Documentation 8.0

ZABBIX

29.10.2026

Contents

用户手册 **5**

欢迎使用 **Zabbix** 文档 **5**

- 开始使用 Zabbix . . . 5
- Zabbix Cloud . . . 5
- 移动应用 . . . 5
- 开发者中心 . . . 5
- 社区与其他资源 . . . 5
- 安装和初始步骤 . . . 5
  - Web 前端安装 . . . 6
  - 快速入门指南 . . . 11
  - 安装：其他指南 . . . 82
  - 升级 . . . 159
  - 已知问题 . . . 174
  - 要求 . . . 182
- 新增功能和升级说明：8.0 . . . 191
  - 模板 . . . 195
  - 媒体类型 . . . 197
  - 监控项 . . . 198
  - 插件 . . . 198
  - 发现和自动注册 . . . 198
  - 低级发现 . . . 198
  - 数据库 . . . 199
  - 进程 . . . 199
  - 安全性 . . . 200
  - 身份验证 . . . 200
  - 小部件 . . . 200
  - 前端 . . . 201
  - 文档 . . . 203
- 升级说明 . . . 203
  - 重大变更 . . . 203
  - 其他 . . . 205
  - 新增功能和升级说明：8.0.x . . . 207
- Zabbix appliance . . . 207
- 进程 . . . 211
  - 服务器 . . . 213
  - proxy . . . 237
  - agent . . . 279
  - Agent 2 . . . 306
  - Java 网关 . . . 361
  - 发送器 . . . 365
  - 获取 . . . 367
  - JS . . . 368
  - Web 服务 . . . 369
  - Bridge adapter . . . 371
  - MCP 服务器 . . . 375
  - 环境变量 . . . 381
- 配置 . . . 383
  - 主机和主机组 . . . 392
  - 监控项 . . . 410
  - 使用触发器进行问题检测 . . . 614

可视化 . . . 662
模板和模板组 . . . 686
开箱即用的模板 . . . 686
事件通知 . . . 699
宏 . . . 746
用户和用户组 . . . 760
定时报表 . . . 769
数据导出 . . . 773
配置导出/导入 . . . 781
正则表达式 . . . 843
事件和解决 . . . 849
问题确认与抑制 . . . 851
手动关闭问题 . . . 854
维护周期 . . . 855
标记 . . . 860
事件关联 . . . 862
发现 . . . 868
网络发现 . . . 868
活动 agent 自动注册 . . . 876
低级发现 . . . 879
虚拟机监控 . . . 901
VMware 监控项键 . . . 902
虚拟机发现键字段 . . . 920
VMware 监控项的 JSON 示例 . . . 924
VMware 监控设置示例 . . . 928
Web 监控 . . . 933
Web 监控项 . . . 941
真实场景 . . . 943
服务监控 . . . 950
服务树 . . . 951
SLA . . . 955
设置示例 . . . 956
服务监控升级 . . . 960
加密和安全 . . . 961
使用证书进行加密 . . . 968
使用预共享密钥进行加密 . . . 975
到数据库的安全连接 . . . 978
前端与服务器之间的安全连接 . . . 989
访问控制 . . . 993
密码学 . . . 997
保护 web 服务器 . . . 997
密钥的存储 . . . 1001
Google Chrome TLS 证书信任 . . . 1008
安全最佳实践 . . . 1010
Web 界面 . . . 1010
菜单 . . . 1011
前端部分 . . . 1018
用户设置 . . . 1267
全局搜索 . . . 1271
前端维护模式 . . . 1273
页面参数 . . . 1274
定义 . . . 1277
其他前端语言 . . . 1277
创建你自己的主题 . . . 1278
调试模式 . . . 1279
Zabbix 使用的 Cookie . . . 1280
时区 . . . 1281
重置密码 . . . 1281
时间段和主机选择器 . . . 1282
配置参数：Zabbix 前端 . . . 1283
API . . . 1290
方法参考 . . . 1294
附录 1. 参考说明 . . . 2115

附录 2. 从 7.4 到 8.0 的变更 . . . 2120
附录 3. 8.0 中的变更 . . . 2123
扩展 . . . 2123
可加载模块 . . . 2125
前端模块 . . . 2133
附录 . . . 2134
安装和设置 . . . 2134
监控项 . . . 2178
支持的函数 . . . 2216
宏 . . . 2216
版本兼容性 . . . 2247
时间段语法 . . . 2248
单位符号 . . . 2249
命令执行 . . . 2250
转义示例 . . . 2251
什么是 Zabbix . . . 2252

Zabbix Cloud **2256**

Zabbix Cloud **2256**
设置和管理 Zabbix Cloud . . . 2256
了解 Zabbix Cloud . . . 2257
在云中部署 Zabbix . . . 2257
节点配置 . . . 2266
概述 . . . 2266
概览选项卡 . . . 2267
访问过滤器选项卡 . . . 2270
加密选项卡 . . . 2271
备份选项卡 . . . 2271
历史记录选项卡 . . . 2271
维护选项卡 . . . 2272
升级选项卡 . . . 2273
添加用户 . . . 2275
简介 . . . 2275
向组织添加用户 . . . 2275
向节点添加用户 . . . 2276
Zabbix Cloud 的主要区别 . . . 2277
简介 . . . 2277
审计日志 . . . 2279
设置 Zabbix Cloud proxy . . . 2281

移动应用 **2288**

移动应用 **2288**
为应用设置 Zabbix . . . 2288
了解该应用 . . . 2288
为应用设置 Zabbix . . . 2288
应用概览 . . . 2290
安全的移动访问 . . . 2297

开发者中心 **2298**

开发者中心 **2298**
基于 Zabbix 构建 . . . 2298
创建插件 . . . 2298
插件接口 . . . 2303
Zabbix 的 Python 库 . . . 2305
安装 . . . 2305
快速入门指南 . . . 2305
使用 Zabbix API . . . 2309
从 Zabbix agent 收集数据 . . . 2311
向 Zabbix 服务器或 proxy 发送数据 . . . 2313
调试日志 . . . 2316
模块 . . . 2317
模块文件结构 . . . 2318

小部件 . . . 2325
教程 . . . 2332
示例 . . . 2358
扩展开发的变更 . . . 2358

Zabbix 手册页 **2359**
zabbix_agent2 . . . 2359
zabbix_agentd . . . 2360
zabbix_get . . . 2362
zabbix_js . . . 2364
zabbix_proxy . . . 2365
zabbix_sender . . . 2366
zabbix_server . . . 2370
zabbix_web_service . . . 2372

版权声明 **2373**

用户手册

欢迎使用 **Zabbix** 文档

这是您使用 Zabbix 监控的首选资源，涵盖从基础设置到高级配置的各个方面。本手册包含安装、配置和运行 Zabbix 所需的全部内容。

Zabbix 8.0 的新增内容

开始使用 **Zabbix**

安装　在您首选的平台上安装 Zabbix 的分步说明，涵盖各种操作系统和数据库配置。

要求　支持的平台和软件先决条件列表，可帮助您为成功部署 Zabbix 做好环境准备。

快速入门指南　简明、面向任务的操作指南，带您了解基础内容——从首次配置步骤到接收第一条问题告警。

Zabbix Cloud

开始使用 **Zabbix Cloud**　按照引导说明快速启动并运行您的 Zabbix Cloud 实例，包括初始配置以及连接受监控设备。

Zabbix Cloud 与本地部署　比较这两种部署方式在管理、可扩展性和维护方面的差异，并了解哪一种能够最无缝地支持您的工作流程。

探索 **Zabbix Cloud**　了解 Zabbix Cloud 的概览，包括其主要功能、优势以及订阅所包含的内容。有关设置和使用的常见问题，请参阅常见问题解答（FAQ）。

移动应用

获取应用　在您的设备上安装 Zabbix 移动应用（即将推出），随时随地开始监控您的 Zabbix 环境。

为应用设置 **Zabbix**　安装 Zabbix bridge adapter，使移动设备能够连接到 Zabbix。然后添加您的设备并开启推送通知。

应用概览　了解使用 Zabbix 移动应用的基础知识。

开发者中心

前端模块　用于构建自定义模块的指南和参考，这些模块可扩展或修改 Zabbix 前端，以适应特定用例。

小部件　对小部件结构和逻辑的细分说明，并提供创建符合您需求的自定义仪表板元素的说明。

插件　关于如何开发和管理 Zabbix 插件的概述，用于扩展功能或与外部系统集成。

社区与其他资源

Zabbix 论坛　一个交流想法、寻找解决方案并分享经验的空间，适合各个 Zabbix 专业水平的用户。

Zabbix 博客　由 Zabbix 团队和社区贡献者精选的新闻、教程和案例研究。

操作视频　一个视频库，包含演示、讨论和技巧，帮助您充分利用 Zabbix。

安装和初始步骤

以下步骤概述了 Zabbix 的设置过程：

1. 验证所需的软件已安装并正在运行。
2. 安装 Zabbix。
3. 安装 Zabbix 前端。
4. 开始监控。

以下提供每个步骤的详细信息。

所需软件　Zabbix 可与以下软件良好配合使用（通常称为 LAMP 堆栈）：

- **Linux** 操作系统：Ubuntu、Debian、CentOS 或其他系统
- **Web** 服务器：Apache 或 NGINX
- 数据库服务器：MySQL/MariaDB 或 PostgreSQL
- **PHP**

在大多数现代 Linux 系统上，PHP 可能已经安装。在这种情况下，您可能只需设置数据库服务器和 Web 服务器。

有关更详细的信息，例如受支持的软件版本，请参阅所需软件。

安装 **Zabbix**　安装 Zabbix 的最简单方式是使用发行版软件包（也可以使用容器安装）。软件包会安装次要的软件依赖项。

针对您的操作系统提供了详细的安装说明，请参阅 **Zabbix** 网站。

选择您的平台并按照提供的说明操作。建议首先安装 Zabbix 服务器、Zabbix agent 2 和前端（Web 界面）。之后，您可以添加 Zabbix proxy 和更多 agent。

安装过程的一部分是创建数据库（用于保存配置和历史数据）并将初始数据导入其中。

安装 **Zabbix Web** 界面　完成 Web 界面的安装向导中的步骤。

系统会要求您输入数据库密码（在上一步中创建），以便前端与数据库通信。

开始监控　登录 Zabbix Web 界面。

使用快速入门指南开始监控常用资源（Linux、Apache、MySQL 或其他资源）。这些快速指南会说明要使用哪些模板来开始监控常用资源。

您可以先参阅用于监控 Linux 服务器的快速入门指南。

Web 前端安装

本节介绍如何安装 Zabbix 前端。请在安装 Zabbix 后端后执行以下步骤。

Zabbix 前端使用 PHP 编写，需要使用已配置 PHP 的受支持 Web 服务器，例如 Apache 或 NGINX。有关所需软件和库的信息，请参阅前端的要求。

如需在 Zabbix Web 前端中使用英语以外的语言，必须在 Web 服务器上安装相应的区域设置，以及 PHP gettext 扩展。请注意，更改语言可能会改变时间/日期格式。

如需为 Zabbix 前端设置 SSL，请参阅为 Zabbix 前端设置 SSL。

安装向导

要安装前端，请在浏览器中打开 Zabbix 前端 URL：

- 对于 Apache：`http://<server_ip_or_name>/zabbix`
- 对于 NGINX：`http://<server_ip_or_name>`

欢迎界面

前端安装向导会打开欢迎界面。

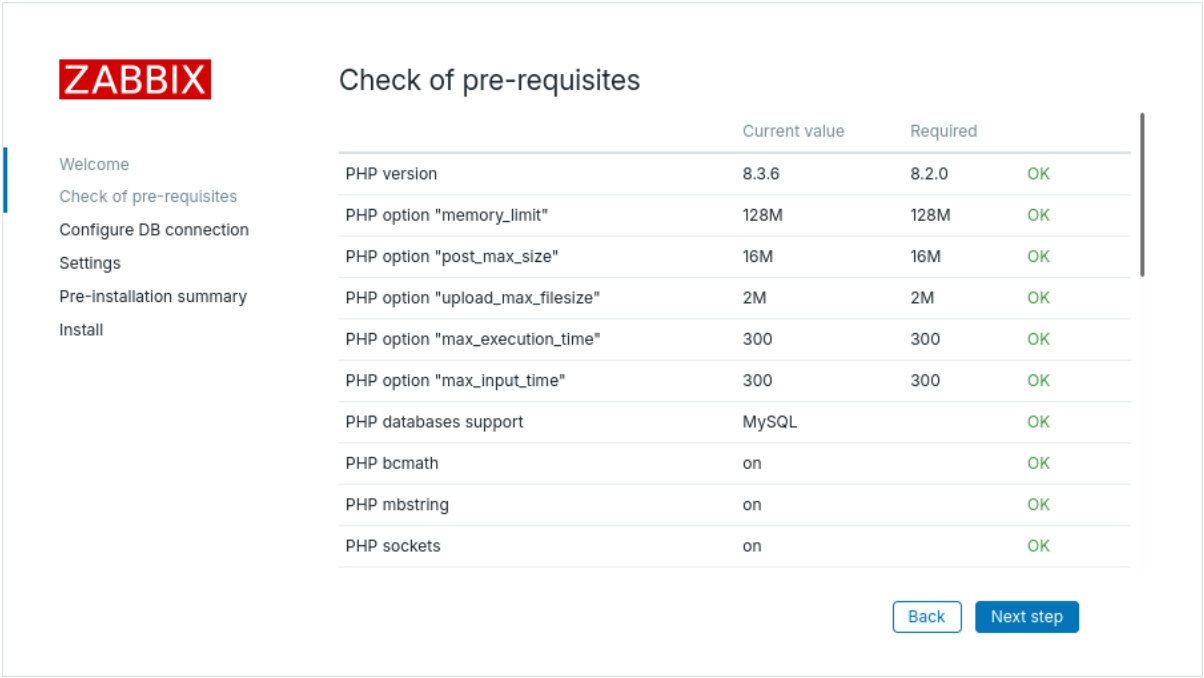


系统范围的默认语言为英语（en_US），并使用美国的时间/日期格式。要更改系统范围的默认语言，请从默认语言下拉菜单中选择所需的语言。

单击下一步按钮继续。

先决条件检查

确保满足所有必需的 Zabbix 前端先决条件。



有关所需和可选 PHP 扩展、所需 PHP 版本以及支持的数据库列表，请参阅要求。

请注意，未满足的可选先决条件会显示为警告状态，即使这些条件未满足，安装过程仍可继续。

Web 服务器所使用的 `php.ini` 文件中，PHP 配置指令的必需值：

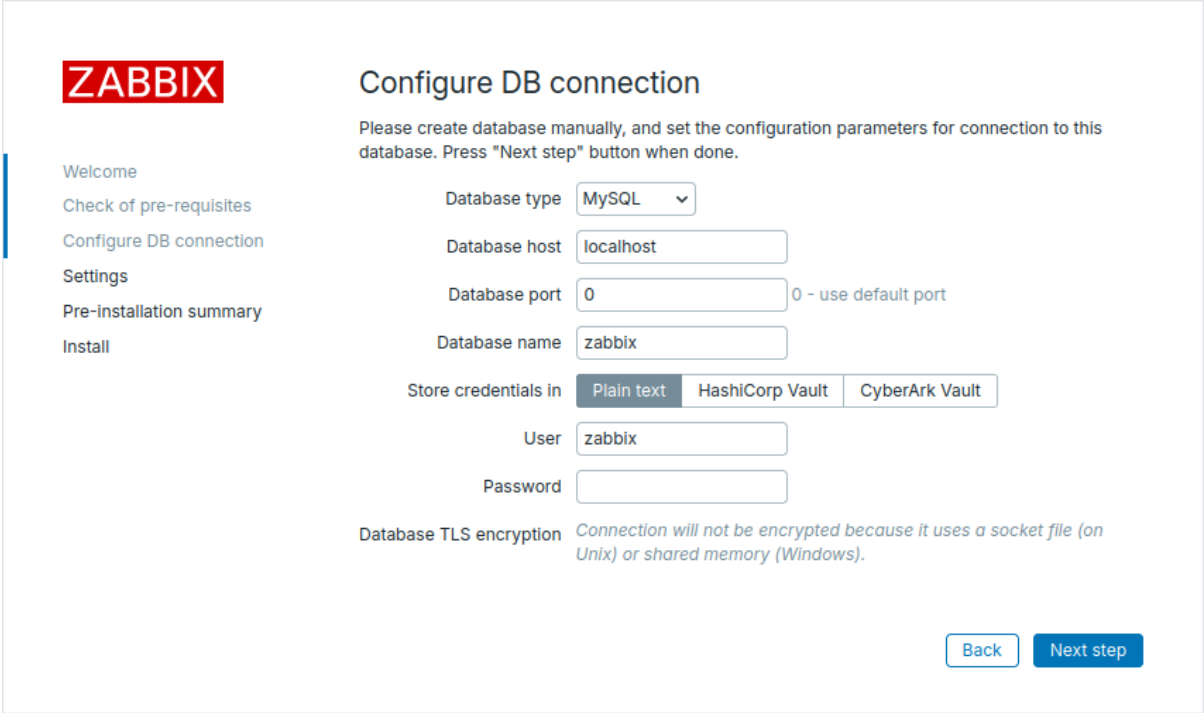
```
memory_limit = 128M ; 最小值
post_max_size = 16M ; 最小值
upload_max_filesize = 2M ; 最小值
max_execution_time = 300 ; 最小值，也允许使用 0 和 -1
max_input_time = 300 ; 最小值，也允许使用 0 和 -1
session.auto_start = 0 ; 必须禁用
arg_separator.output = "&" ; 也允许使用 "&amp;"
```

Attention:
如果需要更改 Web 服务器用户或组，请确认新用户或组对 PHP 会话目录具有所需的权限；否则，Zabbix 安装程序可能无法继续。

如果所有必需的先决条件均已满足，请继续下一步。

配置数据库连接

输入连接数据库的详细信息。



如果 数据库主机字段留空，则使用默认的 Unix 域套接字。

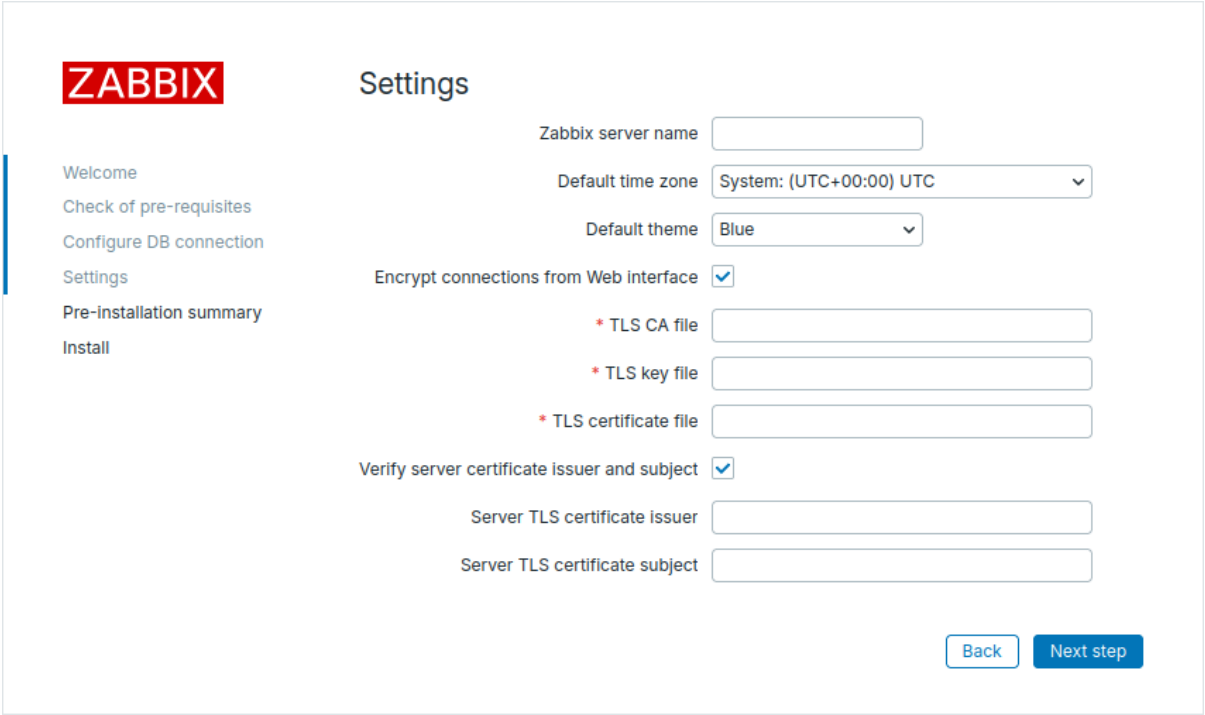
对于 MySQL，在 数据库主机字段中输入 `localhost` 也会通过默认的 Unix 域套接字建立连接。向导不提供配置数据库套接字的选项。要使用自定义套接字，请在 Zabbix 服务器设置中进行配置（例如，在 Zabbix 服务器配置文件（`zabbix_server.conf`）中使用`DBSocket=` 参数）。这样可以确保前端与服务器到数据库的连接设置保持一致。

对于 PostgreSQL，数据库主机字段可以包含多个以逗号分隔的 `host:port` 条目。该字符串会被拆分为多个部分，并按照与 数据库端口 字段相同的方式验证每个端口。验证成功后，系统会将该字符串解析为主机和端口，以构造连接字符串。如果逗号分隔的列表中某个主机未定义端口，则使用 数据库端口的值作为默认端口。如果指定了套接字路径（例如 `/var/run/pgbouncer`），则改用该 Unix 域套接字。

要配置与数据库的安全连接，请选中 数据库 *TLS* 加密复选框，然后转到前端到数据库配置相关字段。在以下情况下，数据库 *TLS* 加密复选框处于禁用状态：- 对于 MySQL，数据库主机设置为 *localhost*。- 对于 PostgreSQL，数据库主机字段的值以斜杠开头，或该字段为空。

要配置 Zabbix 从 HashiCorp Vault 获取机密信息，请在 凭据存储于中选择 *HashiCorp Vault*，然后转到HashiCorp 配置填写展开的字段。要配置 Zabbix 从 CyberArk Vault 获取机密信息，请在 凭据存储于中选择 *CyberArk Vault*，然后转到CyberArk 配置填写展开的字段。

设置



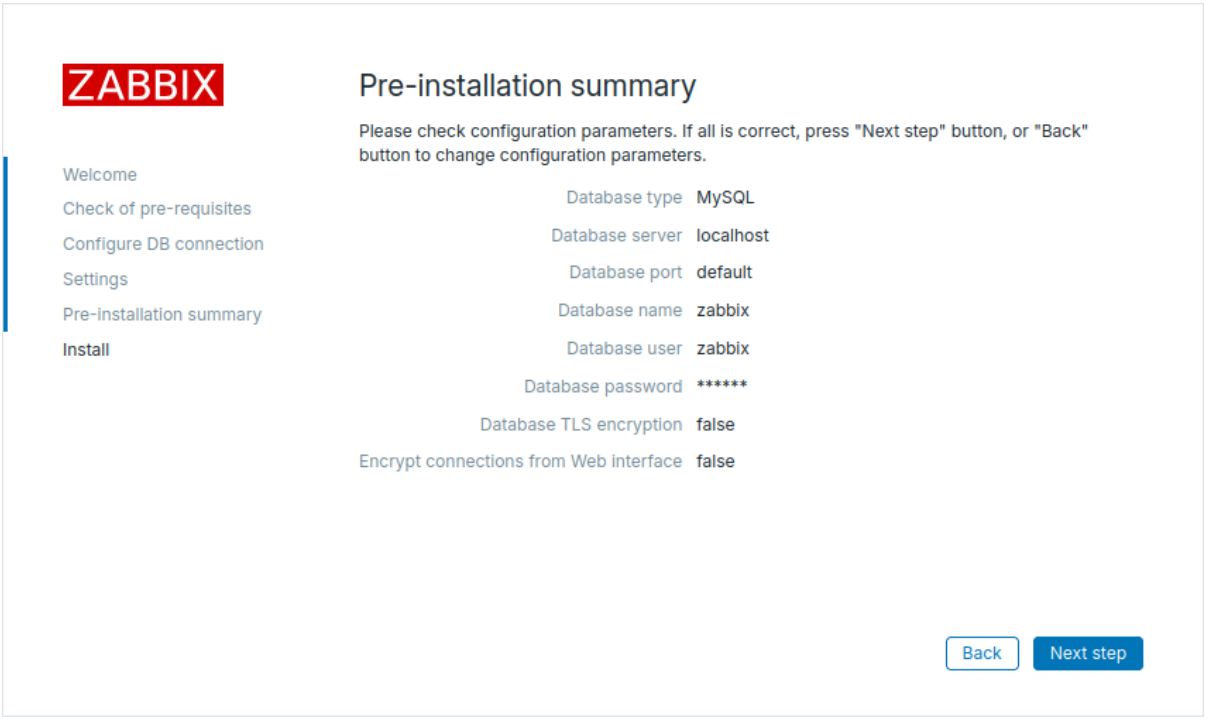
输入 Zabbix 服务器名称是可选的；但是，如果提交了名称，该名称将显示在菜单栏和页面标题中。

为前端设置默认时区和主题。

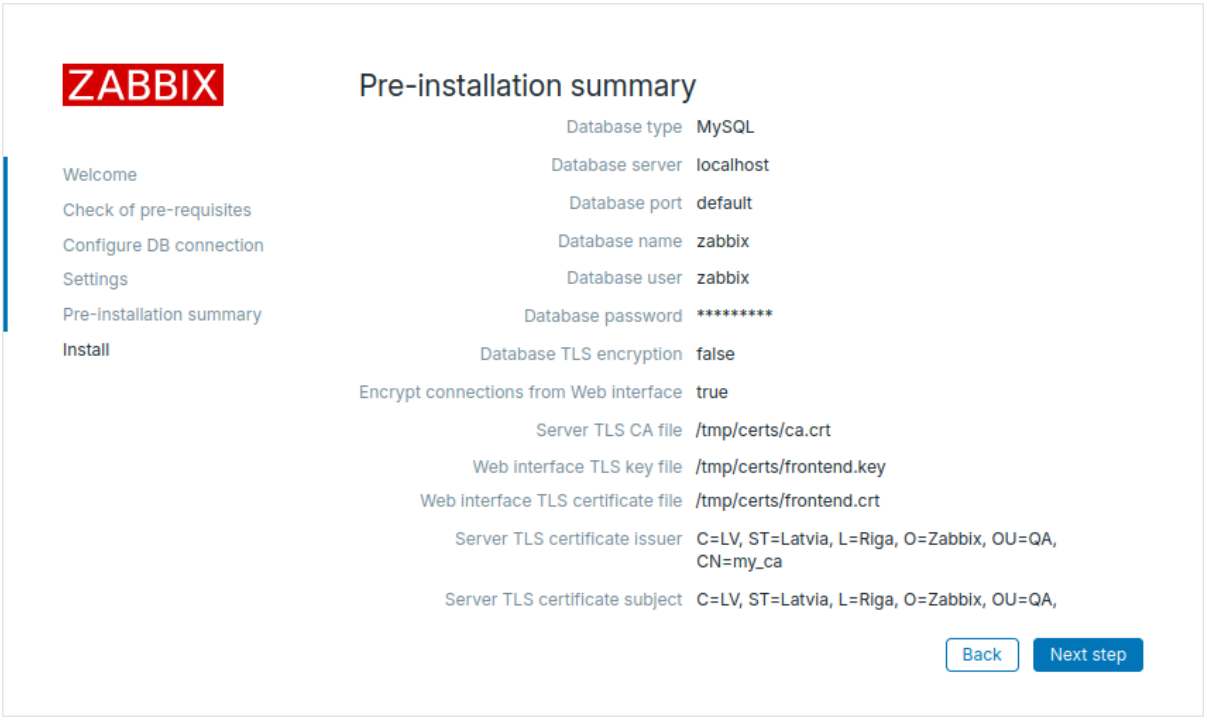
要配置 Zabbix 前端与 Zabbix 服务器之间的 TLS 连接，请选中加密来自 *Web* 界面的连接复选框，然后继续配置 TLS 连接，填写展开的字段。

安装前摘要

查看设置摘要。



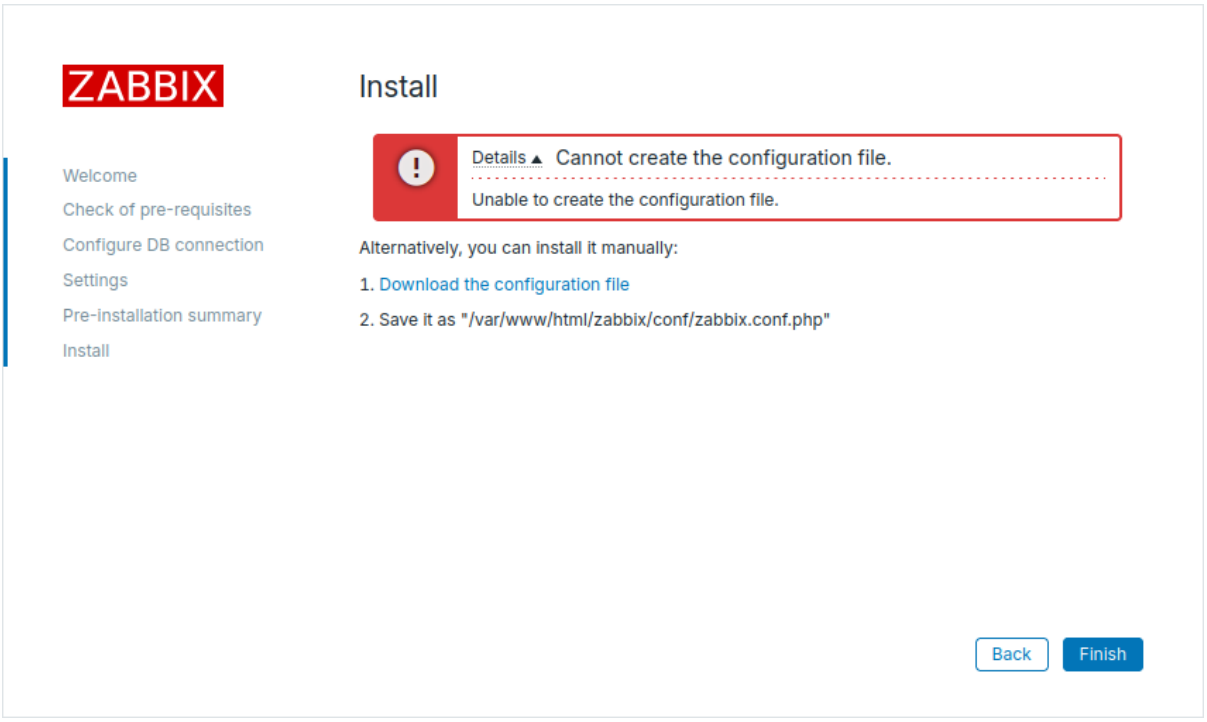
如果已配置 TLS，向导将显示其详细信息。



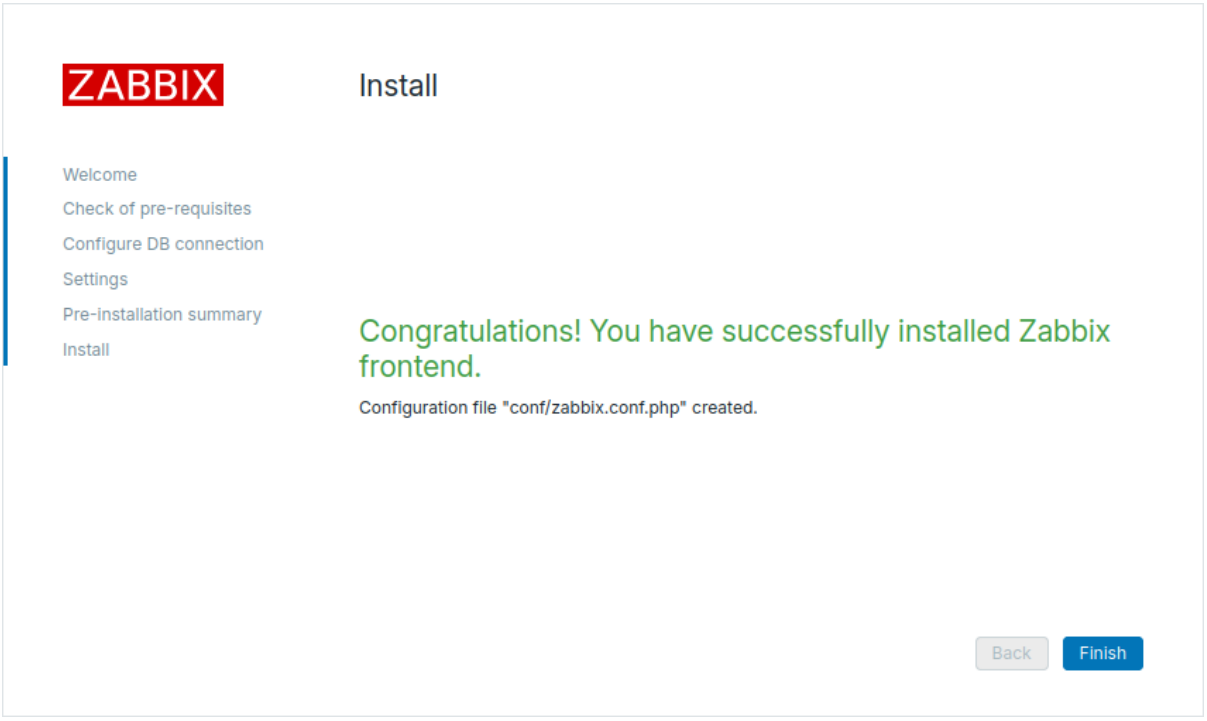
安装

如果 Web 服务器用户对 `conf/` 目录具有写入权限，配置文件将自动保存，您可以立即继续下一步。

如果 Web 服务器用户对 `conf/` 目录没有写入权限，您必须下载配置文件，并将其放置在 Web 服务器 HTML 文档目录下的 `conf/` 目录中，即您复制 Zabbix PHP 文件的位置。

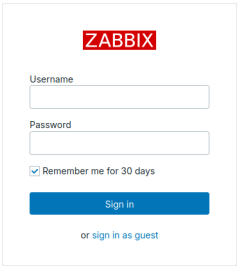


完成安装。



登录

使用默认凭据登录：- 用户名：*Admin* - 密码：*zabbix*



前端配置文件

安装后，您可以通过更新 `zabbix.conf.php` 前端配置文件中的参数来重新配置 Zabbix 前端。

配置文件的默认路径取决于操作系统和 Zabbix 的安装方式：

- 从软件包安装：`/usr/share/zabbix/ui/conf/zabbix.conf.php`
- 从源代码安装：`/var/www/html/zabbix/conf/zabbix.conf.php`
- 使用 appliance：`/etc/zabbix/web/zabbix.conf.php`

有关配置 Zabbix 前端的详细信息，请参阅配置文件选项。

快速入门指南

在安装 Zabbix及其前端后，就可以开始监控了。

使用以下快速入门指南开始监控常用资源：

- Linux 服务器监控
- Apache Web 服务器监控
- MySQL 服务器监控

快速入门指南重点介绍如何使用 Zabbix 官方模板进行监控。
模板是一组预配置的监控元素。
模板可以帮助您扩展监控范围——您可以先在单个监控目标（设备、应用等）上测试模板，对其进行调整，然后将其应用于多个类似的监控目标。

虽然模板适用于开箱即用的监控，但您可能还希望自定义其中的元素。

要更好地了解构成模板的基本元素，请参阅基本配置。

基本配置

这些指南介绍 Zabbix 配置的基本元素：

- 登录并配置用户
- 新主机
- 新监控项
- 新触发器
- 接收问题通知
- 新模板

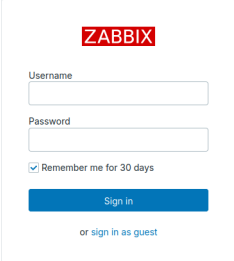
登录和配置用户

概述

在本节中，您将学习如何登录并在 Zabbix 中设置系统用户。

登录

在浏览器中打开 Zabbix 前端 URL：- 对于 Apache：http://<server_ip_or_name>/zabbix - 对于 NGINX：http://<server_ip_or_name>



这是 Zabbix 欢迎界面。输入用户名 **Admin** 和密码 **zabbix**，以Zabbix 超级用户身份登录。您将获得对所有菜单部分的访问权限。

出于安全原因，强烈建议在首次登录后立即更改 Admin 账户的默认密码。

持久登录

若要保持登录状态最长 30 天，请在点击 *Sign in* 之前选择 *Remember for 30 days*。

启用 *Remember for 30 days*：

- 您的会话将保持活动状态 30 天。
- *Auto-logout* 将被覆盖，您会一直保持登录，直到该期限结束。
- 在未来 30 天内再次访问时，系统将自动登录，无需重新输入凭据。

禁用 *Remember for 30 days*：

- 任何先前启用的自动登录都会被清除。
- 会话将根据已配置的 *Auto-logout* 间隔过期。

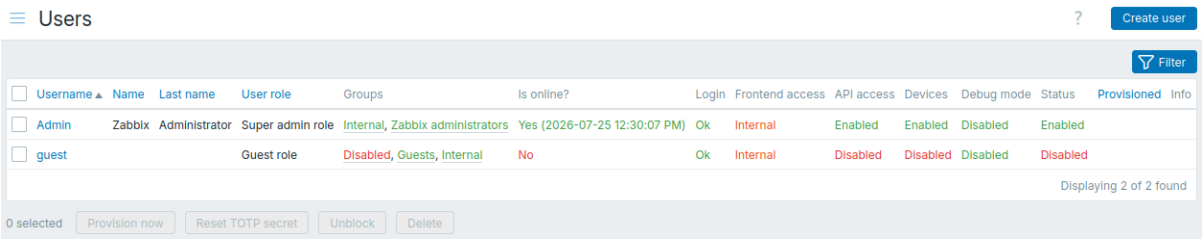
防止暴力破解攻击

如果连续五次登录失败，Zabbix 前端将暂停 30 秒，以防止暴力破解和字典攻击。

成功登录后，将显示最近一次登录失败的 IP 地址。

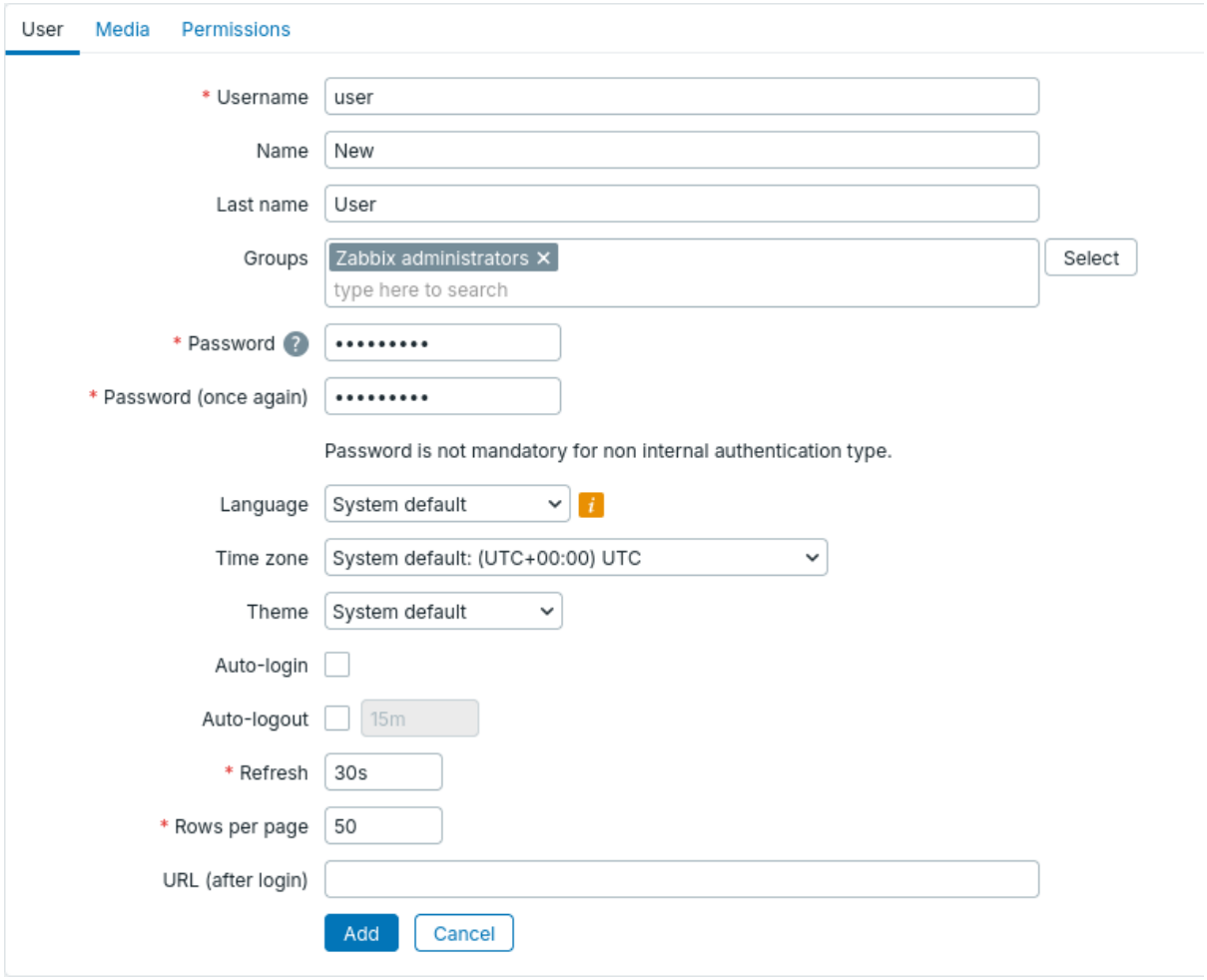
添加用户

要查看用户信息，请在侧边栏的垂直菜单中转到 *Users* > *Users*。



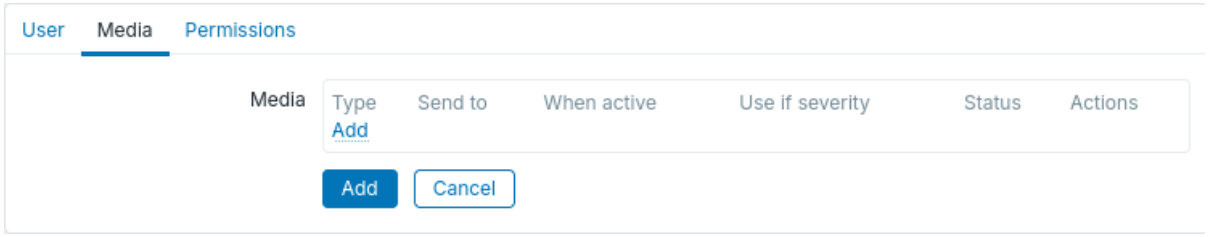
要添加新用户，请在右上角选择 *Create user*。

在新用户表单中，请确保将用户添加到现有的用户组 之一，例如 “Zabbix administrators”。



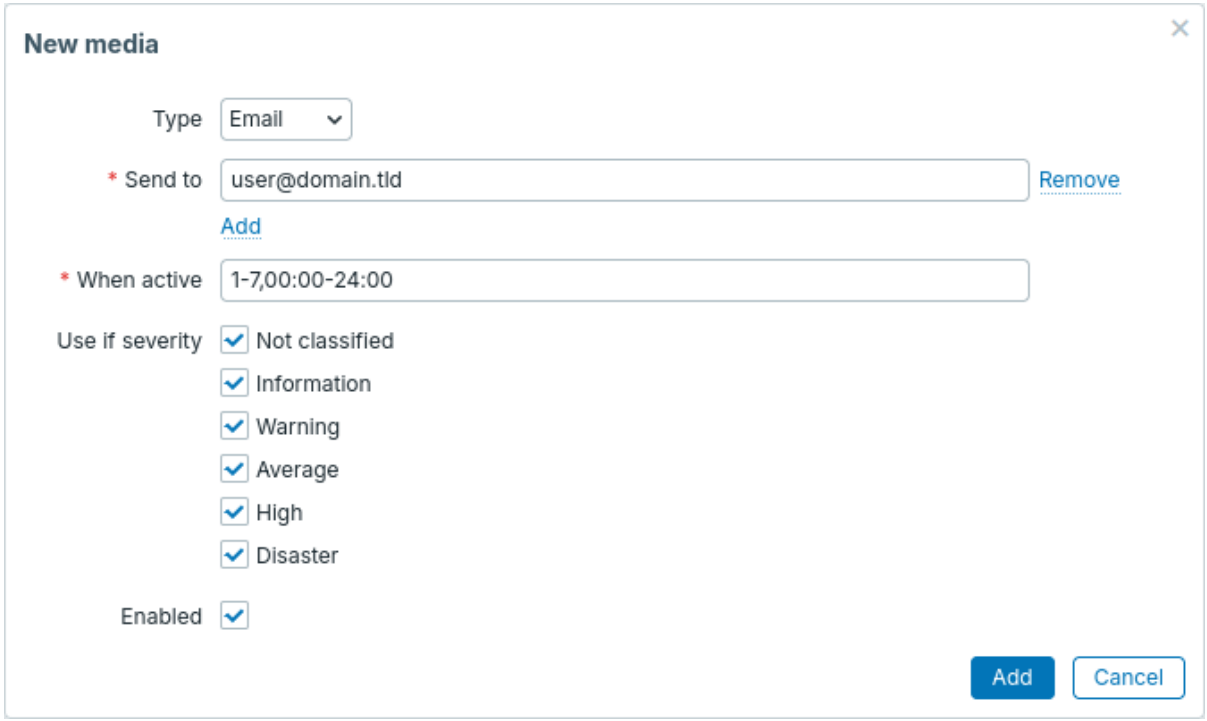
所有必填输入字段都用红色星号标记。有关此配置表单中输入字段的详细信息，请参阅User settings 页面。

默认情况下，新用户没有定义任何媒体（通知传递方式）。要创建一个媒体，请转到 *Media* 选项卡并单击 *Add*。



在弹出窗口中，输入用户的电子邮件地址。
 所有媒体类型默认都处于禁用状态，可在 *Alerts* > *Media types* 中启用。

您可以指定该媒体生效的时间段（格式说明请参见Time period specification 页面）。默认情况下，媒体始终处于活动状态。您还可以自定义该媒体生效的触发器严重性 级别，但现在请先保持全部启用。



单击 *Add* 保存媒体，然后转到 *Permissions* 选项卡。

Permissions 选项卡有一个必填字段 *Role*。该角色决定用户可以查看哪些前端元素，以及允许执行哪些操作。单击 *Select*，然后从列表中选择一个角色。例如，选择 *Admin role* 可允许访问除 *Administration* 之外的所有 Zabbix 前端部分。以后，您可以修改权限或创建更多用户角色。选择角色后，权限将显示在同一选项卡中：

Users

User | Media | Permissions

* Role: Admin role ✕ | Select

User type: Admin

Permissions

User group permissions

Group	Type	Permissions
All groups	Hosts	None
All groups	Templates	None

Proxies: None

Proxy groups: None

Permissions can be assigned for user groups only.

Access to UI elements

Dashboards: Dashboards

Monitoring: Problems | Hosts | Latest data | Maps | Discovery

APM: Traces | Metrics | Logs

Services: Services | SLA | SLA report

Inventory: Overview | Hosts

Reports: System information | Scheduled reports | Availability report | Top 100 triggers | Notifications

Data collection: Template groups | Host groups | Templates | Hosts | Maintenance | Discovery

Alerts: Trigger actions | Service actions | Discovery actions | Autoregistration actions | Internal actions

User profile settings

Redirect URL after login

Access to services

Read-write access to services: None

Read-only access to services: All

Access to modules

Action log | Clock | Discovery status | Favorite graphs | Favorite maps | Gauge | Geomap | Graph | Graph (classic) | Graph prototype | Honeycomb | Host availability | Host card | Host navigator | Item card | Item history | Item navigator | Item value | Map | Map navigation tree | Pie chart | Problem hosts | Problems | Problems by severity | Scatter plot | SLA report | System information | Top hosts | Top items | Top triggers | Trigger overview | URL | Web monitoring

Access to API

Disabled

Devices

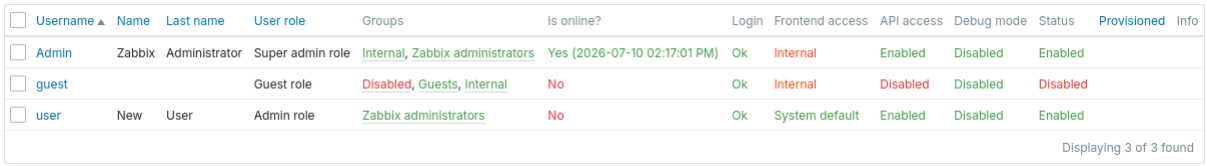
Enabled | Manage own devices

Access to actions

Create and edit dashboards | Create and edit maps | Create and edit maintenance | Add problem comments | Change severity | Acknowledge problems | Suppress problems | Close problems | Execute scripts | Manage API tokens | Manage scheduled reports | Manage SLA | Invoke "Execute now" on read-only hosts | Change problem ranking | Create and edit own media | Select "Server" for monitoring and discovery

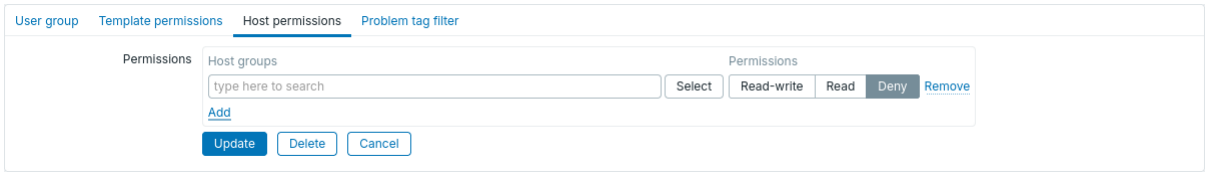
Add | Cancel

在用户属性表单中单击 *Add* 保存用户。新用户将出现在用户列表中。

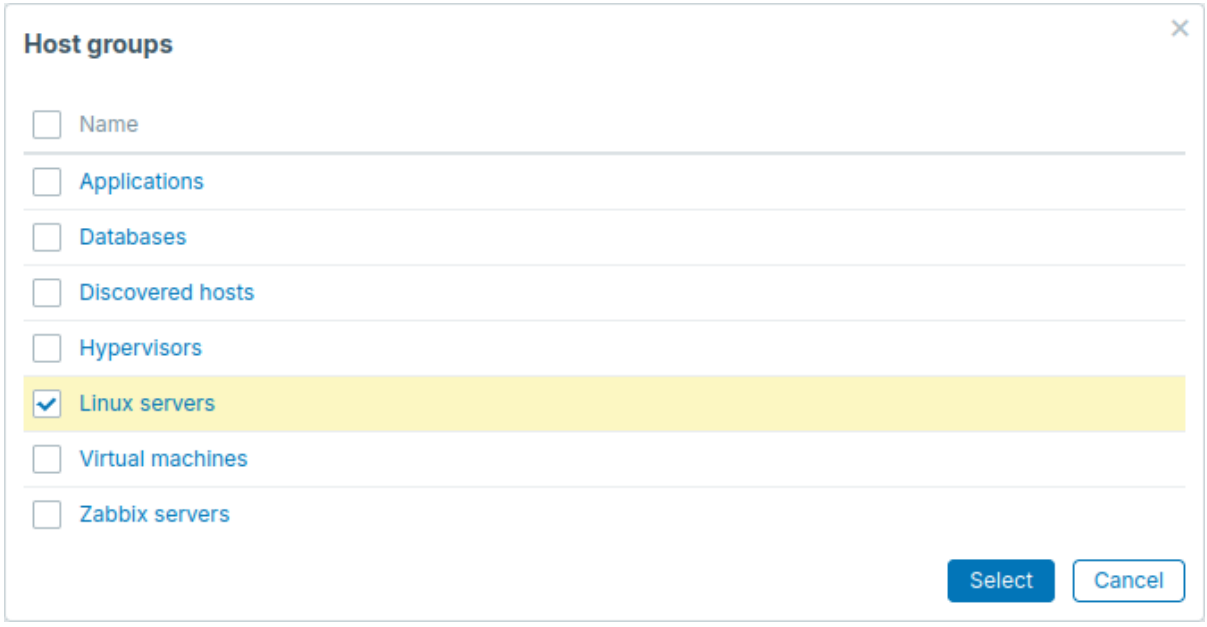


添加权限

默认情况下，新用户没有访问主机和模板的权限。要授予用户权限，请在 *Groups* 列中单击该用户所属的组（此处为 *Zabbix administrators*）。在 *User groups* 属性表单中，转到 *Host permissions* 选项卡，为主机组分配权限。单击主机组选择字段旁边的 Add 以显示该字段：



然后单击字段旁边的 *Select*，查看主机组列表。此用户需要对 *Linux servers* 组具有只读访问权限，因此请在列表中勾选相应的复选框，然后单击 *Select* 确认选择。



单击 *Read* 按钮设置权限级别，然后单击 *Update* 保存对用户组配置所做的更改。

要为模板授予权限，您需要切换到 *Template permissions* 选项卡并指定模板组。步骤与为主机组分配权限完全相同。有关模板的概述，请参见本快速入门中的New template 部分。

> **Attention:**
> 在 Zabbix 中，对主机和模板的访问权限是分配给user groups 的，而不是单个用户。

完成！您现在可以尝试使用新用户的凭据登录。

新主机

概述

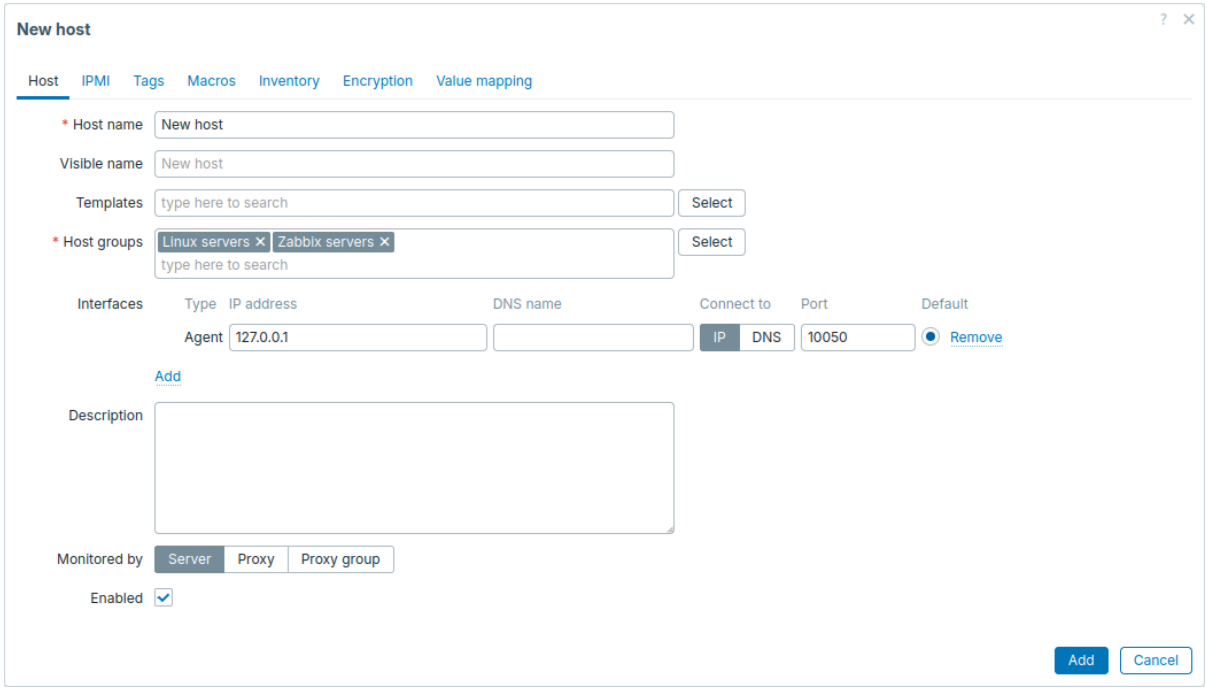
在本节中，您将了解如何设置一个新的主机。

Zabbix 中的主机是您希望监控的网络实体（物理或虚拟）。在 Zabbix 中，什么可以作为 “主机” 的定义相当灵活。它可以是物理服务器、网络交换机、虚拟机或某个应用程序。

添加主机

Zabbix 中已配置主机的信息可在数据采集 > 主机菜单部分以及监测 > 主机菜单部分查看。系统中已经预定义了一个名为 “Zabbix server” 的主机，但我们将学习如何添加另一台主机。

要添加新主机，请点击创建主机。系统将打开主机配置表单。



所有必填输入字段均以红色星号标记。

此处至少需要填写以下内容：

主机名称

- 输入主机名称。允许使用字母数字字符、空格、点、短横线和下划线。

主机群组

- 点击选择按钮选择一个或多个现有群组，或者输入不存在的群组名称以创建新群组。

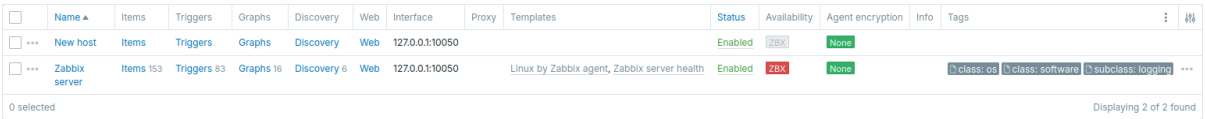
Note:
所有访问权限都分配给主机群组，而不是单个主机。因此，主机必须至少属于一个群组。

接口：***IP*** 地址

- 虽然从技术上讲这不是必填字段，但收集某些指标时需要主机接口。要使用 Zabbix agent 被动检查，请在此字段中指定 agent 的 IP 或 DNS。请注意，还应在 Zabbix agent 配置文件的 “Server” 指令中指定 Zabbix 服务器的 IP 或 DNS。如果 Zabbix agent 和 Zabbix 服务器安装在同一台计算机上，则需要在这两个位置指定相同的 IP/DNS。

其他选项可以保持不变，使用其默认值。

完成后，点击添加。新主机应显示在主机列表中。



可用性列包含每个接口的主机可用性指示器。定义 Zabbix agent 接口后，可以使用 agent 可用性图标（标记为 *ZBX*）来确定主机的可用性：

- ZBX - 尚未确定主机状态；尚未进行任何指标检查。
- ZBX - 主机可用，指标检查成功。
- ZBX - 主机不可用，指标检查失败（将鼠标光标移到图标上可查看错误消息）。通信可能存在错误，原因可能是接口凭据不正确。请检查 Zabbix 服务器是否正在运行，并稍后尝试刷新页面。

新监控项

概述

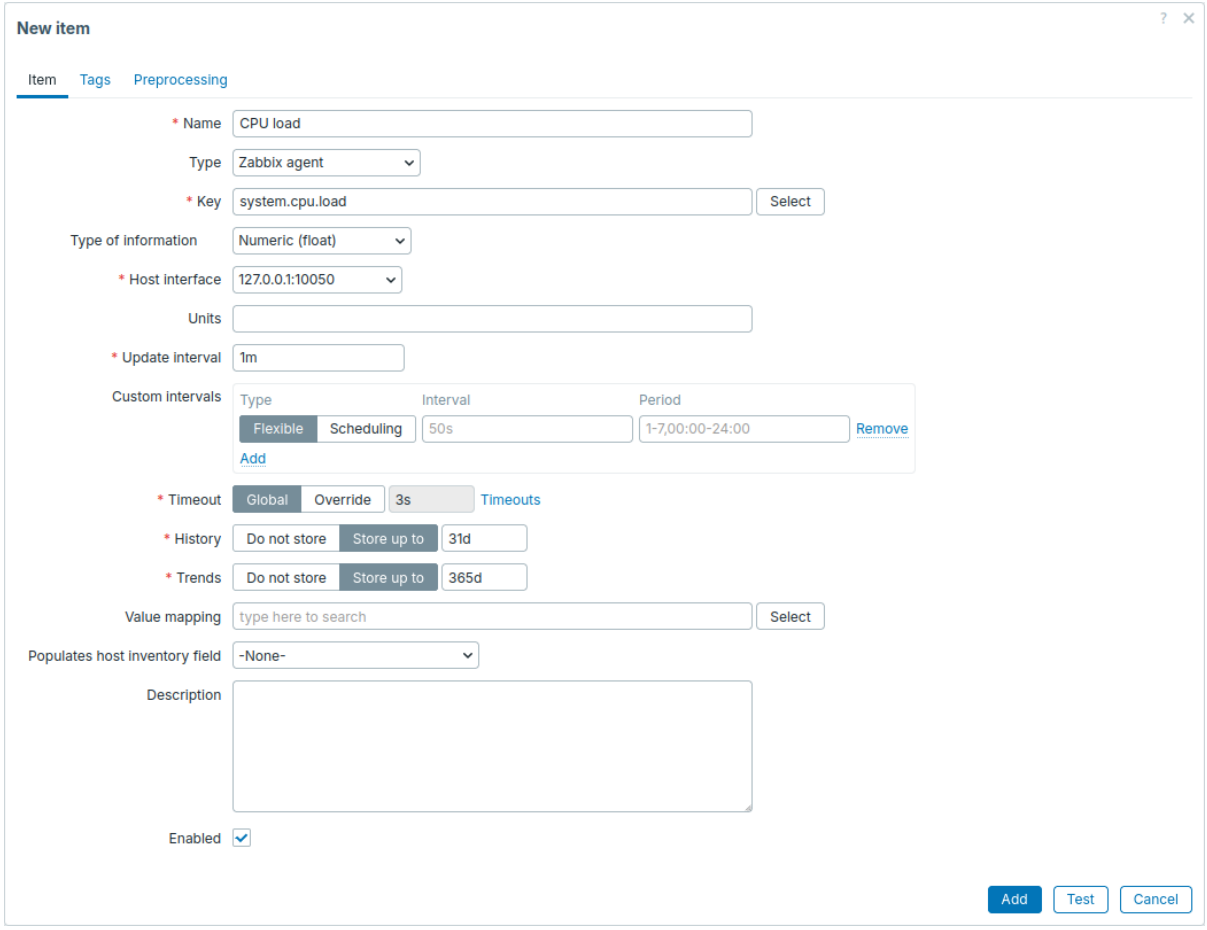
在本节中，您将了解如何设置监控项。

监控项是 Zabbix 中收集数据的基础。没有监控项，就没有数据——因为只有监控项才能定义单个指标，或者定义从主机收集哪类数据。

添加监控项

所有监控项都围绕主机进行分组。因此，要配置一个示例监控项，我们进入 *Data collection* > *Hosts*，并找到我们创建的"New host"。

点击"New host" 这一行中的 *Items* 链接，然后点击 *Create item*。此时会打开一个监控项配置表单。



所有必填输入字段都会用红色星号标记。

对于我们的示例监控项，需要输入的关键信息如下：

Name

- 输入 *CPU load* 作为值。这将是显示在列表及其他位置中的监控项名称。

Key

- 手动输入 *system.cpu.load* 作为值。这是监控项的技术名称，用于标识将要采集的信息类型。这个特定的 key 只是 Zabbix agent 自带的一个预定义 key。

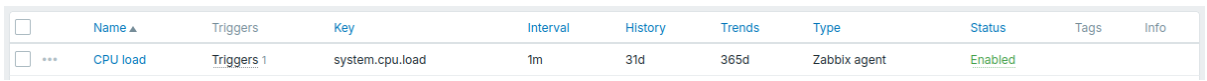
Type of information

- 此属性定义预期数据的格式。对于 *system.cpu.load* key，此字段会自动设置为 *Numeric (float)*。

Note:
你也可以将item history 的保留天数减少到 7 天或 14 天。这是一个好做法，可以减轻数据库保存大量历史值的负担。

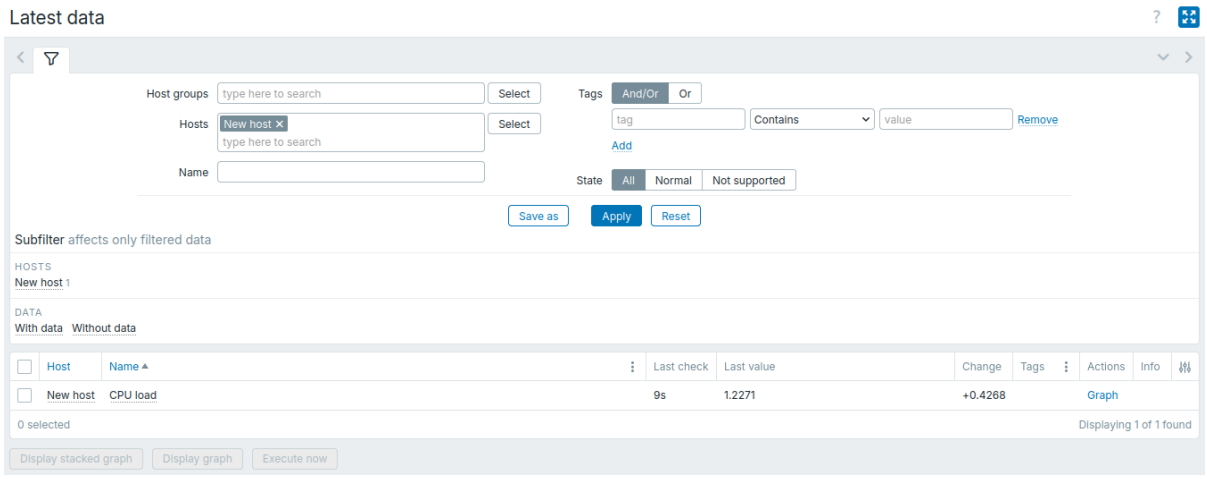
其他选项 可以保持不变，使用默认值。

完成后，点击 *Add*。新的监控项应会出现在监控项列表中。



查看数据

定义好监控项后，你可能会想知道它是否真的在收集数据。为此，请转到 *Monitoring* > *Latest data*，在筛选器中选择'New host'，然后点击 *Apply*。



不过，第一条数据最多可能需要 60 秒才会到达。默认情况下，服务器会以这个频率读取配置变更，并获取要执行的新监控项。

如果你在'Change' 列中看不到数值，可能目前只收到一个值。请等待 30 秒，看看是否会有另一个值到达。

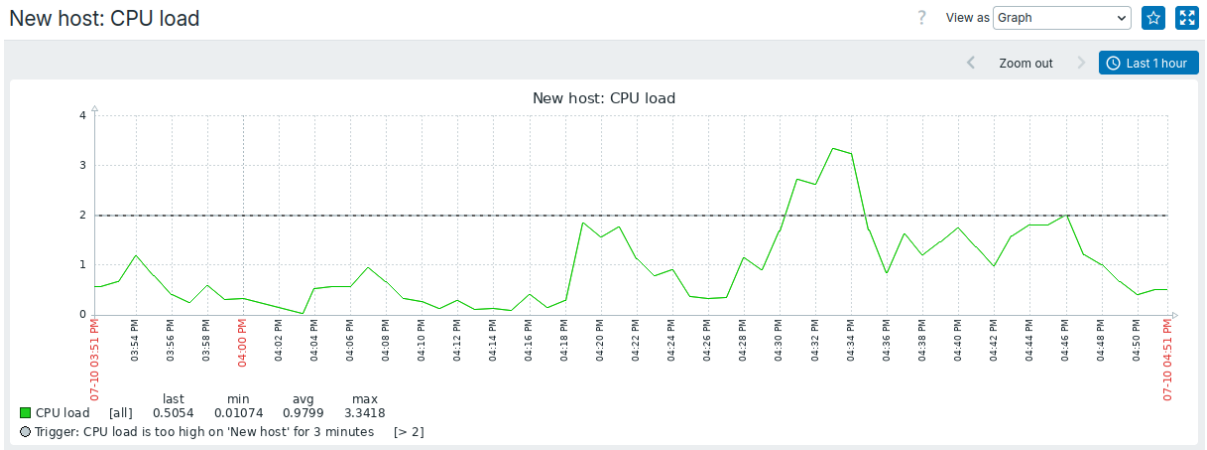
如果你没有像截图中那样看到有关监控项的信息，请确保：

- 你已按截图所示准确填写监控项的'Key' 和'Type of information' 字段；
- agent 和服务器都在运行；
- 主机状态为'Monitored'，且可用性图标为绿色；
- 在主机筛选器中选择的主机正确；
- 该监控项已启用。

图形

监控项运行一段时间后，您可能会希望查看一些可视化内容。无需进行任何额外配置，任何受监控的数值型监控项都可以使用简单图形。这些图形会在运行时生成。

要查看图形，请转到监测 > 最新数据，然后点击监控项旁边的图形链接。



新触发器

概述

在本节中，你将学习如何设置触发器。

监控项只负责收集数据。要自动评估传入的数据，我们需要定义触发器。触发器包含一个表达式，用于定义数据可接受水平的阈值。

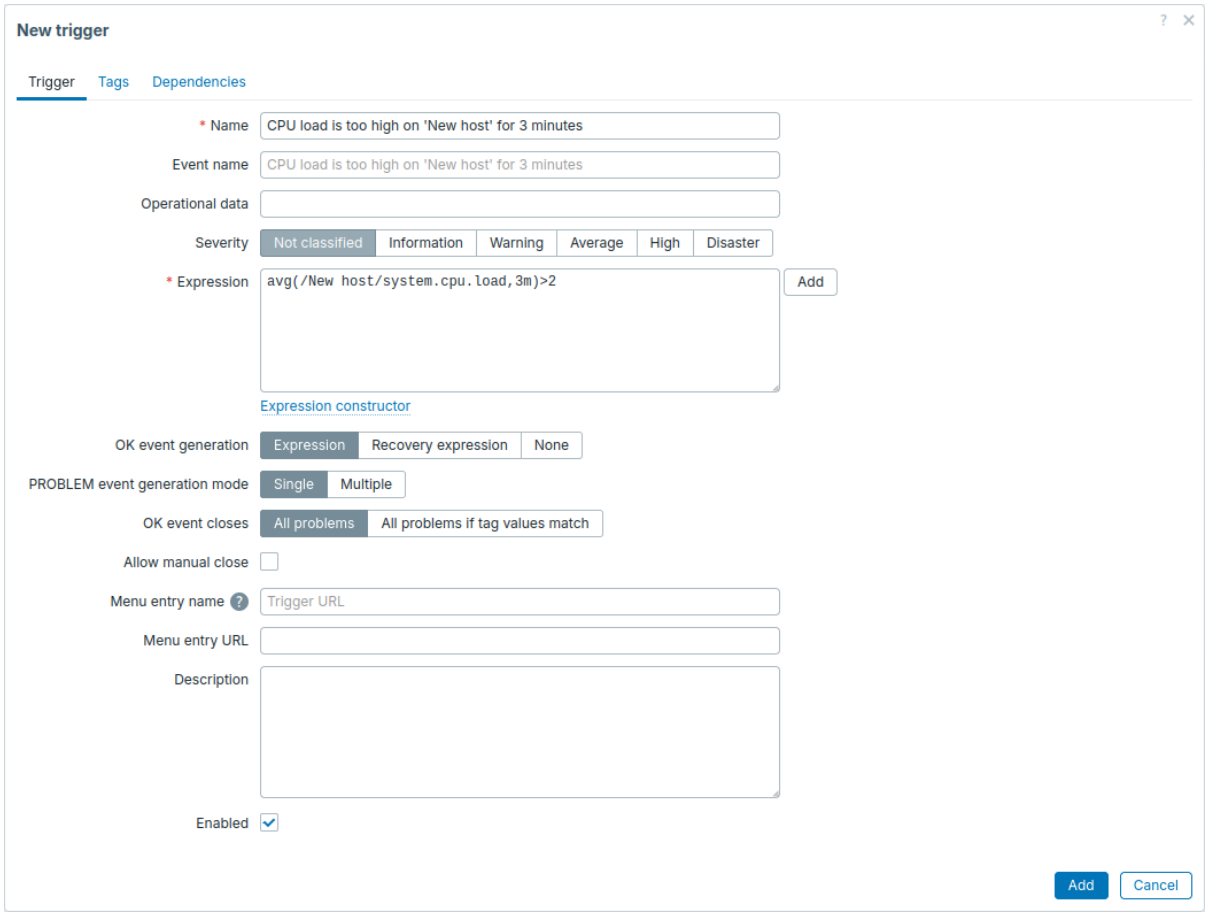
如果传入的数据超过了该水平，触发器将 "触发" 或进入 "Problem" 状态，表示发生了某些可能需要关注的情况。如果该水平再次恢复到可接受范围，触发器将返回到 "Ok" 状态。

添加触发器

要为我们的监控项配置触发器：

1. 转到 数据采集 > 主机。
2. 点击'New host' 所在行中的 触发器链接。
3. 点击 创建触发器。

将打开触发器配置表单。



对于我们的触发器，这里需要输入的关键信息是：

名称

- 输入 *CPU load is too high on 'New host' for 3 minutes* 作为值。这将是显示在列表及其他位置中的触发器名称。

表达式

- 输入：`avg(/New host/system.cpu.load,3m)>2`

这就是触发器表达式。请确保表达式输入正确，连最后一个符号也不要出错。这里的监控项键（`system.cpu.load`）用于引用该监控项。这个特定表达式的基本含义是：当 3 分钟内的 CPU 负载平均值超过 2 时，就会超过问题阈值。您可以进一步了解触发器表达式语法。

