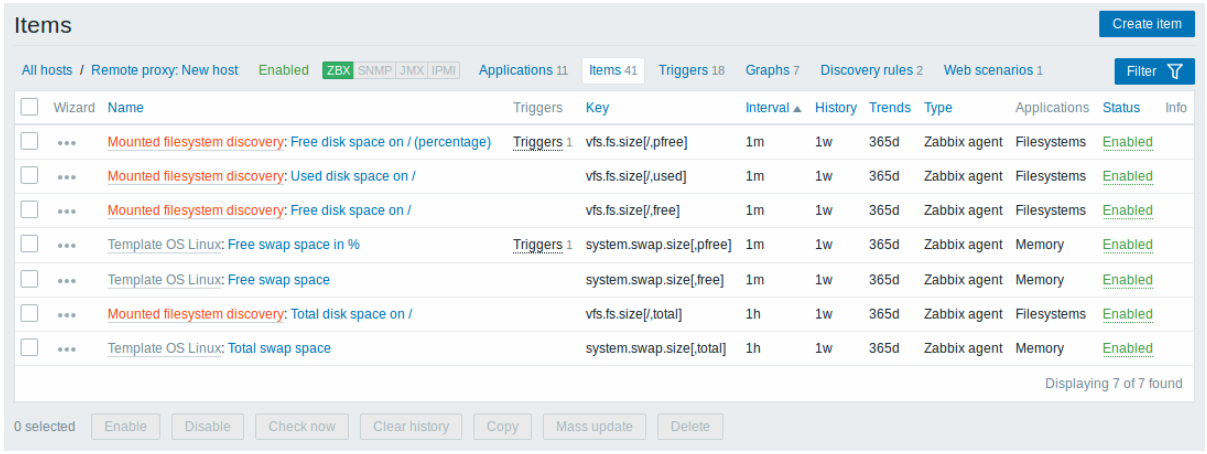
2 监控项

概览

可以从 *Configuration(*配置*)* → *Templates(*模板*)* 中访问模板的监控项列表，然后单击相应模板的监控项。

在 *Configuration(*配置*)* → *Hosts(*主机*)* 中可以访问主机的监控项列表，然后单击相应主机的监控项。

显示现有的监控项列表：



显示数据：

参数描	说明
Wizard	向导图标是指向向导的链接，用于根据监控项创建触发器。
Host(主机) 主	监控项。 当过滤选择了多个主机时，此列才会显示.
Name(名称) 监	项的名称，显示为监控项详细信息的蓝色链接。 单击监控项名称链接将打开该监控项配置表单. 如果主机监控项属于模板，模板名称将显示在监控项名称之前，作为灰色链接。 单击模板链接将打开模板级别的监控项列表。 如果项目是从监控项原型创建的，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监控项原型列表。
Triggers(触发器) 将鼠	移动到触发器上将显示一个信息框，显示与该监控项相关联的触发器。 触发器的数量以灰色显示。
Key	显示监控项的键。
Interval(间隔) 显	检查频率。 请注意，通过点击 Check now 按钮，也可以立即检查被动监控项。
History(历史记录) 将显示	控项数据记录将保留多少天。
Trends(趋势) 将	示将保留多少天的监控项趋势记录。

参数描	说明
*Type(*类型*)* 显	监控项类型（Zabbix 代理，SNMP 代理，简单检查等）。
*Applications(*应用程序*)* 显示监	项的应用集。
*Status(*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。

要配置新监控项，请单击右上角的 *"Create item*（创建项目）*"* 按钮。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

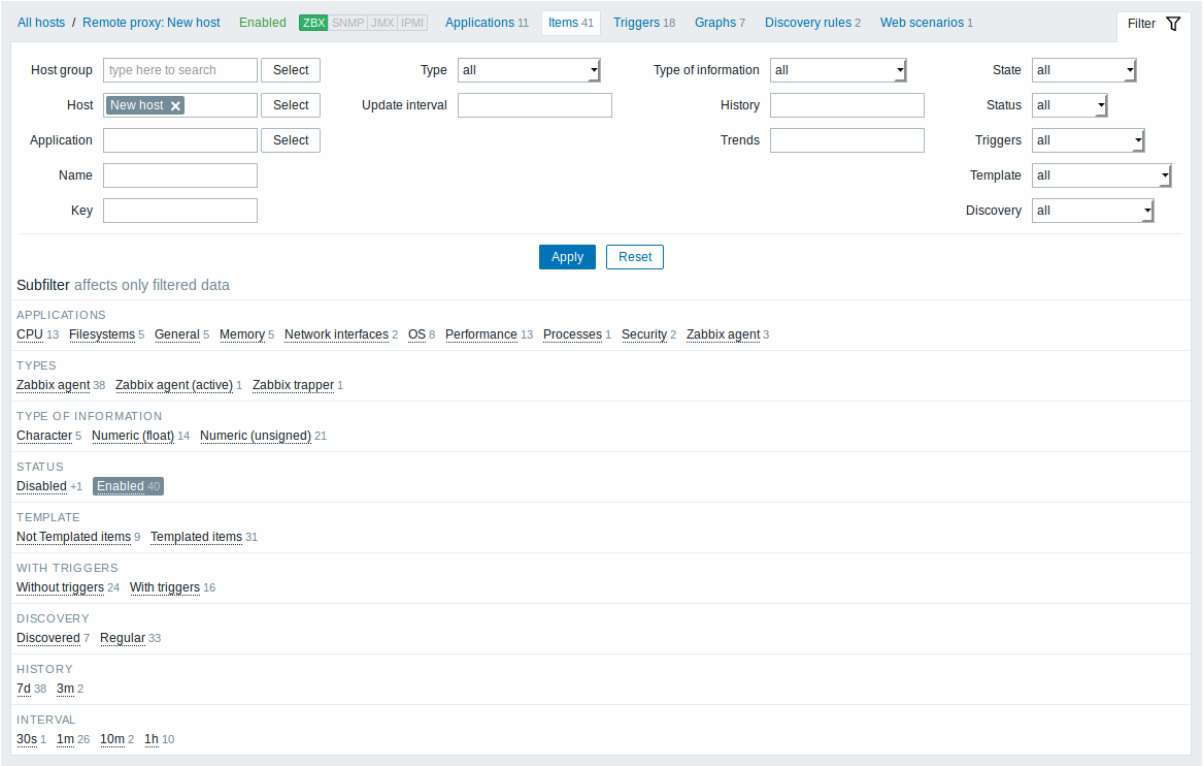
- *Enable* - 将项目状态更改为 启用
- *Disable* - 将项目状态更改为 禁用
- *Check now(*立即检查*)* - 立即执行新项目值的检查。仅支持被动检查（请参阅更多详细信息）。请注意，当立即检查值时，不会更新配置缓存，因此这些值将不会反映对项目配置的最新更改。
- *Clear history(*清除历史记录*)* -删除项目的历史记录和趋势数据
- *Copy(*复制*)* - 将项目复制到其他主机或模板
- *Mass update(*批量更新*)* - 更新多个项目的多属性
- *Delete* - 删除监控项

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。



参数描	
Host groups(主机组) 按一	或多个主机组过滤。 指定父主机组将隐式选择所有嵌套主机组。
Hosts(主机) 按	个或多个主机过滤。
Application(应用集) 按应	集过滤。
Name(名称) 按	控项名称过滤。
Key	按监控项的 key 过滤。
Type(类型) 按	控项类型（zabbix agent,SNMP agent 等）过滤
Update interval（更新间隔）按监控项更	间隔过滤。
Type of information(信息类型) 按信息	型（无符号数字、浮点数等）过滤。
History（历史记录）按监控项的	史记录保留时间进行过滤。
Trends(趋势) 按	控项的趋势保留多长时间进行过滤.
State(状态）按监	项状态进行过滤 - 正常或不支持。
Status（状态）按监控	状态过滤 - 启动或禁用
Triggers	使用（或不使用）触发器过滤监控项。
Template（模板）筛选从	板继承（或不继承）的监控项。
Discovery（自动发现）筛选由低级	现发现（或未发现）的监控项。

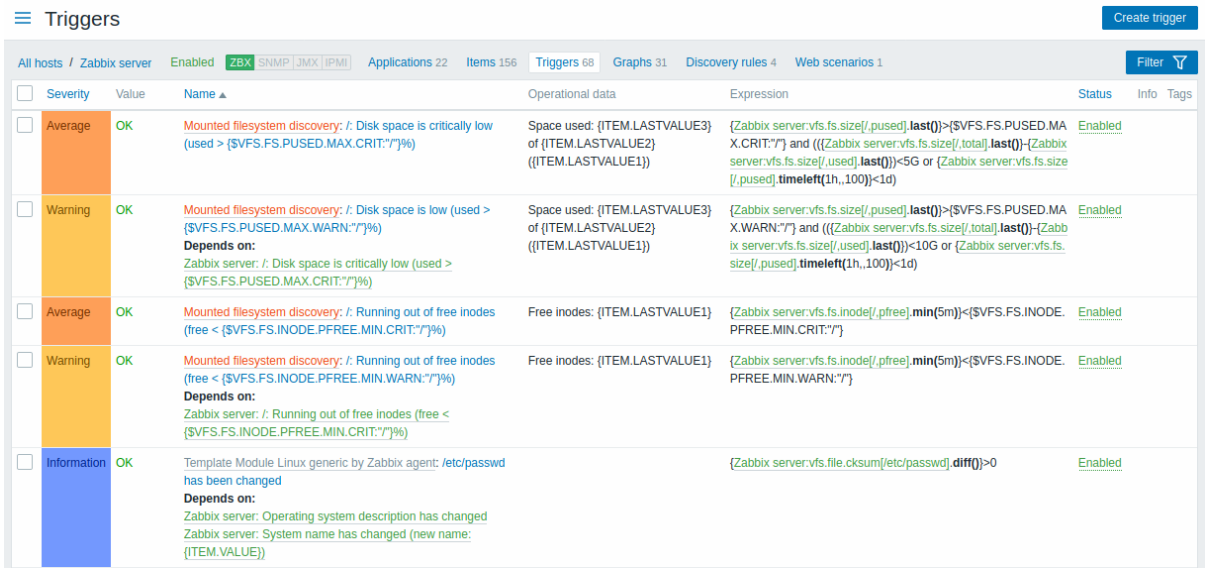
在过滤器下方的子滤波器 below the filter 提供进一步的过滤选项（已经过滤的数据）。您可以选择具有公共参数值的项目组。如果单击一个组，它将突出显示，只有具有此参数值的项目保留在列表中。

3 触发器

概述

可以从 *Configuration*（配置）→ *Templates*（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 *Configuration*（配置）→ *Hosts*（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：

参数描	说明
*Severity (*严重程度*)*	触发器的严重性由名称和单元格背景颜色显示。
Value	显示触发值： **OK**-触发器处于 OK 状态 问题-触发器处于问题状态
*Host(*主机）触发	的主机。 仅当在过滤器中选择了多个主机时，才会显示此列。
Name（名称）触发器	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。
*Operational data *（动态数据）	触发器的操作数据定义，包含任意字符串和宏，这些字符串和宏将在 *Monitoring → Problems* 中动态解析。
*Expression *（表达式）	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 *（状态）	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）。

参数描	说明
Info（信息）如果一	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。
Tags（标签）如果触	器包含标签，则标签名称和值将显示在此列中。

要配置新触发器，请单击右上角的 *"Create trigger(*创建触发器*)"* 按钮。

批量编辑选项

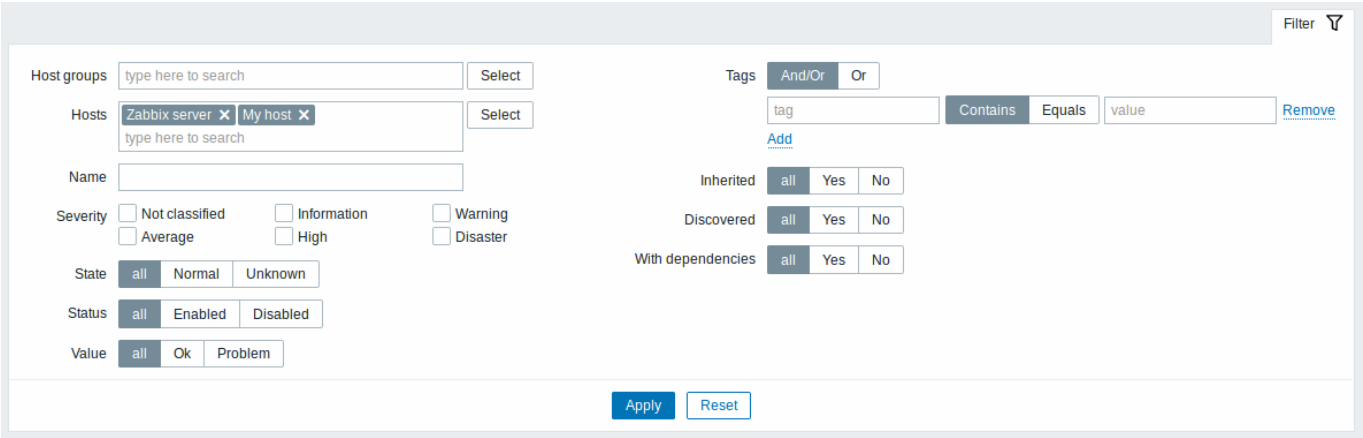
列表下方的按钮提供了一些批量编辑选项：

- *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。

使用过滤器

您可以使用过滤器仅显示您感兴趣的触发器。过滤器位于表格上方。



参数	描
*Host groups(*主机组*)*	按一个或多个主机组过滤。指定父主机组将隐式选择所有嵌套主机组。
*Hosts(*主机*)*	按一个或多个主机过滤。\\如果上面已经选择了主机组，则主机选择仅限于这些组。

参数	描
Name(名称) 按	发器名称过滤。
Severity(严重程度) 选择以	种或几种触发严重性进行过滤。
State(状态) 按	发状态过滤。
Status(状态) 按	发状态过滤。
Value	按触发值过滤。

参数	描
Tags(标签)	按发器标签名称和标签值过滤。可以设置几个条件。条件有两种计算类型： **And/Or** - 必须满足所有条件，将使用“或”条件对具有相同标签名称的条件进行分组 **or** - 如果满足

参数描	
*Inherited(*继承*)* 筛	从模板继承(或不继承)的触发器。
*Discovered(*发现）筛选	低级发现发现(或未发现)的触发器。
With dependencies（依赖项）过滤具有	或不具有)依赖项的触发器。

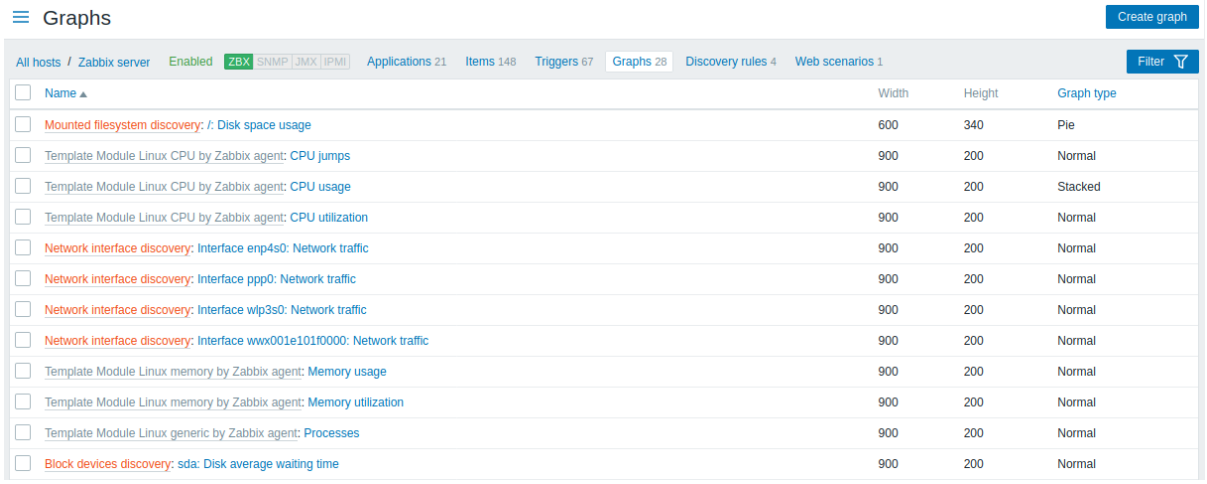
4 图形

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：

参数描	说明
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.配置表单. 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；将显示图形类型-正常，堆叠，饼图或爆炸。

配置新的图，可以点击顶部右上角的 *Create graph(*创建图形*)* 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

Using filter

You can filter graphs by host group and template/host. For better search performance, data is searched with macros unresolved.

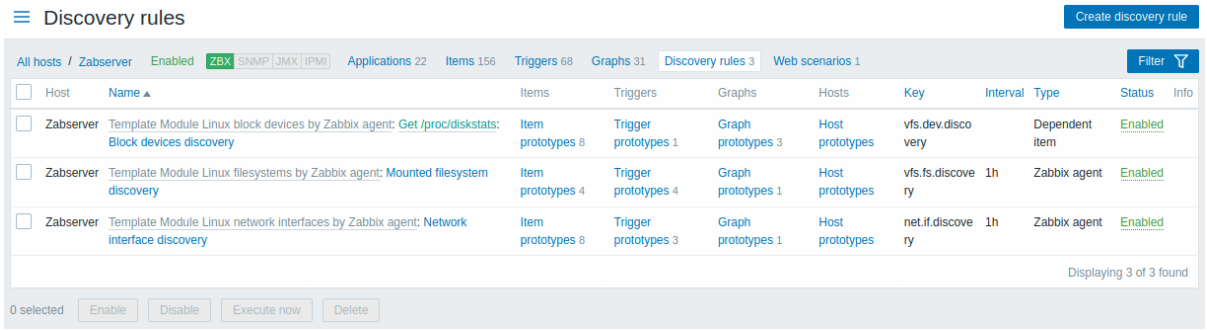
5 发现规则

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:

参数描	说明
Host（主机）显示可	的主机名。 如果没有可见的主机名，则会显示技术主机名。
Name（名称）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。配置表单. 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（监控项）显示监控	原型列表的链接。 现有监控项原型的数量用灰色来显示。
Triggers（触发器）显示触发	原型列表的链接。 现有触发器原型的数量用灰色来显示。
Graphs（图形）显示图	型列表的链接。 现有图原型的数量用灰色来显示。
Hosts（主机）显示主	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（间隔）显示执	发现的频率。 请注意，也可以通过按列表下面的立即检查按钮立即执行发现。
Type（类型）显示用	发现的监控项类型(Zabbix agent, SNMP agent, 等).
Status（状态）显示发	规则状态-启用，禁用或不支持。通过单击状态，您可以将其更改-从“启用”更改为“禁用”（然后再更改）；从“不支持”到“禁用”(并返回)。
Info（信息）如果一	正常，则此列中不会显示任何图标。如果有错误，将显示带有叉号的红色正方形图标。将鼠标移到该图标上，您将看到带有错误说明的工具提示。

配置新的低级别发现规则，可以点击顶部右上角的 *Create discovery(*创建发现规则）按钮。

批量编辑选项

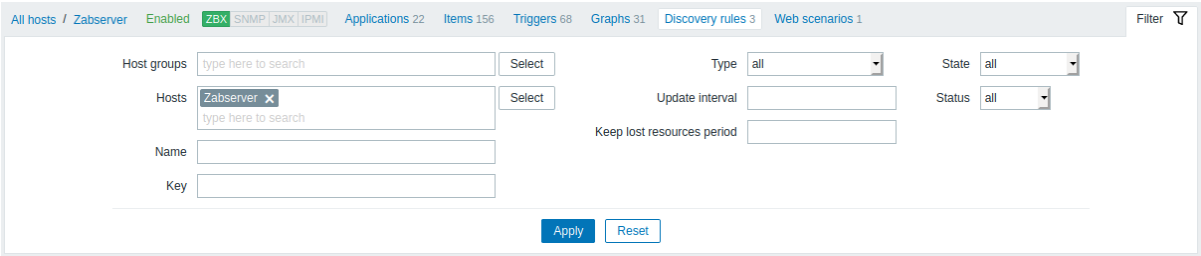
列表下面的按键会提供一些批量编辑选项：

- *Enable* - 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。
- *Check now* - 立即根据发现规则执行发现。查看更多详细信息。请注意，当立即执行发现时，配置缓存不会更新，因此结果将不会反映对发现规则配置的最新更改
- *Delete* - 删除低级别发现规则。

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

筛选

“筛选器” 链接位于发现规则列表上方。如果单击它，则将提供一个过滤器，您可以在其中按主机组，主机，名称，项目密钥，项目类型和其他参数过滤发现规则。



参数描	
*Host groups(*主机组*)* 按一	或多个主机组过滤. 指定父主机组将隐式选择所有嵌套主机组.
*Hosts(*主机）按一	或多个主机过滤。
*Name(*名称*)* 按	现规则名称过滤。
key	按发现项键过滤。
*Type(*类型*)* 按	现项目类型过滤。
*Update interval(*更新间隔*)* 按更新	隔过滤。 不适用于 Zabbix 陷印器和相关项目。
*Keep lost resources period(*保留丢失资源期限*)* 按保持丢失资源	间过滤。
SNMP OID	按 SNMP OID 过滤。 仅在选择 SNMP 代理作为类型时可用。
*State(*状态*)* 按	现规则状态（所有/正常/不支持）过滤。
*Status(*状态*)* 按	现规则状态（全部/启用/禁用）过滤。

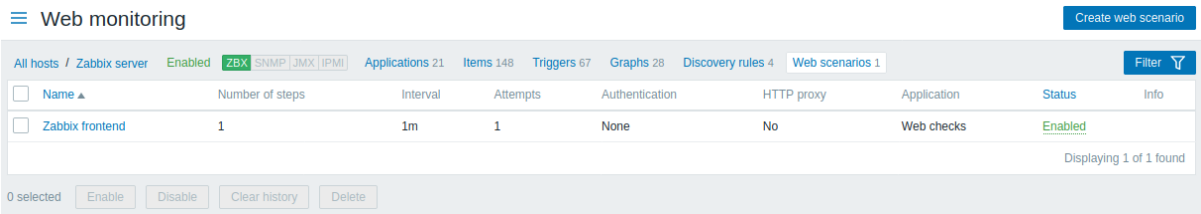
6 Web 场景

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:

参数描	说明
Name	Web 场景名称。点击 web 场景名称来打开 web 场景。配置表单.
*Number of steps(*步骤）场景	包含的步骤数。
Update interval（更新间隔）场景执行的	率。
Attempts（尝试次数）执行了多少	执行 Web 场景步骤的尝试。
Authentication（验证）显示身	验证方法-基本，NTLM 为无。
HTTP proxy（*http* 代理）显示	TTP proxy 或者在不应用的情况下选择'No'。
Application（应用集）显示 We	场景应用集
Status（状态）显示 W	b 方案状态-启用或禁用。通过单击状态可以更改它。
Info（信息）如果一	正常，则此列中不会显示任何图标。如果有错误，将显示带有叉号的红色正方形图标。将鼠标移到该图标上，您将看到带有错误说明的工具提示。

配置新的 web 场景，可以点击顶部右上角的 *Create web scenario*（创建 *Web* 方案）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

Using filter

You can use the filter to display only the scenarios you are interested in. For better search performance, data is searched with macros unresolved.

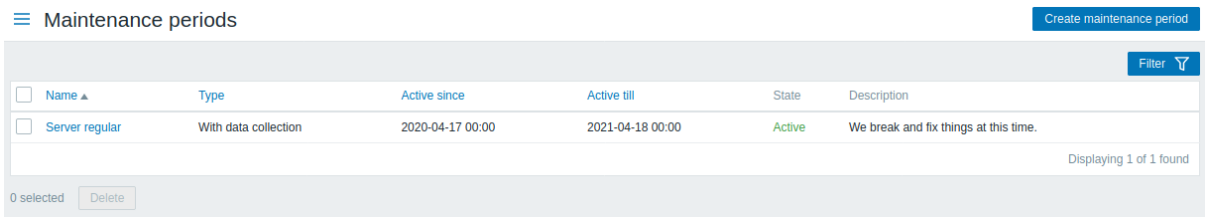
The *Filter* link is available above the list of web scenarios. If you click on it, a filter becomes available where you can filter scenarios by host group, host and status.

4 维护

概述

在 Configuration → Maintenance（配置 → 维护）里，用户可以为主机维护和配置维护时段。

显示现有维护期间及其详细信息的清单。



显示数据:

参数描	
Name	维护时段的名称。点击维护时段名称打开维护时段。配置表单.
Type	显示维护时段的类型：*With data collection* 或 *No data collection*（数据收集或无数据收集）
Active since	执行维护时段的开始时间和数据。 注意：此时间不会激活维护期；维护期限需要单独设置。
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。

要配置新的维护期限，请单击右上角的 *"Create maintenance period(*创建维护期限*)"* 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：

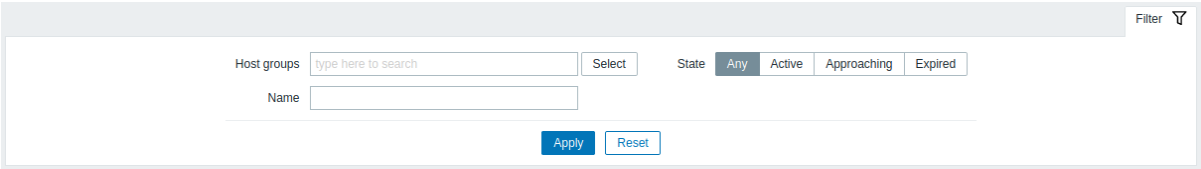
- *Delete* - 删除维护时段。

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 *Delete(*删除*)* 的按钮

过滤器

由于该列表可能包含许多维护期，因此可能需要过滤掉您真正需要的维护期。

该过滤器链接可用维护周期列表上方。如果单击它，则可以使用一个过滤器，您可以在其中按主机组，名称和状态过滤维护期限。



5 动作

概述

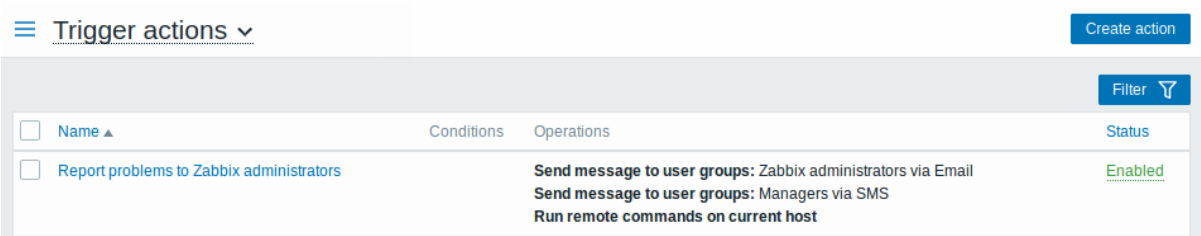
在 *Configuration* → *Actions (*配置 → 动作*)* 部分中，用户可以配置和维护动作。

显示现有动作及其详细信息的列表。显示的动作是分配给所选事件源的动作（触发，发现，自动注册动作）。

要查看分配给其他事件源的操作，请从标题下拉列表中更改源。

对于没有超级管理员权限的用户，将根据权限设置显示操作。这意味着在某些情况下，由于某些权限限制，没有超级管理员权限的用户无法查看完整的操作列表。如果满足以下条件，则会向用户显示没有超级管理员权限的操作：

- 用户在操作条件下对主机组，主机，模板和触发器具有读写访问权限
- 用户对操作，恢复和更新操作中的主机组，主机和模板具有读写权限
- 用户对操作组，恢复操作和更新操作中的用户组和用户具有读取权限



显示数据:

参数描	说明
Name（名称）动作名	。单击动作名称将打开动作配置表单.
Conditions（条件）显示动	条件。

参数描	说明
Operations（操作）显示动	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）。根据所选的操作类型，操作可以是通知，也可以是远程命令。
Status（状态）显示动	状态。- *Enabled* 或者 *Disabled*. 通过点击状态来修改它。 参见升级 获取更多细节.

要配置新动作，请单击右上角的 " *Create action(*创建动作*)*" 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：

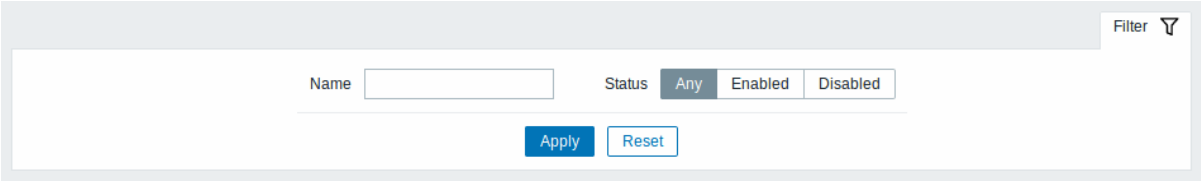
- *Enable* - 改变动作的状态至 Enabled 可用
- *Disable* - 改变动作的状态至 Disabled 不可用
- *Delete* - 删除动作

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。

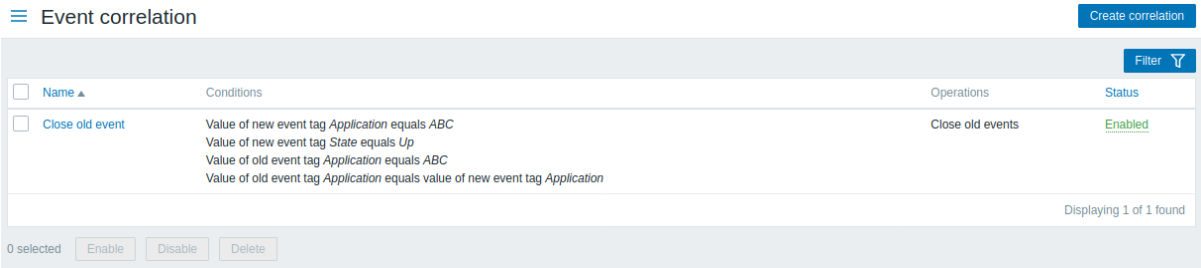
动作列表上面的 Filter （过滤器）连接是可用的。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤操作。



6 事件关联

概述

在 *Configuration* → *Event correlation*（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：

参数描	说明
Name（名称）关联规	的名称。单击关联规则名称打开规则配置表.
Conditions（条件）显示关	规则条件。
Operations（操作）显示关	规则操作。

参数描	说明
Status（状态）显示关	规则状态 - 启用或者禁用. 点击状态可以更改。

点击右上角 *Create correlation*（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：

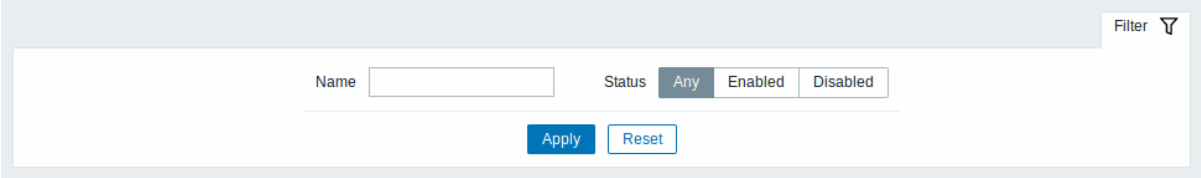
- *Enable* - 将相关关联状态更改为启用
- *Disable* -将相关关联状态更改为 禁用
- *Delete* - 删除关联规则

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。

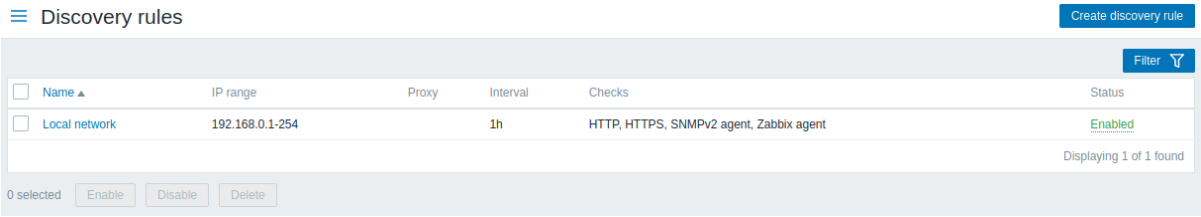


7 自动发现

概述

在 *Configuration* → *Discovery*（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：

参数描	说明
*Name(*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单.
IP range(IP 范围*)* 显	用于网络扫描的 IP 地址范围。
Proxy（代理）如果由	理执行发现，则显示代理名称。
Interval（间隔）显示执	发现的频率。
Checks（检查类型）显示用于发	的检查类型。
Status（状态）显示操	状态-启用或禁用。 通过单击状态可以更改它。

要配置新的发现规则，请单击右上角的 *"Create discovery*（创建发现规则）*"* 按钮。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

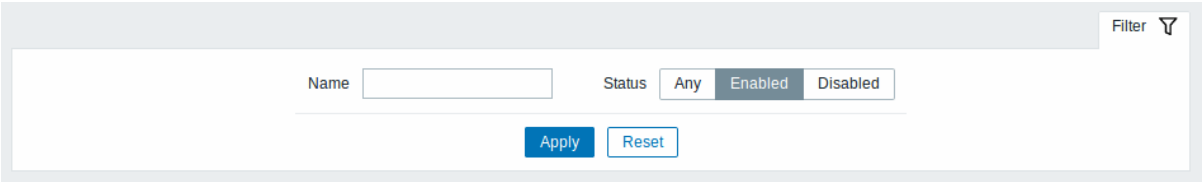
- *Enable(*启用*)* - 将发现规则状态更改为 启用
- *Disable(*禁用*)* - 将发现规则状态更改为 禁用
- *Delete(*删除*)* - 删发现规则

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。



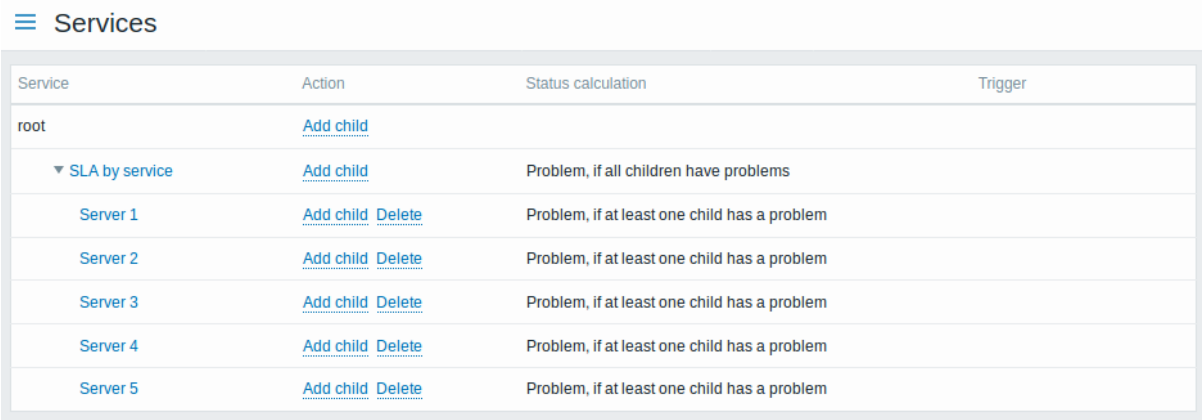
8 IT 服务

概述

在 *Configuration* → *Services*(配置 → *IT* 服务*)* 中用户可以配置和维护 IT 服务层次结构。

首次打开此部分时，它仅包含一个 *root* 条目

您可以将其用作构建受监视基础结构层次结构的起点。单击 *Add child*（添加子项）以添加服务，然后单击添加的服务下方的其他服务。



有关添加服务的详细信息，请参阅IT 服务 部分.

5 管理

概述

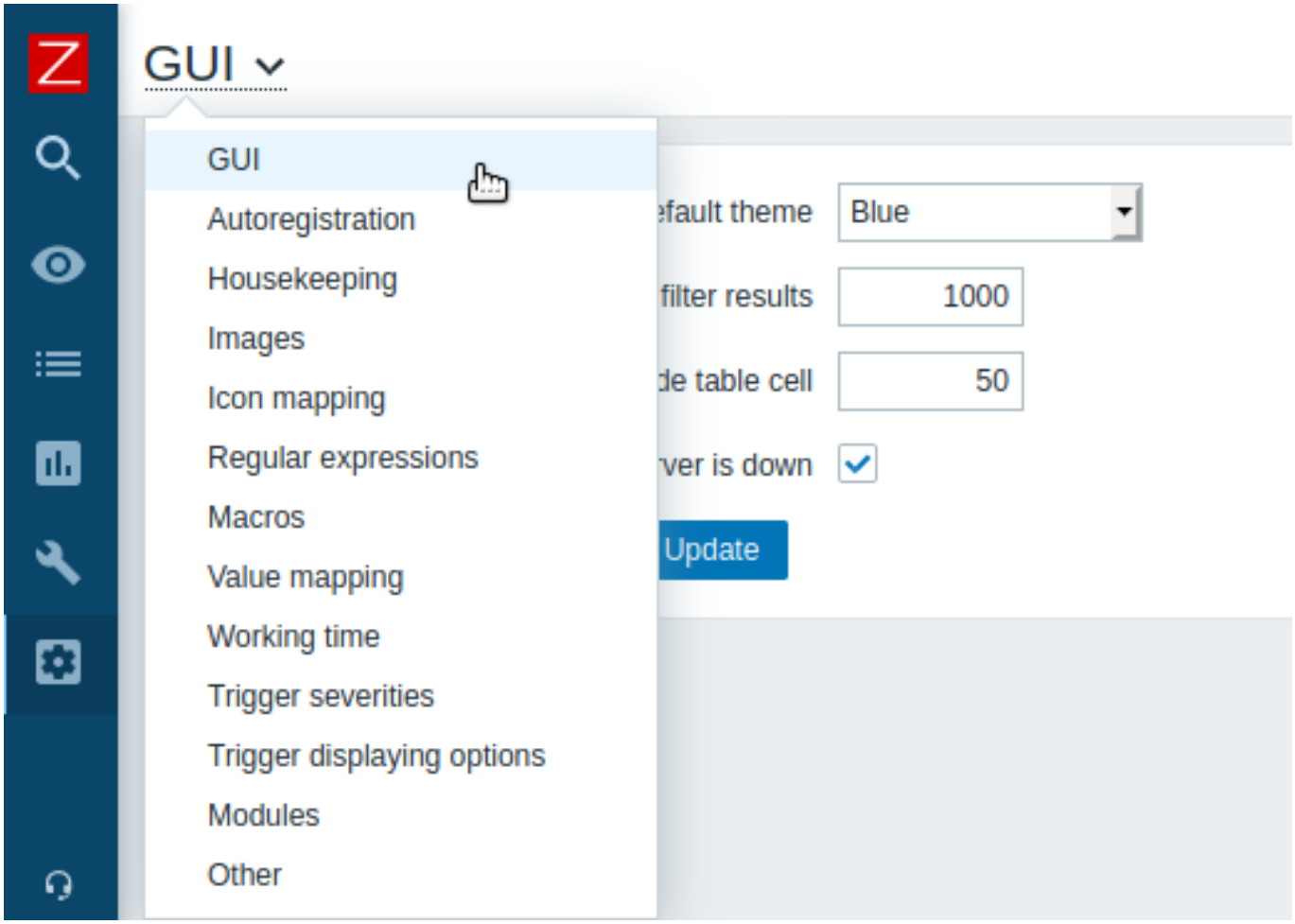
“管理” 菜单用于 Zabbix 的管理功能。该菜单仅对超级管理员 类型的用户可用。

1 常规设置

概述

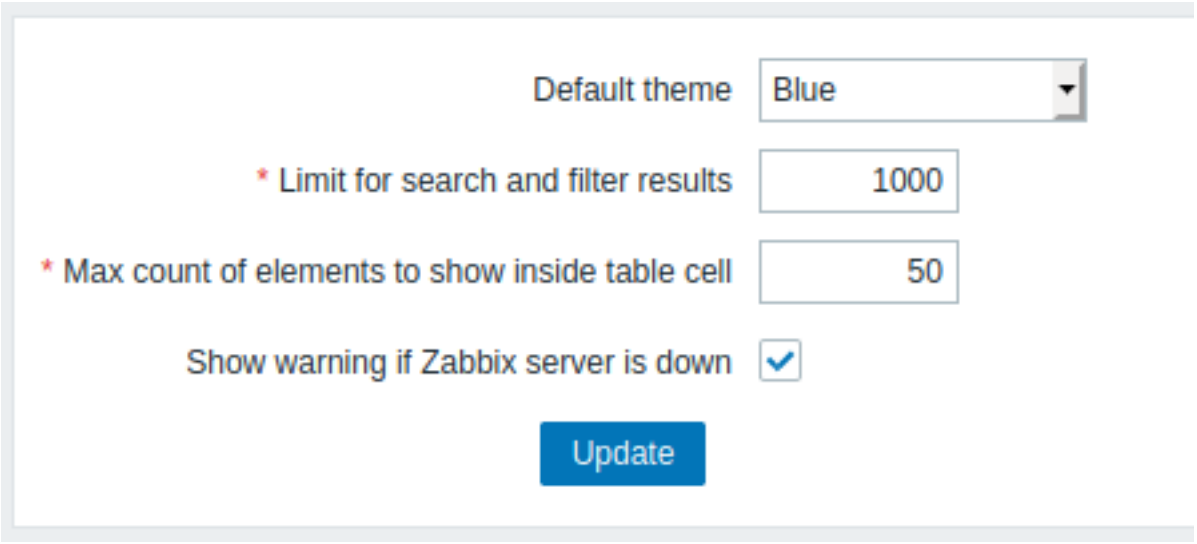
Administration → *General (“*管理*”*→*“*常规*)”* 部分包含许多屏幕，用于设置与前端相关的默认设置和自定义 Zabbix。

标题下拉菜单允许您在不同的管理屏幕之间切换。



1 GUI

此屏幕提供了一些与前端相关的默认设置的自定义。



配置参数：

参数	描
Default theme (默认主题)	未在个人资料中设置特定主题的用户的默认主题。

参数	描
*Limit for search and filter results (*搜索和过滤结果的限制*)* 注意：例如，如果设	将在Web界面列表中显示的最大元素(行)数量，例如在Configuration→Hosts中。为’50’，则在所有受影响的前端列表中仅显示前50个元素。如果某个列表包含五十多个元素，则该

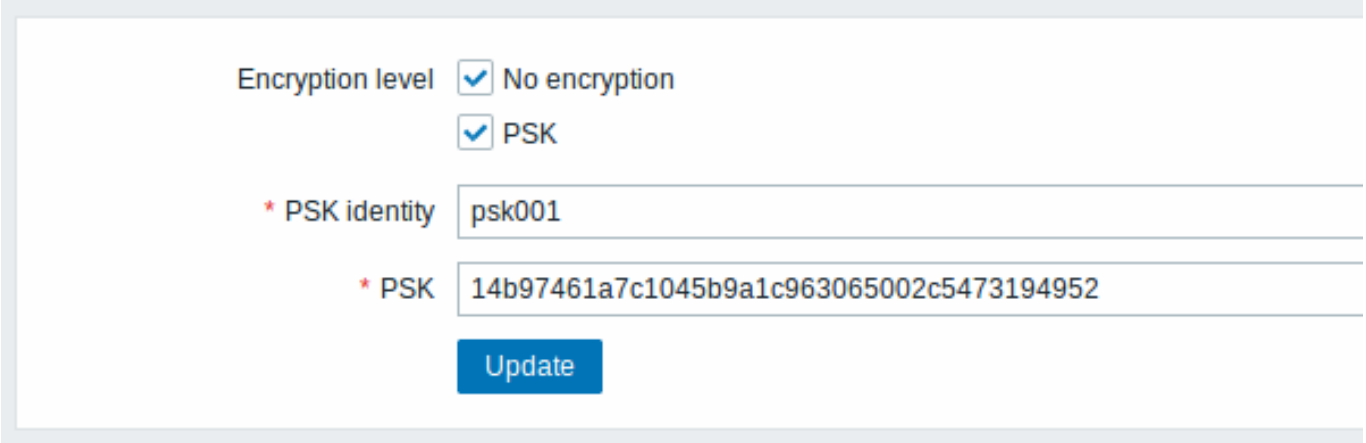
参数	描
*Max count of elements to show inside table cell (*在表格单元格中显示的最大元素数*)*	对于显示在单个表单元格中的条目，将不会显示超出此处配置的条目。

参数	描
*Show warning if Zabbix server is down (*如果 *Zabbix* 服务器已关闭，则显示警告*)* 从 Zabbix 2.0.1	如果无法访问 Zabbix 服务器(可能已关闭)，则此参数使警告消息显示在浏览器窗口中。即使用户向下滚动页面，该消息仍然可见。如果将鼠标移到其上方，该消息将

参数描

2 自动注册

在此屏幕中，您可以配置活动代理自动注册的加密级别。



标有星号的参数是强制性的。

配置参数：

范围描	
*Encryption level (*加密等级*)* **N	选择一个或两个选项进行加密级别： encryption(不加密)**-**允许未加密的连接 ** **PSK**-允许使用具有预共享密钥的TLS加密的连接

范围描	
PSK identity (PSK 身份) 如	输入预共享密钥标识字符串。“PSK”选作此字段仅适用加密级别。不要将敏感信息放在PSK身份中，它会通过网络未经加密地传输，以通知接收者要使用哪个PSK。

范围描	
PSK	输入预共享密钥(偶数个十六进制字符)。 最大长度:如果 Zabbix 使用 GnuTLS 或 OpenSSL 库,则为 512 个十六进制数字(256 字节 PSK),如果 Zabbix 使用 mbed TLS (PolarSSL) 库,则为 64 个十六进制数字

范围描

另请参阅：安全自动注册

3 管家

管家是一个定期过程，由 Zabbix 服务器执行。该过程将删除过时的信息和用户删除的信息。

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，可以针对每个任务分别启用或禁用整理任务，这些事件包括：事件和警报/ IT 服务/审计/用户会话/历史记录/趋势。如果启用了管家服务，则可以设置将数据记录保留多少天，然后再由管家删除。

删除 项目/触发器 *(item/trigge)* 也将删除该项目/触发器所产生的问题。

而且，只有事件与事件无关，事件才会被管家删除。这意味着，如果事件是问题事件或恢复事件，则在删除相关问题记录之前，不会将其删除。管家将首先删除问题，然后再删除事件，以避免陈旧事件或问题记录的潜在问题。

对于历史和趋势，还有其他选择：*Override item history period(*覆盖项目历史记录周期*)* 以及 *Override item trend period(*覆盖项目趋势期*)*. 此选项允许全局设置项目历史记录/趋势将保留多少天（1 小时至 25 年；或 “0”），在这种情况下，将覆盖 *History storage period/Trend storage period (*历史记录存储期/趋势存储周期*)* 中为单个项目设置的值项目配置中的字段。请注意，对于具有配置选项 “不保留历史记录” 和/或 “不保留趋势” 的项目，存储期限不会被覆盖。

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。

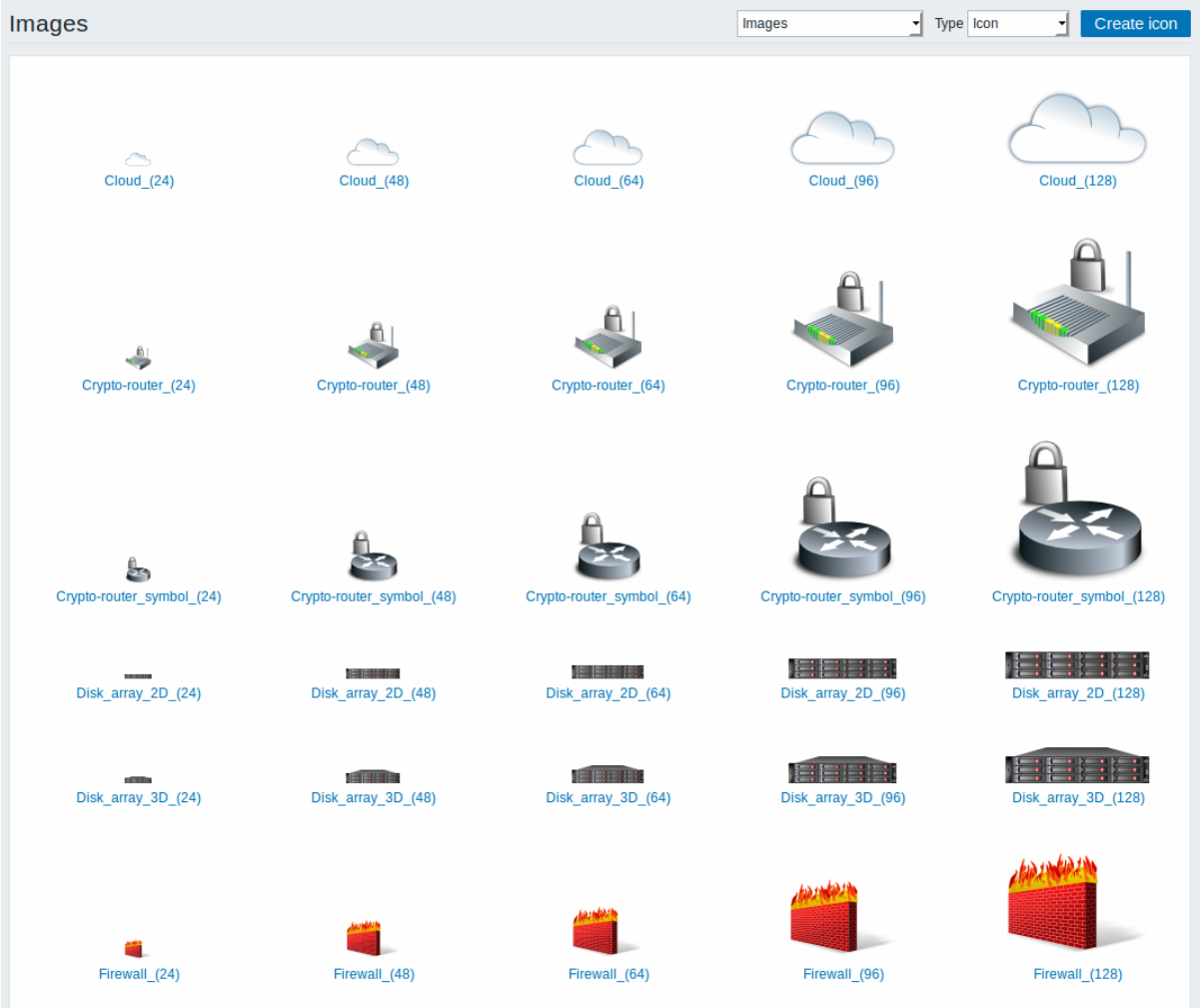
<note important> 如果使用 TimescaleDB，为了充分利用 TimescaleDB 对历史记录和趋势表的自动分区，必须启用 “覆盖项目历史记录周期” 和 “覆盖项目趋势周期” 选项。否则，保留在这些表中的数据仍将存储在分区中，但是，管家将通过删除单个记录而不是通过删除过时的分区来清理历史记录和趋势。:::

时间段字段中支持时间后缀，例如 1d（一天），1w（一周）。最少为 1 天（历史记录为 1 小时），最长为 25 年。

*Reset defaults(*重置默认值*)* 按钮可以还原所做的任何更改。

4 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。

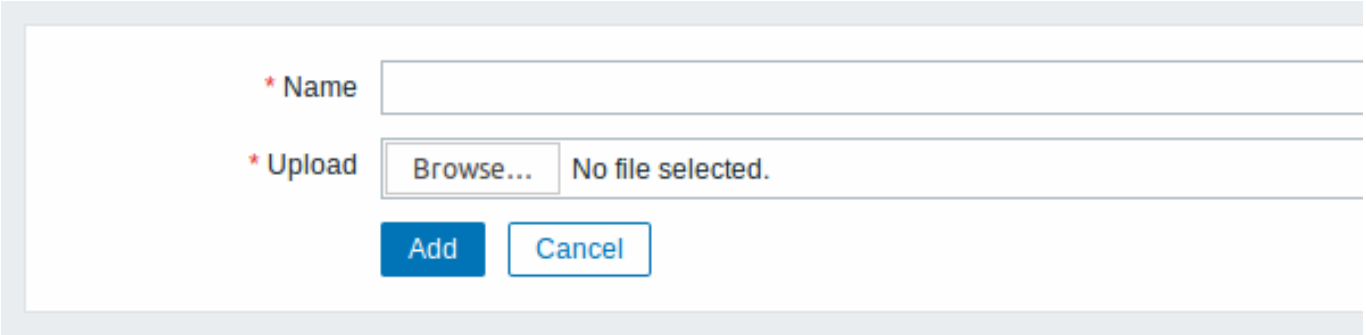


类型下拉菜单允许您在图标和背景图像之间切换：

- 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 *Create icon(*创建图标*)* 或者 *Create background(*创建背景*)* 按钮添加自己的图像。



图像属性：

参数描	n
Name(名称) 图	的唯一名称。
Upload(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。

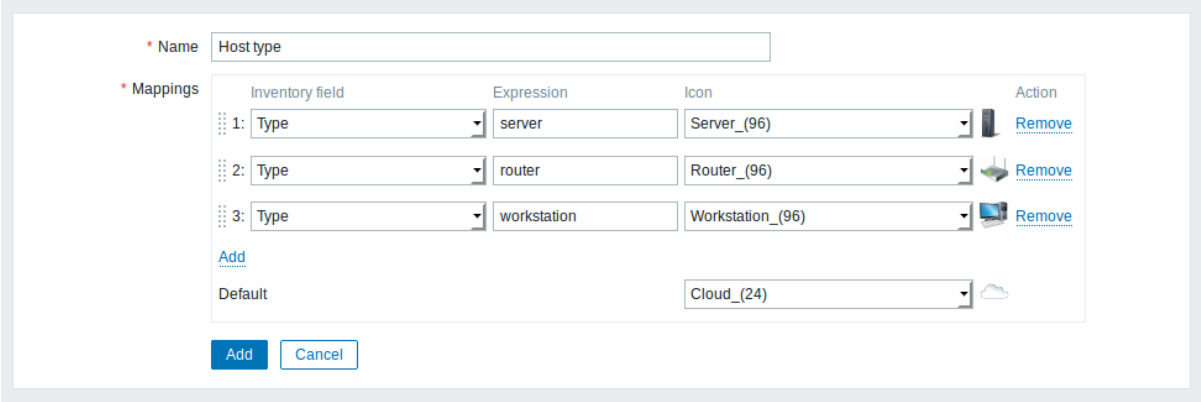
如果图像大小接近 1 MB，max_allowed_packet 的 MySQL 配置参数的默认值为 1MB，则图像的上传可能会失败。在这种情况下，增加 max_allowed_packet 参数。

5 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 *Create icon map(创建图标地图)* 。