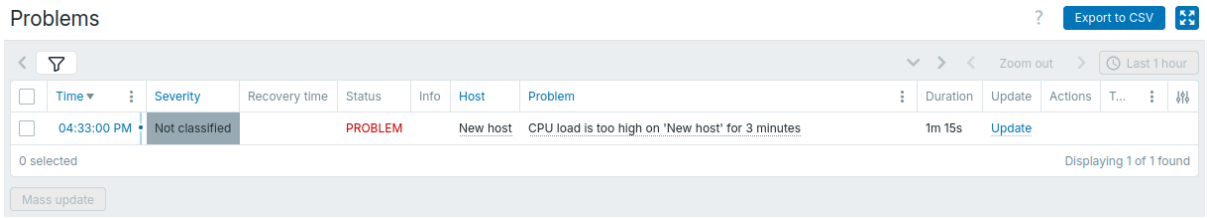
其他选项可以保持不变，使用默认值。

完成后，点击 添加。新的触发器应会出现在触发器列表中。

显示触发器状态

定义了触发器后，您可能会想查看其状态。

如果 CPU 负载超过了您在触发器中定义的阈值，问题将显示在 监控 > 问题中。



状态列中的闪烁表示触发器状态最近发生了变化，即在过去 30 分钟内发生的变化。

接收问题通知

概述

在本节中，您将学习如何在 Zabbix 中以通知的形式设置告警。

当监控项收集数据、触发器在出现问题情况时设计为“触发”时，即使没有主动查看 Zabbix 前端，配置某种告警机制来通知重要事件也会很有用。

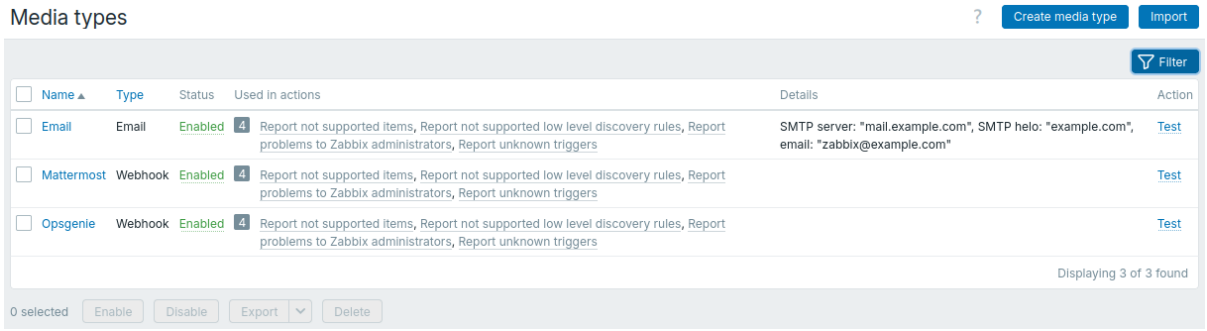
这就是通知的作用。

电子邮件是问题通知最常用的传递方式，因此我们将学习如何设置电子邮件通知。

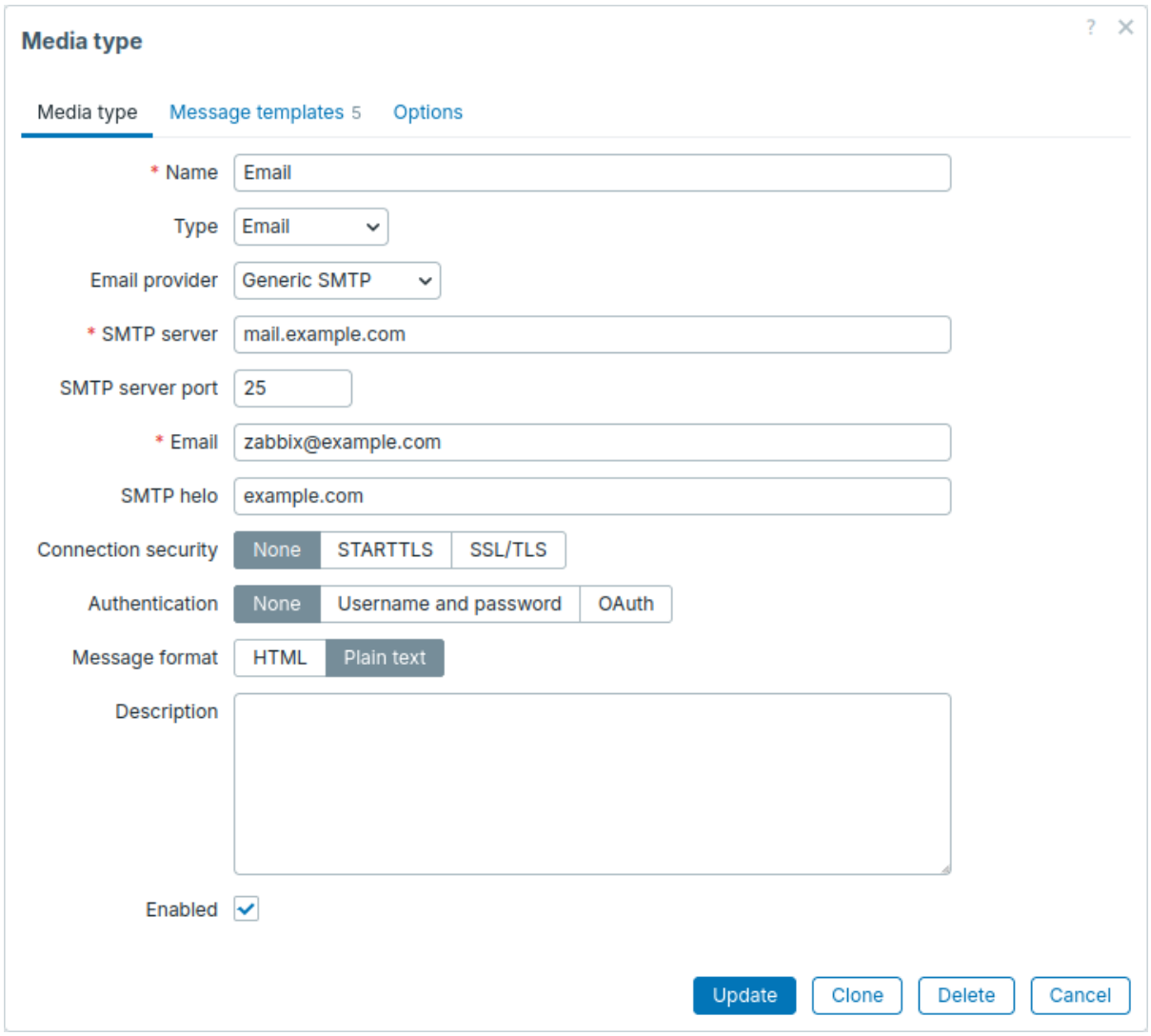
Email 设置

最初，Zabbix 中已经预定义了几种通知传递方式。Email 就是其中之一。

要配置 email 设置，请转到 *Alerts* > *Media types*，然后在预定义的媒体类型列表中单击 *Email*。



此时会打开 email 设置配置表单。



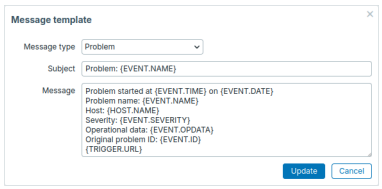
所有必填输入字段都用红色星号标记。

在 *Media type* 选项卡中，将 *SMTP server*、*SMTP helo* 和 *Email* 的值设置为适合您环境的配置。

Note:
Email 字段中的值将用作从 Zabbix 发送的通知的“发件人”地址。

接下来，需要定义问题消息的内容。该内容通过消息模板来定义，并在 *Message templates* 选项卡中进行配置。

单击 *Add* 创建消息模板，并选择 *Problem* 作为消息类型。

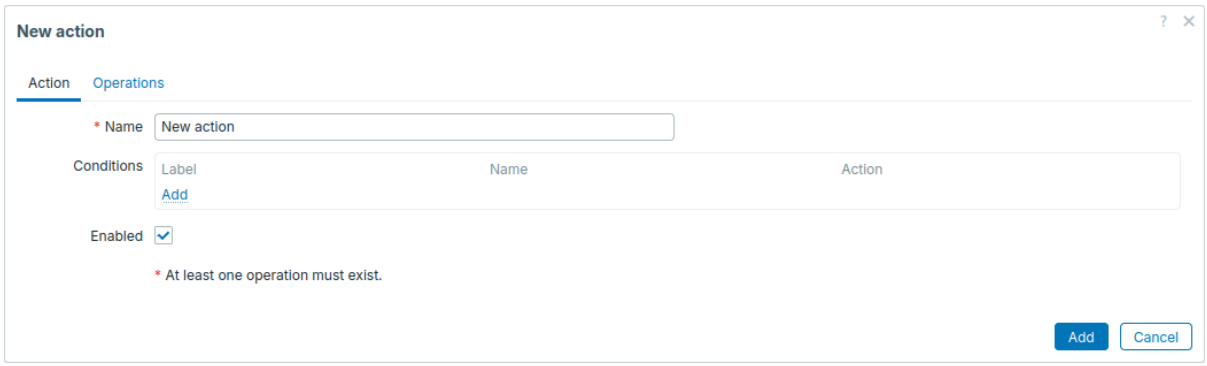


准备好后单击 *Add* 并保存表单。

现在您已经将 *Email* 配置为可用的媒体类型。还必须通过定义具体的传递地址将该媒体类型关联到用户，否则它不会被使用。

新建动作

在 Zabbix 中，发送通知是动作 的功能之一。
因此，要设置通知，请转到 *Alerts* > *Actions* > *Trigger actions*，然后点击 *Create action*。

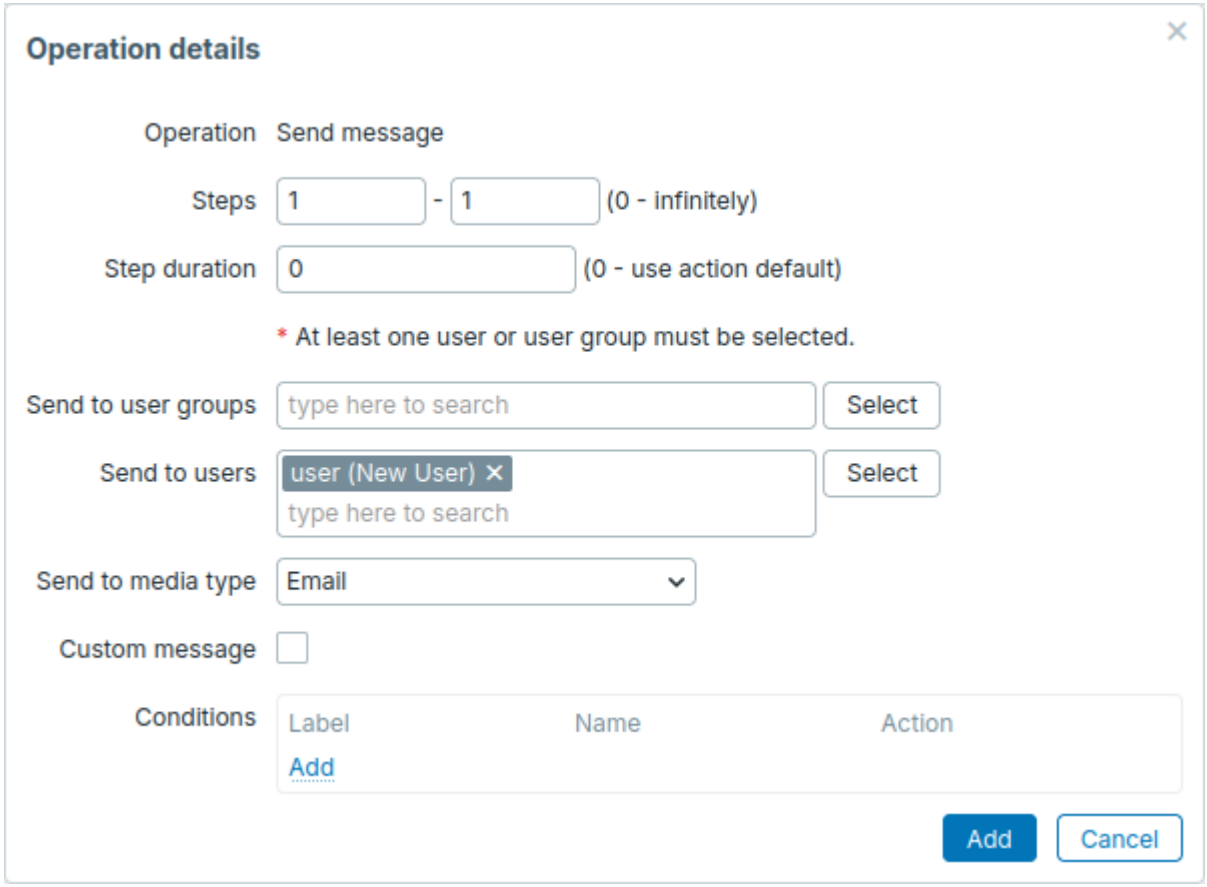


所有必填输入字段都用红色星号标记。

在此表单中，为该动作输入一个名称。

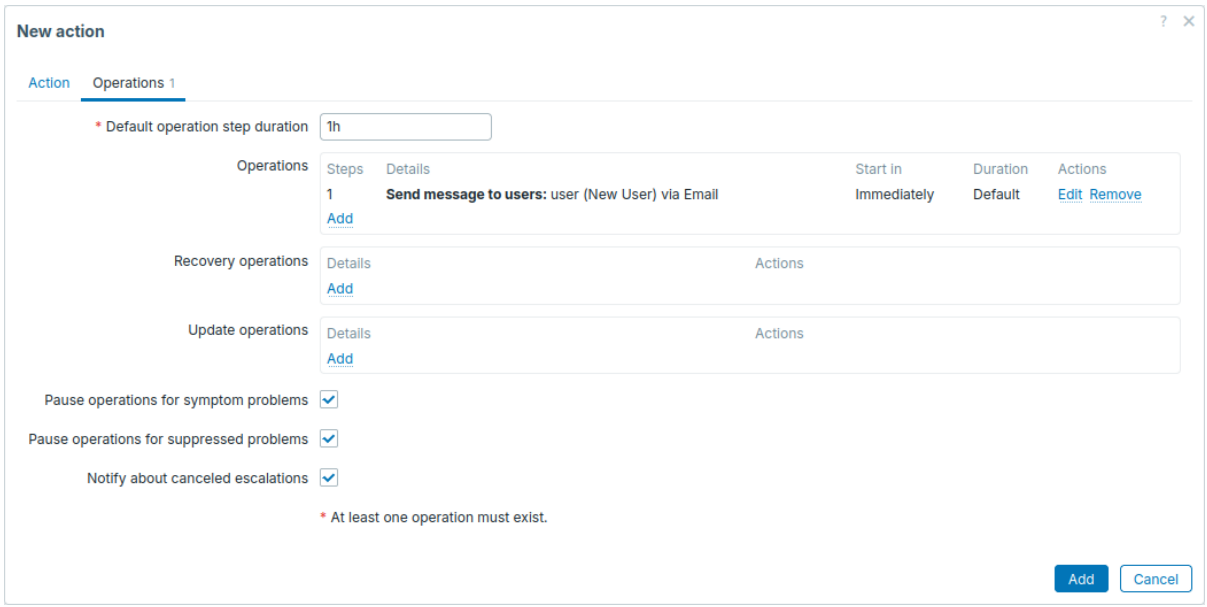
在最简单的情况下，如果我们不再添加任何更具体的条件，那么当任何触发器状态从 “Ok” 变为 “Problem” 时，就会执行该动作。

我们仍然需要定义该动作应执行什么操作，这在 *Operations* 选项卡中完成。
在 *Operations* 区块中点击 *Add*，这将打开一个新的操作表单。



所有必填输入字段都用红色星号标记。

在这里，点击 *Send to Users* 区块中的 *Select*，并选择我们已定义的用户（'user'）。
将 *Send to media type* 的值选择为"Email"。
完成后，点击 *Add*，该操作应会被添加：



对于一个简单的动作配置，这就是全部内容，因此请在动作表单中点击 *Add*。

接收通知

现在，既然已经配置好了发送通知，真正收到一条通知就很有意思了。
为此，我们可以故意增加主机上的负载，这样我们的触发器 就会 "触发"，并收到一条问题通知。

在主机上打开控制台并运行：

```
cat /dev/urandom | md5sum
```

你可以运行一个或多个 这些进程。

现在转到 *Monitoring* > *Latest data*，查看'CPU Load' 的值是否已经升高。请记住，要让我们的触发器 "触发"，'CPU Load' 的值必须连续 3 分钟超过'2'。
一旦满足条件：

- 在 *Monitoring* > *Problems* 中，你应该会看到该触发器处于闪烁的 **Problem** 状态；
- 你应该会在电子邮件中收到一条问题通知。

Attention:
如果通知不起作用：

- 再次确认电子邮件设置和动作都已正确配置
- 确保你创建的用户至少对生成事件的主机具有读取权限，如 添加用户 步骤中所述。该用户作为'Zabbix administrators' 用户组的一员，必须至少对该主机所属的'Linux servers' 主机组具有读取权限。
- 此外，你还可以通过进入 *Reports* > *Action log* 查看动作日志。

新模板

概述

在本节中，您将了解如何设置模板。

之前我们学习了如何设置监控项和触发器，以及如何获取主机的问题通知。

虽然这些步骤本身都提供了很大的灵活性，但如果需要对例如一千台主机执行这些操作，步骤可能会显得过多。此时，自动化功能将非常有用。

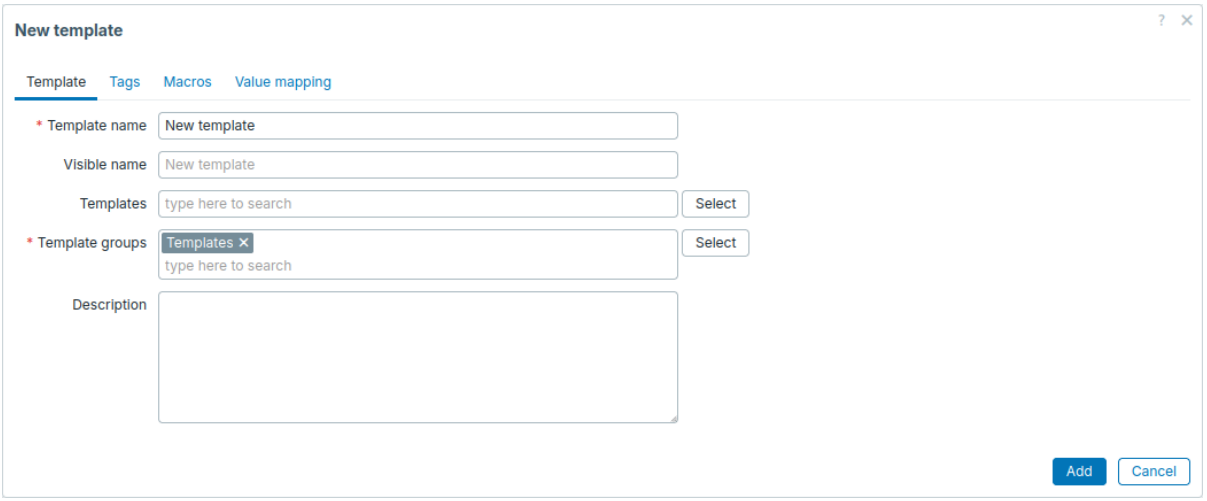
模板正是为此而设计的。

模板允许您将有用的监控项、触发器和其他实体进行分组，以便通过一步操作将其应用于主机，从而反复重复使用。

将模板链接到主机后，主机会继承模板中的所有实体。因此，基本上可以非常快速地应用一组预先准备好的检查。

添加模板

要开始使用模板，我们首先必须创建一个模板。为此，在 数据采集 > 模板中点击 创建模板。这将打开一个模板配置表单。



所有必填输入字段都用红色星号标记。

这里需要输入的参数如下：

模板名称

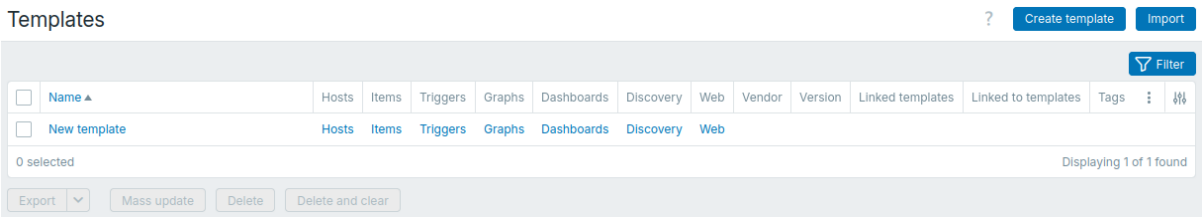
- 输入模板名称。允许使用字母数字字符、空格和下划线。

模板组

- 通过点击 选择按钮选择一个或多个组。模板必须属于某个组。

Note:
对模板组的访问权限是在用户组 配置中的 模板权限选项卡上分配的，方式与主机权限相同。所有访问权限都是分配给组，而不是单个模板，因此必须至少将模板加入一个组。

完成后，点击 添加。你的新模板应会显示在模板列表中。你也可以使用过滤器 来查找你的模板。

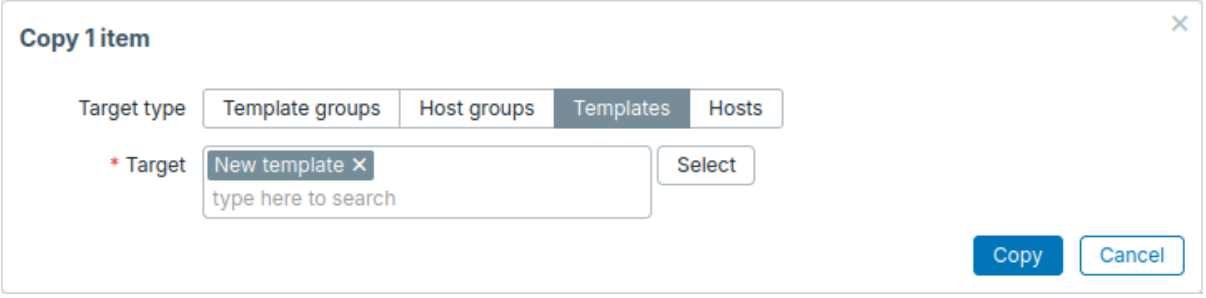


如你所见，模板已经在那里，但其中还没有任何内容 - 没有监控项、触发器或其他实体。

将监控项添加到模板

要将监控项添加到模板：

1. 转到 数据采集 > 主机。
2. 在列表中找到 “New host”，然后点击 监控项超链接。
3. 在列表中勾选 “CPU Load” 监控项。
4. 点击列表下方的 复制。
5. 选择 模板选项卡。
6. 选择要将监控项复制到的模板。



所有必填输入字段均以红色星号标记。7. 点击 复制。

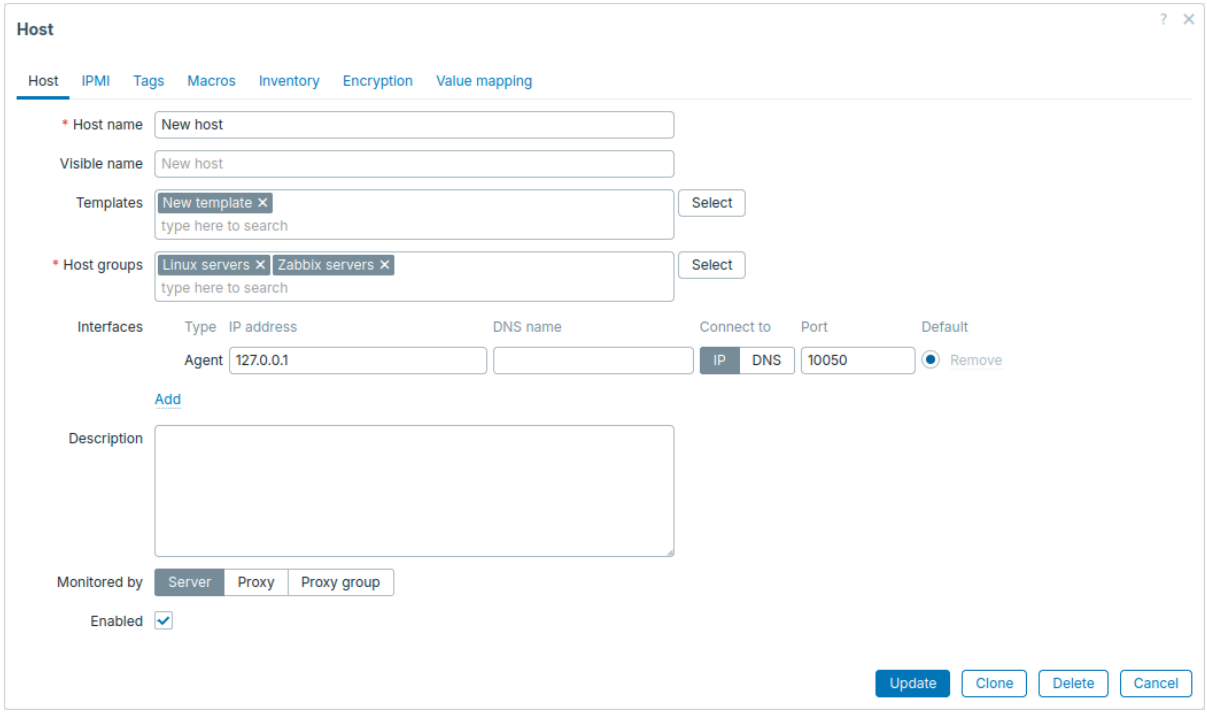
现在，如果转到 数据采集 > 模板，“New template” 中应该会有一个新的监控项。

目前我们只添加一个监控项，但同样地，您可以继续向模板添加其他监控项、触发器或其他实体，直到它成为一组相当完整的实体，以满足特定用途（监控操作系统、监控单个应用程序）。

将模板链接到主机

模板准备就绪后，只需将其添加到主机即可。操作步骤如下：

1. 转到 数据采集 > 主机。
2. 单击 “新建主机” 以打开其配置表单，然后找到 模板字段。
3. 在 模板字段中开始输入 新建模板。新创建的模板名称应显示在下拉列表中。
4. 向下滚动并选择该模板。确认它已显示在 模板字段中。



5. 在表单中单击 更新以保存更改。

现在，模板已添加到主机，并包含该模板所持有的所有实体。

这样也可以将其应用于其他主机。对模板级别的监控项、触发器和其他实体所做的任何更改，都会传播到链接了该模板的主机。

将预定义模板链接到主机

您可能已经注意到，Zabbix 自带一组适用于各种操作系统、设备和应用程序的预定义模板。为了快速开始监控，您可以将其中合适的模板链接到主机，但请注意，这些模板需要根据您的环境进行细致调整。有些检查可能并不需要，而轮询间隔也可能过于频繁。

有关模板的更多信息，请参阅相关说明。

Apache web server 监控

要开始监控 Apache web server，请按以下步骤操作：

1. 确保可以访问 Apache 统计信息。
2. 应用 Apache 监控模板。
3. 查看最新数据。

要使用本指南，您必须已安装 Zabbix 服务器和 web 前端installed。您必须以默认的 *Admin* 用户登录 Zabbix。

访问 Apache 统计信息

此步骤的目标是确保 Zabbix 能够读取 Apache web 服务器统计信息。

在 Apache 中，服务器性能和活动的实时信息由内置的 `mod_status` 模块提供。启用后，它会创建一个可通过 web 访问的页面（通常位于 `/server-status`），用于显示性能和活动数据。

要验证 Apache web 服务器是否已启用 `mod_status`，请运行：

在 Debian/Ubuntu 上	在基于 RHEL 的系统上
`apache2ctl -M | grep status`	`httpd -M | grep status`

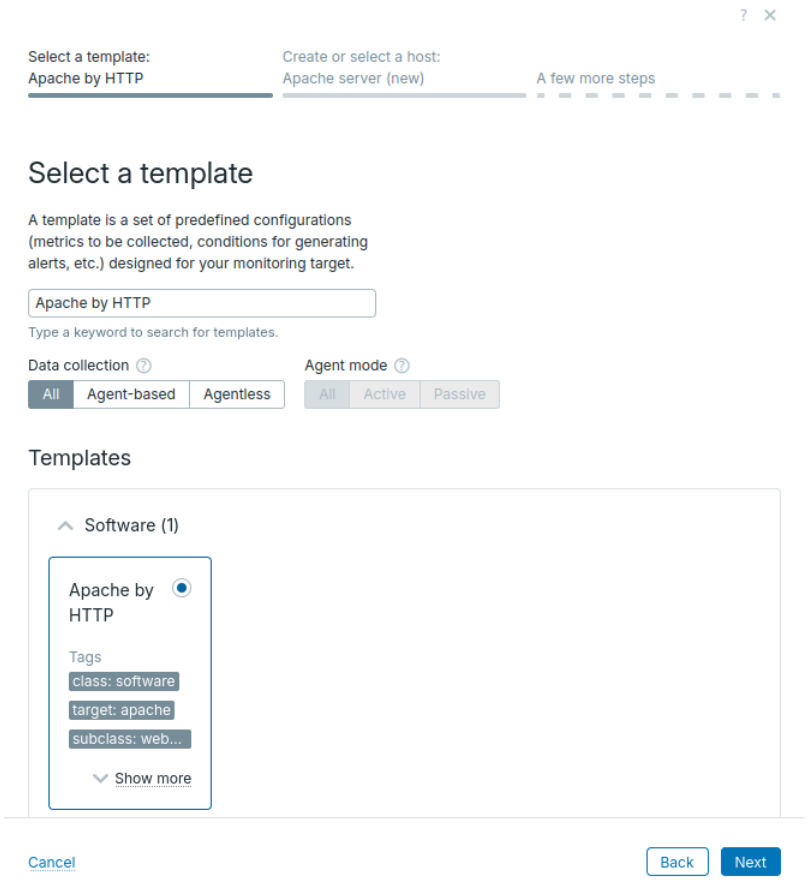
如果输出中看到"`status_module (shared)`"，则表示状态模块已经启用。否则，您需要启用状态模块。

应用 Apache 监控模板

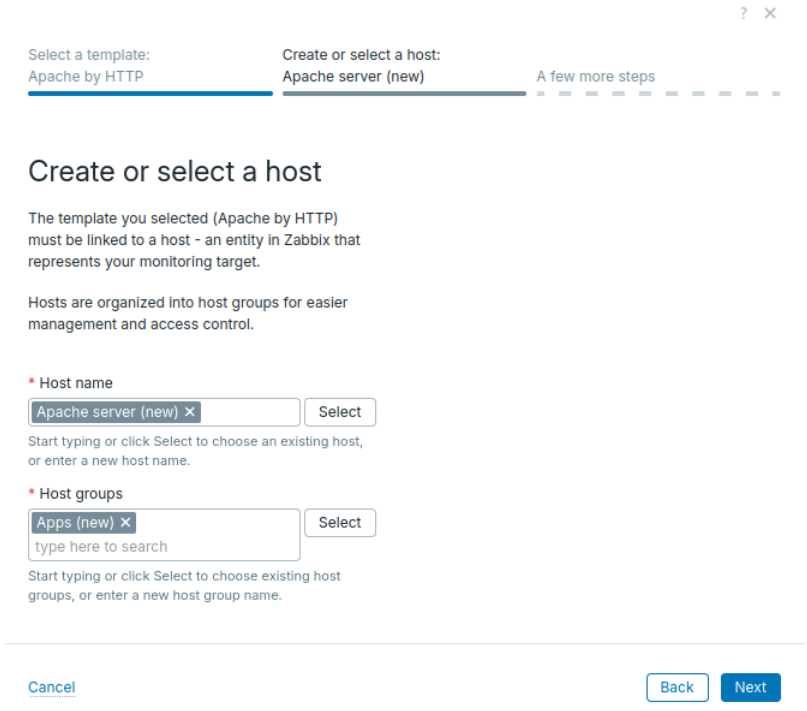
在 Zabbix 中，此模板可直接使用。它包含预先配置好的元素，可用于数据采集、可视化和分析。

要应用该模板：

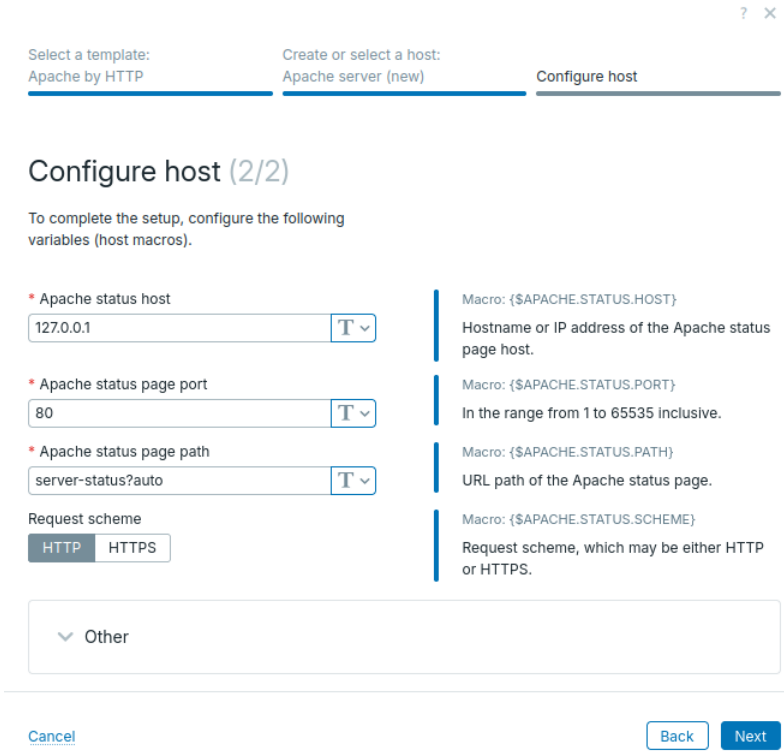
1. 在 数据采集 > 主机中启动主机向导。
2. 选择 *Apache by HTTP* 模板。



3. 创建一个 Apache 主机，并为其添加一个组。
   这个虚拟主机将包含与 Apache 相关的数据。



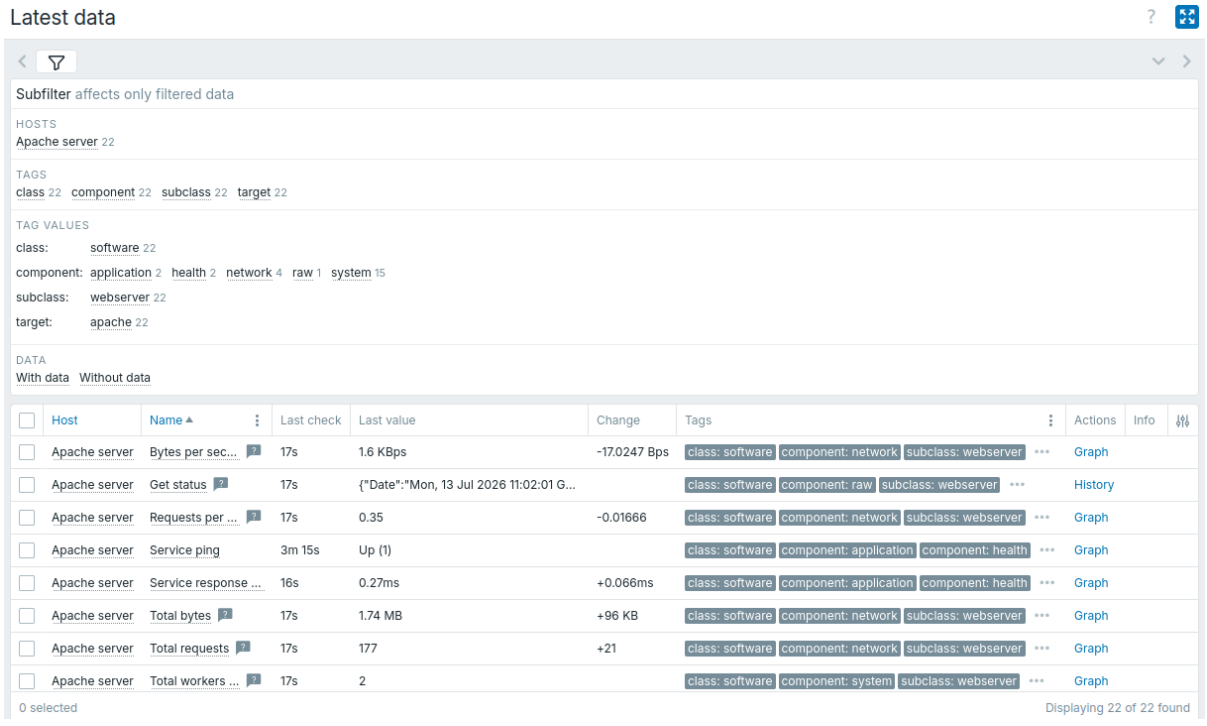
4. 配置主机。
   重要的是，输入 Apache web 服务器地址（本地机器上为 127.0.0.1，远程机器上为 IP 地址）。



其他设置可以保留默认值。

5. 要完成主机向导，请点击 创建和 完成。

查看最新数据



恭喜，Apache web 服务器监控已启动！

Zabbix 提供可视化选项，并在出现任何问题时发送告警。要配置电子邮件告警，请参见接收问题通知。

Linux 服务器监控

要开始监控 Linux 服务器，请按以下流程操作：

1. 应用 Linux 监控模板。
2. 在服务器上安装 Zabbix agent。
3. 查看最新数据。

要开始监控非本地 Linux 服务器，您需要知道其 IP 地址以及 Zabbix 服务器的 IP 地址。

将使用 Linux 服务器上的 Zabbix agent 来监控它。安装 agent 的说明将在流程中提供。

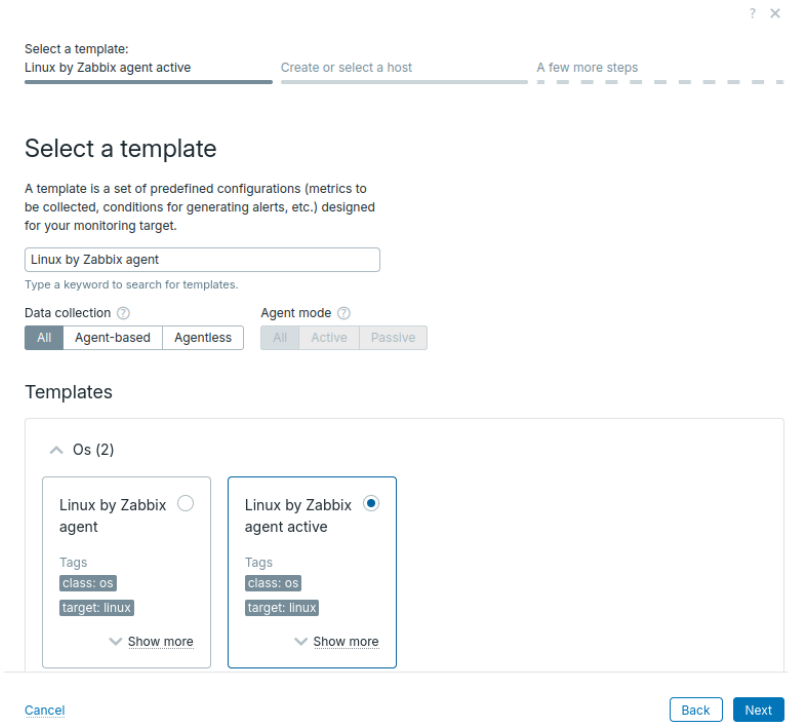
要使用本指南，您必须已安装 Zabbix 服务器和 Web 界面installed。您必须以默认的 *Admin* 用户登录 Zabbix。

应用 Linux 监控模板

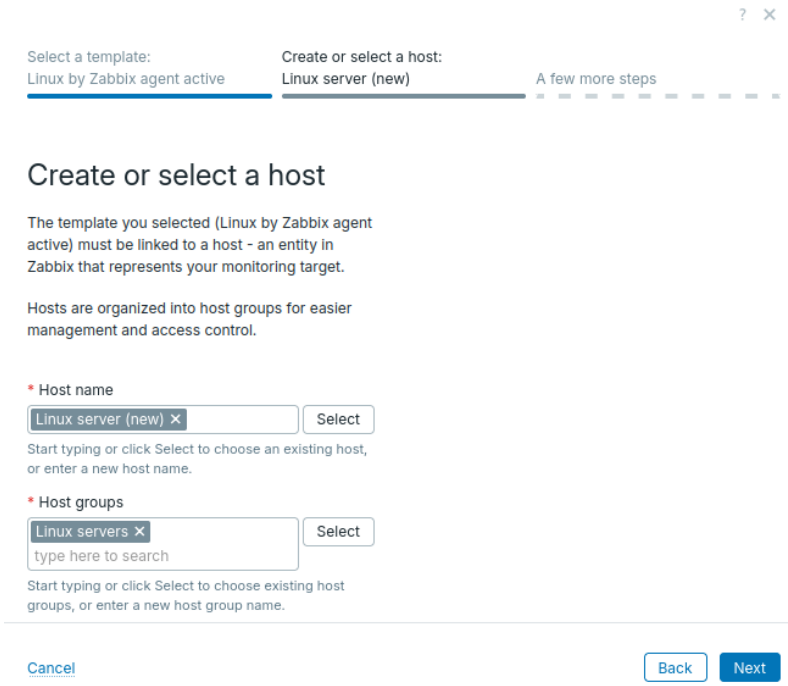
在 Zabbix 中，此模板已可直接使用。它包含预先配置好的元素，可用于数据采集、可视化和分析。

要应用该模板：

1. 在 数据采集 > 主机中启动主机向导。
2. 选择 *Linux by Zabbix agent active* 模板。



3. 为 Linux 服务器创建一个主机，并为其添加一个组。此虚拟主机将包含与 Linux 服务器监控相关的所有数据。



4. 配置并安装 agent。

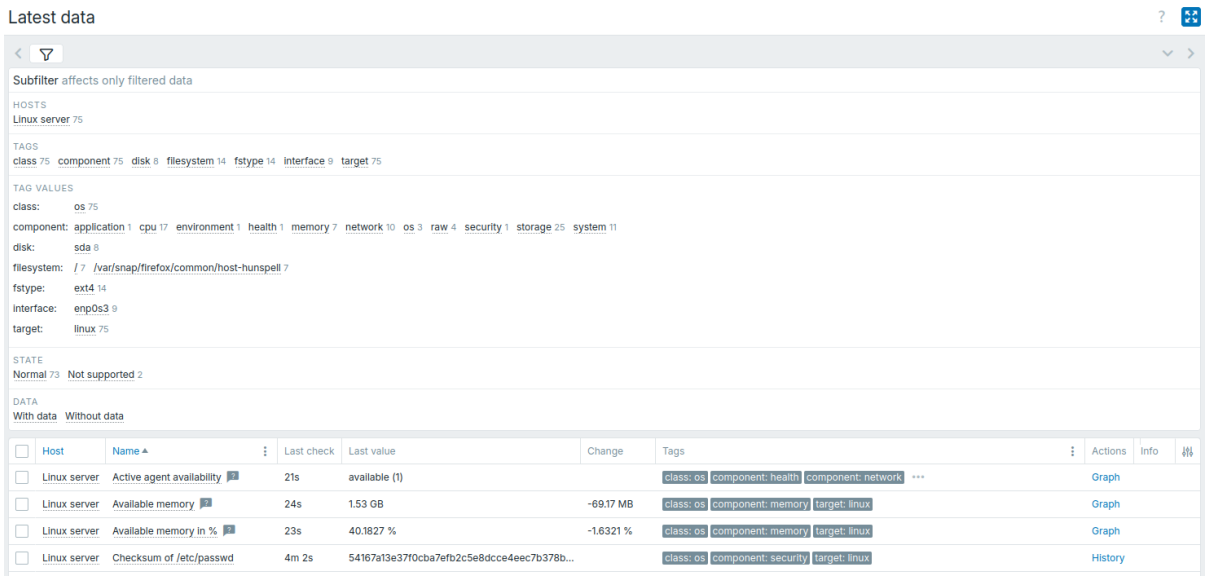


要安装 agent，请在 Linux 服务器上执行提供的命令。这将安装 agent，并使用加密方式配置 agent 与 Zabbix 服务器的连接。
 如果您已经安装了 agent，可以跳过此步骤（点击 下一步）。不过，系统会要求您提供预共享密钥标识和预共享密钥，以便连接到 agent。

其他设置可保持默认值。

5. 完成主机配置：点击 创建和 完成。

查看最新数据



恭喜，Linux 服务器监控已启动！

Zabbix 提供可视化选项，并在出现任何问题时发送告警。要配置电子邮件告警，请参见接收问题通知。

MySQL 服务器监控

要开始监控 MySQL 服务器，请按照以下流程操作：

1. 创建受限的 MySQL 用户。
2. 应用 MySQL 监控模板。
3. 查看最新数据。

要开始监控 MySQL 服务器，您需要知道其 DSN，以及 Zabbix 服务器的 IP 地址。

将使用 Zabbix agent 来监控 MySQL 服务器。安装 agent 的说明将在流程中提供。

要使用本指南，您必须已安装 Zabbix 服务器和 web 前端。您必须以默认的 *Admin* 用户登录 Zabbix。

创建权限受限的 MySQL 用户

建议创建一个 MySQL 用户，仅授予监控 MySQL 所需的权限。将 <password> 替换为强度高且唯一的密码。

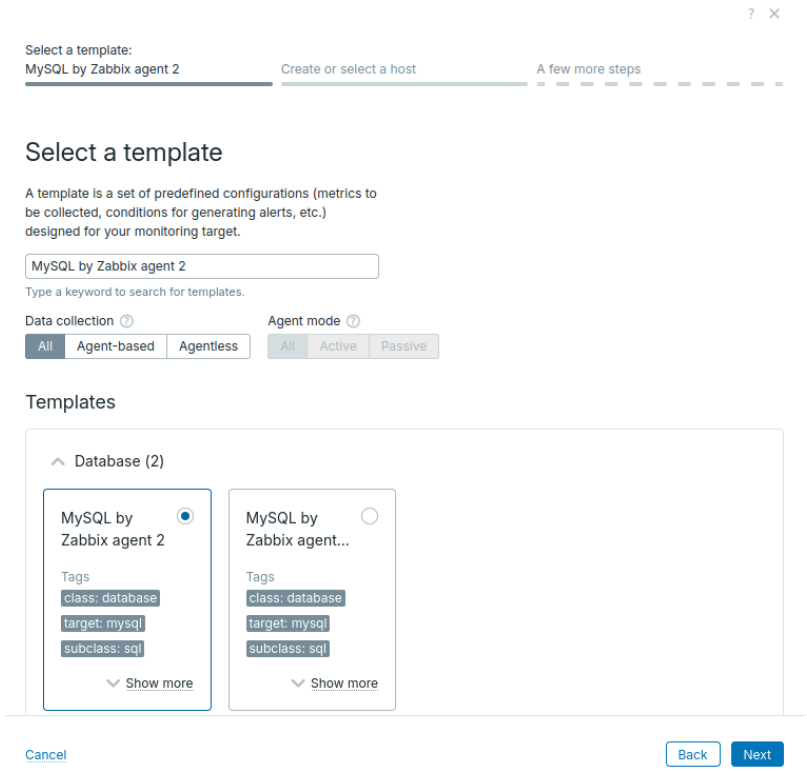
```
mysql> CREATE USER 'zbx_monitor'@'%' IDENTIFIED BY '<password>';
mysql> GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';
```

应用 MySQL 监控模板

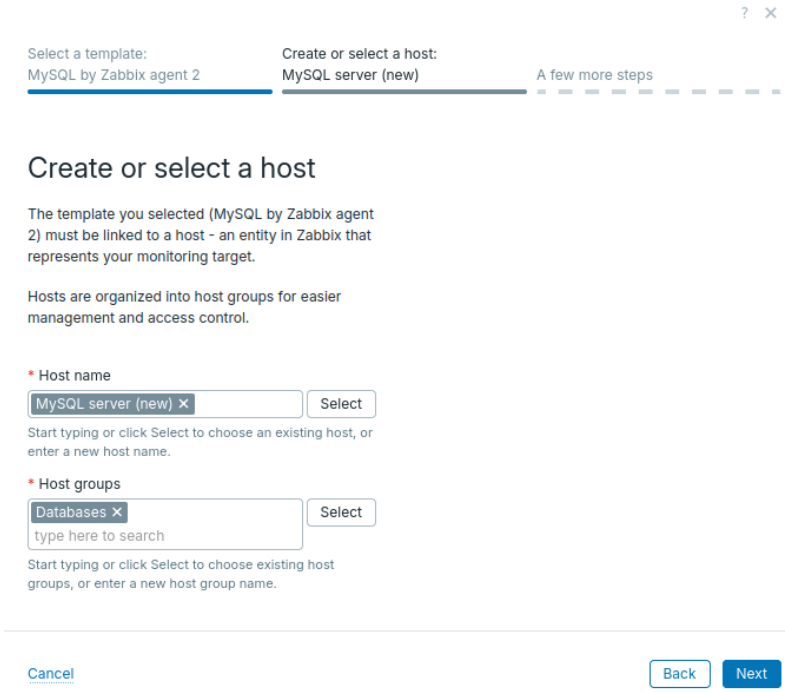
在 Zabbix 中，此模板已可直接使用。它包含预先配置好的元素，可用于数据采集、可视化和分析。

要应用该模板：

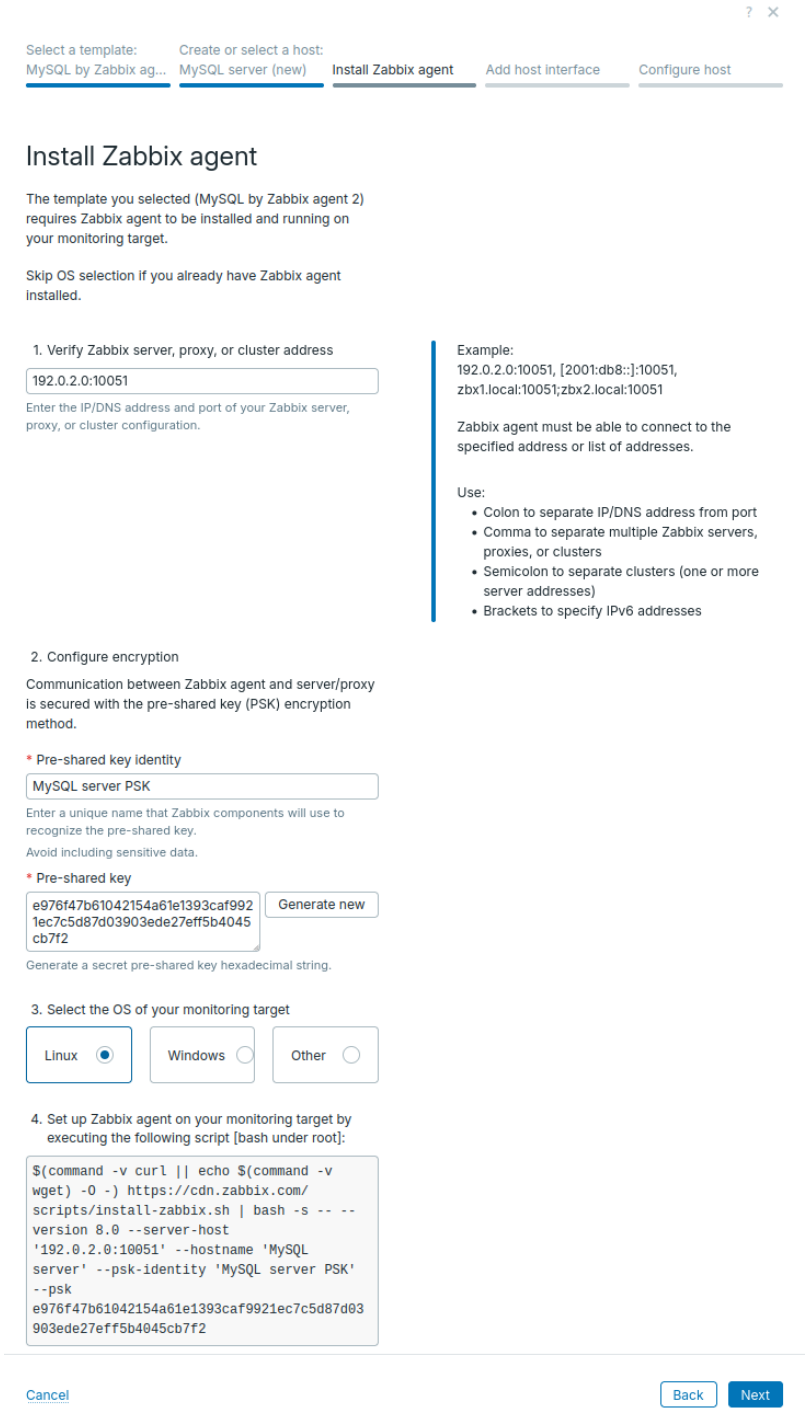
1. 在 数据采集 > 主机中启动主机向导。
2. 选择 *MySQL by Zabbix agent 2* 模板。



3. 创建一个 MySQL 服务器主机，并为其添加一个组。这个虚拟主机将包含与 MySQL 服务器监控相关的所有数据。

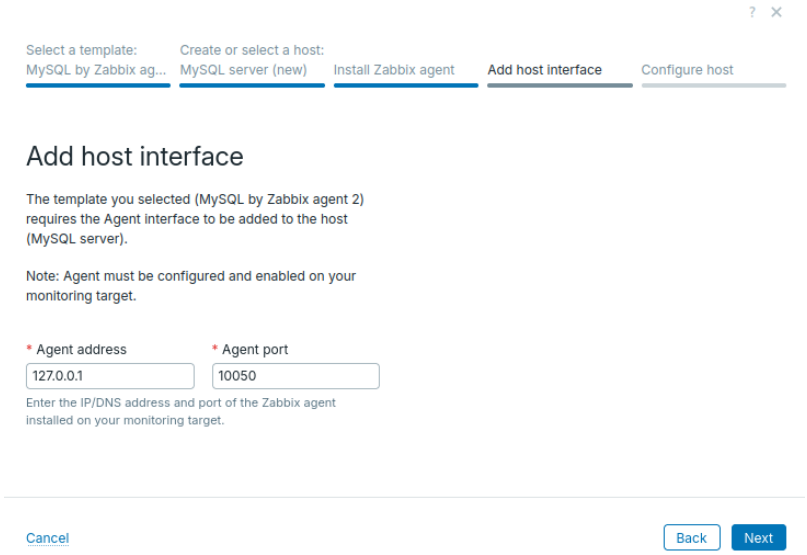


4. 配置并安装 agent 2（Zabbix agent 2 开箱即用地提供 MySQL 监控）。

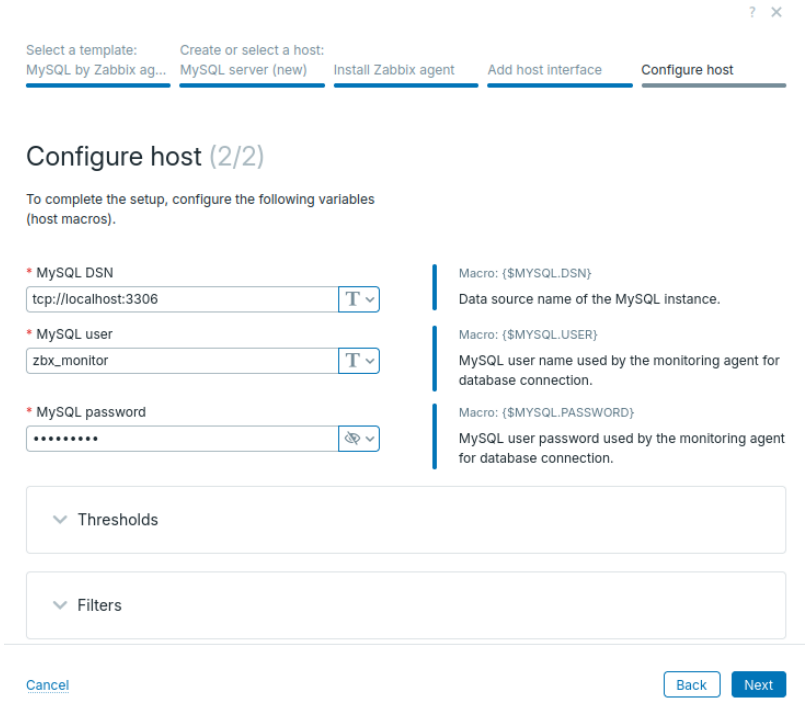


要安装 agent，请在托管 MySQL 服务器的机器上执行所提供的命令。这将安装 agent，并使用加密方式配置 agent 与 Zabbix 服务器的连接。
 如果你已经安装了 agent 2，可以跳过此步骤（点击 下一步）。不过，系统会要求你提供预共享密钥标识和预共享密钥，以便连接到 agent。

5. 输入 agent（Linux 服务器）的 IP 地址和端口。



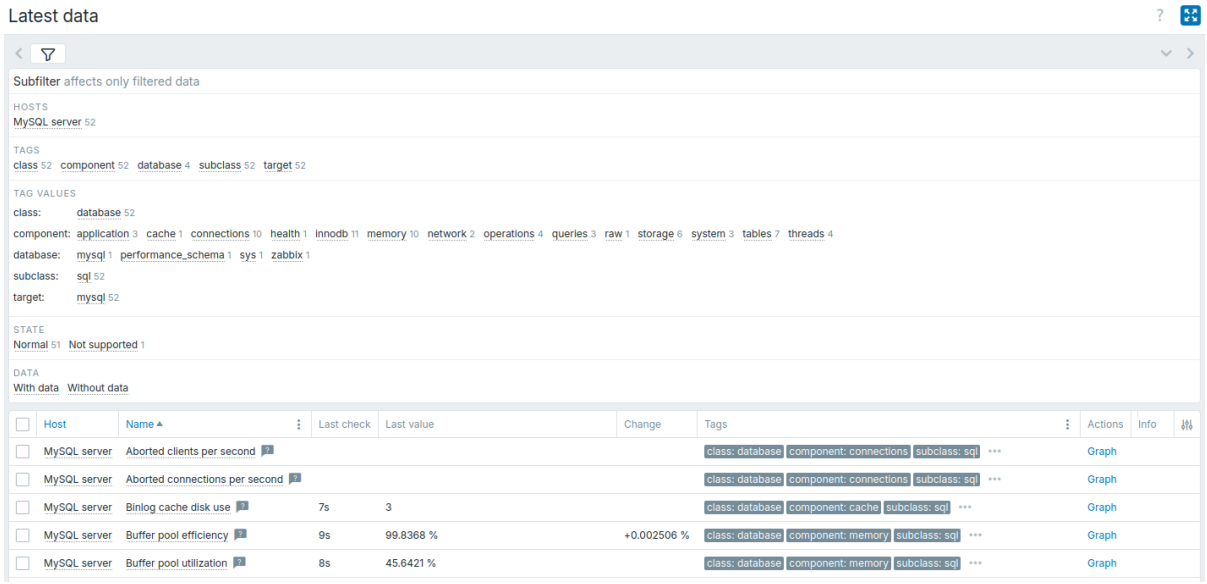
6. 完成主机配置。重要的是，系统会要求你输入 MySQL 服务器 DSN、MySQL 用户名和密码。请提供你创建的 `zbx_monitor` 用户及其密码。



其他设置可以保持默认值。

6. 要完成主机向导，请点击 创建和 完成。

查看最新数据



恭喜，MySQL 服务器监控已启动！

Zabbix 提供可视化选项，并在出现任何问题时发送告警。要配置电子邮件告警，请参见接收问题通知。

更多指南

本 документация 部分包含用于为一些常见的监控目标配置 Zabbix 的快速方案。

它面向 Zabbix 新用户设计，可作为导航器，帮助您浏览其他文档部分，这些部分包含解决该任务所需的信息。

以下快速参考指南可用：

- 使用 Zabbix agent 监控 Linux
- 使用 Zabbix agent 监控 Windows
- 通过 HTTP 监控 Apache
- 使用 Zabbix agent 2 监控 MySQL
- 使用 Zabbix 监控 VMware
- 使用 Zabbix 监控网络流量
- 使用主动检查通过 Zabbix 监控网络流量
- 使用浏览器监控项监控网站
- 使用 Zabbix agent 2（被动）监控网站证书
- 使用 Zabbix 监控网络交换机或路由器
- 使用主动检查监控 Windows 事件日志

使用 **Zabbix agent** 监控 **Linux**

介绍

本页面将引导您完成使用 Zabbix 开始对 Linux 机器进行基础监控所需的步骤。
本教程中描述的步骤适用于任何基于 Linux 的操作系统。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用对您的 Linux 机器进行基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或者需要更高级的配置，请参阅 Zabbix 手册中的Configuration 部分。

先决条件

在继续阅读本指南之前，您需要根据适用于您操作系统的说明，下载并安装 Zabbix 服务器和 Zabbix 前端。

安装 Zabbix agent

Zabbix agent 是负责收集数据的进程。

检查您的 Zabbix 服务器版本：

```
zabbix_server -V
```

在您要监控的 Linux 机器上安装相同版本的 Zabbix agent（推荐）。
根据您的监控需求，这台机器可以是安装了 Zabbix 服务器的同一台机器，也可以是完全不同的机器。

选择最适合的安装方式：

- 作为 Docker 容器运行 - 请参阅 Zabbix Docker 仓库 中可用镜像列表。
- 从 Zabbix 软件包 安装（适用于 Alma Linux、CentOS、Debian、Oracle Linux、Raspberry Pi OS、RHEL、Rocky Linux、SUSE Linux Enterprise Server、Ubuntu）。
- 从源代码 编译。

配置 Zabbix 进行监控

Zabbix agent 可以以主动或被动模式（同时）收集指标。

> **Note:**
> 被动检查是一种简单的数据请求。Zabbix 服务器或 proxy 会请求某些数据（例如 CPU 负载），然后 Zabbix agent 将结果发送回服务器。主动检查需要更复杂的处理。agent 必须先从服务器中检索一个监控项列表以进行独立处理，然后再批量将数据发送回去。更多信息请参见被动和主动 agent 检查。

Zabbix 提供的监控模板通常有两种选择 - Zabbix agent 模板和 Zabbix agent（主动）模板。使用第一种选项时，agent 将以被动模式收集指标。此类模板会提供相同的监控结果，但使用不同的通信协议。

后续的 Zabbix 配置取决于您选择的是主动 还是被动 Zabbix agent 检查模板。

被动检查

Zabbix agent

1. 在安装 agent 的计算机上打开agent 配置文件。

```
sudo vi /etc/zabbix/zabbix_agentd.conf
```

2. 将 Zabbix 服务器的 IP 地址或 DNS 名称（以及可选端口）添加到 server 参数中。例如：

```
Server=192.0.2.1:10051
```

Zabbix agent 将使用此地址，仅接受来自指定 Zabbix 服务器或 proxy 的传入连接，以检索数据。

3. 重启 Zabbix agent。

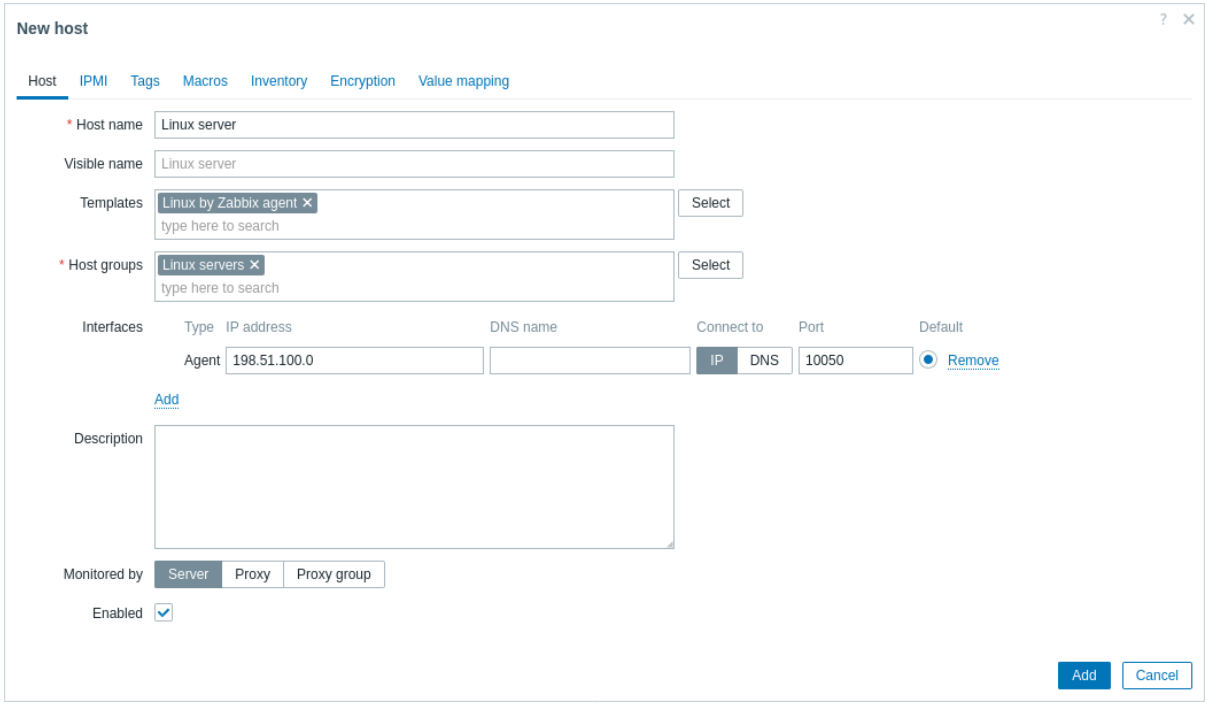
```
systemctl restart zabbix-agent
```

Zabbix 前端

1. 登录 Zabbix 前端。

2. 在 Zabbix web 前端中创建主机。

- 在 *Host name* 字段中，输入主机名称（例如，“Linux server”）。
- 在 *Templates* 字段中，输入或选择将要链接到主机的模板 “Linux by Zabbix agent”。
- 在 *Host groups* 字段中，输入或选择主机组（例如，“Linux servers”）。
- 在 *Interfaces* 参数中，添加 *Agent* 接口，并指定安装了 agent 的 Linux 机器的 IP 地址或 DNS 名称。



3. 单击 *Add* 以添加主机。此主机将代表被监控的 Linux 机器。

主动检查

Zabbix agent

1. 在安装 agent 的机器上打开agent 配置文件：

```
sudo vi /etc/zabbix/zabbix_agentd.conf
```

2. 将 Zabbix 服务器的 IP 地址或 DNS 名称（以及可选端口）添加到 ServerActive 参数中。例如：

```
ServerActive=192.0.2.1:10051
```

Zabbix agent 将使用此地址连接到 Zabbix 服务器的 trapper 端口（默认值：10051），并请求主动检查所需的配置数据。

3. 定义 Hostname 参数。该参数必须与将在Zabbix 前端中定义的主机名匹配。在本例中为：

```
Hostname=Linux server
```

Hostname 值必须匹配，因为对于主动检查，Zabbix agent 会使用该值从服务器检索正确的主机配置。具体来说，agent 会发起与服务器的连接，并使用 Hostname 值标识自身。然后，服务器会提供该主机的监控配置。如果这些值不一致，agent 将无法接收适当的配置，从而导致指标缺失或出现监控问题。

4. 重启 Zabbix agent。

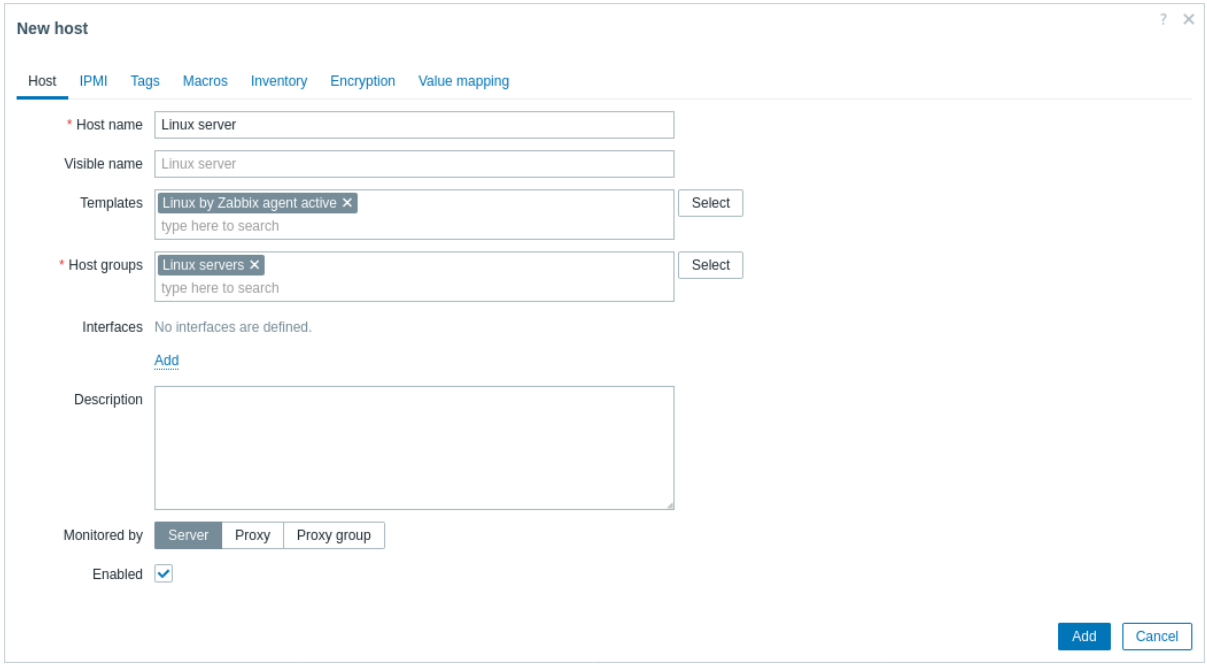
```
systemctl restart zabbix-agent
```

Zabbix 前端

1. 登录 Zabbix 前端。

2. 创建主机 在 Zabbix web interface 中。

- 在 *Host name* 字段中，输入一个主机名（例如，“Linux server”），该名称应与前面在 agent 配置文件中定义的 *Hostname* 参数值匹配。
- 在 *Templates* 字段中，输入或选择模板 “Linux by Zabbix agent active”，该模板将被链接到主机。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，“Linux servers”）。
- 保持 *Interfaces* 未定义；对于主动检查，不需要接口，因为 agent 会主动发起到服务器的连接，而不是监听来自服务器的连接。

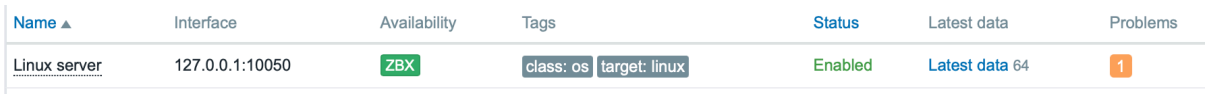


3. 点击 *Add* 以添加主机。此主机将代表被监控的 Linux 机器。

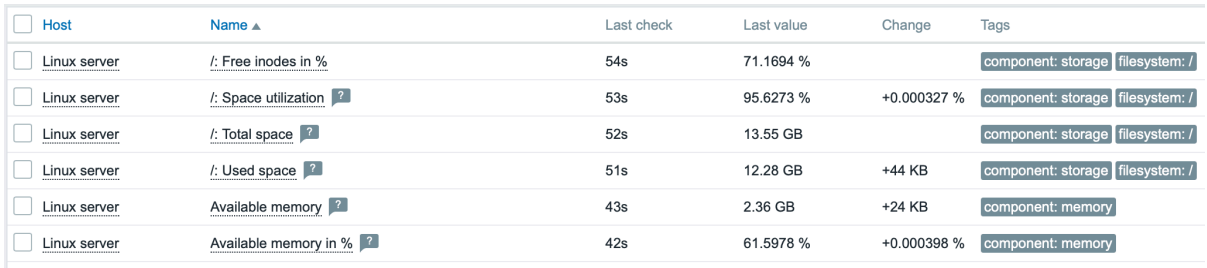
查看已收集的监控项

恭喜！此时，Zabbix 已经在监控你的 Linux 机器了。

要查看已收集的监控项，请打开 *Monitoring* > *Hosts* 菜单部分，然后点击主机旁边的 *Latest data*。



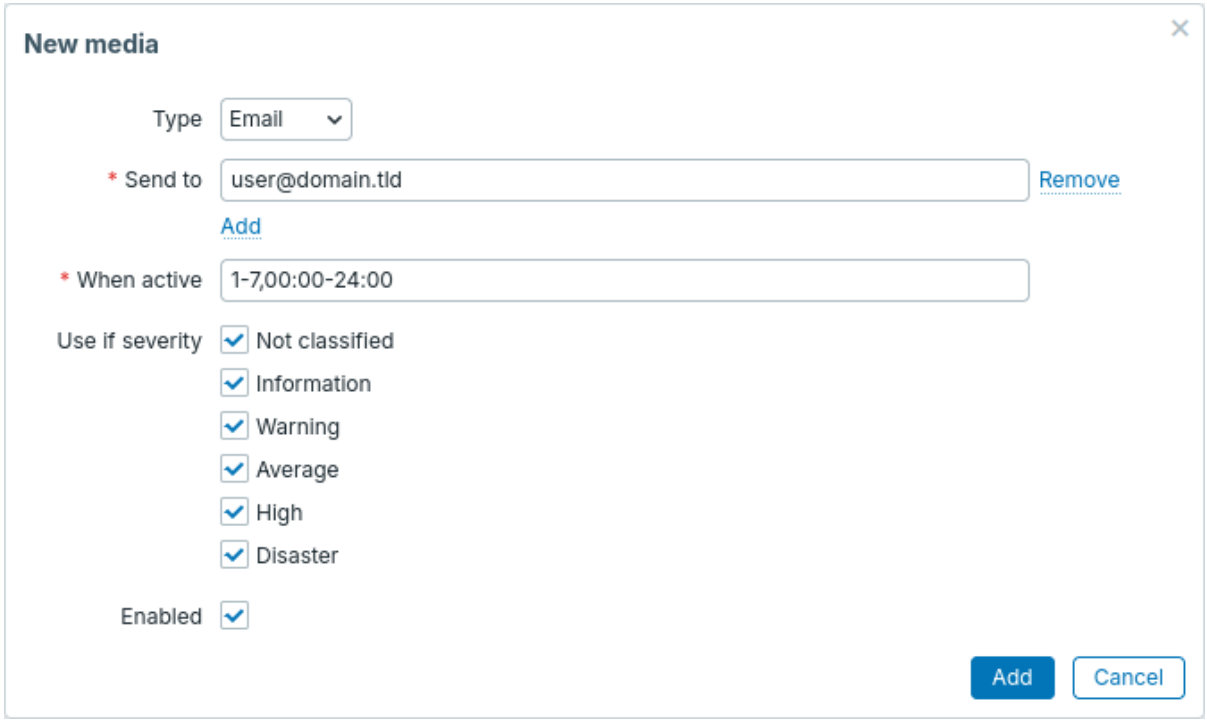
此操作将打开一个列表，其中包含从 Linux 服务器主机收集到的所有最新监控项。



设置问题告警

Zabbix 可以通过多种方式通知您基础设施中的问题。
本指南提供了发送电子邮件告警的配置步骤。

1. 转到 *User settings* > *Profile*，切换到 *Media* 选项卡，并
添加您的电子邮件。



2. 按照接收问题通知指南进行操作。

下次 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

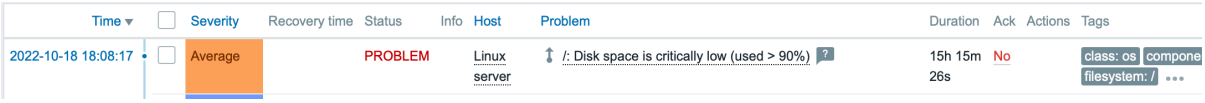
测试你的配置

在 Linux 上，你可以通过运行以下命令模拟高 CPU 负载，并因此收到问题告警：

```
cat /dev/urandom | md5sum
```

你可能需要运行多个 md5sum 进程，才能使 CPU 负载超过阈值。

当 Zabbix 检测到该问题时，它会显示在 *Monitoring* > *Problems* 部分中。



如果已配置告警，你还会收到问题通知。

另请参见：

- 创建监控项 - 如何开始监控额外指标（不使用模板的自定义监控）。
- Zabbix agent 监控项, Windows 上的 Zabbix agent 监控项 - 可使用 Windows 上的 Zabbix agent 监控的完整指标列表。
- 问题升级 - 如何创建多步骤告警场景（例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，再向数据中心经理发送消息）。
- 从软件包安装 - 如何使用适用于各种 Linux 发行版的官方 RPM 和 DEB 软件包安装 Zabbix 组件，以确保获得最新功能和错误修复。