配置参数：

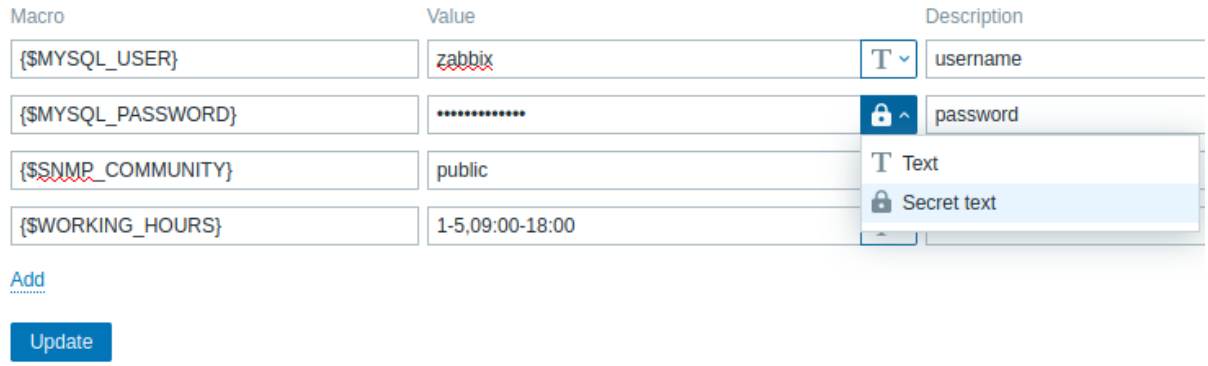
参数描	
Name(名称) 图	地图的唯一名称。
Mappings(映射) 映	列表。映射顺序决定哪一个优先级。您可以使用拖放方式在列表上下移动映射。
Inventory field(库存字段) 将要查	一个匹配的主机库存字段。
Expression(表达式) 描述	配的正则表达式。
Icon(图标) 如	找到表达式的匹配，则使用图标。
Default(默认) 要	用的默认图标。

6 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 部分。

7 宏

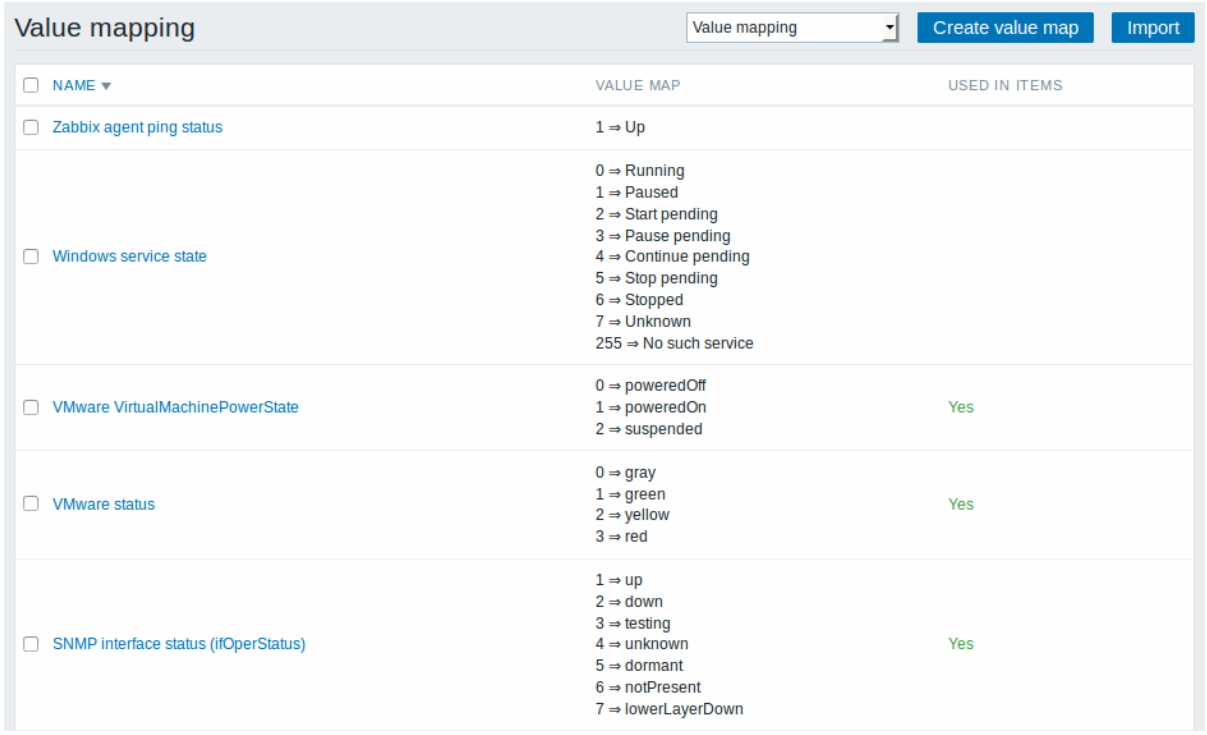
本节允许将系统范围的宏定义为宏值对。还支持添加描述。



更多细节，参见用户宏。

8 值的映射

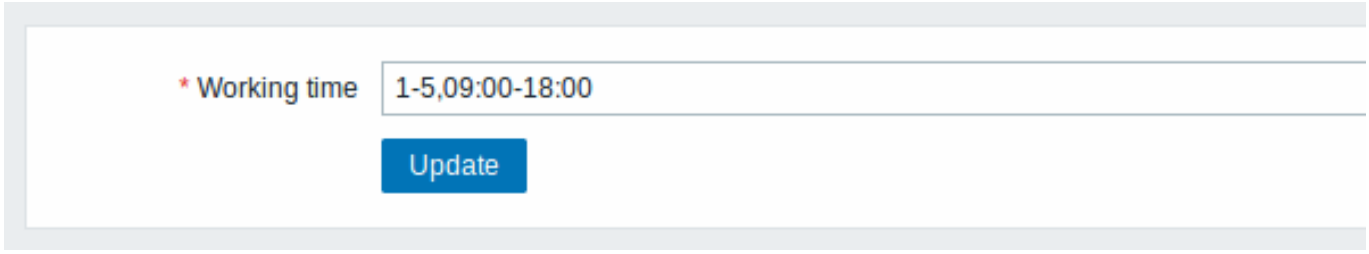
本部分允许管理对于 Zabbix 前端中输入数据的可读表示有用的值映射。



更多细节，参见值映射部分。

9 工作时间

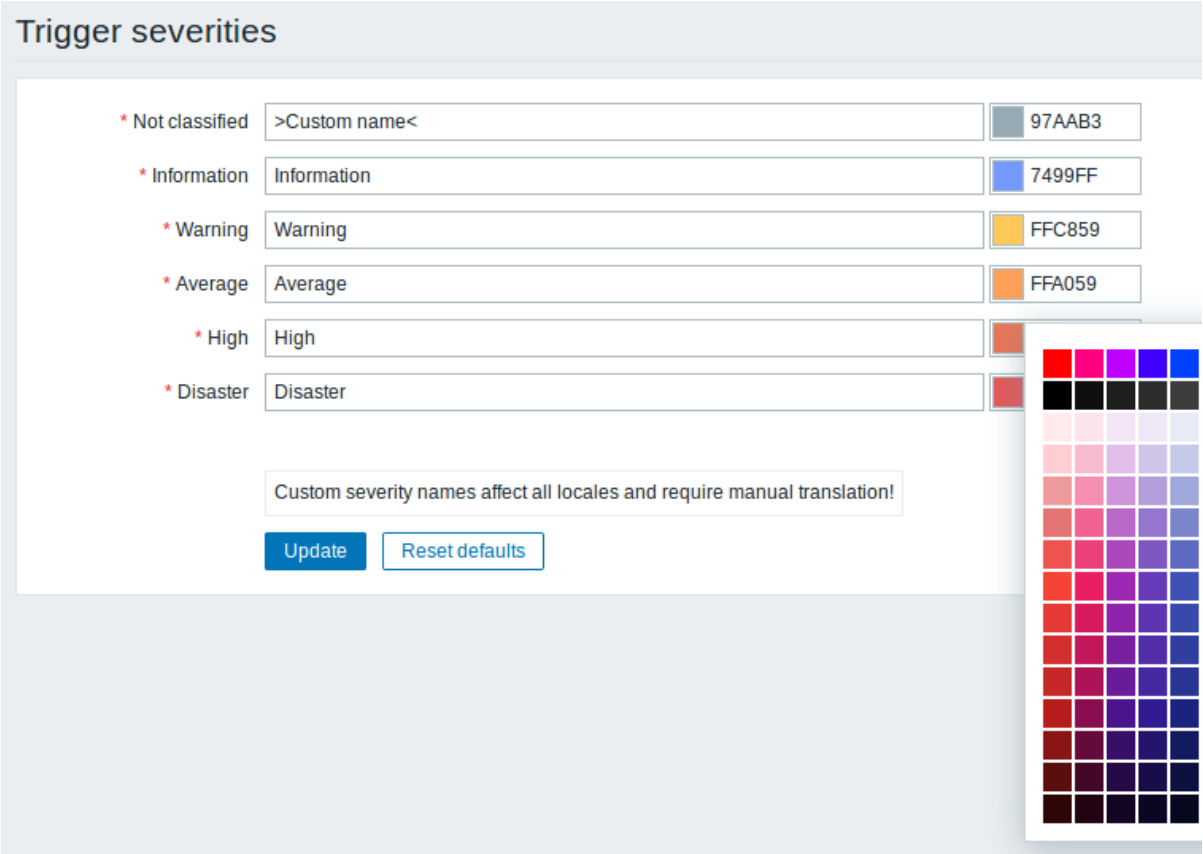
工作时间是系统范围的参数，用于定义工作时间。工作时间显示为图形中的白色背景，而非工作时间显示为灰色。



有关时间格式的说明，请参见时间段规范页面。支持用户宏（自 Zabbix 3.4.0 起）。

10 触发严重级

此部分允许自定义触发严重级名称和颜色

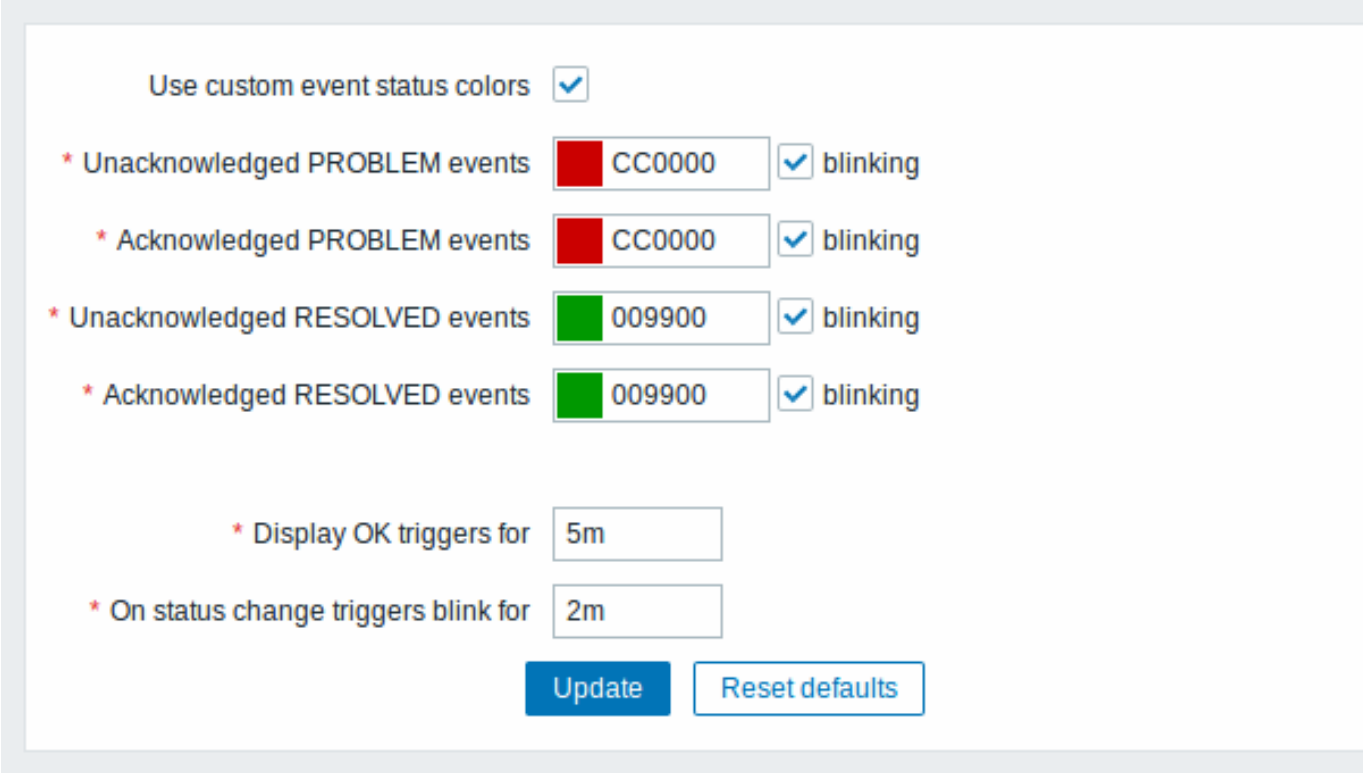


您可以输入新名称和颜色代码，或单击颜色从提供的调色板中选择其他颜色。

有关更多信息，请参见自定义触发严重级页面。

11 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。



在使用 *Use custom event status colors(*自定义事件状态的颜色*)* 选项允许打开的颜色确认/未确认的问题定制。

同样，可以自定义显示 OK 触发器和在触发器状态更改时闪烁的时间段。最大值为 86400 秒（24 小时）。周期字段中支持时间后缀，例如 5m，2h，1d。

12 模块

本部分允许管理自定义的前端模块。

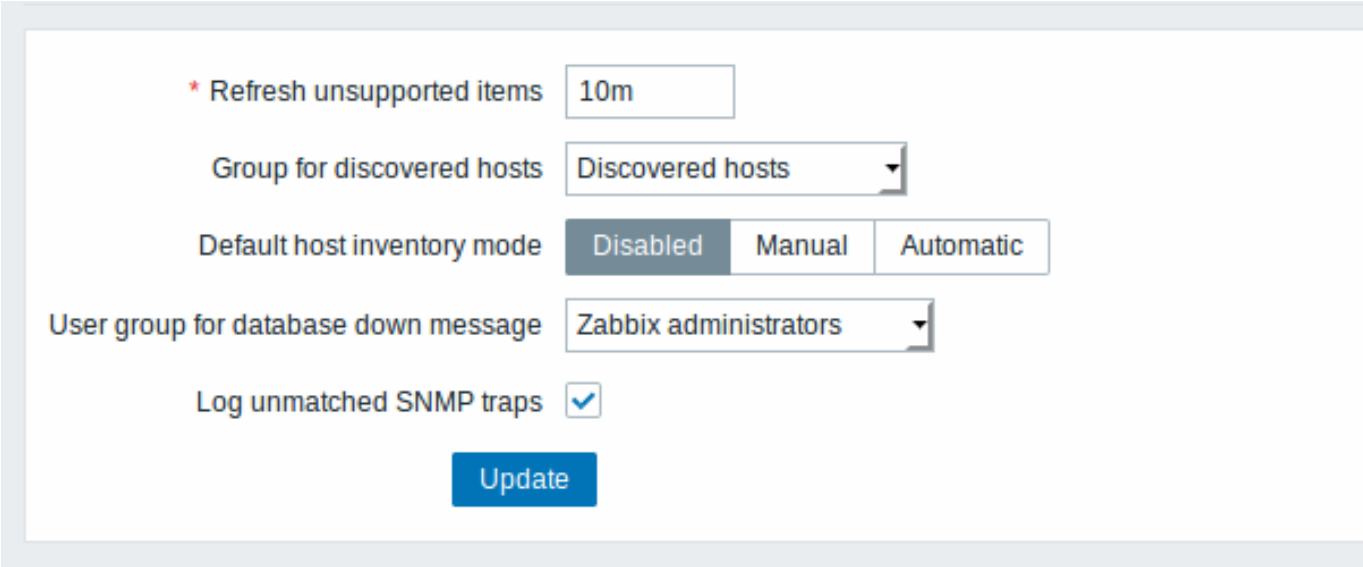
general_modules.png

单击 *Scan directory(*扫描目录*)* 以注册/取消注册任何自定义模块。已注册的模块及其详细信息将显示在列表中。未注册的模块将从列表中删除。

您可以按名称或状态（启用/禁用）过滤模块。单击列表中的模块状态以启用/禁用模块。您也可以通过在列表中选择启用/禁用模块，然后单击列表下方的启用/禁用按钮来批量启用/禁用模块。

13 其他参数

此部分允许配置其他前端参数。



参数	描
*Refresh unsupported items *（刷新不支持的项目）	Zabbix se由于用户参数错误或代理不支持某些项目，因此某些项目可能不受支持。Zab-bix可以配置为定期激活不支持的项目。ver将在此处设置的N个周期(最多1天)内激

参数	描
*Group for discovered hosts *（查找主机的组）	通过网络发现和代理自动注册发现的主机将自动放置在此处选择的主机组中。

参数	描
*Default host inventory mode *（默认主机清单模式）	主机清单的默认模式。 每当服务器或前端创建新的主机或主机原型时，都会遵循该命令，除非在主机发现/自动注册过程中被“设置主机清单模式”操作覆盖。

参数	描
*User group for database down message *（数据库关闭消息的用户组）	Zabbix 服务器取决于发送警报消息的用户组或“无”。端数据库的可用性。没有数据库，它就无法工作。如果数据库关闭，则 Zabbix 可以通知选定的用户。通知将使用所有已配置的用户媒体

参数描	
*Log unmatched SNMP traps *（记录不匹配的 *SNMP* 告警）	如果没有找到对应的 SNMP 接口，则记录SNMP的告警日志。

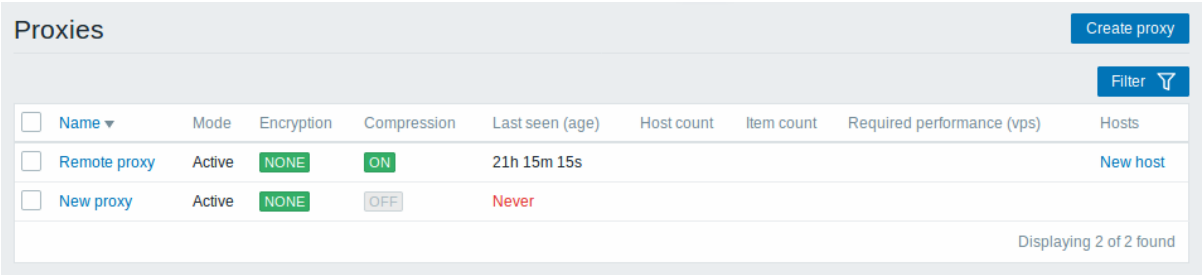
2 代理

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：

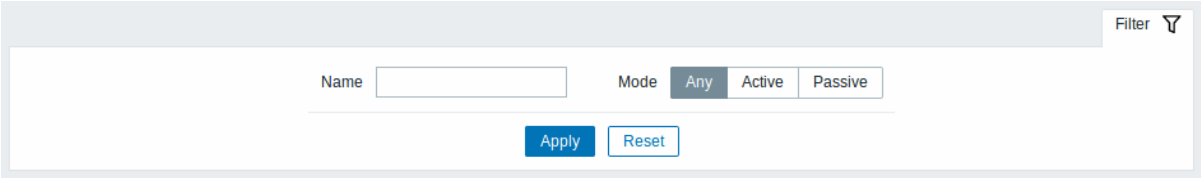
- *Enable hosts* - 将被 proxy 监控的 host 的状态改为 Monitored （监控）
- *Disable hosts* -将被 proxy 监控的 host 的状态改为 Not monitored （不监控）
- *Delete* - 删除 proxy

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。

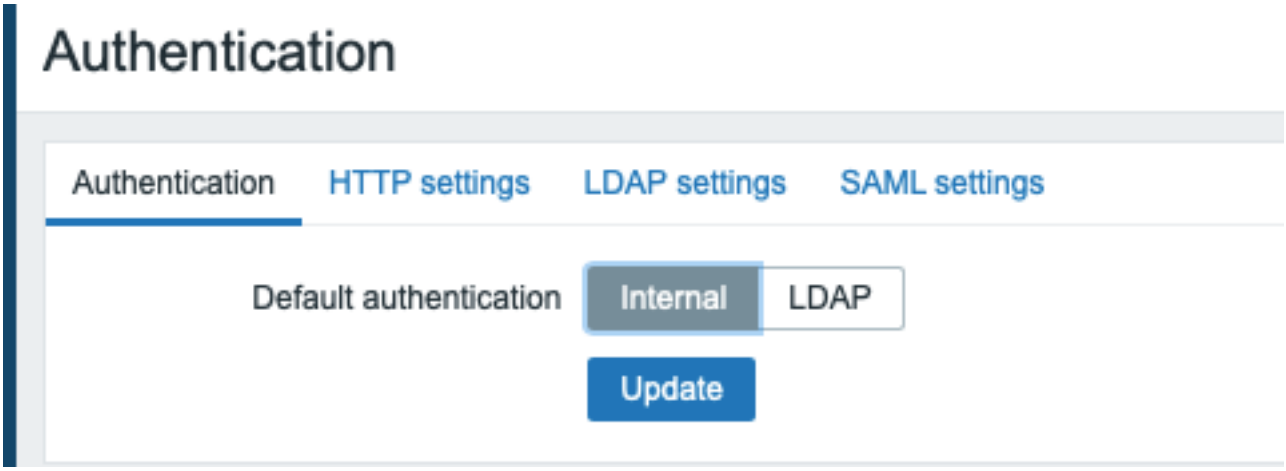


3 身份验证

概述

在 *Administration* → *Authentication*（管理 → 身份验证）验证中，可以指定对 Zabbix 的全局用户身份验证方法。可用的方法有内部，HTTP，LDAP 和 SAML 身份验证。

请注意，可以在用户组级别上微调身份验证方法。



默认情况下，全局使用内部 Zabbix 身份验证。改变：

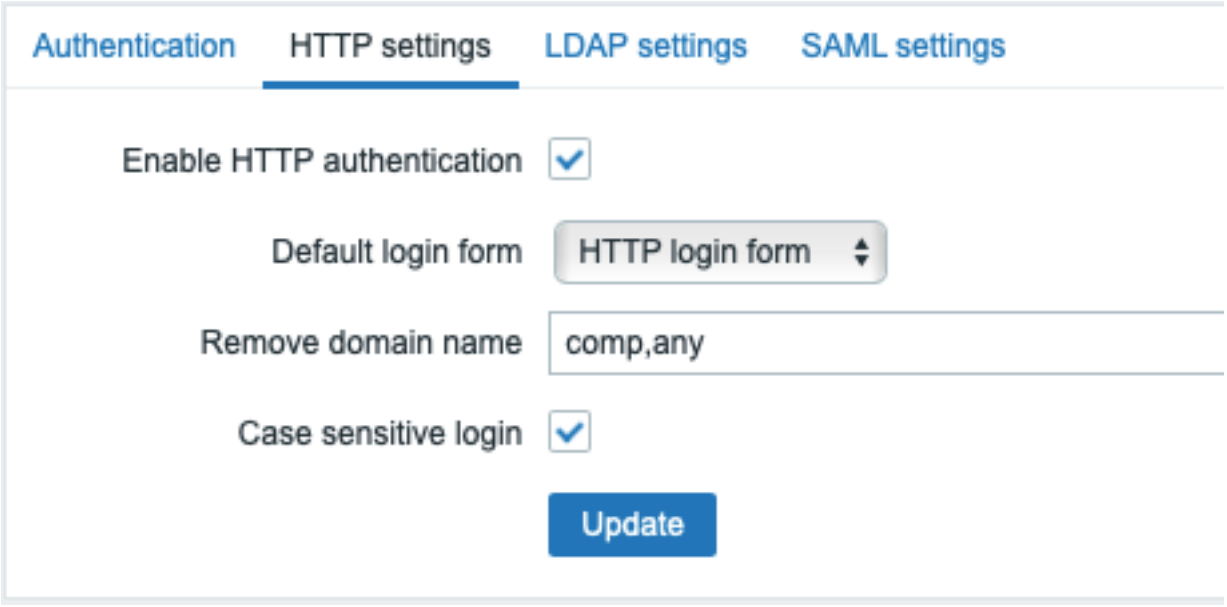
- 到 HTTP-导航到 “ HTTP 设置” 标签并输入身份验证详细信息；
- 到 LDAP-选择 LDAP 作为默认身份验证，然后在 LDAP 设置标签中输入身份验证详细信息；
- 到 SAML-导航到 “ SAML 设置” 标签，然后输入身份验证详细信息。

完成后，单击表单底部的 “更新”。

HTTP 认证

基于 HTTP 或 Web 服务器的身份验证（例如：基本身份验证，NTLM / Kerberos）可用于检查用户名和密码。请注意，用户也必须存在于 Zabbix 中，但是不会使用其 Zabbix 密码。

Attention:
当心！在打开 Web 服务器身份验证之前，请确保已对其进行配置并正常工作。



配置参数：

参数描	说明
*Enable HTTP authentication *（启用 *HTTP* 身份验证）	选中该复选框以启用 HTTP 身份验证。
*Default login form *（默认登录表单） **Zabbi	指定是否将未经身份验证的用户定向到：login form**-**标准 **Zabbix** 登录页面。** **HTTP login form**-HTTP 登录页面。建议 `index_http.php` 仅对页面启用基于 Web 服务器的身份验证。如果将默认登录表单设置为“ HTTP 登录页面”，并且 Web 服务器身份验证模块将在 `$_SERVER` 变量中设置有效的用户登录名，则该用户将自动登录。支持 `$_SERVER` 键 `PHP_AUTH_USER`，`REMOTE_USER`，`AUTH_USER`。

参数描	说明
*Case sensitive login *（区分大小写的登录）例如，禁用区分大小	取消选中该复选框可禁用用户名区分大小写的登录（默认情况下启用）。 的登录并使用 “ ADMIN” 用户登录，即使 Zabbix 用户为 “ Admin”。 请注意，如果区分大小写的登录禁用，则 Zabbix 数据库中存在多个具有相似别名（例如 Admin，admin）的用户时，将拒绝登录。

Note:
如果进行 Web 服务器身份验证，则所有用户（即使前端访问权限设置为 Internal）都将由 Web 服务器而不是 Zabbix 进行身份验证！

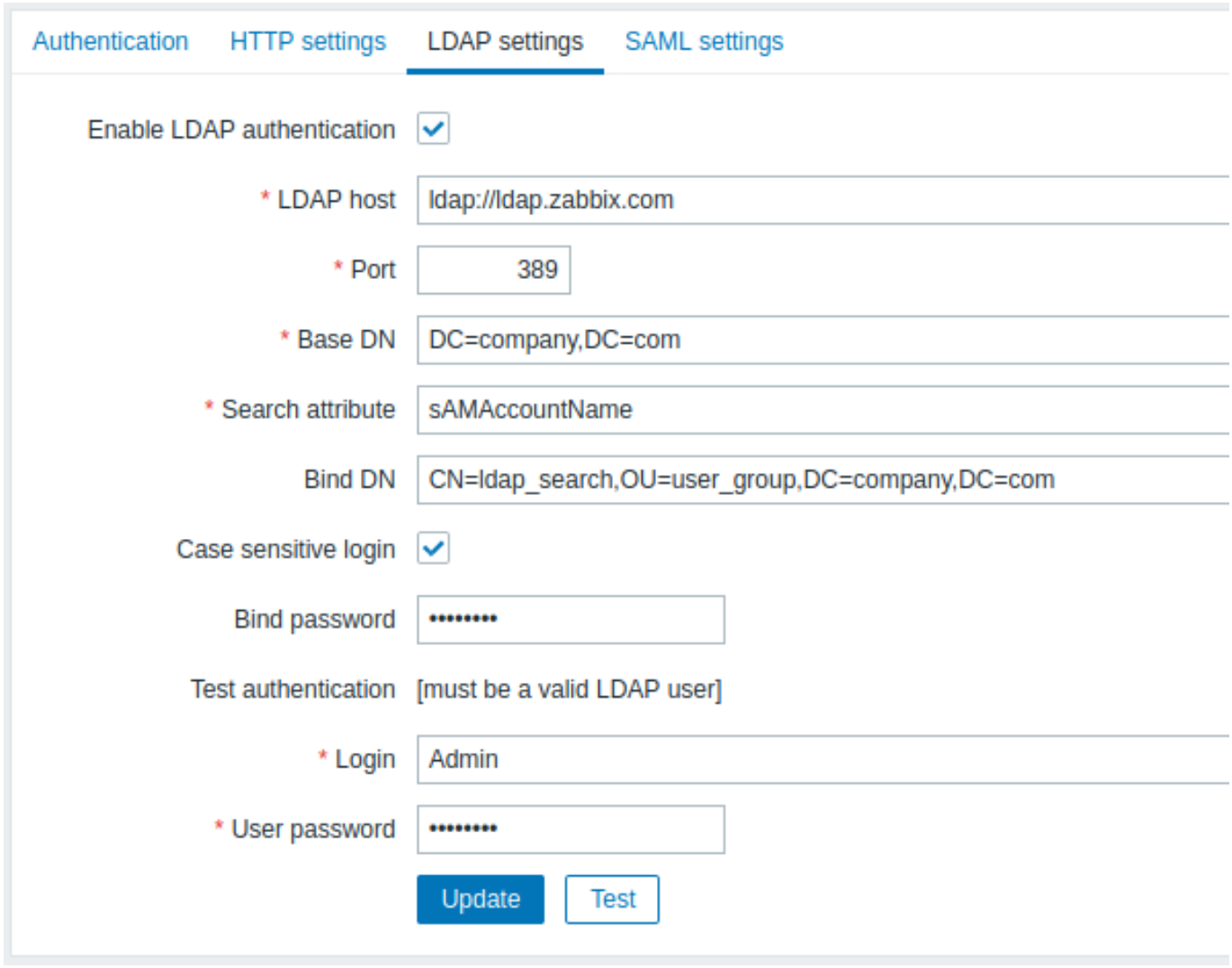
Note:
对于无法使用 HTTP 凭据（默认设置为 HTTP 登录格式）登录的内部用户，导致 401 错误，您可能需要 `ErrorDocument 401 /index.php?form=default` 在基本身份验证指令中添加一行，它将重定向到常规 Zabbix 登录格式。

LDAP 验证

外部 LDAP 身份验证可用于检查用户名和密码。请注意，用户也必须存在于 Zabbix 中，但是不会使用其 Zabbix 密码。

全局设置 LDAP 身份验证后，Zabbix 仍可以对某些用户组进行身份验证。这些组必须将前端访问权限设置为 “内部”。反之亦然，如果全局使用内部身份验证，则可以指定 LDAP 身份验证详细信息，并将其用于前端访问设置为 LDAP 的特定用户组。

Zabbix LDAP 身份验证至少与 Microsoft Active Directory 和 OpenLDAP 一起使用。



配置参数：

参数描	说明
Enable LDAP authentication	启动 LDAP 认证，选中复选框以启用 LDAP 身份验证。
LDAP host	LDAP 服务器名称。例如：ldap：*ldap.zabbix.com * 对于安全的 *LDAP* 服务器，请使用 *ldaps* 协议。* ldaps*：ldap.zabbix.com 在 OpenLDAP 2.xx 和更高版本中，可以使用格式为 ldap：// hostname：port 或 ldaps：// hostname：port 的完整 LDAP URI。

参数描	说明
Port	LDAP 服务器的端口。默认值为 389。 对于安全 LDAP 连接，端口号通常为 636。 使用完整 LDAP URI 时不使用。
Base DN	搜索帐户的基本路径： ou = 用户，ou = 系统（对于 OpenLDAP）， DC = 公司，DC = com（对于 Microsoft Active Directory）
Search attribute	用于搜索的 LDAP 帐户属性： uid（对于 OpenLDAP）， sAMAccountName（对于 Microsoft Active Directory）
Bind DN	用于在 LDAP 服务器上进行绑定和搜索的 LDAP 帐户，例如： uid = ldap_search，ou = system（对于 OpenLDAP）， CN = ldap_search， OU = user_group，DC = company，DC = com（对于 Microsoft Active Directory） 匿名绑定也是支持的。
Case-sensitive login	取消选中该复选框可禁用用户名区分大小写的登录（默认情况下启用）。 例如，禁用区分大小写的登录并使用“ ADMIN”用户登录，即使 Zabbix 用户为“ Admin”。 请注意，如果区分大小写的登录禁用，则 Zabbix 数据库中存在多个具有相似别名（例如 Admin，admin）的用户时，将拒绝登录。

参数描	说明
Bind password	用于绑定和搜索 LDAP 服务器的帐户的 LDAP 密码。
Test authentication	测试部分的标题
Login	测试用户的名称（当前已在 Zabbix 前端中登录）。该用户名必须存在于 LDAP 服务器中。 如果 Zabbix 无法认证测试用户，则不会激活 LDAP 认证。
User password	测试用户的 LDAP 密码。

<note Warning> 万一证书出现问题，为了使 LDAP 连接（ldaps）正常工作，您可能需要 TLS_REQCERT allow 在/etc/openldap/ldap.conf 配置文件中添加一行。这可能会降低与 LDAP 目录连接的安全性。:::

Note:
建议创建一个单独的 LDAP 帐户（Bind DN），以使用 LDAP 中的最小特权在 LDAP 服务器上执行绑定和搜索，而不要使用真实的用户帐户（用于登录 Zabbix 前端）。
这样的方法提供了更高的安全性，并且在用户更改 LDAP 服务器中自己的密码时不需要更改“绑定”密码。
在上表中是 ldap_search 帐户名。

SAML 身份验证

SAML 2.0 身份验证可用于登录 Zabbix。请注意，用户必须存在于 Zabbix 中，但是不会使用其 Zabbix 密码。如果身份验证成功，则 Zabbix 将匹配本地用户名（别名）与 SAML 返回的用户名属性。

Note:
如果启用了 SAML 身份验证，则用户将能够在本地登录或通过 SAML 单一登录之间进行选择。

设置身份提供者

为了使用 Zabbix，需要以以下方式配置 SAML 身份提供程序（onelogin.com，auth0.com，okta.com等）

- 断言使用者 URL 应设置为 `<path_to_zabbix_ui>/index_sso.php?acs`
- 单一登出网址应设置为 `<path_to_zabbix_ui>/index_sso.php?sls`

`<path_to_zabbix_ui>` 示例：

`https://example.com/zabbix/ui, http://another.example.com/zabbix, http://<任意公网IP地址>/zabbix`

设置 ZABBIX

Attention:
如果要在前端使用 SAML 身份验证，则需要安装 php-openssl。

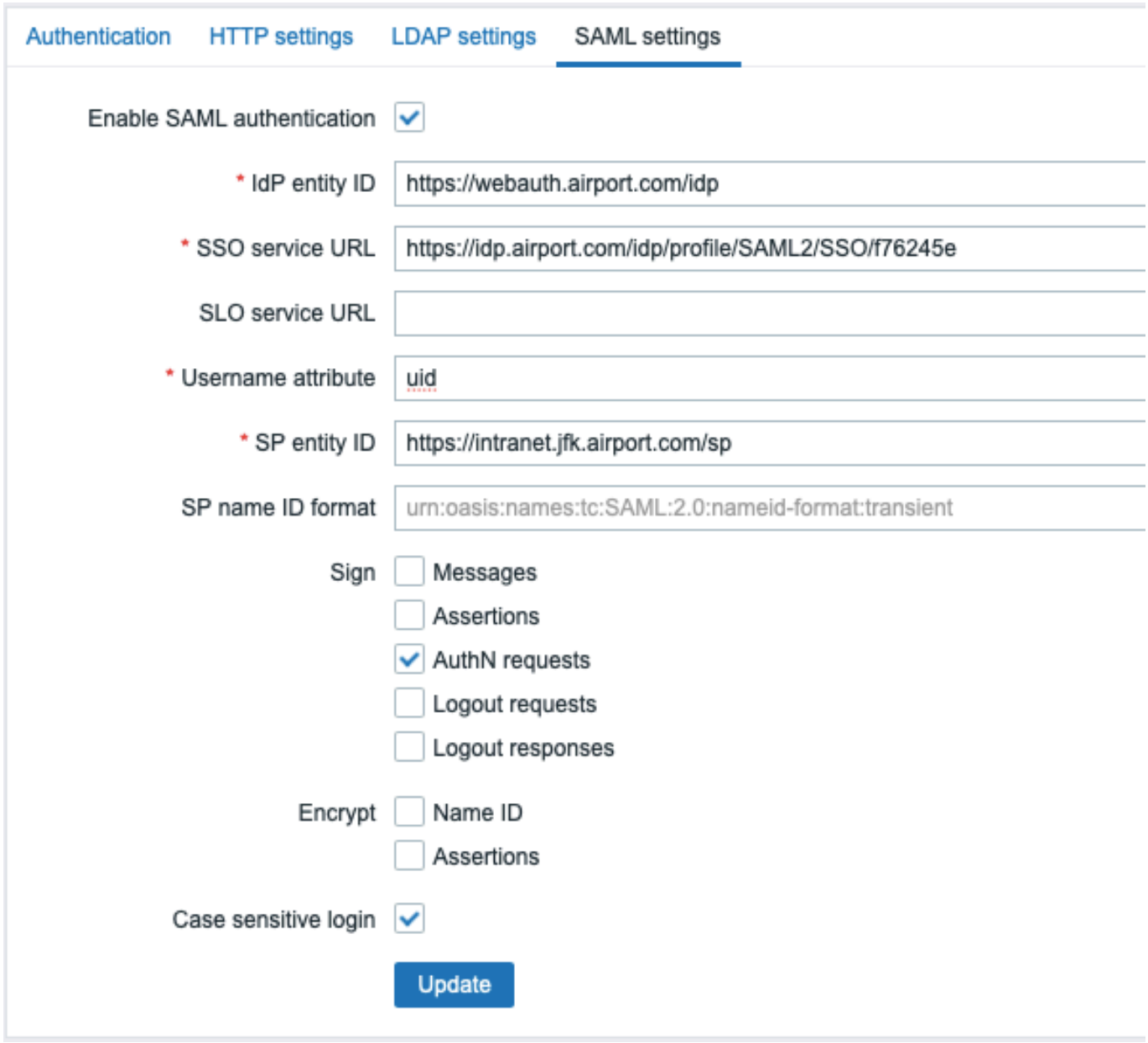
要使用 SAML 身份验证，应按以下方式配置 Zabbix：

1. 除非zabbix.conf.php中提供了自定义路径，否则私钥和证书应存储在 ui/conf/certs/中。

默认情况下，Zabbix 将在以下位置查找：

- ui/conf/certs/sp.key-SP 私钥文件
- ui/conf/certs/sp.crt-SP 证书文件
- ui/conf/certs/idp.crt-IDP 证书文件

2. 所有最重要的设置都可以在 Zabbix 前端中进行配置。但是，可以在配置文件中指定其他设置。



在 Zabbix 前端中可用的配置参数：

参数描	说明
Enable SAML authentication	选中复选框以启用 SAML 身份验证。
IDP entity ID	SAML 身份提供者的唯一标识符。
SSO service URL	登录时，URL 用户将被重定向到.
SLO Service URL	注销时，URL 用户将被重定向到。如果保留为空，将不使用 SLO 服务。

参数描	说明
Username attribute	登录到 Zabbix 时用作用户名的 SAML 属性。 支持的值列表由身份提供商确定。 示例： Examples: uid userprincipalname samaccountname username userusername urn:oid:0.9.2342.19200300.10 urn:oid:1.3.6.1.4.1.5923.1.1.1 urn:oid:0.9.2342.19200300.10
SP entity ID	SAML 服务提供者的唯一标识符。
SP name ID format	定义应使用的名称标识符格式。 示例： urn:oasis:names:tc:SAML:2.0: format:persistent urn:oasis:names:tc:SAML:2.0: format:transient urn:oasis:names:tc:SAML:2.0: format:kerberos urn:oasis:names:tc:SAML:2.0: format:entity
Sign	标记复选框以选择应为其启用 SAML 签名的实体： Messages（消息） Assertions（断言） AuthN requests（AuthN 请求） Logout requests（注销请求） Logout responses（注销响应）
Encrypt	标记复选框以选择应为其启用 SAML 加密的实体： Assertions（断言） Name ID（名称 ID）

参数描	说明
Case-sensitive login	选中该复选框以启用区分大小写的登录名（默认情况下禁用）。 例如，禁用区分大小写的登录并使用“ ADMIN”用户登录，即使 Zabbix 用户为“ Admin”。 请注意，如果区分大小写的登录禁用，则 Zabbix 数据库中存在多个具有相似别名（例如 Admin，admin）的用户时，将拒绝登录。

高级设置

可以在 Zabbix 前端配置文件（*zabbix.conf.php*）中配置其他 SAML 参数：

- $ SSO ['SP_KEY'] ='<SP 私钥文件的路径 >';
- $ SSO ['SP_CERT'] ='<SP 证书文件的路径 >';
- $ SSO ['IDP_CERT'] ='<IDP 证书文件的路径 >';
- $ SSO ['SETTINGS']

Note:
Zabbix 使用OneLogin 的 SAML PHP 工具包库（版本 3.4.1）。$ SSO ['SETTINGS'] 部分的结构应类似于库使用的结构。有关配置选项的说明，请参见官方库文档。

只能将以下选项设置为 $ SSO ['SETTINGS'] 的一部分：

- strict（严格的）
- compress（压缩）
- contactPerson（联系人）
- organization（组织）
- sp（仅此列表中指定的选项）

```
  * attributeConsumingService
  * x509certNew

* idp（仅此列表中指定的选项）
  * singleLogoutService（仅一个选项）
    * responseUrl（响应Url）

  * certFingerprint（证书指纹）
  * certFingerprintAlgorithm（证书指纹算法）
  * x509certMulti
```

- 安全性（仅此列表中指定的选项）
  - signMetadata（符号元数据）
  - wantNameId（想要的名称标识）
  - requestAuthnContext（请求认证上下文）
  - requestAuthnContextComparison（请求认证上下文比较）
  - wantXMLValidation（想要的 XML 验证）
  - RelaxDestinationValidation（目的验证）
  - destinationStrictlyMatches（严格匹配）
  - rejectUnsolicitedResponsesWithInResponseTo（拒绝未经请求的响应）
  - signatureAlgorithm（签名算法）
  - digestAlgorithm（摘要算法）
  - lowercaseUrlencoding

所有其他选项将从数据库中获取，并且不能被覆盖。在调试选项将被忽略。

配置示例：

```
$SSO['SETTINGS'] = [
  'security' => [
      'signatureAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha384'
      'digestAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#sha384',
      // ...
  ],
  // ...
];
```

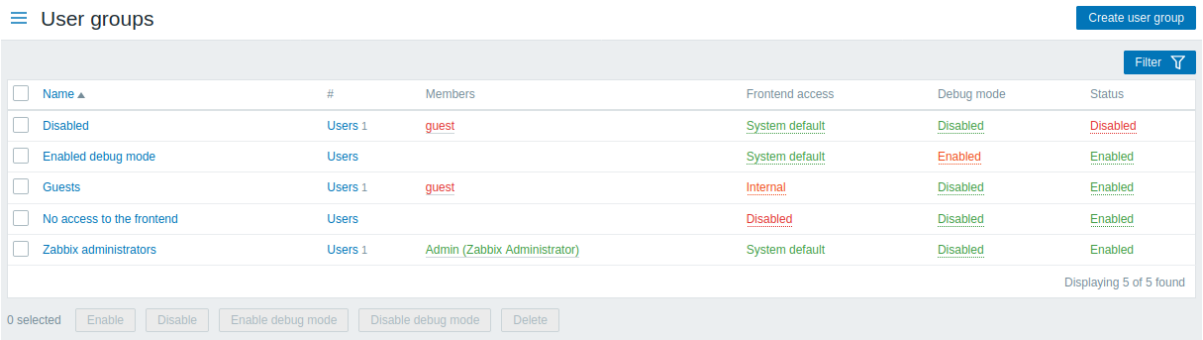
4 用户组

概述

在 *Administration* → *User groups*（管理 → 用户组）部分中，维护系统的用户组。

用户组

显示现有用户组及其详细信息的列表。



显示的数据：

参数描	说明
Name	用户组名称。单击用户组名称可打开用户组配置表单.
#	群组内的用户数。单击 *Users* 将显示在用户列表中过滤掉的各个用户。
Members	用户组中单个用户的别名 (姓名和姓氏在括号中)。单击别名将打开用户配置表单。来自残疾组的用户显示为红色。
Frontend access	显示前端访问级别： **System default** - Zabbix，LDAP 或 HTTP 身份验证；取决于所选择的身份验证方法 **Internal** - 用户通过 Zabbix 进行身份验证，而不考虑系统设置 **Disabled** - 禁用此用户的前端访问 通过单击当前级别可以更改它。
Debug mode	显示调试模式 状态 - *Enabled(启用)* 或 *Disabled(禁用)*. 通过单击状态可以更改它。

参数描	说明
Status	显示用户组状态- 启用或// 禁用//。通过单击状态可以更改它。

要配置新用户组，请单击右上角的 *Create user group(*创建用户组*)* 按钮.

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

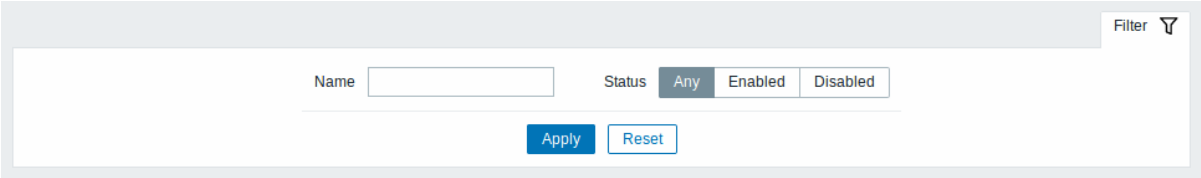
- *Enable* - 将用户组状态更改为 *Enabled*（启用）
- *Disable* - 将用户组状态更改为 *Disabled*（禁用）
- *Enable debug mode*（启用调试模式） - 为用户组启用调试模式
- *Disable debug mode*（禁用调试模式） - 禁用用户组的调试模式
- *Delete* - 删除用户组

要使用这些选项，请在各个用户组之前标记复选框，然后单击所需的按钮。

筛选

由于列表可能包含许多用户组，因此可能需要过滤掉您真正需要的用户组。

Filter（过滤器）链接位于用户组列表上方。如果单击它，将提供一个过滤器，您可以在其中按名称和状态过滤用户组。



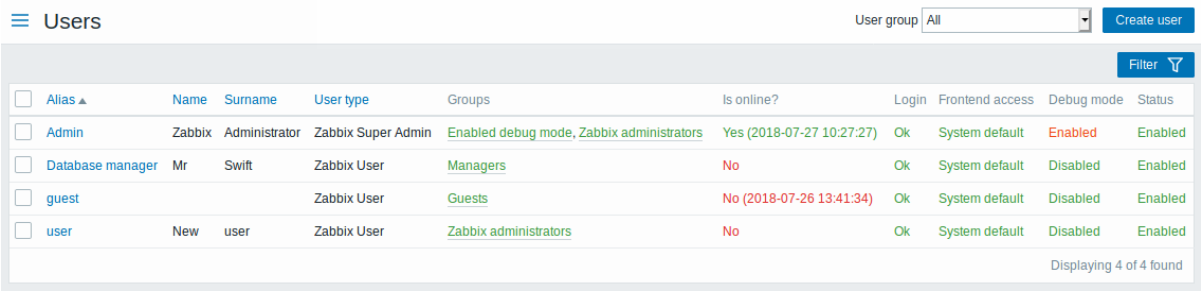
5 用户

概述

在 *Administration* → *Users*（管理 → 用户）部分中，维护系统的用户。

用户

显示现有用户及其详细信息的列表。



从 *Users(*用户*)* 栏中右侧的下拉菜单中，您可以选择显示所有用户还是显示属于某个特定组的用户。

显示的数据：

参数描	说明
Alias	用户的别名，用于登录 Zabbix。单击别名将打开用户配置表单.
Name	用户的名字.
Surname	用户的名字.
User type	显示用户类型 - *Zabbix Super Admin*, *Zabbix Admin* 或 *Zabbix User*.

参数描	说明
Groups	列出了用户所属的组。单击用户组名将打开用户组配置表单。禁用的群组以红色显示。
Is online?	用户的在线状态显示为“是”或“否”。括号中显示用户最后一次活动的时间。
Login	用户的登录状态显示为“Ok”或“Blocked”。当用户尝试登录失败超过 5 次时，用户可能会被暂时阻止。通过单击 Blocked，您可以解除对该用户的阻止。
Frontend access	显示前端访问级别 - *System default*（默认），*Internal*（内部）或 *Disabled*（禁用），取决于整个用户组的一组。
Debug mode	显示调试模式状态 - *Enabled*（启用） or *Disabled*（禁用），具体取决于整个用户组的一组。
Status	显示用户状态 - *Enabled*（启用） or *Disabled*（禁用），取决于整个用户组的一组.

要配置新用户，请单击右上角的 *Create user*（创建用户）按钮。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

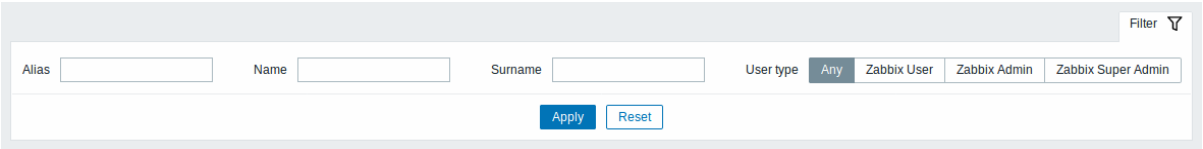
- *Unblock*（取消阻止） - 重新启用对被阻止用户的系统访问权限
- *Delete* - 删除用户

要使用这些选项，请在各个用户之前标记复选框，然后单击所需的按钮。

筛选

由于列表可能包含许多用户，因此可能需要过滤掉您真正需要的用户。

Filter（过滤器）链接位于用户列表上方。如果单击它，将提供一个过滤器，您可以在其中按别名，名称，姓氏和用户类型过滤用户。



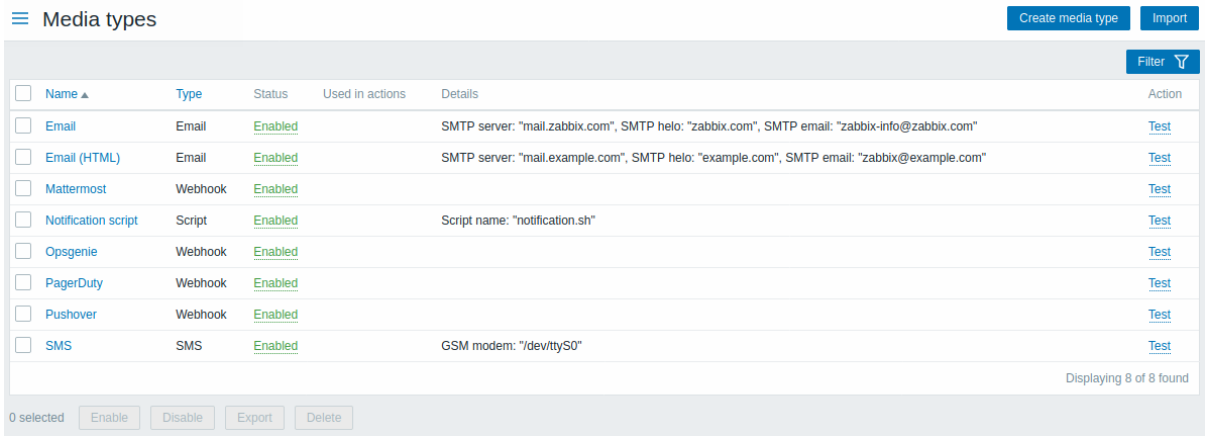
6 媒体类型

概述

在 *Administration* → *Media types*（管理 → 媒体类型）部分中，用户可以配置和维护媒体类型信息。

媒体类型信息包含有关将媒体用作通知的传递渠道的一般说明。特定的详细信息（例如向其发送通知的单个电子邮件地址）由单个用户保存。

显示现有媒体类型及其详细信息的列表。



显示的数据：

参数描	说明
Name	媒体类型的名称。单击名称将打开媒体类型配置表单.
Type	显示媒体类型（电子邮件，SMS 等）。
Status	显示介质类型状态 - *Enabled(*启用*)* 或 *Disabled(*禁用*)*. 通过单击状态可以更改它。
Used in actions	将显示所有直接使用媒体类型的操作 (在 *Send only to*（仅发送到）下拉列表中选择)。单击动作名称将打开动作配置表单。
Details	显示媒体类型的详细信息。

要配置新的媒体类型，请单击右上角的 *Create media type*（创建媒体类型）按钮.

要从 XML 导入媒体类型，请单击右上角的 “导入” 按钮。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

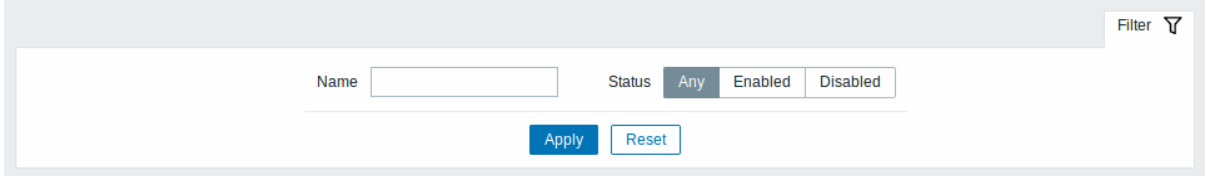
- *Enable* - 将媒体类型状态更改为 *Enabled*（启用）
- *Disable* - 将媒体类型状态更改为 *Disabled*（禁用）
- *Export* - 将媒体类型导出到 XML 文件
- *Delete* - 删除媒体类型

要使用这些选项，请在相应的媒体类型之前选中复选框，然后单击所需的按钮。

筛选

由于列表可能包含多种媒体类型，因此可能需要过滤掉您真正需要的媒体类型。

Filter（过滤器）链接位于媒体类型列表上方。如果单击它，则将提供一个过滤器，您可以在其中按名称和状态过滤媒体类型。



7 脚本

概述

在 *Administration* → *Scripts*（管理 → 脚本）部分中，可以配置和维护用户定义的全局脚本。每个脚本可以根据需要应用于不同的主机。另请参见命令执行.

根据设置的用户权限，可以通过在各个前端位置 （*Dashboard, Problems, Latest data, Maps*）上单击主机来执行脚本，这些脚本也可以作为操作操作运行。这些脚本在 Zabbix server (proxy) 或 agent 上执行

默认情况下，Zabbix agent 和 Zabbix proxy 远程脚本均处于禁用状态。可以通过以下方式启用它们：

- `AllowKey=system.run[*]` 在代理配置中添加参数；
- 在代理配置中（还将在 Zabbix 5.0.2 之前的代理配置中）将 EnableRemoteCommands 参数设置为 “1”

将显示现有脚本的列表及其详细信息。

显示数据:

参数描	说明
Name	脚本的名称。单击脚本名称将打开脚本配置表单.
Type	“脚本类型” 显示为 “脚本” 或 “IPMI 命令”。
Execute on	显示该脚本是在 Zabbix server 或 agent 上执行。
Commands	显示脚本中需要执行的所有命令。
User group	显示脚本可用的用户组 (或所有用户组)。
Host group	显示脚本可用的主机组 (或所有主机组)。
Host access	主机组的权限级别显示为 “读” 或 “写”。只有具有所需权限级别的用户才能访问脚本的执行。

要配置新脚本，请单击右上角的 *Create script*（创建脚本）按钮。

编辑选项

列表下方的按钮提供了一个批量编辑选项:

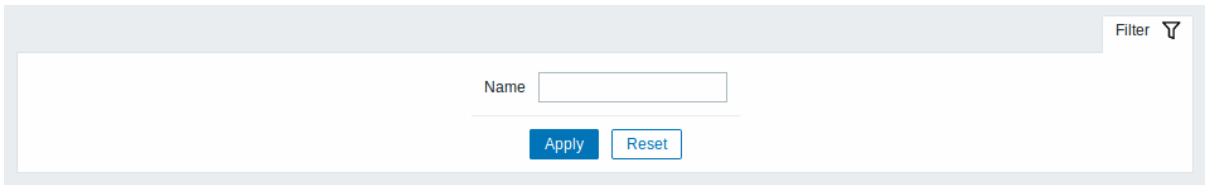
- *Delete* - 删除脚本

要使用此选项，请在各自的脚本之前标记复选框并单击 *Delete*。

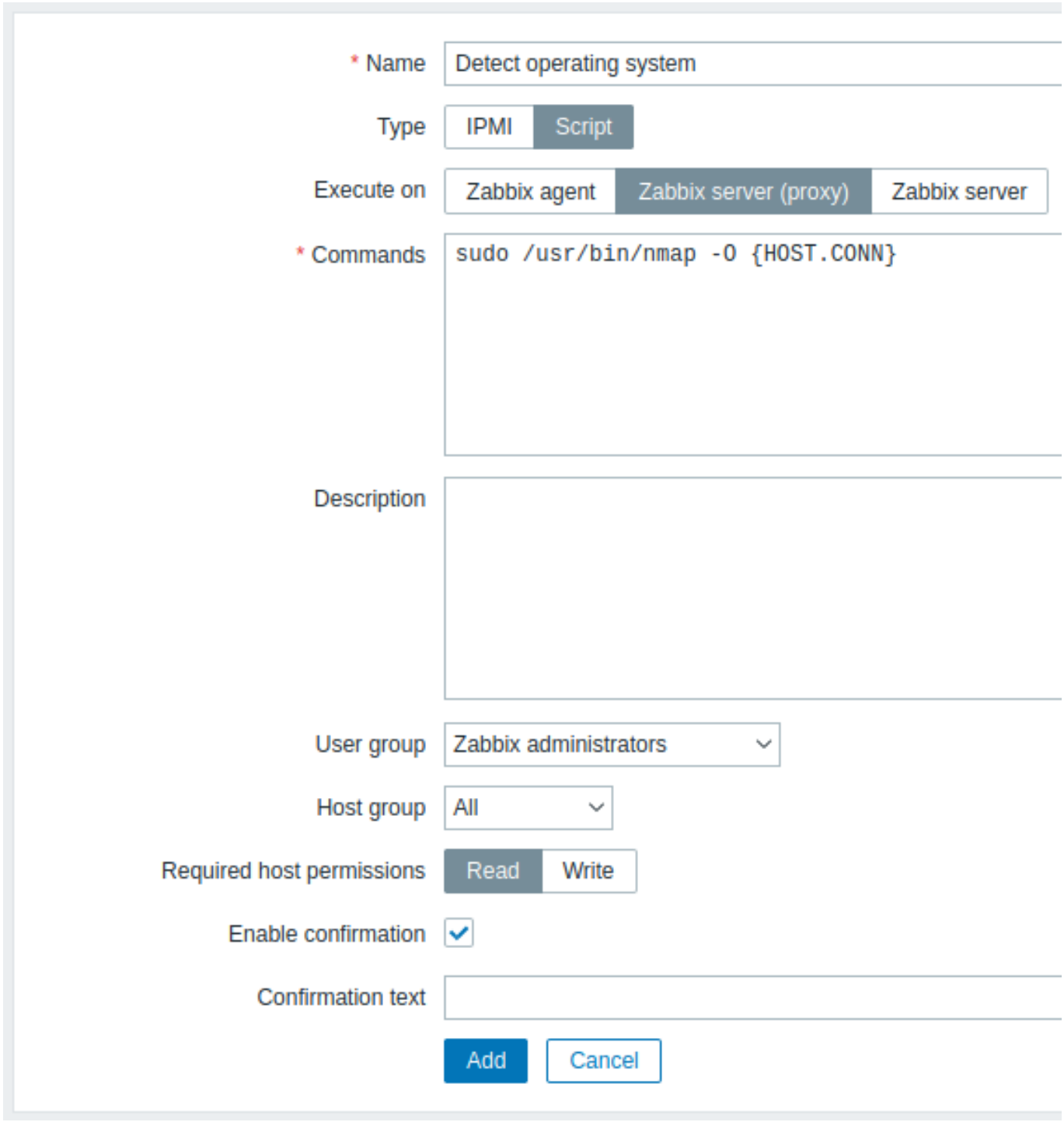
筛选

由于列表可能包含许多脚本，因此可能需要过滤掉您真正需要的脚本。

该 *Filter(过滤器)* 链接可用脚本列表上方。如果单击它，将提供一个过滤器，您可以在其中按名称过滤脚本。



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	说明
Name	脚本的唯一名称。 从 Zabbix 2.2 开始，该名称可以带有所需路径的前缀，例如 `Default/`，将脚本放入相应目录中。通过监视部分中的菜单访问脚本时，将根据给定目录对其进行组织。 脚本的名称不能与现有目录相同（反之亦然）。脚本名称在其目录中必须唯一。 未转义的脚本名称经过唯一性验证，即“ Ping”和“ \Ping”不能添加到同一文件夹中。单个反斜杠会在其后直接转义任何符号。例如，字符“ /”和“ \”可以用反斜杠(即\ /或\\）转义。
Type	单击相应的按钮以选择脚本类型-IPMI命令或脚本。
Execute on	单击相应的按钮以在以下脚本上执行脚本： **Zabbix agent** - 该脚本将由Zabbix agent 在主机上执行（如果允许 system.run 项） **Zabbix server (proxy)** - 该脚本将由 Zabbix server 或 proxy(如果启用了enableremotecommand) - 取决于主机是由 server 或 proxy。 **Zabbix server** - 该脚本仅由Zabbix server 执行

参数描	说明
Commands	输入脚本中要执行的命令的完整路径。 支持以下宏: host-related - {HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}; user-related - {USER.ALIAS}, {USER.FULLNAME}, {USER.NAME}, {USER.SURNAME} \\用户自定义宏。 说明： 如果宏可能会解析为带有空格的值(例如，主机名)，不要忘记在需要时引用引号。 从 Zabbix 5.0.2 开始，就支持与用户相关的宏，以允许传递启动脚本的用户的信息。如果脚本在操作操作下自动执行，这些宏将不会被解析。
Description	输入脚本的描述。
User group	选择脚本将可用的用户组 (或者对所有用户组都可用)。
Host group	选择脚本将用于的主机组 (或者对所有用户组都可用)。
Required host permissions	选择主机组的权限级别—读或写。只有具有所需权限级别的用户才能访问脚本的执行。
Enable confirmation	在执行脚本之前，标记复选框以显示确认消息。对于有潜在危险的操作(如重启脚本) 或可能需要很长时间的操作，此特性可能特别有用。

参数描	说明
Confirmation text	为上面的复选框启用的确认弹出框输入自定义确认文本(例如，远程系统将重新启动)。你确定吗?)。要查看文本的样子，请单击字段旁边的 *Test confirmation*。 确认文本也支持脚本命令所支持的所有宏。 注意: 当测试确认消息时，宏将不会被展开。

脚本执行和结果

Zabbix 服务器运行的脚本按照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出现的弹出窗口中。

注意：脚本的返回值是标准输出以及标准错误。

请参见下面的脚本示例和结果窗口：

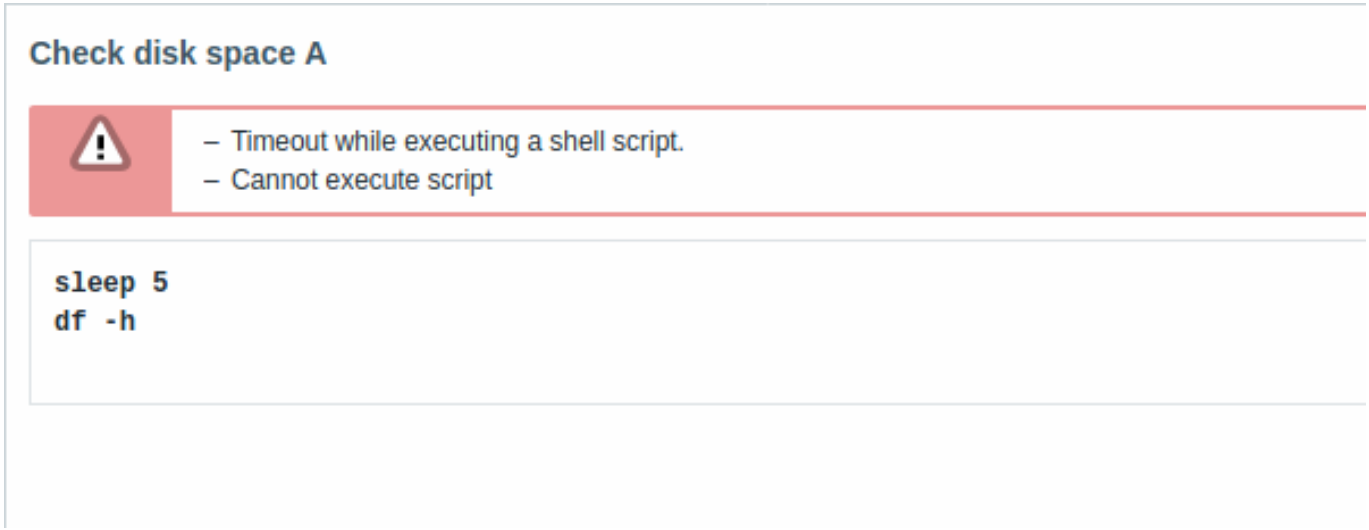
```
uname -v
/tmp/non_existing_script.sh
echo "This script was started by {USER.ALIAS}"
```

Script timeout

Zabbix agent

You may encounter a situation when a timeout occurs while executing a script.

See an example of a script running on Zabbix agent and the result window below:



The error message, in this case, is the following:

```
Timeout while executing a shell script.
```

In order to avoid such a situation, it is advised to optimize the script itself (instead of adjusting Timeout parameter to a corresponding value (in our case, > '5') by modifying the Zabbix agent configuration and Zabbix server configuration).

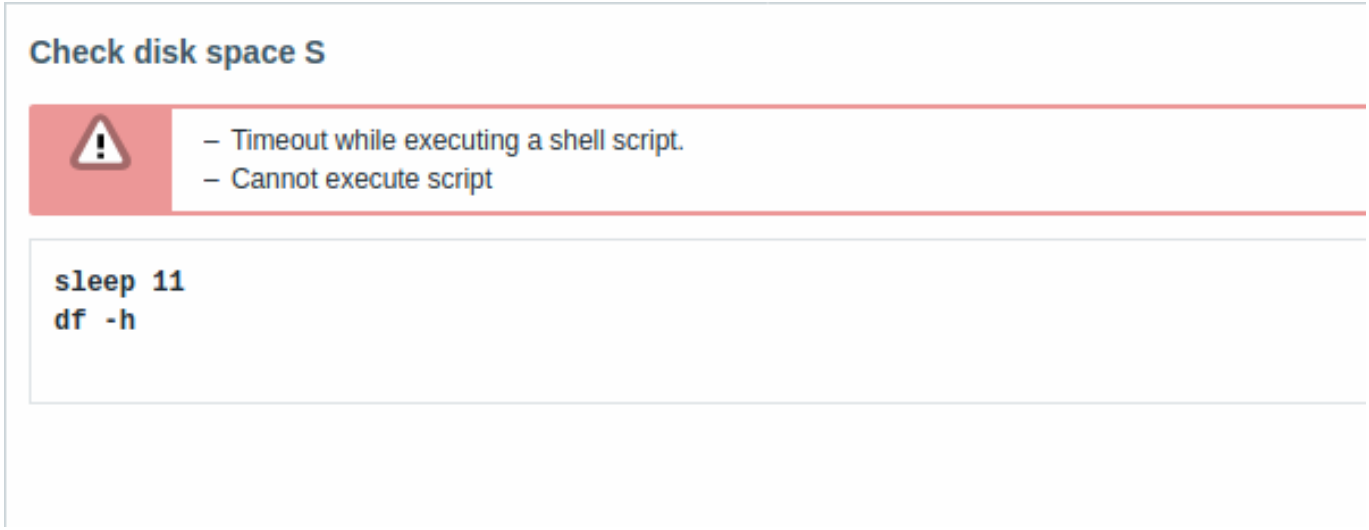
In case still the Timeout parameter is changed in Zabbix agent configuration following error message appears:

```
Get value from agent failed: ZBX_TCP_READ() timed out.
```

It means that modification was made in Zabbix agent configuration and it is required to modify Timeout setting also in Zabbix server configuration.

Zabbix server/proxy

See an example of a script running on Zabbix server and the result window below:



It is also advised to optimize the script itself (instead of adjusting TrapperTimeout parameter to a corresponding value (in our case, > '11') by modifying the Zabbix server configuration).

8 队列

概述

在 *Administration* → *Queue(*管理 → 队列*)* 中，显示等待更新的监控项。

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，请参阅队列 部分。

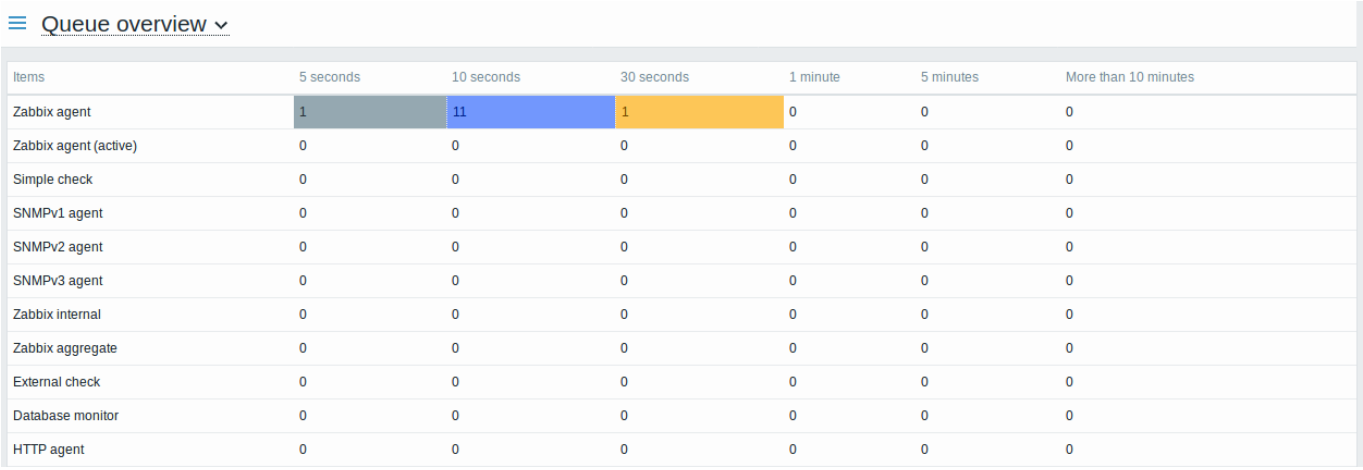
Note:
队列仅在 Zabbix server 运行时可用。

从标题下拉列表中，您可以选择：

- 队列概览（按项目类型）
- proxy 队列概述
- 延迟监控项清单

按项目类型分类概述

在此屏幕中，很容易找到问题是否与一种或多种项目类型有关。



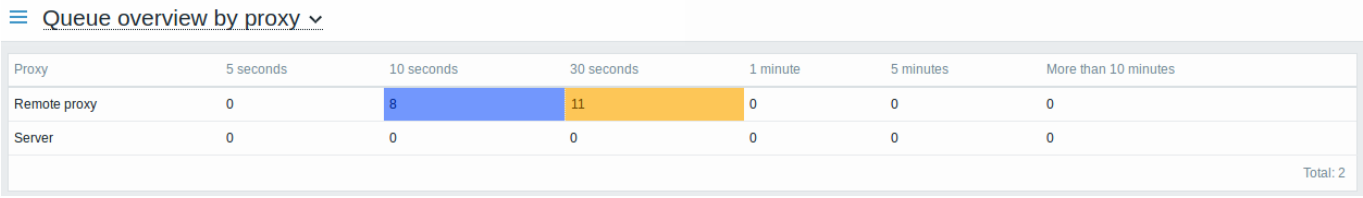
Queue overview

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

每行包含一个项目类型。每列显示等待项的数量-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。

代理概述

在此屏幕中，很容易找到问题是否与代理之一或服务器有关。



Queue overview by proxy

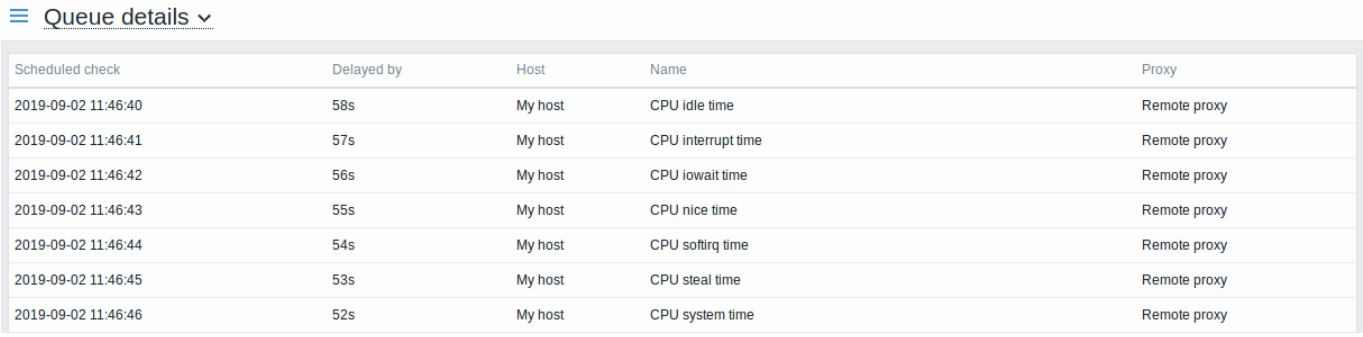
Proxy	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Remote proxy	0	8	11	0	0	0
Server	0	0	0	0	0	0

Total: 2

每行包含一个 proxy，服务器位于列表的最后。每列显示等待项的数量-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。

等待项目清单

在此屏幕中，列出了每个等待的项目。



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

显示的数据：

参数描	说明
Scheduled check	显示检查到期的时间。
Delayed by	显示延迟时间。
Host	显示项目的主机。
Name	显示等待项目的名称。

参数描	说明
Proxy	如果由 proxy 监视主机，则显示代理名称。

可能的错误信息

当没有数据显示并且出现以下错误消息时，您可能会遇到以下情况：

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 server 时，会发生这种情况。

3 用户资料

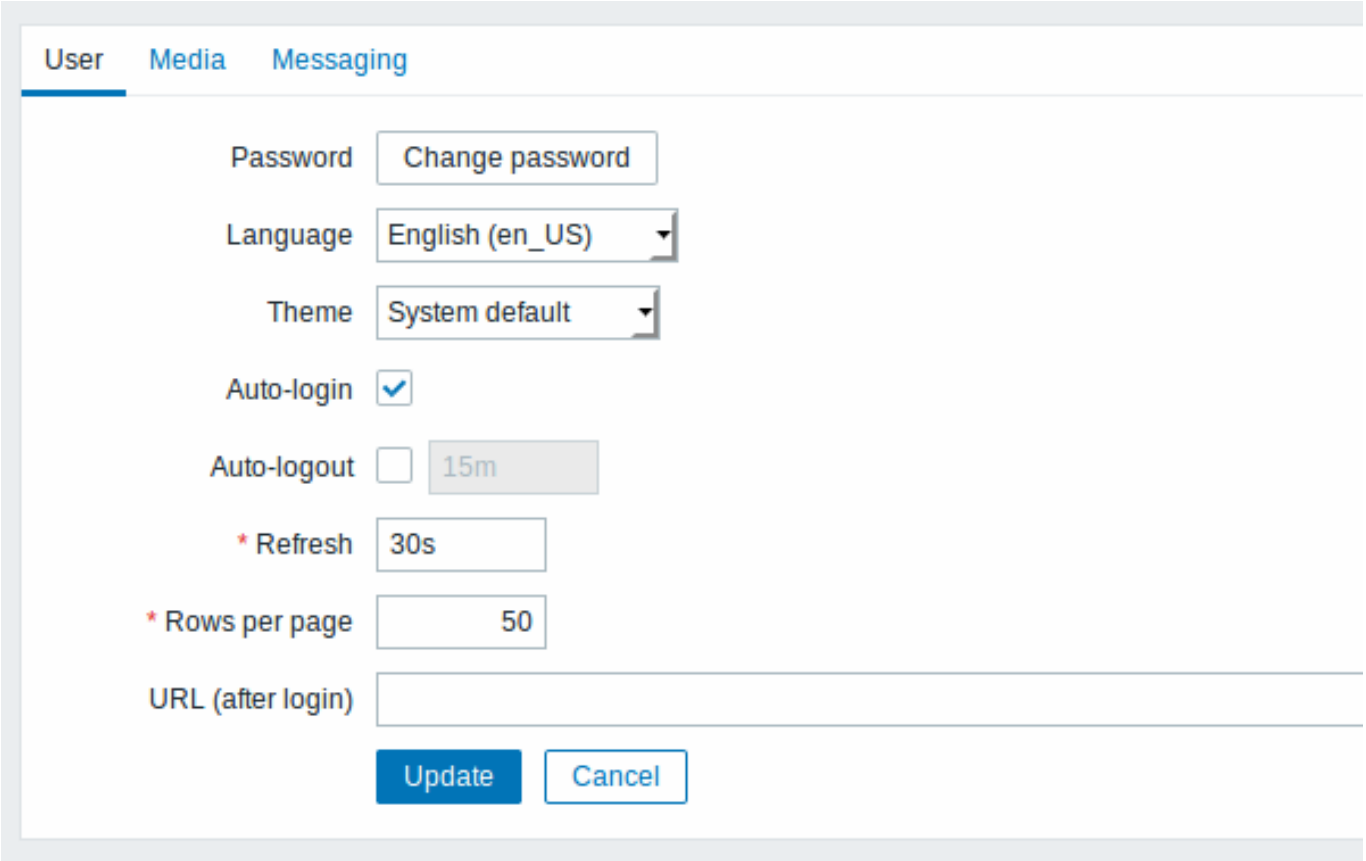
概述

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。

点击 zabbix 窗口右上角的 来访问用户信息.

配置

User 选项卡允许您设置关于用户相关配置.

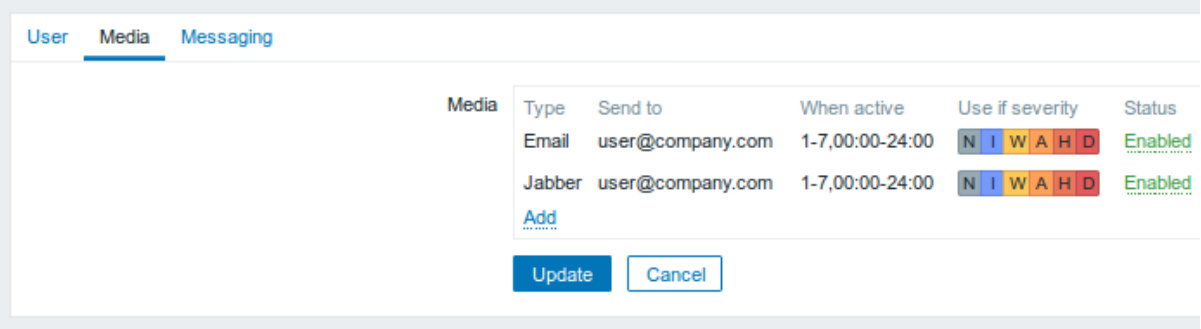


Parameter 参数 D	scription 描述
Password	点击链接显示两个字段，来输入新的密码。

Parameter 参数 D	scription 描述
Language	选择您想要的界面语言。 PHP 的 gettext 扩展是翻译正常运作所必需的。
Theme	为您的资料选择一种特殊的颜色主题: **System default** - 使用默认系统设置 **Blue** - 标准蓝色主题 **Dark** - 暗黑主题 **High-contrast light** - 高对比的浅色主题 **High-contrast dark** - 高对比的暗黑主题
Auto-login	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	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 支持的。如 90s, 5m, 2h, 1d。 请注意，以下选项不起作用： * 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； * 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； * 当选中”30 天记住我” 选项，请登录。
Refresh	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 支持的。如 30s, 5m, 2h, 1d。
Rows per page	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)	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:
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。

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:
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。

可通过 **Messaging** 选项卡, 设置全局通知global notifications。

参考

1. How to install additional locales to be able to select unavailable languages in the user profile

1 全局通知

概述

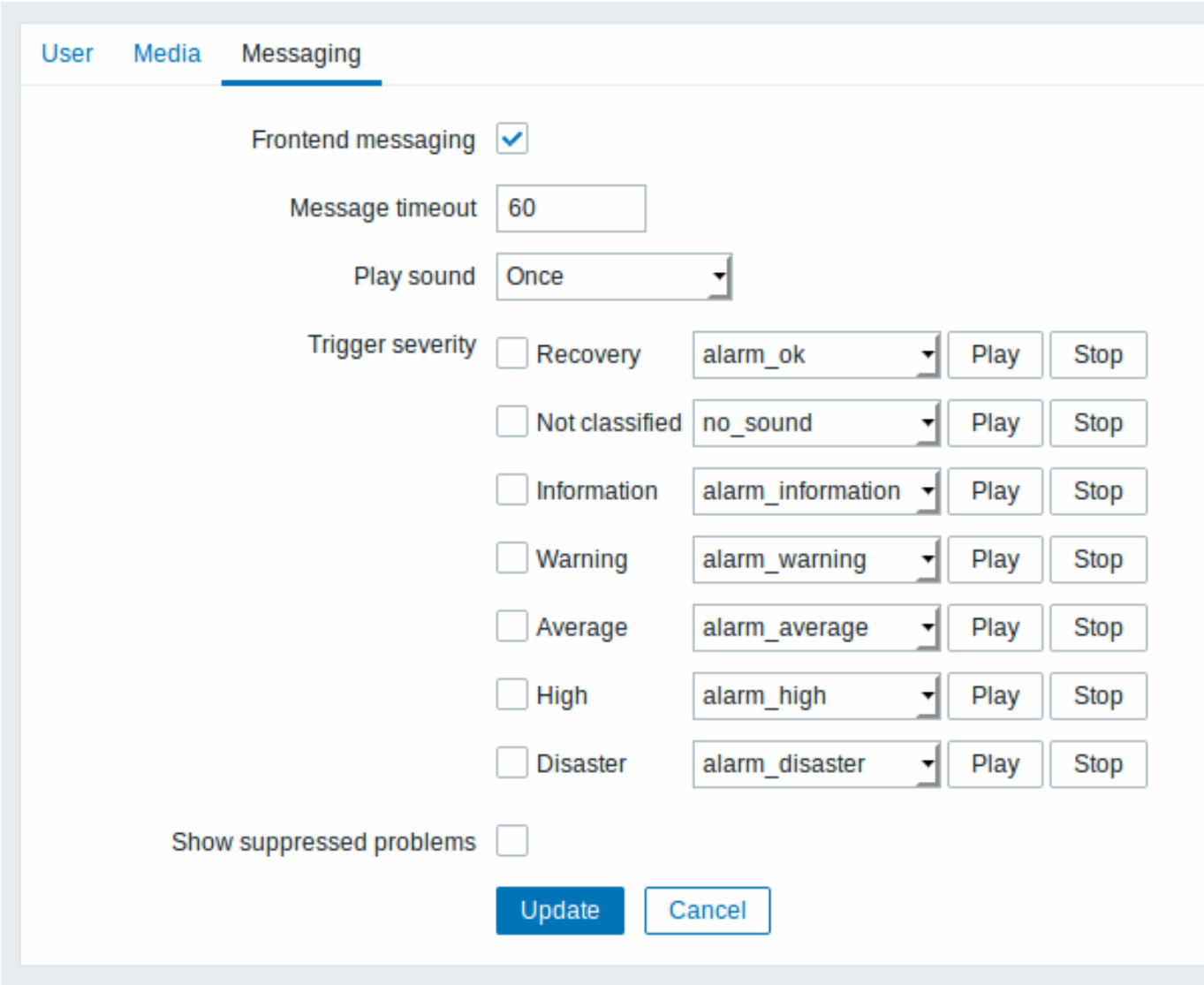
全局通知是一种在 Zabbix 前端屏幕上显示当前正在发生的问题的方法。

如果没有全局通知，则在问题或仪表板之外的其他位置工作将不会显示有关当前正在发生的问题的任何信息。全局通知将显示此信息，无论您在哪里。

全局通知涉及到信息的显示和playing a sound。

配置

可以在profile configuration的// Messaging //选项卡中为每个用户启用全局通知。

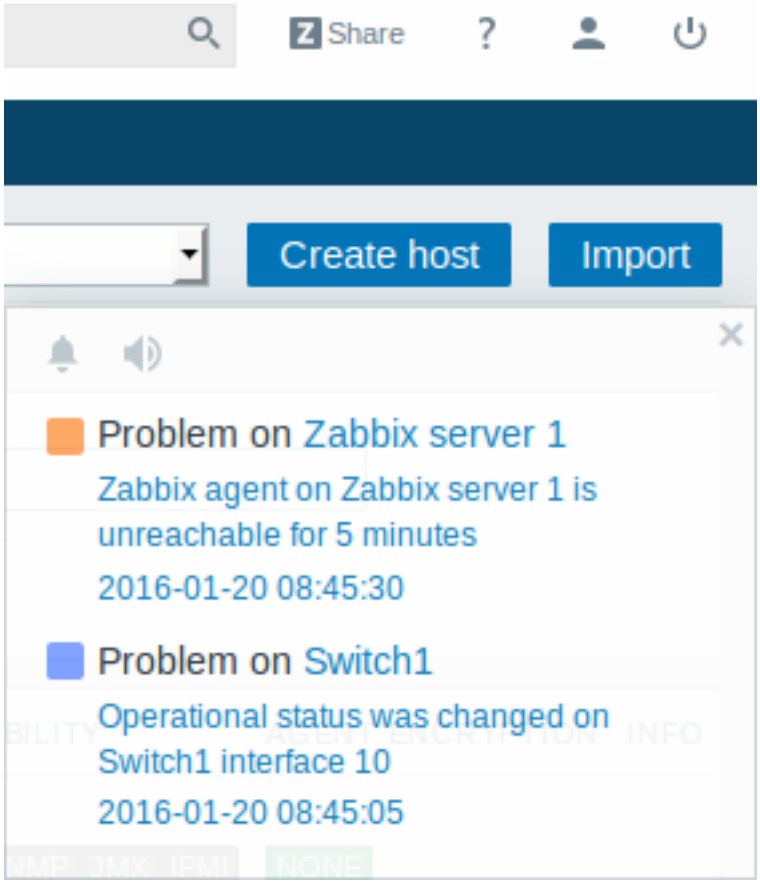


Parameter 参数 D	scription 描述
Frontend messaging	选中该复选框以启用全局通知。
Message timeout	您可以设置消息显示的时间。默认情况下，消息将在屏幕上显示 60 秒。 Time suffixes 支持的，如 30s, 5m, 2h, 1d。
Play sound	您可以设置声音的播放长度。 **Once** - 声音完整播放一次。 **10 seconds** - 声音重复播放 10 秒。 **Message timeout** - 当消息显示时，声音一直播放。
Trigger severity	您可以设置触发严重性，以激活全局通知和声音。您还可以选择适合各种严重程度的声音。 如果未标记严重性，则将不会显示任何消息。同样，将仅针对标记的那些严重性显示恢复消息。因此，如果您标记 Recovery 和 Disaster, 全局通知将会显示问题，以及灾难严重性触发器的恢复。

Parameter 参数 D	scription 描述
Show suppressed problems	标记该复选框以显示有关由于主机维护而将被抑制（未显示）的问题的通知。

全局信息显示

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。



在这个区域内，一些控件是可用的：:

- **Snooze** 键将会静音当前的警报音;
- **Mute/Unmute** 键在播放与不播放警报音之间切换。

2 浏览器中的声音

概述

要在 Zabbix 前端播放声音，必须在用户配置文件消息传递选项卡中启用前端消息传递，并选中所有触发严重性，并且还应该在全局通知弹出窗口中启用声音。

如果由于某些原因无法在设备上播放音频，则全局通知弹出窗口中的按钮 将永久保持 “静音” 状态，并显示消息 “无法支持该设备的通知音频”。将鼠标悬停在按钮 上时将显示。

仅支持 MP3 格式的声音，包括默认的音频片段。

Zabbix 前端的声音已经在 Linux 和 Chrome 上的最新 Firefox / Opera 浏览器，Windows 上的 Firefox，Microsoft Edge，Opera 和 Safari 浏览器中成功测试过。

<note important> 默认情况下，在最新的浏览器版本中可能会禁用自动播放声音。在这种情况下，您需要手动更改此设置。:::

4 全局搜索

可以在 Zabbix 前端中搜索主机，主机组和模板。

搜索输入框位于菜单中 Zabbix 徽标的下方。可以通过按 Enter 或单击搜索图标 来开始搜索。

如果主机的名称的任何部分都包含输入的字符串，则会出现一个下拉列表，列出所有此类主机（匹配的部分以橙色突出显示）。如果该主机的可见名称与作为搜索字符串输入的技术名称匹配，则该下拉列表还将列出该主机；匹配的主机将被列出，但不会突出显示。

属性搜索

可以通过以下属性搜索主机：

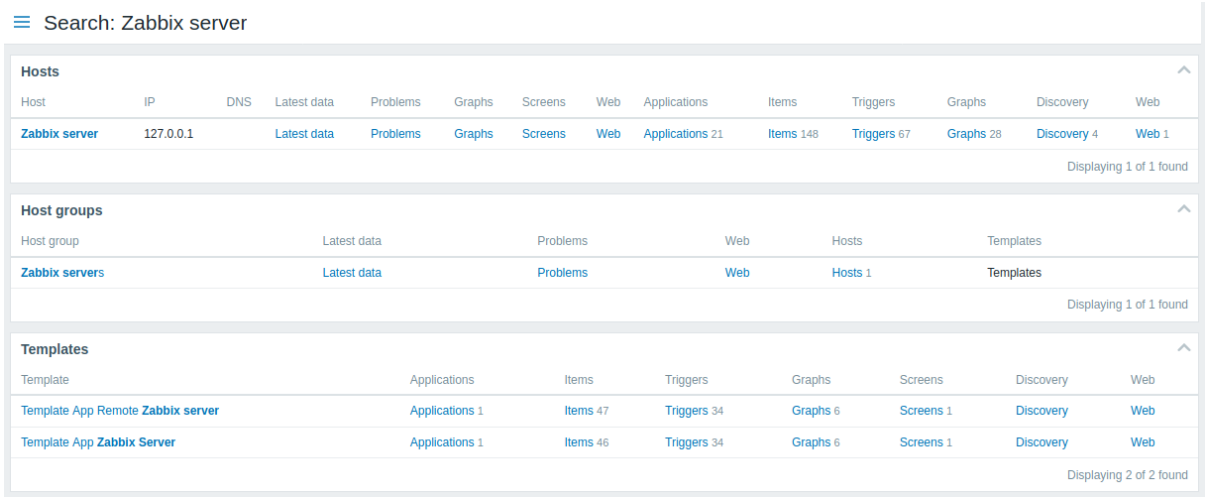
- 主机名
- 可见名
- IP 地址
- DNS 名

指定父主机组间接地选择所有嵌套的主机组

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，则搜索结果中的名称将显示在可见名称下方的括号中。

搜索结果

搜索结果包含三个单独的块，用于主机，主机组和模板。



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。

每个实体都提供指向监视和配置数据的链接。参见links available。

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。

已启用的主机以蓝色显示，已禁用的主机以红色显示。

可用链接

对于每个条目，以下链接可用：

- 主机
  - 监控
    - 最新数据
    - 触发器
    - 异常
    - 图
    - 主机聚合图形
    - Web 场景
  - 配置
    - 主机属性
    - 应用
    - 监控项
    - 触发器
    - 图
    - 发现规则
    - Web 场景
- 主机组
  - 监控
    - 最新数据
    - 监控项
    - 异常
    - 图
    - Web 场景
  - 配置
    - 主机组属性
    - 主机组成员（主机和模板）
- 模板
  - 配置
    - 模板属性
    - 应用
    - 监控项
    - 触发器
    - 图
    - 模板聚合图形
    - 发现规则
    - Web 场景

5 前端维护模式

概述

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。

当 Zabbix 前端处于维护模式时，可以停止 Zabbix 数据库并执行维护任务。

来自指定 IP 地址的用户将能够在维护模式期间正常工作。

配置

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.
define('ZBX_DENY_GUI_ACCESS', 1);

// Array of IP addresses, which are allowed to connect to frontend (optional).
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// Message shown on warning screen (optional).
```

```
$ZBX_GUI_ACCESS_MESSAGE = 'We are upgrading MySQL database till 15:00. Stay tuned...';
```

参数详	信息
ZBX_DENY_GUI_ACCESS	打开维护模式: 1 - 维护模式已打开，其他的数字表示未打开
ZBX_GUI_ACCESS_IP_RANGE	允许连接到前端的 IP 地址数组（可选） 例: `array('192.168.1.1', '192.168.1.2')`
ZBX_GUI_ACCESS_MESSAGE	您可以输入一条消息来通知用户有关维护的信息（可选）

Note:
Mostly the `maintenance.inc.php` file is located in `/conf` of Zabbix HTML document directory on the web server. However, the location of the directory may differ depending on the operating system and a web server it uses.
For example, the location for:

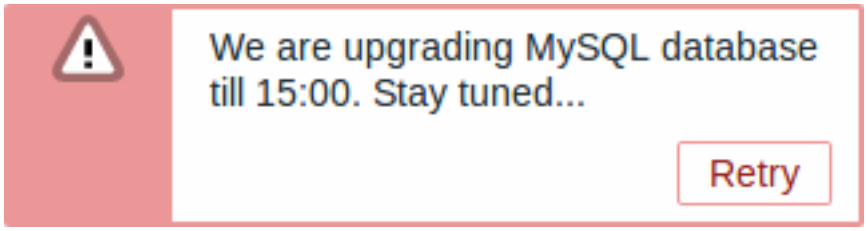
- SUSE and RedHat is `/etc/zabbix/web/maintenance.inc.php`.
- Debian-based systems is `/usr/share/zabbix/conf/`.

See also Copying PHP files.

Parameter	Details
ZBX_DENY_GUI_ACCESS	Enable maintenance mode: 1 – maintenance mode is enabled, disabled otherwise
ZBX_GUI_ACCESS_IP_RANGE	Array of IP addresses, which are allowed to connect to frontend (optional). For example: `array('192.168.1.1', '192.168.1.2')`
ZBX_GUI_ACCESS_MESSAGE	A message you can enter to inform users about the maintenance (optional).

显示

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。



在 ZBX_GUI_ACCESS_IP_RANGE 中定义的 IP 地址可以一直访问前端。

6 页面参数

概述

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 parameter=value 对来传递它们，通过问号 (?) 与 URL 分隔，并通过 & 符号（&）彼此分隔。

监控 → 问题

支持以下参数：

- **sort** - 排序列：时钟，主机，严重性，名称
- **sortorder** - 排序顺序或结果：DESC-降序，ASC-升序
- **filter_set** - 应该为'filter_set = 1' 以使用 " filter_ *" 参数
- **filter_rst** - 应该为'filter_rst = 1' 以重置过滤器元素
- **filter_show** - 过滤器选项 "显示"：1-最近的问题，2-全部，3-处于问题状态
- **filter_groupids** - 过滤器选项 "主机组"：主机组 ID 的数组
- **filter_hostids** - 过滤器选项 "主机"：主机 ID 的数组
- **filter_application** - 过滤器选项 "应用程序"：自由格式字符串
- **filter_triggerids** - 过滤器选项 "触发器"：触发器 ID 的数组
- **filter_name** - 过滤器选项 "问题"：自由格式字符串
- **filter_severity** - 过滤器选项 "最低严重性"：0-未分类，1-信息，2-警告，3-平均，4-高，5-灾难
- **filter_age_state** - 过滤器选项"Age less than": 应为 " filter_age_state = 1"，以启用 " filter_age"。仅在'filter_show' 等于 3 时使用
- **filter_age** - 过滤选项"Age less than": 天
- **filter_inventory** - 过滤器选项 "主机库存"：库存字段的数组：[字段]，[值]
- **filter_evaltype** - 过滤器选项 "标签"，标签过滤策略：0-和/或，2-或
- **filter_tags** - 过滤器选项 "标签"：已定义标签的数组：[标签]，[运算符]，[值]
- **filter_show_tags** - 过滤器选项 "显示标签"：0-无，1-1、2-2、3-3
- **filter_tag_name_format** - 过滤器选项 "标签名称"：0-全名，1-缩写，2-无
- **filter_tag_priority** - 过滤器选项 "标记显示优先级"：标记显示优先级的逗号分隔字符串
- **filter_show_suppressed** - 过滤器选项 "显示受抑制的问题"：应为 " filter_show_suppressed = 1" 以显示
- **filter_unacknowledged** - 过滤器选项 "仅显示未确认"：应为'filter_unacknowledged = 1' 才能显示
- **filter_compact_view** - 过滤器选项 "紧凑视图"：应该为'filter_compact_view = 1' 才能显示
- **filter_show_timeline** - 过滤器选项 "显示时间线"：应为 " filter_show_timeline = 1" 以显示
- **filter_details** - 过滤器选项 "显示详细信息"：应为 " filter_details = 1" 以显示
- **filter_highlight_row** - 过滤器选项 "突出显示整行"（使用问题颜色作为每个问题行的背景色）：应突出显示为 " 1"；仅在设置了 " filter_compact_view" 时可以设置
- **from** - 日期范围的开始，可以是 "相对"（例如：now-1m）
- **to** - 日期范围结束，可以是 "相对"（例如：now-1m）

监控 → 问题

可以使用 URL 参数激活受支持的前端页面中的全屏和 kiosk 模式。例如，在仪表板中：

- **/zabbix.php?action=dashboard.view&fullscreen=1** - 激活全屏模式
- **/zabbix.php?action=dashboard.view&kiosk=1** - 激活 kiosk 模式
- **/zabbix.php?action=dashboard.view&fullscreen=0** - 激活普通模式

7 定义

概述

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。

该文件是位于 Zabbix HTML 文档目录的/ include 中的 `define.inc.php`。

参数

此文件中用户可能感兴趣的参数:

- ZBX_LOGIN_ATTEMPTS

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

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

默认图行周期，以秒为单位。默认为一小时。

- ZBX_MIN_PERIOD

最小图形周期，以秒为单位。默认为一小时。

- ZBX_MAX_PERIOD

最大图形周期，以秒为单位。自 1.6.7 起默认为两年，然后为一年。

- ZBX_HISTORY_PERIOD

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）。如果设置为 0 秒表示无限期。当解析触发器名称中的 {ITEM.VALUE} 宏时，此常量值还会影响该值在过去搜索的距离。

- GRAPH_YAXIS_SIDE_DEFAULT

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: 0

- SCREEN_REFRESH_TIMEOUT (available since 2.0.4)

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: 30

- SCREEN_REFRESH_RESPONSIVENESS (available since 2.0.4)

在聚合图形中使用，并定义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，则将跳过所有有关更新的查询，直到收到响应为止。使用此参数，可以在 N 秒后发送另一个更新查询，而不必等待对第一个查询的响应。

Default: 10

- QUEUE_DETAIL_ITEM_COUNT

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: 500

- ZBX_SHOW_SQL_ERRORS (available since 3.4.0)

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: true

- VALIDATE_URI_SCHEMES (available since 3.4.5)

根据 ZBX_URI_VALID_SCHEMES 中定义的方案白名单验证 URI。

Default: true

- ZBX_URI_VALID_SCHEMES (available since 3.4.2)

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: http,https,ftp,file,mailto,tel,ssh

- ZBX_SHOW_TECHNICAL_ERRORS (available since 3.4.4)

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: false

- ZBX_SESSION_NAME (available since 4.0.0)

字符串用作 Zabbix 前端会话 cookie 的名称

Default: zbx_sessionid

8 自定义主题

概述

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。

步骤 1

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。

步骤 2

将您的主题添加到 APP::getThemes() 方法返回的主题列表中。您可以通过重写 APP 类中的 ZBase::getThemes() 方法来实现。这可以通过在 *include/classes/core/Z.php*: 中的大括号之前添加以下代码来完成：

```
public static function getThemes() {
    return array_merge(parent::getThemes(), array(
        'custom-theme' => _('Custom theme')
    ));
}
```

<note important> 请注意，您在第一对引号中指定的名称必须与主题文件的名称匹配，但不带扩展名。:::

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')
    ));
}
```

请注意，除最后一个主题外，每个主题都必须带有结尾逗号。

Note:
为了改变图形颜色，必须在 *graph_theme* 数据库表格中添加该条目。

步骤 3

激活新主题

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。

享受新的外观吧！

9 调试模式

概述

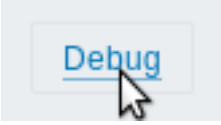
调试模式可用于诊断前端页面的性能问题。

配置

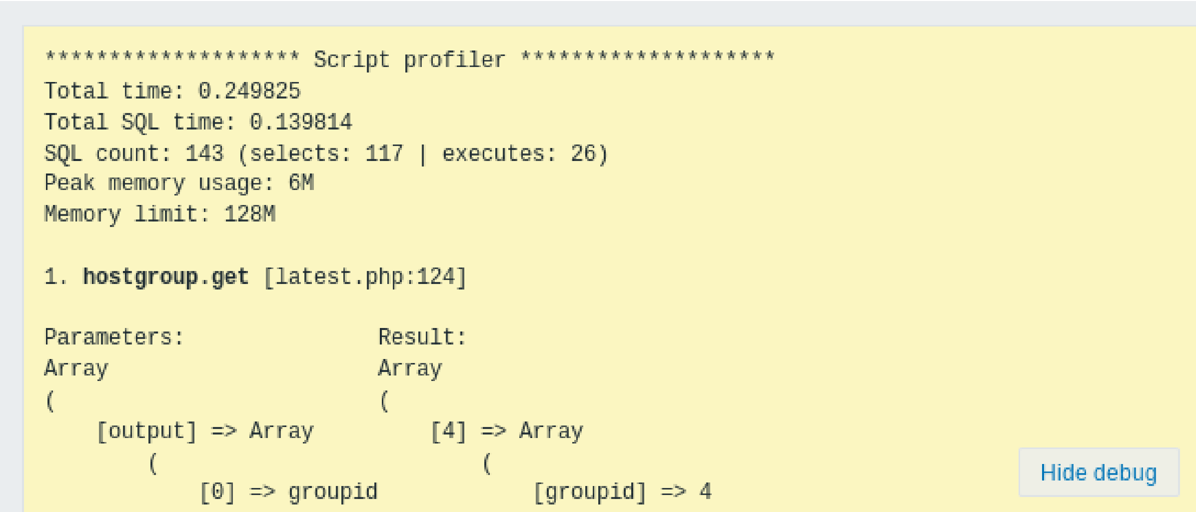
可为属于某个用户组的单个用户激活调试模式：

- 当配置user group时；
- 当查看配置user groups时；

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：



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：



```
******************** Script profiler ********************
Total time: 0.249825
Total SQL time: 0.139814
SQL count: 143 (selects: 117 | executes: 26)
Peak memory usage: 6M
Memory limit: 128M

1. hostgroup.get [latest.php:124]

Parameters:                Result:
Array                      Array
(                          (
    [output] => Array          [4] => Array
        (                          (
            [0] => groupid             [groupid] => 4
```

Hide debug

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:
启用 *Debug mode* 会对前端造成一定的性能影响。

10 Zabbix 使用的 Cookies

概览

本页提供了一张 Zabbix 使用的 Cookies 的列表

Cookie 名	述 Co	kie 值	过期/	大存活	HttpOnly[	1]
PHPSESSID	PHP会话的唯一 ID。 示例：kv 这个值的长度可以在 *php.ini - session.sid_length* 中配置。	p5pu2ru1a2ccvff0g92un87a 会话（浏览会话结	0 (终止)	+	（仅 W	b 服务器启用 HTTPS）

Cookie 名	述 Co	kie 值过期/	大存活 HttpOnly[	1]
ZBX_SESSION_NAME (自 4.0.0 可用)。	会话唯一 ID - 一个 32 个字符长度的字符串示例：004bc0213e7	8bca87fcc3919eea5270 当前时间日期起 1 个月（31 天）	+ (仅 Web 服务器启用 H	TPS）
该字符串用作 Zabbix 前端会话 cookie 的名称。				
默认值：zbx_sessionid				
tab	活动标签页数；此 Cookie 仅在具有多个标签页的页面上使用（例如// 主机 Host ，触发器 Trigger 或动作 Action //配置页面），并被创建在用户从主标签页导航到另一个标签页时（例如，// 标签 Tags 或依赖关系 Dependencies //标签）。 示例：1 会话（浏览会话结束时终止） 0 被用于主标签页	- -		
browserwarning_ignore	是否应忽略有关使用过时的浏览器的警告。yes	会话（浏览会话结束时终止）	- -	
message_ok	页面重新加载时的消息。纯文本消息	会话（浏览会话结束时）或在页面	重新加载时尽快过期。+ -	
message_error	页面重新加载时的错误消息。纯文本消息	会话（浏览会话结束时）或在页面备重	加载时尽快过期。+ -	

> **Note:**
>
> 在 Zabbix cookies 中，不支持 on Zabbix cookies 通过 WEB 服务器指定强制指定 `HttpOnly` 标记。

19. API

概览 Zabbix API 允许你以编程方式检索和修改 Zabbix 的配置，并提供对历史数据的访问。它广泛用于：

- 创建新的应用程序以使用 Zabbix；
- 将 Zabbix 与第三方软件集成；
- 自动执行常规任务。

Zabbix API 是基于 Web 的 API，作为 Web 前端的一部分提供。它使用 JSON-RPC 2.0 协议，这意味着两点：

- 该 API 包含一组独立的方法；

- 客户端和 API 之间的请求和响应使用 JSON 格式进行编码。

有关协议和 JSON 的更多信息可以在 JSON-RPC 2.0 规范 和 JSON 格式主页中找到。

结构　Zabbix API 由许多名义上分组的独立 API 方法组成。每个方法执行一个特定任务。例如，方法 `host.create` 隶属于 *host* 这个 API 分组，用于创建新主机。历史上，API 分组有时被称为 “类”。

> **Note:**
> 大多数 API 至少包含四种方法：`get`，`create`，`update` 和 `delete` ，分别是检索，创建，更新和删除数据，但是某些 API 提供一套完全不同的方法。

执行请求　当完成了前端的安装配置后，你就可以使用远程 HTTP 请求来调用 API。为此，需要向位于前端目录中的 `api_jsonrpc.php` 文件发送 HTTP POST 请求。例如，如果你的 Zabbix 前端安装在 *http://company.com/zabbix*，那么用 HTTP 请求来调用 `apiinfo.version` 方法就如下面这样：

```
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":{}}
```

请求的 `Content-Type` 头部必须设置为以下值之一：`application/json-rpc`, `application/json` 或 `application/jsonrequest`。

<note tip> 你可以使用任何 HTTP 客户端或 JSON-RPC 测试工具手动执行 API 请求，但对于开发应用程序，我们建议使用 社区维护的程序库。:::

示例　以下部分将详细介绍一些使用示例。

验证　在访问 Zabbix 中的任何数据之前，你需要登录并获取身份验证令牌。这可以使用该user.login 方法完成。让我们假设你想要以标准 Zabbix Admin 用户身份登录。然后，你的 JSON 请求将如下所示：

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1,
    "auth": null
}
```

让我们仔细看看示例请求对象。它具有以下属性：

- `jsonrpc` - API 使用的 JSON-RPC 协议的版本; Zabbix API 实现的 JSON-RPC 版本是 2.0;
- `method` - 被调用的 API 方法名;
- `params` - 将被传递给 API 方法的参数;
- `id` - 请求的任意标识符;
- `auth` -用户认证令牌; 如果没有的话可以设置为 null。

如果你正确提供了凭据，API 返回的响应将包含用户身份验证令牌：

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

响应对象又包含以下属性：

- `jsonrpc` - JSON-RPC 协议的版本;
- `result` - 请求返回的数据;
- `id` - 相应请求的 id。

检索主机　我们现在有一个有效的用户身份验证令牌，可以用来访问 Zabbix 中的数据。例如，让我们使用host.get 方法检索所有已配置主机的 ID，主机名和接口：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2,
    "auth": "0424bd59b807674191e7d77572075f33"
}
```

Attention:
请注意，auth 属性现在设置为我们通过调用 user.login 方法获得的身份验证令牌。

响应对象将包含有关主机的请求的数据：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
    "id": 2
}
```

<note tip> 出于性能原因，我们建议始终列出要检索的对象属性，并避免检索所有内容。:::

创建新监控项 让我们使用从上一个请求 host.get 中获得的数据，在主机 “Zabbix server” 上创建一个新监控项 。这个可以通过使用方法item.create：

```
{
    "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
}
```

成功的响应将包含新创建监控项的 ID，可用于在以后请求中引用监控项：

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 3
}
```

Note:
item.create 方法和其他的创建（create）方法，也可以接受对象数组，并通过一次 API 调用中创建多个监控项。

创建多个触发器　因此，如果 create 方法接受数组，我们可以添加多个触发器，像这样 ^-^：

```
{
    "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
}
```

操作成功的响应将包含新创建的触发器的 ID：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 4
}
```

更新监控项　启用监控项，即将其状态设置为 “0”：

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 5
}
```

操作成功的响应将包含被更新的触发器的 ID：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
```

```
    },
    "id": 5
}
```

Note:
item.update 方法以及其他更新方法也可以接受对象数组，并通过一次 API 调用更新多个监控项。

更新多个触发器　启用多个触发器，即将其状态设置为 0：

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": [
        {
            "triggerid": "13938",
            "status": 0
        },
        {
            "triggerid": "13939",
            "status": 0
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 6
}
```

成功的响应将包含被更新的触发器的 ID 数组：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938",
            "13939"
        ]
    },
    "id": 6
}
```

Note:
这是更新的首选方法。一些 API 方法 host.massupdate 允许编写更简单的代码，但不建议使用这些方法，因为它们将在未来的版本中删除。

错误处理　到目前为止，我们试过的一切工作正常。但是，如果我们尝试对 API 调用不正确会发生什么？让我们尝试通过调用host.create创建另一个主机，但省略一个必填 groups 参数。

```
{
    "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"
}
```

这个请求的返回会包含一个错误信息：

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "No groups for host \"Linux server\"."
    },
    "id": 7
}
```

如果发生错误，响应对象将包含 `error` 而不是 `result` 属性，同时 error 将具有以下数据的属性：

- `code` - 错误代码;
- `message` - 一个简短的错误摘要;
- `data` - 更详细的错误消息。

错误可能发生在不同的情况下，例如，使用不正确的输入值，会话超时或试图访问不存在的对象。你的应用程序应该能够优雅地处理这些类型的错误。

API 版本　为了简化 API 版本控制，自 Zabbix 2.0.4 开始，API 的版本与 Zabbix 本身的版本相匹配。你可以使用apiinfo.version 方法查找你正在使用的 API 的版本。这对于调整应用程序以使用特定于版本的功能非常有用。

我们保证在主要版本内部具有向后兼容性。当在主要版本之间进行向后不兼容的更改时，我们通常将旧功能在下一个版本中保留为已弃用，并且仅在此后的版本中将其删除。有时，我们可能会删除主要版本之间的功能，而不提供任何向后兼容性。重要的是，你不要依赖任何弃用的功能，并尽快迁移到较新的替代品。

> **Note:**
> 你可以在API 更改日志中跟踪对 API 所做的所有更改。

进一步阅读　你现在知道了足够开始使用 Zabbix API，但不要停在这里。为了进一步阅读，我们建议你查看可用的 API 列表。

方法参考

本节提供了 Zabbix API 提供的功能的概述，并将帮助你找到可用的类和方法。

监控　Zabbix API 允许你访问在监控时采集历史记录和其他数据。

历史记录

通过检索 Zabbix 监控进程采集的历史记录，用于展示或进一步的处理数据。

History API

趋势

检索由 Zabbix server 可计算的趋势值，来进行展示或进一步处理数据。

Trend API

事件

检索触发器，网络发现和其它 Zabbix 系统生成的事件，以实现更灵活的场景应用或第三方工具集成。

Event API

问题

根据给定的参数检索问题。

Problem API

服务监控

检索有关任何服务的详细服务层可用性信息。

Service SLA calculation

任务

任务管理器允许检查监控项或低级发现规则，且无需重新加载配置。

Task API

配置　Zabbix API 允许您管理监控系统的配置。

主机和主机组

管理主机组，主机及其相关的一切，包括主机接口，主机宏和维护期。

Host API | Host group API | Host interface API | User macro API | Maintenance API

监控项和应用集

定义要监控的监控项。创建或删除应用程序并为其分配监控项。

Item API | Application API

触发器

配置触发器以通知您系统中的问题。管理触发器依赖关系。

Trigger API

图形

编辑图形或单独的图形项，以便更好地展示采集的数据。

Graph API | Graph item API

模板

管理模板并将其链接到主机或其他模板。

Template API

导入和导出

导出和导入 Zabbix 配置数据来进行配置备份，迁移或大规模配置更新。

Configuration API

低级别发现

配置低级发现规则以及项目，触发器和图形原型来监视动态实体。

LLD rule API | Item prototype API | Trigger prototype API | Graph prototype API | Host prototype API

事件关联性

创建自定义事件相关规则。

Correlation API

动作和警报

定义动作和报警，以通知用户某些事件或自动执行远程命令。获取有关生成的警报及其接收者的信息。

Action API | Alert API

服务

管理服务以进行服务级别监视，并检索有关任何服务的详细 SLA 信息。

Service API

仪表板

管理仪表板。

Dashboard API

聚合图形

用于分开编辑全局和模板级聚合图形或单个聚合图形项。

Screen API | Screen item API | Template screen API | Template screen item API

拓扑图

配置拓扑图用于创建 IT 基础架构的详细动态展现。

Map API

Web 检测

用于配置 Web 场景来监控 Web 应用程序和服务。

Web scenario API

网络发现

管理网络级发现规则以自动查找和监控新主机。获得对所发现的服务和主机的信息的完全访问。

Discovery rule API | Discovery check API | Discovery host API | Discovery service API

管理　使用 Zabbix API，您可以更改监控系统的管理设置。

用户

添加有权访问 Zabbix 的用户，将其分配给用户组并授予权限。配置媒体类型和用户接收警报的方式。

User API | User group API | Media type API

通用

用于更改某些全局配置选项。

Icon map API | Image API | User macro API

代理

用于管理分布式监视设置中使用的代理。

Proxy API

脚本

配置和执行脚本以帮助您完成日常任务

Script API

API 信息　检索 Zabbix API 的版本，以便应用程序可以使用特定于版本的功能。

API info API

1. 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

> 动作对象

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。
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。

属性类	描述	
eventsource (required)	integer	*(constant)* 动作将处理的事件的类型。 参见事件"源"属性以获取支持的事件类型列表。
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。

属性类	描述	
ack_longdata	string	确认操作消息文本。
ack_shortdata	string	确认操作消息主题。
status	integer	动作是启动还是禁用。 取值：0 - (默认)启用；1 - 禁用。

属性类	描述	
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. 请注意，升级仅支

属性类	描述	
esc_step_from	integer	启示步骤。默认: 1. 请注意，升级仅支持触发器和内部操作。在触发器操作中，问题恢复和更新操作不支持升级。

属性类	描述	
esc_step_to	integer	结束步骤。默认: 1. 请注意，升级仅支持触发器和内部操作。在触发器操作中，问题恢复和更新操作不支持升级。

属性类	描述	
evaltype	integer	运行状态计算方法。 可能的值: 0 - (默认) AND / OR; 1 - AND; 2 - OR.
opcommand	object	包含操作所运行的命令的数据。 操作命令对象是在下面有详细描述. 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)* 操作ID; `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。 操作消息对象是在下面有详细描述. 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”绑定模板”和”解绑模板”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。

动作恢复操作

动作恢复操作对象定义将在解决问题时执行的操作。可以对触发操作和内部操作执行恢复操作。它具有以下属性。

属性类	描述	
operationid	string	*(只读)* 动作操作的ID。

属性类	描述	
operationtype (必要)	integer	操作的类型触发动作的可能值: 0 - 发送信息; 1 - 远程命令; 11 - 通知所有参与者。 内部操作的可能值: 0 - 发送信息; 11 - 通知所有参与者。

属性类	描述	
actionid	string	恢复操作所属的动作的ID。
opcommand	object	对象，该对象包含有关恢复操作运行的命令的数据。 操作命令对象是described in detail above. 必要用于远程命令操作。

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,* 只读*)* 对象的ID; operationid - *(string)* 操作的ID; groupid - *(string)* 主机组的ID。 必要如果未设置“opcommand_hst”，则用于远程命令操作。

属性类	描述	
opcommand_hst	array	主机运行远程命令。 每个对象具有以下属性: opcommand_hst - *(string,* 只读*)* 对象的ID; operationid - *(string)* 操作的ID; hostid - *(string)* 主机的ID；如果设置为0，则命令将在当前主机上运行。 必要如果

属性类	描述	
opmessage	object	对象，该对象包含有关恢复操作发送的消息的数据。 操作消息对象是described in de-tail above. 必要用于消息操作。

属性类	描述	
opmessage_grp	array	发送消息的用户组。 每个对象具有以下属性: `operationid` - *(string)* 操作的ID; `usrgrpid` - *(string)* 用户组的ID。 必要如果未设置“opmessage_usr”，则用于消息操作。

属性类	描述	
opmessage_usr	array	发送消息的用户。 每个对象具有以下属性: `operationid` - *(string)* 操作的ID; `userid` - *(string)* 用户ID. 必要如果未设置“opmessage_grp”，则用于消息操作。

动作更新操作

动作更新操作对象定义将更新问题时执行操作如（评论、确认、改变严重等级、手动关闭），可以对触发动作进行更新操作，它具有以下性质。

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
operationid	string	*(readonly)* (只读) 动作的 ID。
operationtype (required)	integer	操作类型。 触发动作可选的值: 0 - 发送信息; 1 - 远程命令; 12 - 通知相关人.