使用 **Zabbix agent** 监控 **Windows**

简介

本页面将引导您完成使用 Zabbix 开始对 Windows 机器进行基本监控所需的步骤。

本指南适用对象

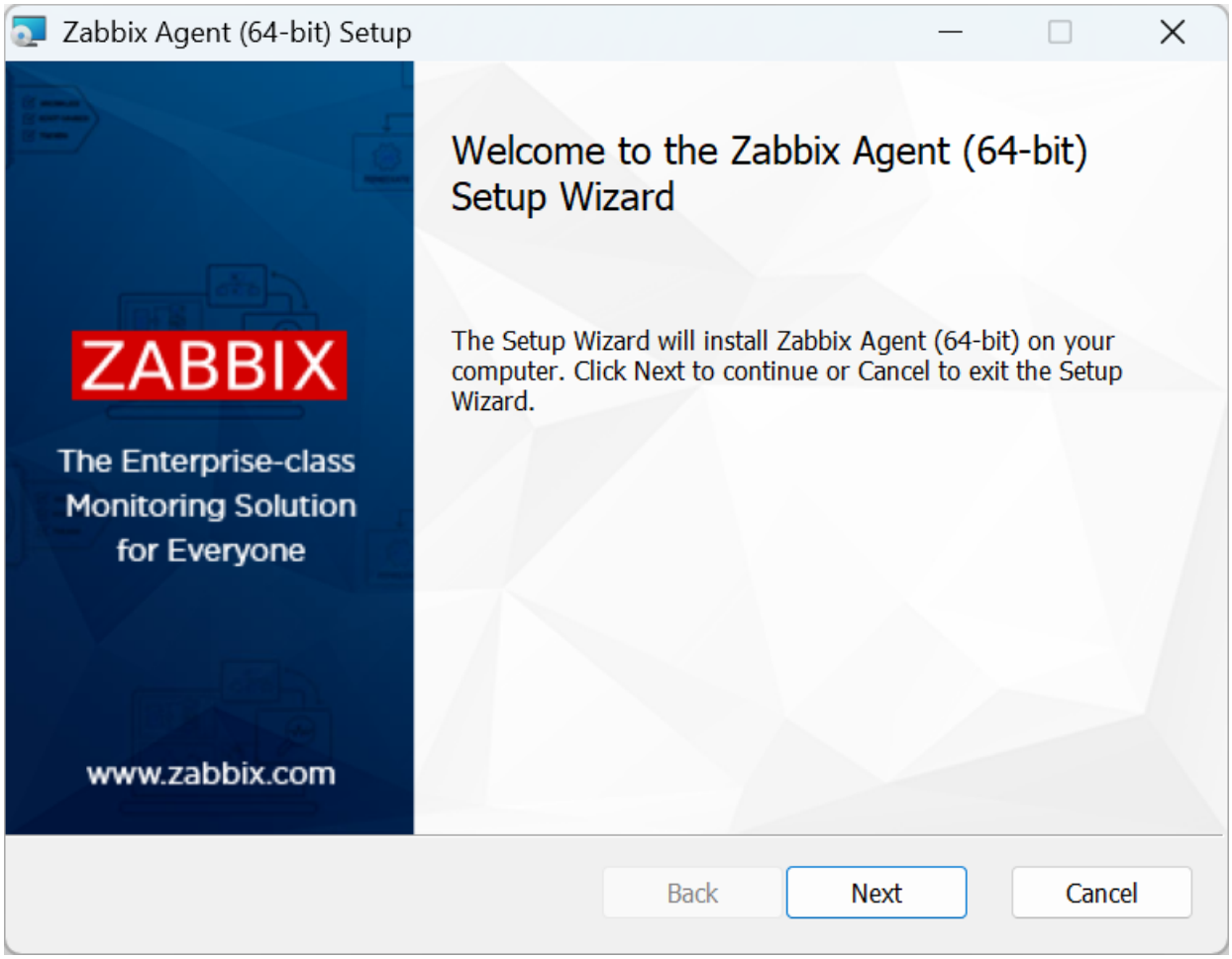
本指南面向 Zabbix 新用户，包含启用 Windows 机器基本监控所需的最少步骤。
如果您需要深入了解自定义选项或需要进行更高级的配置，请参阅 Zabbix 手册中的配置部分。

前提条件

在继续本安装指南之前，您必须根据操作系统的相关说明下载并安装 Zabbix 服务器和 Zabbix 前端。

安装 Zabbix agent

Zabbix agent 是负责收集数据的进程。您需要将其安装到要监控的 Windows 机器上。请按照Windows 的 Zabbix agent 安装说明进行操作。



配置 Zabbix 进行监控

Zabbix agent 可以以主动或被动模式（同时）收集指标。

Note:
被动检查是一种简单的数据请求。Zabbix 服务器或 proxy 会请求某些数据（例如 CPU 负载），然后 Zabbix agent 将结果发送回服务器。主动检查需要更复杂的处理。agent 必须先从服务器中检索一个监控项列表以进行独立处理，然后再批量将数据发送回去。有关更多信息，请参见被动和主动 agent 检查。

Zabbix 提供的监控模板通常有两种选择 - Zabbix agent 模板和 Zabbix agent（主动）模板。使用第一种选项时，agent 将以被动模式收集指标。这些模板会提供相同的监控结果，但使用不同的通信协议。

后续的 Zabbix 配置取决于您选择的是主动 还是被动 Zabbix agent 检查模板。

被动检查

Zabbix 前端

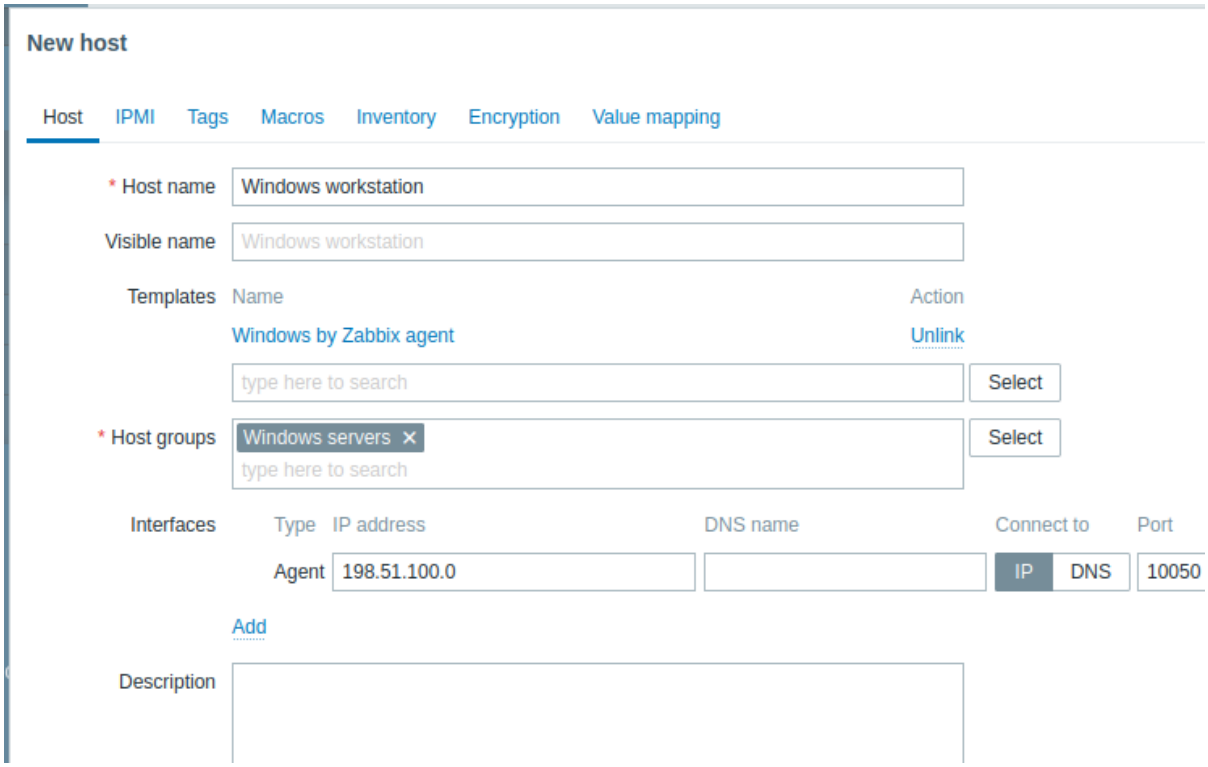
1. 登录 Zabbix 前端。

2. 在 Zabbix web interface 中创建主机。

此主机将代表您的 Windows 机器。

3. 在 *Interfaces* 参数中，添加 *Agent* 接口，并指定安装了 agent 的 Windows 机器的 IP 地址或 DNS 名称。

4. 在 *Templates* 参数中，输入或选择 *Windows by Zabbix agent*。



Zabbix agent

对于被动检查，Zabbix agent 需要知道 Zabbix 服务器的 IP 地址或 DNS 名称。如果您在安装 agent 期间提供了正确的信息，配置文件已经更新。否则，您需要手动指定该信息。打开agent 配置文件，并将 Zabbix 服务器的 IP/DNS 添加到 `Server` 参数中。

示例：

```
Server=192.0.2.22
```

主动检查

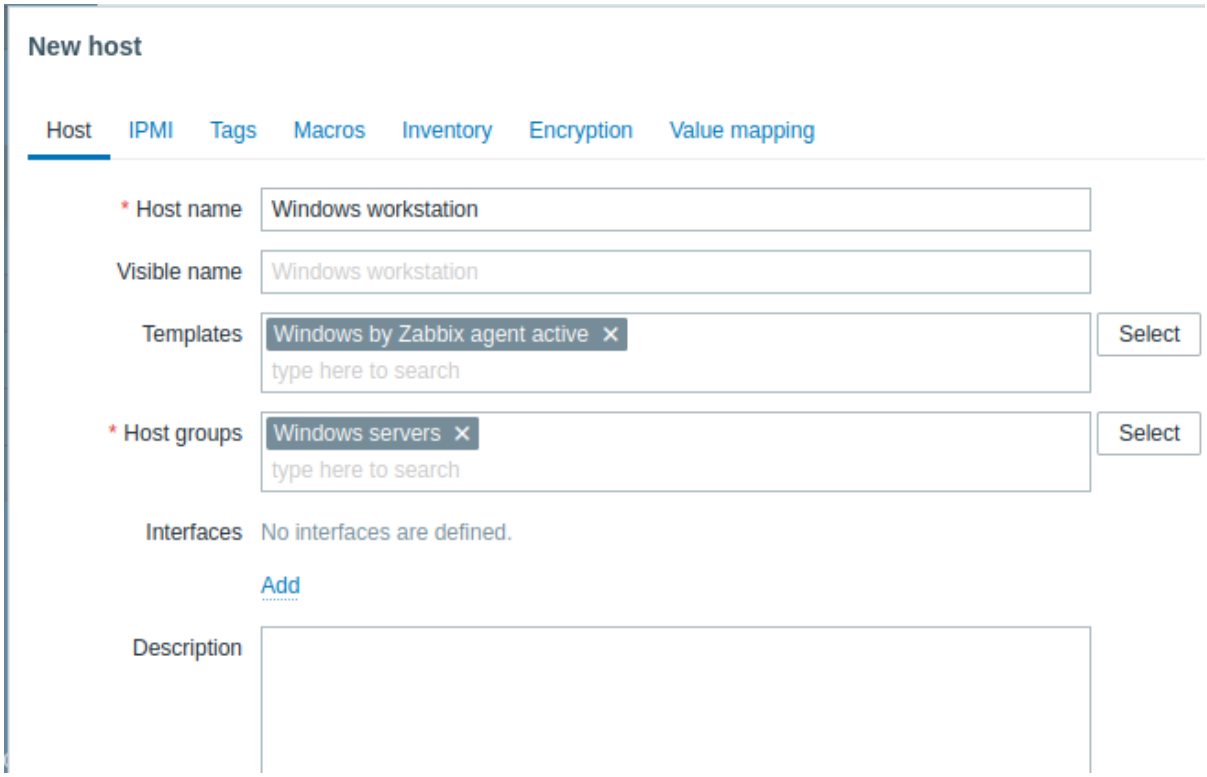
Zabbix 前端

1. 登录 Zabbix 前端。

2. 在 Zabbix web 界面中创建主机。

此主机将代表您的 Windows 机器。

3. 在 *Templates* 参数中，输入或选择 *Windows by Zabbix agent active*。



Zabbix agent

在 `C:\Program files\Zabbix Agent` 文件夹中打开 `zabbix_agentd.conf` 文件，并添加：

- 将你在 Zabbix 前端中创建的主机名称填写到 *Hostname* 参数中。
- 将你的 Zabbix 服务器机器的 IP 地址或 DNS 名称填写到 *ServerActive* 参数中（如果你在 Zabbix agent 安装过程中已提供，该值可能已预先填充）。

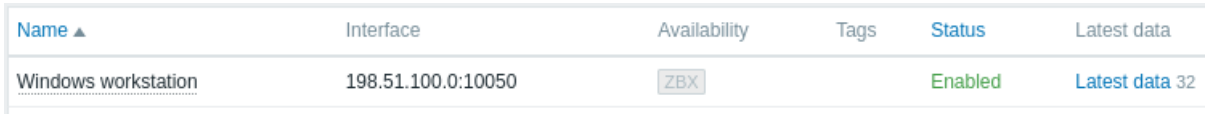
示例：

```
ServerActive= 192.0.2.22
Hostname=Windows workstation
```

查看已收集的监控指标

恭喜！此时，Zabbix 已经在监控你的 Windows 机器了。

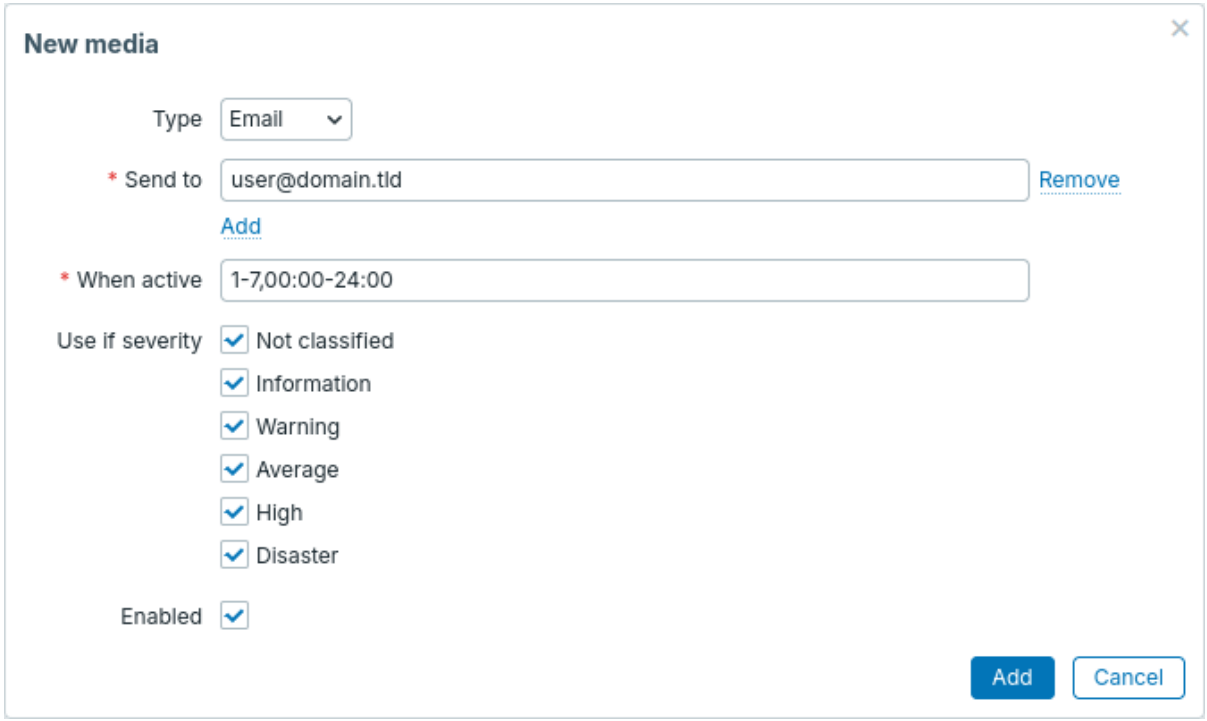
要查看已收集的监控指标，请打开 监控 > 主机菜单部分，然后单击主机旁边的 最新数据。



设置问题告警

Zabbix 可以通过多种方式通知您基础设施中的问题。本指南提供了发送电子邮件告警的配置步骤。

1. 转到 *User settings* > *Profile*，切换到 *Media* 选项卡，并添加您的电子邮件。



2. 按照接收问题通知中的指南进行操作。

下次当 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

Note:
在 Windows 上，您可以使用 CpuStres 工具来模拟高 CPU 负载，从而接收到问题告警。

另请参见：

- 创建监控项 - 如何开始监控其他指标（不使用模板的自定义监控）。
- Zabbix agent 监控项、Windows 上的 Zabbix agent 监控项 - 可使用 Windows 上的 Zabbix agent 监控的全部指标列表。
- 问题升级 - 如何创建多步骤告警场景（例如，先向系统管理员发送消息，然后如果问题在 45 分钟内仍未解决，再向数据中心经理发送消息）。

通过 **HTTP** 监控 **Apache**

介绍

本页面展示了一种快速而简单的方法，无需安装任何额外软件即可开始监控 Apache web 服务器。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用 Apache 安装基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或需要更高级的配置，请参阅 Zabbix 手册中的Configuration 部分。

先决条件

在继续本安装指南之前，您必须根据您的操作系统说明下载并安装 Zabbix 服务器和 Zabbix 前端。

准备 Apache

1. 检查您正在使用的 Apache 版本：

在基于 RHEL 的系统上，运行：

```
httpd -v
```

在 Debian/Ubuntu 上，运行：

```
apache2 -v
```

2. 确保您的 Apache 实例中已启用 Status module。

在基于 RHEL 的系统上，运行：

```
httpd -M | grep status
status_module (shared)
```

在 Debian/Ubuntu 上，运行：

```
apache2ctl -M | grep status
status_module (shared)
```

如果在列表中看不到 status_module，请通过运行以下命令启用该模块：

在基于 RHEL 的系统上，运行：

```
LoadModule status_module /usr/lib/apache2/modules/mod_status.so
```

在 Debian/Ubuntu 上，运行：

```
sudo /usr/sbin/a2enmod status
```

3. 编辑 Apache 配置文件，以允许来自 Zabbix 服务器 IP 的状态报告访问。

在基于 RHEL 的系统上：`/etc/httpd/conf.modules.d/status.conf`：

```
sudo vi /etc/httpd/conf.modules.d/status.conf
```

在 Debian/Ubuntu 上：`/etc/apache2/mods-enabled/status.conf`：

```
sudo vi /etc/apache2/mods-enabled/status.conf
```

将以下行添加到文件中（将 **198.51.100.255** 替换为您的 Zabbix 服务器 IP 地址）：

- 对于 Apache 2.2：
  <Location /server-status> SetHandler server-status

  Order Deny,Allow Deny from all Allow from 198.51.100.255 </Location>
- 对于 Apache 2.4：
  <Location ”/server-status”> SetHandler server-status Require ip 198.51.100.255 </Location>

4. 重启 Apache

在基于 RHEL 的系统上，运行：

```
sudo systemctl restart httpd
```

在 Debian/Ubuntu 上，运行：

```
sudo systemctl restart apache2
```

5. 要检查是否一切配置正确，请运行（将 **198.51.100.255** 替换为您的 Zabbix 服务器 IP 地址）：

```
curl 198.51.100.255/server-status
```

响应应包含 Apache Web 服务器统计信息。

配置 Zabbix 以进行监控

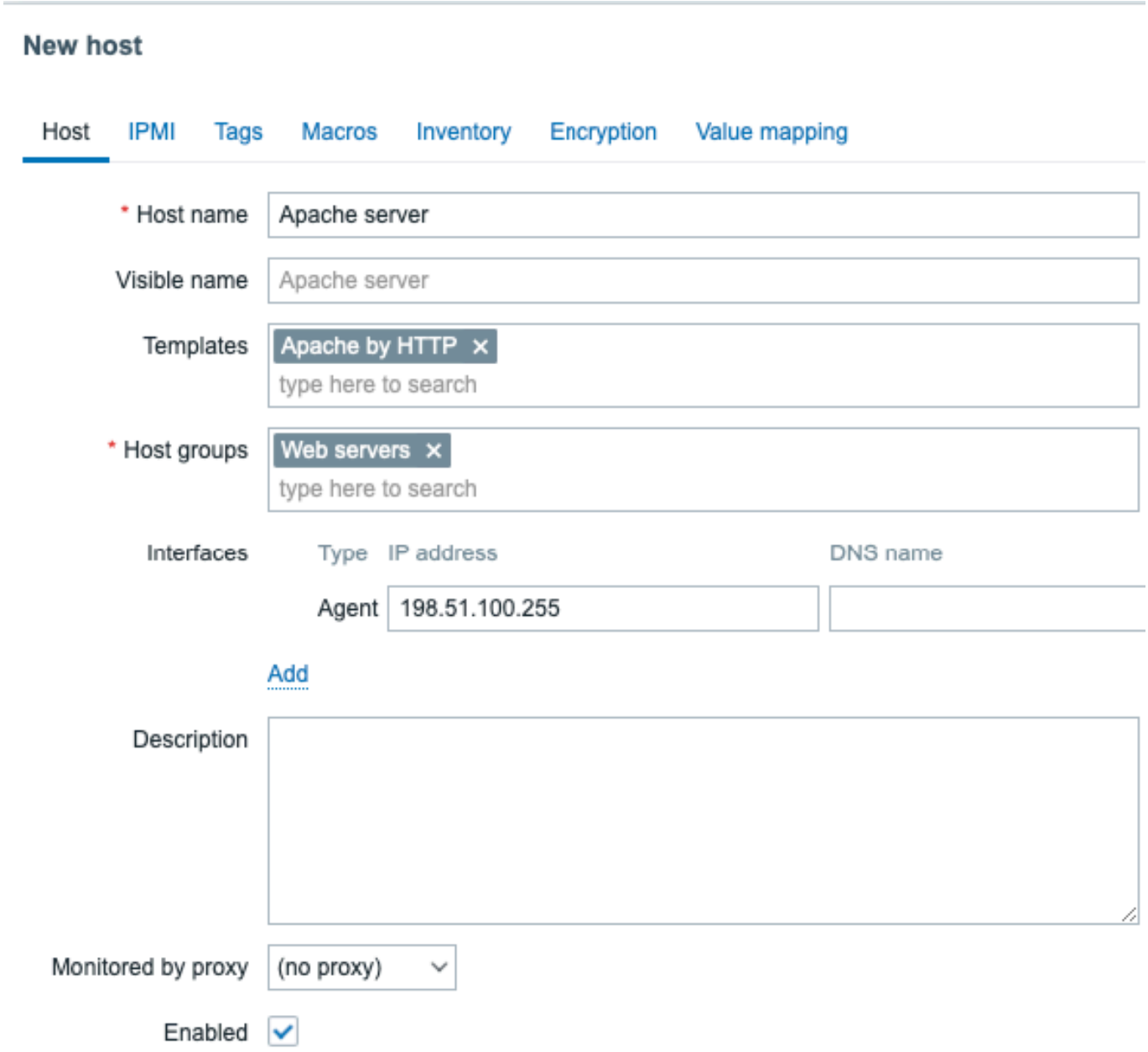
1. 登录 Zabbix 前端。

2. 在 Zabbix web 前端中创建主机。

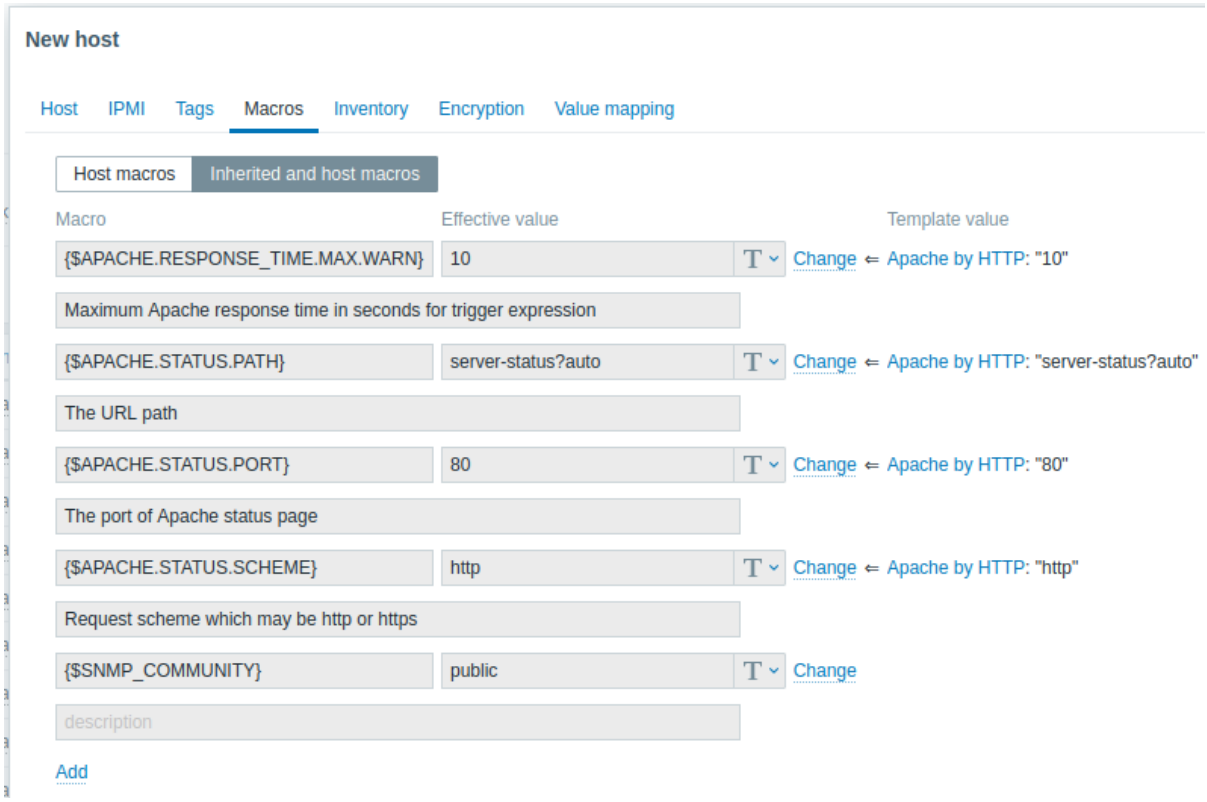
此主机将代表您的 Apache 服务器。

3. 在 *Interfaces* 参数中，添加 *Agent* 接口并指定您的 Apache 实例 IP 地址。
您无需在该机器上安装 **Zabbix agent**，该接口仅用于解析 {HOST.CONN} 宏。
该宏用于模板监控项中，以定位 Apache 实例。

4. 在 *Templates* 参数中，输入或选择 *Apache by HTTP*。



5. 切换到 **Macros** 选项卡，并选择 *Inherited and host macros* 模式。检查宏 {$APACHE.STATUS.PORT} 和 {$APACHE.STATUS.SCHEME} 的值是否适合您的安装设置。
默认情况下，端口为 80，scheme 为 http。如果您使用不同的端口和/或 scheme，请更改宏值。



查看已收集的监控项

恭喜！此时，Zabbix 已经在监控你的 Apache web 服务器。

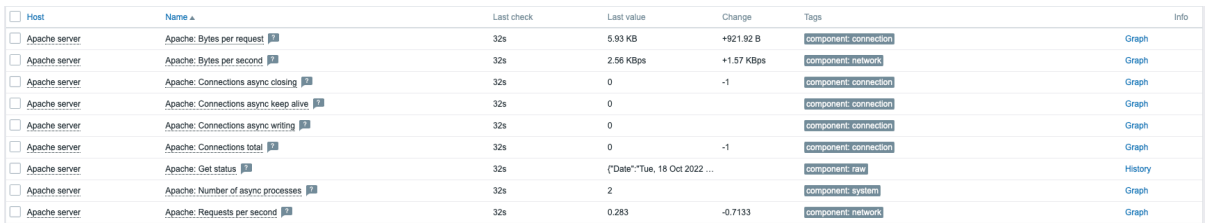
要查看已收集的监控项，请打开 *Monitoring* > *Hosts* 菜单部分，然后单击主机旁边的 *Dashboards*。



此操作将带你进入主机仪表板，其中包含从 Apache /server-status 页面收集到的最重要监控项。



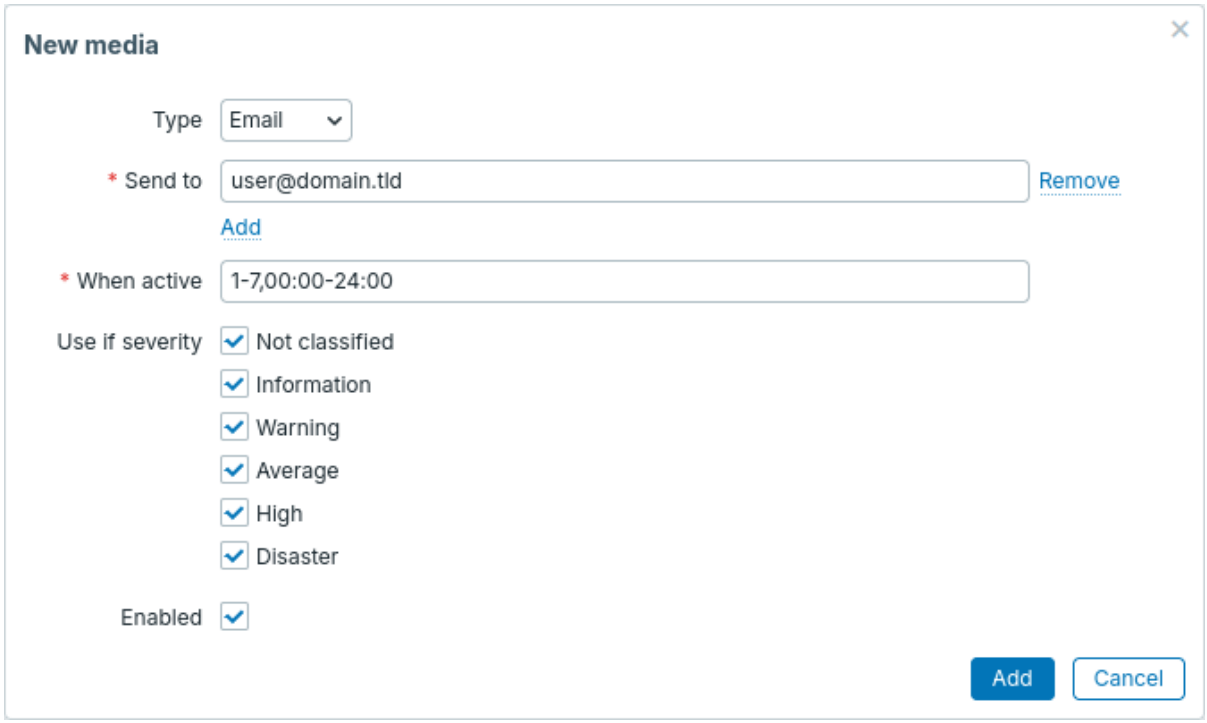
另外，在 *Monitoring* > *Hosts* 中，你也可以单击 *Latest data*，以列表形式查看所有最新收集的监控项。



设置问题告警

Zabbix 可以使用多种方式通知您基础设施中的问题。
本指南提供了发送电子邮件告警的配置步骤。

1. 转到 *User settings* > *Profile*，切换到 *Media* 选项卡，并添加您的电子邮件。



2. 按照接收问题通知指南进行操作。

下次当 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

测试您的配置

要模拟真实问题并接收测试问题告警：

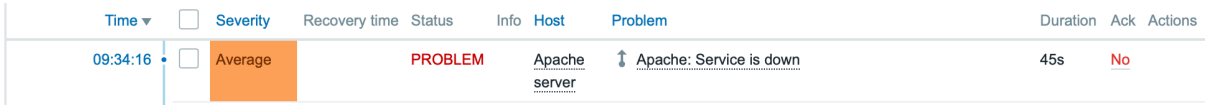
1. 在 Zabbix 中打开 *Apache server* 主机配置。

2. 切换到 Macros 选项卡，并选择 *Inherited and host macros*。

3. 点击 {$APACHE.STATUS.PORT} 宏旁边的 *Change*，并设置为不同的端口。

4. 点击 *Update* 保存主机配置。

5. 几分钟后，Zabbix 将检测到 *Apache service is down* 问题，因为此时它无法连接到该实例。
它将显示在 *Monitoring* > *Problems* 部分。



如果已配置告警，您还会收到问题通知。

6. 将宏值改回原值以解决问题，并继续监控 Apache。

另请参阅：

- Web 服务器加固 - 用于提高 Web 服务器安全性的推荐设置。
- 创建监控项 - 如何开始监控其他指标。
- HTTP 监控项 - 如何使用 HTTP agent 监控自定义指标。
- 问题升级 - 如何创建多步骤告警场景
  （例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，再向数据中心经理发送消息）。

使用 **Zabbix agent 2** 监控 **MySQL**

简介

本页将引导您完成开始对 MySQL 服务器进行基础监控所需的步骤。

要监控 MySQL 服务器，有几种方法：Zabbix agent、Zabbix agent 2，或 Open Database Connectivity (ODBC) 标准。本指南主要介绍使用 Zabbix agent 2 监控 MySQL 服务器，这是由于其在各种环境中都能实现无缝配置，因此是推荐的方法。不过，本页也提供了其他方法的说明，因此您可以根据自身需求选择最适合的一种。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用 MySQL 服务器基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或需要更高级的配置，请参阅 Zabbix 手册中的Configuration 部分。

前提条件

在继续阅读本指南之前，您需要根据您的操作系统说明下载并安装 Zabbix 服务器、Zabbix 前端和 Zabbix agent 2。

根据您的环境，本指南中的某些步骤可能会略有不同。本指南基于运行 Ubuntu 的环境。

创建 MySQL 用户

要监控 MySQL 服务器，Zabbix 需要能够访问它及其进程。您的 MySQL 安装中已经有一个具备所需访问级别的用户（在安装 Zabbix 时创建的用户"zabbix"），不过，对于简单监控来说，这个用户拥有的权限比必要的更多（例如删除数据库、从表中删除条目等权限）。因此，需要创建一个仅用于监控 MySQL 服务器的 MySQL 用户。

1. 连接到 MySQL 客户端，创建一个"zbx_monitor" 用户（将"zbx_monitor" 用户的 *<password>* 替换为您选择的密码），并向该用户 GRANT 必要的权限：

```
mysql -u root -p
# Enter password:

mysql> CREATE USER 'zbx_monitor'@'%' IDENTIFIED BY '<password>';
mysql> GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';
mysql> quit;
```

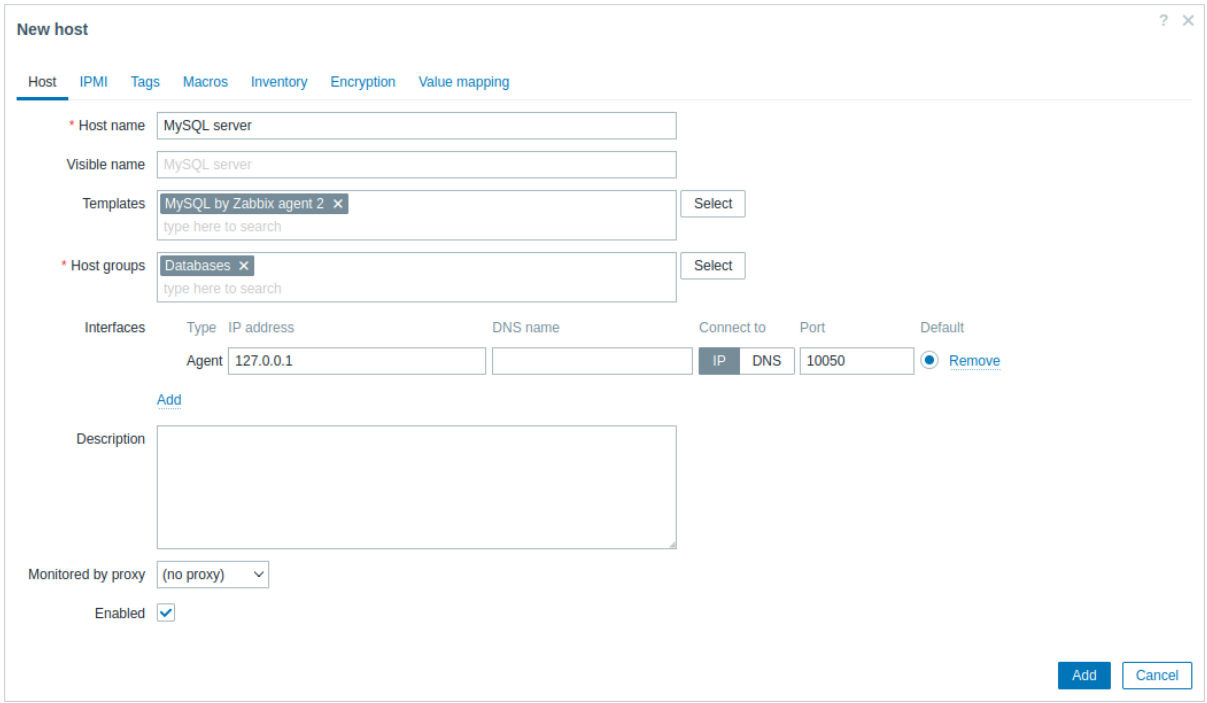
创建用户后，您可以继续下一步。

配置 Zabbix 前端

1. 登录 Zabbix 前端。

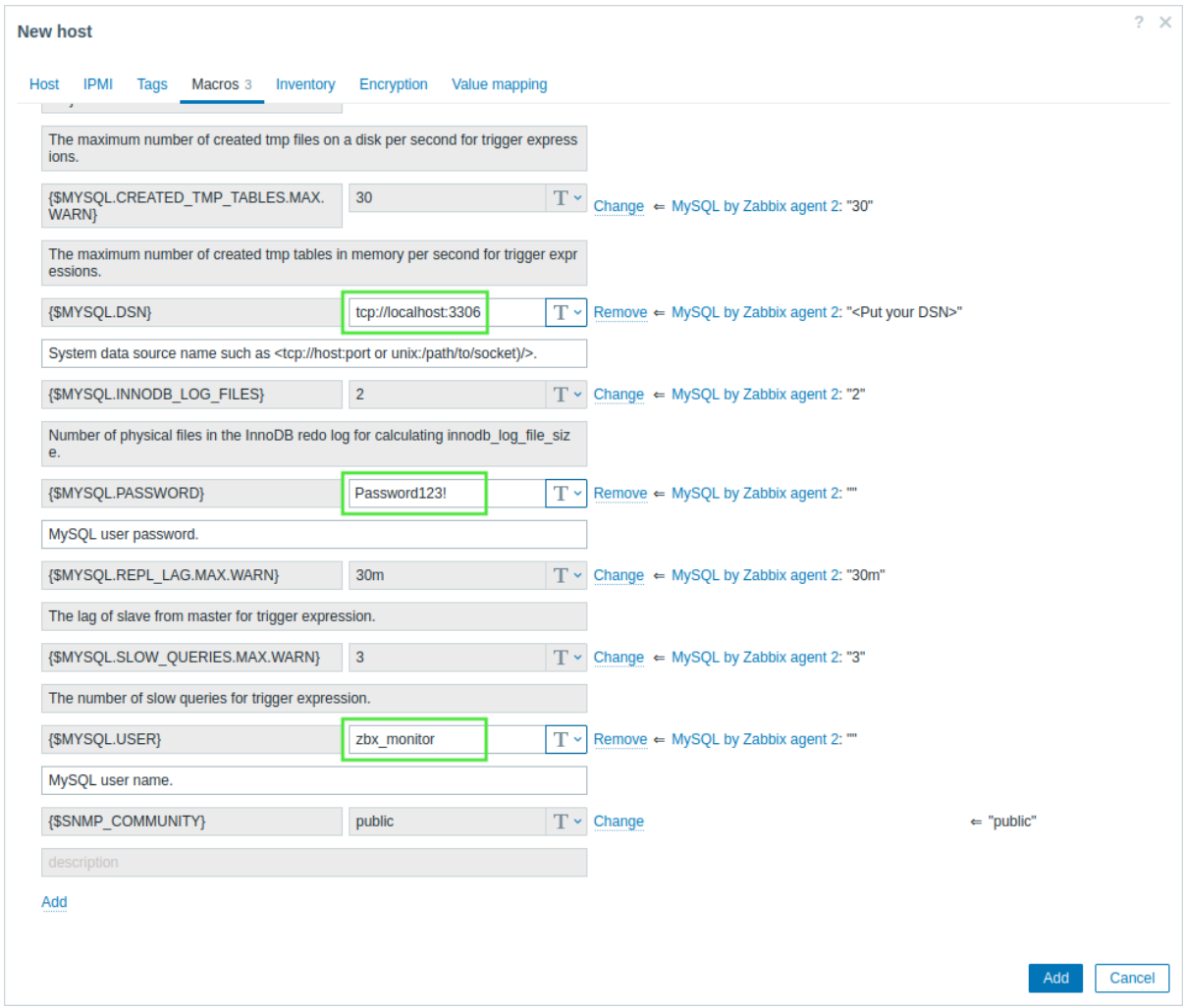
2. 在 Zabbix web 界面中创建主机：

- 在 *Host name* 字段中，输入主机名称（例如，"MySQL server"）。
- 在 *Templates* 字段中，输入或选择将要链接到主机的模板 "MySQL by Zabbix agent 2"。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，"Databases"）。
- 在 *Interfaces* 字段中，添加一个类型为 "Agent" 的接口，并指定你的 MySQL server IP 地址。本指南使用 "127.0.0.1"（localhost）来监控安装在与 Zabbix 服务器和 Zabbix agent 2 相同机器上的 MySQL server。



- 在 *Macros* 选项卡中，切换到 *Inherited and host macros*，查找以下宏，并点击宏值旁边的 *Change* 进行更新：
  - {$MYSQL.DSN} - 设置 MySQL server 的数据源（来自 MySQL Zabbix agent 2 插件配置文件中命名会话的连接字符串）。本指南使用默认数据源 "tcp://localhost:3306" 来监控安装在与 Zabbix 服务器和 Zabbix agent 2 相同机器上的 MySQL server。

- {$MYSQL.PASSWORD} - 设置之前创建的 MySQL 用户“zbx_monitor” 的密码。
- {$MYSQL.USER} - 设置之前创建的 MySQL 用户“zbx_monitor” 的用户名。

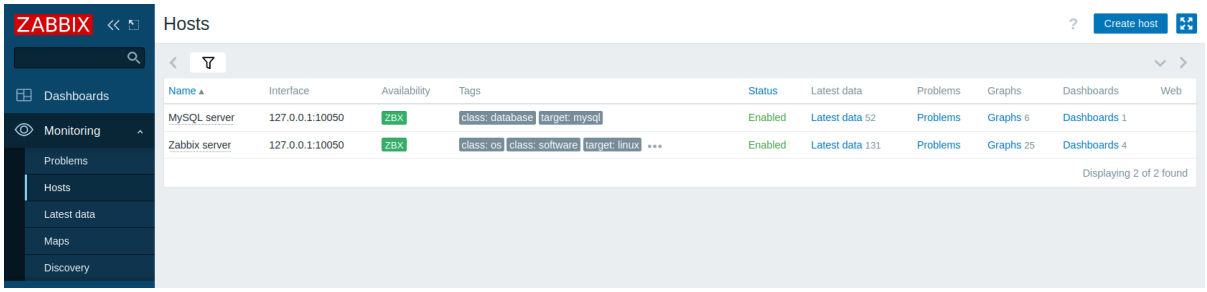


3. 点击 *Add* 添加主机。此主机将代表你的 MySQL server。

查看已收集的监控指标

恭喜！此时，Zabbix 已经在监控你的 MySQL 服务器了。

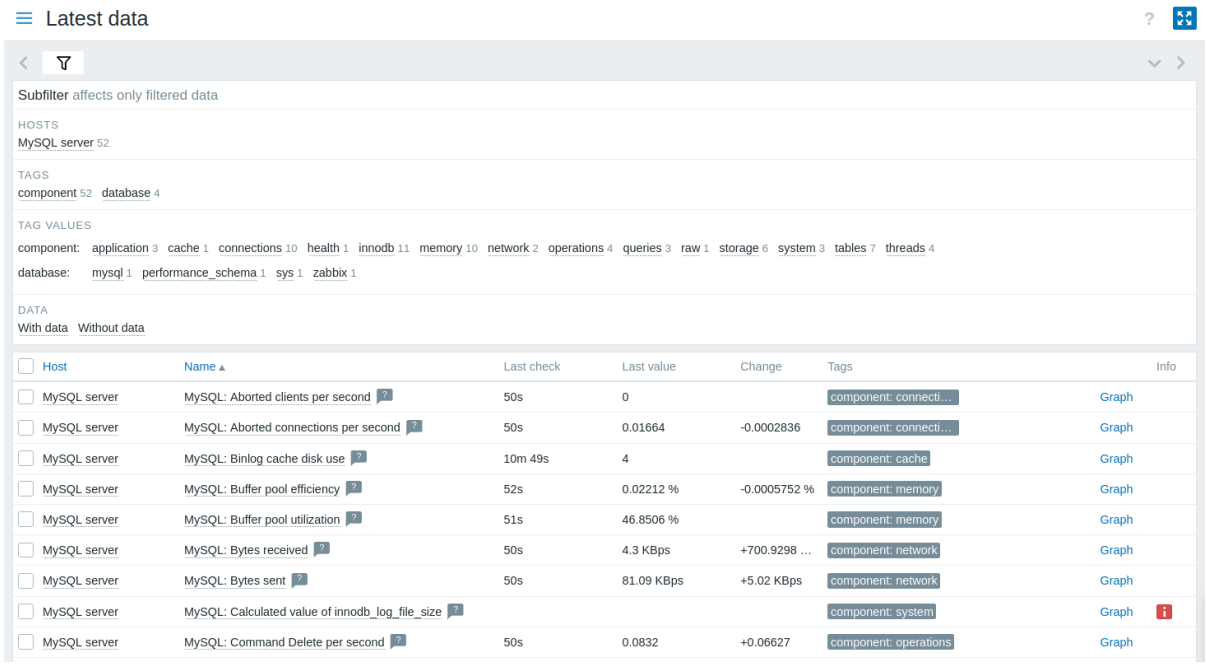
要查看已收集的监控指标，请转到监控 > 主机 菜单部分，并点击主机旁边的 仪表板。



此操作将带你进入主机仪表板（在模板级别配置），其中包含从 MySQL 服务器收集到的最重要监控指标。



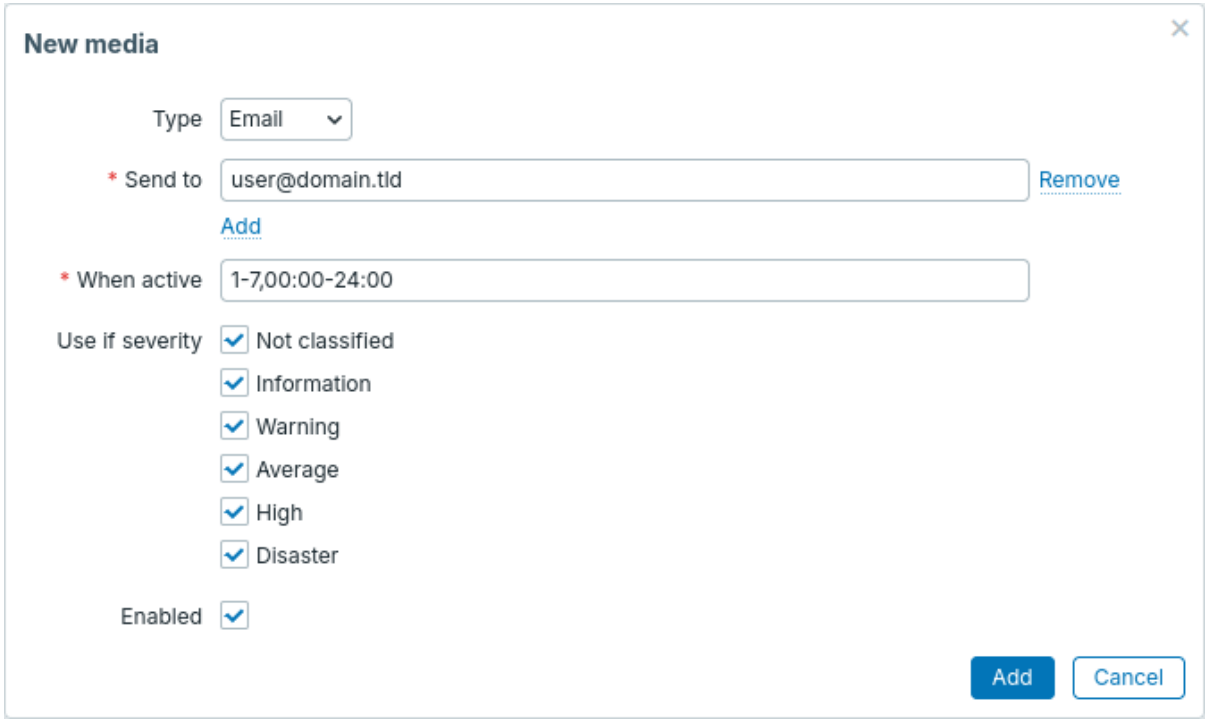
另外，你也可以在监控 > 主机 菜单部分点击最新数据，以列表形式查看所有最新收集的监控指标。请注意，监控项 *MySQL: Calculated value of innodb_log_file_size* 预期不会有数据，因为该值将根据过去一小时内的数据计算得出。



设置问题告警

Zabbix 可以通过多种方式通知您基础设施中的问题。本指南提供了发送电子邮件告警的基本配置步骤。

1. 导航到用户设置 > 个人资料，切换到 媒体选项卡，并添加您的电子邮件。



2. 按照接收问题通知指南进行操作。

下次当 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

测试你的配置

要测试你的配置，我们可以通过在 Zabbix 前端中更新主机配置来模拟一个真实问题。

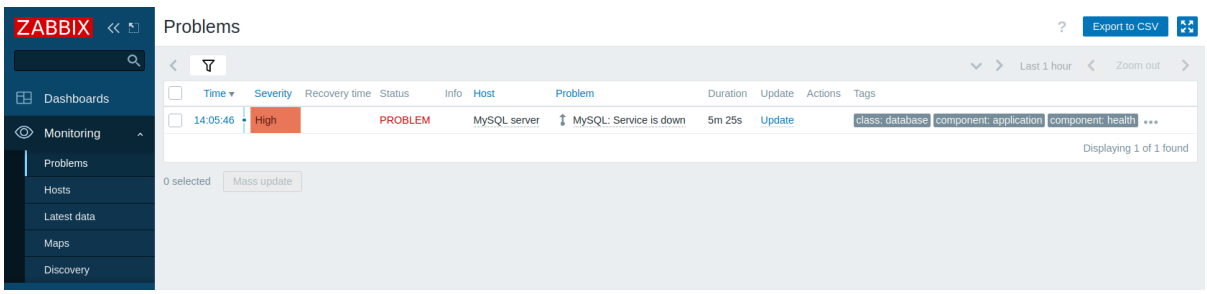
1. 在 Zabbix 中打开你的 MySQL 服务器主机配置。

2. 切换到 *Macros* 选项卡，并选择 *Inherited and host macros*。

3. 点击例如之前配置的 {$MYSQL.USER} 宏值旁边的 *Change*，并设置一个不同的 MySQL 用户名。

4. 点击 *Update* 以更新主机配置。

5. 过一会儿，Zabbix 将检测到”MySQL: Service is down” 问题，因为它将无法连接到 MySQL 服务器。该问题将显示在*Monitoring > Problems* 中。



如果已配置 告警，你还会收到问题通知。

6. 将宏值改回之前的值，以解决该问题并继续监控 MySQL 服务器。

监控 MySQL 的其他方法

除了使用 Zabbix agent 2 监控 MySQL 服务器外，您还可以使用 Zabbix agent 或开放数据库连接（ODBC）标准。虽然推荐使用 Zabbix agent 2，但某些环境可能不支持 Zabbix agent 2，或者需要采用自定义方法。

Zabbix agent 和 ODBC 的关键区别在于数据采集方式 - Zabbix agent 直接安装在 MySQL 服务器上，并使用其内置功能收集数据，而 ODBC 则依赖 ODBC 驱动程序与 MySQL 服务器建立连接，并通过 SQL 查询检索数据。

尽管许多配置步骤与使用 Zabbix agent 2 监控 MySQL 服务器相似，但仍有一些重要差异 - 您需要配置 Zabbix agent 或 ODBC，才能监控 MySQL 服务器。以下说明将引导您了解这些差异。

使用 Zabbix agent 监控 MySQL

要使用 Zabbix agent 监控 MySQL 服务器，您需要根据操作系统的说明下载并安装 Zabbix 服务器、Zabbix 前端和 Zabbix agent。

在成功安装所需的 Zabbix 组件后，您需要按照创建 *MySQL* 用户 部分中的说明创建一个 MySQL 用户。

创建 MySQL 用户后，您需要配置 Zabbix agent，使其能够与 MySQL 服务器建立连接并对其进行监控。这包括为执行自定义 agent 检查配置多个用户参数，以及向 Zabbix agent 提供连接到 MySQL 服务器所需的凭据，使用的是之前创建的 “zbx_monitor” 用户。

配置 **Zabbix agent**

1. 导航到 Zabbix agent 附加配置目录。

```
cd /etc/zabbix/zabbix_agentd.d
```

> **Attention:**
> Zabbix agent 附加配置目录应与 Zabbix agent 配置文件位于同一目录中；默认路径取决于操作系统和 Zabbix 的安装方式。

无需在 Zabbix agent 配置文件中定义监控 MySQL 服务器所需的所有用户参数，这些参数将在附加配置目录中的单独文件内定义。

2. 在 Zabbix agent 附加配置目录中创建 `template_db_mysql.conf` 文件。

```
vi template_db_mysql.conf
```

3. 将 `template_db_mysql.conf` 文件（位于 Zabbix 仓库中）的内容复制到所创建的 `template_db_mysql.conf` 文件中，然后保存。

4. 重启 Zabbix agent 以更新其配置。

```
systemctl restart zabbix-agent
```

配置好 Zabbix agent 用户参数后，即可继续配置允许 Zabbix agent 访问 MySQL 服务器的凭据。

5. 导航到 Zabbix agent 主目录（如果系统中不存在该目录，则需要创建它；默认值：*/var/lib/zabbix*）。

```
cd /var/lib/zabbix
```

6. 在 Zabbix agent 主目录中创建一个 *.my.cnf* 文件。

```
vi .my.cnf
```

7. 将以下内容复制到 *.my.cnf* 文件中（将 *<password>* 替换为”zbx_monitor” 用户的密码）。

```
[client]
user='zbx_monitor'
password='<password>'
```

配置 **Zabbix** 前端并测试您的配置

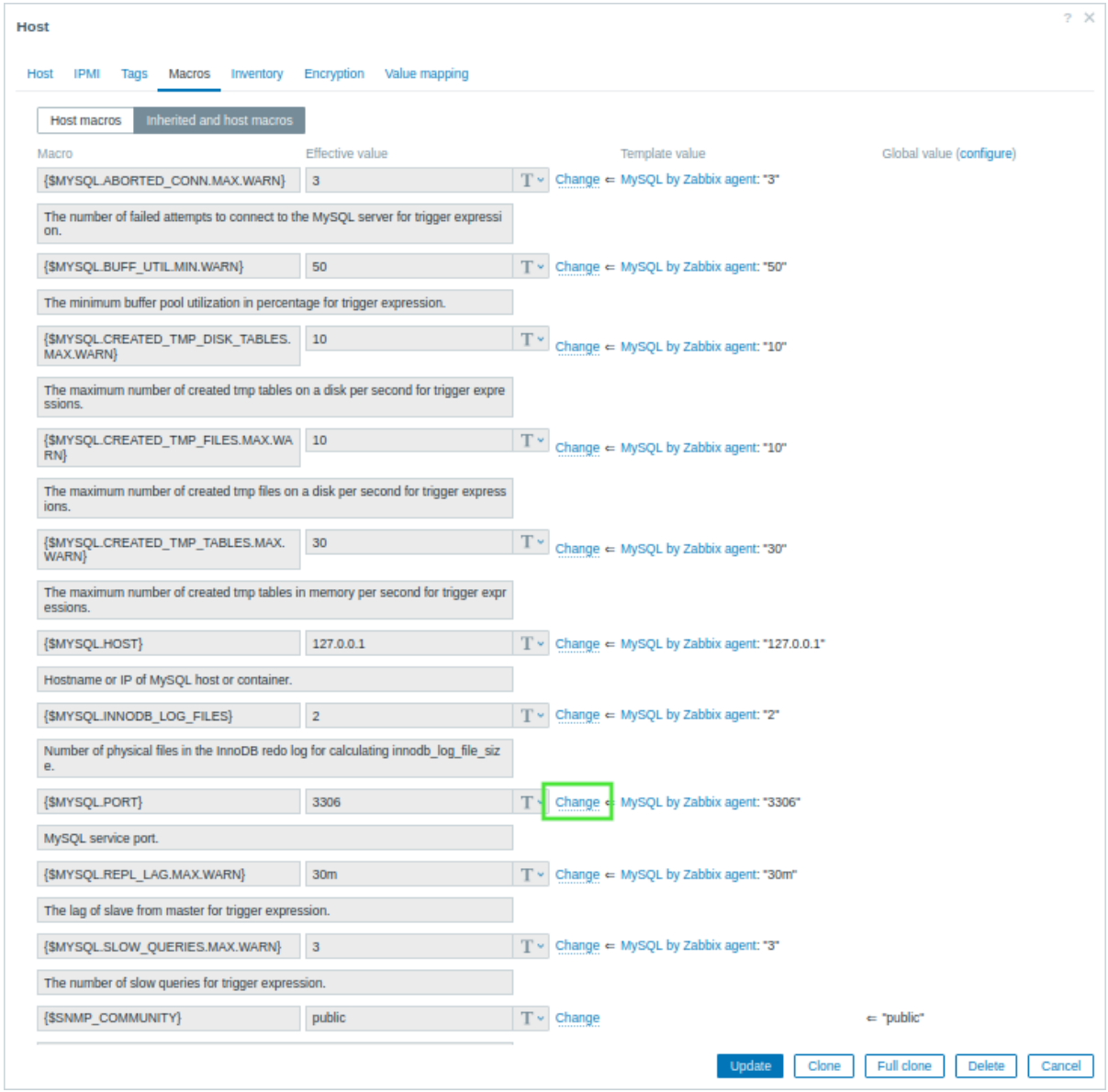
要配置 Zabbix 前端，请按照配置 *Zabbix* 前端 部分中的说明进行操作，并做以下调整：

- 在 *Templates* 字段中，输入或选择将要链接 到主机的模板”MySQL by Zabbix agent”。
- 无需配置 *Macros*。

完成 Zabbix 前端配置后，您可以查看收集到的指标 并设置问题告警。

要测试您的配置，请按照测试您的配置 部分中的说明进行操作，并做以下调整：

- 在 MySQL 服务器主机配置的 *Inherited and host macros* 部分，点击 {$MYSQL.PORT} 宏值旁边的 *Change*，并设置一个不同的端口（例如”6033”）。



使用 ODBC 监控 MySQL

要通过 ODBC 监控 MySQL 服务器，您需要下载并安装 Zabbix 服务器和 Zabbix 前端。

在成功安装所需的 Zabbix 组件后，您需要按照创建 *MySQL* 用户 部分中的说明创建一个 MySQL 用户。

创建 MySQL 用户后，您需要设置 ODBC。这包括安装最常用的开源 ODBC API 实现之一 - unixODBC - 以及 unixODBC 驱动程序，并编辑 ODBC 驱动程序配置文件。

配置 **ODBC**

1. 安装 unixODBC。推荐使用 Linux 操作系统默认的软件包仓库来安装 unixODBC。

```
apt install unixodbc
```

2. 安装 MariaDB unixODBC 数据库驱动。尽管您使用的是 MySQL 数据库，但出于兼容性原因，这里使用 MariaDB unixODBC 驱动。

```
apt install odbc-mariadb
```

3. 检查 ODBC 配置文件 *odbcinst.ini* 和 *odbc.ini* 的位置。

```
odbcinst -j
```

执行此命令后的结果应类似于以下内容。

```
unixODBC 2.3.9
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
...
```

4. 要为监控 MySQL 数据库配置 ODBC 驱动，您需要驱动名称，该名称位于 *odbcinst.ini* 文件中。在下面的 *odbcinst.ini* 文件示例中，驱动名称为"MariaDB Unicode"。

```
[MariaDB Unicode]
Driver=libmaodbc.so
Description=MariaDB Connector/ODBC(Unicode)
Threading=0
UsageCount=1
```

5. 将以下内容复制到 *odbc.ini* 文件中（将 *<password>* 替换为"zbx_monitor" 用户的密码）。本指南使用"127.0.0.1"（localhost）作为 MySQL 服务器地址，用于监控安装在与 ODBC 驱动相同机器上的 MySQL 服务器。请注意数据源名称（DSN）"test"，在配置 Zabbix 前端时将会用到它。

```
[test]
Driver=MariaDB Unicode
Server=127.0.0.1
User=zbx_monitor
Password=<password>
Port=3306
Database=zabbix
```

配置 **Zabbix** 前端并测试您的配置

要配置 Zabbix 前端，请按照配置 *Zabbix* 前端 部分中的说明进行操作，并做以下调整：

- 在 *Templates* 字段中，输入或选择将要链接到主机的模板"MySQL by ODBC"。
- 无需配置 *Interfaces*。
- 在 MySQL 服务器主机配置的 *Inherited and host macros* 部分中，{$MYSQL.DSN} 宏值应设置为 *odbc.ini* 文件中的 DSN 名称。

完成 Zabbix 前端配置后，您可以查看收集到的指标、设置问题告警并测试您的配置。

另请参阅

- 创建监控项 - 如何开始监控其他指标。
- 问题升级 - 如何创建多步骤告警场景（例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，再向数据中心经理发送消息）。
- ODBC 监控 - 如何在其他 Linux 发行版上设置 ODBC，以及如何使用 ODBC 开始监控其他与数据库相关的指标。
- 模板 *MySQL by Zabbix agent* - 关于 *MySQL by Zabbix agent* 模板的更多信息。
- 模板 *MySQL by Zabbix agent 2* - 关于 *MySQL by Zabbix agent 2* 模板的更多信息。
- 模板 *MySQL by ODBC* - 关于 *MySQL by ODBC* 模板的更多信息。

使用 **Zabbix** 监控 **VMware**

介绍

本页将引导您完成开始对 VMware 进行基本监控所需的步骤。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用 VMware 基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或需要更高级的配置，请参阅 Zabbix 手册中的Virtual machine monitoring 部分或Configuration 部分。

前提条件

在继续阅读本指南之前，您需要根据您的操作系统说明下载并安装 Zabbix 服务器和 Zabbix 前端。

根据您的环境不同，本指南中的某些步骤可能会略有差异。本指南基于运行 Ubuntu 的环境。

默认假定 VMware 已经完成配置。本指南不涉及 VMware 的配置。

配置 Zabbix 服务器

要监控 VMware，必须启用 Zabbix 的 *vmware collector* 进程。有关 VMware 监控执行方式的更多信息，请参阅虚拟机监控。

1. 打开 Zabbix 服务器配置文件。

```
vi /etc/zabbix/zabbix_server.conf
```

2. 在 Zabbix 服务器配置文件中找到并将StartVMwareCollectors 参数设置为 2或更高（默认值为 0）。

```
###### Option: StartVMwareCollectors
#####       Number of pre-forked vmware collector instances.
#####
```

```
##### Mandatory: no
##### Range: 0-250
##### Default:
##### StartVMwareCollectors=0

StartVMwareCollectors=2
```

3. 重启 Zabbix 服务器。

```
systemctl restart zabbix-server
```

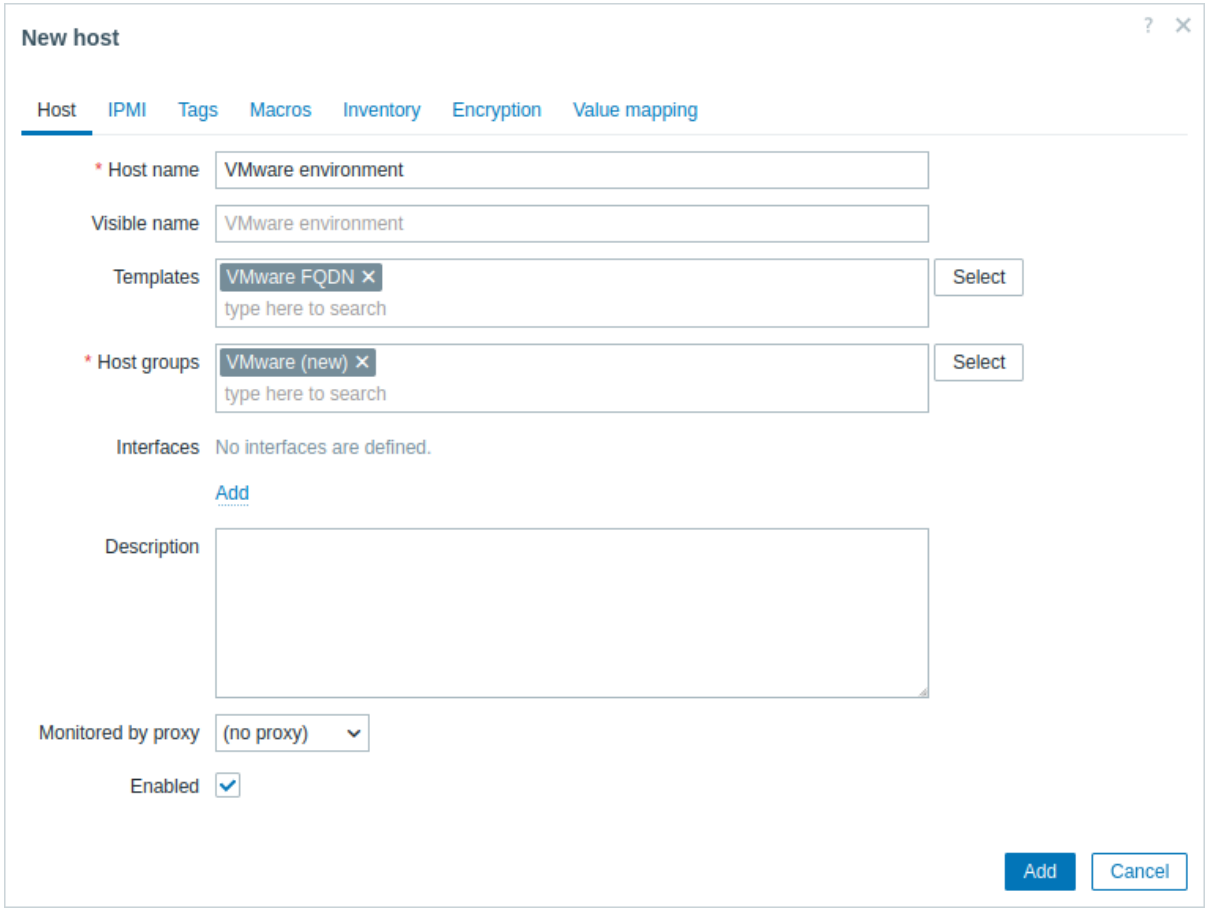
启动 *vmware collector* 进程后，继续执行下一步。

配置 Zabbix 前端

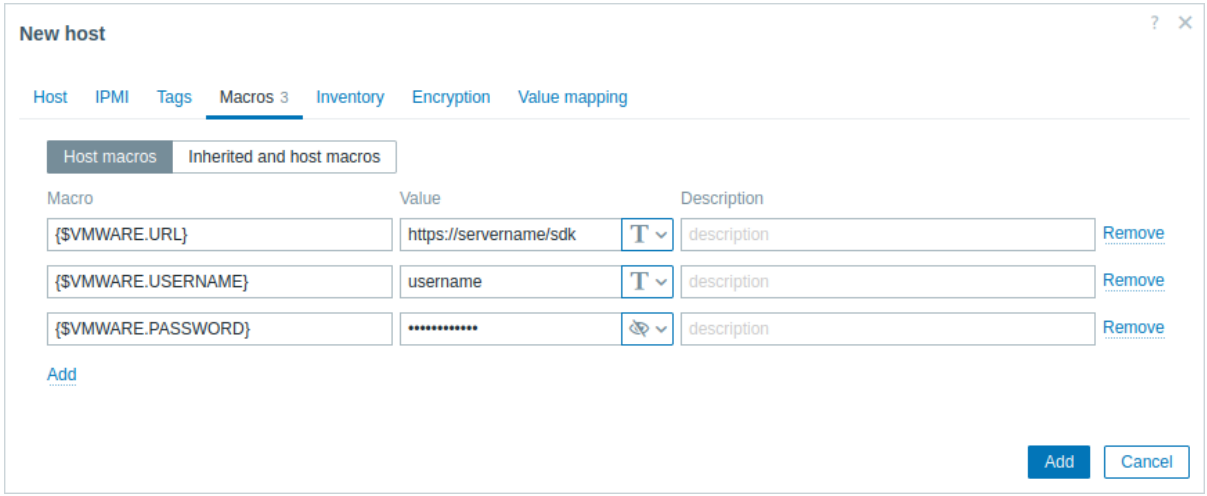
1. 登录 Zabbix 前端。

2. 在 Zabbix web 界面中创建主机：

- 在 *Host name* 字段中，输入主机名称（例如，“VMware environment”）。
- 在 *Templates* 字段中，输入或选择 “VMware FQDN”（或 “VMware”）模板。有关这些模板的更多信息，请参见虚拟机监控。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，一个新的主机组 “VMware”）。



- 在 *Macros* 选项卡中，设置以下主机宏：
  - {$VMWARE.URL} - VMware 服务（vCenter 或 ESXi hypervisor）SDK URL (https://servername/sdk)
  - {$VMWARE.USERNAME} - VMware 服务用户名
  - {$VMWARE.PASSWORD} - VMware 服务 {$VMWARE.USERNAME} 用户密码



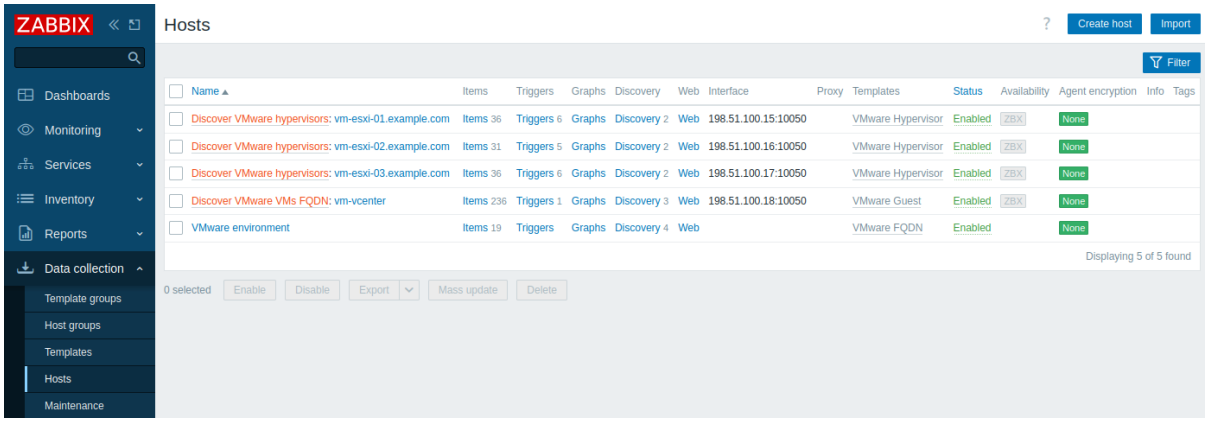
3. 点击 *Add* 按钮创建主机。此主机将代表您的 VMware 环境。

查看已收集的监控指标

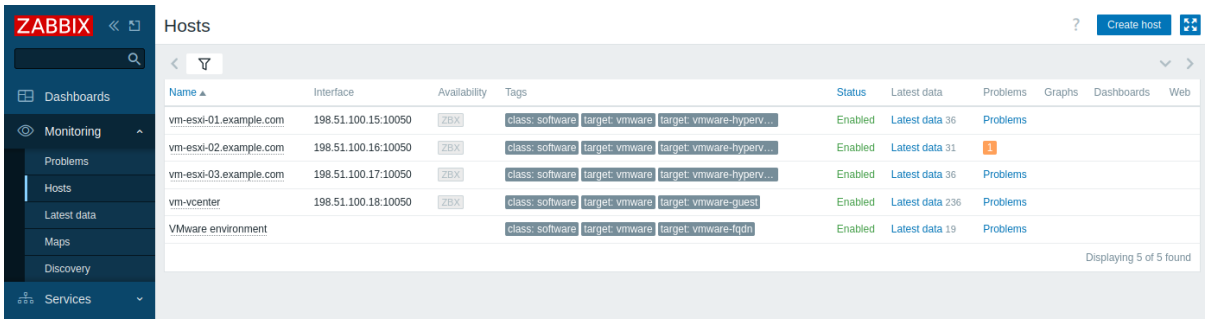
恭喜！此时，Zabbix 已经在监控你的 VMware 环境了。

根据 VMware 环境的配置，Zabbix 可能会发现并为发现的实体创建主机。请注意，如有必要，也可以手动执行主机的发现和创建。

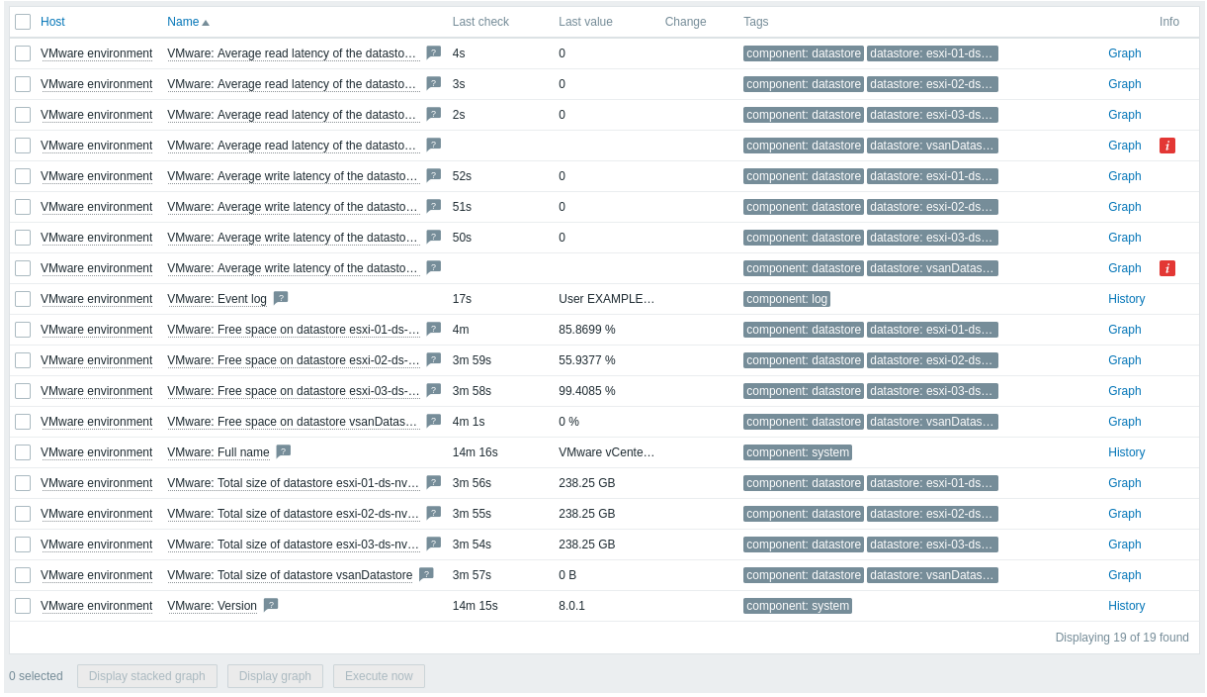
要查看已创建的主机，请导航到数据采集 > 主机 菜单部分。



要查看已收集的监控指标，请导航到监控 > 主机 菜单部分，并点击已创建的 “VMware environment” 主机旁边的 最新数据，或点击为已发现实体创建的某个主机旁边的 最新数据。



此操作将打开所选主机收集到的所有最新监控指标列表。

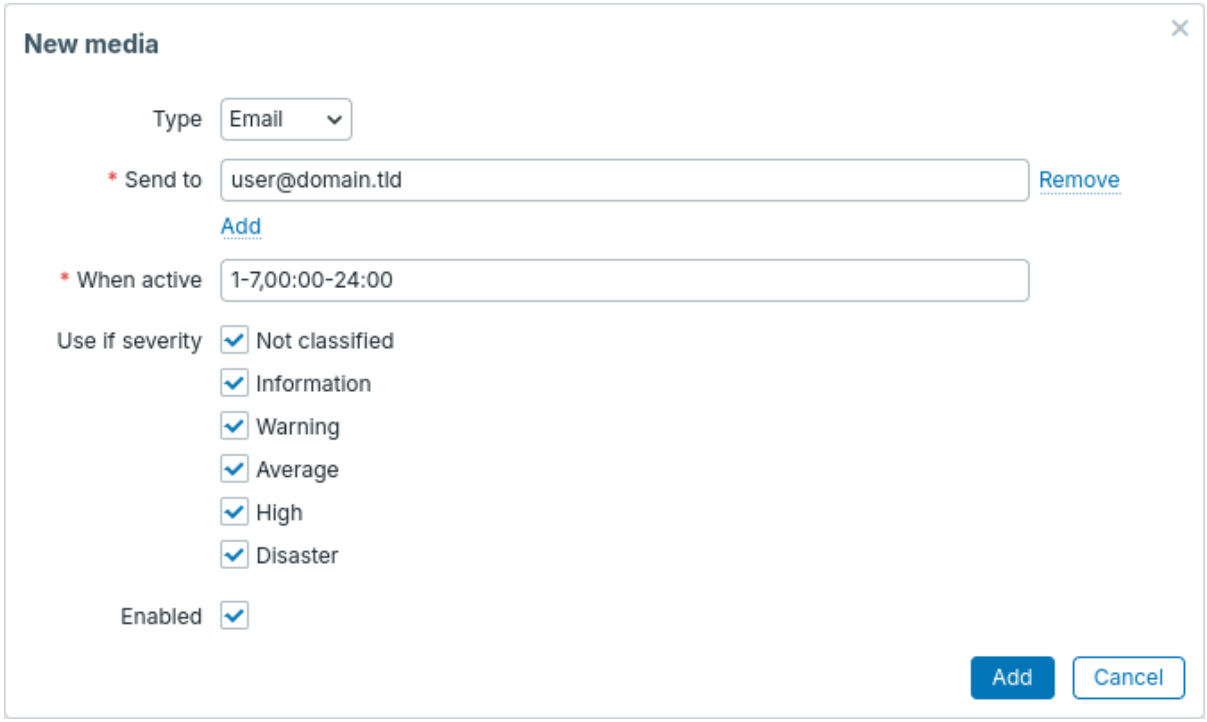


请注意，某些监控项没有数据，并处于 不受支持状态。这是因为 Zabbix 无法在特定的数据存储上找到有效的性能计数器，因为该计数器在正在监控的 VMware 环境中未启用。

设置问题告警

Zabbix 可以通过多种方式通知你基础架构中的问题。本指南提供了发送电子邮件告警的基本配置步骤。

1. 导航到用户设置 > 个人资料，切换到 媒体选项卡，并添加你的电子邮件。



2. 按照接收问题通知指南进行操作。

下次当 Zabbix 检测到问题时，你应该会通过电子邮件收到告警。

另请参见

- 创建监控项 - 如何开始监控其他指标。
- 问题升级 - 如何创建多步骤告警场景（例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，再向数据中心经理发送消息）。
- 虚拟机监控 - 有关 VMware 监控的更多信息（数据采集过程、服务器配置选项、故障排查指南等）。
- VMware 监控项键值 - 可使用 Zabbix 监控的 VMware 指标完整列表。
- 模板 *VMware* - 有关 *VMware* 模板的更多信息。

- 模板 *VMware FQDN* - 有关 *VMware FQDN* 模板的更多信息。

使用 **Zabbix** 监控网络流量

介绍

本页面将引导您完成使用 Zabbix 开始对网络流量进行基础监控所需的步骤。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用网络流量基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或者需要更高级的配置，请参阅 Zabbix 手册中的Configuration 部分。

先决条件

在继续阅读本指南之前，您需要根据操作系统的说明下载并安装 Zabbix 服务器、Zabbix 前端和 Zabbix agent。
请注意，您应在需要进行流量监控的机器上安装 Zabbix agent。
这台机器可以是安装了 Zabbix 服务器的同一主机，也可以是另一台主机。

本指南将提供在名为 *Remote host* 的独立机器上配置 *eth0* 接口网络流量监控的说明。

配置 Zabbix 进行监控

Zabbix agent 可以以主动或被动模式（同时）收集指标。有关更多信息，请参见被动和主动 agent 检查。本指南将介绍通过被动检查进行监控。

配置 Zabbix agent

1. 在安装 agent 的计算机上打开agent 配置文件：

```
sudo vi /etc/zabbix/zabbix_agentd.conf
```

2. 将 Zabbix 服务器的 IP 地址或 DNS 名称添加到 *Server* 参数中。例如：

```
Server=192.0.2.22
```

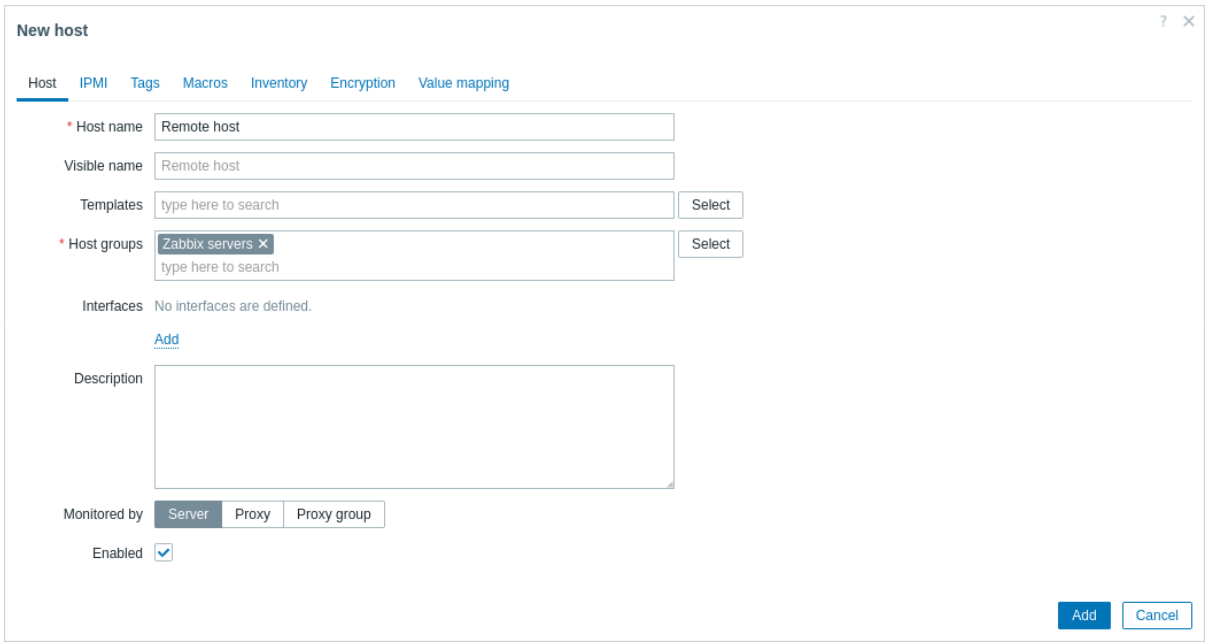
3. 重启 Zabbix agent：

```
systemctl restart zabbix-agent
```

Zabbix 前端

1. 登录 Zabbix 前端。

2. 在 Zabbix Web 界面中创建主机，指定安装了 agent 的机器的 IP 地址或 DNS 名称。



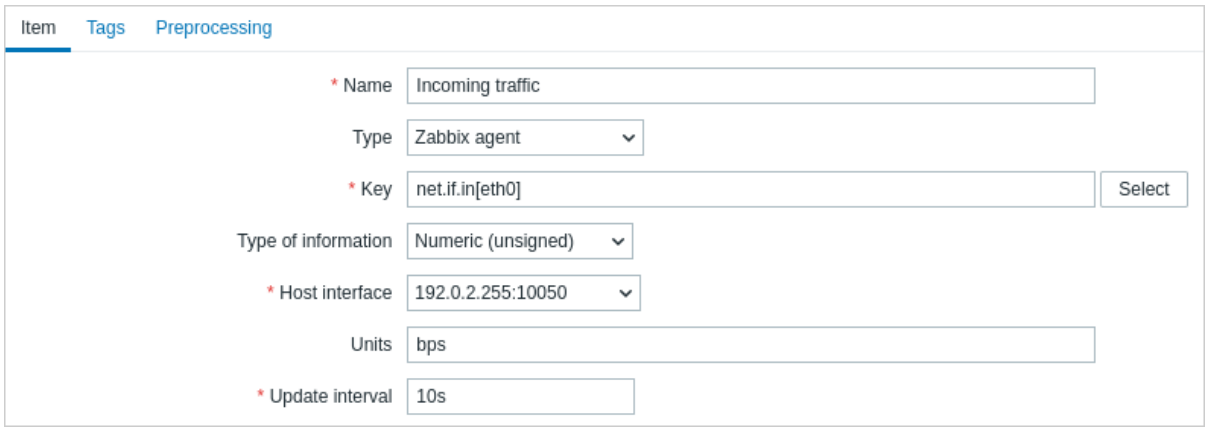
创建监控项

请按照创建监控项中的说明，添加用于流量监控的监控项，即：

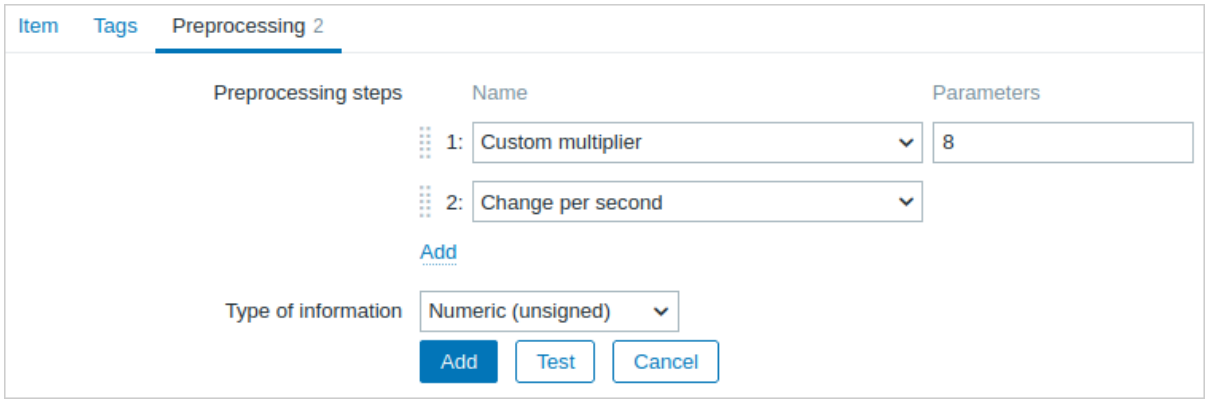
- 入站流量
- 出站流量

- 总流量

入站流量监控项的简单配置如下所示：



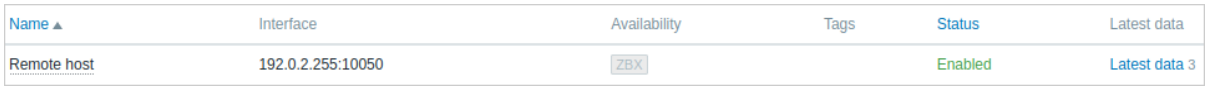
为了使收集到的数据适合实际使用，您可以在创建监控项时设置一些预处理 步骤。在此示例中，这些步骤可以是乘以 8（将字节转换为比特）以及按每秒变化量显示。



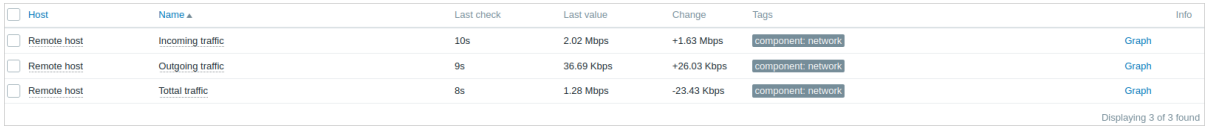
查看收集的数据

恭喜！此时，Zabbix 已经在监控您的网络流量。

要查看收集的指标，请打开监控 > 主机 菜单部分，然后在主机所在行中点击 最新数据。

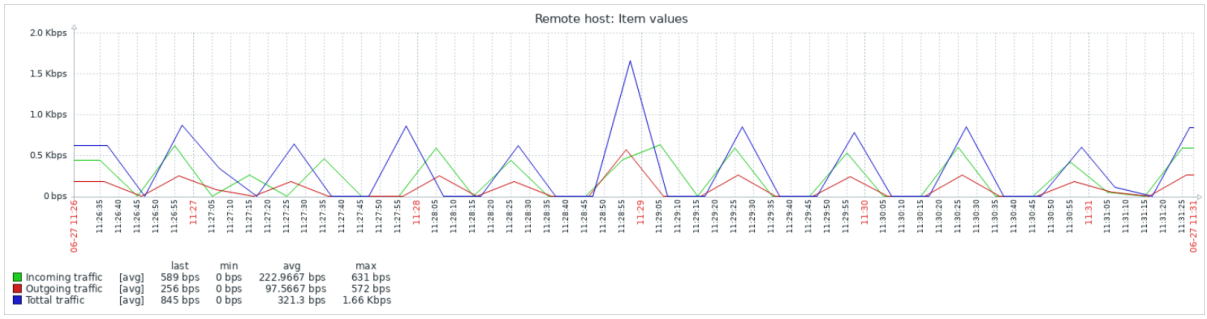


您将看到流量数据。



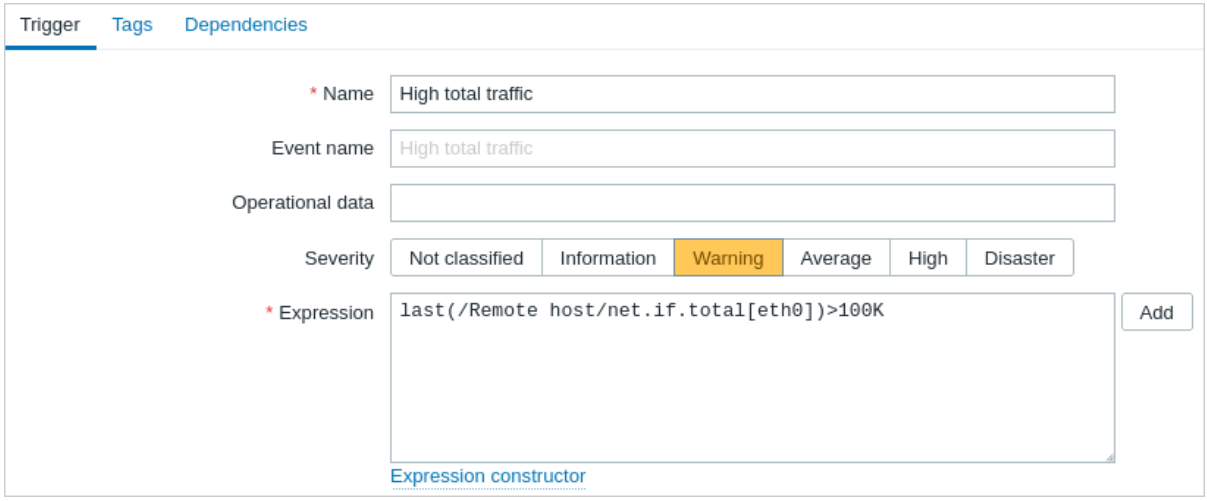
查看图形

收集到的数据可以显示为图形。要查看这些图形，请在 最新数据部分中，单击该监控项所在行中的 图形，或选择所需的监控项，然后单击下方的 显示图形。

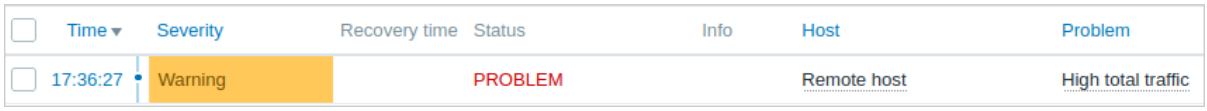


配置触发器

您可以设置触发器 来检测异常网络流量。请参阅配置触发器 的说明，并添加一个用于指示总流量过高的触发器，例如：



现在让流量超过您在触发器表达式中设置的阈值，然后导航到 *Monitoring* > *Problems*，检查该问题是否已列出。



设置问题告警

有多种方式可以接收有关问题的通知。电子邮件是最常用的方式，请按照通过电子邮件设置问题通知的说明进行操作。您也可以选择其他用于发送通知的媒介类型。

另请参阅：

- 问题升级 - 如何创建多步骤告警场景
  (例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，
  再向数据中心经理发送消息)。
- 问题确认 - 如何表示该问题已知，
  对解决过程添加注释，抑制或关闭该问题。
- 使用 Zabbix agent 监控 Linux - 如何通过关联预配置的模板来启动
  对最重要监控项的基本监控。

使用主动检查监控网络流量

介绍

本页面将引导您完成使用主动检查开始对网络流量进行基础监控所需的步骤。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用使用主动检查对网络流量进行基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或者需要更高级的配置，请参阅 Zabbix 手册中的Configuration 部分。

先决条件

在继续阅读本指南之前，您需要根据您的操作系统说明下载并安装 Zabbix 服务器、Zabbix 前端和 Zabbix agent。
请注意，您应在需要进行流量监控的机器上安装 Zabbix agent。
该机器可以是安装了 Zabbix 服务器的同一主机，也可以是另一台主机。

本指南将提供在名为 *Remote host* 的独立机器上配置 *eth0* 接口网络流量监控的说明。

配置 Zabbix 进行监控

Zabbix agent 可以以主动或被动模式（同时）收集指标。有关更多信息，请参见被动和主动 agent 检查。本指南将介绍通过主动检查进行监控。

配置 Zabbix agent

1. 在安装 agent 的计算机上打开agent 配置文件。

如果使用的是 Zabbix agent：

```
sudo vi /etc/zabbix/zabbix_agentd.conf
```

如果使用的是 Zabbix agent 2：

```
sudo vi /etc/zabbix/zabbix_agent2.conf
```

2. 将 Zabbix 服务器的 IP 地址或 DNS 名称（以及可选端口）添加到 ServerActive 参数中。例如：

```
ServerActive=192.0.2.1:10051
```

Zabbix agent 将使用此地址连接到 Zabbix 服务器的 trapper 端口（默认值：10051），并请求主动检查的配置数据。

3. 定义 Hostname 参数。该参数必须与将在Zabbix 前端中定义的主机名匹配。在本例中，该参数为：

```
Hostname=Remote host
```

Hostname 值必须匹配，因为对于主动检查，Zabbix agent 会使用该值从服务器检索正确的主机配置。具体来说，agent 会发起与服务器的连接，并使用 Hostname 值标识自身。然后，服务器会提供该主机的监控配置。如果这些值不一致，agent 将无法接收适当的配置，从而导致指标缺失或监控问题。

4. 重启 Zabbix agent。

如果使用的是 Zabbix agent：

```
systemctl restart zabbix-agent
```

如果使用的是 Zabbix agent 2：

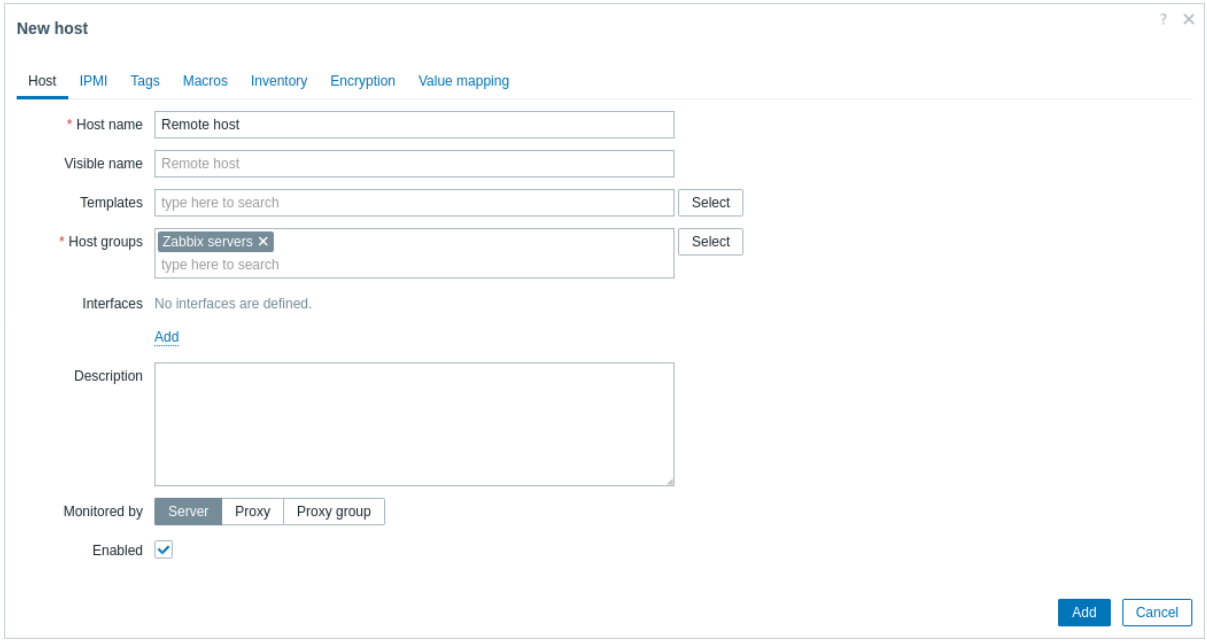
```
systemctl restart zabbix-agent2
```

Zabbix 前端

1. 登录 Zabbix 前端。

2. 在 Zabbix web 界面中创建主机。

- 在 *Host name* 字段中，输入一个主机名（例如，“Remote host”），该名称应与之前在 agent 配置文件中定义的 Hostname 参数值一致。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，“Zabbix servers”）。
- 保持 *Interfaces* 未定义；对于主动检查，不需要接口，因为 agent 会主动发起到服务器的连接，而不是监听来自服务器的连接。



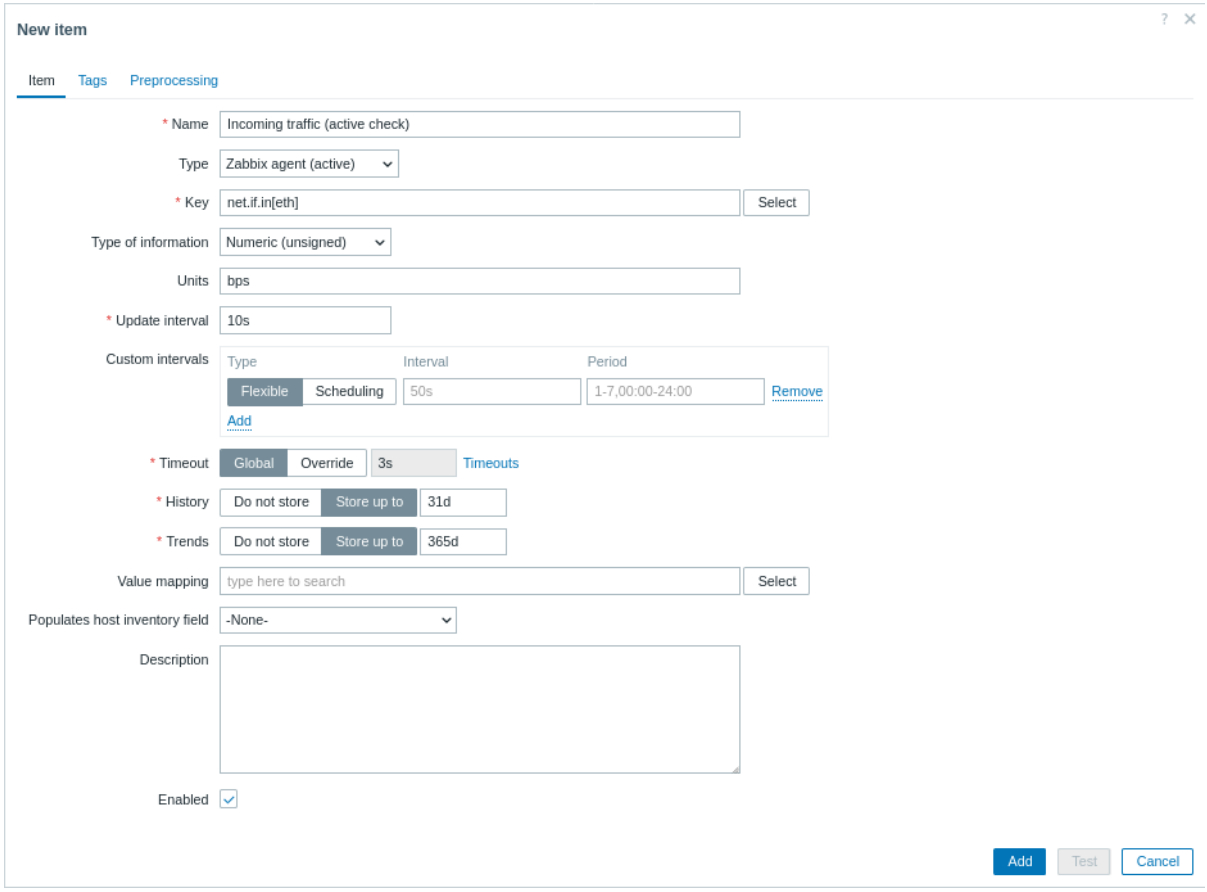
3. 单击 *Add* 以添加主机。此主机将代表被监控的 Linux 机器。

创建监控项

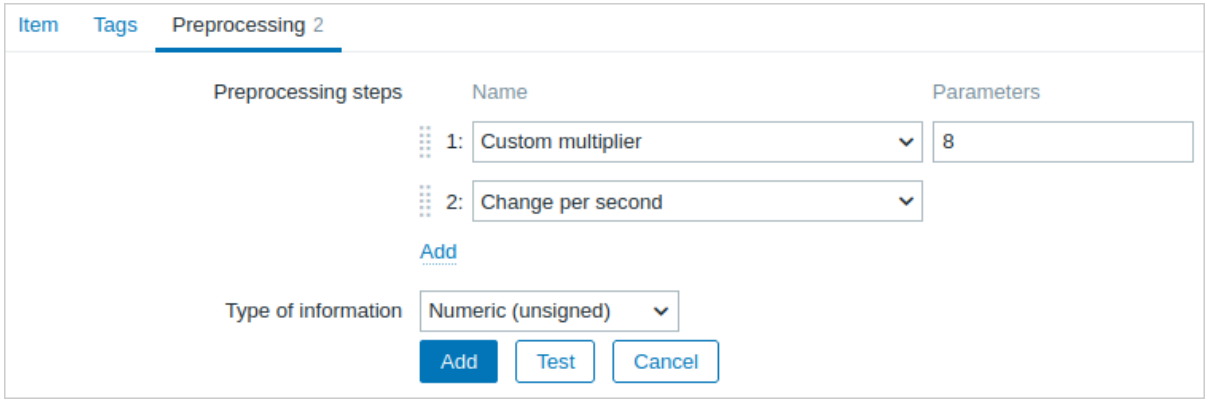
请按照创建监控项中的说明，添加用于流量监控的监控项，即：

- 入站流量
- 出站流量
- 总流量

使用主动检查的入站流量监控项的简单配置如下所示：



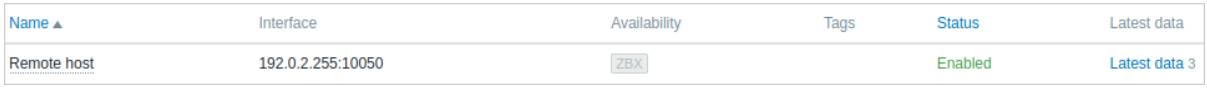
为了使采集到的数据适合实际使用，您可以在创建监控项时设置一些预处理步骤。在此示例中，这些步骤可以是乘以 8（将字节转换为比特）以及按每秒变化量显示。



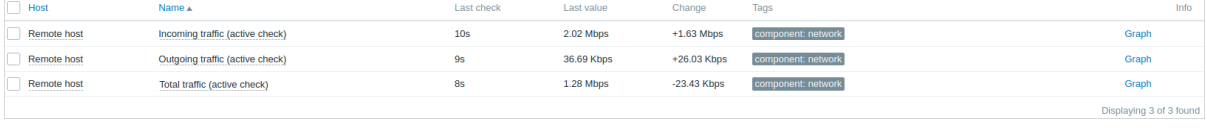
查看已收集的数据

恭喜！此时，Zabbix 已经在监控你的网络流量了。

要查看已收集的监控项，请打开监控 > 主机 菜单部分，并点击该主机所在行中的 最新数据。



你将看到流量数据。



另请参阅：

- 查看图形 - 如何将收集到的数据以图形方式显示。
- 配置触发器 - 如何设置触发器以检测异常网络流量。
- 设置问题告警 - 如何设置有关问题情况的通知。

- 问题升级 - 如何创建多步骤告警场景（例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，再向数据中心经理发送消息）。
- 问题确认 - 如何指示该问题已知，对解决过程添加评论，抑制或关闭该问题。
- 使用 Zabbix agent 监控 Linux - 如何通过链接预配置的模板来开始对最重要的监控项进行基本监控。
- 从软件包安装 - 如何使用适用于各种 Linux 发行版的官方 RPM 和 DEB 软件包安装 Zabbix 组件，确保可以访问最新功能和错误修复。

使用浏览器监控项监控网站

简介

本页将引导您完成使用浏览器监控项开始对网站进行基础监控所需的步骤。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用使用 Browser 监控项对网站进行基本监控所需的最少步骤。如果您正在寻找更深入的自定义选项，或者需要更高级的配置，请参阅 Zabbix 手册中的Browser 监控项 页面或配置 部分。

先决条件

在继续阅读本指南之前，您需要根据您的操作系统说明下载并安装 Zabbix 服务器和 Zabbix 前端。

根据您的环境，本指南中的某些步骤可能会略有不同。本指南基于运行 Ubuntu 的环境。

配置 WebDriver

浏览器监控项需要一个自动化框架（Selenium Server 或普通的 WebDriver，例如 ChromeDriver）作为 web 测试端点，用于控制浏览器并与之交互，执行诸如点击按钮或输入文本之类的测试命令。
作为示例，本指南将使用在 Docker 容器中运行的 Selenium Server 和 Chrome。

假定 Docker 已经配置完成。本指南不涵盖 Docker 的配置。
有关安装说明，请参见 在 Ubuntu 上安装 Docker Engine。

1. 使用以下选项在 Docker 容器中启动带有 Chrome 的 Selenium Server：

- **docker run --name browser** - 运行一个名为"browser" 的新 Docker 容器；
- **-p 4444:4444** - 将主机上的 4444 端口映射到容器中的 4444 端口（这是 Selenium Server 用于接收命令的端口）；
- **-p 7900:7900** - 将主机上的 7900 端口映射到容器中的 7900 端口（这是 Virtual Network Computing (VNC) 服务器使用的端口，可让您远程查看浏览器 GUI；需要 VNC 客户端）；
- **--shm-size="2g"** - 为容器分配 2GB 共享内存（这对 Chrome 正常运行很重要，因为它可能需要大量共享内存以避免崩溃）；
- **-d** - 以分离模式运行容器，这意味着它将在后台运行；
- **selenium/standalone-chrome:latest** - 指定要使用的 Docker 镜像；在本例中为 Selenium Server with Chrome 的最新版本。

```
docker run --name browser \
-p 4444:4444 \
-p 7900:7900 \
--shm-size="2g" \
-d selenium/standalone-chrome:latest
```

2. 确保 browser Docker 容器正在运行且可访问。

- 获取容器的 IP 地址（在本示例中为 192.0.2.1）：

```
ip addr

##### 1: lo: <LOOPBACK,UP,LOWER_UP>
#####     ...
##### 3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> ...
#####     inet 192.0.2.1/16 brd 192.0.255.255 scope global docker0
#####     ...
```

- 使用 Ncat 测试与容器的连接：

```
nc -zv 192.0.2.1 4444

##### Connection to 192.0.2.1 4444 port [tcp/*] succeeded!
```

- 使用 curl 从 Selenium Server 获取网页内容：

```
curl -L 192.0.2.1:4444
```

```
##### <!DOCTYPE html>
##### <html lang="en">
#####
##### <head>
#####   <meta charset="utf-8"/>
#####   <link href="favicon.svg" rel="icon" type="image/svg">
#####   <meta content="width=device-width, initial-scale=1" name="viewport"/>
#####   <link href="logo192.png" rel="apple-touch-icon"/>
#####   <link href="manifest.json" rel="manifest"/>
#####   <title>Selenium Grid</title>
##### </head>
#####
##### <body>
##### ...
```

> **Note:**
> 如需故障排除，请参阅 Docker 文档。

配置 Zabbix 服务器

浏览器监控项由 *browser poller* Zabbix 进程执行和处理。需要通过调整服务器配置参数StartBrowserPollers 来启用这些进程。此外，WebDriverURL 参数应指定之前配置的 Web 测试端点。

默认情况下，StartBrowserPollers 参数设置为 1，因此您只需指定 Web 测试端点。

1. 打开 Zabbix 服务器配置文件：

```
vi /etc/zabbix/zabbix_server.conf
```

2. 在 Zabbix 服务器配置文件中找到并设置WebDriverURL 参数：

```
###### Option: WebDriverURL
#####WebDriver interface HTTP(S) URL. For example, http://localhost:4444 used with Selenium WebDriver star
#####
##### Mandatory: no
##### Default:
##### WebDriverURL=

WebDriverURL=192.0.2.1:4444
```

3. 重启 Zabbix 服务器。

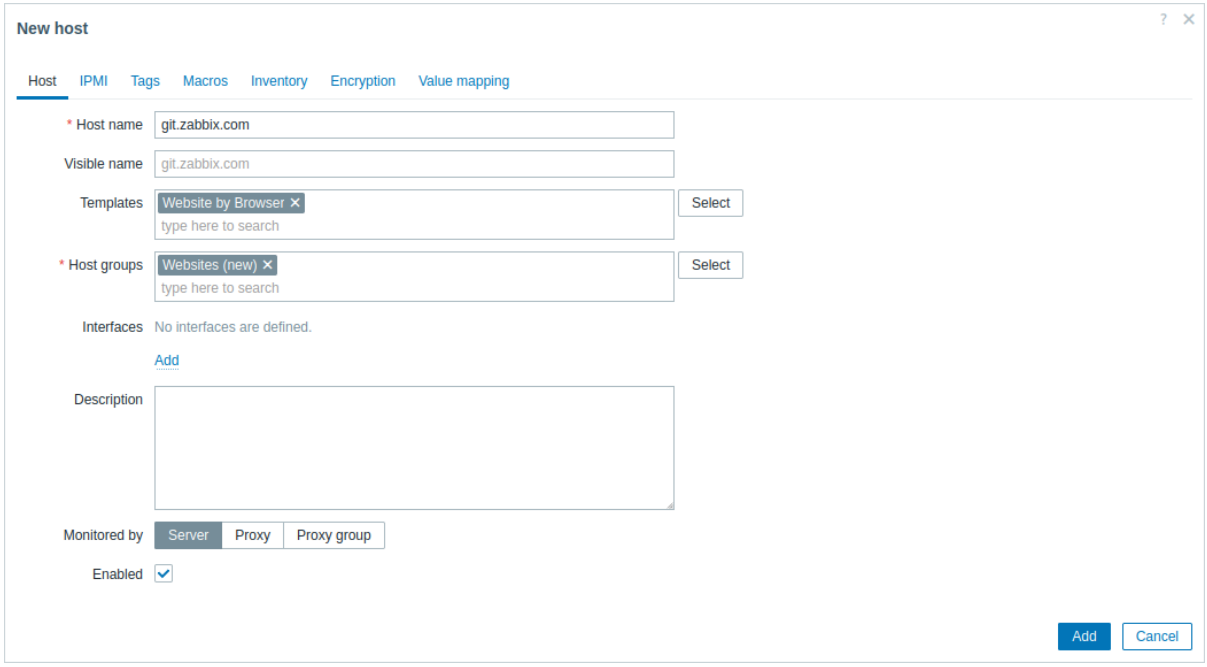
```
systemctl restart zabbix-server
```

配置 Zabbix 前端

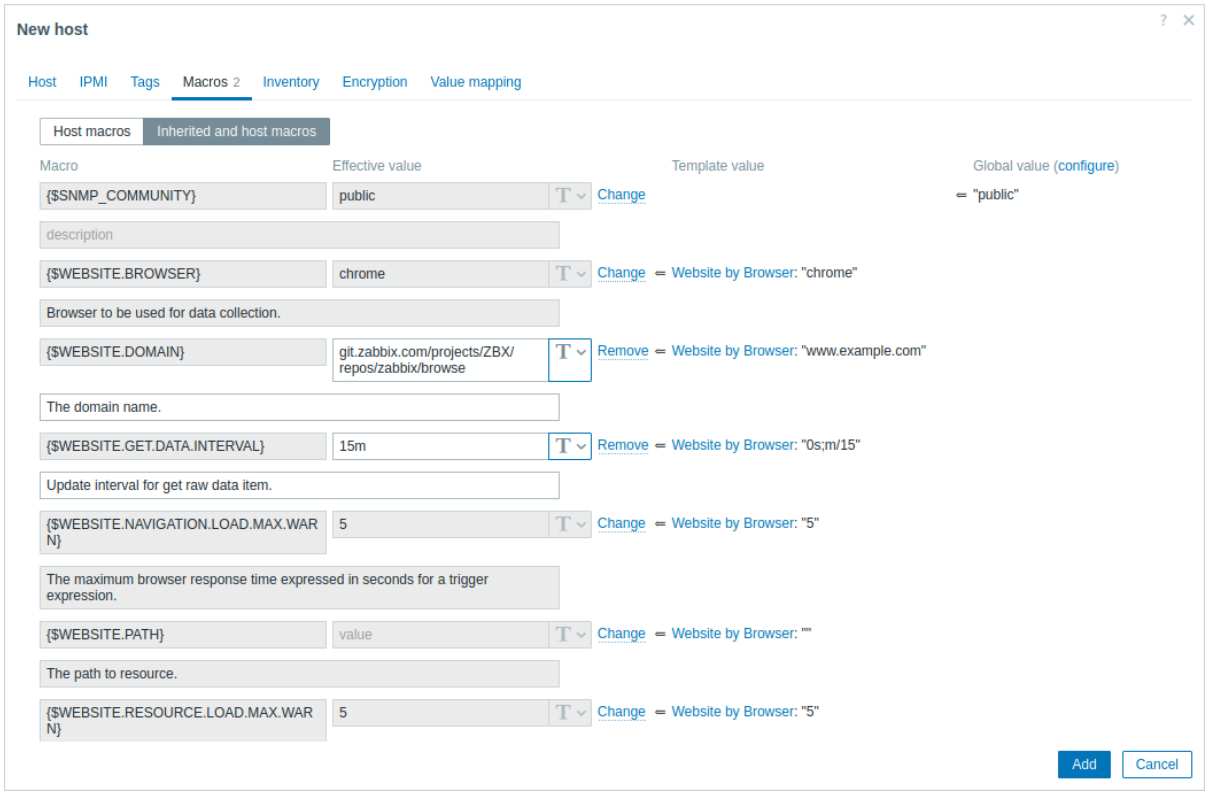
1. 登录 Zabbix 前端。

2. 在 Zabbix Web 界面中创建主机：

- 在 *Host name* 字段中，输入主机名称（例如，“git.zabbix.com”）。
- 在 *Templates* 字段中，输入或选择 “Website by Browser” 模板。有关此模板的更多信息，请参见 Website by Browser。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，一个新的主机组 “Websites”）。



- 在 *Macros* 选项卡中，切换到 *Inherited and host macros*，查找以下宏，并单击宏值旁边的 *Change* 进行更新：
  - {$WEBSITE.DOMAIN} - 域名（例如，git.zabbix.com/projects/ZBX/repos/zabbix/browse）
  - {$WEBSITE.GET.DATA.INTERVAL} - 监控项数据更新间隔（例如，15m）

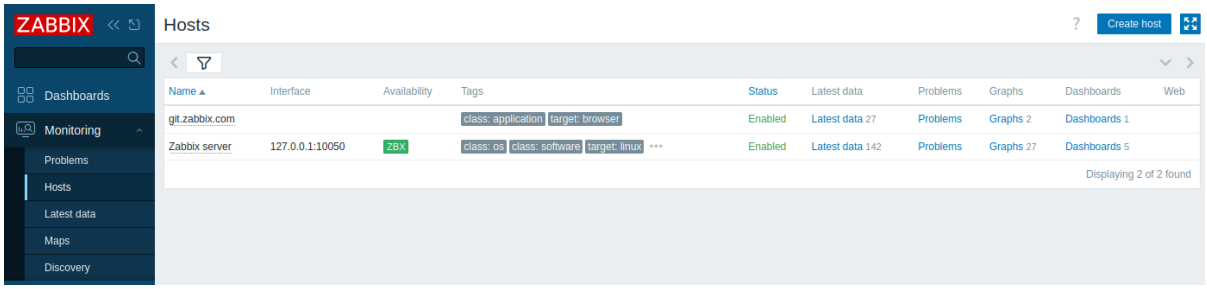


3. 单击 *Add* 按钮创建主机。此主机将代表您要监控的网站。

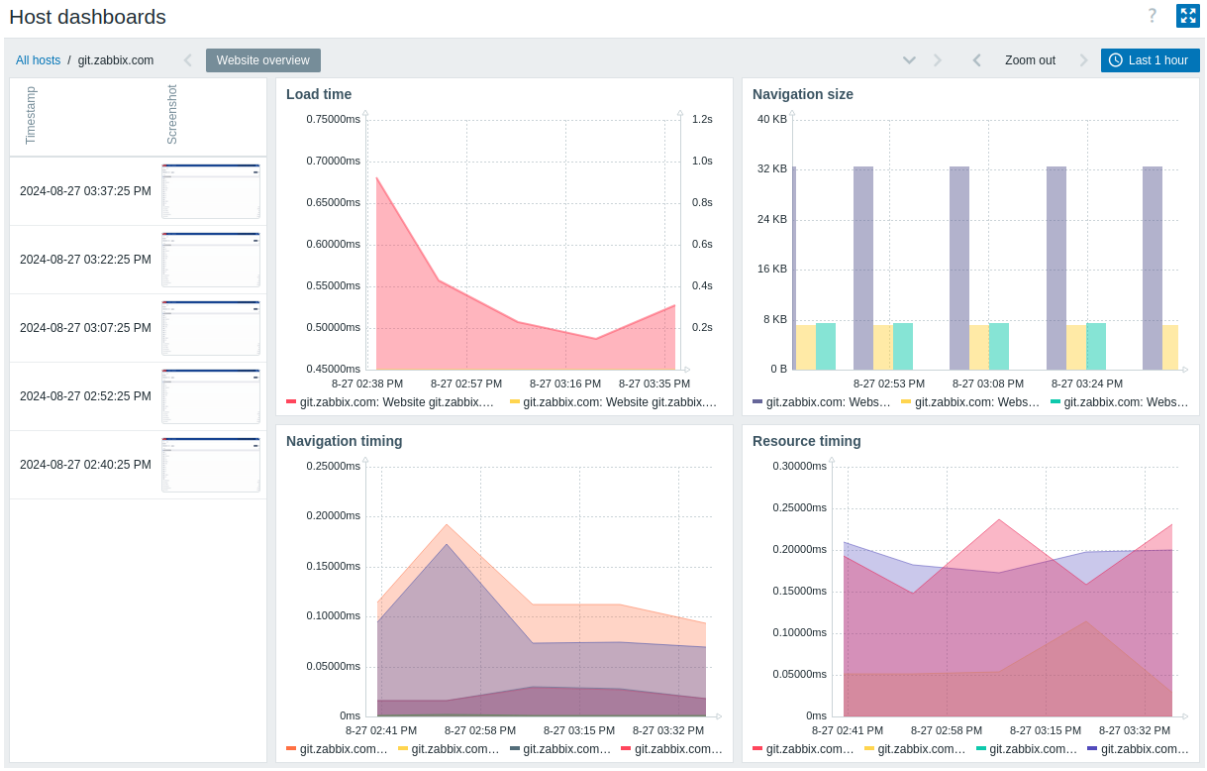
查看已收集的监控数据

恭喜！此时，Zabbix 已经在监控您指定的网站了。

要查看已收集的监控数据，请导航到*Monitoring > Hosts* 菜单部分，并单击主机旁边的 *Dashboards*。



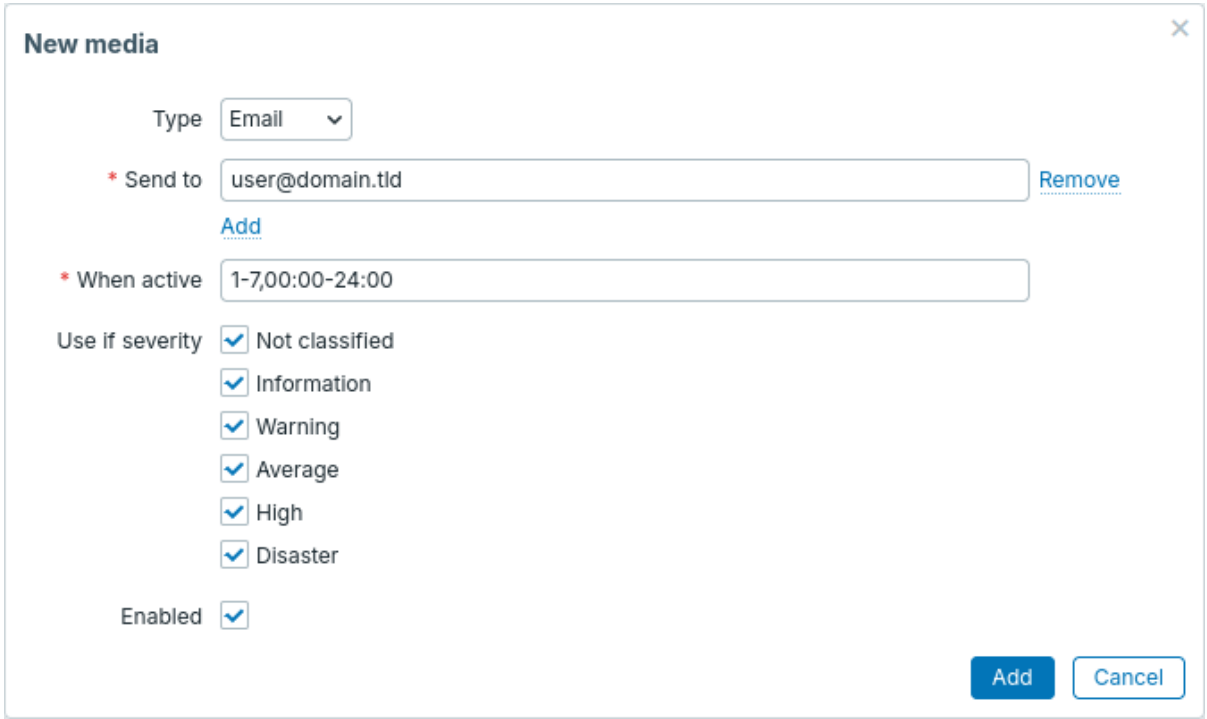
此操作将带您进入主机仪表板（在模板级别配置），其中包含从该网站收集到的最重要监控数据。



设置问题告警

Zabbix 可以通过多种方式就基础设施中的问题向您发送通知。本指南提供了发送电子邮件告警的基本配置步骤。

1. 导航到用户设置 > 个人资料，切换到 媒体选项卡，并添加您的电子邮件。



2. 按照接收问题通知指南进行操作。

下次当 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

测试你的配置

要测试你的配置，我们可以通过在 Zabbix 前端中更新主机配置来模拟一个真实问题。

1. 在 Zabbix 中打开你的网站主机配置。

2. 切换到 *Macros* 选项卡，并选择 *Inherited and host macros*。

3. 点击例如之前配置的 {$WEBSITE.DOMAIN} 宏值旁边的 *Change*，并设置一个错误的域名（例如，/git.zabbix.com/projects/ZBX/repos/zabbix/bro

4. 点击 *Update* 以更新主机配置。

5. 过一会儿，Zabbix 将检测到问题 “Failed to get JSON of the requested website”，因为它将无法连接到指定的网站。该问题将显示在*Monitoring > Problems* 中。



如果已配置 告警，你还会收到该问题通知。

6. 将宏值改回之前的值，以解决该问题并继续监控该网站。

另请参见

- 创建监控项 - 如何开始监控其他指标。
- 问题升级 - 如何创建多步骤告警场景（例如，先向系统管理员发送消息，然后，如果问题在 45 分钟内未解决，再向数据中心经理发送消息）。
- 浏览器监控项 - 如何配置浏览器监控项。
- 模板 Website by Browser - 关于 *Website by Browser* 模板的更多信息。

使用 **Zabbix agent 2**（被动）监控网站证书

简介

本指南全面概述了如何在 Zabbix agent 2 中使用 `web.certificate.get` 键来设置和监控 SSL/TLS 证书。它旨在简化单个或多个网站的证书监控，帮助管理员快速识别诸如证书过期或无效等潜在问题。

本指南适用对象

本指南面向新的 Zabbix 用户，包含启用网站证书基础监控所需的最少步骤。如果您正在寻找深度自定义选项，或需要更高级的配置，请参阅 Zabbix 手册中的Configuration 部分。

前提条件

在继续阅读本指南之前，您需要根据您的操作系统说明下载并安装 Zabbix 服务器、Zabbix 前端和 Zabbix agent 2。本教程假定 Zabbix 服务器和 agent 都安装在同一台机器上；因此，在配置中使用 `127.0.0.1`。

配置 Zabbix agent 2

1. 打开 Zabbix agent 配置文件：

```
sudo vi /etc/zabbix/zabbix_agent2.conf
```

2. 将 Server 参数设置为 127.0.0.1，因为 agent 和服务器运行在同一台机器上：

```
Server=127.0.0.1
```

3. 保存文件并重启 Zabbix agent 2 服务：

```
sudo systemctl restart zabbix-agent2
```

4. 完成 Zabbix agent 2 的安装和配置后，使用以下命令测试其可用性：

```
zabbix_get -s 127.0.0.1 -k web.certificate.get[<website_DNS_name>]
```

Zabbix agent 2 默认包含 WebCertificate 插件，因此无需单独安装或配置。

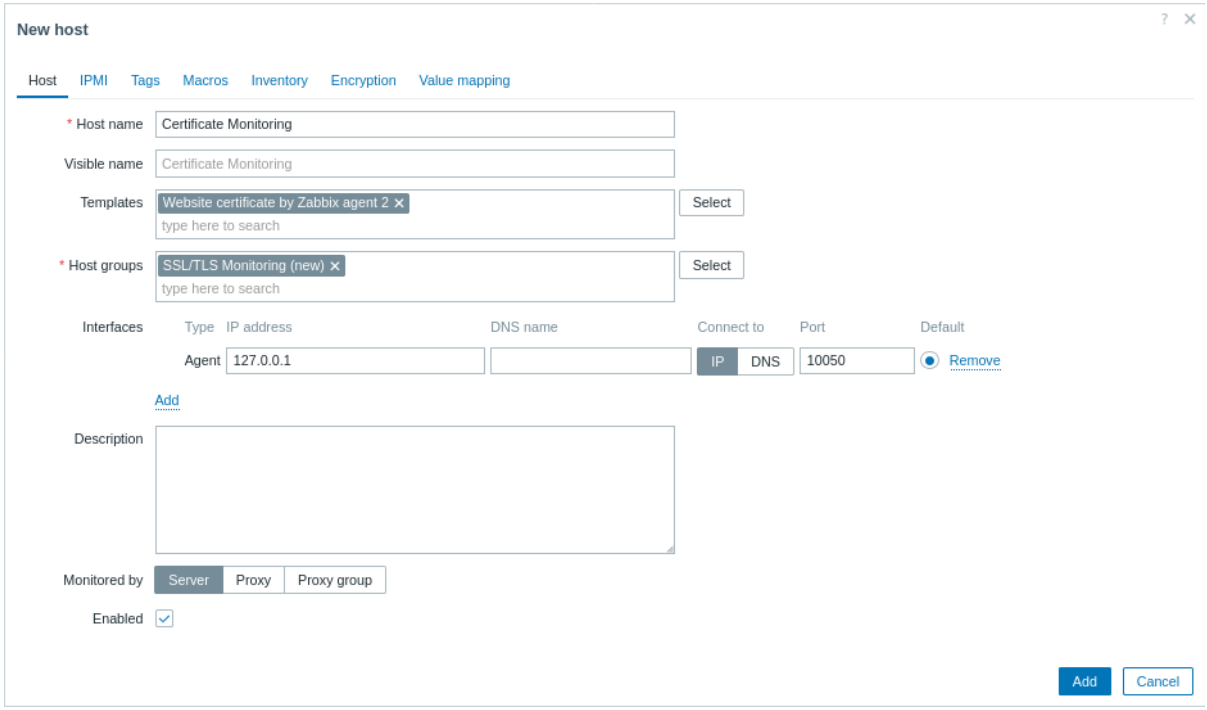
配置 Zabbix 前端

1. 登录到 Zabbix 前端。

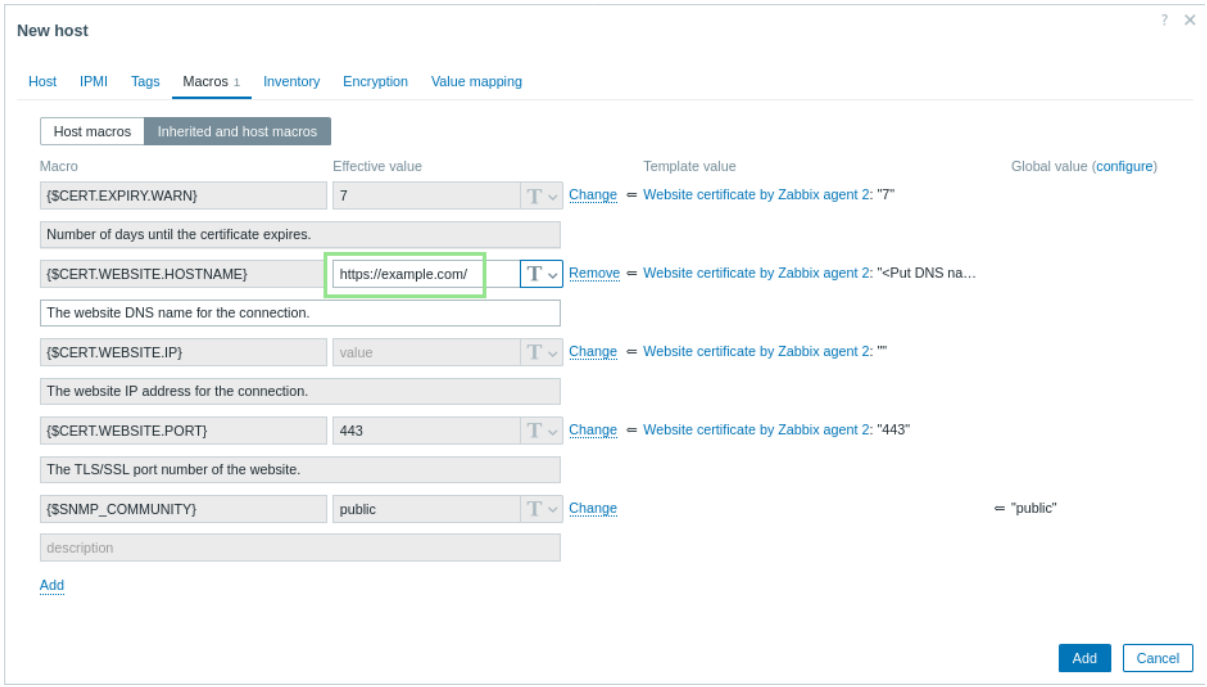
2. 导航到 *Monitoring* > *Hosts*。

3. 点击要监控网站证书的现有主机，或者在必要时创建主机：

- 在 *Host name* 字段中，输入主机名（例如，“Certificate Monitoring”）。
- 在 *Templates* 字段中，输入或选择将链接到主机的模板 “Website certificate by Zabbix agent 2”。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，“SSL/TLS Monitoring”）。
- 在 *Interfaces* 字段中，添加一个类型为 “Agent” 的接口并指定 IP 地址。在本示例中，使用 “127.0.0.1”。



- 在 *Macros* 选项卡中，切换到 *Inherited and host macros*，查找以下宏，并点击宏值旁边的 *Change* 进行更新：
  - `{$CERT.WEBSITE.HOSTNAME}` - 输入所需的网站 DNS 名称作为值。



4. 点击 *Add* 以添加主机。

要监控多个网站，您可以在 `{$CERT.WEBSITE.HOSTNAME}` 宏中指定一个以逗号分隔的主机名列表。可选地，您还可以在 `{$CERT.WEBSITE.PORT}` 宏中提供一个以逗号分隔的端口列表，其中每个端口按顺序对应相应的主机名。例如，要监控端口 8443 上

的 example.com 和 example.org：

- {$CERT.WEBSITE.HOSTNAME}: example.com,example.org
- {$CERT.WEBSITE.PORT}: 443,8443

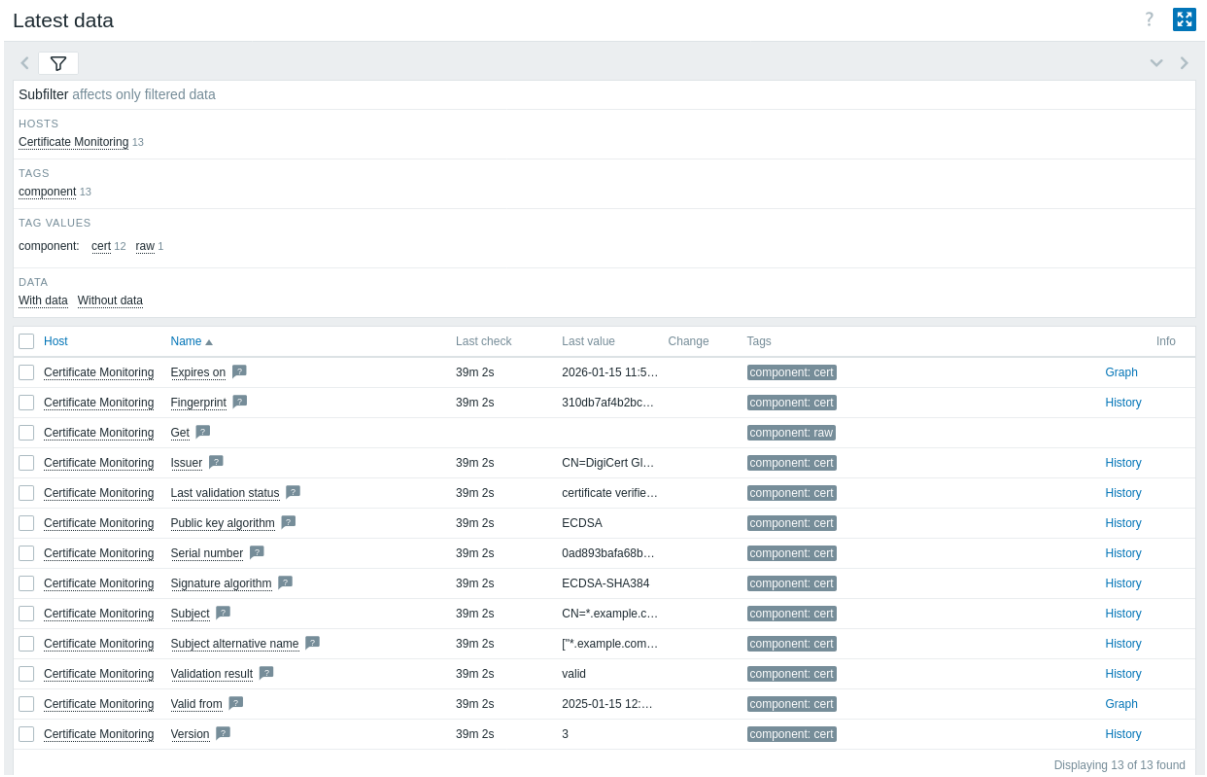
对于在 {$CERT.WEBSITE.HOSTNAME} 宏中指定的每个网站，Zabbix 都会创建一组相应的监控项和触发器。这样可以对每个网站的 SSL 证书进行单独监控和告警。

Host	Name ▲	Last check	Last value
Certificate Monitoring	Get data	11m 16s	{"error":{"message…
Certificate Monitoring	[example.com]: Expires on	11m 6s	2026-01-15 11:59:…
Certificate Monitoring	[example.com]: Fingerprint	11m 6s	310db7af4b2bc90…
Certificate Monitoring	[example.com]: Get		
Certificate Monitoring	[example.com]: Issuer	11m 6s	CN=DigiCert Glob…
Certificate Monitoring	[example.com]: Last validation status	11m 6s	certificate verified …
Certificate Monitoring	[example.com]: Public key algorithm	11m 6s	ECDSA
Certificate Monitoring	[example.com]: Serial number	11m 6s	0ad893bafa68b0b…
Certificate Monitoring	[example.com]: Signature algorithm	11m 6s	ECDSA-SHA384
Certificate Monitoring	[example.com]: Subject	11m 6s	CN=*.example.co…
Certificate Monitoring	[example.com]: Subject alternative name	11m 6s	["*.example.com","…
Certificate Monitoring	[example.com]: Validation result	11m 6s	valid
Certificate Monitoring	[example.com]: Valid from	11m 6s	2025-01-15 12:00:…
Certificate Monitoring	[example.com]: Version	11m 6s	3
Certificate Monitoring	[example.org]: Expires on	10m 52s	2026-01-15 11:59:…
Certificate Monitoring	[example.org]: Fingerprint	10m 52s	3b451cfce915637…
Certificate Monitoring	[example.org]: Get		
Certificate Monitoring	[example.org]: Issuer	10m 52s	CN=DigiCert Glob…
Certificate Monitoring	[example.org]: Last validation status	10m 52s	certificate verified …
Certificate Monitoring	[example.org]: Public key algorithm	10m 52s	ECDSA
Certificate Monitoring	[example.org]: Serial number	10m 52s	0722a749b558476…
Certificate Monitoring	[example.org]: Signature algorithm	10m 52s	ECDSA-SHA384
Certificate Monitoring	[example.org]: Subject	10m 52s	CN=*.example.org…
Certificate Monitoring	[example.org]: Subject alternative name	10m 52s	["*.example.org","e…
Certificate Monitoring	[example.org]: Validation result	10m 52s	valid
Certificate Monitoring	[example.org]: Valid from	10m 52s	2025-01-15 12:00:…

查看已收集的监控指标

恭喜！此时，Zabbix 已经在监控所需的 web 证书。

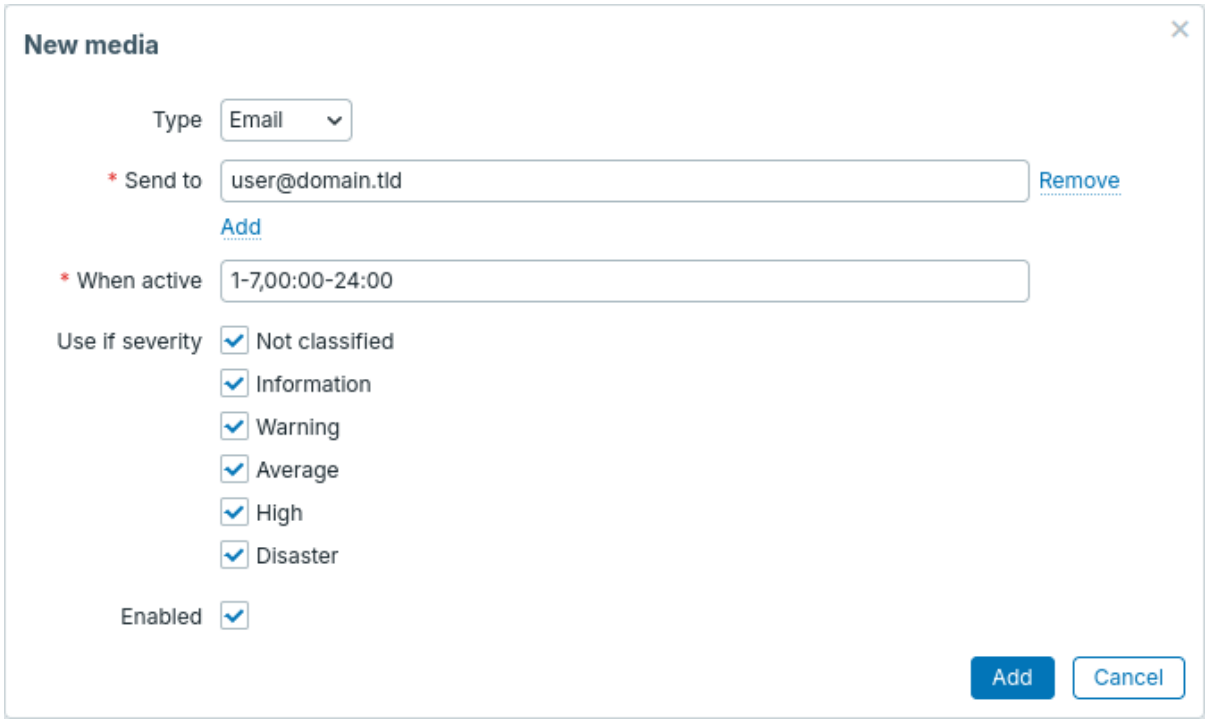
要查看已收集的监控指标，请转到*Monitoring > Hosts* 菜单部分，然后单击主机旁边的*Latest data*，以列表形式查看所有最新收集的监控指标，例如到期日期、颁发者和主题。



设置问题告警

Zabbix 可以通知您基础设施中的问题。本指南提供发送电子邮件告警的基本配置步骤。

1. 导航到用户设置 > 配置文件，切换到 媒体选项卡，并添加您的电子邮件。



2. 按照接收问题通知指南进行操作。

下次当 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

测试你的配置

要测试你的配置，我们可以通过在 Zabbix 前端中更新主机配置来模拟一个真实问题。

1. 在 Zabbix 中打开你的”Certificate Monitoring” 主机配置。

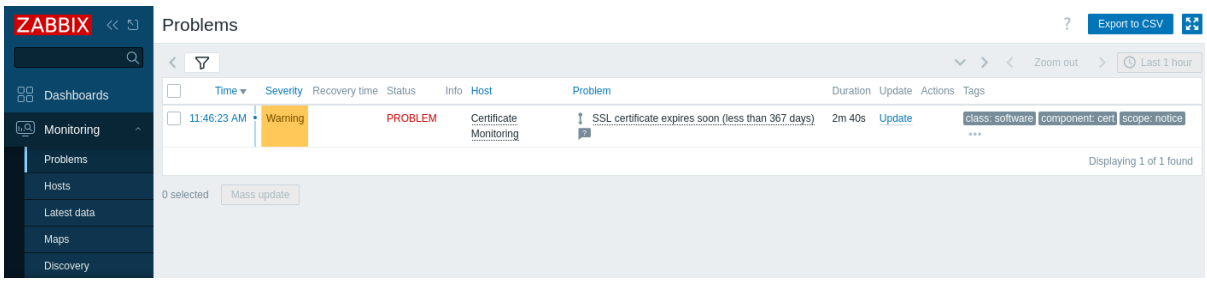
2. 切换到 *Macros* 选项卡，并选择 *Inherited and host macros*。

3. 点击之前配置的 `{$CERT.EXPIRY.WARN}` 宏值旁边的 *Change*，并将其设置为一个非常大的天数（大于 365 天即可），以便在证书过

期前收到警告。

4. 点击 *Update* 以更新主机配置。

5. 过一会儿，Zabbix 将检测到"SSL certificate expires soon" 问题，并显示距离过期还有多少天。该问题将出现在*Monitoring > Problems* 中。



如果已配置告警，你还会收到该问题通知。

6. 将宏值改回之前的值，以解决该问题并继续监控证书值。

另请参见

- Zabbix agent 2 - 列出监控项键值。
- 模板 *Website certificate by Zabbix agent 2* - 关于 *Website certificate by Zabbix agent 2* 模板的更多信息。
- 模板 *Website certificate by Zabbix agent 2 active* - 关于 *Website certificate by Zabbix agent 2 active* 模板的更多信息。
- 使用 Browser 监控项监控网站 - 如何使用 Browser 监控项开始对网站进行基础监控。

使用 **Zabbix** 监控网络交换机或路由器

介绍

本指南将引导您完成使用 Zabbix 开始对网络交换机或路由器进行基础监控所需的步骤。本文以 Cisco 路由器为例，但该流程同样适用于任何支持 SNMP 的网络设备。

本指南适用对象

本指南面向希望快速启用网络设备基础监控的新 Zabbix 用户和网络管理员。如果您需要更深入的自定义或高级配置选项，请参阅SNMP agent 页面或 Zabbix 手册的Configuration 部分。

前提条件

在继续阅读本指南之前，请确保您已具备以下条件：

- 已安装 Zabbix 服务器和 Zabbix 前端：请按照您的操作系统对应的说明进行安装（参见 从软件包安装 和Web 界面安装）。
- 如果要监控本地网络指标，请安装 Zabbix agent。
- 支持 SNMP 的设备：已启用 SNMP 的网络交换机或路由器（例如 Cisco 路由器）。
- 已安装MIB 文件：安装 MIB 文件后，Zabbix 可以将数字 OID 转换为人类可读的名称和描述。如果没有正确的 MIB 支持，您可能只能看到数字值，这会使配置监控项和排查问题更加困难。

在 Ubuntu 上安装 MIB 文件：

1. 安装 MIB 下载器软件包：

```
sudo apt-get update
sudo apt-get install snmp-mibs-downloader
```

如果您需要添加厂商特定的 MIB（例如来自 Cisco、Juniper），请将它们放置到相应的 MIB 目录中：

- 对于基于 Linux 的系统，常见位置包括 `/usr/share/snmp/mibs/` 或 `/usr/local/share/snmp/mibs/`。
- 对于 Zabbix 安装，MIB 文件可以存储在 `/var/lib/zabbix/mibs/`。

请确保 `MIBDIRS` 环境变量或 `snmp.conf` 文件包含正确的路径。

要验证系统是否识别新的 MIB，请使用：

```
snmptranslate -IR -On <MIB-NAME>::<object>
```

有关详细说明，请参阅您的 SNMP 库文档：

- Cisco MIBs
- Juniper MIBs

2. 编辑 `/etc/snmp/snmp.conf`，并注释掉以 `mibs :` 开头的行，以允许系统加载所有可用的 MIB。

3. 通过运行 `snmpwalk` 进行验证（例如，`snmpwalk -v 2c -c <your_community_string> <device_IP>`），并检查 OID 是否以描述性名称显示。

根据您的环境不同，本指南中的某些步骤可能会略有差异。本指南基于 Ubuntu 环境以及正在监控的 Cisco Catalyst 3750V2-24FS 网络设备。

假定您的网络设备已经完成物理安装并已连接。

配置网络设备（Cisco 路由器示例）

对于通过 SNMP 进行监控，您必须配置网络设备以允许 SNMP 查询。下面的示例适用于 SNMPv2，且未考虑现有设置。注意：应用这些命令可能会覆盖当前的 SNMP 配置。

对于 Cisco 路由器，配置通常包括以下步骤。

SNMPv2 示例

1. 启用 SNMP 并设置 community 字符串。

登录到你的 Cisco 路由器控制台并进入配置模式：

```
configure terminal
```

然后，通过指定只读 community 字符串来启用 SNMP。例如：

```
snmp-server community <your_community_string> RO
```

将 `<your_community_string>` 替换为你的安全 community 字符串。注意：RO（只读）选项允许 SNMP 从设备检索数据，但会阻止任何配置更改。

出于安全考虑，建议仅将 SNMP 访问限制为必要的设备。有关配置访问控制列表（ACL）的进一步指导，请参阅 Cisco 官方文档。

2. 保存配置。

保存更改，以确保 SNMP 设置在重启后仍然保留：

```
write memory
```

SNMPv3 示例

SNMPv3 通过身份验证和加密提供增强的安全性。其配置比 SNMPv2 更安全，应结合您的设备专用文档进行验证。

1. 创建一个 SNMP 组。

配置一个启用了隐私（加密）的 SNMPv3 组：

```
configure terminal
snmp-server group <your_group> v3 priv
```

2. 创建一个 SNMP 用户。

添加一个具有身份验证和隐私的 SNMPv3 用户。请将占位符替换为您所需的值：

```
snmp-server user <your_user> <your_group> v3 auth md5 <auth_password> priv aes 128 <priv_password>
```

3. 保存配置：

```
write memory
```

如需更多详细信息或特定型号的说明，您可以参考外部 Cisco SNMP 配置教程。不过，本指南提供了启用 SNMP 监控的基本步骤。

配置 Zabbix 前端

在 Zabbix 前端中创建主机

1. 登录 Zabbix 前端。

2. 添加新主机。

导航到 *Data collection* > *Hosts*，然后点击 *Create host*。

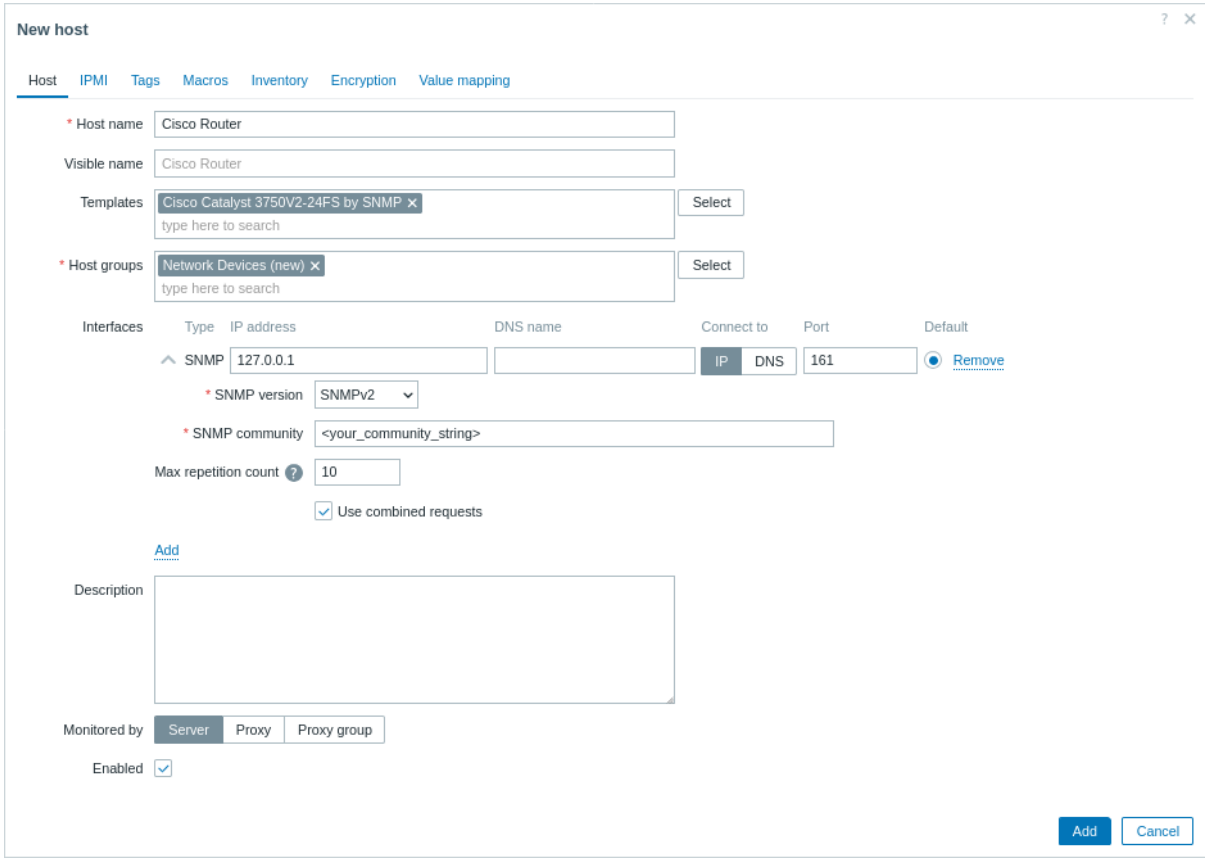
- *Host name*: 为您的设备输入名称（例如，“Cisco Router”）。
- Host groups: 选择现有组，或创建一个新组，例如 “Network Devices”。
- Interfaces:
  - 在 Interfaces 下点击 *Add*。
  - 选择 *SNMP* 作为接口类型。
  - 输入您的 Cisco 路由器的 IP 地址或 DNS 名称。