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
opcommand	object	该对象包含关于更新操作运行的命令的数据。 远程命令的请求操作命令查看动作详细操作命令。

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
opcommand_grp	array	包含运行远程命令的主机组对象的数组。 数组中的每个对象具有以下属性： groupid - *(string)* 主机组的ID. \\如果opcommand_hs没有设置，执行远程命令操作的请求。

属性 [	型](/manual/api/reference_commentary#data_types) 描述
opcommand_hst	array

包含运行远程命令的主机对象的数组。

数组中的每个对象具有以下属性：
hostid
-
(string)
主机的ID;
如果设置为0，命令在本地主机运行。

如果opcommand_g
没有设置，执行远程

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
opmessage	object	该对象包含关于更新操作发送的消息的数据。 操作信息对象的详细描述。

属性 [	型](/manual/api/reference_commentary#data_types) 描述
opmessage_grp	array

发送消息的用户组.

数组中的每个对象具有以下属性：
usrgrpid - *(string)* 用户ID。

如果没有设置 opmessage_u
只 send message 操作请求。
忽略 send update message 操作.

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
opmessage_usr	array	发送消息的用户. 数组中的每个对象具有以下属性： `userid` - *(string)* 用户 ID。 如果没有设置 `opmessage_us` 只 `send message` 操作请求。忽略 `send update message` 操作.

动作过滤

action filter 对象定义执行配置的操作必须满足的一组条件。它具有以下属性。

属性类	描述	
conditions (必要)	array	用于筛选结果的筛选条件集过滤条件评估方法。 可能值: 0 - and/or; 1 - and; 2 - or; 3 - 自定义表达式。
evaltype (必要)	integer	

属性类	描述	
eval_formula	string	*(只读)* 用户定义的表达式，用于评估具有自定义表达式的过滤器的条件。表达式必须包含通过其 `formulaid` 引用特定过滤条件的ID。表达式中使用的ID必须与过滤条件

属性类	描述	
formula	string	用户定义的表达式，用于使用自定义表达式计算过滤器的条件。表达式必须包含通过其“formulaid”引用特定筛选条件的id。表达式中使用的id必须与过滤器条件

属性类	描述

动作过滤条件

动作过滤条件对象定义在运行操作动作之前必须检查的特定条件。

属性类	描述	
conditionid	string	*(只读)* 动作条件的ID。

属性类	描述	
conditiontype (必要)	integer	条件类型。 触发操作的可能值: 0 - 主机组; 1 - 主机; 2 - 触发器; 3 - 触发器名称; 4 - 触发严重程度; 6 - 时间段; 13 - 主机模板; 15 - 应用; 16 - 维护状态; 25 - 事件标签; 26 - 事件标记值.

属性类	描述	
value (必要)	string	要与之比较的值。
value2	string	要与之比较的辅助值。 条件类型为 *26* 时触发操作所必需的。
actionid	string	*(只读)* 条件所属操作的 ID.

属性类	描述	
formulaid	string	用于自定义表达式引用条件的任意唯一ID。只能包含大写字母。用户在修改过滤条件时必须定义ID，但在以后请求时会重新生成。

属性类	描述	
operator	integer	条件运算符。和可能的值: 0 - *(default)* =; 1 - <>; 2 - like; 3 - not like; 4 - in; 5 - >=; 6 - <=; 7 - not in.

Note:
为了更好地理解如何使用具有各种类型表达式的过滤器，看看例子检索动作和创建动作方法。

每种条件类型都支持以下运算符和值。

条件条	名称支持操作	期望值	
0	Host group	=, <>	主机组ID。
1	Host	=, <>	主机ID。
2	Trigger	=, <>	触发器ID。
3	Trigger name	like, not like	触发器名称。

条件条	名称支持操作	期望值	
4	Trigger severity	=, <>, >=, <=	触发严重性。参考触发器级别属性获取支持的触发器级别列表
5	Trigger value	=	触发值。参考触发器值属性获取支持的触发器值列表.
6	Time period	in, not in	事件被触发的时间，参考time pe-riod.

条件条	名称支持操作	期望值	
7	Host IP	=, <>	要用逗号分隔的一个或多个IP范围。参考network discovery configuration参阅有关支持的IP范围格式的更多信息

条件条	名称支持操作	期望值	
8	Discovered service type	=, <>	发现服务的类型。服务类型与用于检测服务的发现检查的类型相匹配。参考discovery check ”type” property获取支持的类型列表。
9	Discovered service port	=, <>	一个或多个端口范围以逗号分隔。

条件条	名称支持操作	期望值	
10	Discovery status	=	发现对象的状态。 可能的值: 0 - 主机或服务启动; 1 - 主机或服务关闭; 2 - 发现主机或服务; 3 - 主机或服务失去连接。

条件条	名称支持操作	期望值	
11	Uptime or downtime duration	>=, <=	指示发现的对象处于当前状态的时间(秒)。
12	Received values	=, <>, >=, <=, like, not like	执行 zabbix 代理、snmpv1、snmpv2 或 snmpv3 发现检查时返回的值。
13	Host template	=, <>	链接模板 ID。
15	Application	=, like, not like	应用程序的名称。

条件条	名称支持操作	期望值	
16	Maintenance status	in, not in	No value 必要: 使用 “in” 运算符意味着主机必须处于维护状态，“not in”- 不处于维护状态。
18	Discovery rule	=, <>	发现规则的 ID。
19	Discovery check	=, <>	发现检查的 ID。
20	Proxy	=, <>	代理的 ID。

条件条	名称支持操作	期望值	
21	Discovery object	=	触发发现事件的对象类型。 可能值: 1 - 发现主机; 2 - 发现服务。
22	Host name	like, not like	主机名。

条件条	名称支持操作	期望值	
23	Event type	=	特定内部事件。 可能值: 0 - 监控项处于“不支持”状态; 1 - 监控项处于“正常”状态; 2 - LLD规则处于“不支持”状态; 3 - LLD规则处于“正常”状态; 4 - 触发器处于“未知”

条件条	名称支持操作	期望值	
24	Host metadata	like, not like	自动注册主机的元数据。
25	Tag	=, <>, like, not like	事件标记。
26	Tag value	=, <>, like, not like	事件标记值。

创建动作

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警告并且主机等于 10084 或 10106 。公式 ID A 和（B 或 C)。

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
}
```

创建 AGNENT 自动注册规则

当主机名中包含”SRV” 或元数据中包含”CentOS” 时，向”Linux servers” 主机组中添加主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Register Linux servers",
        "eventsource": "2",
        "status": "0",
        "filter": {
            "evaltype": "2",
            "formula": "A or B",
            "conditions": [
                {
                    "conditiontype": "22",
                    "operator": "2",
                    "value": "SRV",
                    "value2": "",
                    "formulaid": "B"
                },
                {
                    "conditiontype": "24",
                    "operator": "2",
                    "value": "CentOS",
                    "value2": "",
                    "formulaid": "A"
                }
            ]
```

```
        },
        "operations": [
            {
                "actionid": "9",
                "operationtype": "4",
                "esc_period": "0",
                "esc_step_from": "1",
                "esc_step_to": "1",
                "evaltype": "0",
                "opgroup": [
                    {
                        "operationid": "16",
                        "groupid": "2"
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            19
        ]
    },
    "id": 1
}
```

参见

- 动作过滤
- 动作操作

来源

CAction::create() in *frontends/php/include/classes/api/services/CAction.php*.

删除动作

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。

示例

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*.

更新动作

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	动作更新操作，以替换现有的更新操作。

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*.

查询动作

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	返回过滤属性中的动作筛选器。
selectOperations	query	返回动作属性中操作属性。
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	
limit	integer	

参数类	说明
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

检索触发动作

检索所有配置的触发器操作以及操作条件和操作。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectRecoveryOperations": "extend",
        "selectAcknowledgeOperations": "extend",
        "selectFilter": "extend",
        "filter": {
            "eventsource": 0
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

返回:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "3",
            "name": "Report problems to Zabbix administrators",
            "eventsource": "0",
            "status": "1",
            "esc_period": "1h",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [],
                "eval_formula": ""
            },
            "acknowledgeOperations": [
                {
                    "operationid": "31",
                    "operationtype": "12",
                    "evaltype": "0",
                    "opmessage": {
                        "default_msg": "1",
```

```
                    "subject": "",
                    "message": "",
                    "mediatypeid": "0"
                }
            },
            {
                "operationid": "32",
                "operationtype": "0",
                "evaltype": "0",
                "opmessage": {
                    "default_msg": "0",
                    "subject": "Updated: {TRIGGER.NAME}",
                    "message": "{USER.FULLNAME} updated problem at {EVENT.UPDATE.DATE} {EVENT.UPDATE.T
                    "mediatypeid": "1"
                },
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage_usr": []
            },
            {
                "operationid": "33",
                "operationtype": "1",
                "evaltype": "0",
                "opcommand": {
                    "type": "0",
                    "scriptid": "0",
                    "execute_on": "0",
                    "port": "",
                    "authtype": "0",
                    "username": "",
                    "password": "",
                    "publickey": "",
                    "privatekey": "",
                    "command": "notify.sh"
                },
                "opcommand_hst": [
                    {
                        "hostid": "0"
                    }
                ],
                "opcommand_grp": []
            }
        ],
        "operations": [
            {
                "operationid": "3",
                "actionid": "3",
                "operationtype": "0",
                "esc_period": "0",
                "esc_step_from": "1",
                "esc_step_to": "1",
                "evaltype": "0",
                "opconditions": [],
                "opmessage": [
                    {
                        "default_msg": "1",
                        "subject": "",
                        "message": "",
                        "mediatypeid" => "0"
```

```
                        }
                    ],
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ]
                }
            ],
            "recoveryOperations": [
                {
                    "operationid": "7",
                    "actionid": "3",
                    "operationtype": "11",
                    "evaltype": "0",
                    "opconditions": [],
                    "opmessage": {
                        "default_msg": "0",
                        "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
                        "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger
                        "mediatypeid": "0"
                    }
                }
            ]
        }
    ],
    "id": 1
}
```

检索发现动作

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*.

2. 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

这类要与告警一起使用.

对象引用:

- Alert

可用的方法:

- alert.get - 搜索告警

告警对象

以下是 `alert` API 的使用方法

告警

> **Note:**
> 告警是由 Zabbix server 创建，无法通过 API 修改。

告警对象包含有关某些动作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	告警ID。

特性类	描述	
actionid	string	告警生成的动作ID.
alerttype	integer	告警类型。 可能的值： 0 - 信息; 1 - 远程命令。
clock	timestamp	告警生成的时间。
error	string	告警发送信息或者执行一个命令产生的报错信息。
esc_step	integer	告警动作步骤。

特性类	描述	
eventid	string	触发动作的事件ID。
mediatypeid	string	用于发送消息的报警媒介类型的ID。
message	text	消息文本。用于消息告警。
retries	integer	Zabbix尝试发送消息的次数。

特性类	描述	
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	显示告警动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。

获取告警

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。
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。

参数类	描述	
hostids	string/array	只返回来自指定主机的对象生成的alerts。
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。

参数类	描述	
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。 参考事件对象属性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。 参考事件来源属性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	在 `mediatype` 属性中以数组形式返回消息警报的媒体类型.

参数类	描述	
selectUsers	query	以 `user` 属性中的数组形式返回邮件的收件人。
sortfield	string/array	按提交参数对结果排序。 可提交的参数: `alertid`, `clock`, `eventid` and `status`.
countOutput	boolean	参考注释中描述了所有“get”方法的公共参数。
editable	boolean	
excludeSearch	boolean	

参数类	描述
filter	object
limit	integer
output	query
preservekeys	boolean
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	boolean

返回值

(integer/array) 返回如下:

- 数组对象;
- 如果使用了"countOutput" 参数，则返回对检索对象的计数值。

范例

通过动作 ID 检索警报

返回动作 id 为 "3" 的所有告警。

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

- 主机

- 媒体类型
- 用户

来源

CAlert::get() in *frontends/php/include/classes/api/services/CAlert.php*.

3.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

API 版本信息

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*.

4. 应用集

这个类用于管理应用集。

对象引用:

- 应用集

相关方法：

- 创建应用集
- 删除应用集
- 检索应用集
- 添加监控项到应用集
- 更新应用集

创建应用集

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*.

删除应用集

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*.

应用集对象

以下是 应用集 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。

属性类	说明	
hostid (required)	string	应用集所属主机的ID。 不能进行更新。
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。

应用集添加监控项

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

- 监控项

来源

CApplication::massAdd() in *frontends/php/include/classes/api/services/CApplication.php*.

更新应用集

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*.

检索应用集

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	

Parameter	Type	Description
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
}
```

参考

- 主机
- 监控项

来源

CApplication::get() in *frontends/php/include/classes/api/services/CApplication.php*.

5. 审计日志

该方法用于审计日志。

对象引用:

- 审计日志对象

可用方法:

- auditlog.get - 检索审计日志记录

> 1. 审计日志对象

以下对象与审计日志直接相关。

审计日志

审计日志对象包含有关用户操作的信息。它具有以下属性。

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
auditid	string	*(只读 ID)* 审计日志的ID。
userid	string	审计日志项的用户标识ID。

属性 [	型](/manual/api/reference_commentary#data_types) 描述
action	integer 审计日志项的动作。 取值: 0 - 增加; 1 - 更新; 2 - 删除; 3 - 登录; 4 - 登出; 5 - 启用; 6 - 禁用.
clock	timestamp 审计日志项创建时间戳。

属性 [	型](/manual/api/reference_commentary#data_types) 描述
resourcetype	integer

审计日志项资源类型。

取值: 0 - 用户; 2 - 配置 Zab-bix; 3 - 媒介类型; 4 - 主机; 5 - 动作; 6 - 图表; 7 - 图表元素; 11 - 用户组; 12 - 应用; 13 - 触发器; 14 - 主机组;

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
note	string	审计日志项简单描述。
ip	string	审计日志项使用者的IP。
resourceid	string	审计日志项资源标识符ID。
resourcename	string	审计日志项资源人可读名称。

审计日志详细信息

属性类	描述	
table_name	string	数据库表名称。
field_name	string	数据库表字段名称。
oldvalue	string	数据库表字段旧值。
newvalue	string	数据库表字段新值。

2. 检索审计日志

描述

```
integer/array auditlog.get(object parameters)
```

该方法允许根据给定的参数检索审计日志记录。

参数

(object) 定义所需输出的参数。

该方法接受以下参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
auditids	string/array	仅返回具有给定ID的审计日志。
userids	string/array	仅返回由给定用户创建的审计日志。
time_from	timestamp	仅返回在给定时间之后或在给定时间创建的审计日志项。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
time_till	timestamp	仅返回在给定时间之前或指定时间创建的审计日志项。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
selectDetails	query	返回每个字段更改为的审计日志项如细节属性. 仅适用于带动作项"*1*-更新",对于其他类型的操作，返回空数组。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
sortfield	string/array	按给定属性对结果进行排序。 取值: `auditid`, `userid`, `clock`.

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个数组，其中的键是属性名称，并且值可以是单个值或要匹配的值数组。 另外支持按属性字段过滤: `table_name`, `field_name`.

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
search	object	字段内容中的子字符串搜索: note, ip, resourcenam oldvalue, newvalue 不区分大小写。
countOutput	boolean	该参数对于参考注释中描述的所有 get 方法都是通用的。
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

（integer/array）传回:

- 对象数组;
- 如果使用 countOutput 参数，则为检索到的对象的计数。

示例

检索审计日志

检索两个最新的审计日志记录。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": "extend",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 2
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "auditid": "189",
            "userid": "1",
            "clock": "1580913141",
            "action": "3",
            "resourcetype": "0",
            "note": "",
            "ip": "127.0.0.1",
            "resourceid": "0",
            "resourcename": ""
        },
        {
            "auditid": "188",
            "userid": "1",
            "clock": "1580903029",
            "action": "3",
            "resourcetype": "0",
            "note": "",
            "ip": "127.0.0.1",
            "resourceid": "0",
            "resourcename": ""
        }
    ],
    "id": 2
}
```

检索在 oldvalue 字段中具有子字符串”test” 的审计日志记录。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": ["auditid", "resourcename"],
        "search": {
            "newvalue": "test"
        },
        "selectDetails": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```


响应:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "auditid": "5",
            "resourcename": "Mattermost2",
            "details": [
                {
                    "table_name": "media_type",
                    "field_name": "event_menu_url",
                    "oldvalue": "http://test",
                    "newvalue": "http://test{EVENT.TAGS.__test}"
                }
            ]
        },
        {
            "auditid": "7",
            "resourcename": "Email",
            "details": [
                {
                    "table_name": "media_type",
                    "field_name": "name",
                    "oldvalue": "Email",
                    "newvalue": "Email test"
                }
            ]
        }
    ],
    "id": 20
}
```


参见

- 审计日志对象

来源

CAuditLog::get() in *ui/include/classes/api/services/CAuditLog.php*.

6. 自动注册

该方法用于自动注册。

对象引用:

- 自动注册

可用方法:

- autoregistration.get - 检索自动注册
- autoregistration.update - 更新自动注册

> 1. 自动注册对象

以下对象与自动注册 API 直接相关。

自动注册

自动注册对象具有以下属性。

属性 [	型](/manual/api/reference_commentary#data_types) 描述
tls_accept	integer

自动注册允许的传入连接的类型。

取值:
1 - 允许不安全的连接;
2 - 允许用 PSK 连接 TLS.
3 - 允许通过 PSK 连接不安全和 TLS。
.

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
tls_psk_identity	string	*(只写)* PSK 认证字符串. 不要将敏感信息放在 PSK 身份中，它会通过网络以未加密的方式传输，以通知接收者要使用哪个 PSK。

属性 [	型](/manual/api/reference_commentary#data_types) 描述	
tls_psk	string	(只写) PSK 值字符串(偶数个十六进制字符)。

2. 更新

说明

`object autoregistration.update(object autoregistration)`

该方法用于更新已存在的自动注册。

PARAMETERS 参数

(object) 自动注册属性会被更新。

RETURN VALUES 返回值

(boolean) 成功更新后返回布尔值 true。

例子

请求:

```
{
    "jsonrpc": "2.0",
    "method": "autoregistration.update",
    "params": {
        "tls_accept": "3",
        "tls_psk_identity": "PSK 001",
        "tls_psk": "11111595725ac58dd977beef14b97461a7c1045b9a1c923453302c5473193478"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

来源

CAutoregistration::update() in *ui/include/classes/api/services/CAutoregistration.php*.

3. 获取

说明

`object autoregistration.get(object parameters)`

该方法用于根据给定的参数来获取自动注册对象。

参数

(object) 定义需要输出的参数。

该方法只支持一个参数。

参数 [	据类型](/zh/manual/api/reference_commentary#data_types) 描述	
output	query	该参数对于参考注释中描述的所有 `get` 方法都是通用的。

返回值

(object) 返回自动注册对象。

例子

请求:

```
{
   "jsonrpc": "2.0",
    "method": "autoregistration.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tls_accept": "3"
    },
    "id": 1
}
```

来源

CAutoregistration::get() in *ui/include/classes/api/services/CAutoregistration.php*.

7. 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

配置导入

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.
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； updateExistin - (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; updateExistin - (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*.

配置导出

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*.

8. 联系

这个类是设计用于联系。

对象引用：

- 联系

可用的方法：

- correlation.create - 创建新的联系
- correlation.delete - 删除联系
- correlation.get - 获取联系
- correlation.update - 更新联系

> 对象

下列对象与联系 API 直接相关。

联系

联系对象具有以下属性。

属性类	描述
correlationid	字符串 *（ 读）* 联系的 ID。
name （需要的）	字符串 联 丨 的名称。
description	字符串 联 丨 的描述。
status	整数 丨 系是启用的还是禁用的。 可能的值有： 0 - （默认）启用的； 1 - 禁用的。

联系操作

联系操作对象定义了当一个联系被执行时，该操作的行为表现。它具有如下属性。

属性类	描述	
type （需要的）	整数操	类型。 可能的值： 0 - 关闭旧事件。 1 - 关闭新事件。

联系过滤

联系过滤对象定义了配置联系操作时，必须满足的一组条件。它具有如下属性。

属性类	描述	
evaltype (需要的)	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 (需要的)	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

联系过滤条件

联系过滤条件对象定义了在运行联系操作前必须检查的特定条件。

属性类	描述	
type (需要的)	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 =, &l	;>, like, not like 字符串	
5	新事件标签值 =, &l	;>, 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联系。

参数

(object/array) 要创建的联系。

另外，对于标准联系属性，该方法还接受以下参数。

参数类	描述	
operations（需要的）	数组与	建联系相关的操作。
filter（需要的）	对象与	联系 '相关的过滤对象。

返回值

(object) 返回一个对象，该对象包含 "correlationids" 属性下创建的联系的 ID。返回的 ID 的顺序与所传递的联系的顺序相匹配。

示例

创建一个新的事件标签联系

使用具有一个条件和一个操作的评估方法 AND/OR 创建一个联系。默认情况下，这个联系将被启用。

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联系。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

- 联系过滤
- 联系操作

来源

CCorrelation::create() in *frontends/php/include/classes/api/services/CCorrelation.php*.

删除

描述

`object correlation.delete(array correlationids)`

这个方法允许删除联系。

参数

(array) 要删除的联系的 ID。

返回值

(object) 返回一个对象，该对象包含 “correlationids” 属性下删除的联系的 ID。

示例

删除多个联系

删除 2 个联系。

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联系。

参数

(object/array) 要更新的联系的属性。

必须为每个联系定义 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，对于标准联系属性，该方法接受以下参数。

参数类	描述	
filter	对象替	当前筛选的联系筛选对象。
operations	数组替	已存在的操作的联系操作。

返回值

（object）返回一个对象，该对象包含 “correlationids” 属性下更新的联系的 ID。

示例

禁用联系

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

- 联系过滤
- 联系操作

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联系。

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的联系。
selectFilter	查询返	`filter`属性中的联系过滤。
selectOperations	查询返	`operations`属性中的联系操作。

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 [	用评论](/manual/api/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

检索联系

检索所有具有相关条件和操作的已配置过的联系。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

- 联系过滤
- 联系操作

来源

CCorrelation::get() in *frontends/php/include/classes/api/services/CCorrelation.php*.

9. 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

> 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 （需要的）	字符串仪表	名称。
userid	字符串仪表	属主的用户 ID。
private	整数仪	板共享的类型。 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 (需要的)	字符串仪表	小部件的类型。 可能的值： actionlog - 动作记录； 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 11。
y	整数仪	板顶部的垂直位置。 有效值范围从 0 到 63。
width	整数小	件的宽度。 有效值范围从 1 到 12。
height	整数小	件的高度； 有效值范围从 1 到 32。
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 （需要的）	整数小	件字段的值。 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 （需要的）	混合型取决	类型的小部件字段值。

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 （需要的）	字符串用户	ID。
permission （需要的）	整数权	级别的类型。 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 （需要的）	字符串用户	ID。
permission \\（需要的）整数	权限级别的类	。 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。

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	数组将 仪表板创建的仪表板小组件。
users	数组将 仪表板上创建的仪表板用户共享。
userGroups	数组将 仪表板上创建的仪表板用户组共享。

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相匹配。

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。

参数

(array) 要删除的仪表板的 ID。

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。

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	已存在的仪表板小部件的仪表板小组件。 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	已存在的部件的仪表板用户共享。

参数类	描述	
userGroups	数组替	已存在的部件的仪表板用户组共享。

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。

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/数组只返回拥	给定ID的仪表板。
selectWidgets	查询返	有`widgets`属性，并在仪表板中使用的小部件。

参数类	描述	
selectUsers	查询返	在 `users` 属性中共享仪表板的用户。
selectUserGroups	查询返	在 `userGroups` 属性中共享仪表板的用户组。
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`。

参数类	描述	
countOutput	布尔值在 [	用评论](/manual/api/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*.

10. 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的信息。其具有以下属性。

属性类	描述	
dhostid	字符串发现	主机的 ID。
druleid	字符串用于	测主机的发现规则的 ID。

属性类	描述	
lastdown	时间戳发现	主机最后异常的时间。
lastup	时间戳发现	主机最后正常的时间。
status	整数发	的主机是正常还是异常。如果一个主机至少还有一个活动的发现服务，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发现的主机。

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/数组只返回拥	给定ID的被发现主机。
druleids	字符串/数组只返回由	定的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	查询返	发现规则，该规则规定被发现主机在drules属性中以数组形式存在。
selectDServices	查询返	已发现服务，该服务运行在dservices属性中的主机上。 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 [	用评论](/manual/api/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*.

11. 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 dhost 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的信息。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。

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/数组只返回拥	给定 ID 的被发现服务。
dhostids	字符串/数组只返回被	现的服务，该服务属于给定的被发现主机。

参数类	描述	
dcheckids	字符串/数组只返回由	定的发现检查检测到的已发现的服务。
druleids	字符串/数组只返回被	定的发现规则检测到的服务。
selectDRules	查询返	发现规则，该规则规定被发现服务在 `drules` 属性中以数组形式存在。

参数类	描述	
selectDHosts	查询返	服务属于的已发现主机，该主机在dhosts属性中以数组形式存在。
selectHosts	查询返	与hosts属性中的服务具有相同IP地址的主机。 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： 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 [	用评论](/manual/api/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

检索在被发现主机"11" 上发现的所有被发现的服务。

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*.

12. 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。

发现检查

发现检查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) * 发现检查的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	字符串用逗号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 SNMP 社区字符串。 SNMPv1 和 SNMPv2 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 环境名称。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 (需要的)	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

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发现检查。

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/数组只返回拥	给定ID的发现检查。
druleids	字符串/数组只返回发	检查，该检查属于给定的发现规则。
dserviceids	字符串/数组只返回发	检查，该检查已检测到给定的已发现服务。

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：`dcheckid` 和 `druleid`。
countOutput	布尔值在 [	用评论](/manual/api/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发现检查

检索被发现规则”6” 使用的所有发现检查。

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*.

13. 发现规则

该类用于处理网络发现规则。

> **Note:**
> 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.

对象引用:

- 发现规则

可用方法:

- 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

> 1. 发现规则对象

以下是与 drule API 相关的对象。

发现规则

发现规则对象用于定义网络发现规则. 它有如下属性:

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。
name (必选)	string	发现规则名称。

属性类	描述	
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.

2.drule.create

描述

对象 `drule.create(object/array discroveryRules)`

该方法用于创建新的发现规则。

参数

(对象/数组) 要创建的发现规则。

除了标准的发现规则属性之外，该方法还接受以下参数：

参数类	描述	
dchecks (必选)	array	为发现规则创建发现检查。

返回值

(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。

示例

创建发现规则

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 10050 端口运行的 Zabbix agent 下。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

参考

- 发现检查

来源

CDRule::create() in *frontends/php/include/classes/api/services/CDRule.php*.

3.drule.delete

描述

对象 `drule.delete(array discoveryRuleIds)`

该方法用于删除发现规则。

参数

（数组）要删除的发现规则的 ID。

返回值

（对象）在 `druleids` 属性下，返回包含已删除的发现规则的 ID 的对象。

示例

删除多个发现规则

删除两个发现规则

请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.delete",
    "params": [
        "4",
        "6"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "4",
            "6"
        ]
    },
```

```
    "id": 1
}
```

来源

CDRule::delete() in *frontends/php/include/classes/api/services/CDRule.php*.

4.drule.get

描述

整数/数组 `drule.get(object parameters)`

该方法用于根据给定的参数获取发现规则。

参数

（对象）定义所需输出的参数。

该方法支持以下参数。

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.

参数类	描述	
selectDHosts	query	在 dhosts 属性下，返回发现规则创建的发现主机。 支持 count.
limitSelects	integer	限制子选项返回的记录数 适用于以下选项: selectDChecks - 结果按 dcheckid 排序; selectDHosts - 结果按 dhostsid 排序

参数类	描述	
sortfield	string/array	结果按给定属性排序. 可能的值: `druleid` 和 `name`.
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	

返回值

（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。

示例

获取所有发现规则

获取所有已配置的发现规则和检查已使用的发现规则。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.get",
    "params": {
        "output": "extend",
        "selectDChecks": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```


参考

- Discovered host
- Discovery check
- 已发现主机
- 发现检查

来源

CDRule::get() in *frontends/php/include/classes/api/services/CDRule.php*.

5.drule.update

描述

对象 `drule.update(object/array discoveryRules)`

该方法用于更新已存在的发现规则。

参数

（对象/数组）更新的发现规则属性。

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。

除了标准的发现规则属性外，该方法有以下参数：

参数类	描述	
dchecks	array	替代已存在的发现检查.

返回值

（对象）在 `druleids` 属性下，返回包含已更新的发现规则的 ID 对象。

示例

更改发现规则的 IP 范围

将发现规则的 IP 范围更改为 192.168.2.1-255 。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "drule.update",
    "params": {
        "druleid": "6",
        "iprange": "192.168.2.1-255"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

参考

- 发现检查

来源

CDRule::update() in *frontends/php/include/classes/api/services/CDRule.php*.

14.Event 事件

事件　这个类用于配合事件使用

对象引用:

- 事件

可用方法:

- event.get - 获取事件
- event.acknowledge - 确认事件

> 1.Event 对象

以下对象与 event[事件] API 直接相关。

事件

Note:
事件是由 Zabbix server 创建，并且不能通过 API 进行修改。

事件对象具有以下属性：

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 - active agent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。 触发器事件可能的值: 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	已恢复事件的名称。

属性类	描述	
value	integer	相关对象的状态。 触发器事件可能的值: 0 - 正常; 1 - 异常. 发现事件可能的值: 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。 可能的值: 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 tag

The event tag object has the following properties.

Property	Type	Description
tag	string	Event tag name.
value	string	Event tag value.

Media type URLs

Object with media type url have the following properties.

Property	Type	Description
name	string	Media type defined URL name.
url	string	Media type defined URL value.

Results will contain entries only for active media types with enabled event menu entry. Macro used in properties will be expanded, but if one of properties contain non expanded macro both properties will be excluded from results. Supported macros described on page.

2.event.acknowledge

描述

对象 `event.acknowledge(object/array parameters)`

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。
- 新增信息。
- 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。

Attention:
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’。

参数

（对象/数组）包含事件 ID 和应执行的更新操作的参数。

参数类	描述	
eventids (必选)	string/object	确认事件的ID。
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 - 新增消息; 8 - 更改严重等级。

参数类	描述	
message	string	消息文本。如果操作包含'新增消息'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。

返回值

（对象）在 `eventids` 属性下，返回一个包含被更新事件的 ID。

示例

确认一个事件

确认一个事件并留下一个信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 6,
        "message": "Problem resolved."
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427"
        ]
    },
    "id": 1
}
```

更改事件的严重等级

更改多个事件的严重等级并留下一个信息。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427",
            "20428"
        ]
    },
    "id": 1
}
```

来源

CEvent::acknowledge() in *frontends/php/include/classes/api/services/CEvent.php*.

3.event.get

描述

整数/数组 event.get(object parameters)

此方法用于根据给定参数来获取事件。

参数

（对象）定义所需输出的参数。

此方法支持以下参数：

参数类	描述	
eventids	string/array	仅返回具有给定ID的事件。
groupids	string/array	仅返回所属主机组的对象创建的事件。
hostids	string/array	仅返回所属主机的对象创建的事件。

参数类	描述	
objectids	string/array	仅返回由给定对象创建的事件。
applicationids	string/array	仅返回所属应用的对象创建的事件。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	若设置为“true”，则只返回已被确认的事件。

参数类	描述	
severities	integer/array	仅返回符合所属严重程度的事件。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	

参数类	描述
limit	integer
output	query
preservekeys	boolean
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	boolean

返回值

（整数/数组）返回:

- 一个数组对象;
- 如果使用了 `countOutput` 参数，返回获取对象的数值。

示例

获取触发器事件

从触发器"13926." 中获取最新事件。

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
}
```

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
}
```


按时间段获取事件

在 2012-10-9 至 2012-10-10 时间段内，以反向时间顺序获取所有已被创建的事件。

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
}
```

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
}
```


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

来源

CEvent::get() in *frontends/php/include/classes/api/services/CEvent.php*.

15. 图表

图表　这个类用于配合监控项使用

参考对象:

- 图表

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

> 1. 图表对象

以下对象与 图表 API 直接相关。

图表

图表对象具有以下属性:

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。

属性类	描述	
width (必选)	integer	图表的宽度(单位:像素)。
flags	integer	*(readonly)* 图表的来源。 可能值: 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。

属性类	描述	
percent_left	float	百分比线(左)。 默认: 0。
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。 可能值: 0 - (默认)以2D展示; 1 - 以3D展示。

属性类	描述	
show_legend	integer	是否在图表上显示图例。 可能值: 0 - 隐藏; 1 - *(默认)* 显示。
show_work_period	integer	是否在图表上显示工作时间. 可能值: 0 - 隐藏; 1 - *(默认)* 显示。
templateid	string	*(只读)* 父模板图表的ID。

属性类	描述	
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。
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) 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。

2.graph.create

描述

对象 graph.create(object/array graphs)

此方法用于创建新的图表。

参数

(对象/数组) 要创建的图表。

除了标准图表属性 外，此方法还接受以下参数。

参数类	描述	
gitems (必选)	array	创建到图表中的监控项。

返回值

(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相匹配。

示例

创建一个图表

创建一个包含两个监控项的图表。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

参考

- 图表监控项

来源

CGraph::create() in *frontends/php/include/classes/api/services/CGraph.php*.

3.graph.delete

描述

对象 `graph.delete(array graphIds)`

此方法用于删除图表。

参数

.（数组）要删除的图表的 ID。

返回值

（对象）在 `graphids` 属性下，返回一个包含已删除图表的对象。

示例

删除多个图表

删除两个图表。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.delete",
    "params": [
        "652",
        "653"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

来源

CGraph::delete() in *frontends/php/include/classes/api/services/CGraph.php*.

4.graph.get

描述

整数/数组 graph.get(object parameters)

此方法用于根据给定参数来获取图表。

参数

（对象）定义所需输出的参数。

此方法支持以下参数：

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。

参数类	描述	
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。
itemids	string/array	仅返回包含给定监控项的图表。
templated	boolean	若设置为真(`true`),仅返回属于模板的图表。

参数类	描述	
inherited	boolean	若设置为真(`true`),仅返回从模板继承的图表。
expandName	flag	在图表名称中展开宏。
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。

参数类	描述	
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

（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值。

示例

从主机中获取图表

从主机"10107"中获取所有图表，并依据名称进行排序。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.get",
    "params": {
        "output": "extend",
        "hostids": 10107,
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```


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

来源

CGraph::get() in *frontends/php/include/classes/api/services/CGraph.php*.

5.graph.update

描述

对象 graph.update(object/array graphs)

此方法用于更新已存在的图表。

参数

(对象/数组) 要更新的图表属性。

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。

除了标准图表属性 之外，此方法还接受以下参数。

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。

返回值

(对象) 在 graphids 属性下，返回一个包含已更新图表的 ID 的对象。

示例

设置 Y 刻度的最大值

设置 Y 刻度的最大值为固定值 100。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graph.update",
    "params": {
        "graphid": "439",
        "ymax_type": 1,
        "yaxismax": 100
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

来源

CGraph::update() in *frontends/php/include/classes/api/services/CGraph.php*.

16. 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

> 1. 图表监控项对象

以下对象与 `graphitem` API 直接相关。

图表监控项

Note:
图表监控项只能通过 `graph` API 进行修改。

图表监控项具有以下属性:

属性类	描述	
gitemid	string	*(必选)* 图表监控项的ID。

属性类	描述	
color (必选)	string	绘制图形监控项的颜色，使用十六进制码表示。
itemid (必选)	string	监控项的ID。
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。

属性类	描述	
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。
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。

属性类	描述	
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。
yaxisside	integer	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。

2.graphitem.get

描述

整数/数组 `graphitem.get(object parameters)`

此方法用于根据给定参数来获取图表监控项。

参数

（对象）定义所需输出的参数。

此方法支持以下参数：

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。

参数类	描述	
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	

返回值

（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。

示例

从图表中获取图表监控项

获取图表中使用的所有图表监控项以及有关监控项和主机的其他信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphitem.get",
    "params": {
        "output": "extend",
        "graphids": "387"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```


参考

- 图表

来源

CGraphItem::get() in *frontends/php/include/classes/api/services/CGraphItem.php*.

17. 图表原型

图表原型

这个类用于配合图表原型使用。

对象引用:

- 图表原型

可用方法:

- graphprototype.create - 新建一个图表原型
- graphprototype.delete - 删除一个图表原型
- graphprototype.get - 获取一个图表原型
- graphprototype.update - 更新一个图表原型

> 1.Graph prototype object

以下对象与 graphprototype API 直接相关。

图表原型

图表原型对象具有以下属性:

属性类	描述	
graphid	string	*(只读)* 图表原型的ID。
height (必选)	integer	图表原型的高度(单位：像素)。
name (必选)	string	图表原型的名称。
width (必选)	integer	图表原型的宽度(单位：像素)。

属性类	描述	
graphtype	integer	图表原型布局类型。 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。
percent_right	float	右侧百分比线。 默认: 0。

属性类	描述	
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)* 图表原形的父模板的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。

2.graphprototype.create

描述

对象 `graphprototype.create(object/array graphPrototypes)`

此方法用于创建新的图表原型。

参数

（对象/数组）将要创建的图表原型。

除了标准图表原型参数外, 此方法还接受以下参数：

参数类	描述	
gitems (必选)	array	创建到图表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。

返回值

（对象）在 `graphids` 属性下，返回一个包含已被创建的图表原型 ID 的对象。返回的 ID 的顺序与传递的图表原型的顺序相匹配。

示例

创建一个图表原型

创建一个含有两个监控项的图表原型。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

参考

- Graph item
- 图表监控项

来源

CGraphPrototype::create() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

3.graphprototype.delete

描述

对象 `graphprototype.delete(array graphPrototypeIds)`

此方法用于删除图表原型。

参数

（数组）需要删除的图表原型的 ID。

返回值

（对象）在 `graphids` 属性下，返回一个包含已经删除的图表原型的 ID 的对象。

示例

删除多个图表原型

删除两个图表原型

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
}
```

来源

CGraphPrototype::delete() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

4.graphprototype.get

描述

整数/数组 `graphprototype.get(object parameters)`

此方法用于根据给定的参数来获取图表原型。

参数

（对象）定义所需输出的参数。

此方法支持以下参数:

参数类	描述	
discoveryids	string/array	仅返回属于给定自动发现规则的图表原型。

参数类	描述	
graphids	string/array	仅返回含有给定ID的图表原型。
groupids	string/array	仅返回属于给定主机组的主机的图表原型。
hostids	string/array	仅返回属于给定主机的图表原型。

参数类	描述	
inherited	boolean	如果设置此参数为 `true`，则仅返回从模板继承的图表原型。
itemids	string/array	仅返回包含给定监控项原型的图表原型。

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表原型。
templateids	string/array	仅返回属于给定模板的图表原型。
selectDiscoveryRule	query	在 `discoveryRul` 属性下，返回图表原型所属的低级别发现规则。

参数类	描述	
selectGraphItems	query	在`gitems`属性下，返回在图表原型中使用的图表监控项。
selectGroups	query	在`groups`属性下，返回图表原型所属的主机组。
selectHosts	query	在`hosts`属性下，返回图表原型所属的主机。

参数类	描述	
selectItems	query	在 `items` 属性下，返回在图表原型中使用的监控项以及监控项原型。
selectTemplates	query	在 `templates` 属性下，返回图表原型所属的模板。

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 - 图表原型所属主机的

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	

返回值

（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。

示例

从低级别发现规则获取图表原型

从低级别发现规则获取所有图表原型。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```

参考

- 发现规则
- 图表监控项
- 监控项
- 主机
- 主机组
- 模板

来源

CGraphPrototype::get() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

5.graphprototype.update

描述

对象 `graphprototype.update(object/array graphPrototypes)`

此方法用于更新已存在的图表原型。

参数

(对象/数组) 需要更新的图表原型。

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。

除了标准图表原型属性外，此方法还接受以下参数：

参数类	描述	
gitems	array	用于替换现有图形监控项的图表监控项。如果图表监控项项定义了 gitemid 属性，它将被更新，否则将创建一个新的图表监控项。

返回值

(对象) 在 graphids 属性下，返回一个已更新的图表原型的对象的 ID。

示例

更新图表原型大小

将图表原型的大小从 1100 更新为 400(单位：像素)。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.update",
    "params": {
        "graphid": "439",
        "width": 1100,
        "height": 400
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

来源

CGraphPrototype::update() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

18. 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 - 检索历史数据。

> 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，无法通过 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
clock	时间戳获取	的时间
itemid	字符串 it	m 相关的 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
clock	时间戳获取	的时间
itemid	字符串 it	m 相关的 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
clock	时间戳获取	的时间
itemid	字符串 it	m 相关的 ID
ns	整数获	值时的纳秒

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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 [	型](/zh/manual/api/reference_commentary#data_types) 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 it	m 相关的 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 [	型](/zh/manual/api/reference_commentary#data_types) 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 it	m 相关的 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 事件日志条目时间
value	文本获	到的值

获取

描述

`integer/array history.get(object parameters)`

该方法允许根据给定的参数检索历史数据。

参考: 已知问题

<note important> 如果内置数据管理 housekeeper 尚未移除已删除实体的历史数据，则此方法可能会返回该数据。:::

参数

`(object)` 定义期望输出的参数。

该方法支持以下参数：

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 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` 和 `clock`。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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：注释参考。
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

(integer/array) 返回任一:

- 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 *ui/include/classes/api/services/CHistory.php*.

19. 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 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
available	整数 *	只读) *Zabbix 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)* Zabbix agent 不可用状态的下一次轮询时间。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
error	字符串 *(	读)* Zab-bix agent 不可用时的错误信息。
errors_from	时间戳 *(	读)* Zab-bix 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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
inventory_mode	整数主	资产清单填充模式。 可能的值： -1 - 禁用； 0 - (默认) 手动； 1 - 自动。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 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+。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
ipmi_disable_until	时间戳 *(	读)* IPMI agent 不可用状态的下一次轮询时间。
ipmi_error	字符串 *(	读)* IPMI agent 不可用时的错误信息。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
jmx_disable_until	时间戳 *(	读)* JMX agent 不可用状态的下一次轮询时间。
jmx_error	字符串 *(	读)* JMX agent 不可用时的错误信息。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
maintenanceid	字符串 *(	读)*目前对主机生效的维护模式ID。
name	字符串主机	见名。 默认: `host`属性值。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
proxy_hostid	字符串用于 控主机的Proxy服务器的hostid。
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)* SNMP agent不可用状态的下一次轮询时间。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
snmp_error	字符串 *(	读)* SNMP agent 不可用时的错误信息。
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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
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。不要将敏感信息放在 PSK 身份中，它会通过网络以未加密的方式传输，以通知接收者要使用哪个 PSK。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 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 A。
13	macaddress_b	字符串 MA	地址 B。

ID	属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
29	model	字符串型号	
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	字符串详细	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 A。
60	poc_1_phone_b	字符串主 P	C 电话 B。
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 A。
67	poc_2_phone_b	字符串次 P	C 电话 B。
69	poc_2_screen	字符串次 P	C 显示名。
8	serialno_a	字符串序列	A。
9	serialno_b	字符串序列	B。
48	site_address_a	字符串所在	址 A。
49	site_address_b	字符串所在	址 B。
50	site_address_c	字符串所在	址 C。
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 A。
19	software_app_b	字符串应用	件 B。
20	software_app_c	字符串应用	件 C。
21	software_app_d	字符串应用	件 D。
22	software_app_e	字符串应用	件 E。
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	字符串类型	情。
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	。

主机标签

主机标签对象具有以下属性。

属性类	描述	
tag (必选)	字符串主机	签名称。
value	字符串主机	签值。

创建

描述

`object host.create(object/array hosts)`

这个方法可以用来创建主机。

参数

(`object/array`) 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 [	型](//zh/manual/api/reference_commentary#data_types) 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。
tags	对象/数组主机 [	签](//zh/manual/api/reference/host/object#host_tag)。
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

(`object`) 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

创建一个具有 IP 接口和标签且名为 “Linux Server” 的主机，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"
            }
        ],
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
```

```
        "macros": [
            {
                "macro": "{$USER_ID}",
                "value": "123321"
            },
            {
                "macro": "{$USER_LOCATION}",
                "value": "0:0:0",
                "description": "latitude, longitude and altitude coordinates"
            }
        ],
        "inventory_mode": 0,
        "inventory": {
            "macaddress_a": "01234",
            "macaddress_b": "56768"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "107819"
        ]
    },
    "id": 1
}
```

创建一个 SNMP 接口的主机

创建一个带有 SNMPv3 接口的”SNMP host” 主机包含以下信息。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "SNMP host",
        "interfaces": [
            {
                "type": 2,
                "main": 1,
                "useip": 1,
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": 3,
                    "bulk": 0,
                    "securityname": "mysecurityname",
                    "contextname": "",
                    "securitylevel": 1
                }
            }
        ],
        "groups": [
            {
                "groupid": "4"
            }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10658"
        ]
    },
    "id": 1
}
```

Creating a host with PSK encryption

Create a host called "PSK host" with PSK encryption configured. Note that the host has to be pre-configured to use PSK.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "PSK host",
        "interfaces": [
            {
                "type": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050",
                "useip": 1,
                "main": 1
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "tls_accept": 2,
        "tls_connect": 2,
        "tls_psk_identity": "PSK 001",
        "tls_psk": "1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "id": 1
}
```

参考

- 主机组
- 模板
- 用户宏
- 主机接口
- 主机资产清单
- 主机标签

来源

CHost::create() in ui/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 ui/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 [	型](//zh/manual/api/reference_commentary#data_types) 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

- 主机. 更新
- 用户宏
- 主机接口

来源

CHost::massRemove() in ui/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 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
hosts (必选)	对象/数组要更新	主机。 机必须已定义过 hostid 属性。
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。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
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_m 替换。
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 [	型](//zh/manual/api/reference_commentary#data_types) 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 templateid 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

- 主机. 更新
- 主机. 批量添加
- 主机. 批量删除
- 主机组
- 模板
- 用户宏
- 主机接口

来源

CHost::massUpdate() in ui/include/classes/api/services/CHost.php.