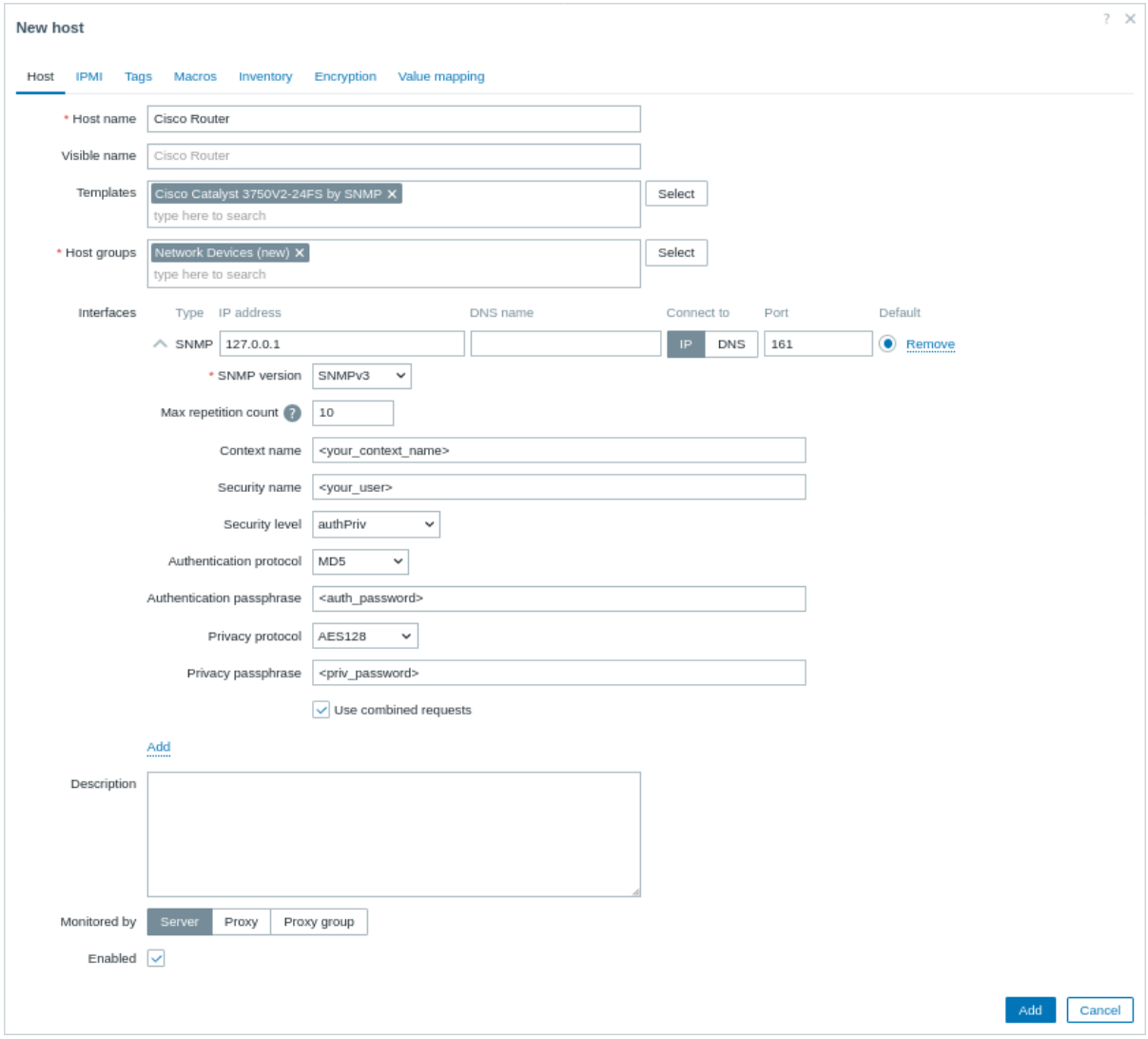
- 设置默认 SNMP 端口（通常为 161）。
- 使用下拉菜单选择合适的 SNMP 版本（例如，SNMPv2）。
- 对于 SNMPv1/v2，在 *SNMP community* 字段中输入 community 字符串。对于 SNMPv3，系统会提示输入其他凭据（*Context name*、*Security name* 和 *Security level* 等）。

3. 关联模板

在 *Templates* 字段中，选择最适合您设备的 SNMP 模板。Zabbix 为许多设备系列提供了一系列预置的SNMP templates。例如，如果您正在监控 Cisco 设备，请选择与该设备操作系统或型号相对应的模板（例如 Cisco IOS SNMP 或 Cisco Catalyst 3750<device model> SNMP）。

4. 点击 *Add* 保存主机。



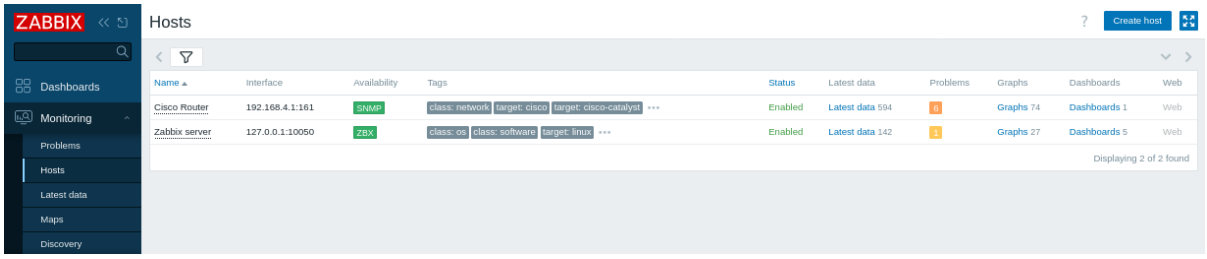


查看已收集的监控指标

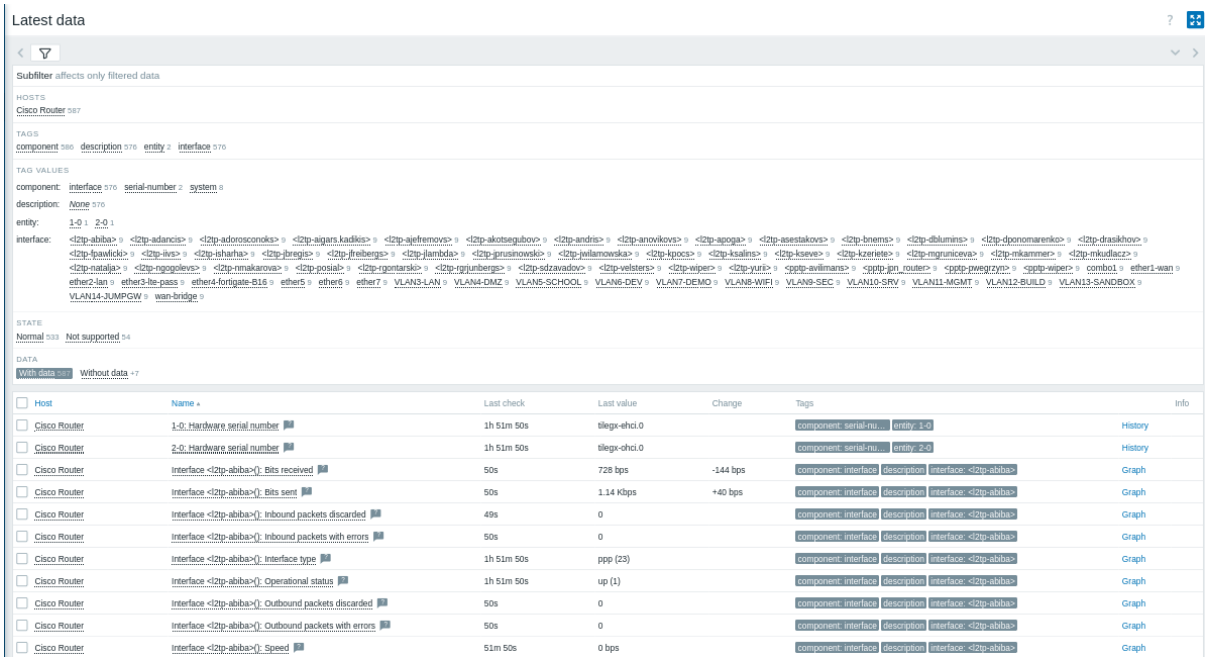
恭喜！Zabbix 现在已配置完成，可以监控您的网络设备。

最新数据：

- 在 Zabbix 前端中，导航到 Monitoring > Latest data。



- 选择您的”Cisco Router” 主机（或已发现的主机），以查看硬件和网络运行时间、ICMP 丢包、ping 和响应时间等监控指标。



- 图形和屏幕：

要可视化性能数据，请单击 SNMP 监控项旁边的 *Graphs*，以查看详细指标。

接下来，您可以：

- 添加自定义 SNMP 监控项，以监控更多指标。
- 设置问题告警，以接收潜在问题的通知。

创建 SNMP 监控项

一旦主机设置完成，您就可以创建监控项来监控特定指标。注意：如果您使用的是模板，这一步是可选的，因为模板中已经包含默认的监控项集合。

1. 确定 SNMP OID：

使用 `snmpwalk` 命令列出设备上可用的 OID。例如：

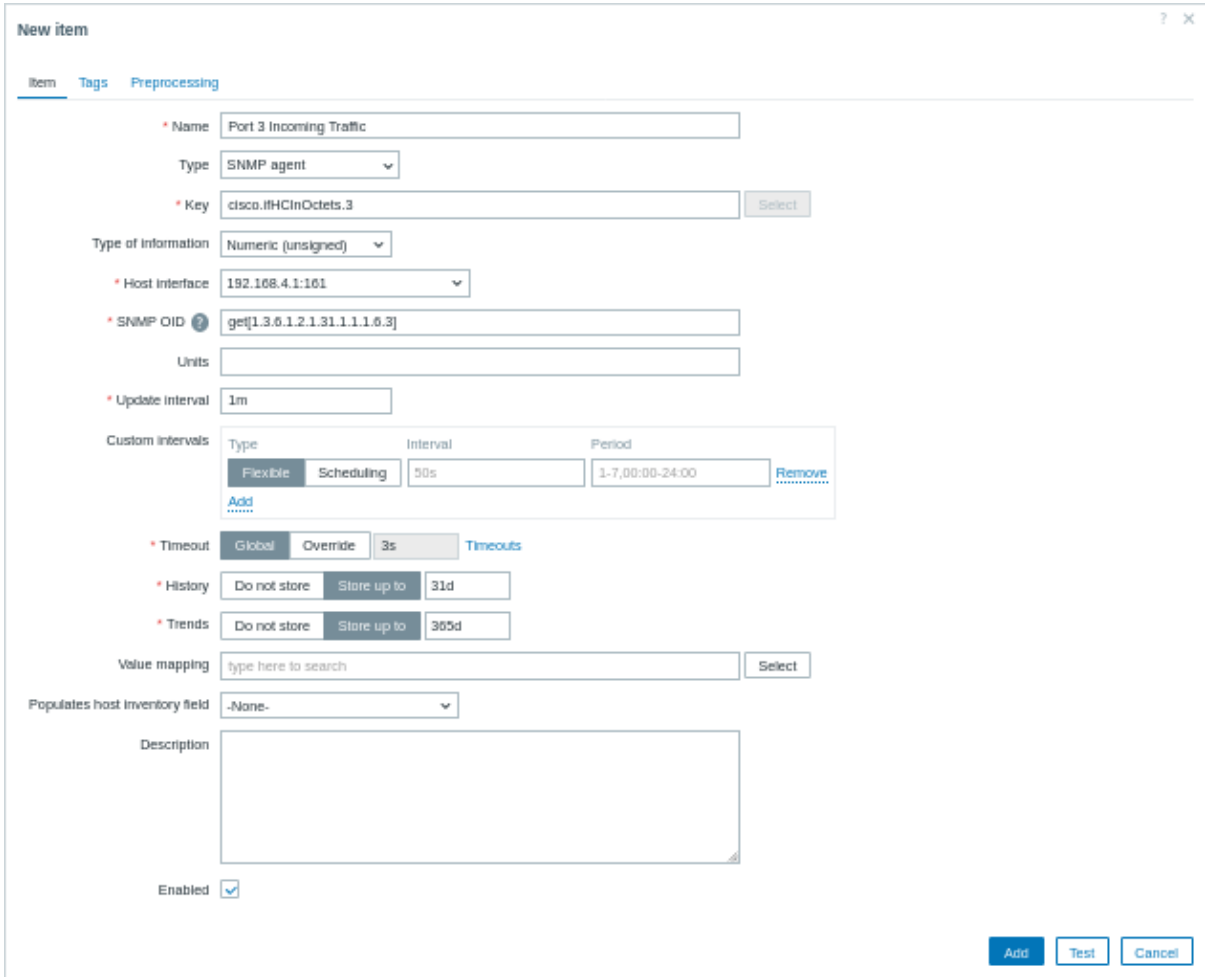
```
snmpwalk -v 2c -c <your_community_string> <device_IP> .
```

找到您希望监控的指标对应的 OID（例如，IF-MIB::ifHCInOctets.3 表示端口 3 的入站流量）。要获取数值型 OID，您可以使用：

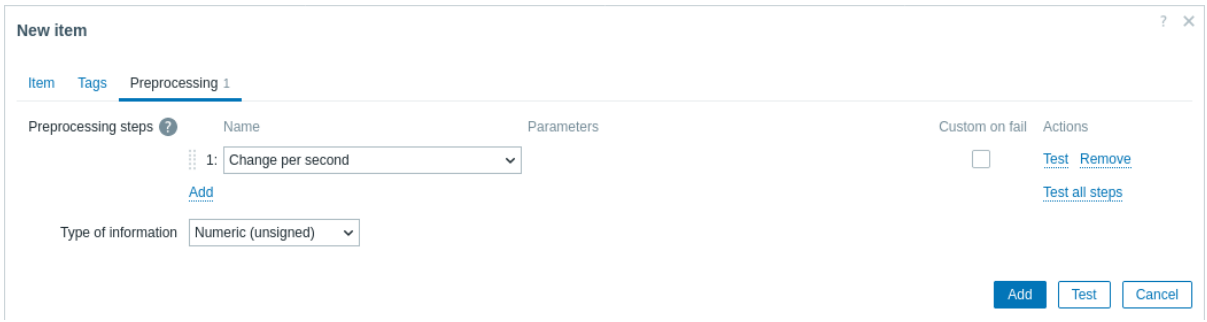
```
snmpget -v 2c -c <your_community_string> -On <device_IP> IF-MIB::ifHCInOctets.3
```

2. 创建 SNMP 监控项：

- 导航到 数据采集 > 主机，然后在您的 SNMP 主机上点击 监控项选项卡，并点击 创建监控项。
- 名称：输入一个描述性名称（例如，“端口 3 入站流量”）。
- 类型：选择 *SNMP agent*。
- 键值：提供一个有意义的键值（例如，`cisco.ifHCInOctets.3`）。
- 主机接口：确保已选择 SNMP 接口。
- *SNMP OID*：使用支持的格式之一输入 OID，例如：
  - `get[1.3.6.1.2.1.31.1.1.1.6.3]` 用于获取单个值；
  - `walk[1.3.6.1.2.1.31.1.1.1.6.3]` 用于异步检索一个值子树。



- 预处理（如需要）：如果该监控项返回的是累积计数器（例如接口流量），请切换到 预处理选项卡，并添加类似 “每秒变化” 的预处理步骤来计算速率。



要在一次 SNMP 事务中检索多个值，您可以使用语法 `walk[OID1,OID2,...]` 指定多个 OID。

在数字 OID 和 MIB 名称之间转换 OID

在使用 SNMP 时，您可能需要在数字 OID 和其对应的 MIB 名称之间进行转换。
这种转换有助于更轻松地识别和排查指标问题。

- 将 MIB 名称转换为数字 OID：使用带有 `-On` 选项的 `snmptranslate` 命令。
  例如，要将 MIB 名称 `IF-MIB::ifHCInOctets.3` 转换为其数字 OID，请运行：

```
snmptranslate -On IF-MIB::ifHCInOctets.3
```

该命令可能输出：

```
.1.3.6.1.2.1.31.1.1.1.6.3
```

- 将数字 OID 转换为其 MIB 名称：使用带有 `-IR`（或 `-m ALL`）选项的 `snmptranslate` 命令来反向转换。
  例如，要将数字 OID `.1.3.6.1.2.1.31.1.1.1.6.3` 转回其 MIB 名称，请运行：

```
snmptranslate -IR -On .1.3.6.1.2.1.31.1.1.1.6.3
```

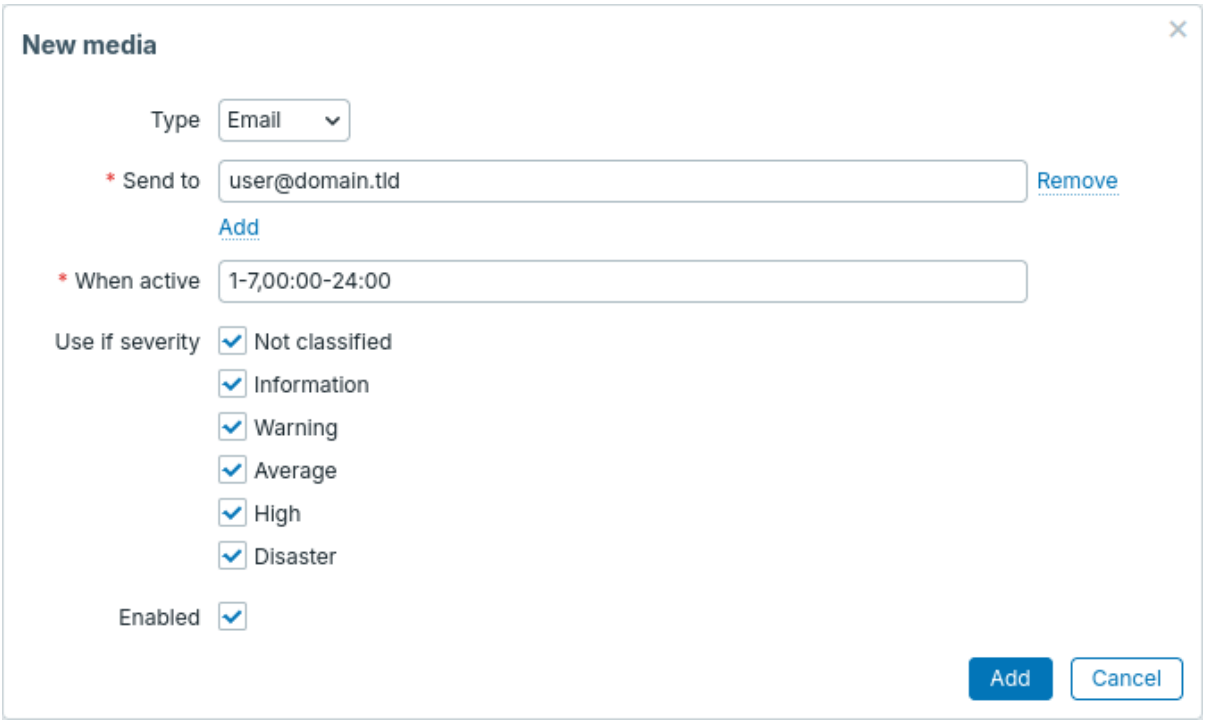
该命令可能输出：

```
IF-MIB::ifHCInOctets.3
```

设置问题告警

本指南提供发送电子邮件告警的基本配置步骤。

1. 导航到用户设置 > 个人资料，切换到 媒体选项卡，并添加您的电子邮件。



2. 按照接收问题通知指南进行操作。

下次 Zabbix 检测到问题时，您应该会通过电子邮件收到告警。

测试您的配置

为确保 Zabbix 能正确检测网络性能问题，请通过提高 ICMP ping 响应时间阈值来模拟真实问题。

1. 在 Zabbix 中打开您的"Cisco Router" 主机配置。

2. 导航到 *Macros* 选项卡，并选择 *Inherited and host macros*。

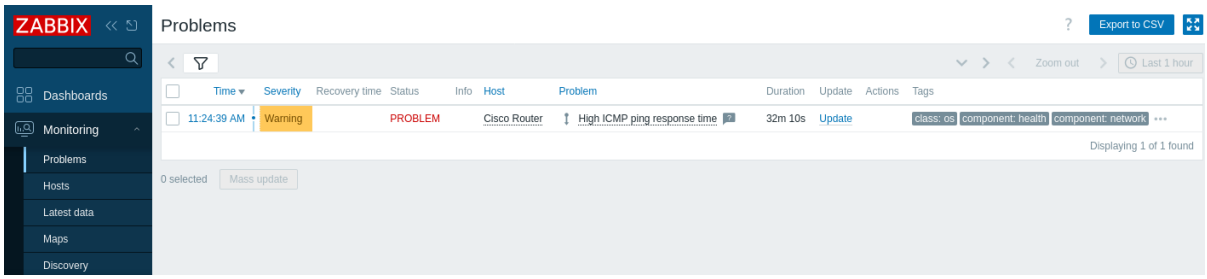
3. 找到 `{$ICMP_RESPONSE_TIME_WARN}` 宏（或类似的响应时间阈值宏）。

4. 将其设置为一个非常低的值（例如 0.001），以便在 ping 响应超过该值时触发告警。

5. 点击 *Update* 应用更改。

6. 等待片刻，让 Zabbix 检测到模拟的问题。

7. 导航到 *Monitoring* > *Problems*，确认是否出现告警（例如"High ICMP ping response time"）。



如果已配置告警，您还应收到问题告警。

8. 将宏值恢复为原始设置，然后点击 *Update* 保存更改。

9. 确认问题已解决，并且已从 *Problems* 部分消失。

SNMP 监控故障排查

如果您注意到 Zabbix 前端中的 SNMP 图标显示为红色，或者没有采集到数据，请尝试以下步骤：

1. 检查 SNMP 连通性。

对于 SNMPv2，请从您的 Zabbix 服务器运行以下命令：

```
snmpwalk -v 2c -c <community_string> <device_IP> .
```

此命令用于验证设备是否会响应 SNMP 查询。

对于 SNMPv3，请包含相应的 SNMPv3 凭据：

```
snmpwalk -v3 -u <your_user> -l authPriv -a MD5 -A <auth_password> -x AES -X <priv_password> <device_IP> .
```

这将验证 SNMPv3 凭据是否正确，以及设备是否能够安全响应。

2. 确保已按照先决条件 安装并启用 MIB 文件。为确保这一点，在查询网络设备时，以下命令不应返回错误：

```
snmpwalk -v 2c -c <your_community_string> <device_IP> ifInOctets
```

这应返回已翻译的 OID，且不应出现错误。

3. 确认 Zabbix 中配置的 SNMP 版本和凭据与设备上设置的一致。例如，检查 Zabbix 主机配置中的 SNMP 设置，并将其与设备配置进行比对。在 Cisco 设备上，您可以通过运行以下命令检查 SNMP 设置：

```
show running-config | include snmp
```

这可确保 community string（用于 SNMPv2）或 SNMPv3 用户详细信息正确无误。

4. 验证网络设备上是否已正确启用 SNMP。在 Cisco 路由器上，登录控制台并运行：

```
show running-config | include snmp
```

此命令会显示当前生效的 SNMP 配置，并有助于确认 SNMP 是否已正确配置。

5. 确保没有防火墙或网络问题阻止 Zabbix 服务器与设备之间的 SNMP 流量（通常为 161 端口）。您可以使用以下命令测试连通性：

nc -zv <device_IP> 161

`nc -zv` 会检查设备上的 161 端口是否已打开并处于监听状态。

另外，如果您在 Ubuntu 上使用 UFW，请检查防火墙状态：

```
sudo ufw status
```

或者，对于 iptables：

```
sudo iptables -L -n
```

6. 查看 Zabbix 服务器日志文件中是否有任何与 SNMP 相关的错误，以帮助定位问题：

```
tail -f /tmp/zabbix_server.log
```

`tail -f` 允许您实时监控日志更新。

另请参阅：

- 创建监控项 - 了解如何添加额外的指标。
- SNMP agent - 有关使用 Zabbix 进行 SNMP 监控的更多信息。
- 网络设备标准化模板 - 有关可用 SNMP 模板的信息。
- 发现 SNMP OID - 有关在交换机上进行 SNMP 发现的更多信息。
- 配置网络发现规则 - 有关如何配置 Zabbix 用于发现主机和服务的网络发现规则的更多信息。

使用主动检查监控 **Windows** 事件日志

简介

本指南说明如何使用主动检查通过 Zabbix 监控 Windows 事件日志。借助 Zabbix 的 Windows 特定监控项键，您可以实时收集和分析关键事件（例如登录失败尝试、系统错误等）。

本指南适用对象

本指南面向希望监控 Windows 事件日志的新 Zabbix 用户和网络管理员。有关高级配置选项，请参阅Windows 特定监控项键 文档。

先决条件

在继续阅读本指南之前，您需要根据操作系统的说明 下载并安装 Zabbix 服务器和 Zabbix 前端。您还需要在要监控的 Windows 计算机上 下载并安装 Zabbix agent。

配置 Zabbix agent 以监控 Windows 事件日志

1. 在 Windows 主机上打开 Zabbix agent 配置文件，并确保ServerActive 参数设置为 Zabbix 服务器的 IP 地址，且Hostname 参数与将在Zabbix 前端中定义的主机名一致。这样，agent 就可以从指定的 Zabbix 服务器请求其主机的主动检查。例如：

```
ServerActive=192.0.2.1
Hostname=MyWindowsHost
```

2. 重启 Zabbix agent 服务以应用更改：

```
net stop "Zabbix Agent" && net start "Zabbix Agent"
```

3. 检查 Windows 主机是否正在运行：

- 确保 Zabbix agent 服务正在 Windows 主机上运行。
- 检查 Windows 主机是否可以通过端口 10051 连接到 Zabbix 服务器。要从 Windows 主机测试连接，请打开 PowerShell 并运行以下命令：

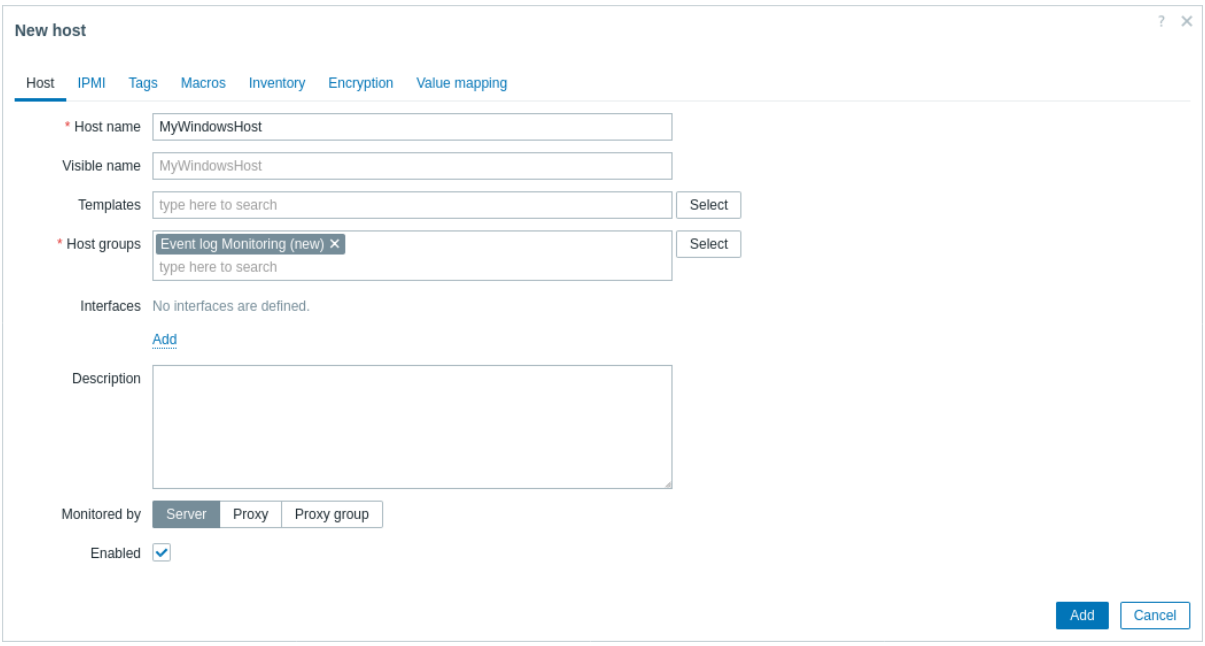
```
Test-NetConnection -ComputerName <Zabbix-server-IP> -Port 10051
```

配置 Zabbix 前端

1. 导航到 *Data collection* > *Hosts* 并创建主机：

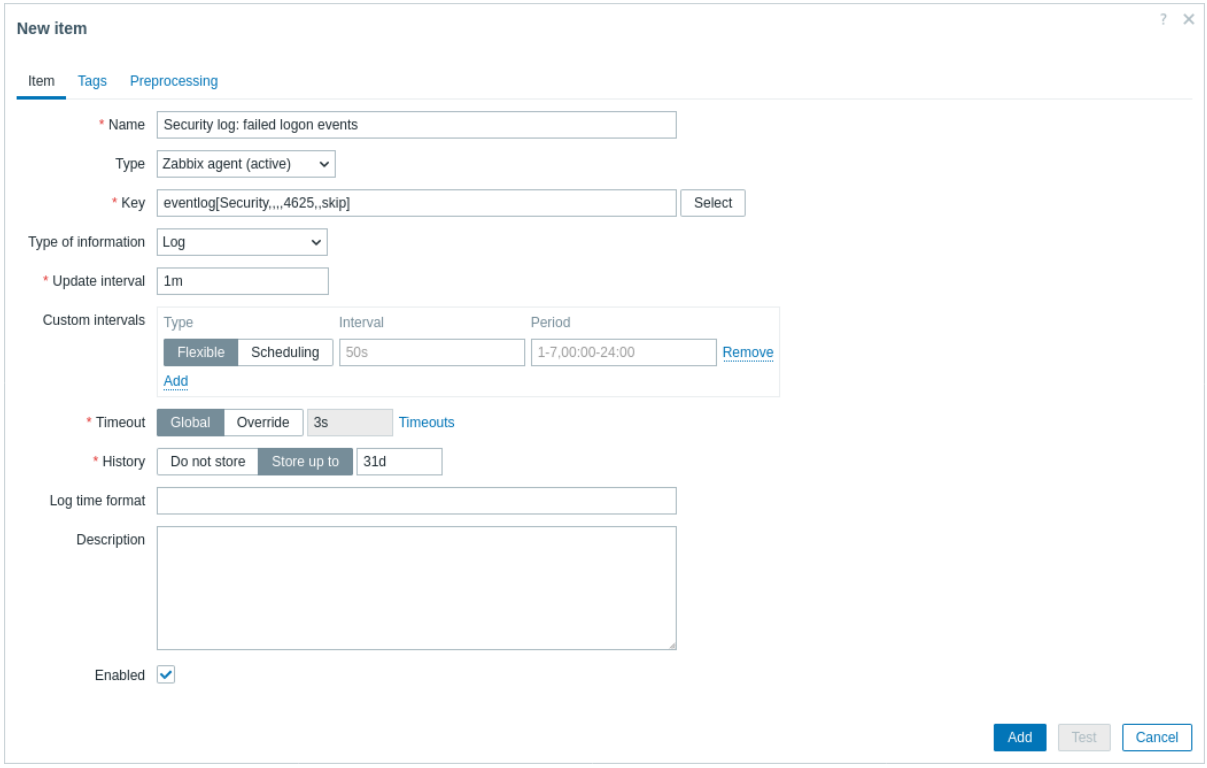
- 在 *Host name* 字段中，输入主机名称（例如，“MyWindowsHost”）。
- 在 *Host groups* 字段中，输入或选择一个主机组（例如，“Event log Monitoring”）。
- 单击 *Add* 保存已配置的主机。

Note:
在 *Templates* 字段中，你可以添加 “Windows by Zabbix agent active” 模板，以便通过观察同一主机上的其他主动监控项是否正在更新来帮助排查问题。



2. 使用以下参数创建一个新的监控项：

- 在 *Name* 字段中，输入一个描述性的监控项名称（例如，“Security log: failed logon events”）。
- 在 *Type* 下拉列表中，选择 “Zabbix agent (active)”（Event log 监控所必需）。
- 在 *Key* 字段中，使用eventlog 监控项键。例如，要监控 Security log 中的登录失败尝试（Event ID: 4625），并忽略早于该监控项上次检查时间的条目（使用 `skip` 参数），请输入以下监控项键：`eventlog[Security,,,,4625,,skip]`
- 在 *Type of information* 下拉列表中，选择 “Log”。



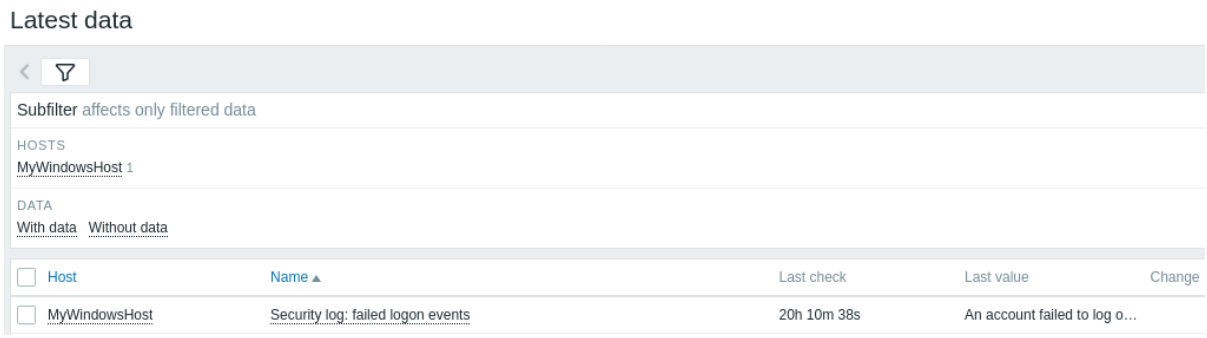
3. 单击 *Add* 保存该监控项。

测试并查看已收集的监控项

恭喜！Zabbix 现在已设置为收集你的 Windows 事件日志。要验证事件日志是否正在被收集，你可以通过注销 Windows 账户并使用错误的凭据尝试登录，来测试”Security log: failed logon events” 监控项。

然后，在 Zabbix 前端中查看已收集的日志：

1. 在 Zabbix 前端中导航到 *Monitoring* > *Latest data*。



2. 在 *Name* 字段中按你的”MyWindowsHost” 主机进行筛选。

3. 点击 *History* 查看已记录的日志值。

MyWindowsHost: Security log: failed logon events

Timestamp	Local time	Source	Severity	Event ID	Value
2025-04-07 11:13:11 PM	2025-04-07 11:12:19 PM	Microsoft-Windows-Security-Auditing	Failure Audit	4625	An account failed to log on. Subject: Security ID: NT AUTHORITY\SYSTEM Account Name: DESKTOP-NNEM26U$ Account Domain: WORKGROUP Logon ID: 0x3E7 Logon Type: 2 Account For Which Logon Failed: Security ID: NULL SID Account Name: Admin Account Domain: DESKTOP-NNEM26U Failure Information: Failure Reason: Unknown user name or bad password. Status: 0xC000006D Sub Status: 0xC000006A

4. 如果没有日志值，请继续查看本指南的Troubleshooting 部分。

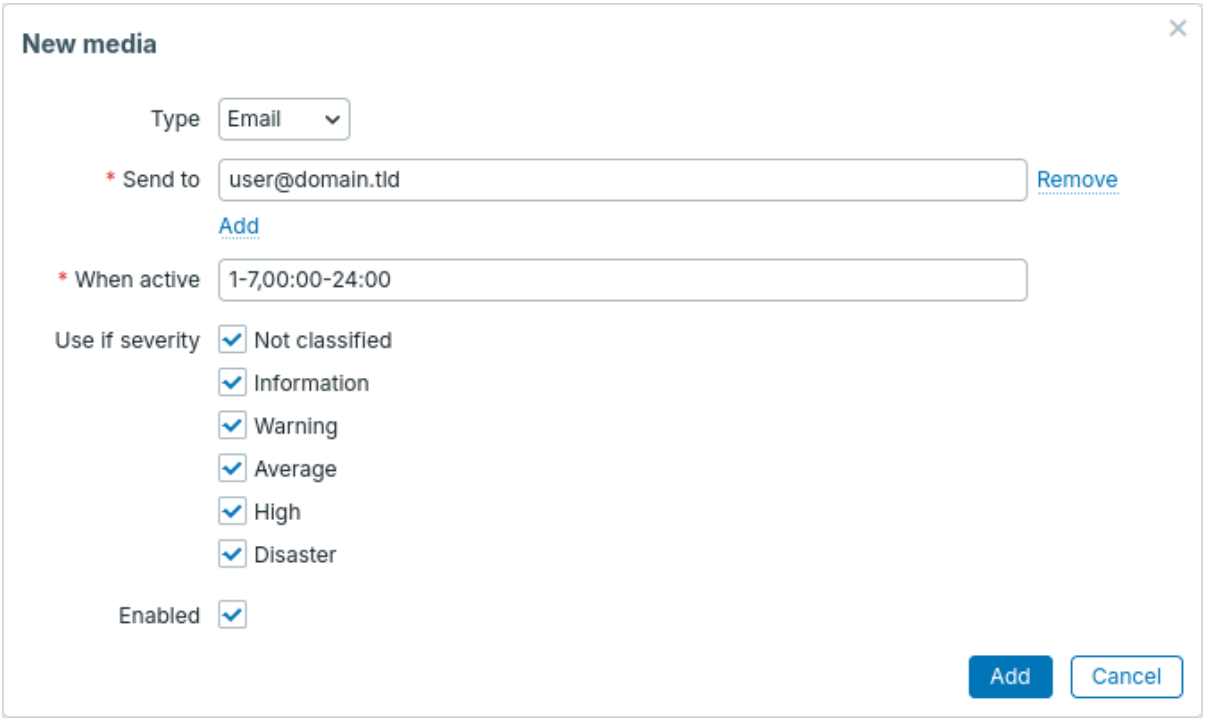
设置问题告警

本指南提供了发送电子邮件告警的基本配置步骤。

1. 导航到数据采集 > 主机，去定义一个触发器，当你的事件日志监控项记录到你关心的模式时，该触发器就会触发。例如，要捕获 Security 日志中的登录失败尝试，请使用find() 函数：

find(/MyWindowsHost/eventlog[Security,,,4625,skip],10m,"like","Logon failed")

2. 导航到用户设置 > 配置文件，切换到 媒体选项卡，并添加你的电子邮件。



3. 按照接收问题通知指南进行操作。

下次当 Zabbix 检测到问题时，你应该会通过电子邮件收到告警。

故障排除

如果在收集或查看 Windows 事件日志时遇到问题，请使用以下提示来识别并解决常见问题：

1. 在 Zabbix 服务器（Linux）上，使用以下命令列出你的 iptables 规则：

```
sudo iptables -L -n
```

并确认是否存在针对 TCP 端口 10051 的 ACCEPT 规则。

2. 确保你的 `eventlog[...]` 键使用的是完全一致的日志名称（区分大小写）、事件 ID、模式（例如 skip）以及其他参数，且与Windows-specific item keys 中显示的内容完全一致。

另请参阅：

- 创建监控项 - 了解如何添加更多指标。
- Microsoft Windows 上的 Zabbix agent - 详细的安装说明。
- 使用 Zabbix agent 监控 Windows - 一份全面的指南，介绍如何使用 Zabbix agent 为 Windows 机器设置基础监控。
- Windows 特定的监控项键值 - 关于 Zabbix agent 支持的 Windows 特定监控项键值的详细信息，包括用于事件日志监控的键值。
- 日志文件监控 - 关于如何配置 Zabbix 以集中监控和分析日志文件的说明，也适用于 Windows 事件日志。

安装：其他指南

如果您正在寻找安装 Zabbix 的最简单方法，请参阅安装和初始步骤。

以下指南介绍了安装 Zabbix 或其组件的其他方法，或特定于系统的安装方法：

- 从容器安装
- 从源代码安装
- 从软件包安装的注意事项
- 使用 Windows MSI 安装 Zabbix agent
- 在 Windows 上构建 Zabbix agent 2
- 在 Windows 上构建 Zabbix agent
- 从 PKG 安装 macOS agent
- IBM AIX 上的 Zabbix agent
- 在 macOS 上构建 Zabbix agent
- Zabbix appliance
- Zabbix 服务器 AWS 部署指南
- Zabbix proxy AWS 部署指南
- Zabbix 服务器 Azure 部署指南
- Zabbix proxy Azure 部署指南
- Google Cloud 部署指南

从容器安装

概述

本页面介绍如何使用 Docker 或 Docker Compose 部署 Zabbix。

Docker Compose 是部署 Zabbix 的最快方式。它会读取配置文件，并按正确顺序自动启动构成完整 Zabbix 环境的所有容器。

Docker（手动部署） 通过逐步部署每个组件来实现相同的结果。

前提条件

开始之前，请确保系统中已安装 Docker（1.12.0 或更高版本）。如果尚未安装，请按照 Docker 安装指南 进行操作。对于 Docker Compose 部署，还需要 Docker Compose（2.24.0 或更高版本）。

某些 Zabbix 组件要求在运行 Docker 的主机上开放特定端口（例如，Zabbix 服务器使用的 10051/TCP，以及 SNMP 陷阱使用的 162/UDP）。有关 Zabbix 组件所使用端口的完整列表，请参阅要求。对于 Zabbix 服务器和 agent，可以在相应容器中设置 `ZBX_LISTENPORT` 环境变量来更改默认端口。

可用的 Docker 镜像

Zabbix 为每个 Zabbix 组件提供了一个 Docker 镜像，所有镜像均发布在 Docker Hub 上。每个镜像用于创建运行相应组件的容器。

Zabbix 组件	Docker 镜像
Agent	zabbix/zabbix-agent

Zabbix 组件	Docker 镜像
Agent 2	zabbix/zabbix-agent2
服务器（MySQL）	zabbix/zabbix-server-mysql
服务器（PostgreSQL）	zabbix/zabbix-server-pgsql
前端（Apache + MySQL）	zabbix/zabbix-web-apache-mysql
前端（Apache + PostgreSQL）	zabbix/zabbix-web-apache-pgsql
前端（Nginx + MySQL）	zabbix/zabbix-web-nginx-mysql
前端（Nginx + PostgreSQL）	zabbix/zabbix-web-nginx-pgsql
Proxy（SQLite3）	zabbix/zabbix-proxy-sqlite3
Proxy（MySQL）	zabbix/zabbix-proxy-mysql
Java 网关	zabbix/zabbix-java-gateway
Web 服务	zabbix/zabbix-web-service
SNMP 陷阱	zabbix/zabbix-snmptraps
MCP 服务器	zabbix/zabbix-mcp-server

Note:
要使用SNMP 陷阱，SNMP 陷阱容器必须与 Zabbix 服务器或 proxy 容器共享一个卷（请参阅 Docker Hub 上的如何使用此镜像以及下面的示例）。

Docker Hub 上的所有 Zabbix 组件镜像均基于受支持的操作系统的最新主要版本。底层操作系统镜像更新后，这些镜像会自动重新构建。

镜像标签

每个镜像都支持使用标签来选择基础操作系统和 Zabbix 版本：

```
zabbix/<image>:<os>-<version>
```

支持的 <os> 值：

- alpine - Alpine Linux
- ubuntu - Ubuntu
- centos - CentOS Stream
- ol - Oracle Linux
- ltsc2022 - Windows 11 LTSC 2022（仅 Zabbix agent）

支持的 <version> 值：

- latest - Alpine Linux 上最新的稳定版 Zabbix 版本
- <os>-latest - 所选操作系统上最新的稳定版 Zabbix 版本
- <os>-trunk - 所选操作系统上最新的开发版（nightly）构建
- <os>-X.X-latest - 所选操作系统上特定 Zabbix 主版本的最新 Zabbix 次版本发布
- <os>-X.X.* - 所选操作系统上的特定 Zabbix 次版本发布

示例：

```
#### Alpine Linux 上最新的稳定版 Zabbix proxy (SQLite3):
docker pull zabbix/zabbix-proxy-sqlite3:latest

#### Ubuntu 上最新的稳定版 Zabbix proxy (SQLite3):
docker pull zabbix/zabbix-proxy-sqlite3:ubuntu-latest

#### Ubuntu 上 Zabbix 服务器 (MySQL) 的最新开发版 (nightly) 构建:
docker pull zabbix/zabbix-server-mysql:ubuntu-trunk

#### Alpine Linux 上 Zabbix 服务器 (MySQL) 的最新 8.0 次版本发布:
docker pull zabbix/zabbix-server-mysql:alpine-8.0-latest

#### Alpine Linux 上 Zabbix 服务器 (MySQL) 的 8.0.1 版本:
docker pull zabbix/zabbix-server-mysql:alpine-8.0.1
```

Docker Compose

Docker Compose 是部署 Zabbix 的最快方式。它会读取配置文件（Compose 文件）并处理整个设置过程——拉取 Docker 镜像、在容器之间创建内部网络、设置存储、初始化数据库，以及按正确顺序启动所有组件。

官方 Zabbix Dockerfiles 仓库提供了可直接使用的 Docker Compose 文件，以及基于 .env 的配置系统，支持不同的操作系统、数据库后端和 Zabbix 组件配置。

1. 克隆仓库，进入仓库目录，并切换到 8.0 版本：

```
git clone https://github.com/zabbix/zabbix-docker.git
cd zabbix-docker
git checkout 8.0
```

2. （可选）使用环境变量自定义部署。默认设置可以跳过此步骤。

3. 部署默认设置——Zabbix 服务器、Web 前端（运行于 NGINX）以及数据库后端（MySQL 或 PostgreSQL），每个组件都在 Alpine Linux 上的独立容器中运行。

```
#### 使用 MySQL 作为数据库：
docker compose -f ./compose.yaml up -d

#### 使用 PostgreSQL 作为数据库：
docker compose -f ./compose_pgsql.yaml up -d
```

容器启动并运行后（通常需要 1-3 分钟），Zabbix 服务器将启动，Web 前端也可供登录。

> **Note:**
> 使用 `docker compose ps` 检查容器状态。除 `zabbix-docker-server-db-init-1` 容器外，所有容器的状态都应为 Up。如果任何容器显示 Exited，请使用 `docker logs -f <container-name>` 检查其日志中的错误。

环境变量

可以使用环境变量自定义 Docker Compose 行为和 Zabbix 组件配置。

Compose 级变量（在 `.env` 文件中定义）用于控制所使用的 Docker 镜像、端口和网络 IP 范围。您可以在 `docker compose` 命令前以内联方式使用这些变量，也可以编辑 `.env` 文件。

例如，以下命令使用基于 Ubuntu 的镜像（而不是默认的 Alpine Linux）部署完整的多容器 Zabbix 环境，并通过 NGINX 使用自定义的 HTTP（8282）和 HTTPS（8443）端口公开 Web 界面：

```
OS=ubuntu \
ZABBIX_WEB_NGINX_HTTP_PORT=8282 \
ZABBIX_WEB_NGINX_HTTPS_PORT=8443 \
docker compose -f ./compose.yaml up -d
```

组件级变量（在 `env_vars/.env_<component>` 文件中定义）用于控制 Zabbix 组件配置。在运行 `docker compose` 命令前，编辑相应的 `.env_<component>` 文件。

例如，您可以通过编辑以下变量调整 Zabbix 服务器被动轮询器数量和缓存设置，并配置 Web 界面的时区：

```
#### env_vars/.env_srv
ZBX_STARTPOLLERS=20
ZBX_CACHESIZE=64M
ZBX_HISTORYCACHESIZE=32M

#### env_vars/.env_web
PHP_TZ=Europe/Paris
```

> **Note:**
> 组件级环境变量对应于 Zabbix 组件配置参数，但采用了不同的命名方式（例如，`ZBX_STARTPOLLERS` 对应于`StartPollers`）。某些变量是 Docker 专用的，并且某些配置参数无法更改（例如，`PIDFile` 和 `LogType`）。使用环境变量时，请参阅 Docker Hub 上每个组件的Docker 镜像说明中的 环境变量部分。

卷

Docker Compose 将持久化数据存储在与 Compose 文件一同创建的 `zbx_env/` 目录中。
该目录可在容器重启和更新后保留组件数据。

`zbx_env/` 的内容已针对每个组件的镜像预先定义。例如：

- `zbx_env/etc/` - 由 Zabbix Web 界面容器 使用
- `zbx_env/usr/` 和 `zbx_env/var/` - 由 Zabbix 服务器容器 使用

有关每个卷的更多信息，请参见 Docker Hub 上各组件Docker 镜像 说明中的 *Allowed volumes* 部分。

Makefile

Zabbix Dockerfiles 仓库还提供了一个 Makefile，可作为执行常用 Docker Compose 任务的快捷方式。无需使用完整的 `docker compose` 命令，您可以使用更简短的 `make` 命令（运行 `make help` 可查看可用选项）：

```
#### 默认部署（Zabbix 服务器、前端、MySQL，全部运行在 Alpine Linux 上）
make up

#### 自定义部署（Zabbix 服务器、使用自定义 NGINX 端口的前端、PostgreSQL，全部运行在 Ubuntu 上）
make up \
  OS=ubuntu \
  DB=pgsql \
  ZABBIX_WEB_NGINX_HTTP_PORT=8282 \
  ZABBIX_WEB_NGINX_HTTPS_PORT=8443
```

Attention:
停止或删除容器时，请始终指定部署时使用的相同数据库类型（例如：`make down DB=pgsql`）。

默认情况下，`make up` 仅启动最小服务集（Zabbix 服务器、前端和数据库）。这样可以使部署保持轻量，并避免启动可能不需要的组件。如需包含其他组件，可以使用以下 Compose 配置文件：

```
#### 默认部署 + Zabbix agent、Java 网关、Web 服务和 SNMP 陷阱：
make up COMPOSE_PROFILES=full

#### COMPOSE_PROFILES=full + Zabbix proxy（MySQL 和 SQLite3）：
make up COMPOSE_PROFILES=all
```

Docker（手动部署）

当你希望逐步部署 Zabbix、运行各个独立组件、将其与现有环境集成，或使用 Podman 等替代容器运行时环境时，请使用手动部署。

例如，要部署一个 Zabbix proxy，请运行以下命令：

```
docker run --name zabbix-proxy-sqlite3 \
  -e ZBX_SERVER_HOST=192.0.2.1 \
  -e ZBX_PROXYMODE=0 \
  -e ZBX_HOSTNAME=zabbix-proxy-sqlite3 \
  -v zabbix-proxy-data:/var/lib/zabbix/db_data \
  --init \
  -d \
  zabbix/zabbix-proxy-sqlite3:alpine-8.0-latest
```

此命令会：

- 拉取 `zabbix/zabbix-proxy-sqlite3:alpine-8.0-latest` Docker 镜像。
- 基于该 Docker 镜像创建并启动一个 `zabbix-proxy-sqlite3` 容器（使用 `--init` 和 `-d` 标志）。
- 通过 `ZBX_SERVER_HOST` 环境变量配置 Zabbix proxy 的`Server` 参数。该参数指定 Zabbix 服务器的 IP 地址，proxy 将从该服务器获取配置数据，并向其发送收集到的数据。请注意，active proxy 运行所需的其他变量使用默认值，因此可以省略。

Note:
环境变量与 Zabbix 组件配置参数之间采用不同的命名方式对应（例如，`ZBX_SERVER_HOST` 对应`Server`）。某些变量是 Docker 特有的，某些配置参数无法更改（例如，`PIDFile` 和 `LogType`）。使用环境变量时，请参考 Docker Hub 上各组件Docker 镜像说明中的 *Environment variables* 部分。

- 将 Docker 管理的存储位置（例如 `/var/lib/docker/volumes/zabbix-proxy-data/`）连接到容器的 `/var/lib/zabbix` 目录，以便即使容器被移除，Zabbix proxy 数据也能永久保存。

Note:
有关每个卷的更多信息，请参阅 Docker Hub 上各组件Docker 镜像 说明中的 *Allowed volumes* 部分。

部署 Zabbix proxy 容器后，你可以继续在 Zabbix 前端中添加 proxy，并配置主机由该 proxy 进行监控。

下面的示例涵盖另外三种部署场景：

- Zabbix 服务器（MySQL）与 Java 网关
- Zabbix 服务器（PostgreSQL）与 SNMP trap
- Zabbix 服务器（MySQL）与 RHEL 8-10 上的 Java 网关

更多示例，请参阅 Docker Hub 上各组件的Docker 镜像 说明。

部署 Zabbix 服务器（MySQL）和 Java 网关

本示例介绍如何部署使用 MySQL 后端的 Zabbix 服务器、用于JMX 监控 的 Java 网关，以及基于 NGINX 的 Web 界面。

1. 创建专用 Docker 网络，使所有 Zabbix 组件容器可以通过容器名称相互访问：

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

2. 运行一个空的 MySQL 服务器容器（将 zabbix_pwd 和 root_pwd 替换为强密码，并在后续步骤中使用这些密码）：

```
docker run --name mysql-server -t \
  -e MYSQL_DATABASE="zabbix" \
  -e MYSQL_USER="zabbix" \
  -e MYSQL_PASSWORD="zabbix_pwd" \
  -e MYSQL_ROOT_PASSWORD="root_pwd" \
  --network=zabbix-net \
  --restart unless-stopped \
  -d mysql:8.4-oracle \
  --character-set-server=utf8mb4 --collation-server=utf8mb4_bin
```

3. 查看 MySQL 日志，并等待 MySQL 准备好接受连接（使用 Ctrl+C 退出日志）：

```
docker logs -f mysql-server

#### ... [Server] /usr/sbin/mysqld: ready for connections. ...
```

4. 运行 Zabbix Java 网关容器：

```
docker run --name zabbix-java-gateway -t \
  --network=zabbix-net \
  --restart unless-stopped \
  -d zabbix/zabbix-java-gateway:alpine-8.0-latest
```

5. 在 MySQL 服务器容器中启用 log_bin_trust_function_creators。此设置会暂时放宽对存储函数的安全限制；在后续步骤中将禁用该设置。启用后，Zabbix 数据库用户无需 SUPER 权限即可创建存储函数。当启用二进制日志记录时，MySQL 要求创建存储函数的用户具有该权限（MySQL 8.0 起默认启用二进制日志记录）。运行以下命令：

```
docker exec -it mysql-server mysql -u root -p'root_pwd' \
  -e "SET GLOBAL log_bin_trust_function_creators = 1;"
```

6. 运行 Zabbix 服务器容器，并将其连接到 Java 网关和 MySQL 服务器容器：

```
docker run --name zabbix-server-mysql -t \
  -e DB_SERVER_HOST="mysql-server" \
  -e MYSQL_DATABASE="zabbix" \
  -e MYSQL_USER="zabbix" \
  -e MYSQL_PASSWORD="zabbix_pwd" \
  -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
  --network=zabbix-net \
  -p 10051:10051 \
  --restart unless-stopped \
  --init \
  -d zabbix/zabbix-server-mysql:alpine-8.0-latest
```

7. 查看 Zabbix 服务器日志，并等待 Zabbix 服务器完成数据库架构初始化（使用 Ctrl+C 退出日志）：

```
docker logs -f zabbix-server-mysql

#### ... [info]: ** Creating 'zabbix' schema in MySQL
#### ... [info]: ** Database schema successfully created!
```

8. 在 MySQL 服务器容器中禁用 log_bin_trust_function_creators。此操作会重新启用禁止非 SUPER 用户创建存储函数的安全限制。运行以下命令：

```
docker exec -it mysql-server mysql -u root -p'root_pwd' \
  -e "SET GLOBAL log_bin_trust_function_creators = 0;"
```

9. 运行 Zabbix Web 界面容器，并将其连接到 Zabbix 服务器和 MySQL 服务器容器：

```
docker run --name zabbix-web-nginx-mysql -t \
  -e ZBX_SERVER_HOST="zabbix-server-mysql" \
  -e DB_SERVER_HOST="mysql-server" \
  -e MYSQL_DATABASE="zabbix" \
  -e MYSQL_USER="zabbix" \
  -e MYSQL_PASSWORD="zabbix_pwd" \
  -e PHP_TZ="Europe/Riga" \
  --network=zabbix-net \
  -p 80:8080 \
  --restart unless-stopped \
  -d zabbix/zabbix-web-nginx-mysql:alpine-8.0-latest
```

容器启动并运行后，Zabbix 服务器将启动，Web 界面可用于登录。

> **Note:**
> 使用 `docker ps` 检查容器状态。所有容器都应显示 `Up` 状态。如果任何容器显示 `Exited`，请使用 `docker logs -f <container-name>` 检查其日志中的错误。

部署带 SNMP 陷阱的 Zabbix 服务器（PostgreSQL）

本示例介绍如何部署使用 PostgreSQL 后端的 Zabbix 服务器、SNMP 陷阱，以及基于 NGINX 的 Web 界面。

1. 创建专用 Docker 网络，使所有 Zabbix 组件容器可以通过容器名称相互访问：

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

2. 运行一个空的 PostgreSQL 服务器容器（将 `zabbix_pwd` 替换为强密码，并在后续步骤中使用该密码）：

```
docker run --name postgres-server -t \
  -e POSTGRES_USER="zabbix" \
  -e POSTGRES_PASSWORD="zabbix_pwd" \
  -e POSTGRES_DB="zabbix" \
  --network=zabbix-net \
  --restart unless-stopped \
  -d postgres:latest
```

3. 运行 Zabbix SNMP 陷阱容器：

```
docker run --name zabbix-snmptraps -t \
  -v /zbx_instance/snmptraps:/var/lib/zabbix/snmptraps:rw \
  -v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
  --network=zabbix-net \
  -p 162:1162/udp \
  --restart unless-stopped \
  -d zabbix/zabbix-snmptraps:alpine-8.0-latest
```

4. 运行 Zabbix 服务器容器，并将其连接到 PostgreSQL 服务器和 SNMP 陷阱容器：

```
docker run --name zabbix-server-pgsql -t \
  -e DB_SERVER_HOST="postgres-server" \
  -e POSTGRES_USER="zabbix" \
  -e POSTGRES_PASSWORD="zabbix_pwd" \
  -e POSTGRES_DB="zabbix" \
  -e ZBX_ENABLE_SNMP_TRAPS="true" \
  --network=zabbix-net \
  -p 10051:10051 \
  --volumes-from zabbix-snmptraps \
  --restart unless-stopped \
  --init \
  -d zabbix/zabbix-server-pgsql:alpine-8.0-latest
```

5. 查看 Zabbix 服务器日志，并等待 Zabbix 服务器完成数据库架构初始化（使用 `Ctrl+C` 退出日志）：

```
docker logs -f zabbix-server-pgsql

#### ... [info]: ** Creating 'zabbix' schema in PostgreSQL
#### ... [info]: ** Database schema successfully created!
```

6. 运行 Zabbix Web 界面容器，并将其连接到 Zabbix 服务器和 PostgreSQL 服务器容器：

```
docker run --name zabbix-web-nginx-pgsql -t \
  -e ZBX_SERVER_HOST="zabbix-server-pgsql" \
  -e DB_SERVER_HOST="postgres-server" \
  -e POSTGRES_DB="zabbix" \
  -e POSTGRES_USER="zabbix" \
  -e POSTGRES_PASSWORD="zabbix_pwd" \
  -e PHP_TZ="Europe/Riga" \
  --network=zabbix-net \
  -p 443:8443 \
  -p 80:8080 \
  -v /etc/ssl/nginx:/etc/ssl/nginx:ro \
  --restart unless-stopped \
  -d zabbix/zabbix-web-nginx-pgsql:alpine-8.0-latest
```

容器启动并运行后，Zabbix 服务器将启动，Web 界面可用于登录。

> **Note:**
> 使用 `docker ps` 检查容器状态。所有容器都应显示 `Up` 状态。如果任何容器显示 `Exited`，请使用 `docker logs -f <container-name>` 检查其日志中的错误。

在 RHEL 8-10 上部署带 Java 网关的 Zabbix 服务器（MySQL）

本示例介绍如何在 Red Hat Enterprise Linux 8、9 或 10 上部署使用 MySQL 后端的 Zabbix 服务器、用于JMX 监控 的 Java 网关，以及基于 NGINX 的 Web 界面。

在 Red Hat Enterprise Linux 上，建议使用 Podman 而不是 Docker 作为容器运行时。Podman 的工作方式与 Docker 类似，但不要求以 root 身份运行后台服务，因此更适合 Red Hat 环境。

1. 创建名为 `zabbix` 的新 pod，并公开 Zabbix Web 界面和 Zabbix 服务器 trapper 的端口：

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

2. （可选）在 `zabbix` pod 中运行 Zabbix agent 容器：

```
podman run --name zabbix-agent \
  -e ZBX_SERVER_HOST="127.0.0.1,localhost" \
  --restart=always \
  --pod=zabbix \
  -d registry.connect.redhat.com/zabbix/zabbix-agent-80:latest
```

3. 在 RHEL 主机上创建 `./mysql/` 目录：

```
mkdir -p ./mysql
```

4. 运行一个空的 MySQL 服务器容器（将 `zabbix_pwd` 和 `root_pwd` 替换为强密码，并在后续步骤中使用这些密码）：

```
podman run --name mysql-server -t \
  -e MYSQL_DATABASE="zabbix" \
  -e MYSQL_USER="zabbix" \
  -e MYSQL_PASSWORD="zabbix_pwd" \
  -e MYSQL_ROOT_PASSWORD="root_pwd" \
  -v ./mysql/:/var/lib/mysql/:Z \
  --restart=always \
  --pod=zabbix \
  -d mysql:8.4 \
  --character-set-server=utf8mb4 --collation-server=utf8mb4_bin
```

5. 查看 MySQL 日志，并等待 MySQL 准备好接受连接（使用 `Ctrl+C` 退出日志）：

```
podman logs -f mysql-server

#### ... [Server] /usr/sbin/mysqld: ready for connections. ...
```

6. 运行 Zabbix Java 网关容器：

```
podman run --name zabbix-java-gateway -t \
  --restart=always \
```

```
  --pod=zabbix \
  -d registry.connect.redhat.com/zabbix/zabbix-java-gateway-80
```

7. 在 MySQL 服务器容器中启用 log_bin_trust_function_creators。此设置会暂时放宽对存储函数的安全限制；在后续步骤中将禁用该设置。启用后，Zabbix 数据库用户无需 SUPER 权限即可创建存储函数。当启用二进制日志记录时，MySQL 要求创建存储函数的用户具有该权限（MySQL 8.0 起默认启用二进制日志记录）。运行以下命令：

```
podman exec -it mysql-server mysql -u root -p'root_pwd' \
  -e "SET GLOBAL log_bin_trust_function_creators = 1;"
```

8. 运行 Zabbix 服务器容器，并将其连接到 Java 网关和 MySQL 服务器容器：

```
podman run --name zabbix-server-mysql -t \
  -e DB_SERVER_HOST="127.0.0.1" \
  -e MYSQL_DATABASE="zabbix" \
  -e MYSQL_USER="zabbix" \
  -e MYSQL_PASSWORD="zabbix_pwd" \
  -e ZBX_JAVAGATEWAY="127.0.0.1" \
  --restart=always \
  --init \
  --pod=zabbix \
  -d registry.connect.redhat.com/zabbix/zabbix-server-mysql-80
```

9. 查看 Zabbix 服务器日志，并等待 Zabbix 服务器完成数据库架构初始化（使用 Ctrl+C 退出日志）：

```
podman logs -f zabbix-server-mysql

#### ... [info]: ** Creating 'zabbix' schema in MySQL
#### ... [info]: ** Database schema successfully created!
```

10. 在 MySQL 服务器容器中禁用 log_bin_trust_function_creators。此操作会重新启用禁止非 SUPER 用户创建存储函数的安全限制。运行以下命令：

```
podman exec -it mysql-server mysql -u root -p'root_pwd' \
  -e "SET GLOBAL log_bin_trust_function_creators = 0;"
```

11. 运行 Zabbix Web 界面容器，并将其连接到 Zabbix 服务器和 MySQL 服务器容器：

```
podman run --name zabbix-web-mysql -t \
  -e ZBX_SERVER_HOST="127.0.0.1" \
  -e DB_SERVER_HOST="127.0.0.1" \
  -e MYSQL_DATABASE="zabbix" \
  -e MYSQL_USER="zabbix" \
  -e MYSQL_PASSWORD="zabbix_pwd" \
  -e PHP_TZ="Europe/Riga" \
  --restart=always \
  --pod=zabbix \
  -d registry.connect.redhat.com/zabbix/zabbix-web-mysql-80
```

Note:
zabbix pod 将 zabbix-web-mysql 容器的 8080/TCP 端口以 80/TCP 端口（HTTP）的形式公开到主机。

容器启动并运行后，Zabbix 服务器将启动，Web 界面可用于登录。

Note:
使用 docker ps 检查容器状态。所有容器都应显示 Up 状态。如果任何容器显示 Exited，请使用 docker logs -f <container-name> 检查其日志中的错误。

从源代码安装

您可以通过从源代码编译来获取最新版本的 Zabbix。
另请参见获取 Zabbix 源代码。

此处提供了从源代码安装 Zabbix 的分步教程。

安装 Zabbix 守护进程

1 下载源代码归档文件

前往 Zabbix 下载页面 并下载源代码归档文件。下载完成后，运行以下命令解压源代码：

```
tar -zxvf zabbix-8.0.0.tar.gz
```

> **Note:**
> 请在命令中输入正确的 Zabbix 版本。它必须与已下载归档文件的名称匹配。

2 创建用户账户

所有 Zabbix 守护进程都在非特权系统用户下运行。
如果 Zabbix 守护进程从非特权用户账户启动，它将继续以该用户身份运行。

在默认配置中，如果守护进程以 `root` 启动，它将切换到必须存在的 `zabbix` 用户账户。要创建 `zabbix` 系统用户和组，请运行以下命令。

基于 RedHat 的系统：

```
groupadd --system zabbix
useradd --system -g zabbix -d /usr/lib/zabbix -s /sbin/nologin -c "Zabbix Monitoring System" zabbix
```

基于 Debian 的系统：

```
addgroup --system --quiet zabbix
adduser --quiet --system --disabled-login --ingroup zabbix --home /var/lib/zabbix --no-create-home zabbix
```

无需为 Zabbix 前端单独创建用户账户。

安全建议

如果 Zabbix 服务器 和agent 在同一台机器上运行，建议让它们在不同的用户账户下运行。如果两者以同一用户身份运行，agent 将能够访问服务器的配置文件，这可能会将敏感信息（例如数据库密码）暴露给 Zabbix 中任何具有 Admin 级别权限的用户。

> **Attention:**
> 以 `root`、`bin` 或任何其他具有特殊权限的账户运行 Zabbix 都存在安全风险。

主目录（可选）

Zabbix 进程不需要主目录，因此通常不建议创建。但是，如果您需要依赖主目录的功能（例如将 MySQL 凭据存储在 $HOME/.my.cnf 中），则可以使用以下命令创建主目录。

在基于 RedHat 的系统上，运行：

```
mkdir -m u=rwx,g=rwx,o= -p /usr/lib/zabbix
chown zabbix:zabbix /usr/lib/zabbix
```

在基于 Debian 的系统上，运行：

```
mkdir -m u=rwx,g=rwx,o= -p /var/lib/zabbix
chown zabbix:zabbix /var/lib/zabbix
```

3 创建 Zabbix 数据库

对于 Zabbix 服务器 和proxy 守护进程，以及 Zabbix 前端，都需要一个数据库。运行 Zabbix agent 则不需要数据库。

已提供用于创建数据库 schema 并插入数据集的 SQL 脚本。Zabbix proxy 数据库只需要 schema，而 Zabbix 服务器数据库除了 schema 之外还需要数据集。

创建好 Zabbix 数据库后，请继续执行后续的 Zabbix 编译步骤。

4 配置源代码

构建 Zabbix 服务器、Zabbix proxy 或 Zabbix agent 需要 C99 及 GNU 扩展。可以通过设置 CFLAGS="-std=gnu99" 显式指定此版本：

```
export CFLAGS="-std=gnu99"
```

> **Note:**
> 如果从 Zabbix Git 仓库安装，则必须先运行：
> `./bootstrap.sh`

配置 Zabbix 服务器或 proxy 的源代码时，必须指定要使用的数据库类型。服务器或 proxy 进程一次只能编译一种数据库类型。

要查看所有受支持的配置选项，请在解压后的 Zabbix 源代码目录中运行：

```
./configure --help
```

要配置 Zabbix 服务器和 agent，可以运行类似以下的命令：

```
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --wit
```

要配置 Zabbix 服务器（使用 PostgreSQL 等），可以运行：

```
./configure --enable-server --with-postgresql --with-net-snmp
```

要配置 Zabbix proxy（使用 SQLite 等），可以运行：

```
./configure --prefix=/usr --enable-proxy --with-net-snmp --with-sqlite3 --with-ssh2
```

要配置 Zabbix agent，可以运行：

```
./configure --enable-agent
```

或者，对于 Zabbix agent 2：

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

Note:
构建 Zabbix agent 2 必须安装受支持的 Go 版本。

编译选项说明：

- `--enable-agent` - 编译 Zabbix agent，以及Zabbix get 和Zabbix sender 命令行工具。
- `--with-libcurl` - 虚拟机监控、SMTP 身份验证、`web.page.*` Zabbix agent 监控项以及遥测查询监控项所必需。另请参阅：要求（libcurl）。
- `--with-libxml2` - 虚拟机监控所必需。
- `--with-libpcre2[=DIR]` - Zabbix 始终使用 PCRE2 库进行编译；此选项仅允许指定自定义 PCRE2 安装路径。
- `--with-mysql=/path/to/mysql_config` - 指定特定 MySQL 客户端库配置的路径。在安装了多个版本的 MySQL 或 MariaDB 时很有用。
- `--enable-static` - 静态链接库（Solaris不支持）。如果计划将编译后的二进制文件分发到未安装所需库的系统，请使用此选项。构建 Zabbix 服务器时不建议使用。要静态构建服务器，需要每个外部库的静态版本。configure 脚本不会自动检查这一点。
- `--with-stacksize=<value>` - 设置每个线程的堆栈大小（以 KB 为单位）（例如，`--with-stacksize=512`）。如果 Zabbix 因堆栈溢出而崩溃或冻结，可以增大此值（例如，在默认线程堆栈限制较低的系统上执行预处理期间）。
- `--with-apm`（仅与 `--enable-proxy` 一起使用）- Zabbix proxy 上的应用性能监控（APM）所必需。另请参阅：要求（g++、gRPC、protobuf、Abseil）。

Attention:
如果 `./configure` 因缺少库或其他问题而失败，请检查 `config.log` 文件以获取详细的错误信息。
例如，如果缺少 `libssl`，直接显示的错误消息可能会产生误导：

```
checking for main in -lmysqlclient... no
configure: error: Not found mysqlclient library
```

在这种情况下，`config.log` 会显示实际原因：

```
/usr/bin/ld: cannot find -lssl
/usr/bin/ld: cannot find -lcrypto
```

另请参阅：

- 编译支持加密的 Zabbix
- 已知的编译问题

5 制作并安装所有内容

Note:
如果从 Zabbix Git 仓库 安装，需要先运行：

```
$ make dbschema
```

```
make install
```

此步骤应由具有足够权限的用户运行（通常为'root'，或通过使用 `sudo`）。

默认情况下，运行 `make install` 会将守护进程二进制文件（zabbix_server、zabbix_agentd、zabbix_proxy）安装到 /usr/local/sbin，并将客户端二进制文件（zabbix_get、zabbix_sender）安装到 /usr/local/bin。

Note:
如需指定除 /usr/local 之外的其他位置，请在前一步配置源代码时使用 --prefix 键，例如 --prefix=/home/zabbix。在这种情况下，守护进程二进制文件将安装在 <prefix>/sbin 下，而实用程序将安装在 <prefix>/bin 下。手册页将安装在 <prefix>/share 下。

6 检查并编辑配置文件

- 编辑 Zabbix agent 配置文件（zabbix_agentd.conf）。

对于每个安装了 zabbix_agentd 的主机，都需要配置此文件。

必须在文件中指定 Zabbix 服务器的 **IP** 地址。来自其他主机的连接将被拒绝。

- 编辑 Zabbix 服务器配置文件（zabbix_server.conf）。

必须指定数据库名称、用户和密码（如果使用）。

如果是小型安装（最多监控十台主机），其余参数使用默认值即可满足需求。不过，如果希望最大限度地提升 Zabbix 服务器（或 proxy）的性能，则应更改默认参数。

- 如果安装了 Zabbix proxy，请编辑proxy 配置文件（zabbix_proxy.conf）。

必须指定服务器 IP 地址和 proxy 主机名（服务器必须能够识别该主机名），以及数据库名称、用户和密码（如果使用）。

Note:
使用 SQLite 时，必须指定数据库文件的完整路径；不需要数据库用户和密码。

7 启动守护进程

在服务器端运行 zabbix_server。

```
zabbix_server
```

Note:
请确保您的系统允许分配 36MB（或稍多一些）的共享内存，否则服务器可能无法启动，并且您会在服务器日志文件中看到“Cannot allocate shared memory for <type of cache>.”。这种情况可能发生在 FreeBSD、Solaris 8 上。

在所有被监控的机器上运行 zabbix_agentd。

```
zabbix_agentd
```

Note:
请确保您的系统允许分配 2MB 的共享内存，否则 agent 可能无法启动，并且您会在 agent 日志文件中看到 “Cannot allocate shared memory for collector.”。这种情况可能发生在 Solaris 8 上。

如果您已安装 Zabbix proxy，请运行 zabbix_proxy。

```
zabbix_proxy
```

安装 Zabbix Web 前端

复制 PHP 文件

Zabbix 前端是用 PHP 编写的，因此运行它需要一个支持 PHP 的 web 服务器。安装只需将 ui 目录中的 PHP 文件复制到 web 服务器的 HTML 文档目录中即可。

Apache web 服务器的 HTML 文档目录常见位置包括：

- /usr/local/apache2/htdocs（从源代码安装 Apache 时的默认目录）
- /srv/www/htdocs（OpenSUSE、SLES）
- /var/www/html（Debian、Ubuntu、Fedora、RHEL）

建议使用子目录，而不是 HTML 根目录。要创建子目录并将 Zabbix 前端文件复制到其中，请执行以下命令，将 <htdocs> 替换为实际目录：

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

如果计划使用英语以外的其他语言，请参阅安装其他前端语言了解说明。

安装前端

有关 Zabbix 前端安装向导的信息，请参见Web 界面安装 页面。

安装 Zabbix agent 2 可加载插件

只有在您希望监控内置插件未覆盖的目标时，才需要安装 Zabbix agent 2 可加载插件（例如，MongoDB 服务器或集群、PostgreSQL 及其分支等）。请参阅可加载插件 和内置插件 的完整列表。

Attention:
在安装插件之前，请先查看其 README 文件。其中可能包含特定要求和安装说明。

要从源代码安装，请先 下载 并解压可加载插件源代码归档文件。

要编译插件，请进入解压后的插件目录并运行 `make`：

```
make
```

Note:
构建 Zabbix agent 2 可加载插件时，必须安装受支持的 Go 版本。

只要 Zabbix agent 2 能够加载，插件可执行文件可以放在任意位置。请在插件配置文件中指定插件二进制文件的路径，例如，对于PostgreSQL 插件，可在 `postgresql.conf` 中配置：

```
Plugins.PostgreSQL.System.Path=/path/to/executable/zabbix-agent2-plugin-postgresql
```

插件配置文件的路径必须在 Zabbix agent 2 配置文件的 `Include` 参数中指定：

```
Include=/path/to/plugin/configuration/file/postgresql.conf
```

有关配置插件的更多详细信息，请继续查看设置。

Zabbix 提供的可加载插件使用简单的 makefile，并包含以下构建目标：

- `make` - 构建插件
- `make clean` - 删除构建插件时创建的所有文件
- `make check` - 运行自检（需要真实的监控目标，例如 PostgreSQL 数据库）
- `make style` - 使用 `golangci-lint` 检查 Go 代码风格
- `make format` - 使用 `go fmt` 格式化 Go 代码
- `make dist` - 创建包含所有依赖项的源代码归档文件

安装 Java 网关

只有在需要监控 JMX 应用程序时，才需要安装 Java 网关。Java 网关是轻量级的，不需要数据库。

要从源代码安装，首先下载并解压源代码归档文件。

要编译 Java 网关，请使用 `./configure` 脚本并带上 `--enable-java` 选项。建议指定 `--prefix` 选项，以请求使用不同于默认 `/usr/local` 的安装路径，因为安装 Java 网关会创建完整的目录树，而不仅仅是一个可执行文件。

```
./configure --enable-java --prefix=$PREFIX
```

要将 Java 网关编译并打包为 JAR 文件，请运行 `make`。请注意，执行此步骤时，您的路径中需要有 `javac` 和 `jar` 可执行文件。

```
make
```

现在，您会在 `src/zabbix_java/bin` 中得到一个 `zabbix-java-gateway-$VERSION.jar` 文件。如果您希望直接从发行版目录中的 `src/zabbix_java` 运行 Java 网关，那么可以继续阅读配置和运行Java 网关的说明。否则，请确保您有足够的权限，然后运行 `make install`。

```
make install
```

请继续查看设置以了解有关配置和运行 Java 网关的更多详细信息。

安装 Zabbix web service

只有在您希望使用计划报告时，才需要安装 Zabbix web service。

要从源代码安装，请先下载并解压源代码归档文件。

要编译 Zabbix web service，请运行带有 `--enable-webservice` 选项的 `./configure` 脚本。

> **Note:**
> 必须安装受支持的Go 版本 才能构建 Zabbix web service。

在安装了 web service 的机器上运行 zabbix_web_service：

```
zabbix_web_service
```

有关配置计划报告生成的更多详细信息，请继续查看设置。

安装 Zabbix MCP 服务器

要从源代码安装，请先下载并解压源代码归档文件。

要编译 Zabbix MCP 服务器，请在 MCP 服务器源代码目录中运行以下命令：

```
make build
```

> **Note:**
> 必须安装受支持的 Go 版本才能构建 Zabbix MCP 服务器。

要在 `zabbix_mcp_server -V` 的输出以及 Zabbix API 请求的 `User-Agent` 中报告特定版本，请设置 `VERSION` 变量，例如 `make build VERSION=8.0.0`。

要将二进制文件安装到 `/usr/local/sbin`，并将示例配置文件安装到 `/etc/zabbix`，请运行：

```
sudo make install
```

编辑配置文件 `/etc/zabbix/zabbix_mcp_server.conf`，并至少设置以下参数：

- `FrontendURL` - Zabbix API URL，以 `api_jsonrpc.php` 结尾；
- `AllowedIP` - 允许连接的 MCP 客户端地址。

要加密来自 MCP 客户端的连接，请设置 `TLSAccept=cert`，并指定 `TLSCertFile` 和 `TLSKeyFile`；或者在负责处理 TLS 的受信任反向 proxy 后运行 MCP 服务器。

如果 `zabbix` 用户不存在（例如，计算机上仅安装了 MCP 服务器），请按照创建用户账户中的说明创建该用户。

验证配置文件并启动 zabbix_mcp_server：

```
zabbix_mcp_server -c /etc/zabbix/zabbix_mcp_server.conf -T
zabbix_mcp_server -c /etc/zabbix/zabbix_mcp_server.conf
```

有关配置服务器和连接 MCP 客户端的更多详细信息，请参阅MCP 服务器。

获取 Zabbix 源代码

获取 Zabbix 源代码有多种方式：

- 您可以从 Zabbix 官方网站下载已发布的稳定版本
- 您可以从 Zabbix 官方网站的开发者页面下载 nightly build
- 您可以从 Git 源代码仓库系统获取最新的开发版本：
  - 完整仓库的主要地址为 https://git.zabbix.com/scm/zbx/zabbix.git
  - Master 分支和受支持的发布版本也镜像到了 Github：https://github.com/zabbix/zabbix

必须安装 Git 客户端才能克隆该仓库。官方命令行 Git 客户端软件包在各发行版中通常名为 **git**。例如，在 Debian/Ubuntu 上安装时，请运行：

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

要获取全部 Zabbix 源代码，请切换到您希望放置代码的目录并执行：

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

编译问题

这些是关于从源码编译 Zabbix 的已知问题。对于所有其他情况，请参见已知问题页面。

位于非标准位置的库

Zabbix 允许您指定位于非标准位置的库。在下面的示例中，Zabbix 将从指定的非标准位置运行 `curl-config`，并使用其输出确定要使用的正确 libcurl。

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

如果这是系统中安装的唯一 libcurl，这样可以正常工作；但如果在标准位置还安装了另一个 libcurl（例如由包管理器安装），则可能无法正常工作。当 Zabbix 需要较新版本的库，而其他应用程序仍需要较旧版本时，就会出现这种情况。

因此，当同一组件也存在于标准位置时，指定非标准位置中的组件并不总是有效。

例如，如果您使用安装在 /usr/local 中的较新 libcurl，同时系统中仍安装了 libcurl 软件包，Zabbix 可能会选错库，从而导致编译失败：

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

这里，函数 curl_easy_header() 在较旧的 /usr/lib/x86_64-linux-gnu/libcurl.so 中不可用，但在较新的 /usr/local/lib/libcu 中可用。

问题出在链接器标志的顺序上，一种解决方案是在 LDFLAGS 变量中指定库的完整路径：

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

请注意 -Wl,--no-as-needed 选项，在某些系统上可能需要使用该选项（另请参见 Debian-based 系统上的默认链接选项）。

某些系统上的堆栈大小过小

如果由于堆栈溢出导致 Zabbix 崩溃或冻结，您可以在配置源代码时使用 --with-stacksize 选项来增加每个线程的堆栈大小。在默认线程堆栈限制较低的系统上可能会出现此问题，尤其是在预处理期间，因为此时会创建多个线程。

以下示例将每个线程的堆栈大小设置为 512 KB：

```
./configure --enable-server --with-mysql --with-stacksize=512
```

您可以在基于 Linux 的系统上使用 ulimit -s 命令在运行时检查系统线程堆栈限制。

从软件包安装的说明

概述

Zabbix 8.0 的官方预发布软件包可在 Zabbix 网站上获取。您可以选择操作系统和 Zabbix 组件，以生成安装说明和命令。

除以下软件包安装说明外，请同时参阅安装和设置页面。

以下 Linux 发行版提供软件包：

- Red Hat Enterprise Linux 及其衍生版：AlmaLinux、Amazon Linux 2023、CentOS Stream、CentOS 7、Oracle Linux、Rocky Linux
- Debian、Ubuntu、Raspberry Pi OS、Raspbian
- SUSE Linux Enterprise Server、openSUSE Leap

> **Attention:**
> 某些操作系统发行版（尤其是基于 Debian 的发行版）提供其自有的 Zabbix 软件包。这些软件包不受 Zabbix 支持，并且可能已过时，或缺少最新功能和错误修复。建议仅使用来自 Zabbix 官方软件仓库的官方软件包。如果您之前从操作系统的软件仓库安装了 Zabbix，请参阅从操作系统软件仓库升级 Zabbix 软件包的步骤。

软件包支持 MySQL/PostgreSQL 数据库以及 Apache/Nginx Web 服务器。请注意，Zabbix 服务器和 proxy 不能共用同一个数据库；如果两者安装在同一台主机上，请使用不同的数据库名称。

如有需要，可在 Zabbix 官方软件仓库中获取 Zabbix agent/agent 2、Zabbix get 和 Zabbix sender 的独立软件包。

Zabbix 还为非 Linux 操作系统提供预编译的 Zabbix agent 二进制文件；请参阅：

- 从 MSI 安装 Windows agent
- 从 PKG 安装 macOS agent
- IBM AIX 上的 Zabbix agent
- 旧版二进制文件（适用于较旧或较少见的系统，例如 HP-UX、NetBSD、Tru64 以及较旧版本的 SLES）

软件包安装说明

以下说明适用于所有系统：

- 如果使用 PostgreSQL，在 Zabbix 服务器/proxy 配置中设置 DBHost=localhost（或 IP 地址），会使 PostgreSQL 使用网络套接字，而不是本地 UNIX 套接字；相关设置说明请参阅SELinux 配置。
- 如果使用 TimescaleDB，请参阅其他TimescaleDB 设置。
- 如果安装 Zabbix Java 网关（用于监控 JMX 应用程序），请参阅适用于基于 RHEL 的系统 和基于 Debian 的系统 的其他设置说明。

- 如果安装 Zabbix MCP 服务器，请安装 zabbix-mcp-server 软件包，并在启动服务前更新配置文件（至少更新 FrontendURL 和 AllowedIP）。
- 要以 root 身份运行 Zabbix agent，请参阅以 root 身份运行 agent。

以下说明适用于 RHEL 及其衍生系统：

- 如果已为 EL9 启用 EPEL 软件仓库，而该仓库也提供 Zabbix 软件包，则必须在安装官方 Zabbix 软件包前将其从软件包解析中排除；请参阅EPEL Zabbix 软件包的意外安装。
- 要在 Red Hat UBI 环境中安装 Zabbix 软件包，请参阅Red Hat UBI 环境中的 RHEL Zabbix 软件包。
- 要使用ICMP ping 监控项，fping 软件包也可从 Zabbix 官方软件仓库 获取。

SELinux 配置

Zabbix 使用基于 socket 的进程间通信。
在启用了 Security-Enhanced Linux（SELinux）的系统上，您可能需要添加 SELinux 规则，以允许 Zabbix 在 SocketDir 目录中创建/使用 UNIX 域 socket。
socket 文件由 Zabbix 服务器（alerter、preprocessing、IPMI）和 Zabbix proxy（IPMI）使用，并且在进程运行期间会存在。

在 SELinux 处于 enforcing 模式时，执行以下命令以启用 Zabbix 前端和服务器之间的通信：

对于 RHEL 7（及更高版本）、AlmaLinux、CentOS Stream、Oracle Linux、Rocky Linux 8（及更高版本）：

```
setsebool -P httpd_can_connect_zabbix on
```

如果数据库通过网络访问（对于 PostgreSQL，包括 localhost），还需要允许 Zabbix 前端连接到数据库：

```
setsebool -P httpd_can_network_connect_db on
```

对于 RHEL 7 之前的版本：

```
setsebool -P httpd_can_network_connect on
setsebool -P zabbix_can_network on
```

应用 SELinux 设置后，重启 Apache：

```
systemctl restart httpd
```

可选地，您可以从 Zabbix 官方仓库 安装预定义的 zabbix-selinux-policy 包。
该包适用于所有受支持的操作系统版本，旨在简化 Zabbix 部署，并避免用户因配置复杂而关闭 SELinux。
 对于 AlmaLinux、CentOS Stream、RHEL、Rocky Linux 和 Oracle Linux 的 10 版本，SELinux 策略由 zabbix-selinux-policy-main 包提供；zabbix-selinux-policy 包依赖于 zabbix-selinux-policy-main 或 selinux-policy-targeted-extra。
zabbix-selinux-policy-main 包与 selinux-policy-targeted-extra 冲突；因此，这两个包不能同时安装。

> **Attention:**
> 为获得最高安全性，建议使用自定义 SELinux 设置。

zabbix-selinux-policy（对于 AlmaLinux、CentOS Stream、RHEL、Rocky Linux 和 Oracle Linux 的 10 版本，则为 zabbix-selinux-policy-main）包包含一个基础 SELinux 策略，允许 Zabbix 创建和使用 socket，并启用 HTTPd 连接到 PostgreSQL（由前端使用）。

源 zabbix_policy.te 文件包含以下规则：

```
module zabbix_policy 1.2;

require {
    type zabbix_t;
    type zabbix_port_t;
    type zabbix_var_run_t;
    type postgresql_port_t;
    type httpd_t;
    class tcp_socket name_connect;
    class sock_file { create unlink };
    class unix_stream_socket connectto;
}

#####============= zabbix_t ==============
allow zabbix_t self:unix_stream_socket connectto;
allow zabbix_t zabbix_port_t:tcp_socket name_connect;
allow zabbix_t zabbix_var_run_t:sock_file create;
allow zabbix_t zabbix_var_run_t:sock_file unlink;
```

```
allow httpd_t zabbix_port_t:tcp_socket name_connect;

####============= httpd_t ==============
allow httpd_t postgresql_port_t:tcp_socket name_connect;
```

Debuginfo 软件包

Debuginfo 软件包包含 Zabbix 二进制文件的调试符号。它们不是正常安装或运行所必需的，但对于高级故障排查很有用。

请注意，Debuginfo 软件包适用于以下版本：Red Hat Enterprise Linux 7 及更早版本（包括衍生版），以及 SUSE Linux Enterprise Server 15 及更早版本（包括衍生版）。

要启用 zabbix-debuginfo 仓库：

- 在 RHEL 7 上，编辑 /etc/yum.repos.d/zabbix.repo，并为 zabbix-debuginfo 部分设置 enabled=1：

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo - $basearch
baseurl=http://repo.zabbix.com/zabbix/8.0/stable/rhel/7/$basearch/debuginfo/
enabled=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
gpgcheck=1
```

- 在 SUSE 上，编辑 /etc/zypp/repos.d/zabbix.repo，并为 zabbix-debuginfo 部分设置 enabled=1：

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo
type=rpm-md
baseurl=https://repo.zabbix.com/zabbix/8.0/unstable/sles/15/$basearch/debuginfo/
gpgcheck=1
gpgkey=https://repo.zabbix.com/zabbix/8.0/unstable/sles/15/$basearch/debuginfo/repodata/repomd.xml.key
enabled=0
update=1
```

启用后，安装这些软件包：

- 在 RHEL 上，安装一个包含所有 Zabbix 组件调试信息的单一软件包：

```
dnf install zabbix-debuginfo
```

- 在 SUSE 上，安装按组件划分的 debuginfo 软件包：

```
zypper install zabbix-<component>-debuginfo
```

启用不稳定版本发布仓库

以下说明适用于启用不稳定的 Zabbix 发布仓库（默认已禁用），这些仓库用于 Zabbix 小版本发布候选版。

首先，安装或更新到最新的 zabbix-release 软件包。要在系统上启用 rc 软件包，请执行以下操作：

Red Hat Enterprise Linux

打开 /etc/yum.repos.d/zabbix.repo 文件，并为 zabbix-unstable 仓库设置 enabled=1。

```
[zabbix-unstable]
name=Zabbix Official Repository (unstable) - $basearch
baseurl=https://repo.zabbix.com/zabbix/8.0/unstable/rhel/8/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
```

Debian/Ubuntu

打开 /etc/apt/sources.list.d/zabbix.list 并取消注释 “Zabbix 不稳定仓库”。

```
#### Zabbix unstable repository
deb https://repo.zabbix.com/zabbix/8.0/unstable/debian bullseye main
deb-src https://repo.zabbix.com/zabbix/8.0/unstable/debian bullseye main
```

SUSE

打开 /etc/zypp/repos.d/zabbix.repo 文件，并将 zabbix-unstable 仓库的 enabled=1 设置为启用。

```
[zabbix-unstable]
name=Zabbix Official Repository
type=rpm-md
baseurl=https://repo.zabbix.com/zabbix/8.0/unstable/sles/15/x86_64/
gpgcheck=1
gpgkey=https://repo.zabbix.com/zabbix/8.0/unstable/sles/15/x86_64/repodata/repomd.xml.key
enabled=1
update=1
```

从 **MSI** 安装 **Windows agent**

概述

Zabbix agent 可使用 32 位或 64 位 MSI 安装包在 Windows 上安装，可从下载页面获取。

MSI 安装的最低操作系统要求如下：

- 对于 **Zabbix agent**： Windows XP（64 位）或 Windows Server 2003
- 对于 **Zabbix agent 2**： Windows 10（32 位）或 Windows Server 2016

32 位安装包不能安装在 64 位系统上。

安装包包括：

- TLS 支持（TLS 配置为可选）
- Zabbix get 和Zabbix sender 工具（可与 Zabbix agent/agent 2 一起安装，也可单独安装）

> **Attention:**
> Zabbix agent 2 安装包不包含可加载插件（MongoDB、PostgreSQL、MSSQL），需要单独下载并安装。

可通过安装向导或命令行进行安装。

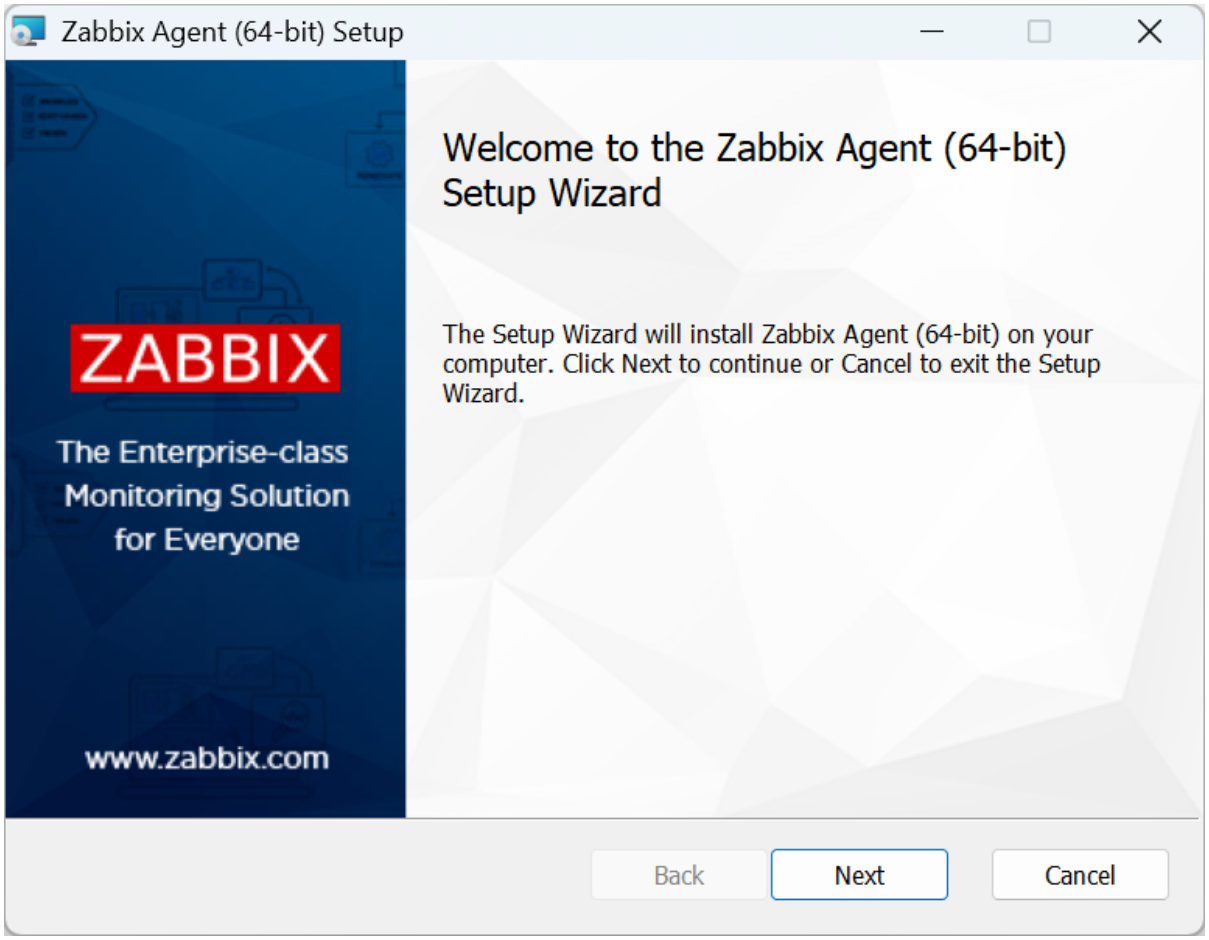
尽管使用 MSI 安装包进行安装已得到完全支持，但建议至少安装 Microsoft .NET Framework 2，以确保正确的错误处理。

> **Attention:**
> 建议使用安装程序提供的默认安装位置。在缺少必要权限的情况下使用自定义位置，可能会影响安装的安全性。

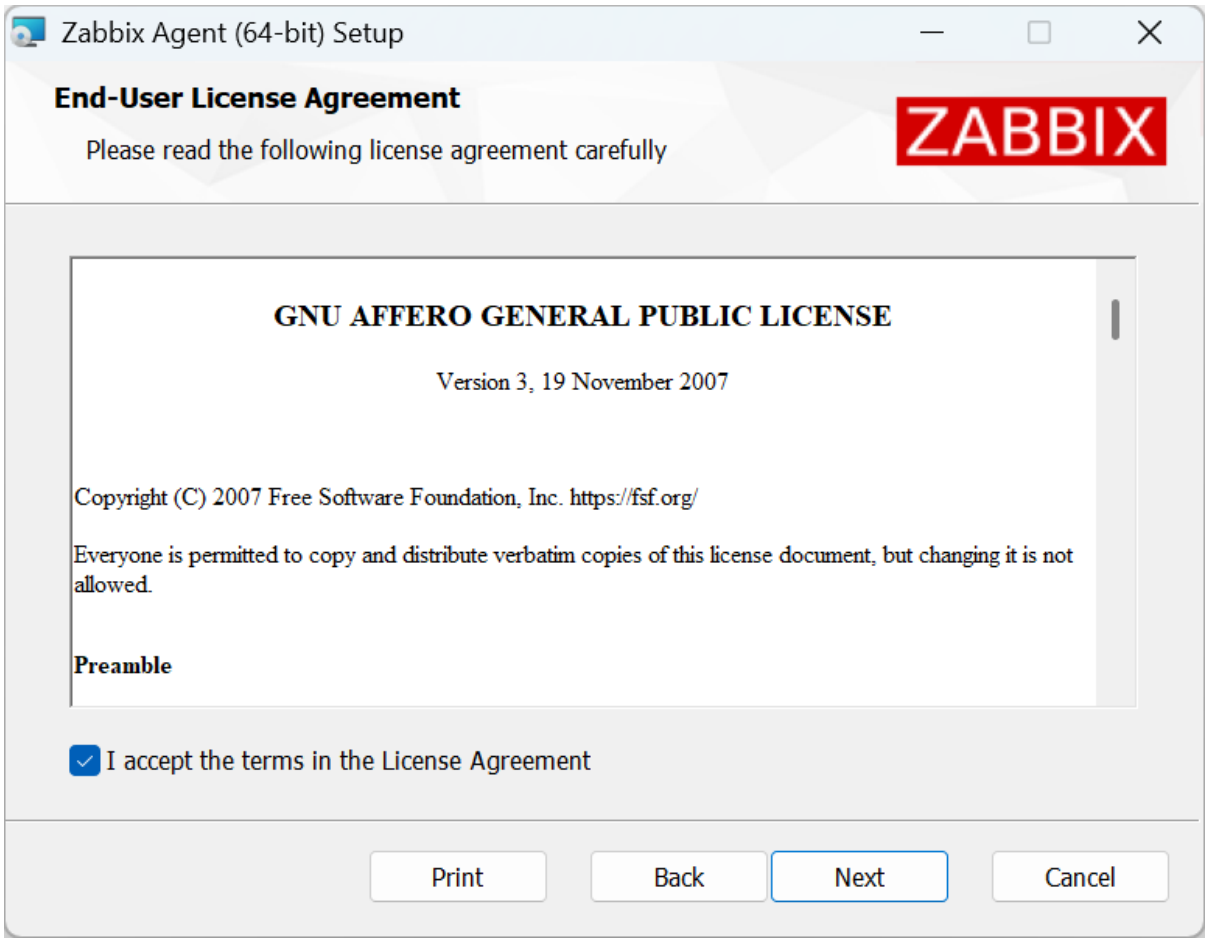
使用安装向导安装

以下安装步骤适用于 Zabbix agent 和 Zabbix agent 2。

1. 双击下载的 MSI 文件以开始安装：

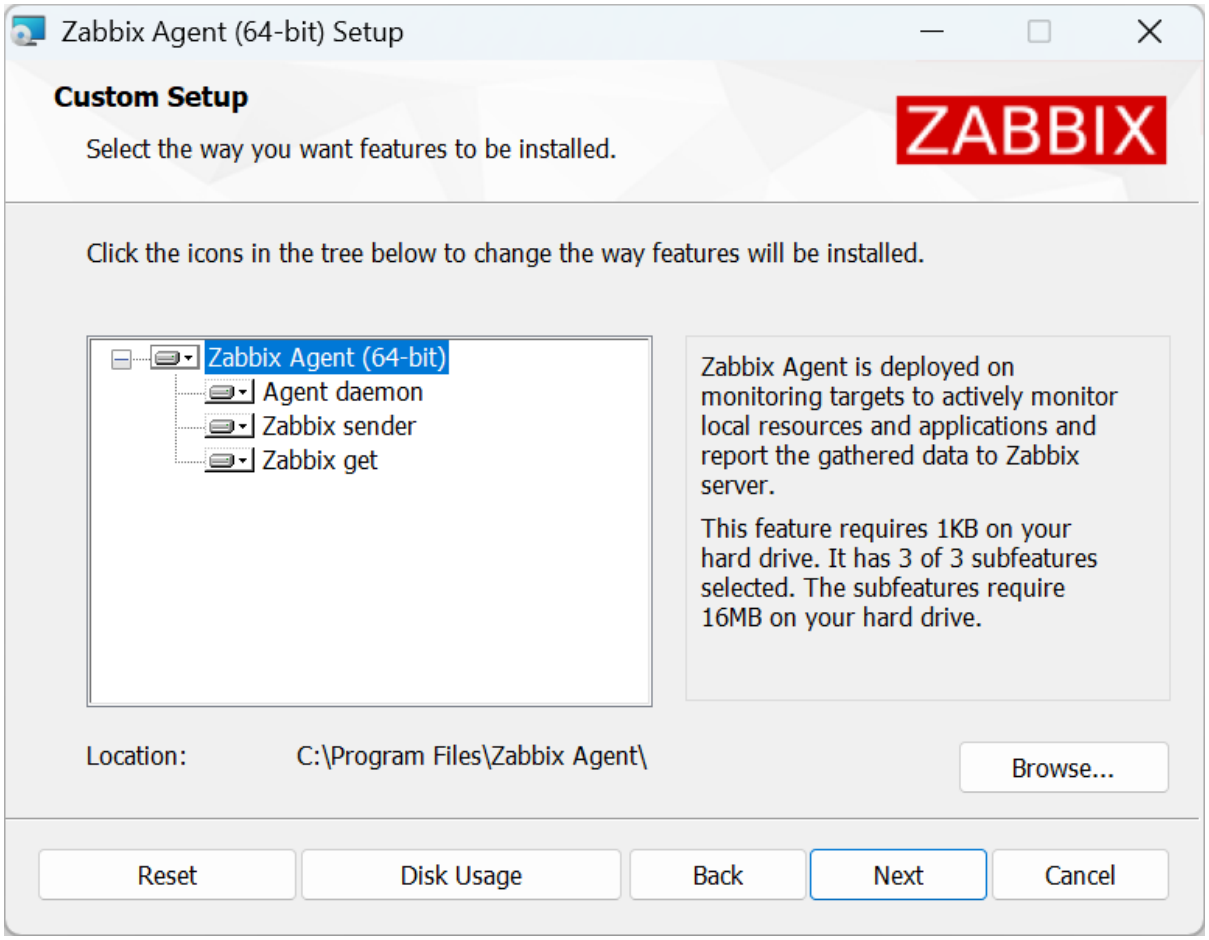


2. 接受最终用户许可协议：



3. 选择要安装的 Zabbix 组件（Agent 守护进程、Zabbix sender、Zabbix get）：

Attention:
建议使用安装程序提供的默认安装位置。在没有必要权限的情况下使用自定义位置可能会危及安装的安全性。

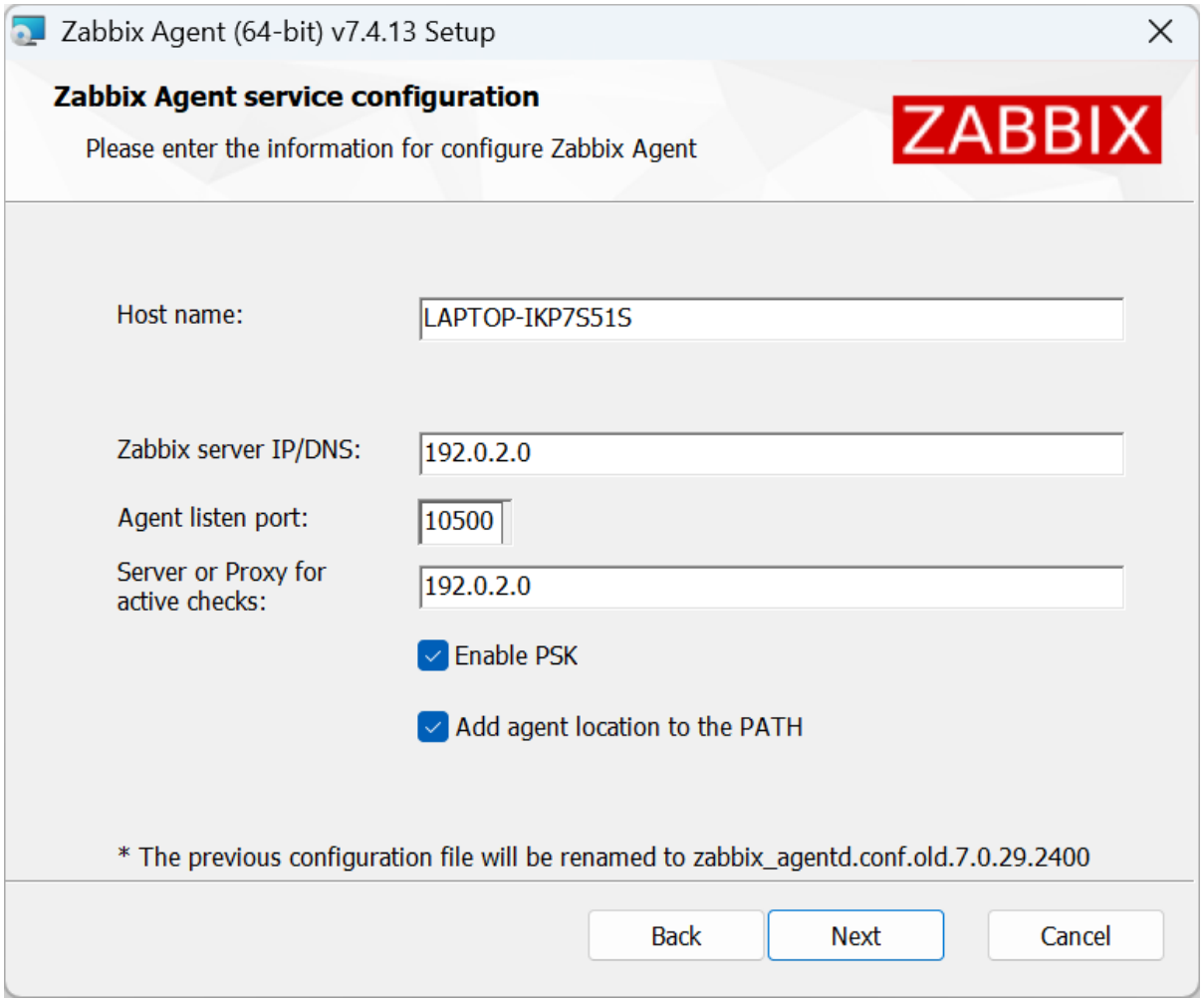


Note:
如果使用 `Program Files` 目录之外的自定义位置，且目标文件夹已存在，安装程序不会修改其权限。在新安装期间，系统会提示您手动设置适当的权限，或继续安装而不修改权限。

4. 配置以下参数。这些参数的值将设置在 Zabbix agent 配置文件中：

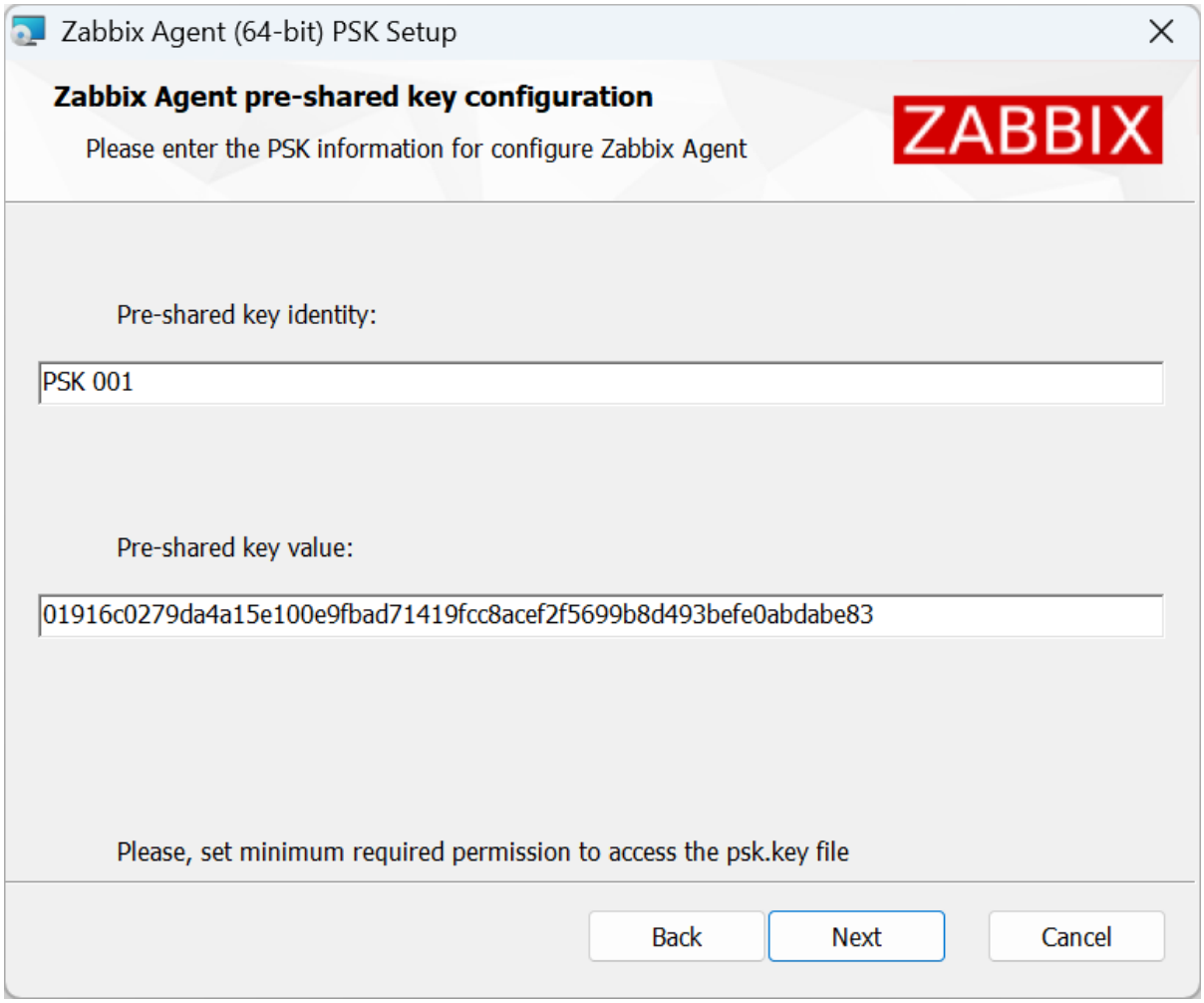
参数	描述
主机名	安装 Zabbix agent 的计算机的主机名。设置Hostname 参数。
Zabbix 服务器 *IP/DNS*	Zabbix 服务器或 Zabbix proxy 的 IP 地址列表（可选用 CIDR 表示法）或 DNS 名称，多个值以逗号分隔。此参数为必填。设置Server 参数。
agent 监听端口	agent 将在此端口上监听来自服务器的连接。设置ListenPort 参数。
用于主动检查的服务器或 *proxy*	用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。服务器/proxy 地址为 IP 地址或 DNS 名称，可选端口以冒号分隔。设置ServerActive 参数。
启用 *PSK*	选中此复选框可使用预共享密钥启用 TLS 支持。将TLSConnect 和TLSAccept 参数设置为 `psk`。
将 *agent* 位置添加到 *PATH*	选中此复选框可将 Zabbix agent 的位置添加到系统 PATH 变量中。

Note:
如果检测到现有的 Zabbix agent，将显示其配置文件中的参数值。此外，现有配置文件将在安装期间重命名，并创建新的配置文件。



5. 如果在上一步中选中了启用 *PSK* 复选框，请配置 PSK 参数。这些参数也将设置在 Zabbix agent 配置文件中：

参数	描述
预共享密钥标识	预共享密钥标识字符串。设置TLSPSKIdentity 参数。
预共享密钥值	预共享密钥字符串值。创建包含密钥的 `psk.key` 文件，并将TLSPSKFile 参数设置为密钥位置（默认：`C:\Program Files\Zabbix Agent\psk.key`）。建议通过调整文件的安全设置来限制对预共享密钥文件的访问，使其仅可由 Zabbix agent（或运行 agent 的用户）读取。

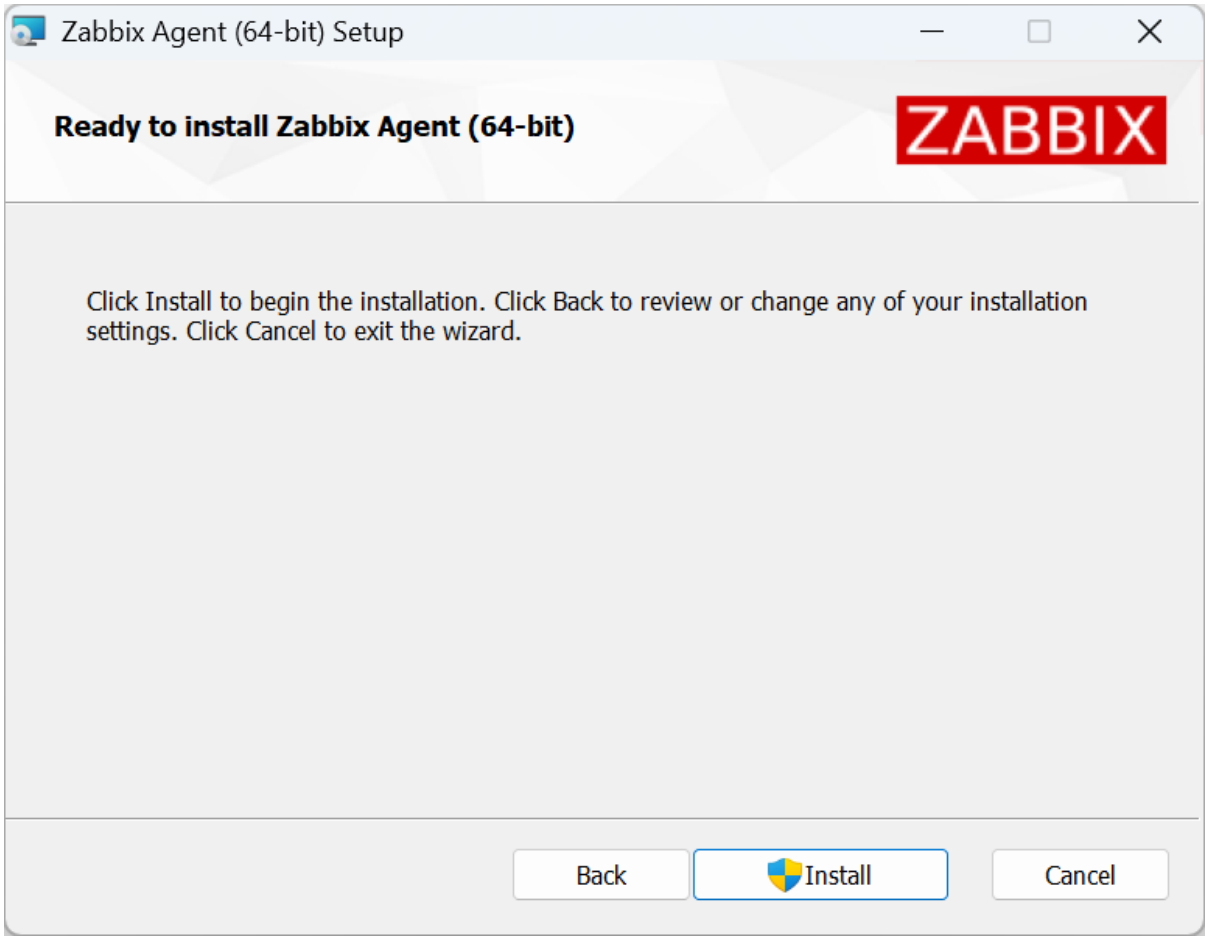


6. 点击安装开始安装。

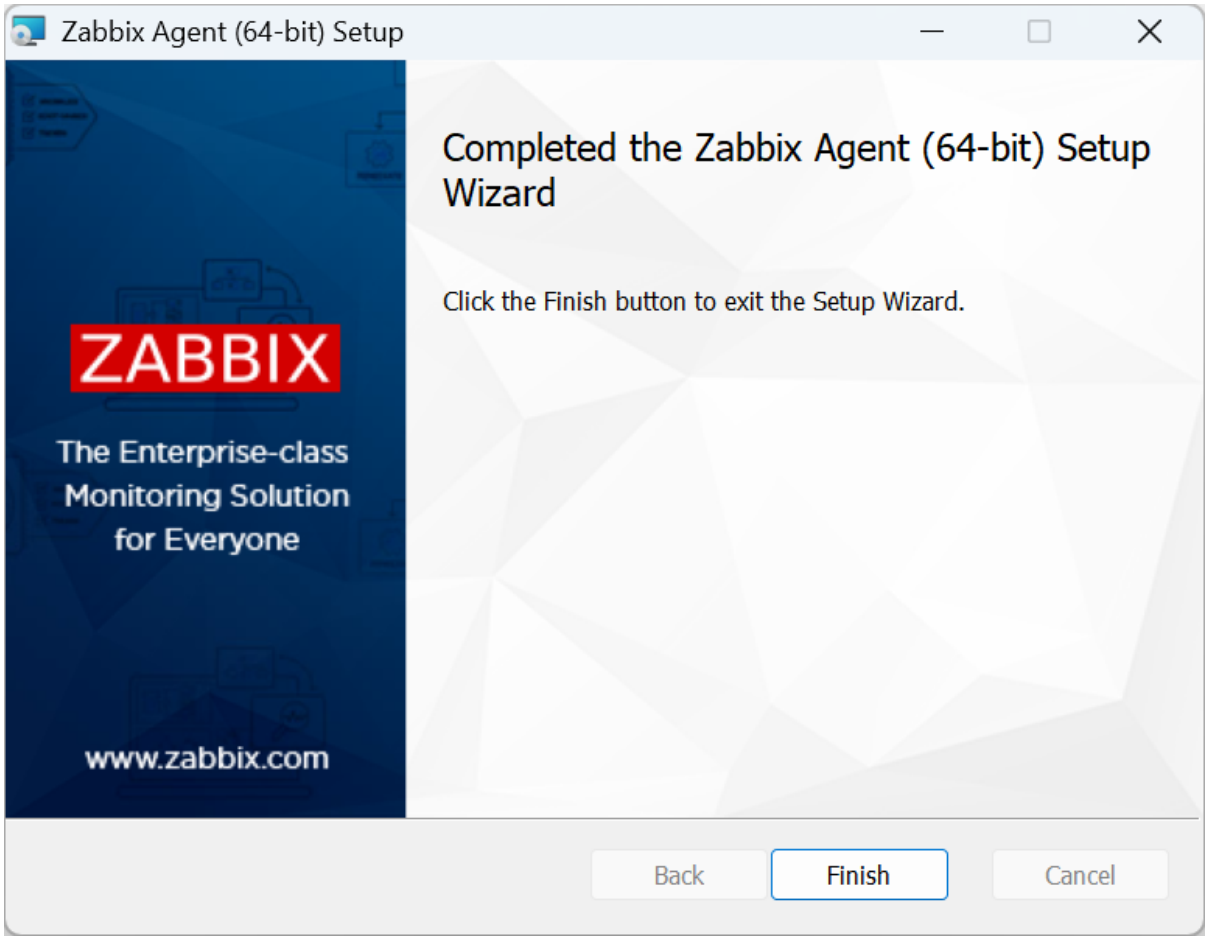
所有选定的 Zabbix 组件和 Zabbix agent 配置文件都将安装到您指定的位置（默认：`C:\Program Files\Zabbix Agent`）。Zabbix agent 2 也同样如此，但其内置插件的其他配置文件将安装到 `zabbix_agent2.d\plugins.d` 子文件夹中。

此外，`zabbix_agentd.exe`（或 `zabbix_agent2.exe`）将被设置为 Windows 服务，并采用延迟自动启动方式（在 Windows Vista/Server 2008 之前的 Windows 版本中则采用自动启动方式）。

如果安装期间正在运行其他版本的 Zabbix agent，系统会提示您选择关闭应用程序并尝试重新启动，或保持应用程序打开；如果选择后者，则需要重新启动计算机。



7. 点击完成按钮退出安装向导。



从命令行安装

可以通过运行 MSI 安装程序并使用 msiexec 从命令行安装 Zabbix agent。例如：

```
msiexec.exe /l*v "C:\package.log" /i "C:\zabbix_agent-8.0.0-windows-amd64-openssl.msi" /qn+ SERVER=192.0.2
```

此方法支持无人值守安装，并可使用参数进行自定义配置。

支持的参数

Zabbix agent MSI 安装包支持以下适用于 Zabbix agent 和 Zabbix agent 2 的参数。

Note:
Zabbix agent/agent2 参数会在安装过程中写入配置文件。单击参数名称可查看其详细说明和配置示例，参见Zabbix agent (Windows) 页面。对于 Zabbix agent 2，请参见Zabbix agent 2 (Windows) 页面。

Parameter	Description
ADDDEFAULT	要安装为默认配置的一组以逗号分隔的组件。更多信息，请参见 ADDDEFAULT property。 可能的值：AgentProgram、GetProgram、SenderProgram、ALL 示例：ADDDEFAULT=AgentProgram,GetProgram
ADDLOCAL	要在本地安装的一组以逗号分隔的组件。更多信息，请参见 ADDLOCAL property。 可能的值：AgentProgram、GetProgram、SenderProgram、ALL 示例：ADDLOCAL=AgentProgram,SenderProgram
ALLOWDENYKEY	用于限制 Zabbix agent 检查 的一组以分号分隔的 AllowKey 或 DenyKey 参数。如有必要，请使用反斜杠转义分隔符（\;）。会在 agent 配置文件中设置AllowKey 和DenyKey 参数。 示例：ALLOWDENYKEY="AllowKey=system.run[more C:\Windows\System32\drivers\etc\hosts\; echo 'File read complete'];DenyKey=system.run[*]"
CONF	Zabbix agent 的模板配置文件 的完整路径名。安装期间，此文件将成为 agent 配置文件。该文件至少必须包含Server 和LogFile 参数。 示例：CONF="C:\full\path\to\example.conf"
DONOTSTART	使用 DONOTSTART=1 可阻止 MSI 安装程序启动 Zabbix agent 服务。
ENABLEPATH	使用 ENABLEPATH=1 可将 Zabbix agent 所在位置添加到系统 PATH 变量。
ENABLEPERSISTENTBUFFER	仅适用于 Zabbix agent 2。启用对主动监控项使用本地持久存储。
HOSTINTERFACE	定义主机接口的可选参数。
HOSTMETADATA	定义主机元数据的可选参数。
HOSTMETADATAITEM	定义用于获取主机元数据的监控项的可选参数。
HOSTNAME	定义主机名的可选参数。
HOSTNAMEITEM	定义用于获取主机名的监控项的可选参数。
INCLUDE	要包含到 Zabbix agent 配置文件中的一组以分号分隔的单个文件或目录中的所有文件。
IGNOREALREADYEXISTSFOLDER	对新安装，使用 IGNOREALREADYEXISTSFOLDER=1 可允许安装到 Program Files 目录之外已存在的目标文件夹中。如果未指定此参数，而该目标文件夹已存在，则安装将失败。
INSTALLFOLDER	Zabbix 组件和 Zabbix agent 配置文件将安装到的文件夹完整路径名。对于 Zabbix agent 2，zabbix_agent2.d\plugins.d 子文件夹中还将安装内置插件 的附加配置文件。 示例：INSTALLFOLDER="C:\Program Files\Zabbix Agent"
LISTENIP	agent 应监听的一组以逗号分隔的 IP 地址。
LISTENPORT	agent 将在此端口上监听来自服务器的连接。
LOGFILE	Zabbix agent 日志文件的名称。
LOGTYPE	日志输出类型。
NONMSICONFNAME	Zabbix agent 的自定义配置文件 的完整路径名。安装期间，此文件中存在的任何有效 agent 配置参数（仅限于本表中列出的参数）都将写入新创建的 agent 配置文件中。该文件至少必须包含Server 参数。 示例：NONMSICONFNAME="C:\full\path\to\example.conf"
PERSISTENTBUFFERFILE	仅适用于 Zabbix agent 2。Zabbix agent 2 应在其中保存 SQLite 数据库的文件。
PERSISTENTBUFFERPERIOD	仅适用于 Zabbix agent 2。在无法连接到服务器或 proxy 时，数据应存储的时间段。
SERVER	一组以逗号分隔的 IP 地址（可选使用 CIDR 表示法）或 Zabbix 服务器或 Zabbix proxy 的 DNS 名称。此参数为必需，除非将 STARTAGENTS 设置为 0。
SERVERACTIVE	用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
SKIP	使用 SKIP=fw 可阻止 MSI 安装程序为 Zabbix agent 添加 Windows 防火墙例外规则。
SOURCEIP	用于到 Zabbix 服务器或 Zabbix proxy 的出站连接的源 IP 地址，或在执行某些监控项（web.page.get、net.tcp.port 等）时建立连接所使用的源 IP 地址。
STARTAGENTS	处理被动检查的 zabbix_agentd 预先 fork 的实例数量。 如果设置为 0，则会禁用被动检查，且 agent 不会在任何 TCP 端口上监听。

Parameter	Description
STARTUPTYPE	Zabbix agent 服务的启动类型。可能的值： **automatic** - 在 Windows 启动时自动启动服务； **delayed** - (默认) 在自动启动的服务完成启动后延迟启动服务（适用于 Windows Vista/Server 2008 及更高版本）； **manual** - 手动启动服务（由用户或应用程序）； **disabled** - 禁用服务，使其无法由用户或应用程序启动。 示例：`STARTUPTYPE=disabled`
STATUSPORT	仅适用于 Zabbix agent 2。如果设置，agent 将在此端口上监听 HTTP 状态请求 (http://localhost:<port>/status)。
TIMEOUT	指定与 Zabbix proxy 或服务器建立连接并交换数据时的等待时间（秒）。
TLSACCEPT	接受的入站连接（用于被动检查）。如果设置为 `psk`，则 TLSCONNECT 也将设置为 `psk`（除非另有指定）。
TLSCAFILE	包含用于对等证书验证的顶级 CA 证书的文件完整路径名。
TLSCERTFILE	包含 agent 证书或证书链的文件完整路径名。
TLSCONNECT	agent 应如何连接到 Zabbix 服务器或 proxy（用于主动检查）。如果设置为 `psk`，则 TLSACCEPT 也将设置为 `psk`（除非另有指定）。
TLSCRLFILE	包含已吊销证书的文件完整路径名。
TLSKEYFILE	包含 Zabbix agent 私钥的文件完整路径名。
TLSPSKFILE	包含 Zabbix agent 预共享密钥的文件完整路径名。如果同时设置了 TLSPSKFILE 和 TLSPSKVALUE，则 TLSPSKVALUE 的值将写入 TLSPSKFILE 指定的文件中。建议通过调整文件的安全设置来限制对预共享密钥文件的访问，使只有 Zabbix agent（或运行 agent 的用户）可以读取它。
TLSPSKIDENTITY	预共享密钥标识字符串。
TLSPSKVALUE	预共享密钥字符串值。如果同时设置了 TLSPSKFILE 和 TLSPSKVALUE，则 TLSPSKVALUE 的值将写入 TLSPSKFILE 指定的文件中。 示例： `TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952`
TLSSERVERCERTISSUER	允许的服务器（proxy）证书颁发者。
TLSSERVERCERTSUBJECT	允许的服务器（proxy）证书主题。
UNSAFEUSERPARAMETERS	允许将所有字符作为参数传递给用户自定义参数。

示例

以下示例安装带有自定义配置的 Zabbix agent。它还会启用 TLS 支持使用预共享密钥。

```
mkdir "C:\Program Files\Zabbix Agent" 2>nul
msiexec.exe /l*v "C:\package.log" /i "C:\zabbix_agent-8.0.0-windows-amd64-openssl.msi" /qn+^
 SERVER=192.0.2.1^
 INSTALLFOLDER="C:\Program Files\Zabbix Agent"^
 HOSTNAME=LAPTOP-IKP7S51S^
 TLSACCEPT=psk^
 TLSCONNECT=psk^
 TLSPSKIDENTITY="PSK 001"^
 TLSPSKFILE="C:\Program Files\Zabbix Agent\psk.key"^
 TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952^
 ENABLEPATH=1^
 ALLOWDENYKEY="AllowKey=system.run[type C:\Windows\System32\drivers\etc\hosts];DenyKey=system.run[*]"
```

下一个示例安装较新版本的 Zabbix agent，并使用一个 模板配置文件（`CONF="C:\agent-template.conf"`）。在安装过程中，此文件将成为 agent 配置文件。若要继承旧配置文件中的参数，请使用参数占位符（例如，`[AllowDenyKey]`）。

```
msiexec.exe /l*v "C:\package.log" /i "C:\zabbix_agent-8.0.1-windows-amd64-openssl.msi" /qn+ NONMSICONFNAME

#### agent-template.conf 示例:
LogFile=[LogFile]
[AllowDenyKey]
Server=192.0.2.8
Hostname=DESKTOP-X9F4A2B
[Include]
[TLSConnect]
[TLSAccept]
[TLSPSKIdentity]
[TLSPSKFile]
```

或者，您也可以使用 自定义配置文件（NONMSICONFNAME="C:\agent-custom.conf"）。在安装过程中，此文件中存在的任何有效 agent 配置参数（仅限于上表中列出的参数）都将写入新创建的 agent 配置文件中。若要保留现有的 agent 配置，请定义需要保留的参数。

```
msiexec.exe /l*v "C:\package.log" /i "C:\zabbix_agent-8.0.1-windows-amd64-openssl.msi" /qn+ NONMSICONFNAME

#### agent-custom.conf 示例:
Server=192.0.2.8
Hostname=DESKTOP-X9F4A2B
```

Zabbix agent 2 可加载插件

Zabbix agent 2 的可加载插件可在 Windows 上使用 64 位 MSI 安装程序包进行安装，这些安装包可通过下载获取。

MSI 安装的最低操作系统要求为 Windows 10（64 位）或 Windows Server 2016。

Attention:
安装插件之前，请先查看其 README 文件。其中可能包含特定要求和安装说明。

与 Zabbix agent/agent2 类似，可加载插件也可以使用安装向导或命令行进行安装。

通过安装向导安装

1. 双击已下载的 MSI 文件以开始安装。

2. 接受最终用户许可协议。

3. 选择要安装的 Zabbix agent 2 可加载插件。

Attention:
建议使用安装程序提供的默认安装位置。如果在缺少必要权限的情况下使用自定义位置，可能会影响安装的安全性。

4. 单击 *Install* 开始安装。

所有选定的 Zabbix agent 2 可加载插件都将安装到您指定的位置（默认：C:\Program Files\Zabbix Agent 2），其配置文件将安装在 zabbix_agent2.d 子文件夹中。

5. 单击 *Finish* 按钮退出安装向导。

从命令行安装

可以通过命令行运行 MSI 安装程序，并使用 msiexec 来安装 Zabbix agent 2 可加载插件。例如：

```
msiexec.exe /l*v "C:\package.log" /i "C:\zabbix_agent2_plugins-8.0.0-windows-amd64.msi" /qn+
```

Zabbix agent 2 可加载插件的 MSI 安装程序包支持以下参数。

Parameter	Description
ADDDEFAULT	要按默认配置安装的、以逗号分隔的组件列表。更多信息，请参见 ADDDEFAULT property。 可能的值：ALL、CephPlugin、EmberplusPlugin、MongodbPlugin、MssqlPlugin、NvidiagpuPlugin、PostgresqlPlugin 示例：ADDDEFAULT=MongodbPlugin,PostgresqlPlugin
ADDLOCAL	要在本地安装的、以逗号分隔的组件列表。更多信息，请参见 ADDLOCAL property。 可能的值：ALL、CephPlugin、EmberplusPlugin、MongodbPlugin、MssqlPlugin、NvidiagpuPlugin、PostgresqlPlugin 示例：ADDLOCAL=MongodbPlugin,MssqlPlugin
INSTALLFOLDER	Zabbix 组件的完整路径名，组件将安装到该文件夹中，其配置文件将安装在 zabbix_agent2.d 子文件夹中。 示例：INSTALLFOLDER="C:\Program Files\Zabbix Agent 2"

在 **Windows** 上构建 **Zabbix agent 2**

概述

本页介绍如何在 Windows 10（64 位或 32 位）上从源代码构建 Zabbix agent 2。

可以在 64 位平台上构建 32 位和 64 位版本，但在 32 位平台上只能构建 32 位版本。

构建 Zabbix agent 2 需要：

- MinGW 构建工具
- Go 编程语言
- OpenSSL（用于 Zabbix 中的加密功能）
- PCRE2（Perl 兼容正则表达式；用于 Zabbix 中的正则表达式模式匹配功能）

您可以使用以下方法之一构建 Zabbix agent 2：

- 使用 vcpkg - 使用 C++ 软件包管理器简化依赖项管理的自动化方法。
- 手动构建 - 需要在编译 agent 之前安装所有依赖项的手动方法。

Attention:
开始构建过程前，请注意以下事项：

- 要执行命令，请使用命令提示符，并由具有足够权限向受保护文件夹写入数据的用户启动。但是，在安装OpenSSL和PCRE2 时，请使用 MSYS2 终端。
- 建议在 C:\Zabbix 中创建工作目录，用于存放所有源文件和构建文件夹。但是，编译后的组件应安装在 C:\Zabbix\x64 中（对于 32 位构建，则安装在 C:\Zabbix\x86 中）。

使用 vcpkg 构建 Zabbix agent 2

本节介绍如何使用 vcpkg 构建 Zabbix agent。vcpkg 是一个可简化依赖项管理以及与 C++ 项目集成的软件包管理器。

1. 下载并安装 Visual Studio 2022 生成工具。安装过程中，请确保选择 使用 C++ 的桌面开发工作负载，其中包含 vcpkg 软件包管理器。

2. 下载并安装 Go（提供 MSI 安装程序）。安装过程中，请确保将 C:\Zabbix\Go 指定为安装文件夹。

3. 下载使用 Microsoft Visual C 运行时库的 MinGW 发行版，例如：

- 对于 64 位构建：x86_64-15.1.0-release-win32-seh-msvcrt-rt_v12-rev0.7z
- 对于 32 位构建：i686-15.1.0-release-win32-dwarf-msvcrt-rt_v12-rev0.7z

Attention:
MinGW 版本必须满足最低要求。

然后，将其解压到 C:\Zabbix\mingw64（对于 32 位构建，则解压到 C:\Zabbix\mingw32）。

4. 初始化 vcpkg 并安装构建 Zabbix agent 2 所需的依赖项（请注意，这可能需要一些时间）：

```
cd C:\Zabbix

set PATH=%PATH%;"C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\VC\vcpkg"
vcpkg new --application
vcpkg add port pcre2
vcpkg add port libiconv
vcpkg add port openssl

#### 对于 64 位构建：
set PATH=C:\Zabbix\mingw64\bin;%PATH%
vcpkg install --triplet x64-mingw-static --x-install-root=x64

#### 对于 32 位构建：
set PATH=C:\Zabbix\mingw32\bin;%PATH%
vcpkg install --triplet x86-mingw-static --x-install-root=x86
```

5. 下载 Zabbix 源代码存档，并将其解压到 C:\Zabbix\zabbix-8.0.0。

6. 进入 Zabbix 构建目录（C:\Zabbix\zabbix-8.0.0\build\mingw），并创建以下 build.bat 脚本：

- 对于 64 位构建：

```
:: 为当前会话将 MinGW 和 Go 添加到系统 `PATH` 变量：
set PATH=C:\Zabbix\mingw64\bin;%PATH%
set PATH=C:\Zabbix\Go\bin;%PATH%

:: 设置 vcpkg 安装路径：
set vcpkg="C:\Zabbix\x64\x64-mingw-static"

:: 设置 Crypt32 库的链接器标志：
SET CGO_LDFLAGS="-lCrypt32"
```

```
:: 运行构建过程：
mingw32-make GOFLAGS="-buildvcs=false" ARCH=AMD64 ^
    PCRE2="%vcpkg%" ^
    OPENSSL="%vcpkg%" ^
    all
```

- 对于 32 位构建：

```
:: 为当前会话将 MinGW 和 Go 添加到系统 `PATH` 变量：
set PATH=C:\Zabbix\mingw32\bin;%PATH%
set PATH=C:\Zabbix\Go\bin;%PATH%

:: 设置 vcpkg 安装路径：
set vcpkg="C:\Zabbix\x86\x86-mingw-static"

:: 设置 Crypt32 库的链接器标志：
SET CGO_LDFLAGS="-lCrypt32"

:: 运行构建过程：
mingw32-make GOFLAGS="-buildvcs=false" ARCH=x86 ^
    PCRE2="%vcpkg%" ^
    OPENSSL="%vcpkg%" ^
    all
```

7. 执行脚本以编译 Zabbix agent 2：

```
build.bat
```

编译完成后，Zabbix agent 2 二进制文件将位于 C:\Zabbix\zabbix-8.0.0\bin\win64(对于 64 位构建)或 C:\Zabbix\zabbix-8.0.0\bi
(对于 32 位构建)。Zabbix agent 2 配置文件位于 C:\Zabbix\zabbix-8.0.0\src\go\conf。

要运行 agent，请将 zabbix_agent2.exe 及其配置文件复制到专用文件夹（例如 C:\Zabbix\agent2），然后运行 agent：

```
mkdir C:\Zabbix\agent2

#### 对于 64 位构建：
copy C:\Zabbix\zabbix-8.0.0\bin\win64\zabbix_agent2.exe C:\Zabbix\agent2\

#### 对于 32 位构建：
copy C:\Zabbix\zabbix-8.0.0\bin\win32\zabbix_agent2.exe C:\Zabbix\agent2\

copy C:\Zabbix\zabbix-8.0.0\src\go\conf\zabbix_agent2.conf C:\Zabbix\agent2\
xcopy /E /I C:\Zabbix\zabbix-8.0.0\src\go\conf\zabbix_agent2.d C:\Zabbix\agent2\zabbix_agent2.d\

C:\Zabbix\agent2\zabbix_agent2.exe -c C:\Zabbix\agent2\zabbix_agent2.conf
```

如有需要，请继续编译 Zabbix agent 2 的可加载插件。

编译 Zabbix agent 2 可加载插件

1. 下载与您的 Zabbix agent 2 版本匹配的 Zabbix 插件源代码（例如，zabbix-agent2-plugin-postgresql-8.0.0.tar.gz），并将其解压到 C:\Zabbix。

> **Attention:**
> 在安装插件之前，请先查看其 README 文件。其中可能包含特定要求和安装说明。

2. 进入解压后的插件目录并编译插件：

```
cd C:\Zabbix\zabbix-agent2-plugin-ember-plus-8.0.0

#### 对于 64 位构建：
mingw32-make ARCH=AMD64

#### 对于 32 位构建：
mingw32-make ARCH=x86
```

编译完成后，Zabbix agent 2 插件二进制文件及其配置文件将位于同一插件目录中。

插件可执行文件可以放在任意位置，只要 Zabbix agent 2 能够加载它即可。请在插件配置文件中指定插件二进制文件的路径，例如 PostgreSQL 插件的 postgresql.conf 中：

```
Plugins.PostgreSQL.System.Path=/path/to/executable/zabbix-agent2-plugin-postgresql
```

插件配置文件的路径必须在 Zabbix agent 2 配置文件的Include 参数中指定：

```
Include=/path/to/plugin/configuration/file/postgresql.conf
```

有关配置插件的更多详细信息，请继续查看设置。

手动构建 Zabbix agent 2

Attention:
这种构建 Zabbix agent 2 的方法适用于需要完全控制构建环境的用户，或处于无法使用 vcpkg的受限环境中的用户。

本节包含手动构建 Zabbix agent 2 的说明，包括安装所需的构建工具和依赖项，然后编译 agent。

设置构建工具

1. 下载并安装 MSYS2（提供 MSI 安装程序）。在安装过程中，请务必将安装目录指定为 C:\Zabbix\msys64。

2. 下载并安装 Go（提供 MSI 安装程序；请参见当前支持的Go 版本）。在安装过程中，请务必将安装目录指定为 C:\Zabbix\Go。

3. 下载使用 Microsoft Visual C 运行时库的 MinGW 发行版；例如：

- 对于 64 位构建：x86_64-15.1.0-release-win32-seh-msvcrt-rt_v12-rev0.7z
- 对于 32 位构建：i686-15.1.0-release-win32-dwarf-msvcrt-rt_v12-rev0.7z

Attention:
MinGW 版本必须满足 最低要求。

然后，将其解压到 C:\Zabbix\mingw64（32 位构建则解压到 C:\Zabbix\mingw32）。

安装 OpenSSL

Note:
要在不支持 TLS 的情况下编译 Zabbix agent，请继续执行安装 PCRE2部分。

1. 以管理员权限打开 MSYS2 MSYS 终端，并运行以下命令：

```
pacman -S perl-Locale-Maketext-Simple
pacman -S nasm
pacman -S make
pacman -S cmake
```

2. 下载 OpenSSL 源代码存档，并将其解压到 C:\Zabbix\openssl-3.5.0。

3. 导航到解压后的 OpenSSL 目录，并创建以下 build.sh 脚本：

- 对于 64 位构建：

```
####!/usr/bin/env bash

export PATH="/c/Zabbix/mingw64/bin:$PATH"
export d="/c/Zabbix/x64/OpenSSL-Win64-350-static"

perl Configure mingw64 no-shared no-capieng no-winstore no-srp no-gost no-dgram no-dtls1-method no-dtls1_2

make
make install
```

- 对于 32 位构建：

```
####!/usr/bin/env bash

export PATH="/c/Zabbix/mingw32/bin:$PATH"
export d="/c/Zabbix/x86/OpenSSL-Win64-350-static"

perl Configure mingw no-shared no-capieng no-winstore no-srp no-gost no-dgram no-dtls1-method no-dtls1_2-m
```

```
make
make install
```

> **Attention:**
> 请确保撤销非管理员用户对 C:\Zabbix\x64\OpenSSL-Win64-350-static 目录的写入权限。否则，agent 将从可由非特权用户修改的路径加载 SSL 设置，从而可能导致安全漏洞。

- no-shared 选项使 libcrypto.lib 和 libssl.lib OpenSSL 静态库自包含，因此 Zabbix 二进制文件无需外部 DLL 即可包含 OpenSSL。这意味着可以将 Zabbix 二进制文件复制到其他 Windows 计算机，而无需 OpenSSL 库；但是，发布新的 OpenSSL bugfix 版本后，需要重新编译 Zabbix agent。
- 不使用 no-shared 选项时，Zabbix 在运行时依赖 OpenSSL DLL。这意味着更新 OpenSSL 时可能无需重新编译 Zabbix agent；但是，将其复制到其他计算机时，也必须复制 OpenSSL DLL。

有关其他 OpenSSL 配置选项的更多信息，请参阅 OpenSSL 文档。

4. 执行脚本配置并安装 OpenSSL（请注意，此过程可能需要一些时间）：

```
cd /c/Zabbix/openssl-3.5.0
./build.sh
```

安装 PCRE2

1. 下载 PCRE2 源代码归档，并将其解压到 C:\Zabbix\pcre2-10.45。

2. 以管理员权限打开 MSYS2 MSYS 终端。然后，在已解压的 PCRE2 目录中创建一个 build 目录并进入该目录：

```
mkdir /c/Zabbix/pcre2-10.45/build
cd /c/Zabbix/pcre2-10.45/build
```

3. 配置 PCRE2：

```
#### For 64-bit builds:
export PATH="/c/Zabbix/mingw64/bin:$PATH"
cmake -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-O2 -g" -DCMAKE_INSTALL_PREFIX="/c/Zabbix/x64/PCRE2" ..

#### For 32-bit builds:
export PATH="/c/Zabbix/mingw32/bin:$PATH"
cmake -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-m32 -O2 -g" -DCMAKE_EXE_LINKER_FLAGS="-Wl,-mi386pe" -DCMAKE
```

> **Note:**
> 如果出现任何错误，建议在尝试重新执行 CMake 构建过程之前删除 CMake 缓存。缓存文件 (CMakeCachecache.txt) 位于已解压的 PCRE2 目录中的 build 目录内。

4. 安装 PCRE2：

```
make install
```

编译 Zabbix agent 2

1. 下载 Zabbix 源代码归档，并将其解压到 C:\Zabbix\zabbix-8.0.0。

如果需要从原始源代码仓库生成源代码归档（例如，应用自定义补丁或从最新源代码构建），请在安装了 Go 的 **Linux** 计算机上运行以下命令（配置 Zabbix agent 2 时需要 Go）：

```
git clone https://git.zabbix.com/scm/zbx/zabbix.git
cd zabbix
./bootstrap.sh
./configure
make dist
```

这将创建一个源代码归档，之后可以将其复制到 Windows 计算机。

2. 以管理员权限打开命令提示符。然后，导航到 Zabbix 构建目录并编译 Zabbix agent；请确保正确指定 OpenSSL 和 PCRE2 的安装目录：

- 对于 64 位构建：

```
cd C:\Zabbix\zabbix-8.0.0\build\mingw
set PATH=C:\Zabbix\mingw64\bin;%PATH%
mklink /D C:\Zabbix\x64\OpenSSL-Win64-350-static\lib C:\Zabbix\x64\OpenSSL-Win64-350-static\lib64
```

```
#### 支持 TLS:
mingw32-make ARCH=AMD64 PCRE2="C:\Zabbix\x64\PCRE2" OPENSSL="C:\Zabbix\x64\OpenSSL-Win64-350-static"

#### 不支持 TLS:
mingw32-make ARCH=AMD64 PCRE2="C:\Zabbix\x64\PCRE2"
```

- 对于 32 位构建：

```
cd C:\Zabbix\zabbix-8.0.0\build\mingw
set PATH=C:\Zabbix\mingw32\bin;%PATH%

#### 支持 TLS:
mingw32-make ARCH=x86 PCRE2="C:\Zabbix\x86\PCRE2" OPENSSL="C:\Zabbix\x86\OpenSSL-Win32-350-static"

#### 不支持 TLS:
mingw32-make ARCH=x86 PCRE2="C:\Zabbix\x86\PCRE2"
```

编译完成后，Zabbix agent 2 二进制文件将位于 C:\Zabbix\zabbix-8.0.0\bin\win64(对于 32 位构建，则位于 C:\Zabbix\zabbix-8.0.
Zabbix agent 2 配置文件位于 C:\Zabbix\zabbix-8.0.0\src\go\conf。

要运行 agent，请将 zabbix_agent2.exe 二进制文件及其配置文件复制到专用文件夹（例如 C:\Zabbix\agent2），然后运行 agent：

```
mkdir C:\Zabbix\agent2
copy C:\Zabbix\zabbix-8.0.0\bin\win64\zabbix_agent2.exe C:\Zabbix\agent2\
copy C:\Zabbix\zabbix-8.0.0\src\go\conf\zabbix_agent2.conf C:\Zabbix\agent2\
xcopy /E /I C:\Zabbix\zabbix-8.0.0\src\go\conf\zabbix_agent2.d C:\Zabbix\agent2\zabbix_agent2.d\

C:\Zabbix\agent2\zabbix_agent2.exe -c C:\Zabbix\agent2\zabbix_agent2.conf
```

如有需要，请继续阅读编译 Zabbix agent 2 可加载插件。

在 **Windows** 上构建 **Zabbix agent**

概述

本页介绍如何在 Windows 10（64 位）上从源代码构建 Zabbix agent。

这些说明适用于支持 Visual Studio 2022 的 Windows 版本。

构建 Zabbix agent 需要：

- C 编译器（包含在 Visual Studio 2022 的 Build Tools 中）
- OpenSSL（用于 Zabbix 的加密功能）
- PCRE2（Perl 兼容正则表达式；用于 Zabbix 中的正则表达式模式匹配功能）

您可以使用以下任一方法构建 Zabbix agent：

- 使用 vcpkg - 使用 C++ 包管理器简化依赖项管理的自动化方法。
- 手动构建 - 需要在编译 agent 前手动安装所有依赖项的方法。

根据您的监控需求，可能还需要其他库。有关更多信息，请参阅要求。

> **Attention:**
> 开始构建过程前，请注意以下事项：

> - 要执行命令，请使用 “x64 Native Tools Command Prompt”（包含在 Visual Studio 2022 的 Build Tools 中），并由具有足够权限向受保护文件夹写入文件的用户启动。
> - 建议在 C:\Zabbix 创建工作目录，用于存放所有源文件和构建文件夹。但是，编译后的组件应安装到 C:\Program Files\Zabbix\x64。

使用 vcpkg 构建 Zabbix agent

本节介绍如何使用 vcpkg 构建 Zabbix agent。vcpkg 是一个可简化依赖项管理以及与 C++ 项目集成的软件包管理器。

1. 下载并安装 Visual Studio 2022 生成工具。安装过程中，请确保选择 使用 C++ 的桌面开发工作负载，其中包含使用 vcpkg 构建 agent 所需的工具：

- C 编译器（Microsoft Visual C++）

- NMake 命令行工具
- vcpkg 软件包管理器
- “x64 Native Tools Command Prompt”

2. 以管理员权限打开 cmd.exe，初始化 vcpkg，并安装构建 Zabbix agent 所需的依赖项（请注意，这可能需要一些时间）：

```
cd C:\Zabbix
vcpkg new --application
vcpkg add port pcre2
vcpkg add port openssl
vcpkg install --triplet x64-windows-static --x-install-root="C:\Program Files\Zabbix\x64"
```

3. 下载 Zabbix 源代码存档，并将其解压到 C:\Zabbix\zabbix-8.0.0。

4. 导航到 Zabbix 构建目录（C:\Zabbix\zabbix-8.0.0\build\win32\project），并创建以下 build.bat 脚本；确保正确指定 OpenSSL 和 PCRE2 的安装目录：

```
:: Set vcpkg installation path:
set vcpkg=C:\Program Files\Zabbix\x64\x64-windows-static

:: Run the build process:
nmake -f Makefile CPU=AMD64 ^
    PCRE2INCDIR="%vcpkg%\include" ^
    PCRE2LIBDIR="%vcpkg%\lib" ^
    TLS=openssl ^
    TLSINCDIR="%vcpkg%\include" ^
    TLSLIBDIR="%vcpkg%\lib" ^
    LIBS="$(LIBS) Crypt32.lib" ^
    all
```

5. 打开 “x64 Native Tools Command Prompt”，执行脚本以编译 Zabbix agent：

```
cd C:\Zabbix\zabbix-8.0.0\build\win32\project
build.bat
```

如果打开的是其他命令提示符，请先初始化 Visual Studio 构建环境：

```
"%ProgramFiles(x86)%\Microsoft Visual Studio\2022\BuildTools\VC\Auxiliary\Build\vcvarsall.bat" x64
cd C:\Zabbix\zabbix-8.0.0\build\win32\project
build.bat
```

请将包含空格的路径放在引号中。

编译完成后，Zabbix 组件二进制文件将位于 C:\Zabbix\zabbix-8.0.0\bin\win64。Zabbix agent 配置文件位于 C:\Zabbix\zabbix-8.0.0

要运行 agent，请将 zabbix_agentd.exe 及其配置文件复制到专用文件夹（例如 C:\Zabbix\agent），然后运行 agent：

```
mkdir C:\Zabbix\agent
copy C:\Zabbix\zabbix-8.0.0\bin\win64\zabbix_agentd.exe C:\Zabbix\agent\
copy C:\Zabbix\zabbix-8.0.0\conf\zabbix_agentd.conf C:\Zabbix\agent\

C:\Zabbix\agent\zabbix_agentd.exe -c C:\Zabbix\agent\zabbix_agentd.conf
```