批量添加

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 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
hosts (必选)	对象/数组要更新	主机。 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",
                "description": "Test description"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10160",
            "10167"
        ]
    },
    "id": 1
}
```

参考

- 主机. 更新
- 主机组
- 模板
- 用户宏
- 主机接口

来源

CHost::massAdd() in *ui/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。

必须为每个主机定义 `hostid` 属性，所有其他属性都是可选的。当只更新指定的属性，其他属性将保持不变。

但是请注意，更新主机名称还会通过主机的名称值来更新主机的可见名称（如果未提供或为空）。

另外，对于标准主机属性，此方法接受如下参数。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
groups	对象/数组替换主	所属的当前主机组。 主机组必须具有定义的 `groupid` 属性。请求中未列出的所有
interfaces	对象/数组替换当	的主机接口。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
tags	object/array	替换当前的主机标签。 请求中未列出的所有标签将被删除。
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。请求中未列出的所有模板将仅取消链接。 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位置。

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
}
```

更新主机标签

用一个新的标签替换所有的标签。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "tags": {
            "tag": "OS",
            "value": "CentOS 7"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

Updating host encryption

Update the host ”10590” to use PSK encryption only for connections from host to Zabbix server, and change the PSK identity and PSK key. Note that the host has to be pre-configured to use PSK.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10590",
        "tls_connect": 1,
        "tls_accept": 2,
        "tls_psk_identity": "PSK 002",
        "tls_psk": "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "id": 1
}
```

参考

- 主机. 批量创建
- 主机. 批量更新
- 主机. 批量删除
- 主机组
- 模板
- 用户宏
- 主机接口
- 主机资产清单
- 主机标签

来源

CHost::update() in ui/include/classes/api/services/CHost.php.

获取

描述

`integer/array host.get(object parameters)`

该方法允许根据指定的参数检索主机。

参数

`(object)` 定义期望输出的参数。

该方法支持以下参数。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 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。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述
dserviceids	字符串/数组返回与指定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/数组返回指定主机ID的主机。
httptestids	字符串/数组返回指定页监测的主机。
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。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述
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。
with_items	标识返含有监控项的主机。 覆盖 `with_monito` 和 `with_simple` 参数。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
with_item_prototypes	flag	仅返回具有项目原型的主机。 覆盖 `with_simple_` 参数。
with_simple_graph_item_prototypes	flag	仅返回具有项目原型的主机，这些主机已启用创建并且具有数字类型的信息。
with_applications	标识返	含有应用集的主机。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述
with_graphs	标识返含有图表的主机。
with_graph_prototypes	flag 只返回具有原型图的主机。
with_httptests	标识返含有web监测的主机。 覆盖with_monito参数。
with_monitored_httptests	标识返含有启动网页监测的主机。
with_monitored_items	标识返启用监控项的主机。 覆盖with_simple_参数。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
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。
with_triggers	标识返	含有触发器的主机. 覆盖 with_monito 参数。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
withProblemsSuppressed	布尔值返回	抑制问题的主机。 可能值： null - (默认) 所有主机； true - 仅已抑制问题的主机； false - 仅未抑制问题的主机。
evaltype	整数标	搜索规则。 可能值： 0 - (默认) 和/或； 2 - 或。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
severities	整数/数组返回只	指定问题严重性的主机。仅当问题对象是触发器时适用。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述
tags	数组/对象仅返回有指定标签的主机。按标记进行精确匹配，并根据运算符的值按标记值区分大小写或不区分大小写。格式：[{"标签": "<tag>"，"值": "<value>"，"操作符": "<operator>...]。空数组将返回所有主

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
inheritedTags	布尔值返回	所有链接的模板中且带有“标签”的主机。默认： 可能值： true - 链接的模板还必须具有给定的标签; false - (默认) 链接的模板标签将被忽略。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
selectApplications	查询在	applications](// 属 性 中 返 回 来 自 主 机 的 应 用 集。 支 持 count。
selectDiscoveries	查询在	discoveries](//z 属 性 中 返 回 来 自 主 机 的 底 层 自 动 发 现。 支 持 count。
selectDiscoveryRule	查询在	discoveryRule] 属 性 中 返 回 创 建 主 机 的 底 层 自 动 发 现 规 则。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
selectGraphs	查询在	graphs](//zh/ma 属性中返回来自主机的图表。 支持 `count`。
selectGroups	query	在groups 属性中返回主机所属的主机组数据。

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述
selectHostDiscovery	查询在 hostDiscovery 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，或者把一个原型主机和一个底层自动发现规则连

参数 [	型](//zh/manual/api/reference_commentary#data_types) 描述	
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
}
```

按主机标签进行检索

检索”Host name” 标签等于”Linux server” 的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server",
                "operator": 1
            }
        ]
    },
    "auth": "7f9e00124c75e8f25facd5c093f3e9a0",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "tags": [
                {
                    "tag": "Host name",
                    "value": "Linux server"
                },
                {
                    "tag": "OS",
                    "value": "RHEL 7"
                }
            ]
        }
    ],
    "id": 1
}
```

检索主机级别和其链接的父类模板中具有这些标签的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "tags": [{"tag": "A", "value": "1", "operator": "0"}],
        "inheritedTags": true
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
```

```
    "result": [
        {
            "hostid": "10623",
            "name": "PC room 1"
        },
        {
            "hostid": "10601",
            "name": "Office"
        }
    ],
    "id": 1
}
```

使用标签和模板标签进行检索

使用标签以及链接到父模板的所有标签检索主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "hostids": 10502,
        "selectTags": ["tag", "value"],
        "selectInheritedTags": ["tag", "value"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10502",
            "name": "Desktop",
            "tags": [
                {
                    "tag": "A",
                    "value": "1"
                }
            ],
            "inheritedTags": [
                {
                    "tag": "B",
                    "value": "2"
                }
            ]
        }
    ],
    "id": 1
}
```

按问题严重性进行检索

检索有” 灾难” 级别问题的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
```

```
        "output": ["name"],
        "severities": 5
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10160",
            "name": "Zabbix server"
        }
    ],
    "id": 1
}
```

检索有”平均”和”高”级别问题的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "severities": [3, 4]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "20170",
            "name": "Database"
        },
        {
            "hostid": "20183",
            "name": "workstation"
        }
    ],
    "id": 1
}
```

参考

- 主机组
- 模板
- 用户宏
- 用户接口

来源

CHost::get() in ui/include/classes/api/services/CHost.php.

20. 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 - 创建新主机组
- hostgroup.delete - 删除主机组
- hostgroup.get - 获取主机组
- hostgroup.massadd - 添加主机组的相关对象
- hostgroup.massremove - 删除主机组的相关对象
- hostgroup.massupdate - 替换或删除主机组的相关对象
- 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
groupid	字符串 *(	读)* 主机组的ID。
name (必选)	字符串主机	的名称。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
internal	整数 *	只读)* 无论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 ui/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 ui/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 [	型](/zh/manual/api/reference_commentary#data_types) 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 ui/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 [	型](/zh/manual/api/reference_commentary#data_types) 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

- 主机组. 更新
- 主机组. 批量添加
- 主机
- 模板

来源

CHostGroup::massUpdate() in ui/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 [	型](/zh/manual/api/reference_commentary#data_types) 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

- 主机
- 模板

来源

CHostGroup::massAdd() in ui/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 ui/include/classes/api/services/CHostGroup.php.

获取

描述

`integer/array 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 [	型](/zh/manual/api/reference_commentary#data_types) 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。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述
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。
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。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
with_graph_prototypes	标识返	给定图表原型包含主机的主机组。
with_hosts_and_templates	标识返	包含主机或模板的主机组。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
with_httptests	标识返	给定web检查包含主机的主机组。 覆盖with_monito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_monito和with_simple参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
with_item_prototypes	标识返	包含带有项目原型主机的主机组。 覆盖 with_simple_ 参数。
with_simple_graph_item_prototypes	标识返	包含带有项目原型的主机的主机组，这些主机已启用创建并且具有数字类型的信息。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述
with_triggers	标识返 给定触发器包含主机的主机组。 覆盖 with_monito 参数。
selectDiscoveryRule	查询在 discoveryRule] 属性中返回创建主机组的发现规则。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述
selectGroupDiscovery	查询在 groupDiscovery 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–(字符串)I 已经发现主机组的 ID； lastcheck–(时间戳) 主机组最后一次

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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

- 主机
- 模板

来源

CHostGroup::get() in ui/include/classes/api/services/CHostGroup.php.

21. 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

> 主机接口对象

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。

<note important> 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 [	型](/zh/manual/api/reference_commentary#data_types) 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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
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 [	型](/zh/manual/api/reference_commentary#data_types) 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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
type (必选)	整数接	类型。可能的值：1 - agent；2 - SNMP；3 - IPMI；4 - JMX。
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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
details	array	Additional object for interface. **Required** if interface 'type' is SNMP.

详情标签

详情（details）对象具有以下属性。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
version	整数 S	MP 接口版本。 可能的值： 1 - SNMPv1； 2 - (默认) SNMPv2c； 3 - SNMPv3

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
bulk	整数是	使用批量SNMP请求。 可能的值： 0 - 不使用批量SNMP请求； 1 - (默认) 使用批量SNMP请求。
community	字符串 SN	P com-mu-nity。仅由SN-MPv1和SN-MPv2接口使用。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
securityname	字符串 SN Pv3 安全名称。仅由 SN-MPv3 接口使用。
securitylevel	整数 S MPv3 安全级别。仅由 SN-MPv3 接口使用。 可能的值： 0 - (默认) - noAuthNoPriv； 1 - authNoPriv； 2 - authPriv。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
authpassphrase	字符串 SN	Pv3 认证密码。仅由 SN-MPv3 接口使用。
privpassphrase	字符串 SN	Pv3 私有密码。仅由 SN-MPv3 接口使用。
authprotocol	整数 S	MPv3 身份验证协议。仅由 SN-MPv3 接口使用。 可能的值： 0 - (默认) - MD5； 1 - SHA。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
privprotocol	整数 S	MPv3 隐私协议。仅由 SN-MPv3 接口使用。 可能的值： 0 - (默认) - DES； 1 - AES。
contextname	字符串 SN	Pv3 上下文名称。仅由 SN-MPv3 接口使用。

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

(object/array)) 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

(object) 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

- 主机接口. 批量添加
- 主机. 批量添加

来源

CHostInterface::create() in ui/include/classes/api/services/CHostInterface.php.

Source

CHostInterface::create() in *ui/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

(array) 要删除主机接口的 ID。

返回值

(object) 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

- 主机接口. 批量删除
- 主机. 批量删除

来源

CHostInterface::delete() in ui/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

(object) 包含要更新的主机的 ID 和要删除的接口的参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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

(object) 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

- 主机接口. 删除
- 主机. 批量删除

来源

CHostInterface::massRemove() in *ui/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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
hosts (必选)	对象/数组要更新	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

- 主机接口. 创建
- 主机. 批量添加
- 主机

来源

CHostInterface::massAdd() in *ui/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

`(object/array)` 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

`(object)` 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 ui/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

`(object)` 包含要更新的主机 ID 和新主机接口的参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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

`(object)` 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

- 主机. 更新
- 主机. 批量更新

来源

CHostInterface::replaceHostInterfaces() in *ui/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

`(object)` 定义期望输出的参数。

该方法支持以下参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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refere
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

(integer/array) 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",
    "result": [
        {
            "interfaceid": "30050",
            "hostid": "30057",
            "main": "1",
            "type": "1",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "10050",
            "details": []
        },
        {
            "interfaceid": "30067",
            "hostid": "30057",
            "main": "0",
            "type": "1",
            "useip": "0",
            "ip": "",
            "dns": "localhost",
            "port": "10050",
            "details": []
        },
        {
            "interfaceid": "30068",
            "hostid": "30057",
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "161",
            "details": {
                "version": "2",
                "bulk": "0",
                "community": "{$SNMP_COMMUNITY}"
            }
        }
    ],
    "id": 1
}
```


参考

- 主机
- 监控项

来源

CHostInterface::get() in *ui/include/classes/api/services/CHostInterface.php*.

22. 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 `hostprototype`API 直接相关。

主机原型

主机原型对象具有以下属性:

属性 [	型](/zh/manual/api/reference_commentary#data_types) 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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
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。
inventory_mode	整数主 资产清单填充模式。 可能的值： -1 - (默认) 禁用； 0 - 手动； 1 - 自动。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
templateid	字符串 *(	读)* 父模板主机原型的ID。
discover	整数主	原型自动发现状态。 可能的值： 0 - (默认)会发现新主机; 1 - 不会发现新主机，现有主机将被标记为丢失。

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 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 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。
macros	对象/数组将为主	原型创建的用户宏。
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

- 组链接
- 组原型
- 用户宏

来源

CHostPrototype::create() in *ui/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

(array) 要删除主机原型的 ID。

返回值

(object) 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 ui/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

`(object/array)` 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
groupLinks	数组组	链接](/zh/manual/来替换主机原型上的当前组链接。
groupPrototypes	数组组	原型](/zh/manual/替换主机原型中已存在的组原型。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
macros	对象/数组 [用户	](/manual/api/r 来 替 换 当 前 的 用 户 宏。 请 求 中 未 列 出 的 所 有 宏 都 将 被 删 除。
templates	对象/数组 [模板	(/zh/manual/ap 来 替 换 当 前 已 连 接 的 模 板。 模 板 必 须 已 定 义 templateid 属 性。

返回值

（object）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

- 组链接
- 组原型
- 用户宏

来源

CHostPrototype::update() in *ui/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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
hostids	字符串/数组返回给定	D 的主机原型。
discoveryids	字符串/数组返回归属	定 LLD 规则的主机原型。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
inherited	布尔值如果	置为 true, 只返回模板从模板继承的项目。
selectDiscoveryRule	查询在	discoveryRule](属性中返回主机原型归属的 LLD 规则。
selectGroupLinks	查询在	groupLinks](/zh 属性中返回主机原型的组链接。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
selectGroupPrototypes	查询在	groupPrototype 属 性 中 返 回 主 机 原 型 的 组 原 型。
selectMacros	查询在	macros](/zh/ma 属 性 中 返 回 主 机 原 型 的 宏。
selectParentHost	查询在	parentHost](/zh 属 性 中 返 回 主 机 原 型 所 属 的 主 机。
selectTemplates	查询在	templates](/zh/ 属 性 中 返 回 连 接 到 主 机 原 型 的 模 板。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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` 和 `status`。
countOutput	布尔值这些	数对于所有 get 方法都是通用的，详情可参考通用 Zabbix API 信息.
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",
            "discover": "0",
            "groupLinks": [
                {
                    "group_prototypeid": "4",
                    "hostid": "10092",
                    "groupid": "7",
                    "templateid": "0"
                }
            ],
            "groupPrototypes": [
                {
                    "group_prototypeid": "7",
                    "hostid": "10092",
                    "name": "{#CLUSTER.NAME}",
                    "templateid": "0"
                }
            ]
        }
    ],
    "id": 1
}
```

参考

- 组链接
- 组原型
- 用户宏

来源

CHostPrototype::get() in *ui/include/classes/api/services/CHostPrototype.php*.

23. 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 [	型](/zh/manual/api/reference_commentary#data_types) 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
iconmappingid	字符串 *(	读)* 图标拓扑图 ID。
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。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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。
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

(`object/array`) 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 [	型](/zh/manual/api/reference_commentary#data_types) 描述
mappings (必选)	数组为 标拓扑创建图标映射。

返回值

(`object`) 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

- 图标映射

来源

CIconMap::create() in ui/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

(array) 需要删除的图标拓扑图 ID。

返回值

(object) 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 ui/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

(object/array) 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。

除了标准图标映射属性外，该方法还接受以下参数。

参数 [	型](/manual/api/reference_commentary#data_types) 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

- 图标映射

来源

CIconMap::update() in ui/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

(object) 定义要输出的参数。

该方法支持以下参数：

参数 [	型](/zh/manual/api/reference_commentary#data_types) 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refere
editable	布尔值::	
excludeSearch	布尔值::	
filter	对象:	:
limit	integer	
output	查询:	:
preservekeys	布尔值::	

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
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

- 图标映射

来源

CIconMap::get() in ui/include/classes/api/services/CIconMap.php.

24. 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 [	ype](/zh/manual/api/reference_commentary#data_types) 描	
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

`(object/array)` 要创建的图像。

除标准图像属性之外，该方法接受以下参数：

属性 [	型](/zh/manual/api/reference_commentary#data_types) 说明	
name (必选)	字符串图像	称。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 说明	
imagetype (必选)	整数图	类型。可能的值： 1 - (默认) 图标； 2 - 背景图片。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 说明	
image (必选)	字符串 Ba 支	e64 编码图像，编码图像的最大大小为 1 MB。可以通过更改 ZBX_MAX_IMAG 常量值来调整最大尺寸的大小。的图像格式为：PNG，JPEG，GIF。

返回值

（object）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 *ui/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

(array) 要删除的图像 ID。

返回值

(object) 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 ui/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

`(object/array)` 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。

除了标准图像属性值外，该方法接受以下参数：

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述
image	字符串 Base64 编码图像。编码图像的最大大小为 1 MB。可以通过更改 ZBX_MAX_IMAGE_SIZE 常量值来调整最大尺寸的大小。支持的图像格式为：PNG，JPEG，GIF。

返回值

(object) 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 *ui/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

`(object)` 定义要输出的参数.

该方法支持如下参数:

属性 [	型](/zh/manual/api/reference_commentary#data_types) 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refere
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 *ui/include/classes/api/services/CImage.php*.

25. 监控项

此类用于管理监控项。

对象引用：

- 监控项

可用的方法：

- item.create - 创建新监控项
- item.delete - 删除监控项
- item.get - 检索监控项
- item.update - 更新监控项

> 监控项对象

以下对象与 “item” API 直接相关。

监控项

> **Note:**
> Web 监控项无法通过 Zabbix API 直接创建，更新或删除。

监控项对象具有以下属性。

属性类	描述	
itemid	string	*(只读)* 监控项 ID。

属性类	描述	
delay (必需)	string 可	更新监控项的时间间隔。接受具有后缀(30s,1m,2h,1d)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自定义时间间

属性类	描述	
hostid (必需的)	string	该监控项所属的主机ID。对于更新操作，这个字段是只读.

属性类	描述		
interfaceid (必需)	string	监控项主机接口的ID。仅用于主机项。适用于Zabbix agent(活动)，Zabbix内部，Zabbix trapper，依赖监控项，Zabbix聚合，数据库监控和计算监控项	
	key_ (必需)	string	监控项key。
name (必需)	string	监控项名称。	

属性类	描述	
type (必需)	integer	项目的类型。 取值范围: 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;

属性类	描述	
url (必需)	string	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.NA {ITEM.ID}，{ITEM.KEY}。
value_type (必需)	integer	监控项信息的类型。取值范围: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.

属性类	描述	
allow_traps	integer	HTTP agent 监控项字段。允许和 trap-per 监控项一样的填充值。 0 - *(默认值)* 不允许接收传入数据。 1 - 允许接收传入数据。

属性类	描述	
authtype	integer	仅在SSH agent监控项或HTTP agent监控项中使用。 SSH agent认证方法取值范围: 0 - *(默认值)* password; 1 - public key. HTTP agent认证方法取值范围 0 - *(默认值)* none 1 - basic 2 - NTLM 3 - Kerberos

属性类	描述	
description	string	监控项说明。
error	string	*(只读)* 当更新监控项出错时的错误文本。
flags	integer	*(只读)* 项目的起源。 取值范围: 0 - 普通监控项; 4 - 自动发现监控项.

属性类	描述	
follow_redirects	integer	HTTP agent 监控项字段，合并数据时跟随重定向。 0 - 不进行重定向. 1 - *(默认值)* 进行重定向.

属性类	描述	

headers	object	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。
		
		事例:
		{
		”User-Agent”: ”Zab-bix”
		}

属性类	描述	
history	string	一个历史数据被保存的时长的时间单位。接受用户宏。 默认值: 90d.
http_proxy	string	HTTP agent 监控项字段。HTTP(S) 代理连接字符串。

属性类	描述	
inventory_link	integer	监控项填充的主机资产的ID。 请参考host inventory page获取支持的主机清单字段及其id。 默认值: 0.
ipmi_sensor	string	IPMI传感器。仅用于IPMI监控项。

属性类	描述	
jmx_endpoint	string	JMX agent 自定义的连接字符串。 默认值: service:jmx:rmi:///jndi/r
lastclock	timestamp	*(只读)* 监控项最后被更新的时间。 此属性将只返回ZBX_HISTORY_PERIOD 配置的周期值。

属性类	描述	
lastns	integer	*(只读)* 监控项最后被更新的纳秒。 此属性将只返回ZBX_HISTORY_PERIOD配置的周期值。
lastvalue	string	*(只读)* 监控项最新的值。 此属性将只返回ZBX_HISTORY_PERIOD配置的周期值。

属性类	描述	
logtimefmt	string	日志条目的时间格式。仅用于日志监控项。
master_itemid	integer	主监控项ID。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。 要求依赖项。

属性类	描述	
mtime	timestamp	上次更新所监视日志文件的时间。仅用于日志项。
output_format	integer	HTTP agent监控项字段。返回数据应被转换成JSON。 0 - *(默认值)* 存储raw. 1 - 转成JSON.

属性类	描述	
params	string	取决于监控项类型的附加参数： - SSH、Telnet 项执行脚本 - SQL 查询数据库监控项; - 计算项目公式.

属性类	描述	
password	string	认证的密码。用于simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. 当JMX使用用户名时，用户名应该和密码一起指定，或者两个属性都应该留空。

属性类	描述	
port	string	监控项监控的端口。仅用于SMNP监控项。
post_type	integer	HTTP agent字段。存储在post属性的post的数据类型。 0 - *(默认值)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent字段。HTTP(S)请求报文。仅用于post_type。

属性类	描述	
prevvalue	string	*(只读)* 监控项的前一个值。 此属性将只返回ZBX_HISTORY_PERIOD配置的周期值。
privatekey	string	私钥文件名。
publickey	string	公钥的文件名。
query_fields	array	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。

属性类	描述	
request_method	integer	HTTP agent 监控项字段。请求方法的类型。 0 - GET 1 - *(默认值)* POST 2 - PUT 3 - HEAD

属性类	描述	
retrieve_mode	integer	HTTP agent 监控项字段。被存储的响应的部分。 0 - *(默认值)* Body. 1 - Headers. 2 - Body 和 HTTP Headers 将存储. 对于 request_method 头，只允许值为 1
snmp_oid	string	SNMP OID.

属性类	描述	
ssl_cert_file	string	HTTP agent 监控项字段。公共 SSL 秘钥的文件路径。
ssl_key_file	string	HTTP agent 监控项字段。私有 SLL 秘钥的文件路径。
ssl_key_password	string	HTTP agent 监控项字段。SSL 秘钥的文件密码。

属性类	描述	
state	integer	(只读)监控项状态. 取值范围: 0 - (默认值)标准; 1 - 不支持.
status	integer	监控项状态. 取值范围: 0 - (默认值)启用; 1 - 禁用.

属性类	描述	
status_codes	string	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。 事例: 200,200-{$M},{$M},200-400
templateid	string	(只读) 父模板的 ID。

属性类	描述	
timeout	string	HTTP agent 监控项字段。监控项数据轮询超时时间。支持用户宏。 默认值: 3s 最大值: 60s
trapper_hosts	string	接受的主机。仅用于 trapper 监控项或者 HTTP agent 监控项。

属性类	描述	
trends	string	时间单位，数据数据被保存的时间长度。也接受用户宏。 默认值: 365d.
units	string	Value units. 值的单位。

属性类	描述	
username	string	认证的用户名。用于simple check, SSH, Telnet, database monitor, JMX and HTTP agent监控项。 要求提供。\\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	关联映射值的ID。

属性类	描述	
verify_host	integer	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 - *(默认值)* 不验证. 1 - 验证.

属性类	描述	
verify_peer	integer	HTTP agent 字段。验证主机的合法性。 0 - *(默认值)* 不验证. 1 - 验证.

监控项预处理

监控项预处理对象有如下属性。

属性类	说明	
type (必需)	integer	预处理选项类型。 取值范围: 1 -自定义乘数; 2 -右纵倾; 3 -左纵倾; 4 -修剪; 5 -正则表达式匹配; 6 -布尔值到小数; 7 -八进制到十进制; 8—十六进制到十进制; 9 -简单的改

属性类	说明	
params (必需)	string	预处理选项使用的其他参数。多个参数以 LF (\n) 字符分隔

属性类	说明	
error_handler (必需)	integer0	预处理步骤失败时使用的动作类型。 可能的值: -错误信息被Zab-bix服务器设置; 1-弃值; 2-设置自定义值; 3-设置自定义错误信息。

属性类	说明	
error_handler_params (必需)	string 如	错误处理程序参数。用于error_handle。error_handle为0或1，则必须为空。当error_handle为2时可以为空。如果er-ror_handler为3，则不能为空。

每种预处理类型都支持以下参数和错误处理程序

预处理类型名称	参数 1	参数 2	参数 3 支持的错	处理程序	
1	Custom multiplier	number 备注 1, 6		0	1, 2, 3
2	Right trim	list of characters 备注 2			
3	Left trim	list of characters 备注 2			
4	Trim	list of characters 备注 2			
5	Regular expression	pattern 备注 3 o	tput 备注 2	0,	, 2, 3
6	Boolean to decimal				0, 1, 2, 3
7	Octal to decimal				0, 1, 2, 3
8	Hexadecimal to decimal				0, 1, 2, 3
9	Simple change				0, 1, 2, 3
10	Change per second				0, 1, 2, 3
11	XML XPath	path 备注 4		0	1, 2, 3
12	JSONPath	path 备注 4		0	1, 2, 3
13	In range	min 备注 1，6 ma	备注 1，6	0, 1,	2, 3

预处理类型名称	参数 1	参数 2	参数 3 支持的错	处理程序	
14	Matches regular expression	pattern 备注 3		0	1, 2, 3
15	Does not match regular expression	pattern 备注 3		0	1, 2, 3
16	Check for error in JSON	path 备注 4		0	1, 2, 3
17	Check for error in XML	path 备注 4		0	1, 2, 3
18	Check for error using regular expression	pattern 备注 3 o	tput 备注 2	0,	, 2, 3
19	Discard unchanged				
20	Discard unchanged with heartbeat	seconds 备注 5，6			
21	JavaScript	script 备注 2			
22	Prometheus pattern	pattern6, 7	output 备注 6，8	0,	1, 2, 3
23	Prometheus to JSON	pattern6, 7	0, 1, 2, 3		
24	CSV to JSON	character 备注 2 c	aracter 备注 2 0,1	0,	, 2, 3
25	Replace search	string 备注 2 r	placement 备注 2		

备注
1 整数或浮点数
2 字符串
3 正则表达式
4 JSONPath 或 XML XPath
5 正整数 (支持时间后缀，如 30s, 1m, 2h, 1d)
6 用户宏、LLD 宏
7 `<metric name>{<label name>= " <label value> " , …}== <value>`。每个 Prometheus 模式组件 (度量值、标签名称、标签值和度量值) 可以是 user 宏或 LLD 宏。
8 Prometheus 的输出格式如下:`<label name>`。

创建

说明

`object item.create(object/array items)`

此方法允许创建监控项。

> **Note:**
> WEB 监控项不能通过 Zabbix API 创建。

参数

`(object/array)` 要创建的监控项。

另外见standard item properties，此方法接受如下参数。

属性类	说明	
applications	array	要添加到监控项的应用 IDs。
preprocessing	array	监控项预处理选项。

返回值

`(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。

示例

创建一个监控项

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 [”609”, ”610”]。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24758"
        ]
    },
    "id": 1
}
```

创建一个主机清单监控项

创建一个 Zabbix agent 监控项填充主机的”OS” 清单字段。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 1
}
```

创建带有预处理的监控项

使用自定义乘法器创建监控项。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Device uptime",
        "key_": "sysUpTime",
        "hostid": "11312",
        "type": 4,
        "snmp_oid": "SNMPv2-MIB::sysUpTime.0",
        "value_type": 1,
        "delay": "60s",
        "units": "uptime",
        "interfaceid": "1156",
        "preprocessing": [
            {
                "type": "1",
                "params": "0.01",
                "error_handler": "1",
                "error_handler_params": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44210"
        ]
    },
    "id": 1
}
```

创建依赖监控项

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。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
```

```
        ]
    },
    "id": 1
}
```

创建 HTTP agent 监控项

创建带有 JSON 响应预处理的 POST 请求的方法监控项。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

来源

CItem::create() in *frontends/php/include/classes/api/services/CItem.php*.

删除

Description 说明

`object item.delete(array itemIds)`

此方法允许删除监控项。

> **Note:**
> WEB 监控项不能通过 Zabbix API 删除。

参数

(array) 要删除的监控下的 IDs。

返回值

(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。

示例

删除多个监控项

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。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.delete",
    "params": [
        "22982",
        "22986"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22982",
            "22986"
        ]
    },
    "id": 1
}
```

来源

CItem::delete() in *frontends/php/include/classes/api/services/CItem.php*.

更新

说明

`object item.update(object/array items)`

此方法允许更新已存在的监控项。

> **Note:**
> WEB 监控项不能通过 Zabbix API 更新。

参数

(object/array) 要更新的监控项的属性。

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。

另外见standard item properties，此方法接受如下参数。

参数类	说明	
applications	array	要替换当前应用的应用的 ID。
preprocessing	array	要替换的当前监控项预处理选项。

返回值

(object) 在 itemids 属性下返回已被更新的监控项的对象的 IDs。

示例

启用一个监控项

启用一个监控项就是设置他的 status 属性为"0"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 1
}
```

更新依赖监控项

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。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "name": "Dependent item updated name",
        "master_itemid": "25562",
        "itemid": "189019"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189019"
        ]
    },
    "id": 1
}
```

更新 HTTP agent 监控项

启用监控项的 trapping 值。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "allow_traps": "1"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ]
    },
    "id": 1
}
```

来源

CItem::update() in *frontends/php/include/classes/api/services/CItem.php*.

Source

CItem::update() in *ui/include/classes/api/services/CItem.php*.

获取

说明

`integer/array item.get(object parameters)`

此方法允许根据给定的参数获取监控项。

参数

`(object)` 参数定义期望输出。

此方法支持如下参数。

参数类	说明	
itemids	string/array	返回给定IDs的监控项。

参数类	说明	
groupids	string/array	返回属于给定组的主机的监控项。
templateids	string/array	返回属于给定模板的监控项。
hostids	string/array	返回属于给定主机的监控项。
proxyids	string/array	返回被给定代理监控的监控项。

参数类	说明	
interfaceids	string/array	返回使用给定主机接口的监控项。
graphids	string/array	返回在给定图表中使用的监控项。
triggerids	string/array	返回给定触发器所使用的监控项。
applicationids	string/array	返回属于给定应用的监控项。

参数类	说明	
webitems	flag	返回结果中包含web监控项。
inherited	boolean	如果设为`true`，返回继承自某个模板的监控项。
templated	boolean	如果设为`true`，返回属于某个模板的监控项。

参数类	说明	
monitored	boolean	如果设为 `true`，返回属于已监控主机的已启用的监控项。
group	string	返回属于给定组名的监控项。
host	string	返回给定主机名的监控项。
application	string	返回属于给定应用名的监控项。

参数类	说明	
with_triggers	boolean	如果设为 `true`，返回在触发器中使用的监控项。
selectHosts	query	在 `host` 属性中已数组的形式返回监控项所属的主机。
selectInterfaces	query	在 `interfaces` 属性中返回在主机接口中使用的监控项。

参数类	说明	
selectTriggers	query	在 triggers 属性中返回使用该监控项的触发器。 支持 count.
selectGraphs	query	在 graphs 属性中返回包含该监控项的图表。 支持 count.
selectApplications	query	在 application 属性中返回监控项所属的应用。

参数类	说明	
selectDiscoveryRule	query	在discoveryRul属性中返回创建该监控项的LLD规则。

参数类	说明	
selectItemDiscovery	query	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。 它有如下属性： itemdiscovery - (string) ID of the item discovery; itemid - (string) ID of the discovered item; parent_itemid - (string) ID

参数类	说明	
selectPreprocessing	query	返回“预处理”属性中的项目预处理选项。\\它具有以下特性: type - (string) 预处理选项类型: 1 -自定义乘数; 2 -右纵倾; 3 -左纵倾; 4 -修剪; 5 -正则表达式匹配; 6 -布尔值到小数;

参数类	说明	
filter	object	返回紧缺匹配给定筛选条件的结果。 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。 支持附加筛选条件： `host` - 监控项所属主机的技术名称。

参数类	说明	
limitSelects	integer	限制子查询所返回的结果的数量。 应用到如下子查询： selectGraphs - 结果排序 name; selectTrigge - 结果排序 description.

参数类	说明	
sortfield	string/array	根据给定的属性对结果排序。 可能的值是: `itemid`, `name`, `key_`, `delay`, `history`, `trends`, `type` and `status`.
countOutput	boolean	在参考说明页面中详细描述了这些对于所有“get”方法都是通用的参数。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	

参数类	说明
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	boolean

返回值

(integer/array) 返回:

- 对象数组;
- 检索对象的计数 (如果使用了 “countOutput” 参数)。

示例

根据 key 查找监控项

从 ID 为”10084” 的主机获取 key 带有”system” 的监控项，并以名称排序。

请求:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23298",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Context switches per second",
            "key_": "system.cpu.switches",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "2552",
            "lastclock": "1351090998",
            "prevvalue": "2641",
            "state": "0",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "sps",
            "error": "",
            "logtimefmt": "",
            "templateid": "22680",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "lastns": "564054253",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "lifetime": "0s",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
```

```
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        },
        {
            "itemid": "23299",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "CPU $2 time",
            "key_": "system.cpu.util[,idle]",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "86.031879",
            "lastclock": "1351090999",
            "prevvalue": "85.306944",
            "state": "0",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "%",
            "error": "",
            "logtimefmt": "",
            "templateid": "17354",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "lastns": "564256864",
            "flags": "0",
            "interfaceid": "1",
            "description": "The time the CPU has spent doing nothing.",
            "inventory_link": "0",
            "lifetime": "0s",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
```

```
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        },
        {
            "itemid": "23300",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "CPU $2 time",
            "key_": "system.cpu.util[,interrupt]",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "0.008389",
            "lastclock": "1351091000",
            "prevvalue": "0.000000",
            "state": "0",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "%",
            "error": "",
            "logtimefmt": "",
            "templateid": "22671",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "lastns": "564661387",
            "flags": "0",
            "interfaceid": "1",
            "description": "The amount of time the CPU has been servicing hardware interrupts.",
            "inventory_link": "0",
            "lifetime": "0s",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
```

```
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 1
}
```

根据 key 查找依赖监控项

从 ID 未”10116” 的主机中获取 key 名包含”apache” 的依赖监控项。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10116",
        "search": {
            "key_": "apache"
        },
        "filter": {
            "type": "18"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23298",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Context switches per second",
            "key_": "system.cpu.switches",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "2552",
            "lastclock": "1351090998",
            "prevvalue": "2641",
            "state": "0",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "sps",
            "error": "",
            "logtimefmt": "",
            "templateid": "22680",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
```

```
        "privatekey": "",
        "lastns": "564054253",
        "flags": "0",
        "interfaceid": "1",
        "description": "",
        "inventory_link": "0",
        "lifetime": "0s",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "0",
        "output_format": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0"
    },
    {
        "itemid": "23299",
        "type": "0",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU $2 time",
        "key_": "system.cpu.util[,idle]",
        "delay": "1m",
        "history": "7d",
        "trends": "365d",
        "lastvalue": "86.031879",
        "lastclock": "1351090999",
        "prevvalue": "85.306944",
        "state": "0",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "error": "",
        "logtimefmt": "",
        "templateid": "17354",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "lastns": "564256864",
        "flags": "0",
        "interfaceid": "1",
        "description": "The time the CPU has spent doing nothing.",
```

```
        "inventory_link": "0",
        "lifetime": "0s",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "0",
        "output_format": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0"
    },
    {
        "itemid": "23300",
        "type": "0",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU $2 time",
        "key_": "system.cpu.util[,interrupt]",
        "history": "7d",
        "trends": "365d",
        "lastvalue": "0.008389",
        "lastclock": "1351091000",
        "prevvalue": "0.000000",
        "state": "0",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "error": "",
        "logtimefmt": "",
        "templateid": "22671",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "lastns": "564661387",
        "flags": "0",
        "interfaceid": "1",
        "description": "The amount of time the CPU has been servicing hardware interrupts.",
        "inventory_link": "0",
        "lifetime": "0s",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
```

```
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 1
}
```

查找 HTTP agent 监控项

根据定义的主机 id 来查找带有 XML post 报文类型的监控项。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "hostids": "10255",
        "filter": {
            "type": "19",
            "post_type": "3"
        }
    },
    "id": 3,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28252",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10255",
            "name": "template item",
            "key_": "ti",
            "delay": "30s",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
```

```
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "localhost",
            "query_fields": [
                {
                    "mode": "xml"
                }
            ],
            "posts": "<body>\r\n<![CDATA[{$MACRO}<foo></bar>]]>\r\n</body>",
            "status_codes": "200",
            "follow_redirects": "0",
            "post_type": "3",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "1",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 3
}
```


使用预处理规则检索项

从 ID 为 “10254” 的主机重新获取所有项目及其预处理规则。

请求:


```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": ["itemid", "name", "key_"],
        "selectPreprocessing": "extend",
        "hostids": "10254"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
```

```
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemid": "23865",
        "name": "http agent example JSON",
        "key_": "json",
        "preprocessing": [
            {
                "type": "12",
                "params": "$.random",
                "error_handler": "1",
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

参考

- 应用集
- 发现规则
- 图形
- 主机
- 主机接口
- 触发器

来源

CItem::get() in *frontends/php/include/classes/api/services/CItem.php*.

26. 监控项原型

该类被设计用来处理监控项原型。

对象引用

- 监控项原型

可用方法：

- itemprototype.create - 创建新监控项原型
- itemprototype.delete - 删除监控项原型
- itemprototype.get - 获取监控项原型
- itemprototype.update - 更新监控项原型

> 监控项原型对象

如下对象与 `itemprototype` API 直接相关。

监控项原型

监控项原型有如下属性。

属性类	描述	
itemid	string	*(只读)* 监控项原型的ID。

属性类	描述	
delay (必需)	string 可	更新监控项的时间间隔。接受具有后缀(30s,1m,2h,1d)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自定义时间间

属性类	描述	
hostid (必须)	string	监控项原型所属的主机的ID。对于更新操作，这个字段是只读.
ruleid (必须)	string	监控项所属的LLD(低级别发现)的ID对于更新操作，这个字段是只读.

属性类	描述	
interfaceid (必须)	string	监控项原型的主机的接口的ID。仅用于主机监控项原型。可选 Zabbix agent (active), Zabbix internal, Zabbix trapper, Dependent item, Zabbix aggregate, database monitor and 计算型监控项.

属性类	描述	
key_ (必须)	string	监控项原型的键。
name (必须)	string	监控项原型的名称。

属性类	描述	
type (必须)	integer	监控项原型的类型。 可能的值： 0 - Zabbix agent; 2 - Zabbix trapper; 3 - simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - cal-

属性类	描述	
url (必须)	string	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	监控项原型信息类型。 可能的值： 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.

属性类	描述	
allow_traps	integer	HTTP agent 监控项原型字段。允许像 trap-per 监控项一样的填充值。 0 - (默认) 不允许接收传入数据。 1 - 允许接收传入数据。

属性类	描述	
authtype	integer	仅用于SSH agent监控项原型或者HTTP agent监控项原型。 SSH agent认证方法可能的值： 0 - *(默认)* password; 1 - public key. HTTP agent认证方式可选值: 0 - *(default)* none 1 - basic 2 - NTLM 3 - Kerberos

属性类	描述	
description	string	监控项原型的说明。
follow_redirects	integer	HTTP agent 监控项原型字段。当合并数据时跟随重定向。 0 - 不要遵循重定向。 1 - (默认) 遵循重定向。

属性类	描述	

headers	object	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。
		
		示例：
		{
		”User-Agent”: ”Zab-bix”
		}

history	string	历史数据应被保存的时间。接受用户宏和 LLD 宏。 默认: 90d.

属性类	描述	
http_proxy	string	HTTP agent 监控项原型字段。HTTP(S) 代理连接字符串。
ipmi_sensor	string	IPMI 传感器，仅用于 IPMI 监控项原型。
jmx_endpoint	string	JMX agent 自定义的连接字符串。 默认值: service:jmx:rmi:///jndi/r

属性类	描述	
logtimefmt	string	日志条目的时间格式。仅用于日志监控项原型。

属性类	描述	
master_itemid	integer	主监控项ID。递归3层递归3层依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。\\依赖项要求。

属性类	描述	
output_format	integer	HTTP agent 监控项原型字段。返回数据应被转换为 JSON 格式。 0 - *(默认)* Store raw. 1 - Con-vert to JSON.

属性类	描述	
params	string	附加参数依赖于监控项原型的类型： - SSH 和 Telnet 项目原型执行脚本； - 数据库监控项目原型的 SQL 查询 - 计算监控项目原型的公式。

属性类	描述	
password	string	认证的密码。用于simple check, SSH, Telnet, database monitor, JMX and HTTP agent监控项原型。
port	string	监控的监控项原型的端口。仅用于SNMP监控项原型。

属性类	描述	
post_type	integer	HTTP agent 监控项原型字段。存储在 post 属性的 post 数据体的类型。 0 - (默认) Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent 监控项原型字段。HTTP(S) 请求报文数据。用于 post_data。
privatekey	string	私钥文件名。

属性类	描述	
publickey	string	公钥文件名。
query_fields	array	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 监控项原型字段。请求方法类型。 0 - GET 1 - *(默认)* POST 2 - PUT 3 - HEAD

属性类	描述	
retrieve_mode	integer	HTTP agent 监控项原型字段。指定那一部分的响应应该被存储。 0 - (默认) Body. 1 - Headers. 2 - HTTP 正文和 HTTP 标题都将被存储 对于 request_method 头，只允许值为 1。

属性类	描述	
snmp_community	string	SNMP 共同体秘钥. 仅用于 SNMPv1 和 SNMPv2 监控项原型。
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent 监控项原型字段。公共 SSL key 文件路径。
ssl_key_file	string	HTTP agent 监控项原型字段。私有 SSL key 文件路径。

属性类	描述	
ssl_key_password	string	HTTP agent 监控项原型字段。SSL key 文件的密码。
status	integer	监控项原型的状态。 可能的值。0 - *(默认)* 启用监控项目原型。1 - 禁用监控项目原型。3 - 不支持的监控项目原型。

属性类	描述	
status_codes	string	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	(只读) 父模板的监控项原型的 ID。

属性类	描述	
timeout	string	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。 默认: 3s 最大值: 60s
trapper_hosts	string	允许主机。用于 trapp 监控项原型或者 HTTP 监控项原型。

属性类	描述	
trends	string	趋势数据被保存的时间。也接受用户宏和LLD宏。 默认: 365d.
units	string	单位

属性类	描述	
username	string	认证的用户名。 用于simple check, SSH, Telnet, database monitor, JMX and HTTP agent监控项原型。 SSH和Telnet监控项原型要求。
valuemapid	string	相关值映射的ID。

属性类	描述	
verify_host	integer	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 - *(默认)* 不验证. 1 - 验证.

属性类	描述	
verify_peer	integer	HTTP agent 监控项原型字段。主机合法性认证。 0 - (默认) 不验证. 1 - 验证.

属性类	描述	
discover	integer	项目原型发现状态。可能的值: 0 - (默认) 新项目将被发现; 1 - 新的物品将不会被发现，现有的物品将被标记为丢失。

监控项预处理

监控项预处理对象有如下属性。

属性类	说明	
type (必需)	integer	预处理选项类型。 取值范围: 1 -自定义乘数; 2 -右纵倾; 3 -左纵倾; 4 -修剪; 5 -正则表达式匹配; 6 -布尔值到小数; 7 -八进制到十进制; 8—十六进制到十进制; 9 -简单的改

属性类	说明	
params (必需)	string	预处理选项使用的其他参数。多个参数以 LF (\n) 字符分隔

属性类	说明	
error_handler (必需)	integer0	预处理步骤失败时使用的动作类型。可能的值: -错误信息被Zab-bix服务器设置; 1-弃值; 2-设置自定义值; 3-设置自定义错误信息。

属性类	说明	
error_handler_params (必需)	string 如	错误处理程序参数。用于error_handle。error_handle为0或1，则必须为空。当error_handle为2时可以为空。如果er-ror_handler为3，则不能为空。

每种预处理类型都支持以下参数和错误处理程序

预处理类型名称	参数 1	参数 2	参数 3 支持的错	处理程序	
1	Custom multiplier	number 备注 1, 6		0	1, 2, 3
2	Right trim	list of characters 备注 2			
3	Left trim	list of characters 备注 2			
4	Trim	list of characters 备注 2			
5	Regular expression	pattern 备注 3 o	tput 备注 2	0,	, 2, 3
6	Boolean to decimal				0, 1, 2, 3
7	Octal to decimal				0, 1, 2, 3
8	Hexadecimal to decimal				0, 1, 2, 3
9	Simple change				0, 1, 2, 3
10	Change per second				0, 1, 2, 3
11	XML XPath	path 备注 4		0	1, 2, 3
12	JSONPath	path 备注 4		0	1, 2, 3
13	In range	min 备注 1，6 ma	备注 1，6	0, 1,	2, 3

预处理类型名称	参数 1	参数 2	参数 3 支持的错	处理程序	
14	Matches regular expression	pattern 备注 3		0	1, 2, 3
15	Does not match regular expression	pattern 备注 3		0	1, 2, 3
16	Check for error in JSON	path 备注 4		0	1, 2, 3
17	Check for error in XML	path 备注 4		0	1, 2, 3
18	Check for error using regular expression	pattern 备注 3 o	tput 备注 2	0,	, 2, 3
19	Discard unchanged				
20	Discard unchanged with heartbeat	seconds 备注 5，6			
21	JavaScript	script 备注 2			
22	Prometheus pattern	pattern6, 7	output 备注 6，8	0,	1, 2, 3
23	Prometheus to JSON	pattern6, 7	0, 1, 2, 3		
24	CSV to JSON	character 备注 2 c	aracter 备注 2 0,1	0,	, 2, 3
25	Replace search	string 备注 2 r	placement 备注 2		

备注
1 整数或浮点数
2 字符串
3 正则表达式
4 JSONPath 或 XML XPath
5 正整数 (支持时间后缀，如 30s, 1m, 2h, 1d)
6 用户宏、LLD 宏
7 <metric name>{<label name>= " <label value> "，…}== <value>。每个 Prometheus 模式组件 (度量值、标签名称、标签值和度量值) 可以是 user 宏或 LLD 宏。
8 Prometheus 的输出格式如下:<label name>。

创建

说明

object itemprototype.create(object/array itemPrototypes)

此方法用于创建新的监控项原型。

参数

(object/array) 需要创建的监控项原型。

除标准项原型属性外，该方法还接受以下参数。

属性类	描述	
ruleid (必须)	string	该项所属的 LLD 规则的 ID。
applications	array	要分配给自动发现监控项的应用程序的 ID。
applicationPrototypes	array	要分配给监控项原型的应用程序原型的名称。
preprocessing	array	预处理选项。

返回值

(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。(object) 返回一个对象，该对象包含在” itemids ” 属性下创建的监控项原型的 id。返回的 id 的顺序与传递的监控项原型的顺序相匹配。

示例

创建一个监控项原型

创建一个监控项原型去监控自动发现的文件系统上的磁盘空间。发现监控项应该每 30 秒更新数字化的 Zabbix agent 监控项。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27666"
        ]
    },
    "id": 1
}
```

创建一个预处理的监控项原型

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。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Incoming network traffic on {#IFNAME}",
        "key_": "net.if.in[{#IFNAME}]",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
        "units": "bps",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": "10",
                "params": "",
                "error_handler": "0",
                "error_handler_params": ""
            },
            {
                "type": "1",
                "params": "8",
                "error_handler": "2",
                "error_handler_params": "10"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

创建依赖监控项原型

创建依赖监控项原型

为 ID 为 44211 的主监控项原型创建一个依赖监控项原型。只有在同一个主机的 (模板/LLD 发现规则) 依赖才可以被接受，因此主监控项原型和依赖监控项原型应该拥有相同的 hostid 和 ruleid。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44212"
        ]
    },
    "id": 1
}
```

创建 HTTP agent 监控项原型

创建带有 URL 使用用户宏，查询字段和自定义选项的 item prototype。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

来源

CItemPrototype::create() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

删除

说明

`object itemprototype.delete(array itemPrototypeIds)`

此方法允许删除监控项原型。

参数

(array) 要删除的监控项原型 IDs.

返回值

(object) prototypeids 属性下在返回一个带有被删除的监控项原型的 IDs.

示例

删除多个监控项原型

删除 2 个监控项原型。
如果主监控项或者监控项原型被删除，依赖其的监控项原型也会被删除。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.delete",
    "params": [
        "27352",
        "27356"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "prototypeids": [
```

```
            "27352",
            "27356"
        ]
    },
    "id": 1
}
```

来源

CItemPrototype::delete() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

更新

说明

`object itemprototype.update(object/array itemPrototypes)`

此方法允许更新存在的监控项原型。

参数

`(object/array)` 监控项原型要更新的属性。

监控项原型的 `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。

除了标准监控项原型属性之外，该方法还接受以下参数。

属性类	描述	
applications	array	要替换当前应用程序的应用程序的 IDS。
applicationPrototypes	array	要替换当前应用程序原型的应用程序原型名称。
preprocessing	array	要替换当前预处理选项的监控项原型的预处理选项。

返回值

`(object)` 在 `itemids` 属性中返回一个包含已被更新的监控项原型的 IDs 对象。

示例

改变监控项原型的接口

改变将被用于发现监控项的主机接口。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "27428",
        "interfaceid": "132"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27428"
        ]
    },
    "id": 1
}
```

更新依赖的监控项原型

使用新的主监控项原型 ID 更新依赖监控项原型。只允许依赖于同一主机 (模板/发现规则)，因此主监控项和依赖监控项应该具有相同的 hostid 和 ruleid。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "master_itemid": "25570",
        "itemid": "189030"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189030"
        ]
    },
    "id": 1
}
```

更新 HTTP agent 监控项原型

改变查询字段并移除所有自定义请求头。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid":"28305",
        "query_fields": [
            {
                "random": "qwertyuiopasdfghjklzxcvbnm"
            }
        ],
        "headers": []
    }
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

来源

CItemPrototype::update() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

Source

CItemPrototype::update() in *ui/include/classes/api/services/CItemPrototype.php*.

获取

说明

`integer/array itemprototype.get(object parameters)`

此方法可以根据提供的参数获取监控项原型。

参数

`(object)` 参数定义期望输出

此方法提供以下参数。

属性类	描述	
discoveryids	string/array	只返回属于给定LLD规则的监控项原型。
graphids	string/array	只返回在给定图标原型中使用的监控项原型。

属性类	描述	
hostids	string/array	只返回属于给定host的监控项原型。
inherited	boolean	如果设为”true”，返回继承自某个模板的监控项原型。
itemids	string/array	返回给定IDS的监控项原型。

属性类	描述	
monitored	boolean	如果设为”true”，只返回已启动的属于已监控主机的监控项原型。
templated	boolean	如果设为”true”，只发挥属于给定模板的监控项原型。
templateids	string/array	只返回属于给定模板的监控项原型。

属性类	描述	
triggerids	string/array	只返回使用在给定触发器原型的监控项原型。
selectApplications	query	在applications属性中返回监控项原型所属的应用。
selectApplicationPrototypes	query	只返回被连接到applicationP属性中的监控项原型的应用原型。

属性类	描述	
selectDiscoveryRule	query	在discoveryRul属性中返回图表原型所属的低级发现规则。
selectGraphs	query	在graphs属性中返回被监控项原型使用的图形原型。 支持count。

属性类	描述	
selectHosts	query	在hosts属性中以数组的形式返回监控项原型所属的host。
selectTriggers	query	在triggers属性中返回监控项原型被使用的触发器原型。 支持count。

属性类	描述	
selectPreprocessing	query	返回“预处理”属性中的项目预处理选项。\\它具有以下特性: type - (string) 预处理选项类型: 1 -自定义乘数; 2 -右纵倾; 3 -左纵倾; 4 -修剪; 5 -正则表达式匹配; 6 -布尔值到小数;

属性类	描述	
filter	object	只返回精确匹配筛选条件的结果。 接受一个数组，数组键为属性名称，值为单个值或者数组。 支持可选筛选条件： host - 监控项原型所属的主机的技术名称。

属性类	描述	
limitSelects	integer	限制子选择返回的记录数。 应用于如下子选择： selectGraphs - 结果将按 name 排序; selectTrigge 结果将按 description 排序。
sortfield	string/array	根据给定的属性排序 可能的值有：itemid, name, key_, delay, type 和 status.