手动构建 Zabbix agent

> **Attention:**
> 这种构建 Zabbix agent 的方法适用于需要完全控制构建环境的用户，或处于受限环境中而无法使用 vcpkg的用户。

本节包含手动构建 Zabbix agent 的说明，包括安装所需的构建工具和依赖项（Perl、OpenSSL、PCRE2），然后编译 agent。

安装构建工具

1. 下载并安装 Build Tools for Visual Studio 2022。安装过程中，请确保选择 *Desktop development with C++* 工作负载，其中包含手动构建 agent 所需的工具：

- C 编译器（Microsoft Visual C++）
- NMake 命令行工具
- “x64 Native Tools Command Prompt”

安装 OpenSSL

Note:
要在不支持 TLS 的情况下编译 Zabbix agent，请继续阅读安装 PCRE2部分。

1. 下载并安装 Strawberry Perl（提供 MSI 安装程序）。安装期间，请确保将 C:\Zabbix\Strawberry 指定为安装文件夹。

2. 安装 Text::Template Perl 模块：

```
cpanm Text::Template
```

3. 验证 Netwide Assembler（NASM；编译 OpenSSL 所需）是否已在安装 Strawberry Perl 期间完成编译：

```
nasm -v
#### NASM version 2.16.01 compiled on May  3 2024
```

如果 NASM 未完成编译，请手动安装。有关更多信息，请参阅 NASM 文档。

4. 下载 OpenSSL 源代码存档，并将其解压到 C:\Zabbix\openssl-3.5.0。

5. 转到解压后的目录并配置 OpenSSL，例如：

```
cd C:\Zabbix\openssl-3.5.0
perl Configure VC-WIN64A no-shared no-capieng no-winstore no-srp no-gost no-dgram no-dtls1-method no-dtls1
```

Attention:
如果在 Windows 上编译 Zabbix agent 时为 OpenSSL 选择自定义目录（例如 C:\zabbix 或 C:\openssl-64bit），请确保撤销非管理员用户对该目录的写入权限。否则，agent 将从可由非特权用户修改的路径加载 SSL 设置，从而可能导致安全漏洞。

- no-shared 选项会使 libcrypto.lib 和 libssl.lib OpenSSL 静态库自包含，因此 Zabbix 二进制文件会包含 OpenSSL，无需外部 DLL。这意味着可以将 Zabbix 二进制文件复制到其他 Windows 计算机，而无需 OpenSSL 库；但是，当发布新的 OpenSSL bugfix 版本时，需要重新编译 Zabbix agent。
- 不使用 no-shared 选项时，Zabbix 在运行时依赖 OpenSSL DLL。这意味着 OpenSSL 更新可能不需要重新编译 Zabbix agent；但是，将其复制到其他计算机时，也必须复制 OpenSSL DLL。

有关其他 OpenSSL 配置选项的更多信息，请参阅 OpenSSL 文档。

6. 编译 OpenSSL 并运行测试（请注意，这可能需要一些时间）：

Attention:
请在没有管理员权限的情况下运行测试；否则，可能会导致意外结果或安全风险。如果某些测试失败，请参阅 OpenSSL 文档进行故障排除。

```
nmake
nmake test
...
All tests successful.
Files=325, Tests=3101, 822 wallclock secs ( 4.81 usr +  0.81 sys =  5.62 CPU)
Result: PASS
```

7. 安装 OpenSSL：

```
nmake install
```

如果只需安装软件组件（库、头文件，但不包括文档），可以使用 nmake install_sw。

安装 PCRE2

1. 下载并安装 CMake（提供 MSI 安装程序）。在安装过程中，请确保将安装文件夹指定为 C:\Zabbix\CMake，并选择 *Add CMake to the PATH environment variable* 选项。

2. 下载 PCRE2 源代码归档，并将其解压到 C:\Zabbix\pcre2-10.45。

3. 在解压后的 PCRE2 目录中创建 build 目录，并进入该目录：

```
mkdir C:\Zabbix\pcre2-10.45\build
cd C:\Zabbix\pcre2-10.45\build
```

4. 配置 PCRE2：

```
cmake -G "NMake Makefiles" -DPCRE_SUPPORT_UNICODE_PROPERTIES=ON -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL
```

Note:
如果发生任何错误，建议在尝试重新执行 CMake 构建过程之前删除 CMake 缓存。缓存（CMakeCachecache.txt）位于解压后的 PCRE2 目录中的 build 目录内。

5. 使用 NMake 构建 PCRE2：

```
nmake
```

6. 安装 PCRE2：

```
cmake --install .
```

编译 Zabbix agent

1. 下载 Zabbix 源代码归档，并将其解压到 C:\Zabbix\zabbix-8.0.0。

如果需要从原始源代码仓库生成源代码归档（例如，应用自定义补丁或从最新源代码构建），请在 **Linux** 计算机上运行以下命令：

```
git clone https://git.zabbix.com/scm/zbx/zabbix.git
cd zabbix
./bootstrap.sh
./configure
make dist
```

这将创建一个源代码归档，之后可以将其复制到 Windows 计算机。

2. 导航到 Zabbix 构建目录，打开 “x64 Native Tools Command Prompt”，然后编译 Zabbix agent（或其他组件）；请确保正确指定 OpenSSL 和 PCRE2 的安装目录：

```
"%ProgramFiles(x86)%\Microsoft Visual Studio\2022\BuildTools\VC\Auxiliary\Build\vcvarsall.bat" x64

cd C:\Zabbix\zabbix-8.0.0\build\win32\project

#### 使用 TLS 支持：
nmake /K -f Makefile_agent PCRE2INCDIR="C:\Program Files\Zabbix\x64\PCRE2\include" PCRE2LIBDIR="C:\Program
nmake /K -f Makefile_get PCRE2INCDIR="C:\Program Files\Zabbix\x64\PCRE2\include" PCRE2LIBDIR="C:\Program F
nmake /K -f Makefile_sender PCRE2INCDIR="C:\Program Files\Zabbix\x64\PCRE2\include" PCRE2LIBDIR="C:\Progra

#### 不使用 TLS 支持：
nmake /K -f Makefile_agent PCRE2INCDIR="C:\Program Files\Zabbix\x64\PCRE2\include" PCRE2LIBDIR="C:\Program
nmake /K -f Makefile_get PCRE2INCDIR="C:\Program Files\Zabbix\x64\PCRE2\include" PCRE2LIBDIR="C:\Program F
nmake /K -f Makefile_sender PCRE2INCDIR="C:\Program Files\Zabbix\x64\PCRE2\include" PCRE2LIBDIR="C:\Progra
```

包含空格的路径必须使用引号括起来。

编译完成后，Zabbix 组件二进制文件将位于 C:\Zabbix\zabbix-8.0.0\bin\win64。Zabbix agent 配置文件位于 C:\Zabbix\zabbix-8.0.0

要运行 agent，请将 zabbix_agentd.exe 及其配置文件复制到专用文件夹（例如 C:\Zabbix\agent），然后运行 agent：

```
mkdir C:\Zabbix\agent
copy C:\Zabbix\zabbix-8.0.0\bin\win64\zabbix_agentd.exe C:\Zabbix\agent\
copy C:\Zabbix\zabbix-8.0.0\conf\zabbix_agentd.conf C:\Zabbix\agent\

C:\Zabbix\agent\zabbix_agentd.exe -c C:\Zabbix\agent\zabbix_agentd.conf -f
```

从 **PKG** 安装 **macOS agent**

概述

可以使用 PKG 安装包在 macOS 上安装 Zabbix agent，这些安装包可通过 下载 获取。

Zabbix agent 软件包提供带加密 和不带加密两种版本。

安装 agent

可以使用图形用户界面或通过命令行安装 agent，例如：

```
sudo installer -pkg zabbix_agent-8.0.0-macos-arm64-openssl.pkg -target /
```

请确保在命令中使用正确的 Zabbix 软件包版本。它必须与已下载软件包的名称匹配。

运行 agent

agent 将在安装或重启后自动启动。

如有必要，您可以编辑位于 /usr/local/etc/zabbix/zabbix_agentd.conf 的配置文件。

要手动启动 agent，可运行：

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

要手动停止 agent：

```
sudo launchctl stop com.zabbix.zabbix_agentd
```

在升级过程中，现有配置文件不会被覆盖。系统会创建一个新的 zabbix_agentd.conf.NEW 文件，以便在必要时用于检查并更新现有配置文件。请记得在手动更改配置文件后重启 agent。

故障排查和移除 agent

本节列出了一些可用于排查问题和移除 Zabbix agent 安装的实用命令。

查看 Zabbix agent 是否正在运行：

```
ps aux | grep zabbix_agentd
```

查看 Zabbix agent 是否已通过软件包安装：

```
pkgutil --pkgs | grep zabbix
com.zabbix.pkg.ZabbixAgent
```

查看通过安装程序软件包安装的文件（请注意，此视图中不会显示开头的 /）：

```
pkgutil --only-files --files com.zabbix.pkg.ZabbixAgent
Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
usr/local/bin/zabbix_get
usr/local/bin/zabbix_sender
usr/local/etc/zabbix/zabbix_agentd/userparameter_examples.conf.NEW
usr/local/etc/zabbix/zabbix_agentd/userparameter_mysql.conf.NEW
usr/local/etc/zabbix/zabbix_agentd.conf.NEW
usr/local/sbin/zabbix_agentd
```

如果 Zabbix agent 是通过 launchctl 启动的，请停止它：

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

删除通过安装程序软件包安装的文件（包括配置和日志）：

```
sudo rm -f /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
sudo rm -f /usr/local/sbin/zabbix_agentd
sudo rm -f /usr/local/bin/zabbix_get
sudo rm -f /usr/local/bin/zabbix_sender
sudo rm -rf /usr/local/etc/zabbix
sudo rm -rf /var/log/zabbix
```

清除 Zabbix agent 的已安装记录：

```
sudo pkgutil --forget com.zabbix.pkg.ZabbixAgent
```

在 **macOS** 上构建 **Zabbix agent**

概述

本节演示如何从源代码构建带有或不带 TLS 的 Zabbix macOS agent 二进制文件。

先决条件

您将需要命令行开发工具（不要求 Xcode）、Automake、pkg-config 以及 PCRE（v8.x）或 PCRE2（v10.x）。如果您想构建带有 TLS 的 agent 二进制文件，还需要 OpenSSL 或 GnuTLS。

要安装 Automake 和 pkg-config，您需要从 https://brew.sh/ 获取 Homebrew 包管理器。要安装它，请打开终端并运行以下命令：

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

然后安装 Automake 和 pkg-config：

```
brew install automake
brew install pkg-config
```

准备 PCRE、OpenSSL 和 GnuTLS 库取决于它们将以何种方式链接到 agent。

如果您打算在已经安装了这些库的 macOS 机器上运行 agent 二进制文件，则可以使用 Homebrew 提供的预编译库。这通常是指那些使用 Homebrew 来构建 Zabbix agent 二进制文件或用于其他目的的 macOS 机器。

如果 agent 二进制文件将在没有这些库共享版本的 macOS 机器上使用，则应从源码编译静态库，并将 Zabbix agent 与这些静态库链接。

使用共享库构建 agent 二进制文件

安装 PCRE2（如有需要，请将以下命令中的 *pcre2* 替换为 *pcre*）：

```
brew install pcre2
```

如果使用 TLS 进行构建，请安装 OpenSSL 和/或 GnuTLS：

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

下载 Zabbix 源代码：

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

构建不带 TLS 的 agent：

```
cd zabbix
./bootstrap.sh
./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6
make
make install
```

构建带 OpenSSL 的 agent：

```
cd zabbix
./bootstrap.sh
./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-openssl=/usr/local/opt/
make
make install
```

构建带 GnuTLS 的 agent：

```
cd zabbix-source/
./bootstrap.sh
./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-gnutls=/usr/local/opt/g
make
make install
```

使用静态库构建不带 TLS 的 agent 二进制文件

假设 PCRE 静态库将安装在 `$HOME/static-libs` 中。这里我们将使用 PCRE2 10.39。

```
PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
```

下载并构建支持 Unicode 属性的 PCRE：

```
mkdir static-libs-source
cd static-libs-source
curl --remote-name https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.tar.gz
tar xf pcre2-10.39.tar.gz
cd pcre2-10.39
./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
make
make check
make install
```

下载 Zabbix 源代码并构建 agent：

```
git clone https://git.zabbix.com/scm/zbx/zabbix.git
cd zabbix
./bootstrap.sh
./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFIX"
make
```

```
make install
```

使用 OpenSSL 静态库构建 agent 二进制文件

构建 OpenSSL 时，建议在成功完成构建后运行 `make test`。即使构建成功，测试有时也可能失败。如果出现这种情况，应在继续之前研究并解决相关问题。

假设 PCRE 和 OpenSSL 静态库将安装到 `$HOME/static-libs`。这里我们将使用 PCRE2 10.39 和 OpenSSL 1.1.1a。

```
PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
OPENSSL_PREFIX="$HOME/static-libs/openssl-1.1.1a"
```

让我们在 `static-libs-source` 中构建静态库：

```
mkdir static-libs-source
cd static-libs-source
```

下载并构建启用了 Unicode 属性支持的 PCRE：

```
curl --remote-name https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.tar.gz
tar xf pcre2-10.39.tar.gz
cd pcre2-10.39
./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
make
make check
make install
cd ..
```

下载并构建 OpenSSL：

```
curl --remote-name https://www.openssl.org/source/openssl-1.1.1a.tar.gz
tar xf openssl-1.1.1a.tar.gz
cd openssl-1.1.1a
./Configure --prefix="$OPENSSL_PREFIX" --openssldir="$OPENSSL_PREFIX" --api=1.1.0 no-shared no-capieng no-
make
make test
make install_sw
cd ..
```

下载 Zabbix 源代码并构建 agent：

```
git clone https://git.zabbix.com/scm/zbx/zabbix.git
cd zabbix
./bootstrap.sh
./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFIX"
make
make install
```

使用 GnuTLS 和静态库构建 agent 二进制文件

GnuTLS 依赖于 Nettle 加密后端和 GMP 算术库。本指南将使用 Nettle 中包含的 mini-gmp，而不是完整的 GMP 库。

在构建 GnuTLS 和 Nettle 时，建议在成功构建后运行 `make check`。即使构建成功，测试有时也会失败。如果出现这种情况，应在继续之前研究并解决相关问题。

假设 PCRE、Nettle 和 GnuTLS 静态库将安装到 `$HOME/static-libs` 中。我们将使用 PCRE2 10.39、Nettle 3.4.1 和 GnuTLS 3.6.5。

```
PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
NETTLE_PREFIX="$HOME/static-libs/nettle-3.4.1"
GNUTLS_PREFIX="$HOME/static-libs/gnutls-3.6.5"
```

让我们在 `static-libs-source` 中构建静态库：

```
mkdir static-libs-source
cd static-libs-source
```

下载并构建 Nettle：

```
curl --remote-name https://ftp.gnu.org/gnu/nettle/nettle-3.4.1.tar.gz
tar xf nettle-3.4.1.tar.gz
cd nettle-3.4.1
./configure --prefix="$NETTLE_PREFIX" --enable-static --disable-shared --disable-documentation --disable-a
make
```

```
make check
make install
cd ..
```

下载并构建 GnuTLS：

```
curl --remote-name https://www.gnupg.org/ftp/gcrypt/gnutls/v3.6/gnutls-3.6.5.tar.xz
tar xf gnutls-3.6.5.tar.xz
cd gnutls-3.6.5
PKG_CONFIG_PATH="$NETTLE_PREFIX/lib/pkgconfig" ./configure --prefix="$GNUTLS_PREFIX" --enable-static --dis
make
make check
make install
cd ..
```

下载 Zabbix 源代码并构建 agent：

```
git clone https://git.zabbix.com/scm/zbx/zabbix.git
cd zabbix
./bootstrap.sh
CFLAGS="-Wno-unused-command-line-argument -framework Foundation -framework Security" \
> LIBS="-lgnutls -lhogweed -lnettle" \
> LDFLAGS="-L$GNUTLS_PREFIX/lib -L$NETTLE_PREFIX/lib" \
> ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFI
make
make install
```

Zabbix 服务器 **AWS** 部署指南

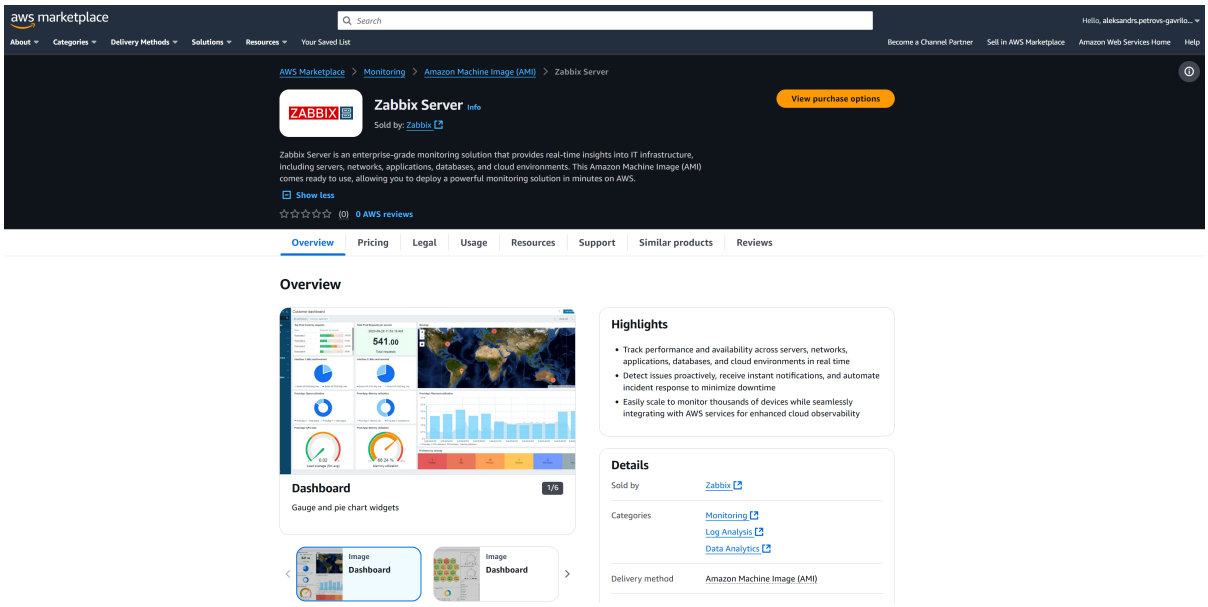
概述

本页面介绍如何使用 AWS 部署 Zabbix 服务器。

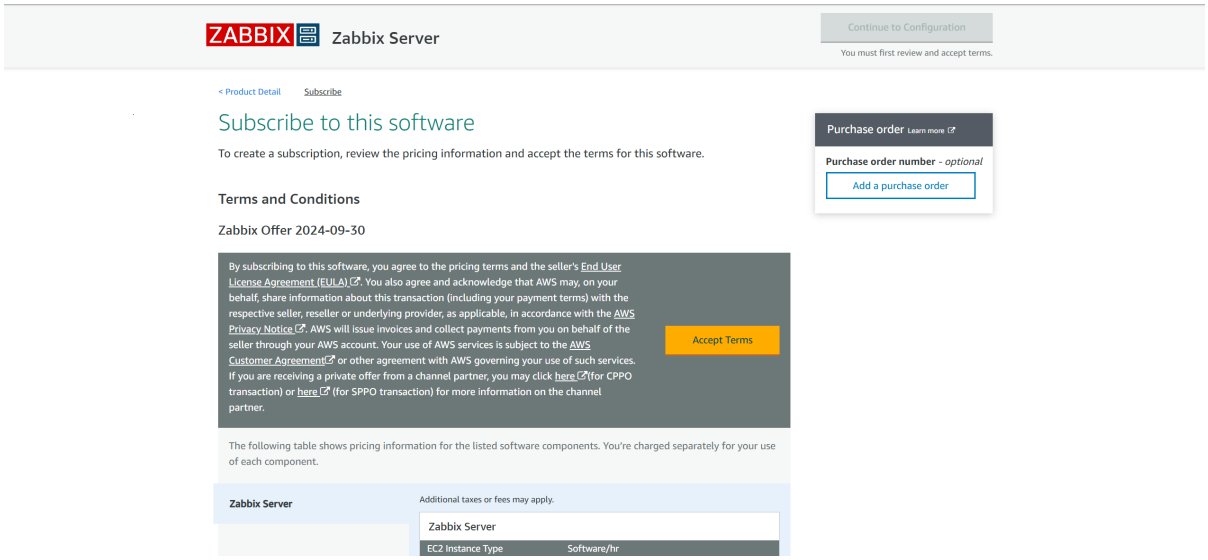
> **Attention:**
> 该平台支持部署多个 Zabbix 版本。请确保阅读与您部署的 Zabbix 版本相同的文档。您可以从版本下拉菜单中选择不同 Zabbix 版本的文档。

部署 appliance

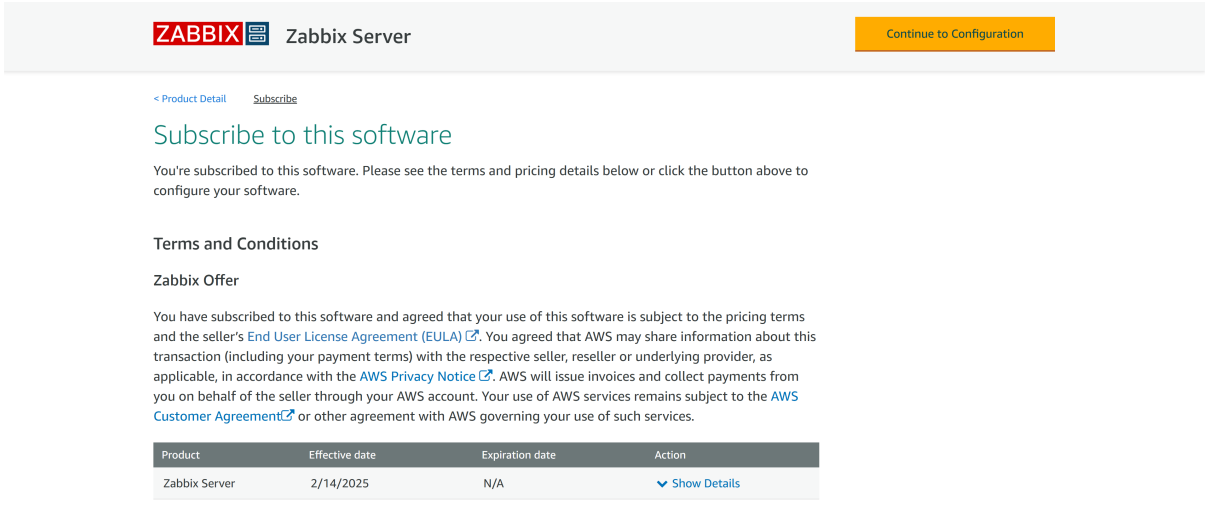
要部署 Zabbix appliance，首先访问 Zabbix AWS Marketplace 页面。接下来，点击 *View purchase options* 以部署映像，并进入订阅页面。



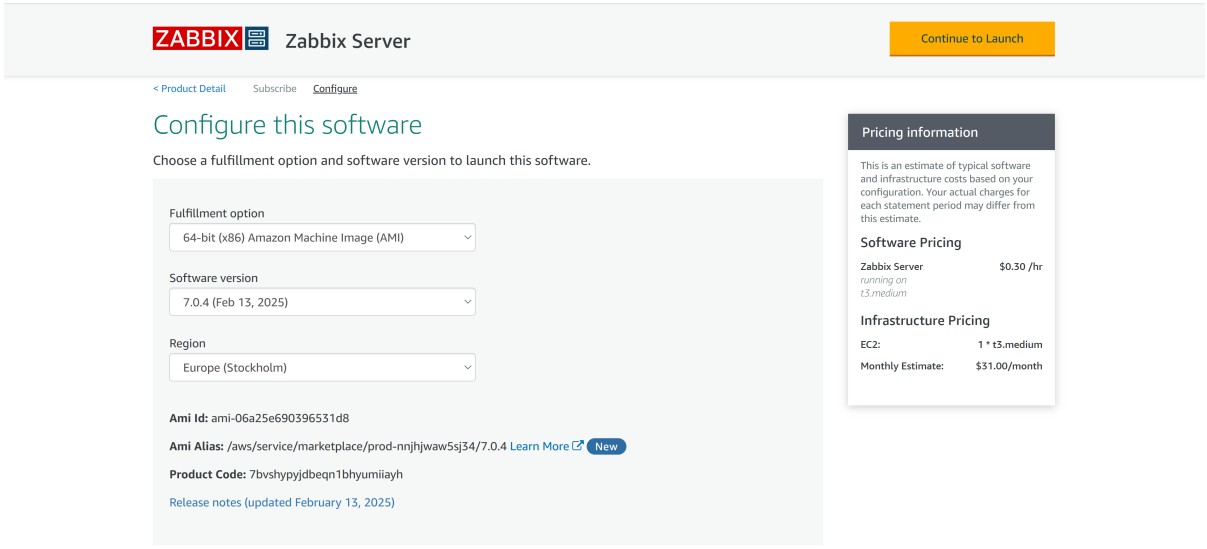
点击 *Accept terms* 按钮以接受 *Terms and Conditions*。



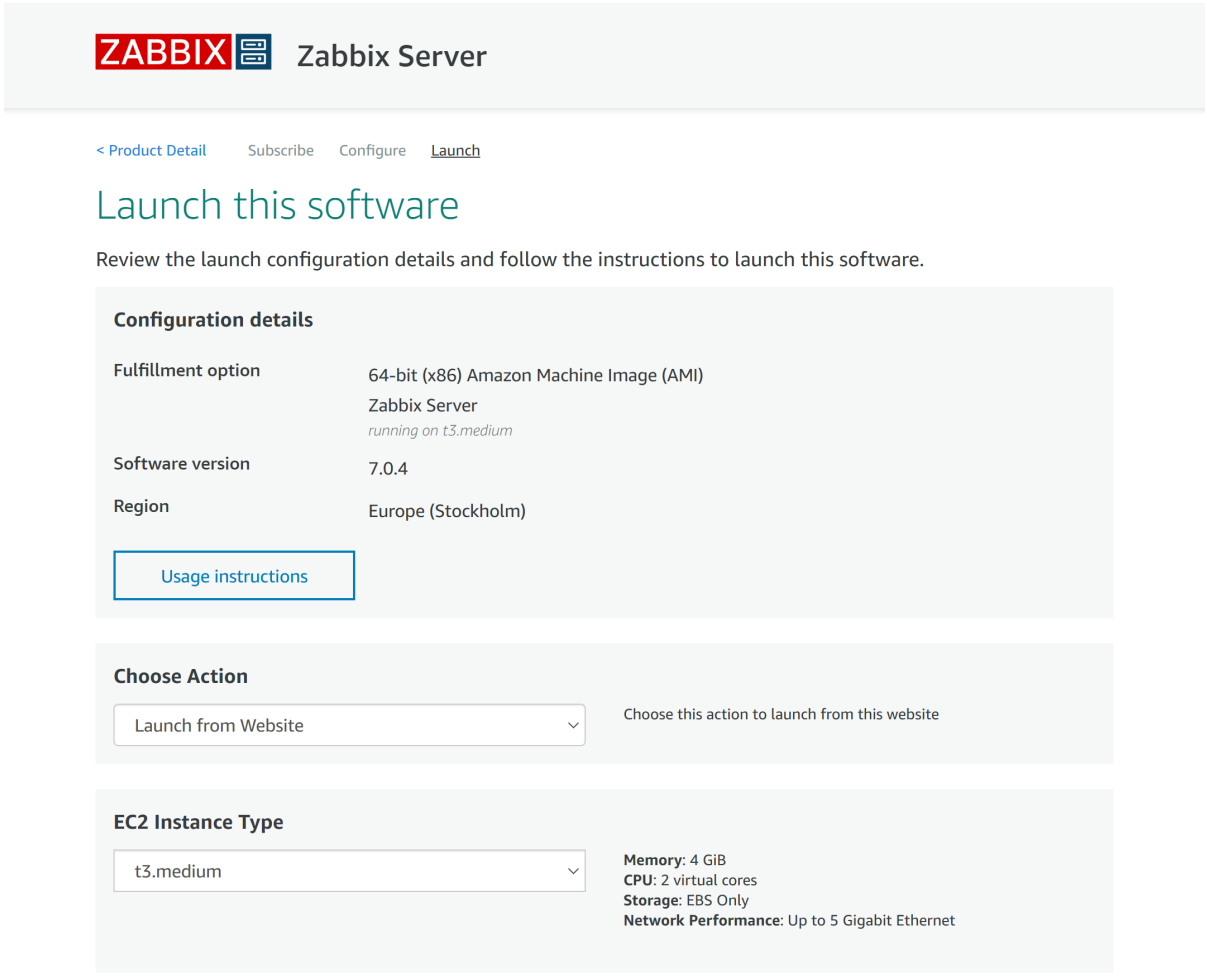
点击 *Continue to Configuration* 按钮，进入 Zabbix appliance 的 Configuration 部分。



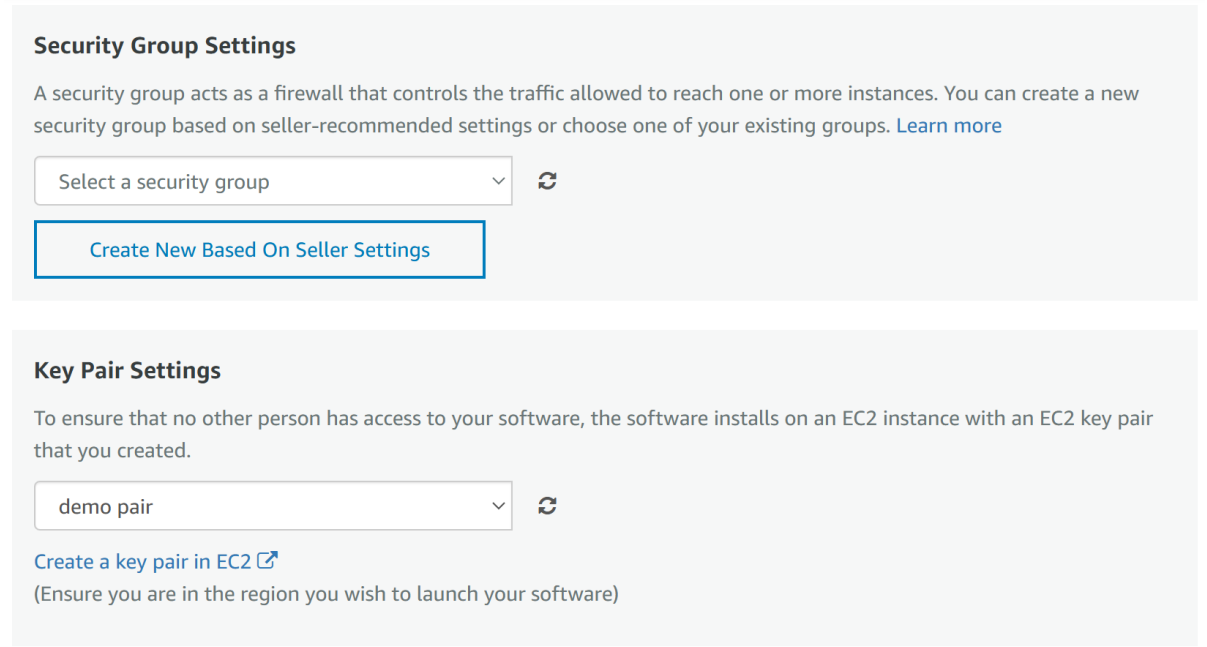
选择 Fulfilment 选项、Zabbix 版本以及首选 Region。



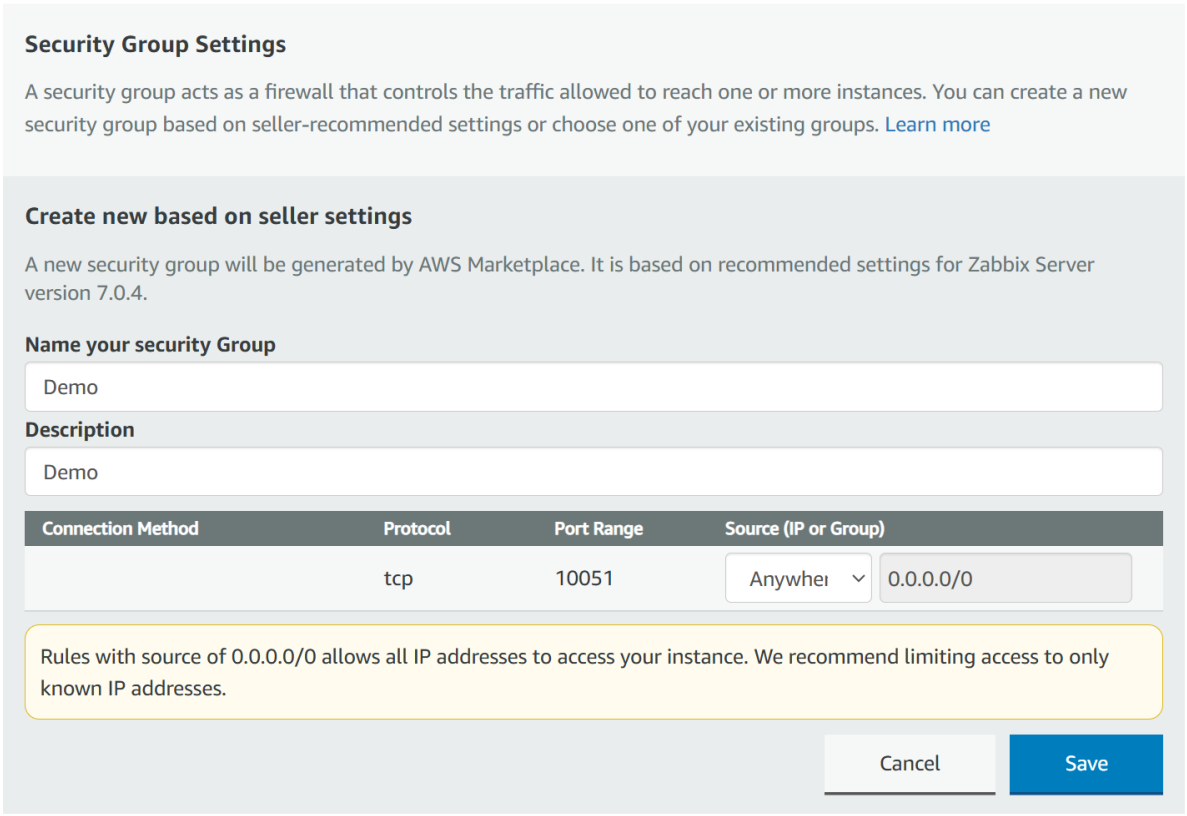
点击 *Continue to Launch* 按钮。在 Launch 页面上，选择首选的 Launch 选项、EC2 Instance Type、VPC 和 Subnet 设置。



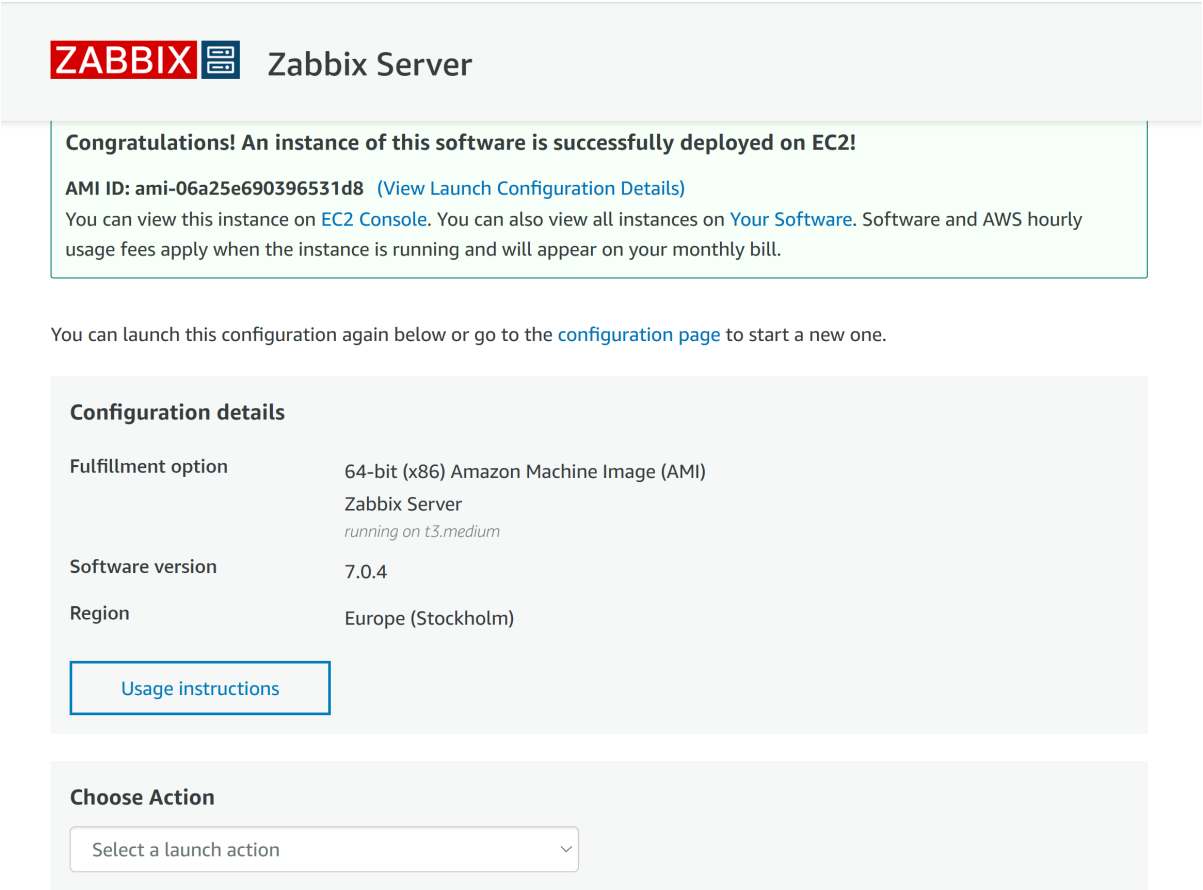
在 Launch 页面底部，指定 Security Group 设置，并选择或创建 EC2 密钥对 - 请务必将私钥保存在安全的位置！



如果选择 *Create a new Security Group based on seller settings*，将允许 TCP 端口 10051（Zabbix 服务器）通过。注意：如果希望通过 SSH 和 HTTP/HTTPS（前端）访问实例，还需要在 Security Group 设置中允许 SSH 和 HTTP/HTTPS 连接。

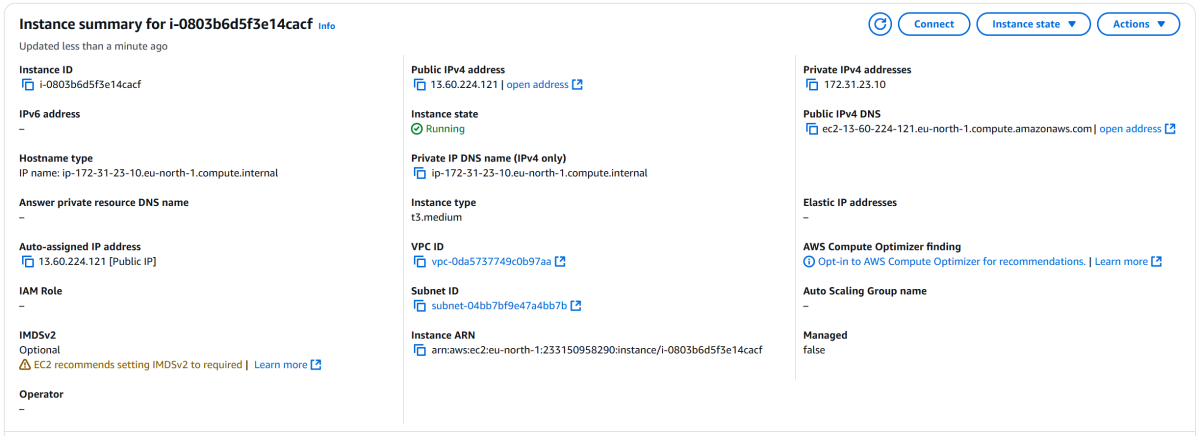


恭喜，您的 Zabbix appliance 已成功部署。

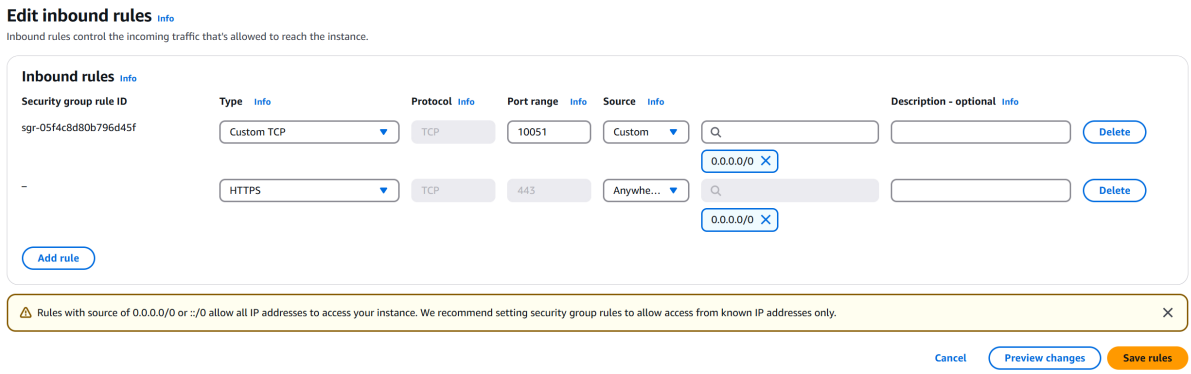


访问设备

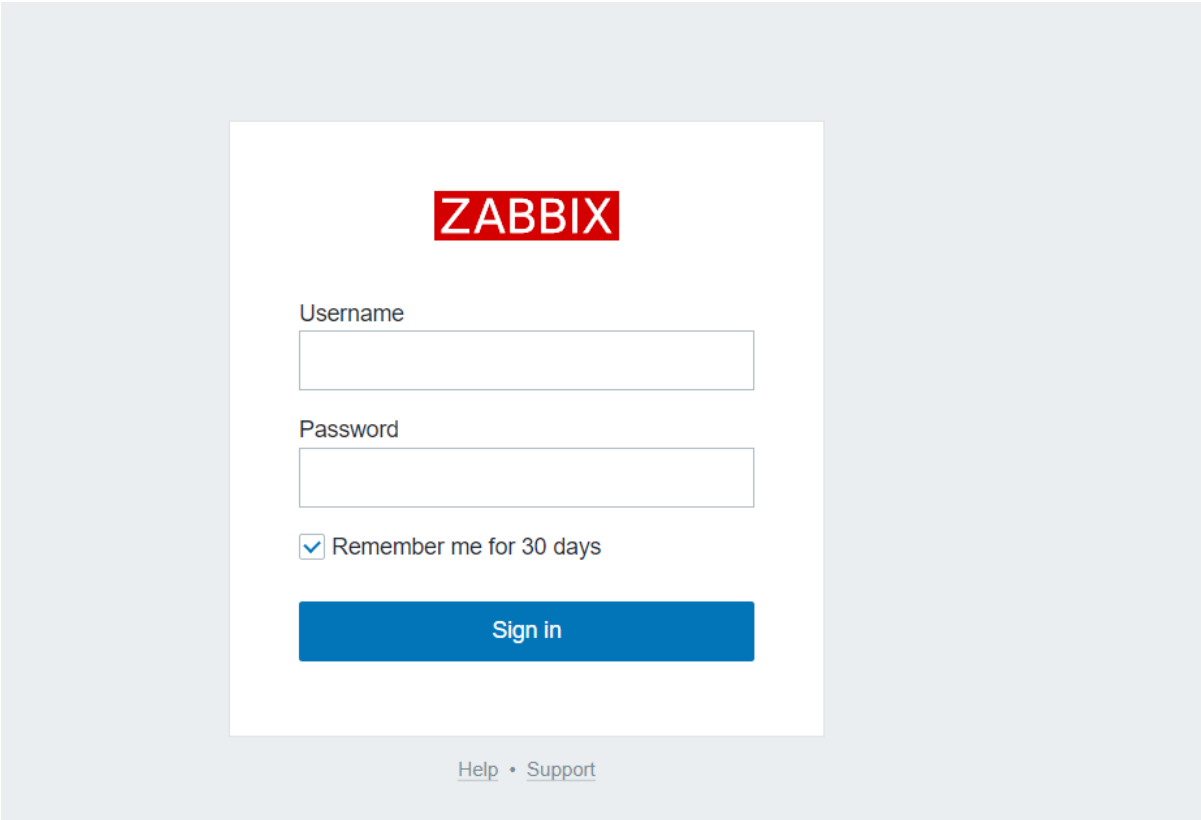
打开 *Instances* 部分，然后选择新部署的 Zabbix 设备实例。



要使用实例的 Public IPv4 地址或 Public IPv4 DNS 访问 Zabbix 前端，请在 Security Group 中允许端口 443 上的入站 HTTPS 流量。



现在，我们可以通过打开 Zabbix 设备的 Public IPv4 地址或 Public IPv4 DNS 来访问 Zabbix 前端。



要获取密码，我们需要通过串行控制台或 SSH 连接到计算机。默认的 SSH 用户名为 rocky。请不要忘记使用之前选择或生成的密钥对。请注意，生成凭据可能需要一些时间。凭据生成可能需要几分钟。请等待几分钟后再尝试连接。

```
Last login: Tue Nov 16 15:16:07 2021 from ec2-13-48-4-200.eu-north-1.compute.amazonaws.com
********************************************************************************

Zabbix frontend credentials:

Username: Admin

Password: gJ8f7XpgJbsnc2BU48g

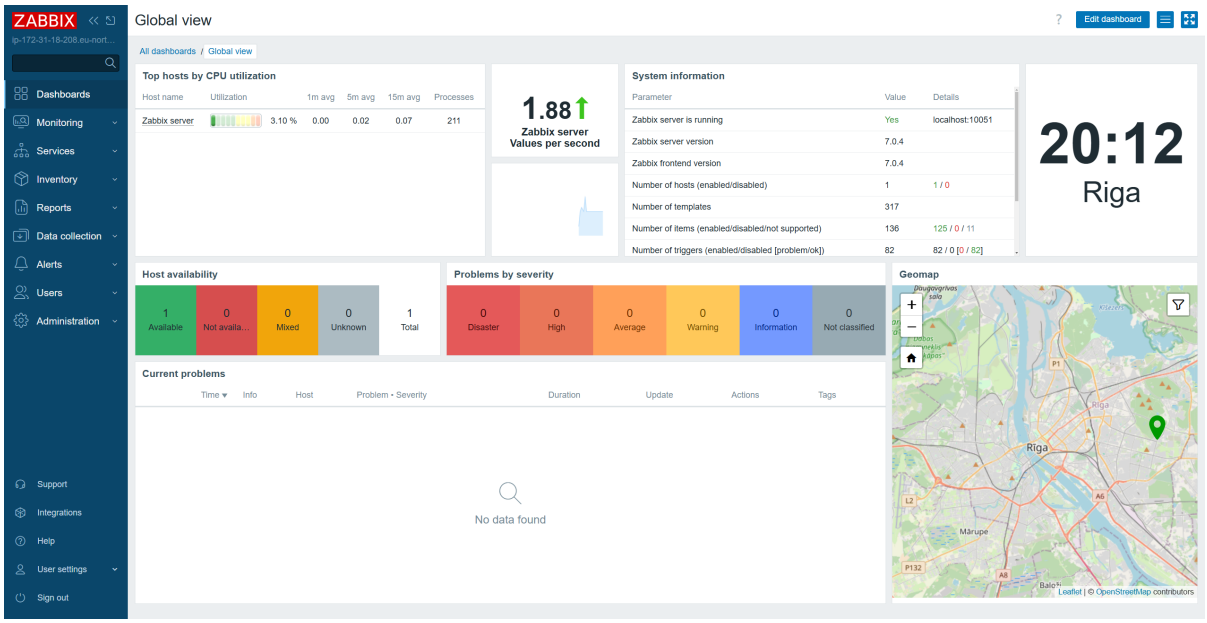
To learn about available professional services, including technical suppport and training, please visit https://www.zabbix.com/services

Official Zabbix documentation available at https://www.zabbix.com/documentation/current/

Note! Do not forget to change timezone PHP variable in /etc/php.d/99-zabbix.ini file.

********************************************************************************
```

现在，您可以使用这些信息登录 Zabbix 前端。



访问数据库

您可以使用 `root` 用户访问 Zabbix 数据库，无需输入密码。

```
[root@ip-172-31-29-138 ~]# mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 124
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

凭据存储在 `/root/.my.cnf` 配置文件中。

可以在 *Administration* > *General* > *GUI* 中更改默认的 Zabbix 前端时区

Zabbix proxy 的 **AWS** 部署指南

概述

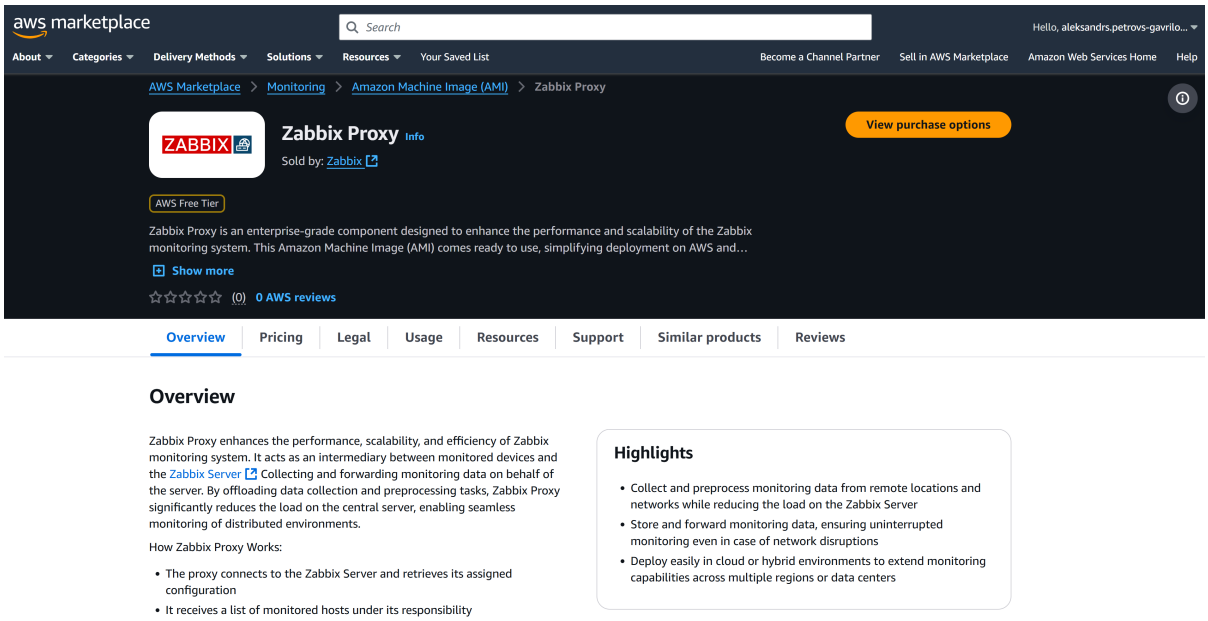
本页面介绍如何使用 AWS 部署 Zabbix proxy。

Attention:
该平台支持部署多个 Zabbix 版本。请务必阅读与您部署的 Zabbix 版本相同的文档。您可以从版本下拉菜单中选择不同 Zabbix 版本的文档。

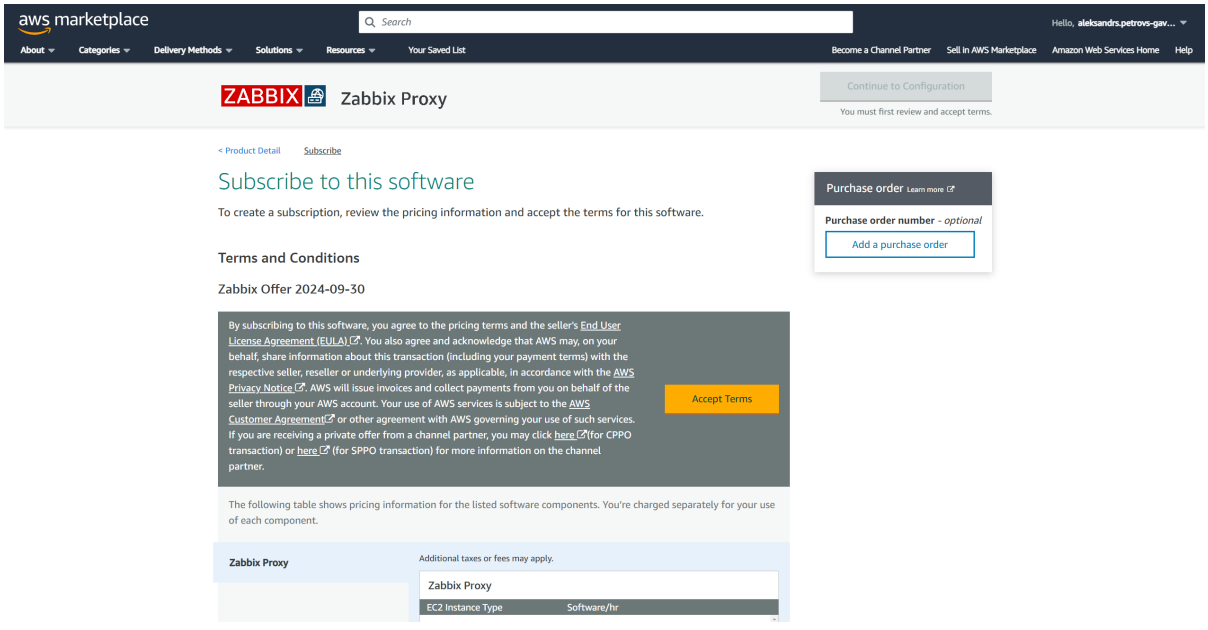
部署 Zabbix proxy

Attention:
使用 Zabbix proxy 组件需要一个正常运行的 Zabbix 服务器实例。

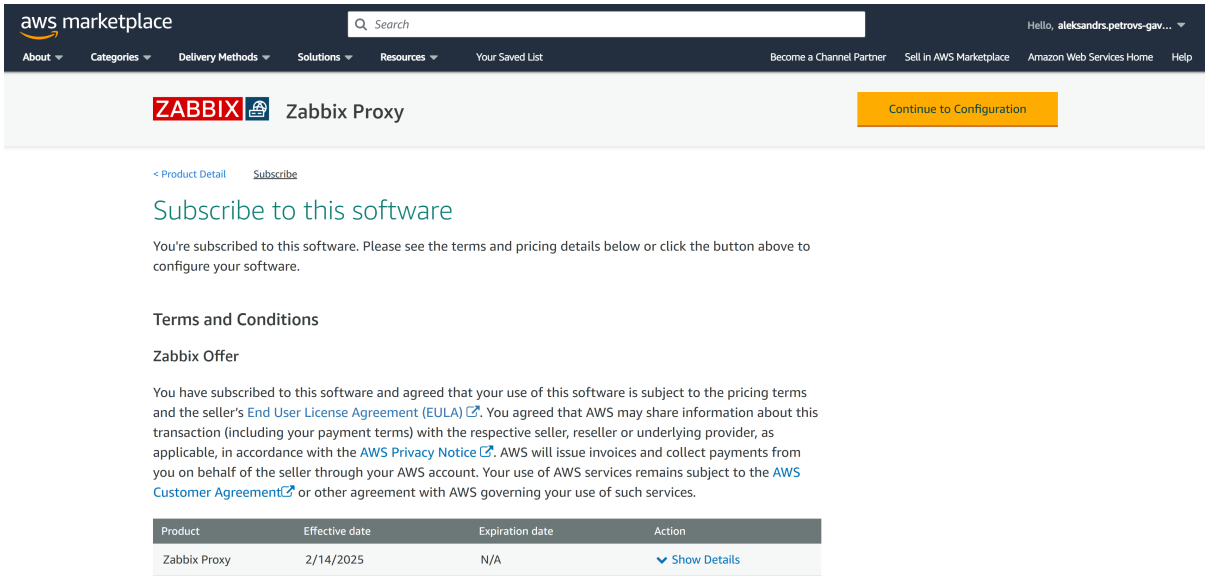
要部署 Zabbix proxy，首先访问 Zabbix AWS marketplace 页面。接下来，单击 *View purchase options* 以部署映像，然后进入订阅页面。



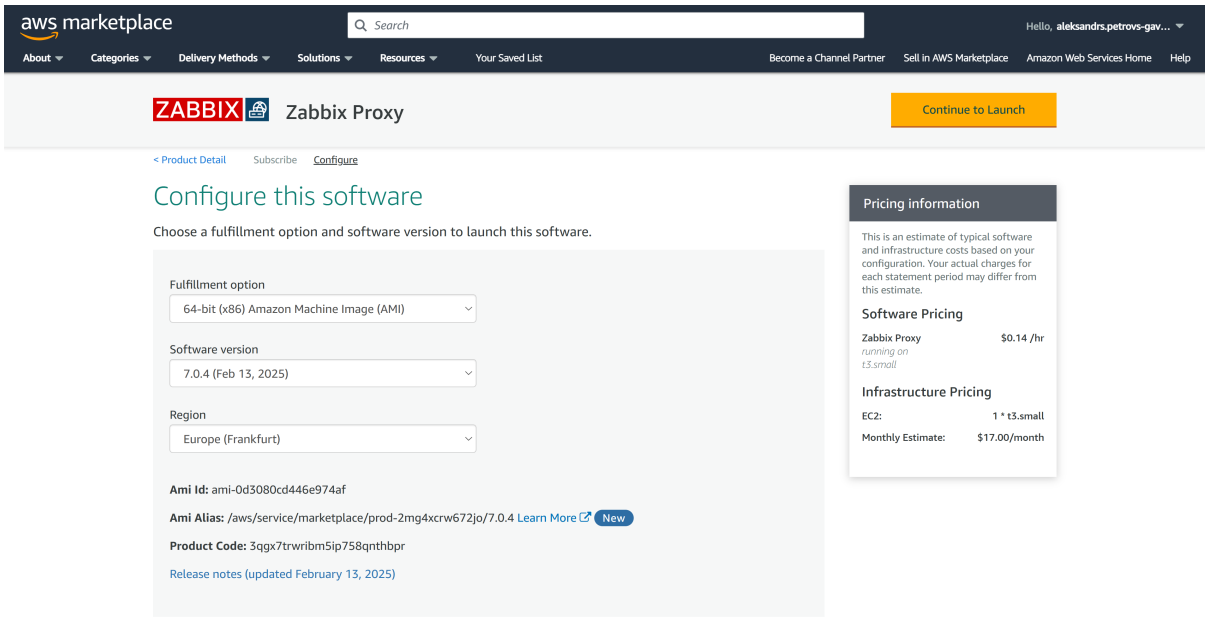
单击 *Accept terms* 按钮以接受 *Terms and Conditions*。



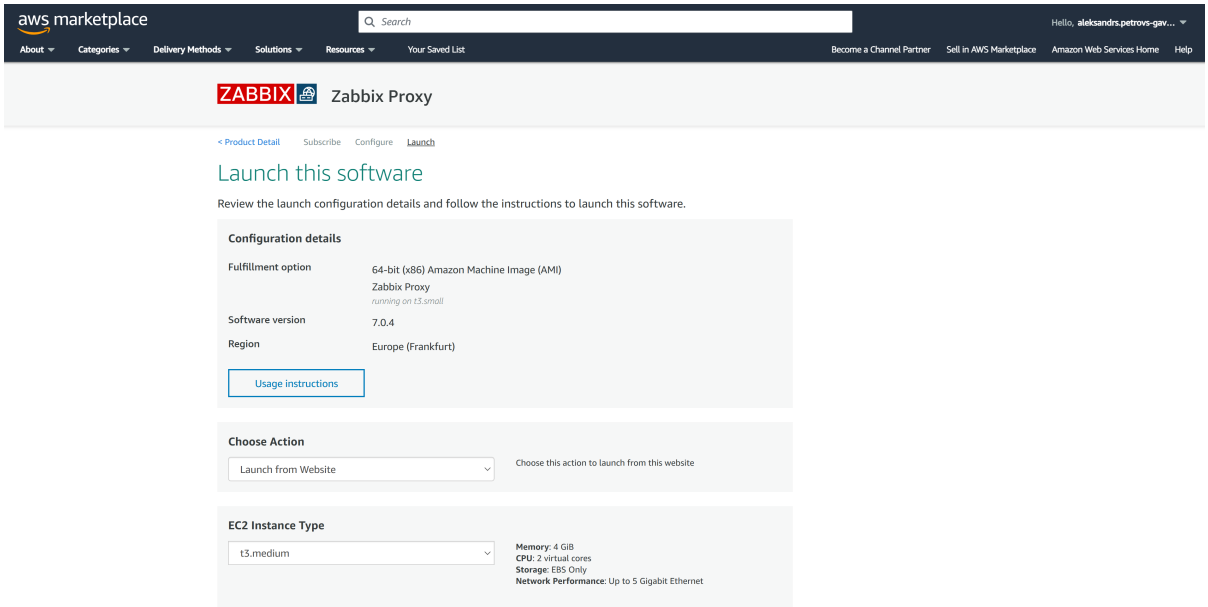
单击 *Continue to Configuration* 按钮，进入 Zabbix proxy 的配置部分。



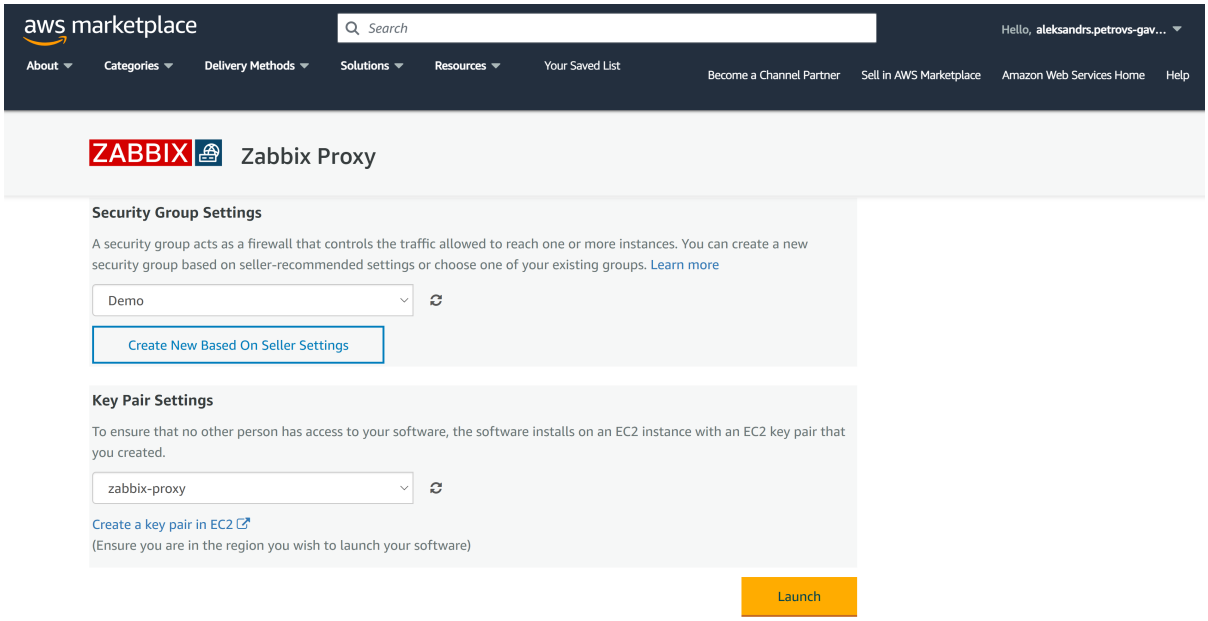
选择 Fulfilment 选项、Zabbix 版本以及首选区域。



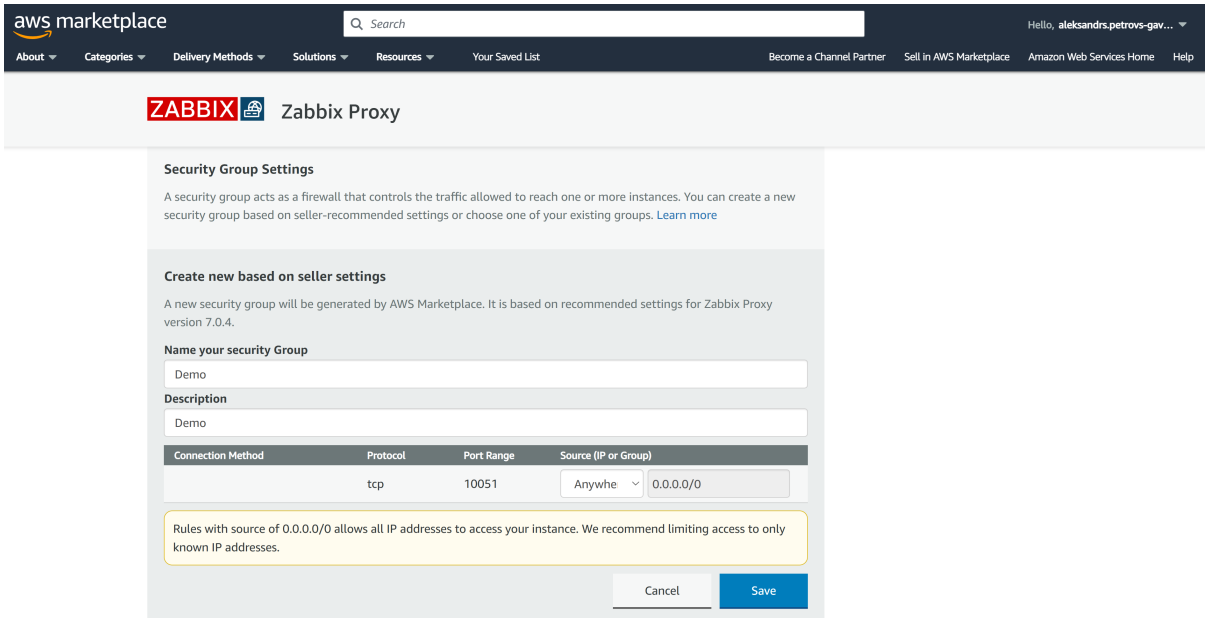
单击 *Continue to Launch* 按钮。在 Launch 页面上，选择首选的 Launch 选项、EC2 实例类型、VPC 和子网设置。



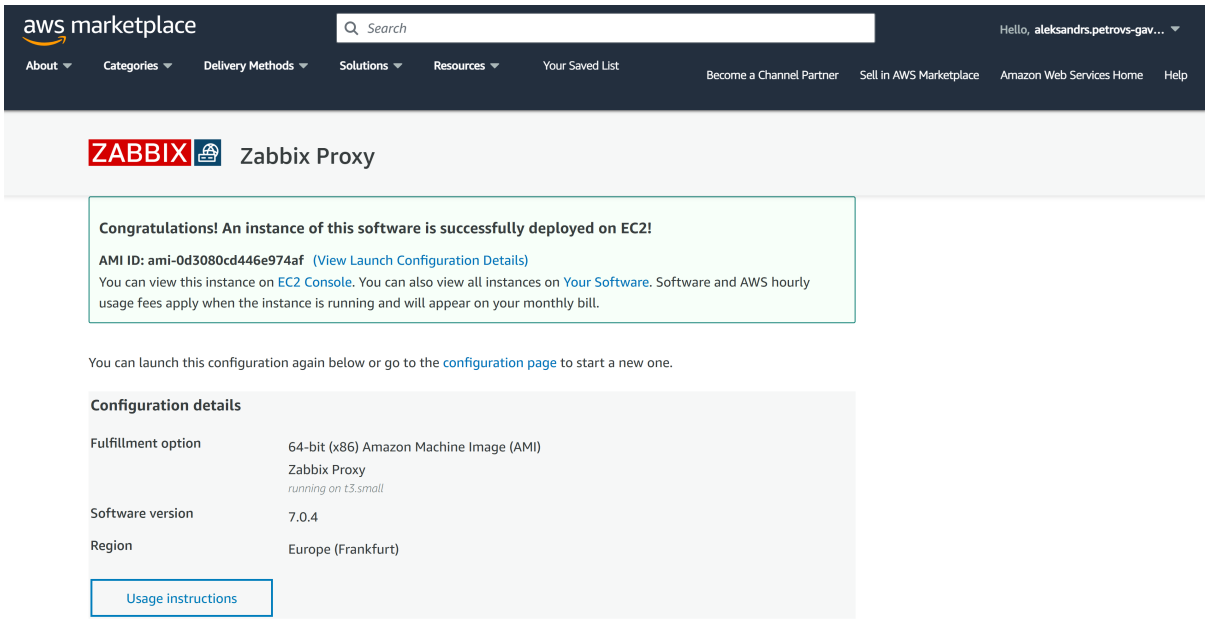
在 Launch 页面底部，指定安全组设置，并选择或创建 EC2 密钥对。请务必将私钥保存在安全的位置。



如果选择 *Create a new Security Group based on the Seller settings*，将允许 TCP 端口 10051（Zabbix 服务器）。注意：如果希望通过 SSH 访问实例，还需要在安全组设置中允许 SSH 连接。

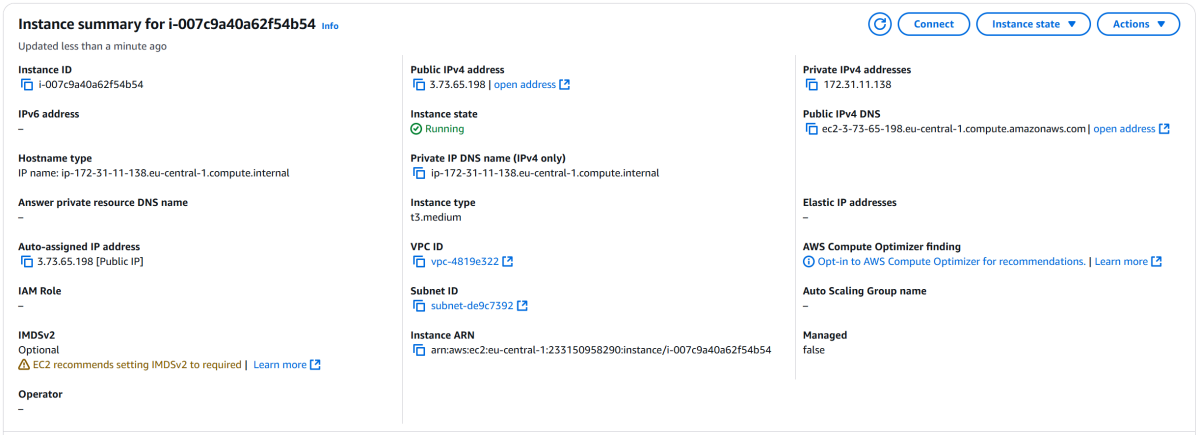


最后，单击 *Launch* 按钮。恭喜，您的 Zabbix proxy 现已开始部署。



配置和访问 Zabbix proxy

在 EC2 控制台中打开 Instances 部分，然后选择新部署的 Zabbix proxy 实例。



使用 SSH 连接到 Zabbix proxy 的 Public IPv4 地址或 Public IPv4 DNS。默认的 SSH 用户名为 `rocky`。请务必使用之前生成的密钥对。

凭据生成可能需要几分钟。请等待几分钟后再尝试连接。



Zabbix proxy 配置文件位于 `/etc/zabbix/zabbix_proxy.conf`。

可以使用您选择的文本编辑器打开配置文件，例如 vi：

```
sudo vi /etc/zabbix/zabbix_proxy.conf
```

初始 proxy 配置包括修改 Zabbix proxy 配置文件中的 `ProxyMode`、`Server` 和 `Hostname` 参数。

可以通过编辑`ProxyMode` 参数来更改 proxy 模式。设置 `ProxyMode=0` 使用主动模式，或设置 `ProxyMode=1` 使用被动模式。

`Server` 参数应包含 Zabbix 服务器的 IP 地址。例如：

```
Server=192.168.50.141
```

`Hostname` 参数应包含 Zabbix proxy 的主机名。此处定义的主机名应与在Zabbix 前端中配置 proxy 时提供的 proxy 名称一致。例如：

```
Hostname=EC2-Frankfurt
```

完成配置后，使用以下命令重启 Zabbix proxy：

```
sudo systemctl restart zabbix-proxy
```

注意：使用 `sudo` 编辑配置文件以及从命令行执行命令。

```
### Option: ProxyMode
#       Proxy operating mode.
#       0 - proxy in the active mode
#       1 - proxy in the passive mode
#
# Mandatory: no
# Default:
# ProxyMode=0

### Option: Server
#       If ProxyMode is set to active mode:
#               IP address or DNS name of Zabbix server to get configuration data from and send data to.
#       If ProxyMode is set to passive mode:
#               List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix
server.
#               Incoming connections will be accepted only from the addresses listed here.
#               If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated
 equally
#               and '::/0' will allow any IPv4 or IPv6 address.
#               '0.0.0.0/0' can be used to allow any IPv4 address.
#               Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
#
# Mandatory: yes
# Default:
# Server=

Server=127.0.0.1
```

```
### Option: Hostname
#       Unique, case sensitive Proxy name. Make sure the Proxy name is known to the server!
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