属性类	描述	
countOutput	boolean	这些参数对于所有在参考说明详细描述的"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	

返回值

(integer/array) 返回:

- 对象数组；
- 已获取到的对象的数量，如果 countOutput 参数被使用。

示例

获取监控项原型

从 LLD 规则中获取所有监控项原型请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
        {
            "itemid": "23077",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10079",
            "name": "Incoming network traffic on en0",
            "key_": "net.if.in[en0]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "bps",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0",
            "discover": "0"
        },
        {
            "itemid": "10010",
            "type": "0",
```

```
            "snmp_oid": "",
            "hostid": "10001",
            "name": "Processor load (1 min average per core)",
            "key_": "system.cpu.load[percpu,avg1]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "The processor load is calculated as system CPU load divided by number of CPU c
            "inventory_link": "0",
            "lifetime": "0",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0",
            "discover": "0"
        }
    ],
    "id": 1
}
```


查找依赖的监控项

为 ID 为”25545” 的 item 查找一个赖的 item。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "filter": {
            "type": "18",
            "master_itemid": "25545"
        },
        "limit": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25547",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Seconds",
            "key_": "apache.status.uptime.seconds",
            "delay": "0",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
```

```
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0",
            "discover": "0"
        }
    ],
    "id": 1
}
```

查找 HTTP agent 监控项原型

为请求方法头定义的 host id 查找 HTTP agent 监控项原型。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "hostids": "10254",
        "filter": {
            "type": "19",
            "request_method": "3"
        }
    },
    "id": 17,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28257",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10254",
            "name": "discovered",
            "key_": "item[{#INAME}]",
            "delay": "{#IUPDATE}",
            "history": "90d",
            "trends": "30d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "28255",
```

```
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "2",
            "interfaceid": "2",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "{#IURL}",
            "query_fields": [],
            "posts": "",
            "status_codes": "",
            "follow_redirects": "0",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0"
        }
    ],
    "id": 17
}
```


参考

- 应用集
- 主机
- 图形原型
- 触发器原型

来源

CItemPrototype::get() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

27.LLD 发现规则

此类设计用于低级发现规则。

对象参考:

- LLD 规则

Available methods:

- discoveryrule.copy - 复制 LLD 规则
- discoveryrule.create - 创建新的 LLD 规则

- discoveryrule.delete - 删除 LLD 规则
- discoveryrule.get - 检索 LLD 规则
- discoveryrule.update - 更新 LLD 规则

> LLD 规则对象

下面的对象直接关联到 `discoveryrule`（发现规则）API。

LLD 规则

低级发现规则对象有如下属性。

属性类	说明	
itemid	string	*(只读)* ID of the LLD rule. LLD 规则的 ID

属性类	说明	
delay (必须)	string	LLD 规则的更新时间间隔。接受带后缀的秒或时间单位，包含或不包含一个或多个自定义时间间隔，它由灵活的时间间隔和调度时间间隔作为序列化字符串组成。也接受用户宏。灵活的间隔可以写成用正斜杠分隔的两个宏。间隔用分号分隔

hostid (必须)	string	LLD 规则所属的 Host 的 ID。

interfaceid (必须)	string	LLD 规则的主机接口 ID。仅用于主机 LLD 规则。 Zabbix agent (active), Zabbix internal, Zabbix trapper and 数据库监控 LLD 规则的可选参数。
key_ (必须)	string	LLD 规则键。
name (必须)	string	LLD 规则名称。
type (必须)	integer	LLD 规则类型。 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; 19 -SNMP agent;
url (必须)	string	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 规则字段。在陷阱监控项类型中也允许填充值 0 - *(默认)* 不允许接受输入数据 1 - 允许输入数据
authtype	integer	只能被 SSH agent 或 HTTP agent 使用 \\SSH agent 认证方法可能的值： 0 - *(默认)* 密码; 1 - 公钥. HTTP agent 认证方法可能的值： 0 - *(默认)* none 1 - basic 2 - NTLM
description	string	LLD 规则说明。
error	string	*(只读)* 如果更新 LLD 规则出问题时的错误文本。
follow_redirects	integer	HTTP agentLLD 规则字段。当合并数据时进行重定向。 0 - 不跟随重定向。 1 - *(默认)* 跟随重定向。
headers	object	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。 事例: { ”User-Agent”: ”Zabbix” }
http_proxy	string	HTTP agent LLD 规则字段。HTTP(S) proxy 连接字符串。

ipmi_sensor	string	IPMI sensor. 只用于 IPMILLD 规则
jmx_endpoint	string	JMX agent 自定义连接字符串。 默认值： service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PO
lifetime	string	不在发现的时间周期内监控项将被删除。 接受秒、带后缀和用户宏的时间单位。 Default: 30d.
master_itemid	integer	主条目 ID。允许递归至多 3 个相关项，且相关项的最大计数为 999。发现规则不能是其他发现规则的主项。相关项必需。
output_format	integer	HTTP agent LLD 规则字段。应返回传递给 JSON. 0 - *(默认)* Store raw. 1 - Convert to JSON.
params	string	依赖于 LLD 规则类型的其他参数： - 为 SSH 何 Telnet LLD 规则执行脚本； - 数据库监控 LLD 规则的 SQL 查询； - 计算类的 LLD 规则公式。
password	string	认证密码。用于 simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD 规则。
port	string	LLD 规则使用的端口。仅 SNMP LLD 规则使用
post_type	integer	HTTP agent LLD 规则字段。post 数据 body 部分存储在 posts 属性中的类型。 0 - *(默认)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent LLD 规则字段。HTTP(S) 请求 body 数据，在 post_type 中使用。
privatekey	string	私钥文件名。
publickey	string	公共键文件的名称。
query_fields	array	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 规则字段。请求方法类型。 0 - GET 1 - *(默认)* POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent LLD 规则字段。指明哪部分响应应被存储起来。 0 - *(默认)* Body. 1 - Headers. 2 - HTTP 正文和 HTTP Headers 都将被存储. 对于 http 请求方法头，只允许值为 1。
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent LLD 规则字段。公共 SSL 键文件路径。
ssl_key_file	string	HTTP agent LLD 规则字段。私有 SSL 键文件路径。
ssl_key_password	string	HTTP agent LLD 规则字段。SSL 键文件密码。

state	integer	*(只读)* LLD 规则的状态。 取值: 0 - *(默认)* 正常; 1 - 不支持
status	integer	Status of the LLD rule. 取值: 0 - *(默认)* 启用 LLD 规则; 1 - 关闭 LLD 规则.
status_codes	string	HTTP agent LLD 规则字段。以逗号分隔的 HTTP 要求的状态码范围。 事例: 200,200-{$M},{$M},200-400
templateid	string	(只读) 父模板 LLD 规则的 ID。
timeout	string	HTTP agent LLD 规则字段。Item 数据轮训请求超时时间。支持用户宏。 默认: 3s 最大值: 60s
trapper_hosts	string	允许的主机。用于 trapper LLD 规则或 HTTP agent LLD 规则。
username	string	用户名进行身份验证。使用简单检查，SSH, Telnet，数据库监控，JMX 和 HTTP 代理 LLD 规则。 SSH 和 Telnet LLD 规则要求。
verify_host	integer	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 - *(默认)* 不验证. 1 - 验证.
verify_peer	integer	HTTP agent LLD 规则字段。主机认证证书合法性。 0 - *(默认)* 不验证. 1 - 验证.

LLD 规则过滤器

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：

属性类	说明	
conditions (必须)	array	用于过滤结果的过滤条件集。

属性类	说明	
evaltype (必须)	integer	过滤条件评价方法。 取值: 0 - and/or; 1 - and; 2 - or; 3 - 自定义表达式.

属性类	说明	
eval_formula	string	*(只读)* 生成的表达式，将用于计算筛选器条件。表达式包含通过其 “formulaid” 引用特定筛选条件的 id。“eval_formula” 的值等于带有自定义表达式的过滤器的 “formula” 的值。

属性类	说明	
formula	string	用户定义的表达式，用于使用自定义表达式计算过滤器的条件。表达式必须包含通过其“formulaid”引用特定筛选条件的id。表达式中使用的id必须与过滤器条件

属性类	说明

LLD rule 过滤器条件

LLD 规则过滤器条件对象定义对 LLD 宏的值执行的单独检查：

属性类	说明	
macro (必须)	string	对 LLD 宏执行检查。
value (必须)	string	比较值
formulaid	string	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。

属性类	说明	
operator	integer	条件运算符. 取值: 8 - (默认) 匹配正则表达式; 9 - 不匹配正则表达式.

Note:
更好地理解如何使用各种类型的表达式的过滤器, 参见发现规则. 查看 和发现规则. 创建 方法页面.

LLD macro path

The LLD macro path has the following properties:

Property	Type	Description
lld_macro (required)	string	LLD macro.
path (required)	string	Selector for value which will be assigned to corresponding macro.

LLD rule preprocessing

The LLD rule preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 5 - Regular expression matching; 11 - XML XPath; 12 - JSONPath; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 20 - Discard unchanged with heartbeat; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message.
error_handler_params (required)	string	Error handler parameters. Used with `error_handler`. Must be empty, if `error_handler` is 0 or 1. Can be empty if, `error_handler` is 2. Cannot be empty, if `error_handler` is 3.

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
5	Regular expression	pattern[1]	output[2]		0, 1, 2, 3
11	XML XPath	path[3]			0, 1, 2, 3
12	JSONPath	path[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[1]			0, 1, 2, 3
16	Check for error in JSON	path[3]			0, 1, 2, 3
17	Check for error in XML	path[3]			0, 1, 2, 3
20	Discard unchanged with heartbeat	seconds[4, 5, 6]			
23	Prometheus to JSON	pattern[5, 7]			0, 1, 2, 3
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		

[1] regular expression
[2] string
[3] JSONPath or XML XPath
[4] positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
[5] user macro
[6] LLD macro
[7] Prometheus pattern following the syntax: `<metric name>{<label name>="<label value>", ...} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro.
[8] Prometheus output following the syntax: `<label name>`.

LLD rule overrides

The LLD rule overrides object defines a set of rules (filters, conditions and operations) that are used to override properties of different prototype objects. It has the following properties:

Property	Type	Description
name (required)	string	Unique override name.

Property	Type	Description
step (required)	integer	Unique order number of the override.
stop	integer	Stop processing next overrides if matches. Possible values: 0 - *(default)* don't stop processing overrides; 1 - stop processing overrides if filter matches.
filter	object	Override filter.
operations	array	Override operations.

LLD rule override filter

The LLD rule override filter object defines a set of conditions that if they match the discovered object the override is applied. It has the following properties:

Property	Type	Description
evaltype (required)	integer	Override filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
conditions (required)	array	Set of override filter conditions to use for matching the discovered objects.

Property	Type	Description
eval_formula	string	*(readonly)* Generated expression that will be used for evaluating override filter conditions. The expression contains IDs that reference specific override filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression.
formula	string	User-defined expression to be used for evaluating conditions of override filters with a custom expression. The expression must contain IDs that reference specific override filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the override filter conditions: no condition can remain unused or omitted. Required for custom expression override filters.

LLD rule override filter condition

The LLD rule override filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties:

Property	Type	Description
macro (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.

Property	Type	Description
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 8 - *(default)* matches regular expression; 9 - does not match regular expression.

LLD rule override operation

The LLD rule override operation is combination of conditions and actions to perform on the prototype object. It has the following properties:

Property	Type	Description
operationobject (required)	integer	Type of discovered object to perform the action. Possible values: 0 - Item prototype; 1 - Trigger prototype; 2 - Graph prototype; 3 - Host prototype.

Property	Type	Description
operator	integer	Override condition operator. Possible values: 0 - *(default)* equals; 1 - does not equal; 2 - contains; 3 - does not contain; 8 - matches; 9 - does not match.
value	string	Pattern to match item, trigger, graph or host prototype name depending on selected object.
opstatus	object	Override operation status object for item, trigger and host prototype objects.
opdiscover	object	Override operation discover status object (all object types).
opperiod	object	Override operation period (update interval) object for item prototype object.
ophistory	object	Override operation history object for item prototype object.
optrends	object	Override operation trends object for item prototype object.
opseverity	object	Override operation severity object for trigger prototype object.
optag	array	Override operation tag object for trigger prototype object.
optemplate	array	Override operation template object for host prototype object.

Property	Type	Description
opinventory	object	Override operation inventory object for host prototype object.

LLD rule override operation status

LLD rule override operation status that is set to discovered object. It has the following properties:

Property	Type	Description
status (required)	integer	Override the status for selected object. Possible values: 0 - Create enabled; 1 - Create disabled.

LLD rule override operation discover

LLD rule override operation discover status that is set to discovered object. It has the following properties:

Property	Type	Description
discover (required)	integer	Override the discover status for selected object. Possible values: 0 - Yes, continue discovering the objects; 1 - No, new objects will not be discovered and existing ones will me marked as lost.

LLD rule override operation period

LLD rule override operation period is an update interval value (supports custom intervals) that is set to discovered item. It has the following properties:

Property	Type	Description
delay (required)	string	Override the update interval of the item prototype. Accepts seconds or a time unit with suffix (30s,1m,2h,1d) as well as flexible and scheduling intervals and user macros or LLD macros. Multiple intervals are separated by a semicolon.

LLD rule override operation history

LLD rule override operation history value that is set to discovered item. It has the following properties:

Property	Type	Description
history (required)	string	Override the history of item prototype which is a time unit of how long the history data should be stored. Also accepts user macro and LLD macro.

LLD rule override operation trends

LLD rule override operation trends value that is set to discovered item. It has the following properties:

Property	Type	Description
trends (required)	string	Override the trends of item prototype which is a time unit of how long the trends data should be stored. Also accepts user macro and LLD macro.

LLD rule override operation severity

LLD rule override operation severity value that is set to discovered trigger. It has the following properties:

Property	Type	Description
severity (required)	integer	Override the severity of trigger prototype. Possible values are: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.

LLD rule override operation tag

LLD rule override operation tag object contains tag name and values that are set to discovered trigger. It has the following properties:

Property	Type	Description
tag (required)	string	Override the tag name of trigger prototype tags.
value	string	Override the tag value of trigger prototype tags.

LLD rule override operation template

LLD rule override operation template object that is linked to discovered host. It has the following properties:

Property	Type	Description
templateid (required)	string	Override the template of host prototype linked templates.

LLD rule override operation inventory

LLD rule override operation inventory mode value that is set to discovered host. It has the following properties:

Property	Type	Description
inventory_mode (required)	integer	Override the host prototype inventory mode. Possible values are: -1 - disabled; 0 - *(default)* manual; 1 - automatic.

创建

说明

`object discoveryrule.create(object/array lldRules)`

此方法允许创建新的 LLD 规则。

参数

(object/array) 要创建的 LLD 规则。

除了标准 LLD 规则属性之外，该方法还接受以下参数。

属性类	描述	
filter	object	LLD 规则 LLD 规则的过滤对象。
preprocessing	array	LLD 规则预处理选项。
lld_macro_paths	array	LLD 规则预处理选项。
overrides	array	LLD 规则覆盖选项。

返回值

(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。

示例

新建 LLD 规则

创建 Zabbix agent LLD 规则去发现以已装入的文件系统。发现监控项（items）将被每 30s 被更新一次。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

使用一个过滤器

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。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

使用自定义表达式的筛选器

创建一个带有过滤器的 LLD 规则，该过滤器将使用自定义表达式来计算条件。LLD 规则必须只发现 “{#MACRO1}” 宏值同时匹配正则表达式 “regex1” 和 “regex2” 的对象，“{#MACRO2}” 宏值同时匹配 “regex3” 或 “regex4” 的对象。公式 id“A”、“B”、“C” 和 “D” 是任意选择的。

请求:

```
{
    "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

使用自定义查询字段和报头

创建具有自定义查询字段和标题的 LLD 规则。

请求:

```
{
    "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",
    }
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 35
}
```

参见

- LLD 规则过滤器

来源

CDiscoveryRule::create() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

Creating a LLD rule with overrides

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Discover database host",
        "key_": "lld.with.overrides",
        "hostid": "10001",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
        "interfaceid": "1155",
        "overrides": [
            {
                "name": "Discover MySQL host",
                "step": "1",
                "stop": "1",
                "filter": {
                    "evaltype": "2",
                    "conditions": [
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^mysqld\\.service$"
                        },
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^mariadb\\.service$"
                        }
                    ]
                },
                "operations": [
                    {
                        "operationobject": "3",
                        "operator": "2",
                        "value": "Database host",
                        "opstatus": {
                            "status": "0"
                        },
                        "optemplate": [
                            {
                                "templateid": "10170"
                            }
                        ]
```

```
                    }
                ]
            },
            {
                "name": "Discover PostgreSQL host",
                "step": "2",
                "stop": "1",
                "filter": {
                    "evaltype": "0",
                    "conditions": [
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^postgresql\\.service$"
                        }
                    ]
                },
                "operations": [
                    {
                        "operationobject": "3",
                        "operator": "2",
                        "value": "Database host",
                        "opstatus": {
                            "status": "0"
                        },
                        "optemplate": [
                            {
                                "templateid": "10263"
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "30980"
        ]
    },
    "id": 1
}
```

See also

- LLD rule filter
- LLD macro paths
- LLD rule preprocessing

Source

CDiscoveryRule::create() in *ui/include/classes/api/services/CDiscoveryRule.php*.

删除

说明

`object discoveryrule.delete(array lldRuleIds)`

此方法允许删除 LLD 规则。

参数

(array) 要删除的 LLD 规则的 IDs。

返回值

(object) 在 itemids 下返回一个包含被删除的 LLD 规则的 IDs。

示例

删除多个 LLD 规则

删除 2 个 LLD 规则。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.delete",
    "params": [
        "27665",
        "27668"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "ruleids": [
            "27665",
            "27668"
        ]
    },
    "id": 1
}
```

来源

CDiscoveryRule::delete() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

复制

说明

`object discoveryrule.copy(object parameters)`

此方法允许复制包含所有属性的 LLD 规则到给定的主机。

参数

(object) 参数定义要复制的 LLD 规则和目标主机。

属性类	描述	
discoveryids	array	要复制的 LLD 规则 id。
hostids	array	需要复制 LLD 规则到的主机 id。

返回值

(布尔值) 返回 true 如果复制成功。

事例

将 LLD 规则复制到多个主机

将一条 LLD 规则复制到两台主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.copy",
    "params": {
        "discoveryids": [
            "27426"
        ],
        "hostids": [
            "10196",
            "10197"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

来源

CDiscoveryrule::copy() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

更新

说明

`object discoveryrule.update(object/array lldRules)`

此方法允许更新已存在的 LLD 规则。

参数

`(object/array)` 要更新的 LLD 规则属性。

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。

另外见标准的 LLD 规则属性，此方法接受如下参数。

属性类	描述	
filter	object	LLD 规则 LLD 规则的过滤对象。
preprocessing	array	LLD 规则预处理选项。
lld_macro_paths	array	LLD 规则预处理选项。
overrides	array	LLD 规则覆盖选项。

返回值

`(object)` 在 `itemids` 属性下返回一个包含被更新的 LLD 规则的 IDs。

示例

为 LLD 规则添加一个筛选器

添加一个过滤器，以便 *{#FSTYPE}* 宏的内容与 lld 发现规则 *@File systems for discovery* 的正则表达式匹配。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
```

```
        "itemid": "22450",
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
                    "macro": "{#FSTYPE}",
                    "value": "@File systems for discovery"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

禁用 trapping

禁用 LLD trapping 发现规则。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}",
                "path": "$.json.path"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 36
}
```

来源

CDiscoveryRule::update() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

Updating LLD rule preprocessing options

Update an LLD rule with preprocessing rule “JSONPath”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": "12",
                "params": "$.path.to.json",
                "error_handler": "2",
                "error_handler_params": "5"
            }
        ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::update() in *ui/include/classes/api/services/CDiscoveryRule.php*.

获取

说明

`integer/array discoveryrule.get(object parameters)`

此方法允许根据给定的参数获取 LLD 规则。

参数

`(object)` 参数定义期望输出。

此方法支持如下参数。

属性类	描述	
itemids	string/array	返回给定 IDs 的 LLD 规则。

属性类	描述	
groupids	string/array	只返回属于来自给定组的主机的LLD规则。
hostids	string/array	返回属于给定主机的LLD规则。
inherited	boolean	如果设为`true`，返回自称自某模板的LLD规则。

属性类	描述	
interfaceids	string/array	返回使用给定主机接口的LLD规则。
monitored	boolean	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	如果设为`true`，返回属于(多个)模板的LLD规则。

属性类	描述	
templateids	string/array	返回属于给定给定模板的LLD规则。
selectFilter	query	在`filter`中返回LLD使用的筛选器。
selectGraphs	query	在`graphs`属性中返回属于LLD规则的图表原型。 Supports `count`.

属性类	描述	
selectHostPrototypes	query	在 `hostPrototyp` 属性中返回属于该 LLD 规则的主机原型。 Supports `count`.
selectHosts	query	在 `hosts` 属性下以数组形式返回属于该 LLD 规则的主机。
selectItems	query	在 `items` 下返回属于该 LLD 规则的 item。 Supports `count`.

属性类	描述	
selectTriggers	query	在triggers属性下返回属于该触发器原型。 Supports count.
selectApplicationPrototypes	query	返回一个applicationP属性，其中的应用程序原型属于属于此LLD规则的所有项原型。

属性类	描述	
selectLLDMacroPaths	query	返回一个lld_macro_pa属性，其中包含LLD宏列表和分配给每个相应宏的值的路径。

属性类	描述	
selectPreprocessing	query	返回“预处理”属性中的项目预处理选项。 它具有以下特性: type - (string) 预处理选项类型: 1 -自定义乘数; 2 -右纵倾; 3 -左纵倾; 4 -修剪; 5 -正则表达式匹配; 6 -布尔值到小

属性类	描述	
selectOverrides	query	返回 lld_rule_ove 属性，其中包含在 pro-to-type 对象上执行的覆盖过滤器、条件和操作的列表

属性类	描述
filter	object 仅返回紧缺匹配给定筛选条件的结果。 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` - LLD 规则所属主

属性类	描述	
limitSelects	integer	限制子选择返回的结果的数量。 适用于以下子选择: selctItems; selectGraphs; selectTrigger
sortfield	string/array	根据给定的属性把结果进行排序。 可能的值是: itemid, name, key_, delay, type and status.

属性类	描述	
countOutput	boolean	引用评论中详细描述了这些对于所有“get”方法都是通用的参数。
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）返回:

- 对象数组;
- 检索对象的计数 (如果使用了 “countOutput” 参数)。

示例

从一个主机获取多有的发现规则

获取主机 “10202” 所有的发现规则。

请求:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "27425",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Network interface discovery",
            "key_": "net.if.discovery",
            "delay": "1h",
            "state": "0",
```

```
        "status": "0",
        "trapper_hosts": "",
        "error": "",
        "templateid": "22444",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "interfaceid": "119",
        "description": "Discovery of network interfaces as defined in global regular expression \"Netw
        "lifetime": "30d",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0"
    },
    {
        "itemid": "27426",
        "type": "0",
        "snmp_oid": "",
        "hostid": "10202",
        "name": "Mounted filesystem discovery",
        "key_": "vfs.fs.discovery",
        "delay": "1h",
        "state": "0",
        "status": "0",
        "trapper_hosts": "",
        "error": "",
        "templateid": "22450",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "interfaceid": "119",
        "description": "Discovery of file systems of different types as defined in global regular expr
        "lifetime": "30d",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
```

```
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 1
}
```

检索过滤条件

检索 LLD 规则 “24681” 的名称及其过滤条件。筛选器使用 “and” 求值类型，因此 “formula” 属性为空，并自动生成 “eval_formula”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": [
            "name"
        ],
        "selectFilter": "extend",
        "itemids": ["24681"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "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
}
```

根据 URL 获取 LLD 规则

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。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "hostids": "10257",
        "filter": {
            "type": "19",
            "url": "http://127.0.0.1/discoverer.php"
        }
    },
    "id": 39,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28336",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10257",
            "name": "API HTTP agent",
            "key_": "api_discovery_rule",
            "delay": "5s",
            "history": "90d",
            "trends": "0",
            "status": "0",
            "value_type": "4",
            "trapper_hosts": "",
            "units": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "1",
            "interfaceid": "5",
            "description": "",
```

```
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "http://127.0.0.1/discoverer.php",
            "query_fields": [
                {
                    "mode": "json"
                },
                {
                    "elements": "2"
                }
            ],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": {
                "X-Type": "api",
                "Authorization": "Bearer mF_A.B5f-2.1JcM"
            },
            "retrieve_mode": "0",
            "request_method": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 39
}
```


参考

- 图像原型
- 主机
- 监控项原型
- LLD 规则过滤器
- 触发器原型

来源

CDiscoveryRule::get() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

Source

CDiscoveryRule::get() in *ui/include/classes/api/services/CDiscoveryRule.php*.

28. 维护模式

该类设计用于维护模式。

对象引用:

- 维护模式
- 时间周期

可用的方法:

- maintenance.create - 创建新的维护模式

- maintenance.delete - 删除维护模式
- maintenance.get - 获取维护模式
- maintenance.update - 更新维护模式

> 维护模式对象

如下对象与 `maintenance`API 关联。

维护模式

维护模式对象有如下属性。

属性类	描述	
maintenanceid	string	(只读)维护模式的ID。
name (必须)	string	维护模式的名称。
active_since (必须)	timestamp	维护模式生效的时刻。
active_till (必须)	timestamp	维护模式失效的时刻。
description	string	维护模式说明。

属性类	描述	
maintenance_type	integer	维护模式类型。 可能的值： 0 - *(默认)* 采集监控数据; 1 - 不采集监控数据.
tags_evaltype	integer	问题标签评估方法。 可能的值: 0 -(默认) 和/或; 2 -或。

时间周期

时间周期（time period）对象用于定义维护模式生效的时间周期。它有如下属性。

属性类	描述	
timeperiodid	string	*(只读)* 维护模式ID。

属性类	描述	
day	integer	维护模式生效的月份天次。 月份时间周期要求。

属性类	描述	
dayofweek	integer	维护模式生效的周次。 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。 用于周或月时间周期。仅周

属性类	描述	
every	integer	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。 对于月周期 every 定义该月维护模式生效的周次。可能的值：1 - first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week.

属性类	描述	
month	integer	维护模式必须生效的月份。 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。 要求只有月时间周

属性类	描述	
period	integer	维护模式周期的时间(秒)。 默认: 3600.
start_date	timestamp	维护模式必须生效的日期。 只需要一个时间段
start_time	integer	默认: 当前时间. 一天内维护模式开始的时刻。 天、周、月周期要求。

属性类	描述	
timeperiod_type	integer	时间周期类型。 可能的值： 0 - (默认) 仅一次; 2 - 天; 3 - 周; 4 - 月.

问题标签

属性类	描述	
tag(必须) s	ring 问	标签名称。
operator	integer	条件操作符。 可能的值: 0 - 等于; 2 -(默认) 包含。
timeperiodid	string	标签值

创建

说明

`object maintenance.create(object/array maintenances)`

此方法允许创建新的维护模式。

参数

`(object/array)` 要创建的维护模式。

另外见标准的维护属性，此方法接受如下参数。

属性类	描述	
groupids (必须)	array	要执行维护模式的主机组 IDs。
hostids (必须)	array	要执行维护模式的主机的 IDs。
timeperiods (必须)	array	维护模式时间周期。
tags	array	问题标签 定义哪些问题必须被抑制。如果没有给出标记，所有活动维护主机问题都将被抑制。

<note important> 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）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。

示例

创建一个维护模式

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。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Sunday maintenance",
        "active_since": 1358844540,
        "active_till": 1390466940,
        "tags_evaltype": 0,
        "groupids": [
            "2"
        ],
        "timeperiods": [
            {
                "timeperiod_type": 3,
                "every": 1,
                "dayofweek": 64,
                "start_time": 64800,
                "period": 3600
            }
        ],
        "tags": [
            {
                "tag": "service",
                "operator": "0",
                "value": "mysqld",
            },
            {
                "tag": "error",
                "operator": "2",
                "value": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

参见

- 时间周期

来源

CMaintenance::create() in *frontends/php/include/classes/api/services/CMaintenance.php*.

删除

说明

`object maintenance.delete(array maintenanceIds)`

此方法允许删除维护模式。

参数

(array) 要删除的维护模式的 IDs。

返回值

(object) 在 maintenanceids 属性下返回包含已被删除的维护模式的 ID 对象。

示例

删除多个维护模式

删除 2 个维护模式。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.delete",
    "params": [
        "3",
        "1"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3",
            "1"
        ]
    },
    "id": 1
}
```

来源

CMaintenance::delete() in *frontends/php/include/classes/api/services/CMaintenance.php*.

更新

说明

`object maintenance.update(object/array maintenances)`

此方法允许更新已存在的维护模式。

参数

(object/array) 要更新的维护模式的属性。

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。

另外见标准的维护属性, 此方法接受如下参数。

属性类	描述	
groupids	array	要替换的当前主机组的主机组 IDs。
hostids	array	要替换当前主机的主机 IDs。
timeperiods	array	要替换当前维护模式时间周期的时间周期。

<note important> 每一个维护模式至少一个主机或者一个主机组被定义。:::

返回值

(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。

示例

指定不同的主机

用两个不同的主机替换当前分配给维护 “3” 的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.update",
    "params": {
        "maintenanceid": "3",
        "hostids": [
            "10085",
            "10084"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

参见

- 时间周期

来源

CMaintenance::update() in *frontends/php/include/classes/api/services/CMaintenance.php*.

获取

说明

integer/array maintenance.get(object parameters)

此方法用于根据给定参数获取维护模式。

参数

(object) 定义期望输出的参数。

此方法支持如下参数。

属性类	描述	
groupids	string/array	仅返回指定到给定主机组的维护模式。
hostids	string/array	仅返回指定到给定主机的维护模式。
maintenanceids	string/array	仅返回给定IDs的维护模式。

属性类	描述	
selectGroups	query	在group属性中返回维护模式所指定的主机组。
selectHosts	query	在host属性中返回维护模式所指定的主机。
selectTags	query	在tags属性中返回维护模式所指定的问题标签属性。

属性类	描述	
selectTimeperiods	query	在 `timeperiods` 属性中返回维护模式的时间周期。
sortfield	string/array	根据给定的属性记性排序。 取值范围: `maintenanceid`, `name` and `maintenance_type`. 取值范围： ， `name` and `maintenance_type`。

属性类	描述	
countOutput	boolean	这些参数在参考说明中详细描述的所有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	

返回值

(integer/array) 返回:

- 对象数组;
- 检索对象的计数 (如果使用了 “countOutput” 参数)。

示例

获取维护模式

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。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.get",
    "params": {
        "output": "extend",
        "selectGroups": "extend",
        "selectTimeperiods": "extend",
        "selectTags": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenanceid": "3",
            "name": "Sunday maintenance",
            "maintenance_type": "0",
            "description": "",
            "active_since": "1358844540",
            "active_till": "1390466940",
            "tags_evaltype": "0",
            "groups": [
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "internal": "0"
                }
            ],
            "timeperiods": [
                {
                    "timeperiodid": "4",
                    "timeperiod_type": "3",
                    "every": "1",
                    "month": "0",
                    "dayofweek": "1",
                    "day": "0",
                    "start_time": "64800",
                    "period": "3600",
                    "start_date": "2147483647"
                }
            ],
            "tags": [
                {
                    "tag": "service",
                    "operator": "0",
                    "value": "mysqld",
                },
                {
                    "tag": "error",
                    "operator": "2",
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```


参考

- 主机
- 主机组
- 时间周期

来源

CMaintenance::get() in *frontends/php/include/classes/api/services/CMaintenance.php*.

29. 拓扑图

这个类设计用来处理拓扑图

相关对象:

- Map
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shape
- Map line

可用方法:

- map.create - 创建一个新的拓扑图
- map.delete - 删除拓扑图
- map.get - 获取一个拓扑图
- map.update - 更新一个拓扑图

> 对象

以下内容是关于拓扑图 `map` API.

Map 拓扑图

拓扑图对象具有以下属性。

Property	Type	Description
sysmapid	string	*(readonly)* 拓扑图 ID。
height (required)	integer	拓扑图画布高度。
name (required)	string	拓扑图名称。
width (required)	integer	拓扑图宽度。
backgroundid	string	拓扑图背景图像 ID。
expand_macros	integer	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	如果只有一个触发器告警是否显示详。 可能的值： 0 是只显示数目， 1 是显示触发器详情
grid_align	integer	是否启用网格对齐。 可能的值： 0 是不用 1 是使用
grid_show	integer	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	拓扑图网格的大小。 支持 20, 40, 50, 75，100 像素。 默认是 50
highlight	integer	是否启用图标高亮显示。 可能的值： 0 是不用 1 是使用
iconmapid	string	拓扑图使用图表的 ID。

Property	Type	Description
label_format	integer	是否启用高级标签。 可能的值： 0 是不用 1 是使用
label_location	integer	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	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	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	图像元素自定义标签。 图像元素的自定义标签。
label_string_map	string	拓扑图元素自定义标签。 拓扑图元素自定义标签。
label_string_trigger	string	触发器元素自定义标签 触发器元素自定义标签。
label_type	integer	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。
label_type_host	integer	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	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	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

Property	Type	Description
label_type_map	integer	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	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	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	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。
show_unack	integer	如何显示问题。 可能的值： 0：*(默认)* 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	拓扑图所有用户的 ID
private	integer	拓扑图的共享类型 可能的值： 0：公共的拓扑图 1：*(默认)* 私有的拓扑图

拓扑图元素

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	*(只读)* 拓扑图元素的 ID
elements (required)	array	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	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	用于在默认状态下显示元素的图像的 ID。

Property	Type	Description
areatype	integer	应该如何显示独立的主机组主机。 可能的值 0-*(默认)* 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	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	一个主机组元素如何显示在拓扑图上 可能的值： 0-*(默认)* 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
height	integer	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	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	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	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	元素的标签
label_location	integer	拓扑元素标签的位置。 可能的值： -1: 默认的位置 0-底部 1-左边 2-右边 3-上边
permission	integer	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	*(只读)* 元素所述拓扑图的 ID
urls	array	拓扑图元素的 URL 。 The map element URL object is described in detail below.
use_iconmap	integer	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	主机组元素放置算法 可能的值： 0-网格
width	integer	主机组元素固定的像素宽度。 默认是：200
x	integer	元素的 x 坐标，单位为像素。 默认是：0
y	integer	元素的 y 坐标，单位为像素。 默认是：0

拓扑图元素的主机

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

拓扑图元素中的主机组

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

拓扑图元素中的拓扑图

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

拓扑图元素中的触发器

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

拓扑图元素中的 URL

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.

拓扑图关联

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	在一端连接的第一个拓扑图元素的 ID。
selementid2 (required)	string	另一端连接的第一个拓扑图元素的 ID。
color	string	行颜色作为十六进制颜色代码。 默认是：“000000”
drawtype	integer	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	行标签
linktriggers	array	拓扑图链接触发器用作链接状态指示器。
permission	integer	权限等级类型 可能的值：-1-没有 2-只读 3-可读可写
sysmapid	string	该关联所属拓扑图 ID

拓扑图关联触发器

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	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	关联触发器所属拓扑图 ID。

拓扑图 URL

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	*(*只读*)* 拓扑图 URL ID
name (required)	链接标题。	
url (required)	string	链接 URL
elementtype	integer	拓扑图元素可用 URL 类型 默认：0
sysmapid	string	所属 URL 的拓扑图 ID

拓扑图用户

基于用户的拓扑图权限列表。它具有以下特性:

Property	Type	Description
sysmapuserid	string	*(*只读*)* 拓扑图用户 ID。
userid (required)	string	User ID.
permission (required)	integer	权限等级类型 可能的值： -1-没有 2-只读 3-可读可写

拓扑图用户组

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	*(*只读*)* 拓扑图用户组的 ID
usrgrpid (required)	string	User group ID.
permission (required)	integer	权限等级类型 可能的值： -1-没有 2-只读 3-可读可写

地图形状

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)* 拓扑图形状元素的 ID
type (required)	integer	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	元素的 x 坐标，单位为像素。 默认是：0
y	integer	元素的 y 坐标，单位为像素。 默认是：0
width	integer	以像素为单位的形状宽度。 默认是：200
height	integer	以像素为单位的形状高度。 默认是：200
text	string	文本的形状。
font	integer	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	字体大小，单位是像素 默认：11
font_color	string	字体颜色 默认是：“000000”
text_halign	integer	水平对齐的文本 可能的值： 0-中间（默认） 1-左边 2-右边
text_valign	integer	垂直对齐文本 可能的值： 0-中间（默认） 1-顶部 2-底部

Property	Type	Description
border_type	integer	边界类型 可能的值： 0-没有（默认） 1 - ———— 2 - ┅ 3 - – – –
border_width	integer	边框的宽度，以像素为单位 默认：0
border_color	string	边界的颜色 默认：‘000000’
background_color	string	背景颜色（填充颜色） 默认是：无
zindex	integer	用于定制形状的值 (z-index)。 默认是：0

拓扑图线

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	*(只读)* 拓扑图形状元素的 ID
x1	integer	以像素为单位的直线点 1 的 x 坐标。 默认是：0
y1	integer	以像素为单位的直线点 1 的 y 坐标。 默认是：0
x2	integer	以像素为单位的直线点 2 的 x 坐标。 默认是：200
y2	integer	以像素为单位的直线点 2 的 x 坐标。 默认是：200
line_type	integer	边界类型 可能的值： 0-没有（默认）1 - ———— 2 - ┅ 3 - – – –
line_width	integer	边框的宽度，以像素为单位 默认：0
line_color	string	边界的颜色 默认：‘000000’
zindex	integer	用于定制形状的值 (z-index)。 默认是：0

创建

描述

`object map.create(object/array maps)`

这个方法允许创建一个新的拓扑图。

参数

(object/array) Maps to create.

除了standard map properties之外，该方法还接受以下参数。

Parameter	Type	Description
links	array	拓扑图上创建拓扑图链接
selements	array	拓扑图上创建拓扑图元素
urls	array	拓扑图上创建拓扑图 URL
users	array	拓扑图共享用户
userGroups	array	拓扑图共享用户组
shapes	array	拓扑图上创建拓扑图图形
lines	array	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。

返回值

（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。

例子

创建一个空的拓扑图

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
}
```

创建一个主机拓扑图

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
}
```

创建一个触发器拓扑图

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
}
```

拓扑图共享

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
}
```

拓扑图形状

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

描述

`object map.delete(array mapIds)`

这个方法允许删除拓扑图。

Parameters 参数

(array) 需要删除拓扑图的 IDs。

返回值

(object) 返回包含 “sysmapid” 属性下的已删除拓扑图的 IDS 的对象。

示例如下

删除多个拓扑图

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
}
```

源

CMap::delete() in *frontends/php/include/classes/api/services/CMap.php*.

更新

描述

`object map.update(object/array maps)`

此方法可以用来更新已存在的拓扑图

Parameters 参数

`(object/array)` 更新拓扑图参数。

`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。

除了standard map properties, 此方法还接受以下参数。

Parameter	Type	Description
links	array	拓扑图链接以替换现有的链接。
selements	array	拓扑图元素替换成已存在的拓扑图元素
urls	array	拓扑图 URLs 替换成已存在的 URLs
users	array	拓扑图的共享用户替换成已存在的共享用户
userGroups	array	拓扑图共享用户组替换成已存在的共享用户组
shapes	array	拓扑图图形替换成已存在的图形
lines	array	图谱图的连线替换成已存在的连线

<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. :::

返回值

`(object)` 返回一个对象，该对象包含 “sysmapid” 属性下更新的映射的 ID。

示例如下

调整拓扑图的大小

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
}
```

改变拓扑图的属组

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

源

CMap::update() in *frontends/php/include/classes/api/services/CMap.php*.

获取

描述

`integer/array map.get(object parameters)`

这个方法允许根据给定参数检索出符合条件的拓扑图。

参数

`(object)` 定义所需输出的参数。

此方法支持一下参数。

Parameter	Type	Description
sysmapids	string/array	仅返回给出 IDS 的拓扑图。
userids	string/array	仅返回所给用户 IDS 所属的拓扑图。
expandUrls	flag	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	返回 “iconmap” 属性中拓扑图上使用的图标映射。
selectLinks	query	返回 “links” 属性中元素之间的映射链接。
selectSelements	query	返回” selements ” 属性中的 map 元素。
selectUrls	query	返回 “URLs” 属性中的映射 url。
selectUsers	query	返回与 “users” 属性共享映射的用户。
selectUserGroups	query	返回与 “userGroups” 属性共享映射的用户组。
selectShapes	query	在 “形状” 属性中返回地图中的地图形状。
selectLines	query	在 “lines” 属性中返回地图中的地图行。

Parameter	Type	Description
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:

- 一个数组对象;
- 如果使用了 `countOutput` 参数，则检索对象的计数。

举例

检索一个拓扑图

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

源

CMap::get() in *frontends/php/include/classes/api/services/CMap.php*.

30. 媒介类型

这个类设计用来处理媒介类型。

对象引用:

- Media type

可用方法:

- mediatype.create - 创建新媒介类型
- mediatype.delete - 删除媒介类型
- mediatype.get - 获取媒介类型
- mediatype.update - 更新媒介类型

> 对象

以下对象是直接关联到 `mediatype` 接口

媒介类型

媒介类型参数拥有以下参数

Property	Type	Description
mediatypeid	string	*(readonly)* 媒介类型 ID
description (required)	string	媒介类型名称
type (required)	integer	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	

对于脚本媒体类型，“exec_path” 包含已执行脚本的名称。

对于 Ez 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	GSM 调制解调器的串行设备名称。 用于 SMS 媒介类型
passwd	string	认证的密码 \\用于 Jabber 和 Ez Texting 媒介类型
smtp_email	string	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_server	string	用于电子邮件媒介类型 \\用于电子邮件媒介类型
status	integer	媒介类型是否是启用的 \\可能的值： 0-启用（默认）1-禁用
username	string	用户名或 Jabber 标识符 用于 Jabber and Ez Texting 媒介类型
exec_params	string	脚本参数。 每个参数以新的行提要结束
maxsessions	integer	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	发送警报的最大尝试次数。 \\可能的值： 1-10，默认是 3

attempt_interval	string	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

Webhook parameters

Parameters passed to webhook script when it is called, have the following properties.

Property	Type	Description
name (required)	string	Parameter name.
value	string	Parameter value, support macros. Supported macros described on page.

Message template

The message template object defines a template that will be used as a default message for action operations to send a notification. It has the following properties.

Property	Type	Description
eventsource (required)	integer	Event source. Possible values: 0 - triggers; 1 - discovery; 2 - autoregistration; 3 - internal.
recovery (required)	integer	Operation mode. Possible values: 0 - operations; 1 - recovery operations; 2 - update operations.
subject	string	Message subject.
message	string	Message text.

创建

描述

`object mediatype.create(object/array mediaTypes)`

此方法允许创建新的媒介类型

参数

`(object/array)` 创建媒介类型

该方法接受媒介类型关于standard media type properties。

返回值

`(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。

示例如下

创建一个媒介类型

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
}
```

创建具有自定义选项的媒体类型

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
}
```

源

CMediaType::create() in *frontends/php/include/classes/api/services/CMediaType.php*.

Source

CMediaType::create() in *ui/include/classes/api/services/CMediaType.php*.

删除

描述

`object mediatype.delete(array mediaTypeIds)`

此方法适合删除媒介类型

参数

`(array)` 要删除媒介类型的 IDS

返回值

`(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。

示例如下

删除多个媒介类型

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
}
```

源

CMediaType::delete() in *frontends/php/include/classes/api/services/CMediaType.php*.

更新

描述

`object mediatype.update(object/array mediaTypes)`

此方法允许更新已存在的媒介类型。

参数

`(object/array)` Media type properties to be updated.

`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

返回值

`(object)` 返回包含 `mediatypeids` 属性下所更新 IDs 的对象。

示例如下

启用一个媒介类型

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
}
```

源

CMediaType::update() in *frontends/php/include/classes/api/services/CMediaType.php*.

获取

描述

`integer/array mediatype.get(object parameters)`

此方法用于检索给定参数和符合条件的媒介类型

参数

`(object)` 定义所需输出的参数。

此方法支持一下参数。

Parameter	Type	Description
mediatypeids	string/array	仅返回所给 IDs 的媒介类型。
mediaids	string/array	只返回给定媒体使用的媒介类型。
userids	string/array	只返回给定用户使用的媒介类型。
selectUsers	query	返回 `users` 属性中使用媒介类型的用户。
sortfield	string/array	根据给定的属性对结果进行排序。 可能的值是: `mediatypeid`。
countOutput	boolean	这些参数对于所有的 “get” 方法都是通用的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) 返回如下：

- 一个对象数组；
- 如果使用了 “countOutput” 参数，则检索对象的计数。

示例如下

检索媒介类型

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

源

CMediaType::get() in *frontends/php/include/classes/api/services/CMediaType.php*.

31. 问题

这个类设计用于描述问题。

相关对象:

- Problem

可用方法:

- problem.get - 获取问题信息

> 对象

Note:
问题是由 Zabbix 服务器创建的，不能通过 API 进行修改。

问题对象拥有以下属性

Property	Type	Description
eventid	string	问题事件的 ID
source	integer	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	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	关联对象的 ID。
clock	timestamp	问题事件创建的时间。
ns	integer	问题事件创建的纳秒时间。
r_eventid	string	恢复时间的 ID。

Property	Type	Description
r_clock	timestamp	恢复事件创建的时间。
r_ns	integer	恢复事件创建的纳秒时间。
correlationid	string	事件被全局的关联规则恢复，关联规则的 ID。
userid	string	手动关闭问题的用户 ID。
name	string	解决问题名称。
acknowledged	integer	问题知晓状态 可能的值： 0-不知道 1-知道
severity	integer	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
suppressed	integer	是否抑制问题。 可能的值: 0 - 问题是一个正常状态; 1 - 问题被抑制
opdata	string	使用扩展宏的操作数据。
urls	array of Media type URLs	主动媒介的 URLS 。

Problem

Note:
Problems are created by the Zabbix server and cannot be modified via the API.

The problem object has the following properties.

Property	Type	Description
eventid	string	ID of the problem event.
source	integer	Type of the problem event. Possible values: 0 - event created by a trigger; 3 - internal event.
object	integer	Type of object that is related to the problem event. Possible values for trigger events: 0 - trigger. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
objectid	string	ID of the related object.
clock	timestamp	Time when the problem event was created.
ns	integer	Nanoseconds when the problem event was created.
r_eventid	string	Recovery event ID.
r_clock	timestamp	Time when the recovery event was created.
r_ns	integer	Nanoseconds when the recovery event was created.
correlationid	string	Correlation rule ID if this event was recovered by global correlation rule.
userid	string	User ID if the problem was manually closed.

Property	Type	Description
name	string	Resolved problem name.
acknowledged	integer	Acknowledge state for problem. Possible values: 0 - not acknowledged; 1 - acknowledged.
severity	integer	Problem current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
suppressed	integer	Whether the problem is suppressed. Possible values: 0 - problem is in normal state; 1 - problem is suppressed.
opdata	string	Operational data with expanded macros.
urls	array of Media type URLs	Active media types URLs.

问题标签

问题标签对象具有以下属性。

Property	Type	Description
tag	string	问题标签名字.
value	string	问题标签值.

媒介类型 URLs

具有媒体类型 url 的对象具有以下属性。

Property	Type	Description
name	string	媒介类型定义 URL 名称.
url	string	媒介类型定义 URL 值.

结果将只包含具有启用事件菜单项的活动媒体类型的条目。属性中使用的宏将被展开，但如果其中一个属性包含未展开的宏，则结果中将排除这两个属性。Supported macros described on page.

获取

描述

`integer/array problem.get(object parameters)`

此方法允许根据给定参数检索符合条件的问题

参数

`(object)` 定义所需输出的参数

此方法支持一下参数

Parameter	Type	Description
eventids	string/array	仅返回所给 IDs 的问题。
groupids	string/array	仅返回所属给定主机组对象的问题。

Parameter	Type	Description
hostids	string/array	仅返回所给定主机对象的问题。
objectids	string/array	仅返回所给对象创建的问题。
applicationids	string/array	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	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	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。
acknowledged	boolean	`true`-返回已知晓的问题 返回未知晓的问题
suppressed	boolean	`true` - 仅返回被抑制问题; `false` - 返回问题在正常状态。
severities	integer/array	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	规则标签搜索。 可能的值： 0 - (默认) 与/或；2 - 或
tags	array of objects	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	只返回 ID 大于或等于给定 ID 的问题。
eventid_till	string	只返回 ID 小于或等于给定 ID 的问题。
time_from	timestamp	仅返回问题创建时间在所给时间之后的问题。
time_till	timestamp	仅返回问题创建时间在所给时间之前的问题。

Parameter	Type	Description
selectAcknowledges	query	返回一个 acknowledges 属性更新问题。问题更新按时间倒序排序。 问题更新对象具有以下属性: acknowledgeid - (string) update's ID; userid - (string) ID 为更新事件的用户; eventid - (string) ID 为更新事件; clock - (timestamp) 更新事件的时间; message - (string) 信息是 text 格式; action - (integer) 更新操作类型 (see event.acknowledge); old_severity - (integer) 在此更新操作之前的事件严重性; new_severity - (integer) event severity after this update action; Supports count.
selectTags	query	返回一个tags 问题标签资产. Output format: [{"tag": "<tag>", "value": "<value>"}, ...].
selectSuppressionData	query	返回一个 suppression_data 资产维护列表: maintenanceid - (string) 维护 ID; suppress_until - (integer) 直到问题被抑制。
sortfield	string/array	根据给定的属性对结果进行排序。 可能的的值：eventid
countOutput	boolean	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	

返回值

(integer/array) Returns either:

- 一个数组对象
- 返回检索到对象的数量，如果 countOutput 参数被引用

示例如下

返回触发器问题事件

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
}
```

猜你想看

- Alert
- Item
- Host
- LLD rule
- Trigger

源

CEvent::get() in *frontends/php/include/classes/api/services/CProblem.php*.

32. 代理 **Proxy**

这个类主要用来设计工作于代理 Proxy

相关对象:

- Proxy
- Proxy interface

可用方法:

- proxy.create - 创建一个新的 proxies
- proxy.delete - 删除 proxies
- proxy.get - 获取 proxies
- proxy.update - 更新 proxies

> Proxy 对象

以下对象直接关系到 `proxy`API

代理 Proxy

代理 Proxy 对象拥有以下属性

Property	Type	Description
proxyid	string	*(readonly)* 代理 Proxy 的 id
host (required)	string	代理 Proxy 的名称
status (required)	integer	代理 Proxy 的类型 可能的值： 5 - 主动代理 6 - 被动代理
description	text	代理 Proxy 的描述
lastaccess	timestamp	*(readonly)* 上一次代理连接 Zabbix Server 的时间
tls_connect	integer	连接主机 可能的值： 1 - *(default)* 非加密 2 - 共享密钥（PSK） 3 - 证书
tls_accept	integer	从代理 Proxy 连接 可能的值： 1 - *(default)* 非加密 2 - 共享密钥（PSK） 3 - 证书
tls_issuer	string	证书发行者
tls_subject	string	证书的主题
tls_psk_identity	string	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	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。
proxy_address	string	限制 IP/DNS 与 Zabbix server 通信，在 proxy 主动模式下.
auto_compress	integer	*(readonly) (*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

代理 Proxy 接口

代理接口对象默认接口用于连接被动代理。以下属性

Property	Type	Description
interfaceid	string	*(readonly)* 接口的 ID
dns (required)	string	连接的 DNS 名称 如果通过 IP 地址进行连接，可以为空
ip (required)	string	连接到 IP 地址 如果通过 DNS 名称连接可以为空
port (required)	string	连接端口号.
useip (required)	integer	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	///(只读) // 接口所属的代理的 ID。

创建

描述

`object proxy.create(object/array proxies)`

此方法用于创建新的代理 Proxy

参数

`(object/array)` 创建代理

此外standard proxy properties，此方法接受以下参数。

Parameter	Type	Description
hosts	array	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	创建主机接口用于被动代理 被动代理的需求

返回值

`(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。

示例如下

创建一个主动的代理

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
}
```

创建一个被动 Proxy

创建一个被动 Proxy "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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10284"
        ]
    },
    "id": 1
}
```

See also

- Host
- Proxy interface

源

CProxy::create() in *frontends/php/include/classes/api/services/CProxy.php*.

删除

描述

`object proxy.delete(array proxies)`

此方法允许删除代理

参数

(array) 删除代理的 IDs

返回值

(object) 返回在 proxyids 属性下包含已删除代理的 id 的对象。

示例如下

删除多个代理 Proxy

删除两个代理 Proxy

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.delete",
    "params": [
        "10286",
        "10285"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10286",
            "10285"
        ]
    },
    "id": 1
}
```

源

CProxy::delete() in *frontends/php/include/classes/api/services/CProxy.php*.

更新

描述

`object proxy.update(object/array proxies)`

此方法允许更新已存在的代理 Proxy

代理 Proxy 参数

(object/array) 代理 Proxy 参数被更新

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。

此外standard proxy properties，此方法接受以下参数

Parameter	Type	Description
hosts	array	代理 Proxy 监视的主机。如果一个主机已经被一个不同的代理 Proxy 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	主机接口将会替换已存在的主机接口用于被动代理 Proxy

返回值

（`object`）返回一个对象，该对象包含 `proxyids` 属性下更新的代理 Proxy 的 id。

示例如下

改变一个主机的代理 Proxy

更新代理 Proxy 以监视两个给定的主机。

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
}
```

改变代理的状态

改变代理 Proxy 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
}
```

猜你想看

- Host
- Proxy interface

源

CProxy::update() in *frontends/php/include/classes/api/services/CProxy.php*.

获取

描述

`integer/array proxy.get(object parameters)`

该方法允许根据给定的参数查询代理。

参数

`(object)` 定义所需输出的参数。

此方法支持一下参数。

Parameter	Type	Description
proxyids	string/array	仅返回所给 ID 的代理
selectHosts	query	返回在 `hosts` 属性中代理监控的主机
selectInterface	query	返回在 `interface` 属性中被动代理使用代理接口
sortfield	string/array	根据所给的属性进行排序 可能的值：`hostid`, `host` 和 `status`.
countOutput	boolean	改参数适用于所有的 `get` 方法，详细描述是在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:

- 一个对象数组
- 搜索到对象的数量，如果 `countOutput` 对象被使用

示例如下

检索所有的代理

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": [
        {
            "host": "Active proxy",
            "status": "5",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30091",
            "interface": []
        },
        {
            "host": "Passive proxy",
            "status": "6",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30092",
            "interface": {
                "interfaceid": "30109",
                "hostid": "30092",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "10051"
            }
        }
    ],
    "id": 1
}
```

See also

- Host
- Proxy interface

源

CProxy::get() in *frontends/php/include/classes/api/services/CProxy.php*.

33. 聚合图形

这个类设计工作于聚合图形

相关对象:

- Screen
- Screen user
- Screen user group

可用方法:

- screen.create - 创建新 screen
- screen.delete - 删除 screens
- screen.get - 获取 screens
- screen.update - 更新 screens

> 对象

以下对象直接跟 `screen` API 相关。

聚合图形

聚合图形对象拥有以下属性。

Property	Type	Description
screenid	string	*(只读)* 聚合图形 ID。
name (required)	string	聚合图形名称。
hsize	integer	聚合图形宽度 默认：1
vsize	integer	聚合图形高度 默认：1
userid	string	聚合图形所属用户 ID。
private	integer	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

聚合图形用户

聚合图形的权限基于用户，它拥有以下属性：

Property	Type	Description
screenuserid	string	*(只读)* 聚合图形用户 ID。
userid (required)	string	用户 ID。
permission (required)	integer	权限等级类别 可能的值： 2 - 只读 3 - 读写权限

聚合图形用户组

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)* 聚合图形用户组 ID。
usrgrpid (required)	string	用户组 ID。

Property	Type	Description
permission (required)	integer	权限等级类别 可能的值： 2 - 只读 3 - 读写权限

创建

描述

`object screen.create(object/array screens)`

此方法允许创建新的聚合图形

参数

`(object/array)` 创建聚合图形

此外standard screen properties，此方法接受以下参数：

Parameter	Type	Description
screenitems	array	为聚合图形创建聚合图形项
users	array	聚合图形用户共享在聚合图形上创建
userGroups	array	聚合图形用户组共享在聚合图形上创建

返回值

`(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。

示例如下

创建一个聚合图形

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
}
```

聚合图形分享

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
}
```

猜你想看

- Screen item
- Screen user
- Screen user group

源

CScreen::create() in *frontends/php/include/classes/api/services/CScreen.php*.

删除

描述

`object screen.delete(array screenIds)`

此方法允许删除聚合图形

参数

(array) 删除聚合图形的 IDs

返回值

(object) 返回包含 screenids 属性下的已删除屏幕的 id 的对象。

示例如下

删除多个聚合图形

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
}
```

源

CScreen::delete() in *frontends/php/include/classes/api/services/CScreen.php*.

更新

描述

object screen.update(object/array screens)

此方法允许更新已存在的聚合图形

参数

(object/array) 聚合图形参数将被更新

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。

此外standard screen properties, 此方法接受以下参数

Parameter	Type	Description
screenitems	array	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	聚合图形用户共享替换已存在的元素
userGroups	array	聚合图形用户组共享替换已存在的元素

返回值

(object) 返回一个对象，该对象包含 screenids 属性下更新聚合图形的 id。

示例如下

重命名一个聚合图形

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
}
```

改变聚合图形属主

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

源

CScreen::update() in *frontends/php/include/classes/api/services/CScreen.php*.

获取

描述

`integer/array screen.get(object parameters)`

此方法允许搜索符合所给参数的聚合图形

参数

`(object)` 定义所需输出的参数。

此方法支持以下参数

Parameter	Type	Description
screenids	string/array	返回所给 ID // （单个或者多个）// 的聚合图形。
userids	string/array	返回所给用户 ID // （单个或者多个）// 的聚合图形。
screenitemids	string/array	返回所给聚合图形项的的聚合图形。
selectUsers	query	返回 `users` 属性中与聚合图形共享的用户。
selectUserGroups	query	返回 `userGroups` 属性中与聚合图形共享的用户组。
selectScreenItems	query	返回聚合图形上使用的聚合图形项。
sortfield	string/array	根据所给参数对结果进行排序 可能的值：`screenid` 和 `name`。
countOutput	boolean	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	

返回值

`(integer/array)` Returns either:

- 一个数组对象
- 查看对象的个数，如果 `countOutput` 参数被使用

示例如下

通过 ID 查看一个聚合图形

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

源

CScreen::get() in *frontends/php/include/classes/api/services/CScreen.php*.

34. 聚合图形项

此类设计用来工作于聚合类型项。

相关对象:

- Screen item

可用方法:

- screenitem.create - 创建一个新的聚合图形监控项
- screenitem.delete - 删除聚合图形监控项
- screenitem.get - 获取一个聚合图形监控项
- screenitem.update - 更新聚合图形监控项
- screenitem.updatebyposition - 更新聚合图形监控项在一个特殊的屏幕尺寸

> 对象

以下类直接关联到 `screenitem` API。

聚合类型项

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。

特性类	描述	
screenitemid	string	*(只读)* 聚合图形项的ID。

特性类	描述	
resourcetype (required)	integer	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

特性类	描述	
screenid (required)	string	监控项所属聚合图形的ID。
application	string	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 ”Trig-gers overview”.

特性类	描述	
colspan	integer	屏幕项将跨越的列数。 默认：1
dynamic	integer	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	每行可以展示聚合图像项的个数。 默认：25

特性类	描述	
halign	integer	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	聚合图像项的高度，单位是像素 默认：200

特性类	描述	
max_columns	integer	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

特性类	描述	
resourceid	string	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

特性类	描述	
rowspan	integer	屏幕项目将跨越的行数或行数。 Default: 1.

特性类	描述	
sort_triggers	integer	动作和触发器必须排序。 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

特性类	描述	
style	integer	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

特性类	描述	
url	string	将显示在聚合图形项中的网页的URL。 用于URL屏幕项。
valign	integer	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

特性类	描述	
width	integer	聚合图形项的宽度，单位是像素 默认是：320
x	integer	屏幕上的屏幕项的x坐标，从左到右。 默认：0
y	integer	屏幕上的屏幕项的y坐标，从左到右。 默认：0

创建

描述

```
object screenitem.create(object/array screenItems)
```

此方法允许创建一个新的聚合类型项

参数

(object/array) 创建聚合图形项。

此方法接受聚合图形项关于standard screen item properties。

返回值

(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。

示例如下

创建一个聚合图形项

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
}
```

猜你想看

- screen.update

源

CScreenItem::create() in *frontends/php/include/classes/api/services/CScreenItem.php*.

删除

描述

object screenitem.delete(array screenItemIds)

此方法允许删除一个聚合图形项

参数

(array) 删除聚合图形项的 IDs

返回值

(object) 返回在 screenitemids 属性下包含已删除聚合图形项的 id 的对象。

示例如下

删除多个聚合图像项

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
}
```

猜你想看

- screen.update

源

CScreenItem::delete() in *frontends/php/include/classes/api/services/CScreenItem.php*.

聚合图形监控项. 更新位置信息

描述

`object screenitem.updatebyposition(array screenItems)`

此方法允许在给定聚合图形单元格中更新聚合图形监控项。如果聚合图形单元格为空，将创建一个新的聚合图形。

参数

(array) Screen item properties to be updated.

必须为每个聚合图形监控项定义” x ”，” y ”和” screenid ”属性，所有其他属性都是可选的。只有传递的属性将被更新，其他所有属性将保持不变。

返回值

(object) 返回一个对象，其中包含在 “screenitemids” 属性下更新和创建的屏幕项目的 id。

例子

更改聚合图形监控项源 ID

Change the resource ID for the screen element located in the upper-left cell of the screen. 更改位于屏幕左上角单元格中的聚合图形中元素的资源 ID。

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
}
```

猜你想看

- screenitem.update

源

CScreenItem::update() in *frontends/php/include/classes/api/services/CScreenItem.php*.

聚合图形监控项更新

描述

`object screenitem.update(object/array screenItems)`

此方法允许更新现有的聚合图形监控项。

Parameters

(object/array) Screen item properties 被更新.

必须为每个聚合图形监控项定义” screenitemid ” 属性，所有其他属性都是可选的。只有传递的属性将被更新，其他所有属性将保持不变。

返回值

(object) 返回一个对象，该对象包含在 “screenitemids” 属性下更新的聚合图形监控项的 ID。

例子

设置聚合图形监控项的大小

设置一个宽 500px，高 300px 的聚合图形监控项。

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
}
```

猜你想看

- screenitem.updatebyposition

源

CScreenItem::update() in *frontends/php/include/classes/api/services/CScreenItem.php*.

聚合图形监控项获取

描述

`integer/array screenitem.get(object parameters)`

该方法允许根据给定的参数检索聚合图形监控项。

参数

`(object)` 参数定义所需的输出。

该方法支持以下参数。

Parameter	Type	Description
screenitemids	string/array	只返回具有给定 id 的聚合图形监控项。
screenids	string/array	只返回属于给定聚合图形的聚合图形监控项。
sortfield	string/array	根据给定的属性对结果排序. 可能值: `screenitemid` 和 `screenid`.
countOutput	boolean	对于所有 “get” 方法，这些参数都是通用的，在reference commentary page 页面。
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:

- 一个对象数组;
- 检索对象的计数，如果使用了 “countOutput” 参数。

例子

从聚合图形中查寻聚合图形监控项

查询聚合图形监控项的所有聚合图形。

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
}
```


源

CScreenItem::get() in *frontends/php/include/classes/api/services/CScreenItem.php*.

35. 脚本

这个方法设计，作用于脚本。

相关对象:

- Script

可用方法:

- script.create - 创建一个新的脚本
- script.delete - 删除脚本
- script.execute - 运行脚本
- script.get - 获取脚本
- script.getscriptsbyhosts - 获取主机脚本
- script.update - 更新脚本

> 对象

以下对象直接关联到 `script` API。

脚本

这个脚本对象拥有以下属性。

Property	Type	Description
scriptid	string	*(readonly)* 脚本的 ID。
command (required)	string	运行命令。
name (required)	string	脚本名称。
confirmation	string	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	脚本描述。
execute_on	integer	哪里去运行这个脚本 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
groupid	string	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。 默认: 0.
host_access	integer	运行脚本主机的权限 可能的值： 2 - *(默认)* 读 3 - 写
type	integer	脚本类型 可能的值： 0 - *(默认)* 脚本 1 - IPMI
usrgrpid	string	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 默认: 0。

创建

描述

`object script.create(object/array scripts)`

此方法允许创建一个新的脚本

参数

`(object/array)` Scripts to create.

The method accepts scripts with the standard script properties.

返回值

`(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。

示例如下

创建一个脚本

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
}
```

源

CScript::create() in *frontends/php/include/classes/api/services/CScript.php*.

删除

描述

`object script.delete(array scriptIds)`

此方法允许去删除脚本。

参数

(array) 返回删除脚本的 ID

返回值

(object) 返回一个对象包含在 scriptids 属性之下删除的脚本

示例如下

删除多个脚本

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
}
```

源

CScript::delete() in *frontends/php/include/classes/api/services/CScript.php*.

更新

描述

`object script.update(object/array scripts)`

此方法更新已存在的脚本。

参数

`(object/array)` Script properties to be updated.

`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。

返回值

`(object)` 返回一个对象包含在 `scriptids` 属性下更新脚本的 ID。

示例如下

改变一个脚本的命令

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
}
```

源

CScript::update() in *frontends/php/include/classes/api/services/CScript.php*.

脚本运行

描述

`object script.execute(object parameters)`

此方法允许在一个主机上运行一个脚本。

参数

`(object)` 参数包含要运行脚本的 id 和主机的 id。

Parameter	Type	Description
hostid (required)	string	要运行脚本的主机 ID。
scriptid (required)	string	要运行脚本的脚本 ID。

返回值

(object) 返回脚本执行的结果。

Property	Type	Description
response	string	脚本是否执行成功 可能的值：成功或 失败。
value	string	脚本的输出结果。

示例如下

运行一个脚本

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
}
```

源

CScript::execute() in *frontends/php/include/classes/api/services/CScript.php*.

获取

描述

`integer/array script.get(object parameters)`

此方法允许检索符合所给参数的脚本。

参数

(object) 定义所需输出的参数。

此方法支持以下参数。

Parameter	Type	Description
groupids	string/array	仅能运行在所给主机组的脚本。
hostids	string/array	仅能运行在所给主机的脚本。
scriptids	string/array	仅返回所给 ID 的脚本。
usrgrpids	string/array	仅返回所给用户组可以运行的脚本。
selectGroups	query	返回可以在 `groups` 属性中运行脚本的主机组。
selectHosts	query	返回可以在 `hosts` 属性中运行脚本的主机组。
sortfield	string/array	根据所给参数对参数进行排序 可能的值：`scriptid` 和 `name`。
countOutput	boolean	这些参数对于所有 “get” 方法都是通用的，在reference commentary 中有详细描述。

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	

返回值

(integer/array) Returns either:

- 一个数组对象。
- 检索到对象的数目，如果 countOutput 参数被使用。

示例如下

检索所有脚本

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
}
```

猜你想看

- Host
- Host group

源

CScript::get() in *frontends/php/include/classes/api/services/CScript.php*.

通过主机获取脚本

描述

`object script.getscriptsbyhosts(array hostIds)`

此方法允许检索适用所给主机的脚本。

参数

(string/array) IDs of hosts to return scripts for.（string/array）主机 ID

返回值

（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> 该方法将在 `confirmation` 文本中自动扩展宏。:::

示例如下

通过主机的 ID 检索脚本

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
}
```


源

CScript::getScriptsByHosts() in *frontends/php/include/classes/api/services/CScript.php*.

36. 服务

该类用于配合服务使用。

对象引用:

- 服务
- 服务时间
- 服务依赖
- 服务告警

可用方法:

- service.adddependencies - 增加 IT 服务之间的依赖关系
- service.addtimes - 增加服务时间
- service.create - 创建新的 IT 服务
- service.delete - 删除 IT 服务
- service.deletedependencies - 删除 IT 服务之间的依赖关系
- service.deletetimes - 删除服务时间
- service.get - 检索 IT 服务
- service.getsla - 检索有关 IT 服务的可用性信息
- service.update - 更新 IT 服务

> 对象

以下对象与 `serviceAPI` 直接相关。

Service 服务

服务对象具有以下属性。

属性类	说明	
serviceid	字符串 *(	读)* 服务的 ID。

属性类	说明	
algorithm (required 必须)	整数型用于	算服务状态的算法。 可能的值： 0 - 不计算； 1 - 问题，至少有一个子项有问题。 2 - 问题，所有子项都有问题。
name (required 必须)	字符串服务	名称。
showsla (required 必须)	整数型是否	计算SLA。 可能的值： 0 - 不计算； 1 - 计算。
sortorder (required 必须)	整数型用于	序服务的位置。

属性类	说明	
goodsla	浮点数最低	接受的SLA值，如果SLA降低，则该服务被认为处于有问题状态。 默认值：99.9。

属性类	说明
status	整数型 *(读)* 服务是否处于正常或故障状态。 如果服务处于故障状态，status相当于以下情况之一： - 如果告警级别设置值为2，那么链接触发器的告警级别为”警告”或更高(忽

属性类	说明	
	字符串与服	相关联的触发器只能设置在没有子项的服务上。 默认：0

Service time 服务时间

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。

属性类	说明	
timeid	字符串 *(	读)*服务时间的ID。
serviceid (必须)	字符串服务	ID。不可更新。

属性类	说明	
ts_from (必须)	整数型服务	间生效的时间。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一一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。

属性类	说明	
ts_to (必须)	整数型服务	间关闭的时间。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一一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。

属性类	说明	
type (必须)	整数型服务	间类型可能的值：0 - 计划开机，每周重复；1 - 计划停机，每周重复；2 - 一次性停机。
note	字符串有关	务时间的附加信息。

服务依赖

服务依赖对象表示服务之间的依赖关系，它具有以下属性。

属性类	说明	
linkid	字符串 *(	读)* 服务依赖的ID。

属性类	说明
servicedownid (必须)	字符串被子务依赖的服务ID，一个服务可以有多个子服务。
serviceupid (必须)	字符串依赖父服务的服务ID，一个服务可以有多个父服务，从而形成一张定向图表。

属性类	说明	
soft (必须)	整数型服务	间的依赖关系类型。可能的值：0 - 硬依赖；1 - 软依赖。 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

属性类	说明

服务告警

Note:
不能通过 Zabbix API 直接创建，更新或删除服务告警。

服务告警对象代表服务的状态变化，它具有以下属性。

属性类	说明	
servicealarmid	字符串服务	警的 ID。
serviceid	字符串服务	ID。
clock	时间戳服务	态发生变化的时间。
value	整数型服务	状态。 请参阅service status property以获取许可值列表。

创建

说明

`object service.create(object/array services)`

此方法允许创建新的服务。

参数

`(object/array)` 创建服务。

除标准服务属性之外，该方法接受以下参数。

参数类	说明	
dependencies	数组服	依赖。 每个服务依赖项具有以下参数: - dependsOnServ - *(string* 字符串*)* 被子服务依赖的服务ID。 - soft - *(*整数型*)* 有关依赖关系类型的更多信息，请参阅服务依赖。

参数类	说明	
parentid	字符串硬链	的父服务的ID。
times	数组为	务创建的服务时间。

返回值

(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。

范例

创建服务

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%。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

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

说明

```
object service.delete(array serviceIds)
```

此方法允许删除服务。

与子级服务有硬依赖关系的服务无法被删除。

参数

(array) 要删除的服务 ID。

返回值

返回一个对象，该对象包含在 serviceids 属性中被删除服务的 ID。

示例

删除多个服务

删除两个服务。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.delete",
    "params": [
        "4",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

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

说明

object service.deletedependencies(string/array serviceIds)

此方法允许从服务中删除所有依赖关系。

参数

(string/array) 删除所有依赖关系的服务 ID。

返回值

(object) 返回一个对象，该对象包含在 serviceids 属性中受影响服务的 ID。

示例

从服务中删除依赖关系

从服务"2" 中删除所有依赖项。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.deletedependencies",
```

```
    "params": [
        "2"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "2"
        ]
    },
    "id": 1
}
```

参考

- service.update

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

说明

`object service.deletetimes(string/array serviceIds)`

此方法允许从服务中删除所有服务时间。

参数

(`string/array`) 删除所有服务时间的服务 ID。

返回值

(`object`) 返回一个对象，该对象包含在 `serviceids` 属性中受影响服务的 ID。

实例

从服务中删除服务时间

从服务"2" 中删除所有服务时间。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.deletetimes",
    "params": [
        "2"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "2"
        ]
    },
```

```
    "id": 1
}
```

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

说明

`object service.update(object/array services)`

此方法允许更新现有服务。

参数

(`object/array`) 需要更新的服务属性。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。除标准服务属性之外，该方法接受以下参数。

参数类	说明	
dependencies	数组用	替换当前内容的服务依赖关系。 每个服务依赖项具有以下参数： - `dependsOnServi` - *(字符串)* 依赖服务的服务，即子服务的ID。 - `soft` - *(整数型)* 服务依赖类型；有关依赖关系类型的

参数类	说明	
parentid	字符串硬链	的父服务ID。
times	数组用	替换当前内容的服务时间。

返回值

(object) 返回一个对象，该对象包含在 serviceids 属性中已更新服务的 ID。

示例

设置父服务

使服务"3" 硬链接于父服务"5"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "5",
        "parentid": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

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。
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。

Parameter 参数 T	pe 类型 Des	ription 说明
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。

37. 任务

此类用于管理任务。

可用方法：

- task.create - 创建新的任务。

> 对象

以下对象与 任务 API 直接相关。

<note note> 本节描述任务对象是 Zabbix 5.0.5 及以后版本才支持的。在旧的 Zabbix 版本中正确使用 `task.create`，请参阅任务. 创建文档。:::

任务对象支持的属性如下:

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
taskid	字符 *	只读)* 任务 ID。
type (必须)	整型任	类型。可以是如下值: 1 - 诊断信息; 6 - 立即检查。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
status	整型 *	只读)* 任务状态。 可以是如下值: 1 - 任务新建; 2 - 任务进行中; 3 - 任务完成; 4 - 任务过期。
clock	时间戳 *(	读)* 任务创建时间。
ttl	整型 *	只读)* 任务过期时间(秒)。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
proxy_hostid	字符收	到的诊断信息统计的代理的ID.不含’立即检查’任务。
request (必须)	对象任’	请求对象:即检查’任务对象参照详细描述如下; ’ 诊断信息’任务对象参照详细描述如下。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
result	对象 *	只读)* 诊断信息任务的结果对象。若结果尚未完成会包含空。结果对象详细描述如下。

’ 立即检查’ 请求对象

’ 立即检查’ 任务请求对象有如下属性。

属性 [	型](/zh//manual/api/reference_commentary#data_types) 描述	
itemid	字符监	项和低级别发现规则的 ID。

’ 诊断信息’ 请求对象

诊断信息任务请求对象支持的属性如下。所有属性类型的统计请求对象请参阅详细描述如下。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
historycache	对象历	缓存的统计请求。用于服务端和代理端。
valuecache	对象监	项缓存的统计请求。用于服务端。
preprocessing	对象预	理管理的统计请求。用于服务端和代理端。
alerting	对象告	管理的统计请求。用于服务端。
lld	对象低	别发现的统计请求。用于服务端。

统计请求对象

统计请求对象用于定义应该收集哪些类型的关于服务器/代理内部进程的信息。支持如下属性。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
stats	查询返

的统计对象属性。诊断信息统计类型可用的字段列表参照详细描述如下。

缺省: 所有可用统计字段会返回 `extend`。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	

top	对象统	值返回的排序和限制对象。诊断信息统计类型可用的字段列表参照详细描述如下。
		
		例如:
		{
		“source.alerts”
		10
		}

诊断信息请求类型可用的统计字段列表

各类型的诊断信息请求属性请求可用下列统计字段。

诊断类型可用字	描述	
historycache	items	监控项缓存数量。
	values	值缓存数量。
	memory	共享内存统计 (剩余空间, 使用块数, 剩余块数, 最大剩余块数)。
	memory.data	共享内存缓存统计的历史数据。
	memory.index	共享内存缓存统计的历史索引。
valuecache	items	监控项缓存的数量。
	values	值缓存数量。
	memory	共享内存统计 (剩余空间, 使用块数, 剩余块数, 最大剩余块数)。
	mode	缓存模式。
preprocessing	values	入队值的数量。
	preproc.values	带预处理步骤值的入队数量。
alerting	alerts	入队告警的数量。
lld	rules	入队规则的数量。
	values	入队值的数量。

诊断信息请求类型可用的排序字段列表

如下统计字段可用于排序和限制请求的信息。

诊断类型可用字	[类型](/z	/manual/api/reference_commentary#data_types)
historycache	values	整型
valuecache	values	整型
	request.values	整型
preprocessing	values	整型
alerting	media.alerts	整型
	source.alerts	整型
lld	values	整型

统计结果对象

统计结果对象由任务对象的 `result` 字段获取。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述
status	整型 * 只读)* 任务结果的状态。 可以是如下值: - 1 - 任务执行时产生的错误; - 0 - 任务结果已创建。

属性 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
data	字符/对象结果根	特定诊断信息任务的统计请求对象的结果。若任务执行产生错误时会包含错误信息。

创建

说明

`object task.create(object task)`

该方法允许创建新的任务。

参数

(`object`) 需要创建的任务。此方法接受以下参数。

参数类	说明	
type (必须)	整数型任务	型。许可值：6 - 正在核实。

参数类	说明
itemids (必须)	字符串/数组监控项和级别发现规则的ID。监控项或自动发现规则必须是以下类型：0 - Zabbix agent；1 - SNMPv1客户端；3 - 简单检查；4 - SNMPv2客户端；5 - Zabbix内部；6 - SNMPv3客户端；8 - Zabbix整合；10 - 外部

参数类	说明

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。

Source

CTask::create() in *ui/include/classes/api/services/CTask.php*.

获取

描述

integer/array task.get(object parameters)

task.get 方法可以根据给定的参数获取 task 信息。方法只返回有关 “诊断信息” 任务的详细信息。

<note note> 该方法在 Zabbix 5.0.5 以后的版本可用。:::

对于非超级管理员用户，方法返回权限不足消息。

参数

(object) 参数定义了所需的输出。

该方法支持以下参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
taskids	string/array	返回给定 ID 的 task 信息。
output	query	这些参数对于所有 “get” 方法都是通用的，在参考说明desk 中有
preservekeys	boolean	

返回值

(integer/array) 返回一个对象的数组。

示例

通过 ID 获取 task

获取 task “1” 的所有数据。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "task.get",
    "params": {
        "output": "extend",
        "taskids": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

相应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "taskid": "1",
            "type": "7",
            "status": "3",
            "clock": "1601039076",
            "ttl": "3600",
            "proxy_hostid": null,
            "request": {
                "alerting": {
                    "stats": [
                        "alerts"
                    ],
                    "top": {
                        "media.alerts": 10
                    }
                },
                "lld": {
                    "stats": "extend",
                    "top": {
                        "values": 5
                    }
                }
            },
            "result": {
                "data": {
                    "alerting": {
                        "alerts": 0,
                        "top": {
                            "media.alerts": []
                        },
                        "time": 0.000663
                    },
```

```
                    "lld": {
                        "rules": 0,
                        "values": 0,
                        "top": {
                            "values": []
                        },
                        "time": 0.000442
                    }
                },
                "status": "0"
            }
        }
    ],
    "id": 1
}
```


更多参考

- 任务
- 对象统计结果

源码

CTask::get() in *ui/include/classes/api/services/CTask.php*.

38. 模板

此类用于管理模板。对象引用:

- 模板

可用方法:

- template.create - 创建新模板
- template.delete - 删除模板
- template.get - 检索模板
- template.massadd - 添加相关对象到模板中
- template.massremove - 从模板中删除相关对象
- template.massupdate - 从模板中替换或删除相关对象
- template.update - 更新模板

> 对象

以下对象与 API 模板直接相关。

Template 模板

模板对象具有以下属性。

参数类	说明	
templateid	string	*(只读)* 模板 ID。
host (必须)	string	模板的正式名称。
description	text	模板说明。
name	string	主机的可见名称。 默认：主机的属性值。

模板标签

模板标签对象具有以下属性。

参数类	说明	
tags	string	模板标签名称。
(必须)		
value	string	模板标签名称。

创建

说明

`object template.create(object/array templates)`

此方法允许创建新模板。

参数

`(object/array)` 创建模板。

除了标准模板属性之外，该方法接受以下属性。

参数类	说明	
groups	object/array 主	模板添加到主机组。
(必须)		组必须定义 `groupid` 属性。
tags	object/array	模板标签。
templates	object/array	被链接到模板的模板。
		模板必须定义 `templateid` 属性。
macros	object/array	为模板创建的用户宏。
hosts	object/array	链接到模板的主机。
		主机必须定义 `hostid` 属性。

返回值

`(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。

范例

创建模板

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
}
```

源码

CTemplate::create() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

删除

说明

`object template.delete(array templateIds)`

此方法允许删除模板。

参数

(array) 需要删除的模板 ID。

返回值

(object) 返回一个对象，包含 templateids 属性中被删除模板的 ID。

范例

删除多个模板

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
}
```

源码

CTemplate::delete() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

批量删除

说明

`object template.massremove(object parameters)`

方法允许从多个模板中删除相关对象。

参数

(object) 参数包含需要更新的模板 ID 以及需要删除的对象。

参数类	说明	
templateids (required 必须)	string/array	将要更新的模板 ID。
groupids	string/array	从指定的模板中删除主机组。
hostids	string/array	从主机或模板中取消指定模板（下游）的链接。
macros	string/array	删除指定模板的用户宏。
templateids_clear	string/array	从指定模板（上游）中取消模板链接并清除数据。
templateids_link	string/array	从指定模板（上游）中取消模板链接。

返回值

(object) 返回一个对象，此对象包含在 `templateids` 中已更新模板的 ID。

范例

从组中删除模板

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
}
```

主机中取消模板链接

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
}
```

参考

- template.update
- User macro

源码

CTemplate::massRemove() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

批量更新

说明

`object template.massupdate(object parameters)`

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。

参数

`(object)` 参数包含需要更新的模板 ID 以及需要更新的属性。

除standard template properties之外，该方法接受以下参数。

参数类	说明	
templates (required 必须)	object/array 模	需要更新的模板。 必须已定义 `templateid` 属性。
groups	object/array	替换所属模板的当前主机组。 主机组必须已定义 `groupid` 属性。
hosts	object/array	替换当前链接模板的主机和模板。 主机和模板都必须使用 `hostid` 属性传递唯一 ID。
macros	object/array	替换指定模板上的当前用户宏。
templates_clear	object/array	从指定模板中取消链接并清除数据。 模板必须已定义 `templateid` 属性。
templates_link	object/array	替换当前链接的模板。 模板必须已定义 `templateid` 属性。

返回值

`(object)` 返回一个对象，此对象包含在 `templateids` 中已更新模板的 ID。

范例

替换主机组

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
}
```

参考

- template.update
- template.massadd
- Host group
- User macro

源码

CTemplate::massUpdate() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

批量添加

说明

`object template.massadd(object parameters)`

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。

参数

`(object)` 参数包含需要更新的模板 ID 以及添加到模板的对象。

该方法接受以下参数。

参数类	说明	
templates (required 必须)	object/array 模	需要更新的模板。 必须定义 `templateid` 属性。
groups	object/array	主机组添加指定的模板。 主机组必须定义 `groupid` 属性。
hosts	object/array	将主机和模板链接到指定的模板中。 主机必须定义 `hostid` 属性。

参数类	说明	
macros	object/array	为指定的模板创建用户宏。
templates_link	object/array	将模板链接到指定模板。 模板必须定义 templateid 属性。

返回值

(object) 返回一个对象，此对象包含在 templateids 属性中已更新模板的 ID。

范例

添加模板到组

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
}
```

链接模板到主机

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
}
```

参考

- template.update
- Host
- Host group
- User macro

源码

CTemplate::massAdd() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

更新

说明

`object template.update(object/array templates)`

此方法允许更新现有模板。

参数

`(object/array)` 需要被更新的模板属性。

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。

除standard template properties之外，该方法接受以下参数。

参数类	说明	
groups	object/array	替换所属模板的当前主机组。 主机组必须已定义 `groupid` 属性。
tags	object/array	模板标签替换当前的模板标签。
hosts	object/array	替换当前链接模板的主机和模板。 主机和模板都必须使用 `hostid` 属性传递唯一 ID。
macros	object/array	替换指定模板上的当前用户宏。
templates	object/array	用于替换当前链接的模板，未通过的模板只是被取消链接。 模板必须已定义 `templateid` 属性。
templates_clear	object/array	从指定模板中取消链接并清除数据。 模板必须已定义 `templateid` 属性。

返回值

(object) 返回一个对象，此对象包含在 templateids 属性中已更新模板的 ID。

范例

重命名模板

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
}
```

更新模板标签

用新模板替换所有模板标签。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10086"
        ]
    },
    "id": 1
}
```

源码

CTemplate::update() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

获取

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。

参数

`(object)` 定义需要输出的参数。

此方法支持以下参数。

参数类	说明	
templateids	string/array	只返回指定模板ID的模板。
groupids	string/array	只返回指定主机组所属的模板。
parentTemplateids	string/array	只返回指定子类模板的模板。

参数类	说明	
hostids	string/array	只返回被链接到指定主机的模板。
graphids	string/array	只返回包含指定图形的模板。
itemids	string/array	只返回包含指定监控项的模板。
triggerids	string/array	只返回包含指定触发器的模板。

参数类	说明	
with_items	flag	只返回具有监控项的模板。
with_triggers	flag	只返回具有触发器的模板。
with_graphs	flag	只返回具有图形的模板。
with_httptests	flag	只返回具有web场景的模板。

参数类	说明	
evaltype	integer	标签搜索规则。可能的值：0 - (default) And/Or; 2 - Or.

参数类	说明	
tags	array/object	仅返回具有给定标签的模板。按标记进行精确匹配，并根据运算符的值按标记值进行区分大小写或不区分大小写的搜索。格式： [{"tag": "<tag>", "value": "<value>", "operator": "<operator>" ...] 空数组将返回

参数类	说明	
selectGroups	query	从 groups 属性中返回所属模板的主机组。
selectTags	query	在 tags 属性中返回模板标签。
selectHosts	query	从 hosts 属性中返回被链接模板的主机。支持 count。
selectTemplates	query	从 templates 属性中返回子类模板。支持 count。

参数类	说明	
selectParentTemplates	query	从 parentTempla 属性中返回父类模板。支持 count。
selectHttpTests	query	从 httpTests 属性中返回来自模板的 web 场景。 支持 count。
selectItems	query	从 items 属性中返回来自模板的监控项。支持 count。

参数类	说明	
selectDiscoveries	query	从 `discoveries` 属性中返回来自模板的低级别发现。 支持 `count`。
selectTriggers	query	从 `triggers` 属性中返回来自模板的触发器。 支持 `count`。
selectGraphs	query	从 `graphs` 属性中返回来自模板的图表。 支持 `count`。

参数类	说明	
selectApplications	query	从 `applications` 属性中返回来自模板的应用。支持 `count`。
selectMacros	query	从 `macros` 属性中返回来自模板的宏。
selectScreens	query	从 `screens` 属性中返回来自模板的聚合图形。支持 `count`。

参数类	说明
limitSelects	integer 限制子查询返回的记录数。 应用于以下子查询： selectTempla - 结果将以 name 排序； selectHosts - 以 host 排序； selectParent - 以 host 排序； selectItems - 以 name 排序； selectDiscov - 以 name 排序； selectTrigge - 以 description 排序； selectGraphs - 以 name 排序； selectApplic

参数类	说明	
sortfield	string/array	根据给定的属性为结果排序。 许可值为： hostid,host,na
countOutput	boolean	这些参数十分普遍，适用所有get方法，详情参见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) 返回两者其中任一：

- 一组对象；
- 如果已经使用了 countOutput 参数，则检索对象的计数。

范例

按名称检索模板

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
}
```


按模板标签搜索

检索标记为 “Host name” 等于 “ {HOST.NAME}” 的模板。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}",
                "operator": 1
            }
        ]
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10402",
            "tags": [
                {
                    "tag": "Host name",
                    "value": "{HOST.NAME}"
                }
            ]
        }
    ],
    "id": 1
}
```

参考

- Host group
- Template
- User macro
- Host interface

源码

CTemplate::get() 方法可在 ui/include/classes/api/services/CTemplate.php 中参考。

39. 聚合图形模板

此类用于配合聚合图形模板的使用。

对象引用：

- Template screen

可用方法：

- templatescreen.copy - 复制聚合图形模板。
- templatescreen.create - 创建新的聚合图形模板。
- templatescreen.delete - 删除聚合图形模板。
- templatescreen.get - 检索聚合图形模板。
- templatescreen.update - 更新聚合图形模板。

> 对象

以下对象与 `templatescreen`API 直接相关。

聚合图形模板

聚合图形模板对象具有以下属性。

属性类	说明	
screenid	string	*(readonly 只读)* 聚合图形模板的 ID。
name (required 必填)	string	聚合图形模板的名称。
templateid (required 必填)	string	聚合图形所属模板的 ID。
hsize	integer	聚合图形模板的宽度。 默认：1

属性类	说明	
vsize	integer	聚合图形模板的高度。 默认：1

创建

说明

`object templatescreen.create(object/array templateScreens)`

此方法允许创建新的聚合图形模板。

参数

(object/array) 要创建的聚合图形模板。

除standard template screen properties之外，该方法接受以下参数。

参数类	说明	
screenitems	array	聚合图形上要创建的聚合图形模板项。

返回值

(object) 返回一个对象，该对象包含在 screenids 属性中已创建聚合图形模板 ID，返回 ID 的顺序与传递聚合图形模板的顺序相匹配。

范例

创建聚合图形模板

创建一个 2 行 3 列名为 “Graphs” 的聚合图形模板，并添加一个图形到左上角的格子内。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.create",
    "params": {
        "name": "Graphs",
        "templateid": "10047",
        "hsize": 3,
        "vsize": 2,
        "screenitems": [
            {
                "resourcetype": 0,
                "resourceid": "410",
                "x": 0,
                "y": 0
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "45"
        ]
    },
    "id": 1
}
```

参考

- Template screen item

源码

CTemplateScreen::create() 方法可在 ui/include/classes/api/services/CTemplateScreen.php 中参考。

删除

说明

`object templatescreen.delete(array templateScreenIds)`

此方法允许删除聚合图形模板。

参数

(array) 需要删除的聚合图形模板 ID。

返回值

(object) 返回一个对象，该对象包含在 screenids 属性中已删除聚合图形模板的 ID。

范例

删除多个聚合图形模板

删除两个聚合图形模板。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.delete",
    "params": [
        "45",
        "46"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "45",
            "46"
        ]
    },
    "id": 1
}
```

源码

CTemplateScreen::delete() 方法可在 ui/include/classes/api/services/CTemplateScreen.php 中参考。

复制

说明

`object templatescreen.copy(object parameters)`

此方法允许将聚合图形模板复制到指定的模板中。

参数

(object) 定义了复制的聚合图形模板参数以及目标模板。

参数类	说明	
screenIds (required 必填)	string/array	需要复制的聚合图形模板 ID。
templateIds (required 必填)	string/array	将聚合图形复制到模板的 ID。

返回值

(boolean) 如果复制成功，则返回 true。

范例

复制聚合图形模板

将聚合图形模板"25" 复制到模板"30085"。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.copy",
    "params": {
        "screenIds": "25",
        "templateIds": "30085"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

源码

CTemplateScreen::copy() 方法可在 ui/include/classes/api/services/CTemplateScreen.php 中参考。

更新

说明

`object templatescreen.update(object/array templateScreens)`

此方法允许更新现有的聚合图形模板。

参数

(object/array) 需要更新的聚合图形模板属性。

必须为每个聚合图形模板定义 screenid 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。

除standard template screen properties之外，该方法接受以下参数。

参数类	说明	
screenitems	array	用来替换现有内容的聚合图形项。 聚合图形项通过坐标轴更新，因此每个聚合图形项必须定义 x 和 y 属性。

返回值

(object) 返回一个对象，该对象包含在 screenids 属性中已更新聚合图形模板的 ID。

范例

重命名聚合图形模板

将聚合图形模板重命名为”Performance graphs”。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.update",
    "params": {
        "screenid": "3",
        "name": "Performance graphs"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "3"
        ]
    },
    "id": 1
}
```

源码

CTemplateScreen::update() 方法可在 ui/include/classes/api/services/CTemplateScreen.php 中参考。

获取

说明

`integer/array templatescreen.get(object parameters)`

此方法允许根据指定的参数来检索聚合图形模板。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数类	说明	
hostids	string/array	仅返回指定主机所属的聚合图形模板。
screenids	string/array	仅返回指定 ID 的聚合图形模板。
screenitemids	string/array	仅返回包含指定聚合图形项的聚合图形模板。
templateids	string/arary	仅返回指定模板所属的聚合图形模板。
noInheritance	flag	不返回继承的聚合图形模板。
selectScreenItems	query	返回 `screenitems` 属性中聚合图形模板使用的聚合图形项。
sortfield	string/array	按指定的属性对结果分类。 许可值为：`screenid` 和 `name`。
countOutput	boolean	这些参数非常普遍，适用于所有的 `get` 方法，详情可在reference commentary中参考。
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

返回两者其中任一：

- 一组对象；
- 如果已经使用了 `countOutput` 参数，则检索对象的计数。

范例

从模板中检索聚合图形

从模板"10001"中检索所有聚合图形以及检索所有聚合图形项。

Request 请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.get",
    "params":  {
        "output": "extend",
        "selectScreenItems": "extend",
        "templateids": "10001"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response 响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenid": "3",
            "name": "System performance",
            "hsize": "2",
            "vsize": "2",
            "templateid": "10001",
            "screenitems": [
                {
                    "screenitemid": "20",
                    "screenid": "3",
                    "resourcetype": "0",
                    "resourceid": "433",
                    "width": "500",
                    "height": "120",
                    "x": "0",
                    "y": "0",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                },
                {
                    "screenitemid": "21",
                    "screenid": "3",
                    "resourcetype": "0",
                    "resourceid": "387",
                    "width": "500",
                    "height": "100",
                    "x": "0",
                    "y": "1",
                    "colspan": "1",
                    "rowspan": "1",
```

```
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                },
                {
                    "screenitemid": "22",
                    "screenid": "3",
                    "resourcetype": "1",
                    "resourceid": "10013",
                    "width": "500",
                    "height": "148",
                    "x": "1",
                    "y": "0",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                },
                {
                    "screenitemid": "23",
                    "screenid": "3",
                    "resourcetype": "1",
                    "resourceid": "22181",
                    "width": "500",
                    "height": "184",
                    "x": "1",
                    "y": "1",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                }
            ]
        }
    ],
    "id": 1
}
```


参考

- Template screen item

源码

CTemplateScreen::get() 方法可在 ui/include/classes/api/services/CTemplateScreen.php 中参考。

40. 聚合图形项模板

此类用于配合聚合图形项模板的使用。

对象引用：

- Template screen item

可用方法：

- templatescreenitem.get - 检索聚合图形项模板。

> 对象

以下对象与 `templatescreenitem` API 直接相关。

Template screen item 聚合图形项模板

聚合图形项模板对象定义了显示在聚合图形模板上的元素，它具有以下属性。

特性类	描述	
screenitemid	string	*(只读)* 聚合图形项模板的ID。

特性类	描述	
resourceid (必需)	string	源自显示在聚合图形项模板上父模板的对象ID。根据聚合图形项的类型，`resourceid`属性可引用不同的对象。聚合图形项模板中的时钟和URL类型无法使用。 注意*:* 即使聚

特性类	描述	
resourcetype (必需)	integer	聚合图形项模板类型。 可能值： 0 - 图形； 1 - 简单图形； 3 - 纯文本； 7 - 时钟； 11 - URL； 19 - 简单图形原型； 20 - 图形原型。
screenid (必需)	string	所属聚合图形模板的ID。

特性类	描述	
colspan	integer	聚合图形项模板所跨的列数。 默认值：1。
elements	integer	聚合图形项模板所显示的行数。 默认值：25。

特性类	描述	
halign	integer	指定聚合图形项模板如何在单元格中水平对齐。 可能值：0 - (默认值) 居中；1 - 左对齐；2 - 右对齐。
height	integer	聚合图形项模板的高度(以像素为单位)。 默认值：200。

特性类	描述	
max_columns	integer	指定图形原型或简单图形原型的聚合图形元素具有的最大列数。 默认值：3。
rowspan	integer	聚合图形项模板所跨的行数。 默认值：1。

特性类	描述	
style	integer	聚合图形项模板显示选项。 聚合图形项时钟的可能值： 0 - *(默认值)* 当地时间； 1 - 服务器时间； 2 - 主机时间。 聚合图形项纯文本的可能值： 0 - *(默认值)* 以纯文本显示

特性类	描述	
url	string	在聚合图形项模板中显示网页的URL，由URL聚合图形项模板使用。
valign	integer	指定聚合图形项模板如何在单元格中垂直对齐。 可能值：0 - *(默认值)* 居中；1 - 置顶；2 - 底部。

特性类	描述	
width	integer	聚合图形项模板宽度(以像素为单位)。 默认值：320。
x	integer	在聚合图形上聚合图形项模板的X轴坐标，从左到右计数。 默认值：0。

特性类	描述	
y	integer	在聚合图形上聚合图形项模板的Y轴坐标，从上到下计数。 默认值：0。

获取

描述

`integer/array templatescreenitem.get(object parameters)`

此方法允许根据指定的参数检索聚合图形项模板。

参数

`(object)` 定义所需输出的参数。该方法提供以下参数。

参数类	描述	
screenids	string/array	仅返回指定所属聚合图形模板的聚合图形项模板ID。
screenitemids	string/array	仅返回指定ID的聚合图形项模板。

参数类	描述	
hostids	string/array	为每个聚合图形项模板返回一个额外的“real_resourceid属性，该属性属于指定主机或模板的聚合图形。real_resourc属性包含显示在聚合图形中的对象ID。

参数类	描述	
sortfield	string/array	按给定属性对结果排序。 可能值：`screenitemid`和`screenid`。
countOutput	boolean	这些参数很常用，适用于所有`get`方法，详情可参考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)` 返回两者其中任一：

- 一组对象；
- 若已使用了 `countOutput` 参数，则检索对象的计数。

示例

为聚合图形检索聚合图形项模板

从聚合图形模板"15" 中返回所有聚合图形项模板。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreenitem.get",
    "params": {
        "output": "extend",
        "screenids": "15"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenitemid": "42",
            "screenid": "15",
            "resourcetype": "0",
            "resourceid": "454",
            "width": "500",
            "height": "200",
            "x": "0",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "43",
            "screenid": "15",
            "resourcetype": "0",
            "resourceid": "455",
            "width": "500",
            "height": "270",
            "x": "1",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "max_columns": "3"
        }
    ],
    "id": 1
}
```

来源

CTemplateScreenItem::get() in *frontends/php/include/classes/api/services/CTemplateScreenItem.php*. CTemplateScreenItem::get() 方法可在 *frontends/php/include/classes/api/services/CTemplateScreenItem.php* 中参考。

41. 趋势

此类用于处理趋势数据。

对象引用：

- Trend

可用方法：

- trend.get - 检索趋势数据。

> 对象

以下对象与 `trend` API 直接相关。::: noteclassic

趋势对象根据监控项类型信息而有所不同，它们由 Zabbix server 创建，不能通过 API 进行修改。:::

Float trend 浮点型趋势

浮点型趋势对象具有以下特性。

特性类	描述	
clock	timestamp	获取该值的时间。
itemid	string	相关监控项 ID。
num	integer	在该小时内数值。
value_min	float	每小时最小值。
value_avg	float	每小时平均值。
value_max	float	每小时最大值。

整数型趋势

整数型趋势对象具有以下特性。

特性类	描述	
clock	timestamp	T 获取该值的时间。
itemid	string	相关监控项 ID。
num	integer	在该小时内的数值。
value_min	integer	每小时最小值。
value_avg	integer	每小时平均值。
value_max	integer	每小时最大值。

获取

描述

`integer/array trend.get(object parameters)`

该方法用于根据指定的参数检索趋势数据。

参数

`(object)` 定义所需输出的参数。该方法提供以下参数。

参数类	描述	
itemids	string/array	仅返回指定监控项 ID 的趋势。
time_from	timestamp	仅返回指定时间（包含）之后已采集的值。
time_till	timestamp	仅返回指定时间（包含）之前已采集的值。
countOutput	boolean	计算检索对象的数量。
limit	integer	限制检索对象的数量。
output	query	输出设置的字段。

返回值

(integer/array) 返回两者其中任一：

- 一组对象；
- 若已使用 `countOutput` 参数，则检索对象的计数。

示例

检索监控项趋势数据

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
}
```

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
}
```

来源

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。

42. 触发器

此类用于管理触发器。

对象引用：

- Trigger

可用方法：

- trigger.adddependencies - 添加新的触发器依赖

- trigger.create - 创建新的触发器
- trigger.delete - 删除触发器
- trigger.deletedependencies - 删除触发器依赖
- trigger.get - 检索触发器
- trigger.update - 更新触发器

> 对象

以下对象与 `trigger`API 直接相关。

Trigger 触发器

触发器对象具有以下属性。

属性类	说明	
triggerid	string	*(只读)* 触发器的 ID。
description (必须)	string	触发器的名称。
expression (必须)	string	生成的触发表达式。
comments	string	触发器的附加说明。

属性类	说明	
error	string	*(只读)* 错误概述，如果在更新触发器的状态时出现任何问题。
flags	integer	*(只读)* 原始触发器。 许可值为： 0 - *(默认)* 普通触发器； 4 - 自动发现的触发器。

属性类	说明	
lastchange	timestamp	*(只读)* 触发器最后更改其状态的时间。
priority	integer	触发器的严重性级别。 许可值为： 0 - *(默认)* 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。

属性类	说明	
state	integer	(只读) 触发器的状态。 许可值： 0 - (默认) 触发器状态是最新的； 1 - 当前的触发器状态是未知的。

属性类	说明	
status	integer	触发器是否处于启用状态或禁用状态。 许可值为： 0 - *(默认)* 启用； 1 - 禁用。
templateid	string	*(只读)* 父触发器模板ID。

属性类	说明	
type	integer	触发器是否能够生成多个故障事件。 许可值为： 0 - (默认) 不生成多个事件。 1 - 生成多个事件。
url	string	与触发器相关联的URL。

属性类	说明	
value	integer	(只读) 触发器是否处于正常或故障状态。 许可值为：0 - (默认) OK; 正常；1 - 故障。
recovery_mode	integer	事件恢复生成模式。 许可值为：0 - (默认) 表达式；1 - 恢复表达式；2 - 无。

属性类	说明	
recovery_expression	string	生成的触发恢复表达式。
correlation_mode	integer	事件恢复关闭。 许可值为： 0 - (默认) 所有故障； 1 - 与标签值匹配的所有故障。
correlation_tag	string	用于匹配的标签。

属性类	说明	
manual_close	integer	允许手动关闭。 许可值为： 0 - (默认) 不允许； 1 - 允许。

Trigger tag

The trigger tag object has the following properties.

Property	Type	Description
tag (required)	string	Trigger tag name.
value	string	Trigger tag value.

创建

说明

`object trigger.create(object/array triggers)`

此方法允许创建新的触发器。

参数

`(object/array)` 需要创建的触发器。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 数组依	触发的触发器。 触发器必须已定义 `triggerid` 属性。
tags	array 数组 T	igger tags. 触发器标签。

Attention:
指定的触发器表达式必须为展开式。

返回值

`(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。

范例

创建触发器

创建具有单个触发依赖关系的触发器。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 1
}
```

源码

CTrigger::create() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

删除

说明

`object trigger.delete(array triggerIds)`

此方法允许删除触发器。

参数

(array) 需要删除的触发器 ID。

返回值

(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。

范例

删除多个触发器

删除两个触发器。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.delete",
    "params": [
        "12002",
        "12003"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

源码

CTrigger::delete() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

删除依赖

说明

object trigger.deletedependencies(string/array 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。

参数

(string/array) 需要从触发依赖中删除的触发器。

返回值

(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。

范例

从多个触发器中删除依赖关系

从两个触发器中删除所有依赖关系。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.deleteDependencies",
    "params": [
        {
            "triggerid": "14544"
        },
        {
```

```
            "triggerid": "14545"
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14544",
            "14545"
        ]
    },
    "id": 1
}
```

参考

- trigger.update

源码

CTrigger::deleteDependencies() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

更新

说明

`object trigger.update(object/array triggers)`

此方法用于更新目前的触发器。

参数

`(object/array)` 需要更新的触发器属性。`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。除standard trigger properties之外，该方法接受以下参数。

参数类	说明	
dependencies	array 数组依	触发的触发器。 触发器必须已定义 `triggerid` 属性。
tags	array 数组触	器标签。

Attention:
指定的触发器表达式必须为展开式。

返回值

`(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。

范例

启用触发器

启用触发器，即将其状态设置为 0。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "status": 0
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

替换触发器标签

为触发器替换标签。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

参考

- trigger.adddependencies
- trigger.deletedependencies

源码

CTrigger::update() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

添加依赖

说明

`object trigger.adddependencies(object/array triggerDependencies)`

此方法允许创建新的触发器依赖关系。

参数

`(object/array)` 需要创建的触发器依赖。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 (必须)	string	依赖触发器的 ID。
dependsOnTriggerid (必须)	string	依赖触发的触发器 ID。

返回值

`(object)` 返回一个对象，该对象包含在 `triggerids` 属性中依赖触发器的 ID。

范例

添加触发器依赖

触发器"14092" 依赖于触发器"13565"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.adddependencies",
    "params": {
        "triggerid": "14092",
        "dependsOnTriggerid": "13565"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14092"
        ]
    },
    "id": 1
}
```

参考

- trigger.update

Source 源码

CTrigger::addDependencies() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

获取

说明

`integer/array trigger.get(object parameters)`

此方法允许根据指定的参数检索触发器。

参数

`(object)` 定义需要输出的参数。该方法支持以下参数。

参数类	说明	
triggerids	string/array	仅返回指定ID的触发器。
groupids	string/array	仅返回来自指定主机组中所属主机的触发器。
templateids	string/array	仅返回指定模板所属的触发器。
hostids	string/array	仅返回指定主机所属的触发器。

参数类	说明	
itemids	string/array	仅返回包含指定监控项的触发器。
applicationids	string/array	仅返回来自指定应用集中包含监控项的触发器。

参数类	说明	
functions	string/array	仅返回使用指定函数的触发器。 有关支持的功能列表，请参阅supported trig-ger func-tions页面。
group	string	仅返回来自指定名称的主机组中所属主机的触发器。

参数类	说明	
host	string	仅返回指定名称的所属主机的触发器。
inherited	boolean	仅返回从模板继承的触发器，如果设置为`true`。
templated	boolean	仅返回所属模板的触发器，如果设置为`true`。

参数类	说明	
monitored	flag	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	仅返回所属被监控主机的已启用触发器。

参数类	说明	
maintenance	boolean	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	仅返回事件未确认的触发器。
withAcknowledgedEvents	flag	仅返回所有事件已确认的触发器。

参数类	说明	
withLastEventUnacknowledged	flag	仅返回最后一个未确认事件的触发器。

参数类	说明	
skipDependent	flag	依赖其他触发器的触发器处在故障状态时就跳过。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。

参数类	说明	
lastChangeSince	timestamp	仅返回指定时间之后变更状态的触发器。
lastChangeTill	timestamp	仅返回指定时间之前变更状态的触发器。
only_true	flag	仅返回最近处于故障状态的触发器。

参数类	说明	
min_severity	integer	仅返回严重级别大于或等于指定严重级别的触发器。
expandComment	flag	展开触发器描述中的宏。
expandDescription	flag	展开触发器名称中的宏。
expandExpression	flag	展开在触发器表达式中的函数和宏。

参数类	说明	
selectGroups	query	返回在groups属性中触发器所属的主机组。
selectHosts	query	返回在hosts属性中触发器所属的主机。
selectItems	query	返回在items属性中触发器所包含的监控项。

参数类	说明	
selectFunctions	query	返回在 `functions` 属性中在触发器中使用的函数。 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)* 函数的 ID； `itemid` - *(string)* 使用在函数中的监控项 ID。

参数类	说明	
selectDependencies	query	返回在dependencies属性中依赖触发的触发器。
selectDiscoveryRule	query	返回创建了触发器的低级别发现规则。
selectLastEvent	query	返回在lastEvent属性中最后一个重要触发事件。

参数类	说明	
selectTags	query	返回在tags属性中触发器标签。

参数类	说明	
selectTriggerDiscovery	query	返回在triggerDisco属性中触发器发现对象。触发器发现对象将触发器链接到创建它的触发器原型上。 触发器发现对象具有以下属性： parent_trigg - (string) 创建触发器的触发器原

参数类	说明	
filter	object	仅返回与指定筛选完全匹配的结果。 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。 支持额外的筛选： `host` - 触发器所属主机的正式名称。

参数类	说明	
limitSelects	integer	限制子查询返回的记录数量。 适用于以下子查询： `selectHosts` - 以 `host` 分类结果。
sortfield	string/array	Sort 由指定属性分类结果。 许可值为：`triggerid`, `description`, `status`, `priority`, `lastchange` 和 `hostname`。

参数类	说明	
countOutput	boolean	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	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回两者其中任一：

- 一组对象；
- 如果已经使用了 countOutput 参数，则检索对象的计数。

范例

根据触发器 ID 检索数据

检索触发器"14062" 中使用的所有数据和功能。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "triggerids": "14062",
        "output": "extend",
        "selectFunctions": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```


检索在故障状态的触发器

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。

请求:


```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description",
            "priority"
        ],
        "filter": {
            "value": 1
        },
        "sortfield": "priority",
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```


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
}
```

使用标签检索特定的触发器

使用标签检索特定的触发器。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17578"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```

参考

- Discovery rule
- Item
- Host

- Host group

源码

CTrigger::get() 方法可在 *frontends/php/include/classes/api/services/CTrigger.php* 中参考。

43. 触发器原型

此类用于管理触发器原型。

对象引用：

- Trigger prototype

可用方法：

- triggerprototype.create - 创建新的触发器原型
- triggerprototype.delete - 删除触发器原型
- triggerprototype.get - 检索触发器原型
- triggerprototype.update - 更新触发器原型

> 对象

以下对象与 `triggerprototype` API 直接相关。

触发器

触发器原型对象包含以下属性。

属性类	说明	
triggerid	string	(只读)触发器原型的ID。
description (必须)	string	触发器原型的名称。
expression (必须)	string	生成的触发器表达式。

属性类	说明	
comments	string	Additional comments to the trigger prototype. 触发器原型的附加注释。
priority	integer	触发器原型的严重级别。 许可值： 0 - (默认) 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。

属性类	说明	
status	integer	触发器原型是否在启用状态或禁用状态。 许可值： 0 - *(默认)* 已启用； 1 - 已禁用。
templateid	string	*(只读)* 触发器原型父模板的ID。

属性类	说明	
type	integer	触发器原型是否可以生成多个异常事件。 许可值： 0 - (默认) 不生成多个事件； 1 - 生成多个事件。
url	string	关联到触发器原型的URL。

属性类	说明	
recovery_mode	integer	正常事件生成模式。 许可值为： 0 - (默认) 表达式； 1 - 恢复表达式； 2 - 无。
recovery_expression	string	生成的触发器恢复表达式。

属性类	说明	
correlation_mode	integer	正常事件关闭。 许可值为： 0 - (默认) 所有异常； 1 - 匹配标签值的所有异常。
correlation_tag	string	匹配的标签。
manual_close	integer	允许手动关闭。 许可值为： 0 - (默认) 不允许； 1 - 允许。

Trigger prototype tag

The trigger prototype tag object has the following properties.

Property	Type	Description
tag (required)	string	Trigger prototype tag name.
value	string	Trigger prototype tag value.

创建

描述

`object triggerprototype.create(object/array triggerPrototypes)`

这个方法可以创建新的触发器原型。

参数

(object/array) 需要创建的触发器原型。除standard trigger prototype properties之外，此方法还接受以下参数。

参数类	说明	
dependencies	array	依赖触发器原型的触发器和触发器原型。 触发器必须已定义 `triggerid` 属性。
tags	array	触发器原型标签。

<note important> 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。:::

返回值

(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。

范例

创建触发器原型

创建一个触发器原型来检测磁盘剩余空间是否小于 20%。

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
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17372"
        ]
```

```
    },
    "id": 1
}
```

源码

CTriggerPrototype::create() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

删除

说明

`object triggerprototype.delete(array triggerPrototypeIds)`

此方法允许删除触发器原型。

参数

(array) 需要删除的触发器原型 ID。

返回值

(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。

范例

删除多个触发器原型

删除两个触发器原型。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.delete",
    "params": [
        "12002",
        "12003"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

源码

CTriggerPrototype::delete() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

更新

说明

`object triggerprototype.update(object/array triggerPrototypes)`

此方法允许更新已有的触发器原型。

参数

(object/array) 需要更新的触发器原型Trigger prototype properties。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。除standard trigger prototype properties之外，该方法接受以下参数。

参数类	说明	
dependencies	array	依赖触发器原型的触发器和触发器原型。触发器必须已定义 triggerid 属性。
tags	array	触发器标签。

<note important> 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。:::

返回值

(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。

范例

启用触发器原型

启用一个触发器原型，即将其状态设置为 0。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

替换触发器原型标签

为触发器原型替换标签。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "17373",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17373"
        ]
    },
    "id": 1
}
```

源码

CTriggerPrototype::update() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

获取

说明

`integer/array triggerprototype.get(object parameters)`

此方法允许根据指定的参数检索触发器原型。

参数

`(object)` 定义需要输出的参数。该方法支持以下参数。

参数类	说明	
active	flag	仅返回所属被监控主机的已启用触发器原型。

参数类	说明	
applicationids	string/array	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	仅返回所属指定低级别发现规则的触发器原型。

参数类	说明	
functions	string/array	仅返回使用指定函数的触发器。 有关支持的功能列表，请参阅supported trig-ger func-tions页面。
group	string	仅返回来自指定名称的主机组中所属主机的触发器原型。

参数类	说明	
groupids	string/array	仅返回来自指定主机组中所属主机的触发器原型。
host	string	仅返回指定名称的所属主机的触发器原型。
hostids	string/array	仅返回指定主机所属的触发器原型。

参数类	说明	
inherited	boolean	仅返回从模板继承的触发器原型，如果设置为 `true`。
maintenance	boolean	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`。

参数类	说明	
min_severity	integer	仅返回严重级别大于或等于指定严重级别的触发器原型。
monitored	flag	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。

参数类	说明	
templated	boolean	仅返回所属模板的触发器原型，如果设置为`true`。
templateids	string/array	仅返回指定模板所属的触发器原型。
triggerids	string/array	仅返回指定ID的触发器原型。

参数类	说明	
expandExpression	flag	展开在触发器原型表达式中的函数和宏。
selectDiscoveryRule	query	返回触发器原型所属的低级别发现规则。

参数类	说明	
selectFunctions	query	返回在 functions 属性中在触发器中使用的函数。 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)* 函数的 ID； itemid - *(string)* 使用在函数中的监控项 ID；

参数类	说明	
selectGroups	query	返回在groups属性中触发器原型所属的主机组。
selectHosts	query	返回在hosts属性中触发器所属的主机。
selectItems	query	返回在items属性中触发器所包含的监控项。

参数类	说明	
selectDependencies	query	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	返回在 `tags` 属性中触发器原型标签。

参数类	说明	
filter	object	仅返回与指定筛选完全匹配的结果。 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。 支持额外的筛选： `host` - 触发器原型所属主机的正式

参数类	说明	
limitSelects	integer	限制子查询返回的记录数量。 适用于以下子查询：`selectHosts` - 以 `host` 分类结果。
sortfield	string/array	由指定属性分类结果。 许可值为：`triggerid`, `description`, `status` 和 `priority`。

参数类	说明	
countOutput	boolean	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	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回两者其中任一：

- 一组对象；
- 如果已经使用了 countOutput 参数，则检索对象的计数。

范例

从低级别发现规则中检索触发器原型

从低级别发现规则中检索所有的触发器原型和相关函数。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": "extend",
        "selectFunctions": "extend",
        "discoveryids": "22450"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```


根据标签检索特定的触发器原型

请求:


```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
```

```
    "params": {
        "output": [
            "triggerid",
            "description"
        ]
        "selectTags": "extend",
        "triggerids": [
            "17373"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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
}
```

参考

- Discovery rule
- Item
- Host
- Host group

源码

CTriggerPrototype::get() 方法可在 *frontends/php/include/classes/api/services/CTriggerPrototype.php* 中参考。

44. 用户

该类用于用户的使用.

对象引用:

- 用户

可用的方法:

- user.create - 创建用户
- user.delete - 删除用户
- user.get - 检索用户
- user.login - 登录
- user.logout - 注销
- user.update - 更新用户
- user.checkauthentication - 检查认证

> 用户对象

以下对象与 `user` API 直接相关.

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。
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。

属性类	说明	
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。 默认: 50.
surname	string	姓。

属性类	说明	
theme	string	用户的主题。 可能的值: `default` - *(default)* system default; `blue-theme` - Blue; `dark-theme` - Dark.
type	integer	用户类型。 可能的值: 1 - *(default)* Zabbix user; 2 - Zabbix admin; 3 - Zabbix super admin.

属性类	说明	
url	string	在登录后将用户重定向到页面的URL。

媒介

媒介对象具有以下属性。

属性类	说明	
mediatypeid (required)	string	用于媒介的媒介类型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。 如果类型是媒介类型电子邮件，值被设置为数组。其他类型媒介类型，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 - (默认) enabled; 1 - disabled.

属性类	说明	
severity	integer	触发发送通知告警级别。 严重性以二进制形式存储，每一位表示相应的严重性。例如，12在二进制中等于1100，这意味着通知将由具有警告和平均级别的触发器

属性类	说明	
period	string	当通知可以作为时间段发送或者用分号隔开用户宏。 默认: 1-7,00:00-24:00

创建

描述

`object user.create(object/array users)`

此方法允许创建新的用户。

参数

(`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。

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。

示例

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介。

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

- 媒介
- 用户组

来源

CUser::create() in *frontends/php/include/classes/api/services/CUser.php*.

删除

说明

`object user.delete(array users)`

此方法允许删除用户。

参数

`(array)` 要删除用户 ID。

返回值

`(object)` 返回一个包含 `userids` 属性下删除用户 ID 的对象。

示例

删除多个用户

删除 2 个用户。

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*.

更新

说明

`object user.update(object/array users)`

这个方法允许更新存在的用户。

参数

`(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。

除了标准用户属性之外, 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。

返回值

`(object)` 在 `userids` 属性下, 返回包含更新用户 id 对象。

示例

重命名用户

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*.

检查认证

描述

`object user.checkAuthentication`

此方法检查并延长用户会话。

参数

该方法支持以下参数。

参数 [	型](/zh/manual/api/reference_commentary#data_types) 描述	
extend	boolean	默认值：“true”。将其值设置为 “false” 允许检查会话而不延长其生存
sessionid	string	用户会话 id。

<note important> 调用检查认证方法默认情况下延长用户会话。. :::

返回值

（object）返回包含用户信息的对象.

示例

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.checkAuthentication",
    "params": {
        "sessionid": "8C8447FF6F61D134CEAC740CCA1BC90D"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "alias": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
```

```
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "type": "3",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "debug_mode": true,
        "userip": "127.0.0.1",
        "sessionid": "8C8447FF6F61D134CEAC740CCA1BC90D",
        "gui_access": "0"
    },
    "id": 1
}
```


Note:
响应类似于用户登陆 userData 参数设置为 true 的调用响应（区别在于，用户数据是通过会话 id 而不是用户名/密码检索的）。

来源

CUser::checkAuthentication() in *ui/include/classes/api/services/CUser.php*.

注销

说明

`string/object user.logout(array)`

这个方法用于用户注销 API 并使当前认证令牌失效。

参数

(array) 这个方法接受一个空数组。

返回值

(boolean) 如果用户已成功注销，则返回 `true`。

示例

登出

通过 API 注销。

Request:


```
{
    "jsonrpc": "2.0",
    "method": "user.logout",
    "params": [],
    "id": 1,
    "auth": "16a46baf181ef9602e1687f3110abf8a"
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```


参考

- 登陆

来源

CUser::login() in *frontends/php/include/classes/api/services/CUser.php*.

登录

说明

`string/object user.login(object parameters)`

此方法允许登录到 API 并生成身份验证令牌。

> **Warning:**
> 当使用这个方法的时候, 你必须使用注销方法，防止产生大量的开放会话记录。.

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	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。:::

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。

除了标准用户属性外, 还返回以下信息:

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。 可能值的列表, 请参阅用户组对象的 `gui_access` 属性。
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。:::

示例

认证一个用户

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
}
```

请求已验证用户的信息

验证并返回有关用户的附加信息。

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

- 注销

来源

CUser::login() in *frontends/php/include/classes/api/services/CUser.php*.

获取

说明

`integer/array user.get(object parameters)`

此方法允许根据给定的参数获取用户。

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	
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，在参考说明中有详细描述.
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。

示例

获取用户

获取所有已配置的用户。

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

- 媒介
- 媒介类型
- 用户组

来源

CUser::get() in *frontends/php/include/classes/api/services/CUser.php*.

45. 用户组

此类设计用于处理用户组。

对象引用:

- 用户组

Available methods:

- usergroup.create - 创建新的用户组
- usergroup.delete - 删除用户组
- usergroup.get - 检索用户组
- usergroup.update - 更新用户组

> 用户组对象

以下对象与 `usergroup` 直接相关。

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。

属性类	说明	
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。

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。

属性类	说明	
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。

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。

创建

说明

`object usergroup.create(object/array userGroups)`

此方法允许创建新的用户组。

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。

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。

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。

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

- 权限

来源

CUserGroup::create() in *frontends/php/include/classes/api/services/CUserGroup.php*.

删除

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。

示例

删除多个用户组

删除 2 个用户。

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*.

更新

说明

`object usergroup.update(object/array userGroups)`

此方法允许更新现有的用户组。

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。

返回值

`(object)` 返回包含 “usrgrpids” 属性下更新的用户组的 ID 的对象。

示例

禁用用户组

禁用一个用户组。

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

- 权限

来源

CUserGroup::update() in *frontends/php/include/classes/api/services/CUserGroup.php*.

获取

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。

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。

属性类	说明	
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	在 `tag_filter` 属性中返回基于用户组标记的权限。 它具有以下属性: `groupid` - (string) 主机组的 ID; `tag` - (string) 标记名称; `value` - (string) 标记值.
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 - (integer) 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	参考说明中详细描述了所有“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

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。

示例

检索已启用的用户组

检索所有已启用的用户组。

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

- 用户

来源

CUserGroup::get() in *frontends/php/include/classes/api/services/CUserGroup.php*.

46. 用户宏

该类是用于处理主机宏和全局宏。

对象引用：

- 全局宏
- 主机宏

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

> 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。

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。
macro (required)	string	宏字符串。
value (required)	string	宏的值。

创建主机宏

说明

`object usermacro.create(object/array hostMacros)`

此方法允许创建新的主机宏。

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.

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。

示例

创建主机宏

在主机”10198” 创建主机宏”{$SNMP_COMMUNITY}” 值为”public”。

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*.

创建全局宏

说明

`object usermacro.createglobal(object/array globalMacros)`

此方法允许创建新的全局宏。

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。

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。

示例

创建一个全局宏

创建一个宏"{$SNMP_COMMUNITY}" 值为"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*.

删除主机宏

说明

`object usermacro.delete(array hostMacroIds)`

此方法允许删除主机宏。

参数

(array) 要删除的主机宏的 ID。

返回值

(object) 返回一个包含 "hostMacs" 属性下删除的主机宏 ID 的对象。

示例

删除多个主机宏

删除 2 个主机宏

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*.

删除全局宏

说明

`object usermacro.deleteglobal(array globalMacroIds)`

此方法允许删除全局宏。

参数

(array) 要删除的全局宏的 ID。

返回值

(object) 返回包含 “globalmacroids” 属性下删除的全局宏 ID 的对象。

示例

删除多个全局宏

删除 2 个主机宏。

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*.

更新主机宏

说明

`object usermacro.update(object/array hostMacros)`

此方法允许更新现有的主机宏。

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。

返回值

(object) 返回包含 `hostMacroids` 属性下更新的主机宏的 ID 的对象。

示例

更改主机宏的值

更改主机宏的值为”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*.

更新全局宏

说明

`object usermacro.updateglobal(object/array globalMacros)`

此方法允许更新现有的全局宏。

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。

返回值

(object) 返回包含 “globalmacroids” 属性下更新的全局宏的 ID 的对象。

示例

更改全局宏的值

将全局宏的值更改为 “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*.

获取

说明

`integer/array usermacro.get(object parameters)`

该方法允许根据给定的参数检索主机宏和全局宏。

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。

属性类	说明	
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，在页参考说明 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	

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。

示例

检索主机的主机宏

检索主机”10198” 定义的所有主机宏。

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。

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*.

47. 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 - 创建新的值映射
- valuemap.delete - 删除值映射
- valuemap.get - 检索值映射
- valuemap.update - 更新值映射

> 值映射对象

以下对象与 `valuemap`API 相关。

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.

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。

创建

说明

`object valuemap.create(object/array valuemaps)`

此方法允许创建新的值映射。

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。

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。

示例

创建一个值映射

使用两个映射创建一个值映射。

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

CValueMap::create() in *frontends/php/include/classes/api/services/CValueMap.php*.

删除

说明

`object valuemap.delete(array valuemapids)`

此方法允许删除值映射。

参数

(array) 要被删除的映射的 ID。

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。

示例

删除多个值映射

删除 2 个值映射。

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*.

更新

说明

`object valuemap.update(object/array valuemaps)`

该方法允许更新现有的值映射。

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。

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。

示例

更改值映射名称

将值映射名称更改为 “设备状态”

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
}
```

更改一个值映射的映射

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

CValueMap::update() in *frontends/php/include/classes/api/services/CValueMap.php*.

获取

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。

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，在参考说明中有详细描述.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

`(integer/array)` 返回:

- 一个数组;
- 如果使用了 `countOutput` 参数，则检索到的对象的计数。

示例

检索值映射

检索所有配置的值映射。

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
}
```

检索一个值映射及其映射。

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

CValueMap::get() in *frontends/php/include/classes/api/services/CValueMap.php*.

48.Web 场景

此类用于 Web 场景的使用。

对象引用：

- WEB 场景
- 场景步骤

可用的方法：

- httptest.create - 创建新的 Web 场景
- httptest.delete - 删除 Web 场景
- httptest.get - 检索 Web 场景
- httptest.update - 更新 Web 场景

> Web 场景对象

以下对象与 `webcheck`API 直接相关。

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

属性类	说明	
agent	string	将由Web场景使用的用户代理字符串。 默认: Zabbix
applicationid	string	Web场景所属应用程序的ID。

属性类	说明	
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
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.

属性类	说明	
headers	string	执行请求时将发送的HTTP标题。
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

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。
no (required)	integer	Web场景中步骤的序列号。

属性类	说明	
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。:::

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。

创建

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.

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。

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。

示例

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。

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

- 场景步骤

来源

CHttpTest::create() in *frontends/php/include/classes/api/services/CHttpTest.php*.

删除

说明

`object httptest.delete(array webScenarioIds)`

此方法允许删除 Web 场景。

参数

(array) 要删除的 Web 场景的 ID。

返回值

(object) 返回包含 `httptestids` 属性下删除的 Web 方案的 ID 的对象。

示例

删除多个 Web 场景

删除 2 个 Web 场景

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

CHttpTest::delete() in *frontends/php/include/classes/api/services/CHttpTest.php*.

更新

说明

`object httptest.update(object/array webScenarios)`

此方法允许更新现有的 Web 场景。

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。

返回值

(object) 返回一个对象，该对象包含 `httptestid` 属性下更新的 web 场景的 ID。

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

- 场景步骤

来源

CHttpTest::update() in *frontends/php/include/classes/api/services/CHttpTest.php*.

获取

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。

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。

参数类	描述	
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。
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	

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。

示例

检索网络场景

检索有关 web 场景 “4” 的所有数据。

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

- 主机
- 场景步骤

来源

CHttpTest::get() in *frontends/php/include/classes/api/services/CHttpTest.php*.

Zabbix API 在 **5.0** 版本中的变更

5.0.26 graph

Changes:
ZBX-7706 `graph.get`: Graph availability doesn't depend on permissions to items specified in graph "ymin_itemid" and "ymax_itemid" fields.
Graph having MIN or MAX Y axis linked to inaccessible items will still be accessible but MIN/MAX Y axis works the same way as if specified calculation method is "Calculated".

graphprototype

Changes:
ZBX-7706 `graphprototype.get`: Graph prototype availability doesn't depend on permissions to items specified in graph prototype "ymin_itemid" and "ymax_itemid" fields.

5.0.5 任务

变更:
ZBXNEXT-6167 添加了一个新方法 `task.get`。
ZBXNEXT-6167 为 `task.get` 方法添加了一个新任务类型。
ZBXNEXT-6167 变更了 `task.create` 请求的格式。查阅task.create 可以获得更多详细信息。

5.0.13 configuration

Bug fixes:
ZBX-8999 `configuration.export`: fixed exporting of images separately from other objects

graph

Bug fixes:
ZBX-19200 `graph.get` removed `discover` field from request
ZBX-19388 `graph.update`: fixed method to properly change values on template graph instead of making a new inherited graph if case user has no permissions to child host or template

graphprototype

Bug fixes:
ZBX-19388 `graphprototype.update`: fixed method to properly change values on template graph prototype instead of making a new inherited graph prototype if case user has no permissions to child host or template

host

ZBX-19200 `host.get` removed `discover` field from request

item

ZBX-19200 `item.get` removed `discover` field from request

mediatype

Changes:
ZBXNEXT-6582 increased `maxattempts` from 10 to 100 and `attempt_interval` from 60s to 1h

task

Bug fixes:
ZBX-18941 `task.create`: improved error message by adding item or discovery rule and host name if any of them are not monitored for task with type value 6

template

ZBX-19200 `template.get` removed `discover` field from request

trigger

Bug fixes:
ZBX-19200 `trigger.get` removed `discover` field from request
ZBX-19424 `trigger.create`: fixed trigger creation on PostgreSQL with host name consisting of only numbers

5.0.5 task

Changes:
ZBXNEXT-6167 added a new method `task.get`.
ZBXNEXT-6167 added a new type of task for `task.get` method.
ZBXNEXT-6167 changed format of `task.create` request. See task.create for more details.

附录 1. 参考说明

注释 数据类型

Zabbix API 支持以下输入数据类型:

数据类型描述	
boolean	布尔值，只接受 `true` 或 `false` 两种参数。
flag	如果传递的值不等于 `null` 和 `false`，则认为该值为 `true`。
integer	整数。
float	浮点数。
string	文本字符串。

数据类型	描述
text	长文本字符串。
timestamp	Unix时间戳。
array	有序的值序列，即普通数组。
object	关联数组。
query	用于定义应返回的数据。 可定义为仅返回指定属性的属性名称数组，或者以下预定值之一的数据: `extend` - 返回所有对象属性; `count` - 返回获取到的记录数量，仅支持某些子查询。

<note 注意 >Zabbix API 始终以字符串或数组格式返回:::

属性标签

一些对象属性用短标签来描述它们的行为。可使用以下标签:

- *readonly* - 属性值是自动设置的，不能被客户端定义或修改;
- *constant* - 属性值可以在创建对象时设置，创建后不能被修改。

预留 **ID** 值**"0"** 预留 ID 值"0" 可以用来过滤元素和删除引用的对象。例如，从主机中删除一个引用的代理，proxy_hostid 应该设置为 0 ("proxy_hostid": "0") 或者，要过滤被 zabbix server 监控的主机，proxyids 选项则应该被设置为 0 ("proxyids": "0")。

常用的**"get"** 方法参数 所有 get 方法都支持如下参数:

参数数	类型描述	
countOutput	boolean	返回结果中的记录数，而不是实际的数据。
editable	boolean	如果设置为 `true`，则只返回用户具有写权限的对象。 默认值: `false`。

参数数	类型描述	
excludeSearch	boolean	返回与在search参数中给定的条件不匹配的结果。

参数数	类型描述	
filter	object	仅返回与给定过滤条件完全匹配的结果。 过滤条件是一个数组，其中键是属性名，值可以是单个值或要匹配的值数组。 不适用于text字段。

参数数	类型描述	
limit	integer	限制返回记录的数据。
output	query	要返回的对象属性。 默认值: extend.
preservekeys	boolean	在结果数组中，使用ID作为键。

参数数	类型描述	
search	object	返回给定通配符(不区分大小写)匹配到的结果。 接受一个数组，键是属性名，其值是要搜索的字符串。如果没有其他选项，将执行 LIKE "%…%" 搜索。 仅适用于 string 和 text 字

参数数	类型描述	
searchByAny	boolean	如果设置为true，则返回在filter或search参数中给出的任何条件匹配的结果，而不是所有条件。 默认值:false。

参数数	类型描述	
searchWildcardsEnabled	boolean	如果设置为true，则可以在search中使用”*”作为通配符。 默认值:false。

参数数	类型描述	
sortfield	string/array	按给定属性对结果进行排序。有关可用于排序的属性列表，请参考特定的API get方法描述。宏在排序前不会被展开。

参数数	类型描述	
sortorder	string/array	排序顺序。如果传递数组，则每个值都将与 `sortfield` 参数中给定的相应属性匹配。 可选的值为: ASC - 升序; DESC - 降序。

参数数	类型描述	
startSearch	boolean	search参数将比较字段的开始，即执行LIKE "…%"搜索。 如果searchWildca设置为true，则忽略。

样例 用户权限检查

用户是否有权限修改以 MySQL 或 Linux 开头的主机？

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
        "search": {
            "host": ["MySQL", "Linux"]
        },
        "editable": true,
        "startSearch": true,
        "searchByAny": true
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "0",
    "id": 1
}
```

Note:
结果 0，表示没有拥有读/写权限的主机

不匹配统计

统计主机名称中不包含 ubuntu 的主机数。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
        "search": {
            "host": "ubuntu"
        },
        "excludeSearch": true
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "44",
    "id": 1
}
```

使用通配符搜索主机

查找主机名包含 server 并且接口端口是 10050 或 10071 的主机。将结果限制在 5 个并按主机名降序排序。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "50003",
            "host": "WebServer-Tomcat02",
```

```
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "50004",
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "99032",
            "host": "MySQL server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    ],
    "id": 1
}
```

使用通配符搜索主机并加上 PRESERVEKEYS 参数

如果将参数 preservekeys 添加到上一个请求中，返回的结果是一个关联数组，键是对象的 id。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
```

```
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5,
        "preservekeys": true
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "50003": {
            "hostid": "50003",
            "host": "WebServer-Tomcat02",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50005": {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50004": {
            "hostid": "50004",
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "99032": {
            "hostid": "99032",
            "host": "MySQL server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        "99061": {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    },
    "id": 1
```

```
}
```

附录 **2.** 从版本 **4.4** 到版本 **5.0** 的一些变更

向下不兼容的一些变更　概要

ZBX-18998 added more strict validation for JSON-RPC structure.

动作

变更:
ZBXNEXT-5548 去除了对 def_longdata, def_shortdata, r_longdata, r_shortdata, ack_longdata, ack_shortdata 属性的支持。

监控项, 模板

变更:
ZBXNEXT-5596 去除了以下监控项属性的支持 port, snmp_community, snmpv3_authpassphrase, snmpv3_authprotocol, snmpv3_contextname, snmpv3_privpassphrase, snmpv3_privprotocol, snmpv3_securitylevel, snmpv3_securityname ，在主机接口中添加了同样的属性。添加了监控项 type 20 - SNMP agent，移除了监控项 type 1 - SNMPv1 agent, 4 - SNMPv2 agent, 6 - SNMPv3 agent。

其他变更以及漏洞修复　动作

变更:
ZBXNEXT-5548 将 opmessage 对象中的 default_msg 默认值从 0 变更为 1。

审计日志

变更:
ZBXNEXT-4584 添加了新的审计日志 API，引入了一个新的方法 auditlog.get。

事件

变更:
ZBXNEXT-1882 event.acknowledge: 在 action 中添加了一个允许取消确认事件的新选项。

主机

漏洞修复:
ZBXNEXT-5694 host.get: 修复了带有计数输出的选项 selectScreens 。
变更:
ZBXNEXT-5694 host.get: 添加了一个新选项 withProblemsSuppressed ，该选项返回有被抑制（未显示）的问题的主机 (true), 被抑制（未显示）的问题 (false) 或所有主机 (null - 默认值)。
ZBXNEXT-5694 host.get: 添加了一个新选项 severities ，该选项返回指定故障严重性的主机。
ZBXNEXT-5694 host.get: 添加了一个新选项 inheritedTags ，该选项返回带有从所有已链接的模板中继承标签的主机。
ZBXNEXT-5694 host.get: 添加了一个新选项 selectInheritedTags ，该选项返回已继承的标签，这些标签来自于模板以及父模板的 inheritedTags 。

主机接口

ZBXNEXT-5596 hostinterface.get: 在响应中添加了 details 属性。
ZBXNEXT-2297 hostinterface.get: 添加了新选项 selectMacros ，该选项返回主机原型的用户宏。
ZBXNEXT-2297 hostinterface.create, hostinterface.update: 添加了新属性 macros。

自动发现规则

变更:
ZBXNEXT-3035 添加了对覆盖的支持。
ZBXNEXT-5811 添加了预处理的支持，type 值为”25”。

ZBXNEXT-5879 discoveryrule.get: 添加了新的筛选选项，groupids 允许检索指定主机组的 LLD 规则。

图形原型

变更:
ZBXNEXT-3035 添加了新属性 discover。

主机原型

变更:
ZBXNEXT-3035 添加了新属性 discover。

监控项

变更:
ZBXNEXT-5811 添加了预处理的支持，type 值为"25"。

监控项原型

变更:
ZBXNEXT-3035 添加了新属性 discover。
ZBXNEXT-5811 添加了预处理的支持，type 值为"25"。

媒介类型

变更:
ZBXNEXT-5548 mediatype.create, mediatype.update: 添加了新属性 message_templates。
ZBXNEXT-5548 mediatype.get: 添加了新选项 selectMessageTemplates ，该选项返回在 message_templates 属性中的告警信息模板。

触发器原型

变更:
ZBXNEXT-3035 添加了新属性 discover。

用户宏

ZBXNEXT-2957 usermacro.create, usermacro.createglobal, usermacro.get, usermacro.update, usermacro.updateglobal: 添加了新属性 type。
ZBXNEXT-5849 usermacro.get: 添加了对于值的筛选。

20. 组件

概览　Zabbix 支持通过添加第三方组件或自研的组件来增强 Zabbix 前端的功能，而无需更改 Zabbix 的源代码。

需要注意的是组件代码将被赋予与 Zabbix 源代码相同的权限运行。这意味着：

- 第三方组件可能对 zabbix 环境产生损害。您必须安装可信任的组件；
- 第三方组件代码中的错误可能会导致前端崩溃。如果发生这种情况，只需从前端删除组件代码即可。当 Zabbix 前端重新加载完成后，您将看到一行表明组件被禁用的注释。在Module administration（*administration→general→modules*）中再次点击 *Scan directory* 可以从数据库中删除不生效的组件。

安装　请务必阅读特定组件的安装手册。建议逐个安装新的组件以便于捕捉安装过程中的失败信息。

在安装组件之前：

- 应确保所有下载的组件是从可信来源获取的，否则安装带有恶意代码的组件可能会导致数据丢失等后果。
- 不同版本的同一组件（相同 ID）允许并行安装，但同一时间只允许启用单一版本的组件。

组件安装步骤：

- 将您的组件解包至自己的文件夹中，放入 Zabbix 前端 modules 文件夹下。
- 应确保您的组件文件夹中至少包含 manifest.json 文件。

- 前往Module administration并单击 *Scan directory* 按钮。
- 新的组件将与其版本、作者、描述和状态一起出现在组件列表中。
- 单击其状态启用组件。

故障排除：

故障解	方法
组件在列表中不可见确保在 Zabbi	前端 `modules/your-module/` 件夹中包含 manifest.json 配置文件，若上述条件已满足，那意味着组件版本与 Zabbix 版本存在冲突。若在 `modules/your-module/` 找不到 manifest.json 配置文件，可能是由于您在错误的目录中进行解包导致。
前端崩溃可能是	于组件代码与当前 Zabbix 版本或服务器配置不兼容导致，请删除这些组件文件并重新加载 Zabbix 前端，您将看到这些组件已被禁用。前往Module administration并再次单击 *Scan directory* 来移除这些无效组件。
出现相同的命名、*ID* 或操作的错误信息新组件会尝试去注册一个新的组件	，组件 ID 或操作，如果被已有组件占用，在启用新组件之前，请禁用这些冲突组件(错误信息中已提到)。
出现其他技术错误提示请将这些组件的报错	息反馈至开发者。

组件开发　组件是基于 PHP 语言编写的，建议使用 Zabbix 模型-视图-控制器（MVC）软件设计模式，好处在于其同时也应用于 Zabbix 前端，有利于后续开发。我们推崇使用 PHP 严格模式进行开发，但这并非强制性的要求。

请注意，您可以通过组件功能轻松地在 Zabbix 前端中添加新的菜单项以及对应各自的视图和操作，但暂时无法通过组件功能注册新的 API 或创建新的数据库表。

组件结构

每个组件都包含一个单独的目录（位于 `modules` 目录中），其子目录包含控制器、视图和任何其他代码：

```
example_module_directory/           （必须）
    manifest.json                   （必须）  元数据和操作定义。
```

```
Module.php                                  组件初始化和事件处理。
actions/                                    操作控制器文件。
    SomethingView.php
    SomethingCreate.php
    SomethingDelete.php
    data_export/
        ExportAsXml.php
        ExportAsExcel.php
views/                                      视图文件。
    example.something.view.php
    example.something.delete.php
    js/                                     使用在视图中的JavaScript文件。
        example.something.view.js.php
partials/                                   视图局部文件。
    example.something.reusable.php
    js/                                     使用在局部中的JavaScript文件。
        example.something.reusable.js.php
```

显而易见，自定义组件目录中唯一的强制要求需包含的配置文件是 `manifest.json`，没有此配置文件，组件将无法注册。`Module.php` 负责注册菜单项并处理诸如 “onBeforeAction” 和 “onTerminate” 之类的事件。*actions*、*views* 和 *partials* 目录中包含组件操作所需的 PHP 和 JavaScript 代码。

命名约定

在创建组件之前，首当其冲的是要统一不同组件项之间（如组件目录和文件）的命名约定，这样我们就可以使这些组件项的名称井然有序，通俗易懂。您也可以在module_structure找到示例。

组件项命名	则示例
组件目录小写	a-z]、下划线加十进制数字 example_v2
操作子目录小写 [	-z]、下划线加字符 data_export
操作文件驼峰式	名法，以操作类型作为结尾 SomethingView.php
视图和局部文件小写 [a-	] module.e 单词用点进行分隔 ample.somethi 以 `module.` 作为前缀，后接组件名称 以操作类型和.php 文件扩展名作结尾。
Javascript 文件除	尾文件扩展名为’.js.php’ 之外，与’ 视图和局部文件’ 项的命名规则保持一致。 php module.example.something.view.js

请注意，’module’ 前缀对于视图和部分文件名是必需的，除非您需要覆盖 Zabbix 核心视图或部分文件，该条规则不适用于操作文件命名。

Manifest 配置文件准备

每个组件都需要一个包含以下 JSON 格式的 Manifest.json 配置文件

参数是	必填类型	默认描述	
manifest_version	是	ouble	- anifest 当前版本，当前可支持的版本号为 **1**。
id	是	tring	- 件 ID，同一 ID 的组件在同一时间内仅允许启用一个。
name	是	tring	- 管理菜单中展示的组件名称。
version	是	tring	- 管理菜单中展示的组件版本。
namespace	是	tring	- odule.php 和操作类的 PHP 命名规则。
author	否	tring	” 理菜单中展示的组件作者。
url	否	tring	” 理菜单中展示的组件 URL。
description	否	tring	” 理菜单中展示的组件描述。
actions	否	bject	} 组件中注册的操作，详情参见 “操作”。
config	否	bject	} 件配置。

详情可参见reference中 manifest.json 配置文件说明的部分。

操作

组件会对 manifest.json 配置文件里 *actions* 对象定义的前端操作进行控制，使用这个方法可以定义新的操作，同样可以对已有操作进行定义。每个操作的键值都应该表示操作名称，相应的值应该包含 `class` 键和可选的 `layout` 键和 `view` 键。

一个操作由四个对应项定义：名称、控制器、视图和布局。数据验证和数据准备工作通常在控制器中完成，格式化输出在视图或局部视图中完成，布局则主要负责用菜单、页眉、页脚等元素装饰前端页面。

组件操作必须在 manifest.json 配置文件的 *actions* 对象中进行定义。

参数是	必填类型	默认	描述	
key	是	tring	-	作名称。小写 [a-z]，单词之间用点进行分隔
class	是	tring	-	作的类名称。包含 `actions` 目录中的子目录路径（如果使用到的话）。
layout	否	tring	layout.htmlpage"	作布局
view	否	tring	操	视图

目前已经有一些预定义好的布局，例如 `layout.json` 或 `layout.xml`。这些预定义的布局可用于产生与 HTML 不同效果的操作。您可以在 app/views/directory 目录下浏览预定义的布局，甚至可以创建自己的布局。

有时为了保持控制器的完整性而单独重定义部分操作的视图是有必要的。这种情况下只需要把必要的视图或局部文件放在组件的 `views` 目录下即可。

请参考Reference部分中的动作控制器文件示例。请尽情探索现有 Zabbix 操作的源代码，位于 app/目录中。

Module.php

这个可选的 PHP 文件负责模块初始化和事件处理，'Module' 类应在此文件中定义，包括扩展基类`\Core\CModule`。'Module' 类必须遵循 manifest.json 配置文件中的命名规范。

```
<?php

namespace Modules\Example;
use Core\CModule as BaseModule;

class Module extends BaseModule {
    ...
}
```

详情请参考 Module.php 在Reference中描述的部分。

参考示例　本章节包含前面几个章节中介绍的不同组件元素的基础版本。

manifest.json

```
{
    "manifest_version": 1.0,
    "id": "example_module",
    "name": "Example module",
    "version": "1.0",
    "namespace": "Example",
    "author": "John Smith",
    "url": "http://module.example.com",
    "description": "Short description of the module.",
    "actions": {
        "example.something.view": {
            "class": "SomethingView",
            "view": "module.example.something.view"
        },
        "example.something.create": {
            "class": "SomethingCreate",
            "layout": null
        },
        "example.something.delete": {
            "class": "SomethingDelete",
            "layout": null
        },
        "example.something.export.xml": {
            "class": "data_export/ExportAsXml",
            "layout": null
        },
        "example.something.export.excel": {
```

```
            "class": "data_export/ExportAsExcel",
            "layout": null
        }
    },
    "config": {
        "username": "john_smith"
    }
}
```

Module.php

```php
<?php declare(strict_types = 1);

namespace Modules\Example;

use APP;
use CController as CAction;

/**
 * 有关其他参考，请参阅 Core\CModule 类。
 */
class Module extends \Core\CModule {

    /**
     * 组件初始化。
     */
    public function init(): void {
        // 初始化主菜单（CMenu 类实例）。
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Reports'))
                ->getSubmenu()
                    ->add((new \CMenuItem(_('Example wide report')))
                        ->setAction('example.report.wide.php')
                    )
                    ->add((new \CMenuItem(_('Example narrow report')))
                        ->setAction('example.report.narrow.php')
                    );
    }

    /**
     * 事件句柄，在执行操作前触发。
     *
     * @param CAction $action  负责当前请求的操作实例。
     */
    public function onBeforeAction(CAction $action): void {
    }

    /**
     * 事件句柄，应用退出时触发。
     *
     * @param CAction $action  负责当前请求的操作实例。
     */
    public function onTerminate(CAction $action): void {
    }
}
```

操作控制器

```php
<?php declare(strict_types = 1);

namespace Modules\Example\Actions;

use CControllerResponseData;
use CControllerResponseFatal;
```

```php
use CController as CAction;

/**
 * 操作组件示例。
 */
class SomethingView extends CAction {

    /**
     * 初始化操作。Zabbix 核心调用方法。
     *
     * @return void
     */
    public function init(): void {
        /**
         * 禁用 SID (Sessoin ID) 验证。会话 ID 验证只能用于涉及数据修改的操作，如更新或删除操作。
         * 在这种情况下，必须在 URL 中显示会话 ID，这样一旦会话过期，URL 就会立即过期。
         */
        $this->disableSIDvalidation();
    }

    /**
     * 检查并清除用户输入参数。Zabbix 核心调用方法。如果返回 false，则执行停止。
     *
     * @return bool true on success, false on error.
     */
    protected function checkInput(): bool {
        $fields = [
            'name'  => 'required|string',
            'email' => 'required|string',
            'phone' => 'string'
        ];

        // 只有经过验证的数据才能进一步使用 $this->hasInput() 和 $this->getInput()。
        $ret = $this->validateInput($fields);

        if (!$ret) {
            $this->setResponse(new CControllerResponseFatal());
        }

        return $ret;
    }

    /**
     * 检查用户是否具有执行此操作的权限。Zabbix 核心调用方法。
     * 如果返回 false，则执行停止。
     *
     * @return bool
     */
    protected function checkPermissions(): bool {
        $permit_user_types = [USER_TYPE_ZABBIX_ADMIN, USER_TYPE_SUPER_ADMIN];

        return in_array($this->getUserType(), $permit_user_types);
    }

    /**
     * 准备视图的响应对象。Zabbix 核心调用方法。
     *
     * @return void
     */
    protected function doAction(): void {
        $contacts = $this->getInput('email');
```

```
        if ($this→hasInput('phone')) {
            $contacts .= ', '.$this→getInput('phone');
        }

        $data = [
            'name' => $this→getInput('name'),
            'contacts' => $contacts
        ];

        $response = new CControllerResponseData($data);

        $this→setResponse($response);
    }
}
```

操作视图

```
<?php declare(strict_types = 1);

/**
 * @var CView $this
 */

$this→includeJsFile('example.something.view.js.php');

(new CWidget())
    →setTitle(_('Something view'))
    →addItem(new CDiv($data['name']))
    →addItem(new CPartial('module.example.something.reusable', [
        'contacts' => $data['contacts']
    ]))
    →show();
```

Zabbix 手册页

这些是 Zabbix 进程的 Zabbix 联机帮助页。

zabbix_agent2

专题: 维护命令 (8)
更新日期: 2019-01-29
索引 返回主要目录

名称

zabbix_agent2 - Zabbix agent 2

概要简介

zabbix_agent2 [**-c** *config-file*]
zabbix_agent2 [**-c** *config-file*] **-p**
zabbix_agent2 [**-c** *config-file*] **-t** *item-key*
zabbix_agent2 [**-c** *config-file*] **-R** *runtime-option*
zabbix_agent2 -h
zabbix_agent2 -V

描述

zabbix_agent2 是用于监视各种服务的参数的应用程序。

选项

-c, **--config** *config-file*
使用自定义配置文件（*config-file*）而不是默认的配置文件

-R, **--runtime-control** *runtime-option*
根据 *runtime-option* 执行管理功能.

运行时控制选项**:** **loglevel increase**
增加日志级别

loglevel decrease
降低日志级别

help
列出可用的运行时控件选项

metrics
列出可用指标

version
显示版本

-p, **--print**
打印已知项目并退出。对于每个项目，要么使用通用默认值，要么提供用于测试的特定默认值。这些默认值在方括号中作为项目关键参数列出。返回值括在方括号中，并以返回值的类型作为前缀，并以竖线字符分隔。对于用户参数，类型始终为 t，因为代理无法确定所有可能的返回值。当查询正在运行的代理守护程序时，由于权限或环境可能不同，显示为工作中的项目不能保证在 Zabbix 服务器或 zabbix_get 中可以工作。返回值类型为：

d
带有小数部分的数字。

m
不支持。这可能是由于查询仅在活动模式下工作的项目（例如日志监视项目或需要多个收集值的项目）引起的。权限问题或不正确的用户参数也可能导致不支持状态。

s
文本。最大长度没有限制。

t
文本。与 **s** 相同。

u
无符号整数

-t, **--test** *item-key*
测试单个项目并退出。有关输出说明，请参见**--print**

-h, **--help**
显示此帮助并退出

-V, **--version**
输出版本信息并退出.

档案

/usr/local/etc/zabbix_agent2.conf
Zabbix agent 2 配置文件的默认位置 (如果在编译时没有修改)。

另请参阅

文档 https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(8), **zabbix_js**(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

作者

Zabbix LLC

索引

名称

概要

描述

选项

档案

另请参阅

作者

该文档是由man2html使用手册页创建的。时间：2019 年 10 月 10 日格林尼治标准时间 14:07:57

zabbix_agentd

章节: 维护命令 (8)
更新时间: 2019-01-29
索引 返回主目录

名称

zabbix_agentd - Zabbix agent 守护进程

概要

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

描述

zabbix_agentd 是用于监视各种服务器参数的守护程序。

选项

-c, **--config** *config-file*
使用自定义 配置文件（*config-file*）而不是默认配置文件。

-f, **--foreground**
在前台运行 Zabbix agent.

-R, **--runtime-control** *runtime-option*
根据 *runtime-option* 执行管理功能.

运行时控制选项

log_level_increase[=*target*]
增加日志级别，如果未指定目标，则影响所有进程

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

日志级别控制目标

pid
Process identifier

process-type
指定类型的所有进程（活动检查，收集器，侦听器）

process-type,N
进程类型和编号（例如，listener，3）

pid
进程标识符，最多 65535。对于较大的值，将 target 指定为 “ process-type，N”

-p, **--print**
打印已知项目并退出。对于每个项目，要么使用通用默认值，要么提供用于测试的特定默认值。这些默认值在方括号中作为项目关键参数列出。返回值括在方括号中，并以返回值的类型作为前缀，并以竖线字符分隔。对于用户参数，类型始终为 t，因为代理无法确定所有可能的返回值。当查询正在运行的代理守护程序时，由于权限或环境可能不同，显示为工作中的项目不能保证在 Zabbix 服务器或 zabbix_get 中可以工作。返回值类型为：

d
带有小数部分的数字

m
不支持。这可能是由于查询仅在活动模式下工作的项目（例如日志监视项目或需要多个收集值的项目）引起的。权限问题或不正确的用户参数也可能导致不支持状态.

s
文本。最大长度没有限制。

t
文本。与 **s** 相同.

u
无符号整数.

-t, **--test** *item-key*
测试单个项目并退出。有关输出说明，请参见**--print**.

-h, **--help**
显示此帮助并退出.

-V, **--version**
输出版本信息并退出.

档案

/usr/local/etc/zabbix_agentd.conf
Zabbix agent 配置文件的默认位置（如果在编译时未修改）.

另请参阅

文档 https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_get**(8), **zabbix_js**(8), **zabbix_proxy**(8) **zabbix_sender**(8) **zabbix_server**(8)

作者

Alexei Vladishev <alex@zabbix.com>

索引

名称

概要

描述

选项

档案

另请参阅

作者

该文档是由 man2html 使用手册页创建的。时间：格林尼治标准时间 2020 年 3 月 18 日 20:50:13

zabbix_get

章节: 用户命令 (1)
更新：2020-02-29
索引 返回主目录

名称

zabbix_get - Zabbix get 实用程序

概要

zabbix_get -s 主机名或 *IP* [**-p** *p* 端口号] [**-I** *IP* 地址] **-k** 监控项关键字
zabbix_get -s 主机名或 *IP* [**-p** 端口号] [**-I** *IP* 地址] **--tls-connect cert --tls-ca-file** *CA*-文件 [**--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 主机名或 *IP* [**-p** 端口号] [**-I** *IP* 地址] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* **-k** *item-key*
zabbix_get -h
zabbix_get -V

描述

zabbix_get zabbix_get 是一个命令行实用程序，用于从 Zabbix agent 获取数据.

选项

-s, **--host** 主机名或 *IP*
指定主机的主机名或 IP 地址.

-p, **--port** 端口号
指定主机上运行的代理的端口号。默认值为 10050.

-I, **--source-address** *IP* 地址
指定源 IP 地址.

-k, **--key** *item-key*
指定要为其检索值的监控项的键.

--tls-connect *value*
如何连接到 agent. 值:

unencrypted
不加密连接（默认）

psk
使用 TLS 和预共享密钥进行连接

cert
使用 TLS 和证书进行连接

--tls-ca-file *CA* 文件
包含用于对等证书验证的顶级 CA 证书的文件的完整路径名.

--tls-crl-file *CRL* 文件
包含已撤销证书的文件的完整路径名.

--tls-agent-cert-issuer 证书颁发者*/I>*
允许的代理证书颁发者.

--tls-agent-cert-subject 证书主题
允许的代理证书主题.

--tls-cert-file 证书文件
包含证书或证书链的文件的完整路径名.

--tls-key-file 密钥文件
包含私钥的文件的完整路径名.

--tls-psk-identity *PSK* 身份
PSK 身份字符串.

--tls-psk-file *PSK* 文件
包含预共享密钥的文件的完整路径名.

--tls-cipher13 密码字符串
OpenSSL 1.1.1 或 TLS 1.3 或更高版本的密码字符串。覆盖默认密码套件选择条件。如果 OpenSSL 版本低于 1.1.1，则此选项不可用.

--tls-cipher 密码字符串
GnuTLS 优先级字符串（用于 TLS 1.2 及更高版本）或 OpenSSL 密码字符串（仅用于 TLS 1.2）。覆盖默认密码套件选择条件

-h, **--help**
显示此帮助并退出.

-V, **--version**
输出版本信息并退出.

例子

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

另请参阅

文档 https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_agentd**(8), **zabbix_js**(8), **zabbix_proxy**(8) **zabbix_sender**(1) **zabbix_server**(8)

作者

Alexei Vladishev <alex@zabbix.com>

使用

名称

概要

描述
选项

例子

另请参阅

作者

该文档是由 man2html 使用手册页创建的。时间：2020 年 3 月 18 日格林尼治标准时间 20:50:27

zabbix_js

部分：用户命令（1）
更新时间: 2019-01-29
索引 返回主目录

名称

zabbix_js - Zabbix JS 实用程序

概要

zabbix_js -s 脚本文件 **-p** 输入参数 [**-l** 日志级别] [**-t** 超时]
zabbix_js -s 脚本文件 **-i** 输入文件 [**-l** 日志级别] [**-t** 超时]
zabbix_js -h
zabbix_js -V

描述

zabbix_js 是可用于嵌入式脚本测试的命令行实用程序.

选项

-s, **--script** 脚本文件
指定要执行的脚本的文件名。如果将 “-” 指定为文件名，则将从标准输入中读取脚本.

-p, **--param** *input-param*
指定输入参数.

-i, **--input** 输入文件
指定输入参数的文件名。如果将 “-” 指定为文件名，则将从标准输入中读取输入.

-l, **--loglevel** *log-level*
日志级别.

-t, **--timeout** 超时
指定超时（以秒为单位）.

-h, **--help**
显示此帮助并退出.

-V, **--version**
输出版本信息并退出.

例子

zabbix_js -s script-file.js -p example

另请参阅

文档 https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_agentd**(8), **zabbix_get**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8)

索引

名称
概要
描述
选项
例子
另请参阅

该文档是由man2html, 使用手册页创建的。时间：2020 年 3 月 18 日格林尼治标准时间 21:23:35

zabbix_proxy

章节: 维护命令 (8)
更新时间: 2020-09-04
索引 返回主目录

名称

zabbix_proxy - Zabbix proxy 守护程序

概要

zabbix_proxy [**-c** 配置文件]
zabbix_proxy [**-c** 配置文件] **-R** 运行时选项
zabbix_proxy -h
zabbix_proxy -V

描述

zabbix_proxy 是一个守护程序，用于从设备收集监视数据并将其发送到 Zabbix server.

选项

-c, **--config** 配置文件
使用自定义配置文件而不是默认配置文件.

-f, **--foreground**
在前台运行 Zabbix proxy.

-R, **--runtime-control** 运行时选项
根据 *runtime-option* 执行管理功能.

运行时控制选项

config_cache_reload
重新加载配置缓存。忽略当前是否正在加载缓存。活动的 Zabbix 代理将连接到 Zabbix 服务器并请求配置数据。默认配置文件（除非指定了-c 选项）将用于查找 PID 文件，并将信号发送到进程，列在 PID 文件中.

snmp_cache_reload
重新加载 SNMP 缓存。

housekeeper_execute
执行管家。如果当前正在执行管家，则将其忽略.

diaginfo[=*section*]
记录指定节的内部诊断信息。Section 可以是 historycache, preprocessing。缺省情况下，记录所有节的诊断信息

log_level_increase[=*target*]
如果未指定目标，则增加日志级别，影响所有进程。

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

日志级别控制目标

process-type
指定类型的所有进程（配置同步器，数据发送器，发现器，心跳发送器，历史记录同步器，管家，http 轮询器，icmp pinger，ipmi 管理器，ipmi 轮询器，java 轮询器，轮询器，自我监控，snmp 捕获器，任务管理器，捕获器，无法访问的轮询器，vmware 收集器）

process-type,N
进程类型和编号 (例如，轮询器，3)

pid
进程标识符，最多 65535。对于较大的值，将 target 指定为 “ process-type，N”

-h, **--help**
显示此帮助并退出.

-V, **--version**
输出版本信息并退出.

档案

/usr/local/etc/zabbix_proxy.conf
Zabbix proxy 配置文件的默认位置（如果在编译时未修改）.

另请参阅

文档 https://www.zabbix.com/manuals **zabbix_agentd**(8), **zabbix_get**(1), **[zabbix_sender](zabbix_ser**
zabbix_server(1), **zabbix_js**(1), **zabbix_agent2**(8)

作者

Alexei Vladishev <alex@zabbix.com>

索引

名称

概要

描述

选项

档案

另请参阅

作者

该文档是由 man2html, 使用手册页创建的。时间：2020 年 9 月 4 日格林尼治标准时间 15:49:29

zabbix_sender

章节: 用户命令 (1)
更新: 2020-02-29
索引 返回主目录

名称

zabbix_sender - Zabbix sender 实用程序

概要

zabbix_sender [**-v**] **-z** 服务器 [**-p** 端口] [**-I** *IP*-地址] **-s** 主机 **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** 服务器 [**-p** 端口] [**-I** *IP* 地址] [**-s** 主机] [**-T**] [**-r**] **-i** 输入文件
zabbix_sender [**-v**] **-c** 配置文件 [**-z** 服务器] [**-p** 端口] [**-I** *IP* 地址] [**-s** 主机] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** 配置文件 [**-z** 服务器] [**-p** 端口] [**-I** *IP* 地址] [**-s** 主机] [**-T**] [**-r**] **-i** 输入文件
zabbix_sender [**-v**] **-z** 服务器 [**-p** 端口] [**-I** *IP* 地址] **-s** 主机 **--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

描述

zabbix_sender 是一个命令行实用程序，用于将监视数据发送到 Zabbix server 或 proxy。在 Zabbixserver 上，应使用相应的密钥创建 **Zabbix trapper** 类型的项目。请注意，此值仅接受来自允许的主机字段中指定的主机的值。

选项

-c, **--config** 配置文件
使用 *config-file*。**Zabbix sender** 从代理配置文件中读取服务器详细信息。默认情况下，**Zabbix sender** 不读取任何配置文件。参数只有 **Hostname**，**ServerActive**，**SourceIP**，**TLSConnect**，**TLSCAFile**，**TLSCRLFile**，**TLSServerCertIssuer**，**TLSServerCert-Subject**，**TLSCertFile**，**TLSKeyFile**，**TLSPSKIdentity** 和 **TLSPSKFile** 支持。在 agent 中定义 **ServerActive** 的所有地址配置参数用于发送数据。如果批处理数据发送到一个地址失败，则以下批处理不会发送到该地址。

-z, **--zabbix-server** *server*
Zabbix server 的主机名或 IP 地址。如果主机由代理监视，则应改用代理主机名或 IP 地址。与**--config** 一起使用时，将覆盖代理配置文件中指定的 **ServerActive** 参数的条目。

-p, **--port** *port*
指定服务器上运行的 Zabbix 服务器陷阱程序的端口号。缺省值为 10051。与**--config** 一起使用时，将覆盖代理配置文件中指定的 **ServerActive** 参数的端口条目.

-I, **--source-address** *IP-address*
指定源 IP 地址。当与一起使用**--config**，覆盖 **SourceIP** 在 agentd 配置文件中指定的参数.

-s, **--host** *host*
指定项目所属的主机名（在 Zabbix 前端中注册）。主机 IP 地址和 DNS 名称将不起作用。与**--config** 一起使用时，将覆盖代理配置文件中指定的 **Hostname** 参数.

-k, **--key** *key*
指定要发送值的项目键.

-o, **--value** *value*
指定项目值.

-i, **--input-file** *input-file*
从输入文件加载值。指定**-** as 从标准输入读取值。文件的每一行包含分隔的空格: 。每个值必须在自己的行中指定。每一行必须包含 3 个由空格分隔的条目: ，其中 **<hostname> <key> <value>** 是被监控主机在 Zabbix 前端注册的名称，“key” 是目标项目的 key，“value” 是要发送的值。指定**-**as 以使用代理配置文件中的 **<hostname>** 或 **--host** 参数

输入文件的一行示例:

”Linux DB3” db.connections 43

必须在 Zabbix 前端的项目配置中正确设置值类型。Zabbix 发件人将在一个连接中最多发送 250 个值。输入文件的内容必须采用 UTF-8 编码。输入文件中的所有值均按自上而下的顺序发送。条目必须使用以下规则设置格式:

•
支持带引号和不带引号的条目.

•
双引号是引号字符.

•
带有空格的条目必须用引号引起来.

•
带引号的条目中的双引号和反斜杠字符必须以反斜杠转义.

•
未引用的条目不支持转义.

•
带引号的字符串支持换行转义序列（\n）.

•
从条目的末尾开始修剪换行符转义序列.

-T, **--with-timestamps**
此选项只能与**--input-file** 选项一起使用。输入文件的每一行必须包含 4 个以空格分隔的条目：**<hostname> <key> <timestamp>**

<value>。时间戳记应以 Unix 时间戳记格式指定。如果目标项目具有引用它的触发器，则所有时间戳记必须按升序排列，否则事件计算将不正确.

输入文件的一行示例:

”Linux DB3” db.connections 1429533600 43

有关更多详细信息，请参见选项 **--input-file**.

如果为 “无数据” 维护类型的主机发送带有时间戳的值，则该值将被删除；否则，该值将被删除。但是，可以在过期的维护期内发送带有时间戳的值，并且该值将被接受.

-N, **--with-ns**
该选项只能与--with-timestamps 选项一起使用。输入文件的每一行必须包含 5 个以空格分隔的条目：**<hostname> <key> <timestamp> <value>** 输入文件的一行示例:

”Linux DB3” db.connections 1429533600 7402561 43

有关更多详细信息，请参见选项 **--input-file**.

-r, **--real-time**
收到值后立即一一发送。从标准输入读取时可以使用此功能.

--tls-connect *value*
如何连接到 server 或 proxy。值:

unencrypted
不加密连接（默认）

psk
c 使用 TLS 和预共享密钥进行连接

cert
使用 TLS 和证书进行连接

--tls-ca-file *CA-file*
包含用于对等证书验证的顶级 CA 证书的文件的完整路径名.

--tls-crl-file *CRL-file*
包含已撤销证书的文件的完整路径名.

--tls-server-cert-issuer *cert-issuer*
允许的服务器证书颁发者.

--tls-server-cert-subject *cert-subject*
允许的服务器证书主题.

--tls-cert-file *cert-file*
包含证书或证书链的文件的完整路径名.

--tls-key-file *key-file*
包含私钥的文件的完整路径名.

--tls-psk-identity *PSK-identity*
PSK 身份字符串.

--tls-psk-file *PSK-file*
包含预共享密钥的文件的完整路径名.

--tls-cipher13 *cipher-string*
OpenSSL 1.1.1 或 TLS 1.3 或更高版本的密码字符串。覆盖默认密码套件选择条件。如果 OpenSSL 版本低于 1.1.1，则此选项不可用.

--tls-cipher *cipher-string*
GnuTLS 优先级字符串（用于 TLS 1.2 及更高版本）或 OpenSSL 密码字符串（仅用于 TLS 1.2）。覆盖默认密码套件选择条件.

-v, **--verbose**
详细模式，**-vv** 了解更多详细信息.

-h, **--help**
显示此帮助并退出.

-V, **--version**
输出版本信息并退出.

退出状态

如果已发送值并且服务器已成功处理所有值，则退出状态为 0。如果发送了数据，但是至少一个值的处理失败，则退出状态为 2。如果数据发送失败，则退出状态为 1.

例子

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -k mysql.queries -o 342.45

发送 342.45 作为受监视主机的 mysql.queries 项目的值。使用代理配置文件中定义的受监视主机和 Zabbix 服务器

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -s ”Monitored Host” -k mysql.queries -o 342.45

使用代理配置文件中定义的 Zabbix 服务器发送 342.45 作为 “受监视的主机” 主机的 mysql.queries 项目的值.

zabbix_sender -z 192.168.1.113 -i data_values.txt

将文件 data_values.txt 中的值发送到 IP 为 192.168.1.113 的 Zabbix server。主机名和密钥在文件中定义.

echo ”- hw.serial.number 1287872261 SQ4321ASDF” | zabbix_sender -c /usr/local/etc/zabbix_agentd.conf -T -i -

将带有时间戳的值从命令行发送到代理配置文件中指定的 Zabbix 服务器。输入数据中的短划线表示还应从同一配置文件中使用主机名.

echo ’”Zabbix server” trapper.item ””’ | zabbix_sender -z 192.168.1.113 -p 10000 -i -

从命令行在端口 10000 上将空值的项目发送到 IP 地址为 192.168.1.113 的 Zabbix 服务器。空值必须用空双引号表示.

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

使用带有证书的 TLS 将 342.45 作为 “受监视的主机” 主机中 mysql.queries 项目的值发送到 IP 为 192.168.1.113 的服务器

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

使用带有预共享密钥（PSK）的 TLS 将 342.45 作为 “受监视的主机” 主机中 mysql.queries 项目的值发送到 IP 为 192.168.1.113 的服务器.

另请参阅

文档 https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_agentd**(8), **zabbix_get**(1), **zabbix_js**(1), **zabbix_proxy**(8), **zabbix_server**(8)

作者

Alexei Vladishev <alex@zabbix.com>

索引

名称

概要

描述

选项

退出状态

例子

另请参阅

作者

该文档是由 man2html, 使用手册页创建的。时间：2020 年 3 月 18 日格林尼治标准时间 20:49:51

zabbix_server

章节: 维护命令 (8)
更新时间: 2020-09-04
索引 返回主目录

NAME

zabbix_server - Zabbix server 守护程序

概要

zabbix_server [**-c** 配置文件]
zabbix_server [**-c** 配置文件] **-R** 运行时选项
zabbix_server -h
zabbix_server -V

描述

zabbix_server 是 Zabbix 软件的核心守护程序.

选项

-c, **--config** 配置文件
使用自定义 配置文件而不是默认配置文件.

-f, **--foreground**
在前台运行 Zabbix serve.

-R, **--runtime-control** *runtime-option*
根据 *runtime-option* 执行管理功能.

RUNTIME CONTROL

Runtime control options:

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Default configuration file (unless **-c** option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

diaginfo[=*target*]
Gather diagnostic information in the server log file. Available since Zabbix 5.0.4.

snmp_cache_reload
Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.

housekeeper_execute
Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified

日志级别控制目标

process-type
指定类型的所有进程（警报器，警报管理器，配置同步器，发现程序，自动扶梯，历史记录同步器，管家，http 轮询器，icmp pinger，ipmi 管理器，ipmi 轮询器，java 轮询器，lld 管理器，lld 工作者，轮询器，预处理器管理器，预处理工作者，代理轮询器，自我监控，snmp 陷阱器，任务管理器，计时器，陷阱器，无法访问的轮询器，vmware 收集器）

process-type,N
进程类型和编号 (例如，轮询器，3)

pid
进程标识符，最多 65535。对于较大的值，将 target 指定为 “process-type,N”

-h, **--help**
显示此帮助并退出.

-V, **--version**
输出版本信息并退出.

档案

/usr/local/etc/zabbix_server.conf
Zabbix server 配置文件的默认位置（如果在编译时未修改）.

另请参阅

文档https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_js**(1), **zabbix_agent2**(8)

作者

Alexei Vladishev <alex@zabbix.com>

索引

名称
概要
描述
选项
档案
另请参阅
作者

该文档是由 man2html, 使用手册页创建的。时间：2020 年 9 月 4 日格林尼治标准时间 15:49:23