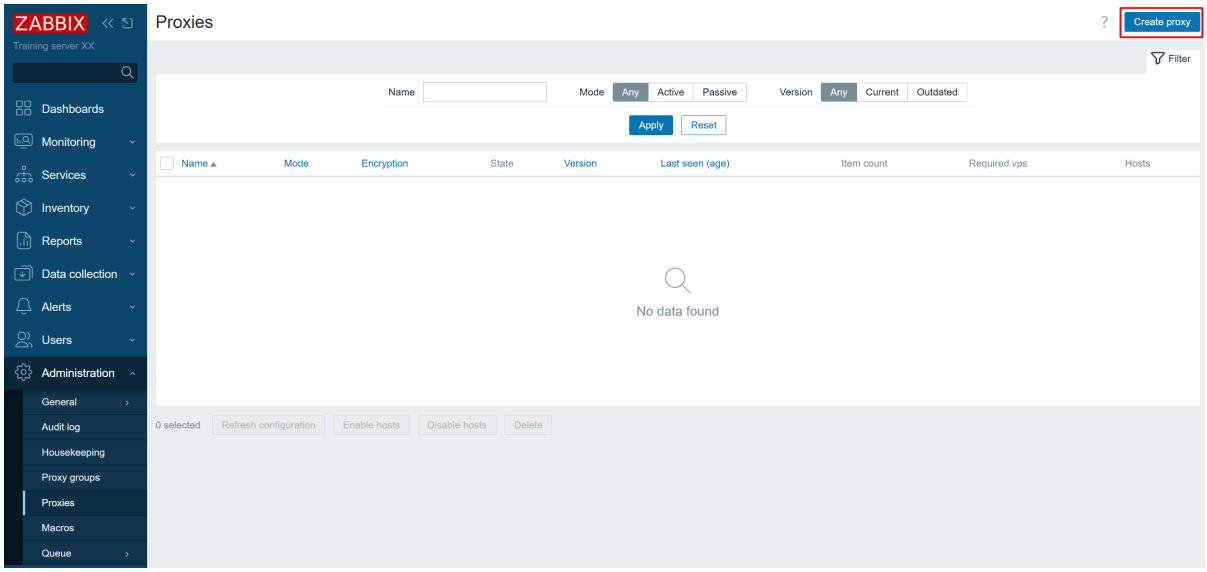
Hostname=EC2-Frankfurt
```

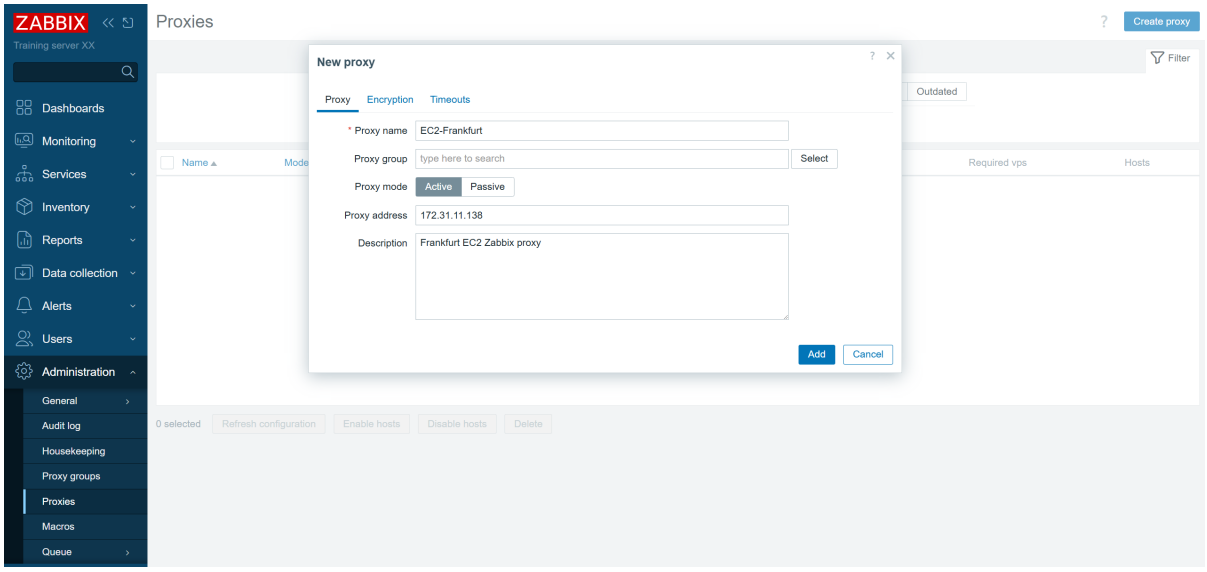
打开 Zabbix 前端并导航至 *Administration* > *Proxies*。单击 *Create proxy* 按钮。

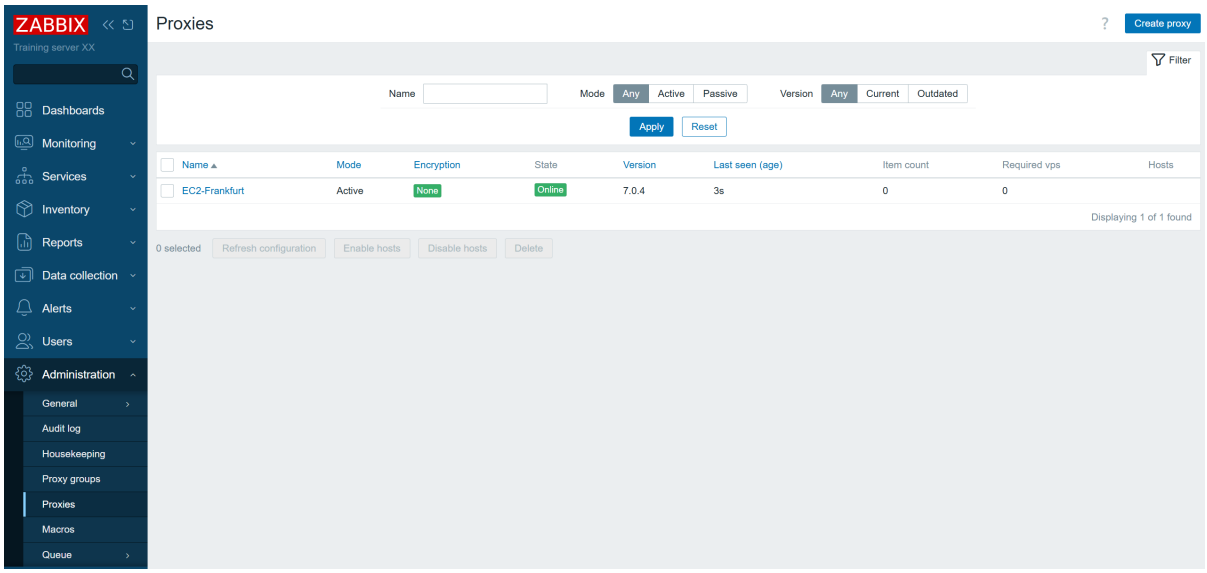
输入与 proxy 配置文件中 `Hostname` 参数值相同的 proxy 名称。

选择 proxy 模式，并提供 proxy IP 地址（主动 proxy 可不填写）。

单击 *Create proxy*。







proxy 数据库位置

SQLite proxy 数据库存储在 `/tmp/zabbix_proxy.sqlite3` 中，由 Zabbix proxy 配置文件中的 DBName 参数指定。

Zabbix 服务器 **Azure** 部署指南

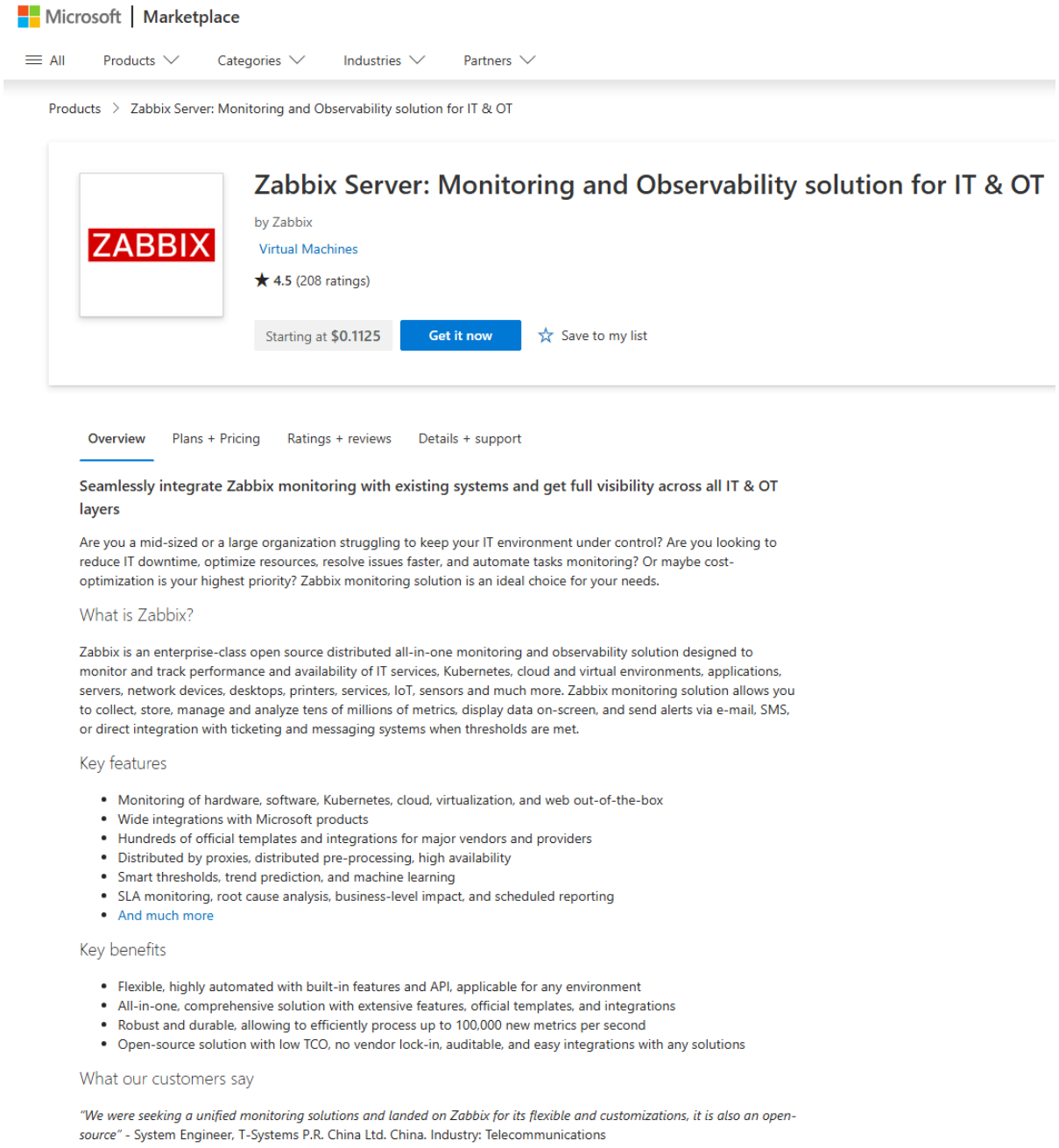
概述

本页面介绍如何使用 Azure 部署 Zabbix 服务器。

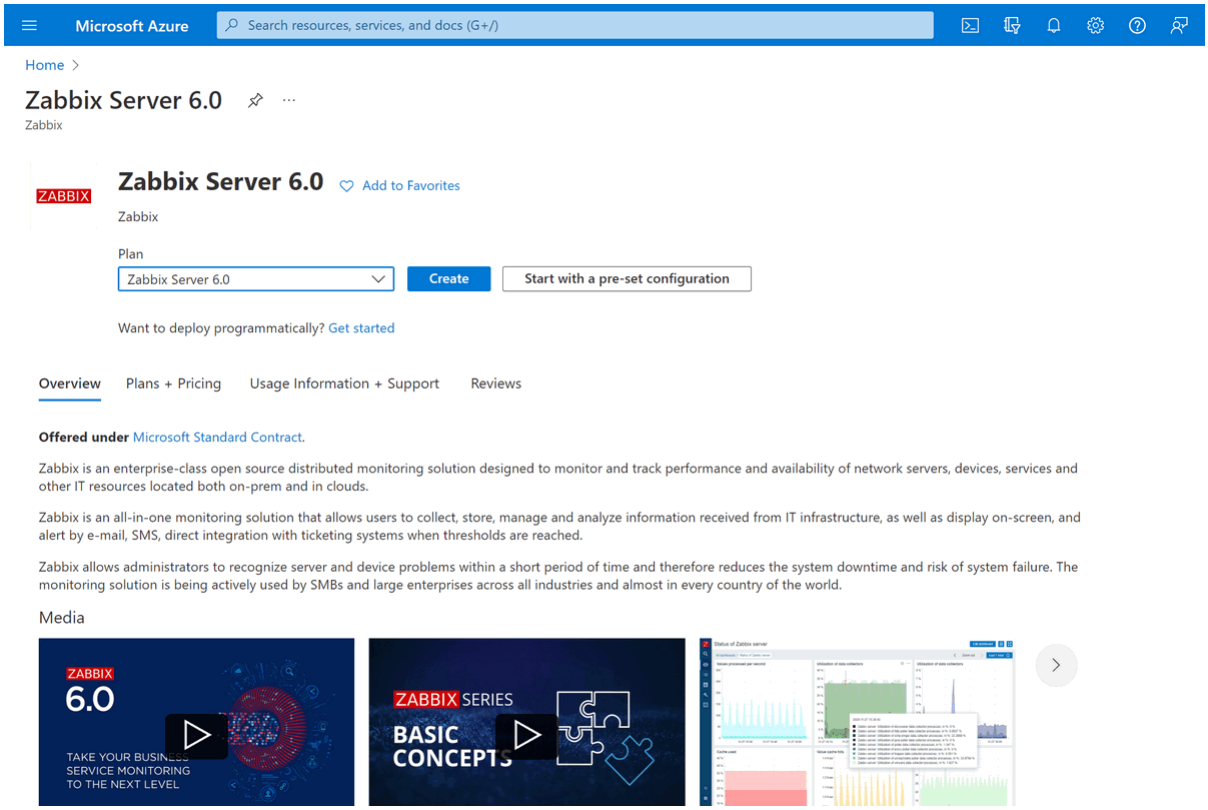
> **Attention:**
> 该平台支持部署多个 Zabbix 版本。请务必阅读与您部署的 Zabbix 版本相同的文档。您可以从版本下拉菜单中选择不同 Zabbix 版本的文档。

部署 appliance

要部署 Zabbix appliance，首先访问 Zabbix Azure Marketplace 页面。查看概览，然后单击 *Get it now*。



查看部署概览、定价计划和其他详细信息。接下来，单击 *Create* 或 *Start with a pre-set configuration*，以部署 Zabbix appliance。

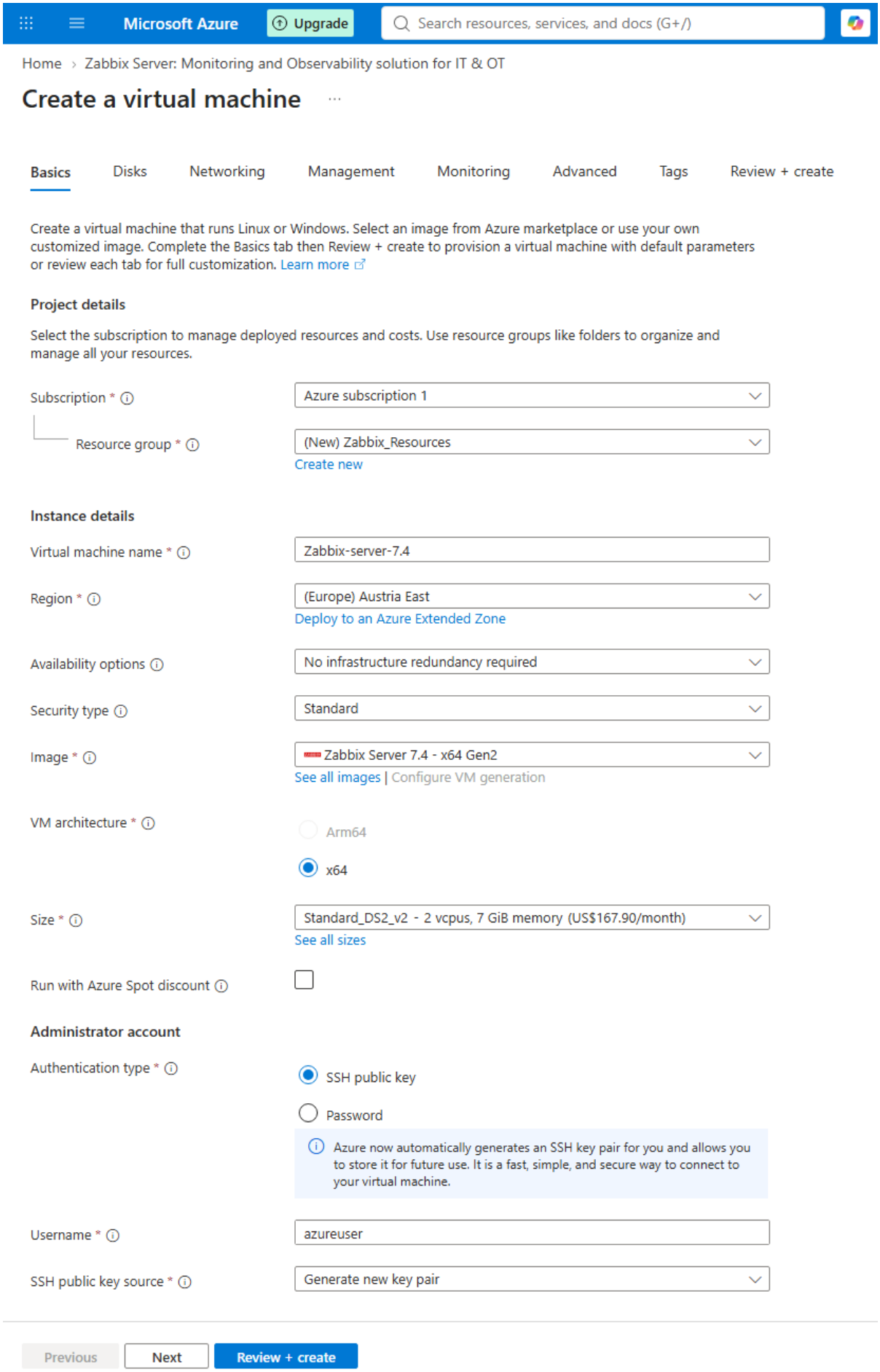


通过选择 VM 大小、创建管理员帐户以及定义磁盘、网络、管理和其他设置来配置 VM。这些设置可能因您的基础设施要求而异。

您可以使用以下配置作为示例：

Basics 选项卡：

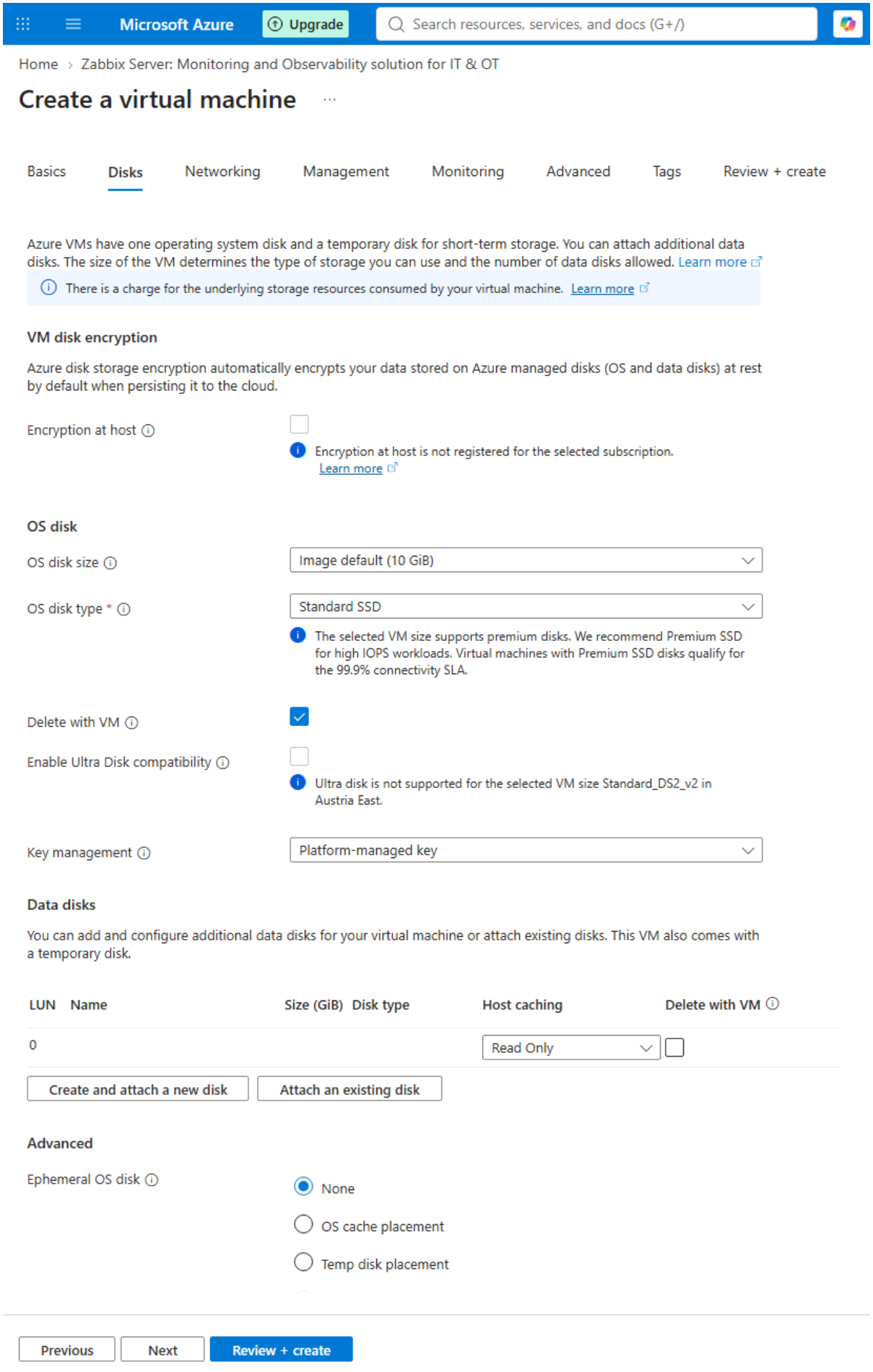
- *Subscription*: Azure subscription 1
- *Resource group*: Zabbix_Resources
- *Virtual machine name*: Zabbix-server--7.4
- *Availability options*: No infrastructure redundancy required
- *Security type*: Standard
- *Image*: Zabbix Server 7.4 - x64 Gen2
- *Size*: Standard_DS2_V2 - 2 vcpus, 7 GiB memory
- *Authentication type*: SSH public key
- *Username*: azureuser
- *SSH public key source*: Generate new key pair



Disks 选项卡：

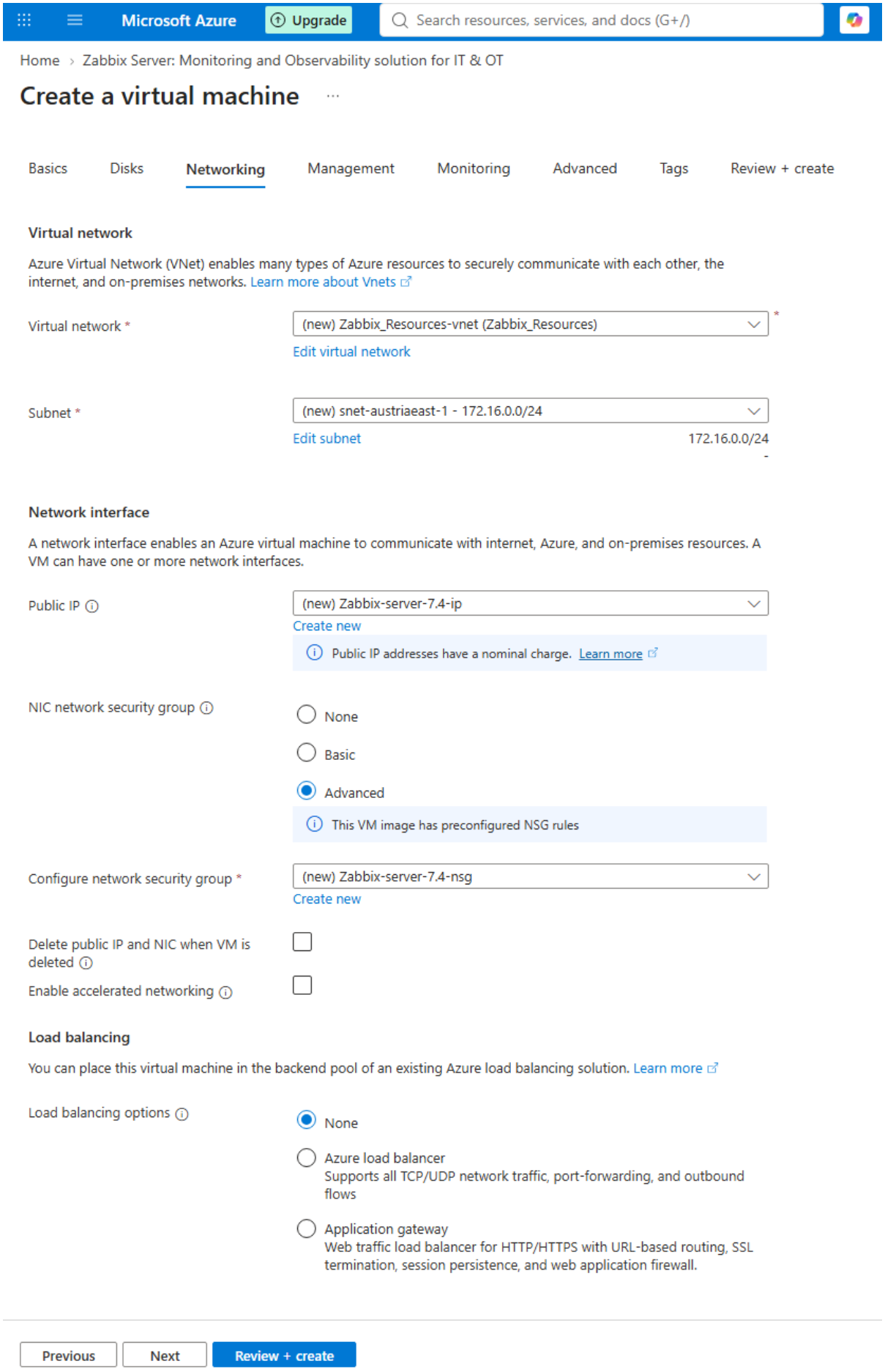
- *Encryption at host*: unchecked
- *OS disk type*: Standard SSD

- *Delete with VM*: checked
- *Enable Ultra Disk compatibility*: unchecked

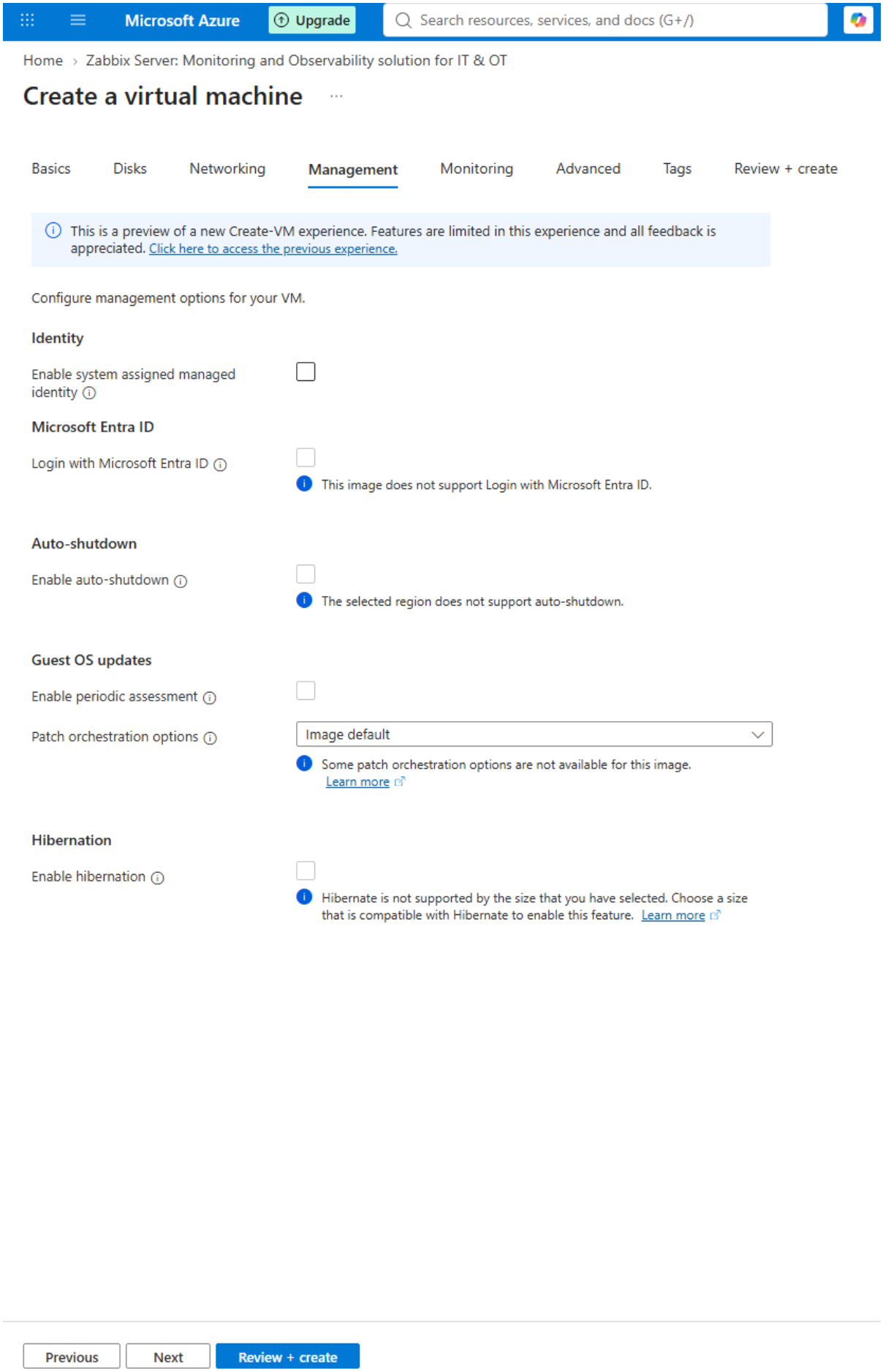


Networking 选项卡：

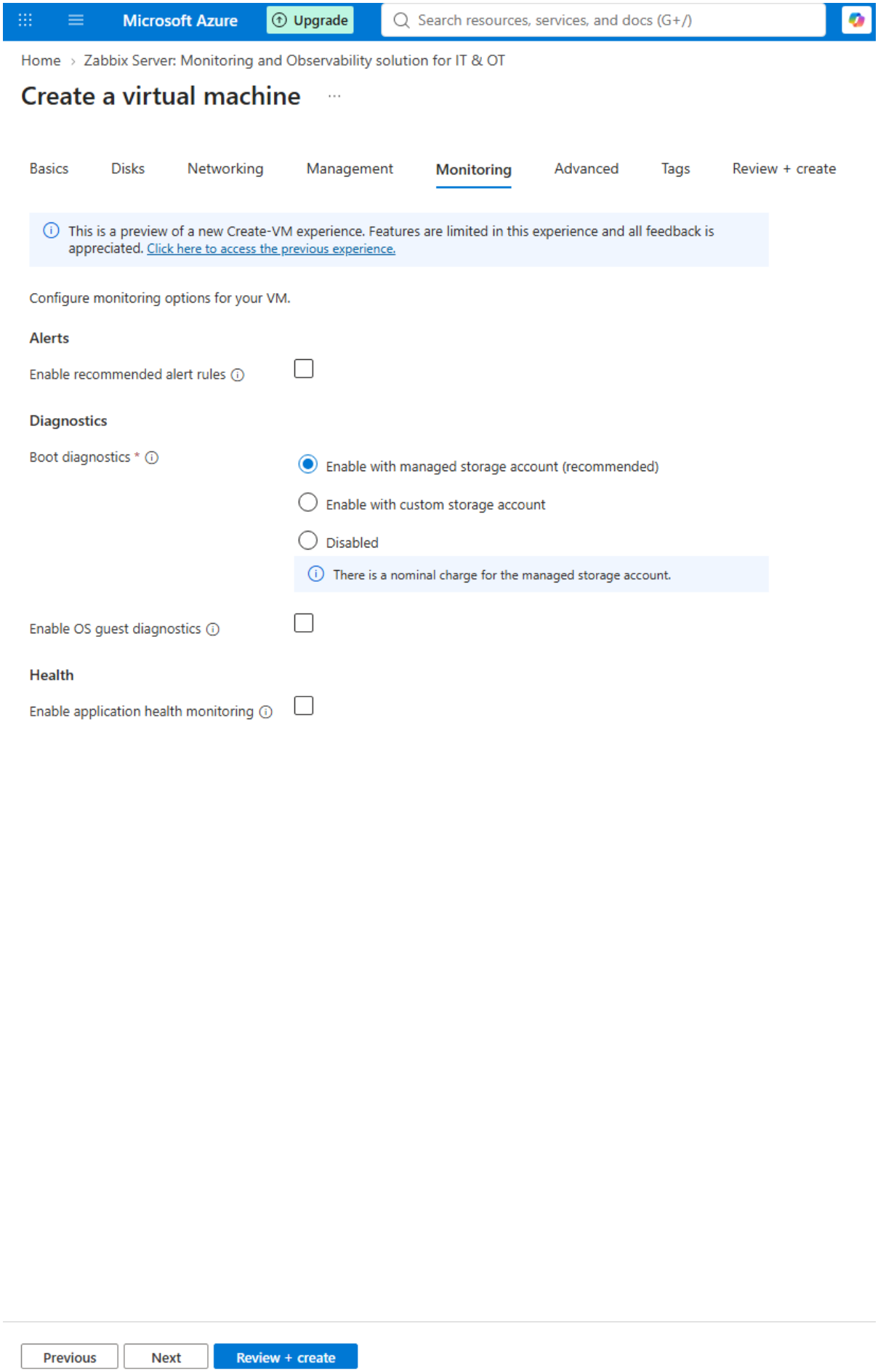
- *Virtual network*: Zabbix_Resources-vnet
- *Public IP*: Zabbix-server-7.4-ip
- *NIC network security group*: Advanced
- *Configure network security group*: Zabbix-server-7.4-nsg



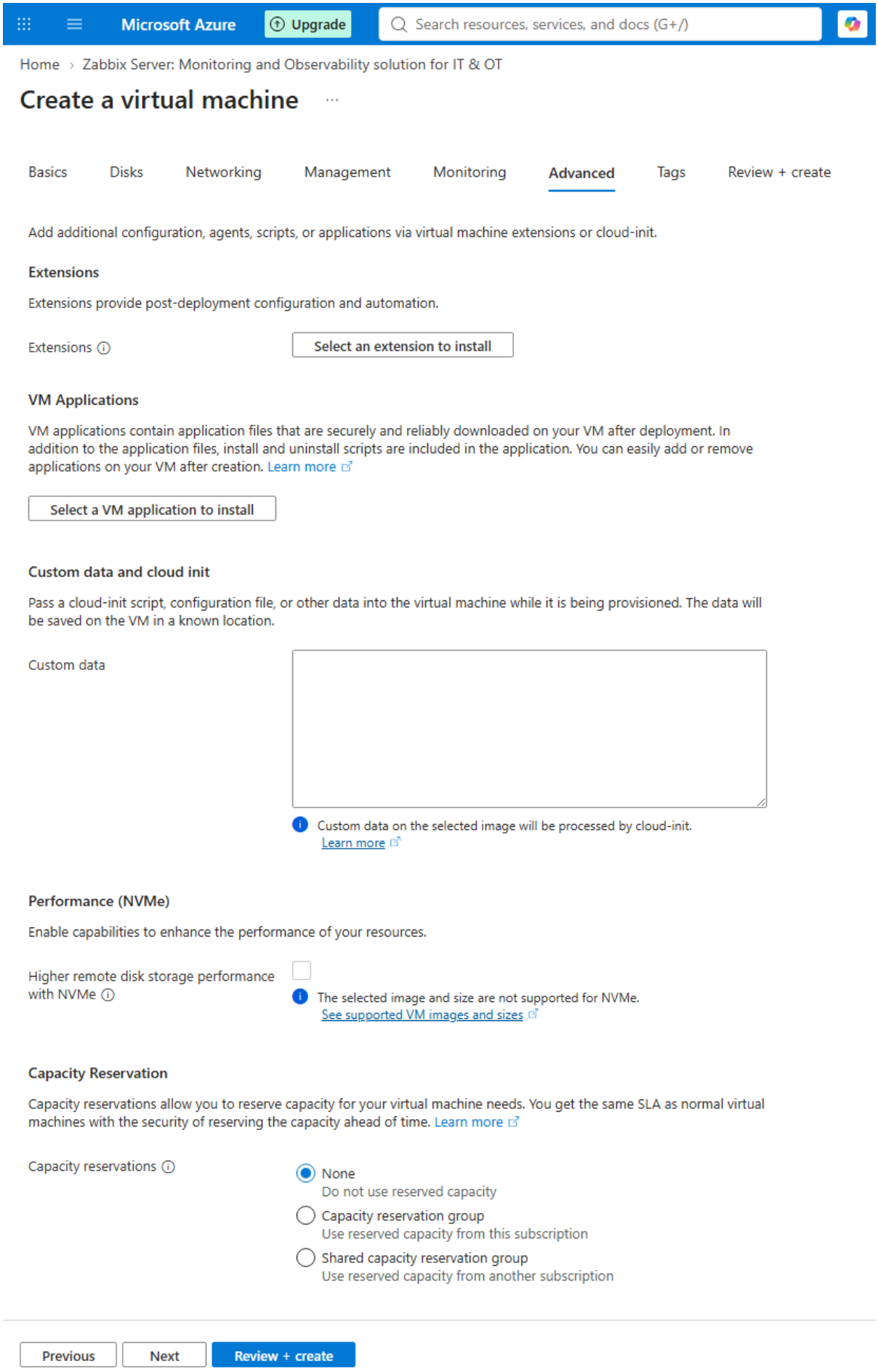
Management 选项卡：参数可以保留默认值。



Monitoring 选项卡：参数可以保留默认值。



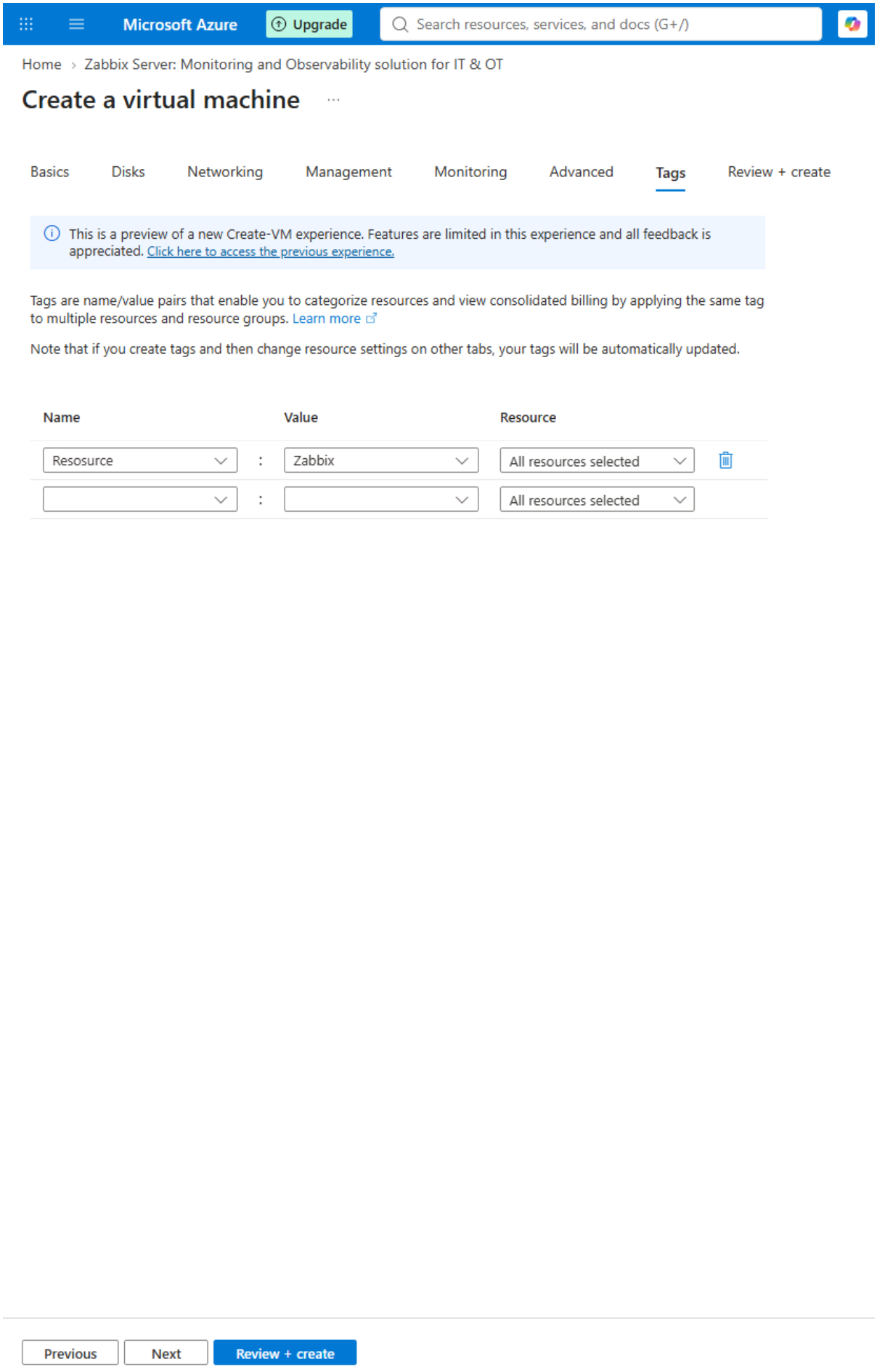
Advanced 选项卡：参数可以保留默认值。



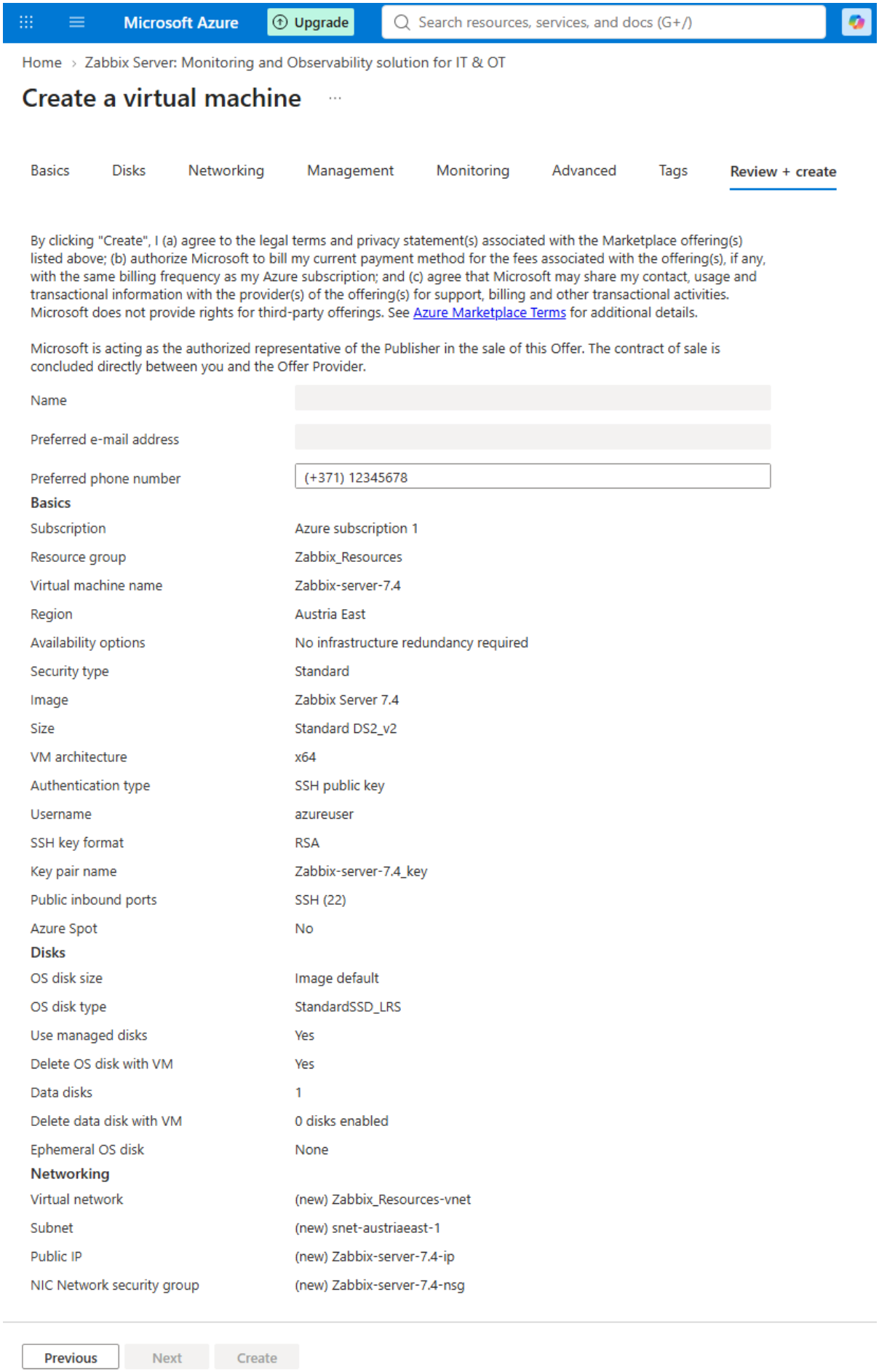
Tags 选项卡：

- *Name*: Resource
- *Value*: Zabbix

- *Resource*: All resource selected



确认 VM 详细信息并创建 VM。



您将被重定向到 VM 部署状态页面。等待部署完成。

部署完成后，打开该资源并记下其公共 IP 地址。现在，您需要使用 SSH（或其他可用方法）连接到 Zabbix 服务器，以获取 Zabbix 前端登录凭据。

```
********************************************************************************

Zabbix frontend credentials:

Username: Admin

Password: zfpvagcs0s+srUUK+EM

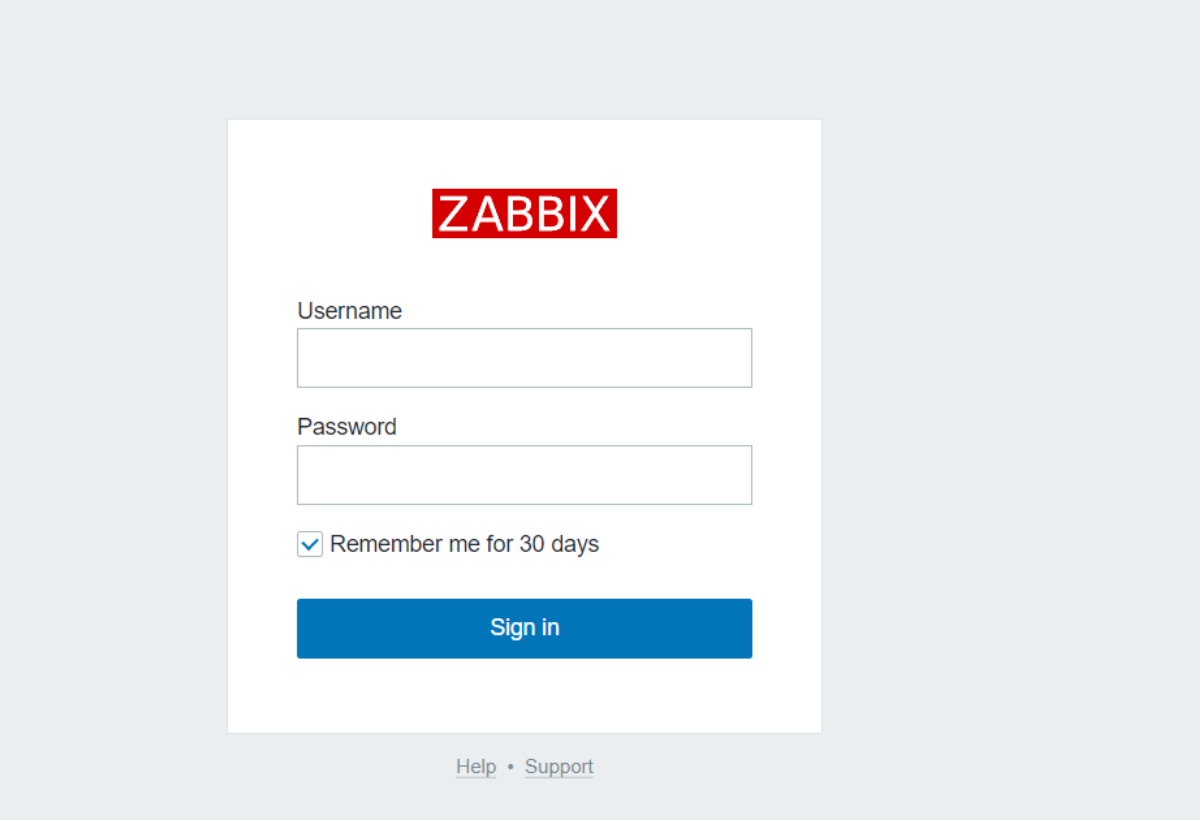

To learn about available professional services, including technical suppport and
 training, please visit https://www.zabbix.com/services

Official Zabbix documentation available at https://www.zabbix.com/documentation/
current/

Note! Do not forget to change timezone PHP variable in /etc/php-fpm.d/zabbix.con
f file.


********************************************************************************
[azureuser@Zabbix-server-6 ~]$
```

在 Web 浏览器中打开 Zabbix appliance 的公共 IP 地址，以访问 Zabbix 前端。登录时，使用上一步获取的用户名和密码。



访问数据库

您可以使用 `root` 用户访问 Zabbix 数据库，无需输入密码。凭据存储在 `/root/.my.cnf` 配置文件中。

Zabbix proxy 的 **Azure** 部署指南

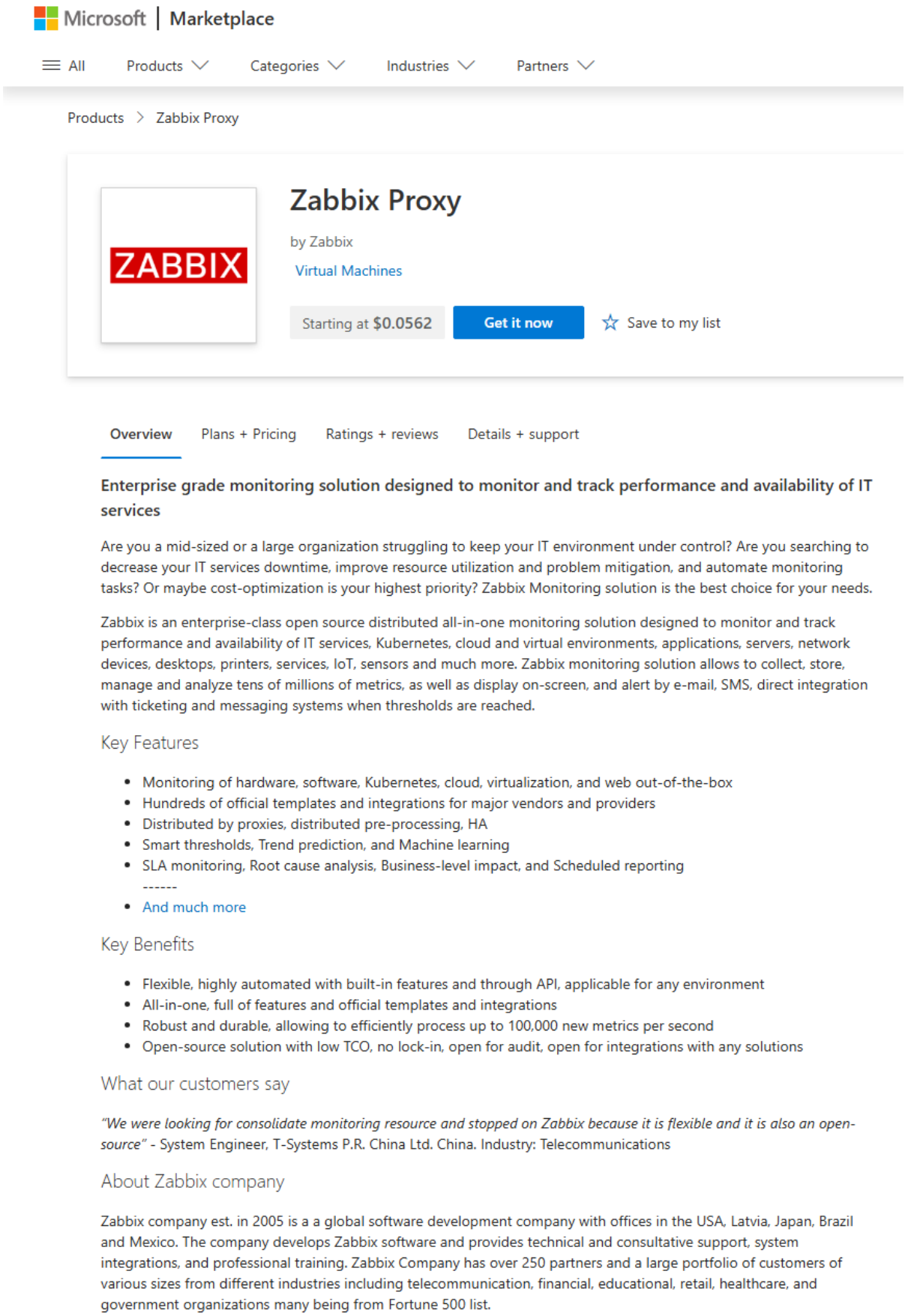
概述

本页面介绍如何使用 Azure 部署 Zabbix proxy。

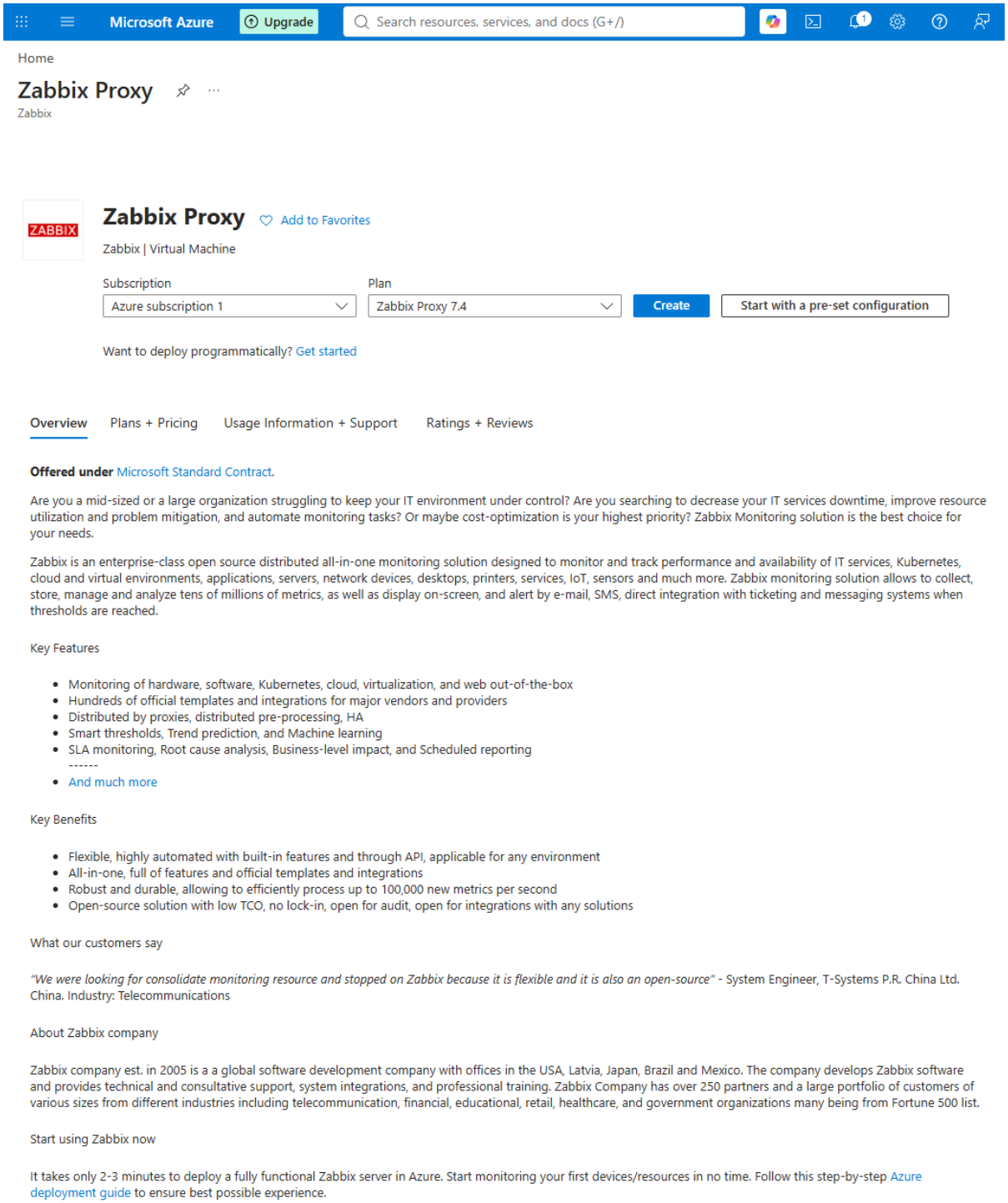
> **Attention:**
> 该平台支持部署多个 Zabbix 版本。请务必阅读与您所部署的 Zabbix 版本相同的文档。您可以从版本下拉菜单中选择不同 Zabbix 版本的文档。

部署 Zabbix proxy

要部署 Zabbix proxy，首先访问 Zabbix Azure marketplace 页面。点击 *Get it now*。



在此处可以查看当前部署的其他信息，例如总体概览、定价计划等。点击 *Create* 或 *Start with a pre-set configuration*。

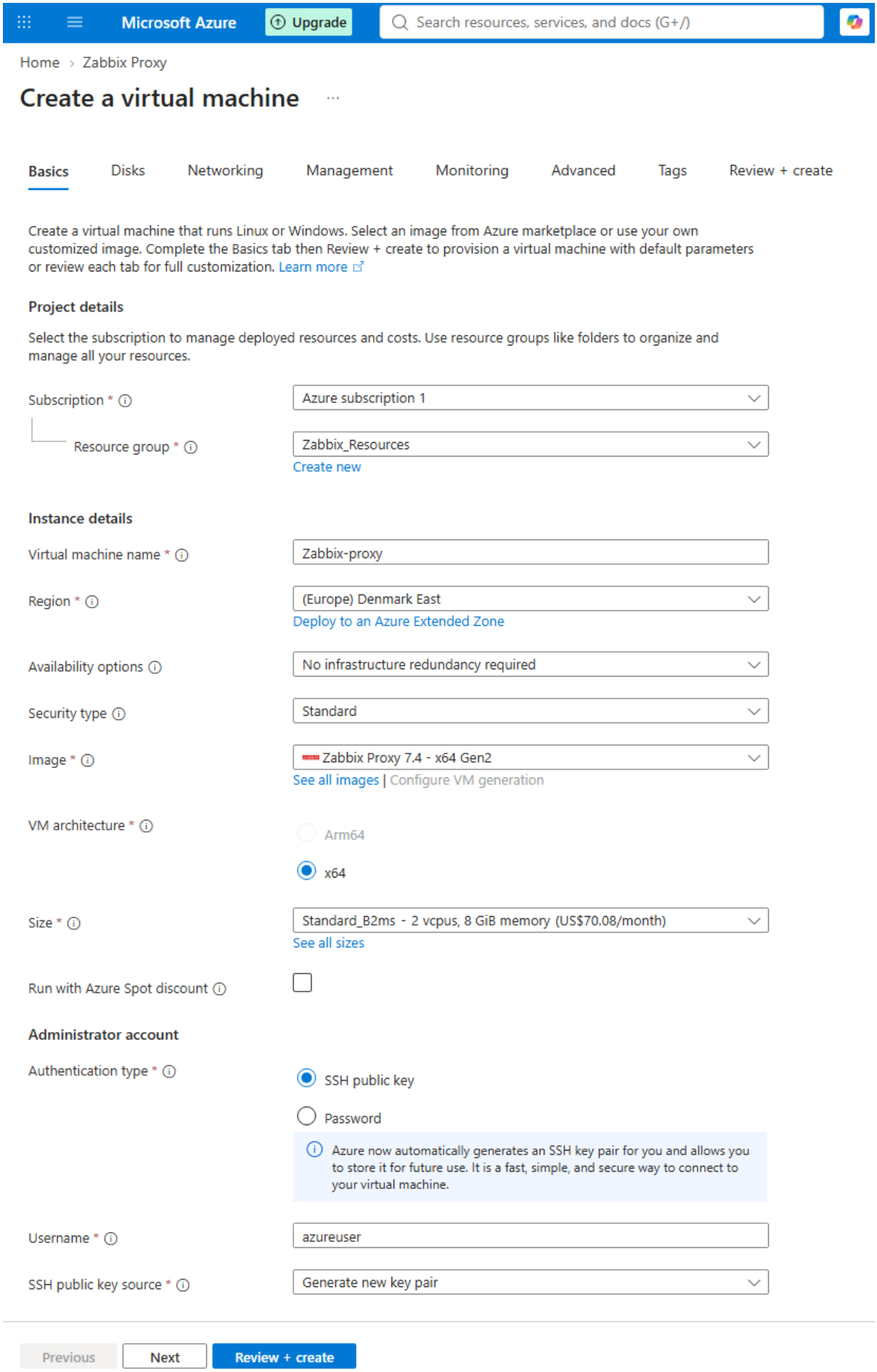


继续进行详细的 VM 配置：选择 VM 大小、管理员帐户、磁盘、网络、管理和其他设置。这些设置可能因基础设施要求而异。

可以使用以下配置作为示例：

Basics 选项卡：

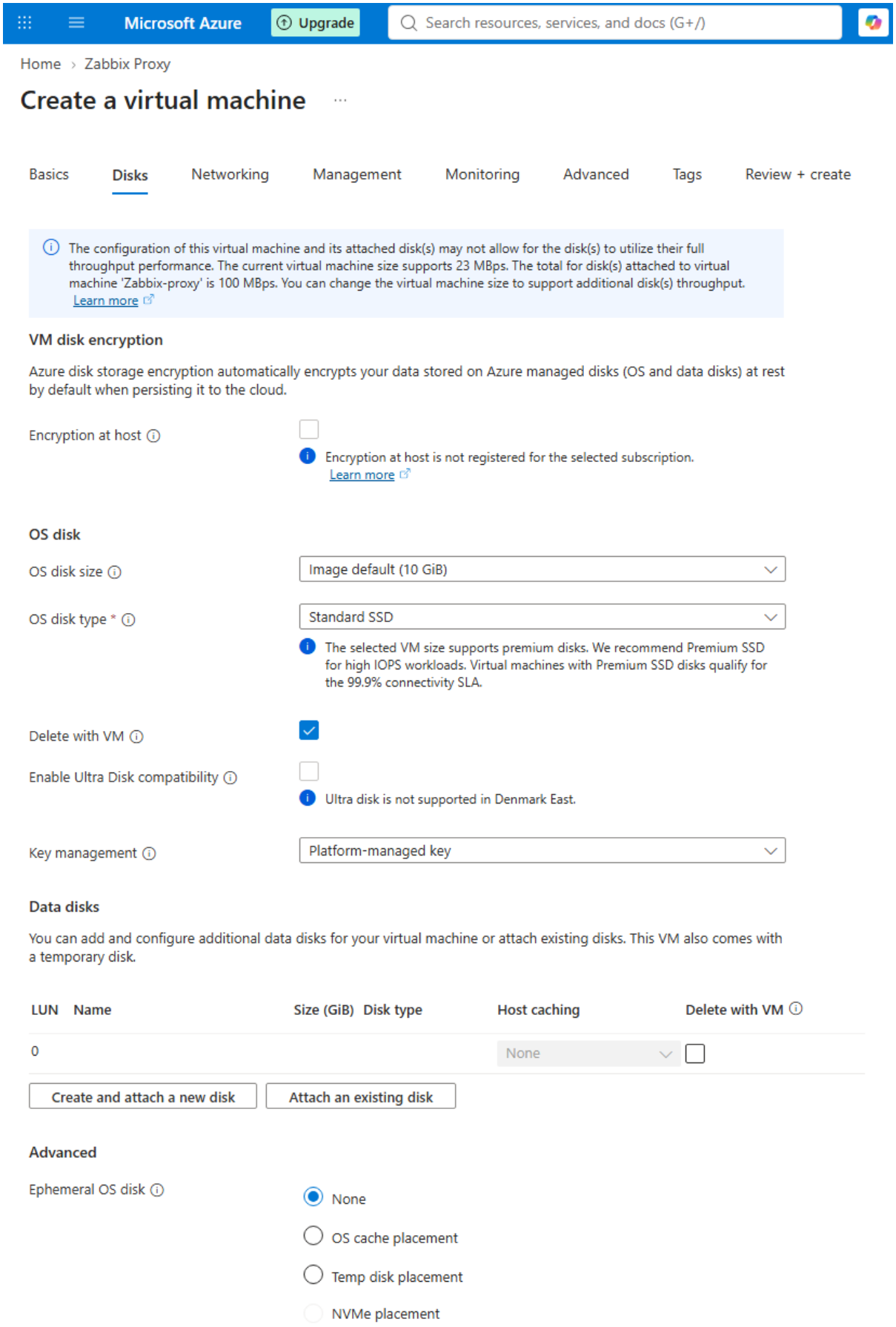
- *Subscription*: Azure subscription 1
- *Resource group*: Zabbix_Resources
- *Virtual machine name*: Zabbix-proxy
- *Availability options*: No infrastructure redundancy required
- *Security type*: Standard
- *Image*: Zabbix Proxy 7.4 - x64 Gen2
- *Size*: Standard_B2ms - 2 vcpus, 8 GiB memory
- *Authentication type*: SSH public key
- *Username*: azureuser
- *SSH public key source*: Generate new key pair



Disks 选项卡：

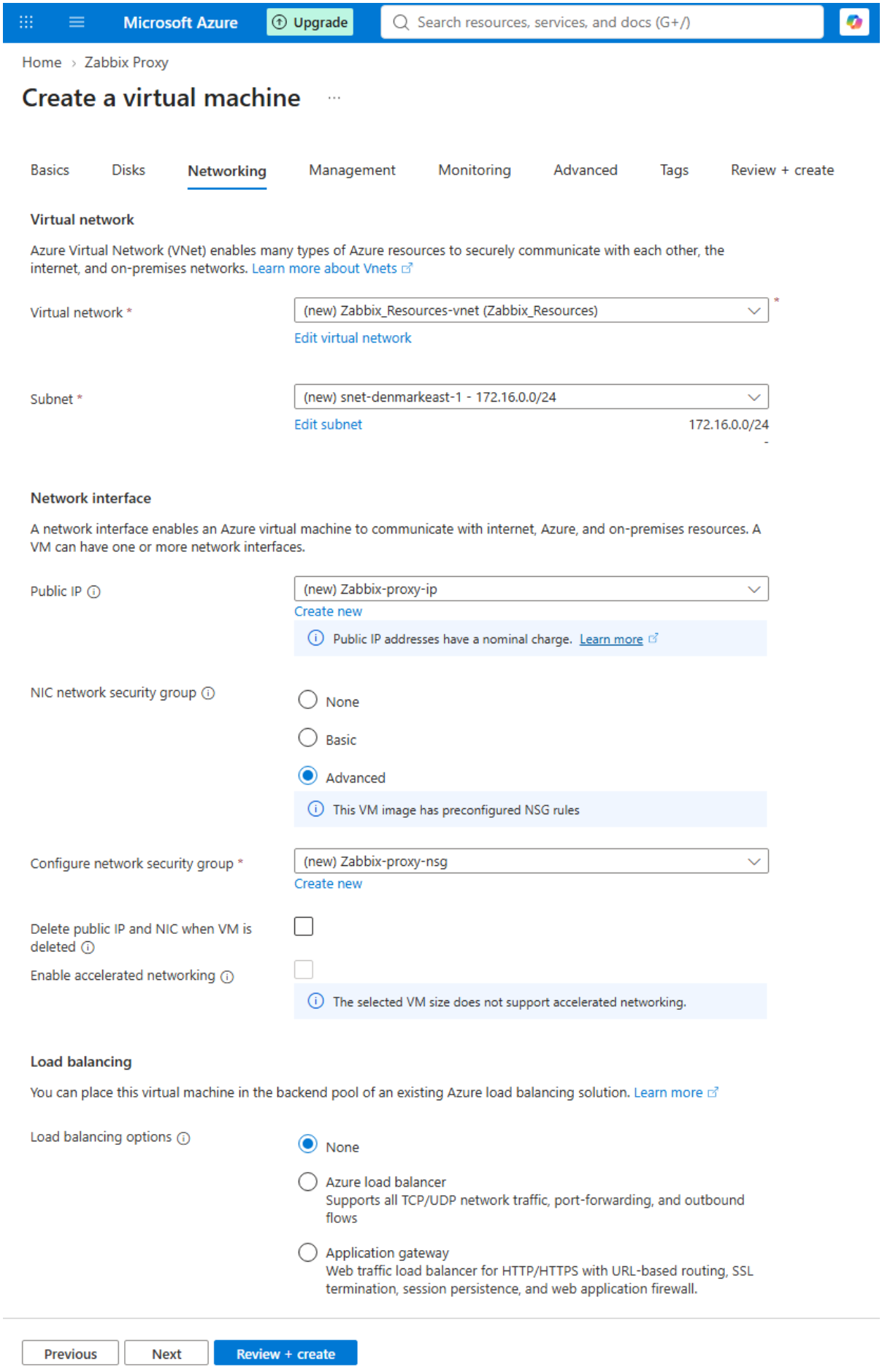
- *Encryption at host*: unchecked

- *OS disk type*: Standard SSD
- *Delete with VM*: checked
- *Enable Ultra Disk compatibility*: unchecked

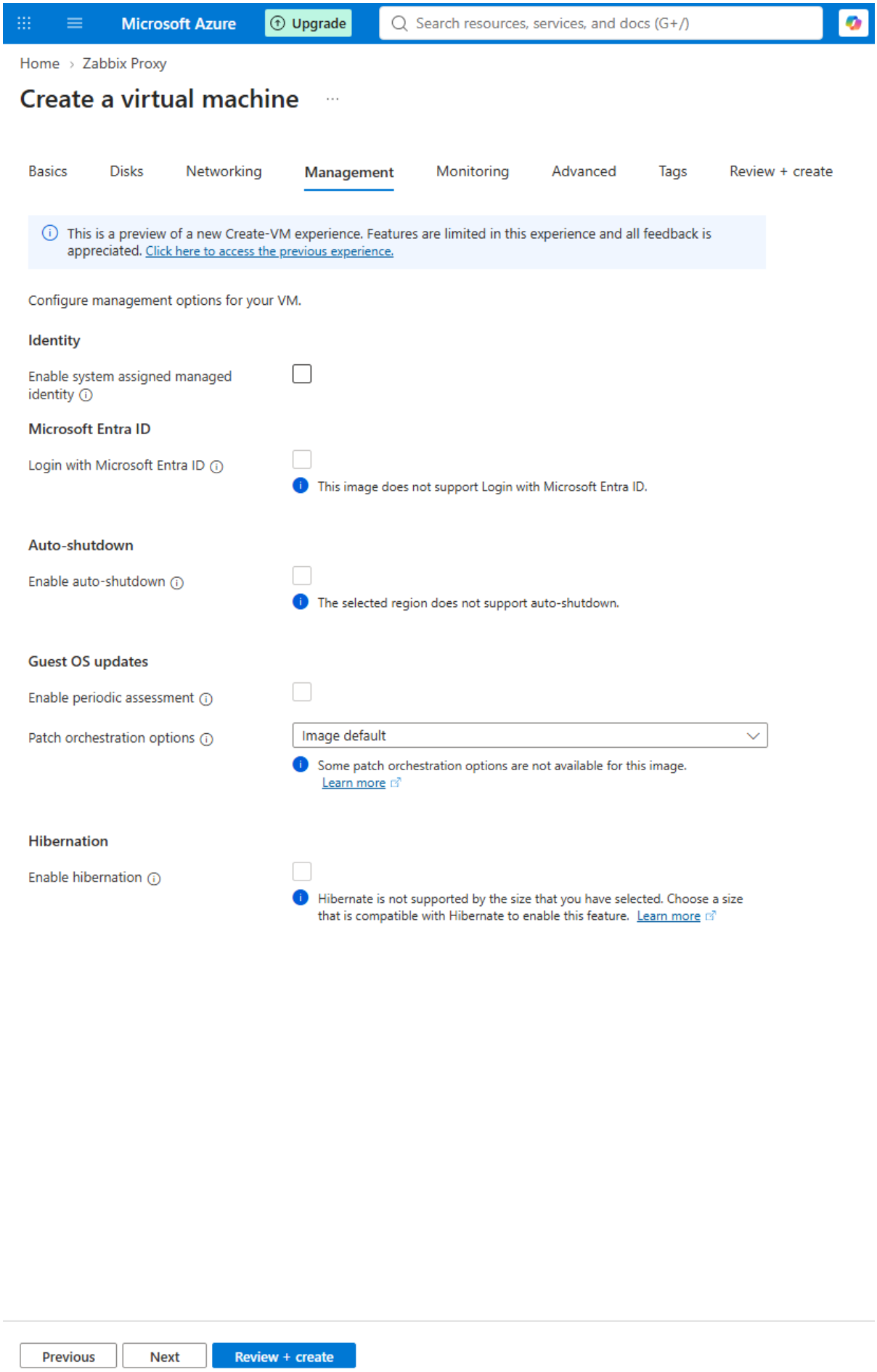


Networking 选项卡：

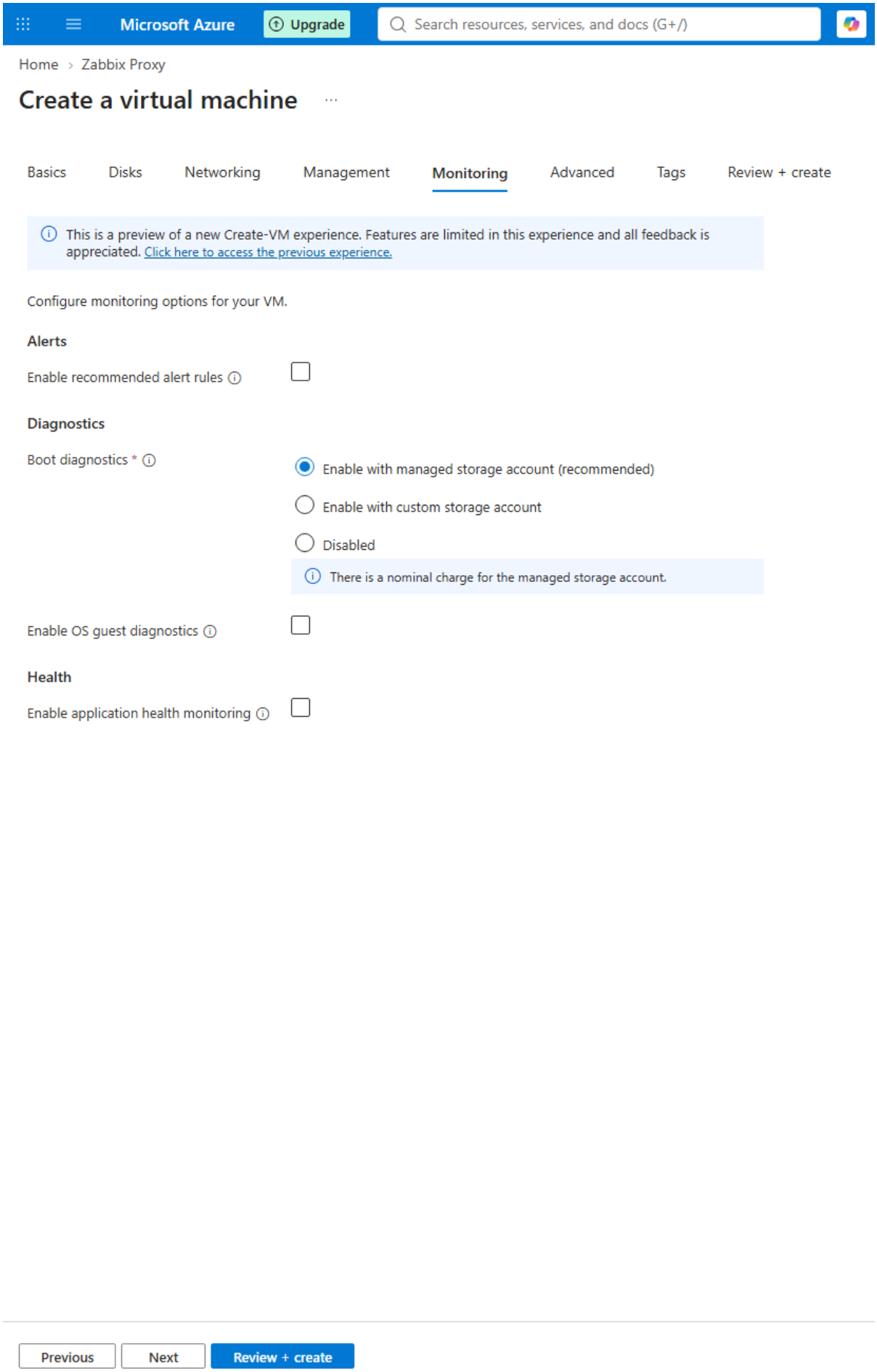
- *Virtual network*: Zabbix_Resources-vnet
- *Public IP*: Zabbix-proxy-ip
- *NIC network security group*: Advanced
- *Configure network security group*: Zabbix-proxy-nsg



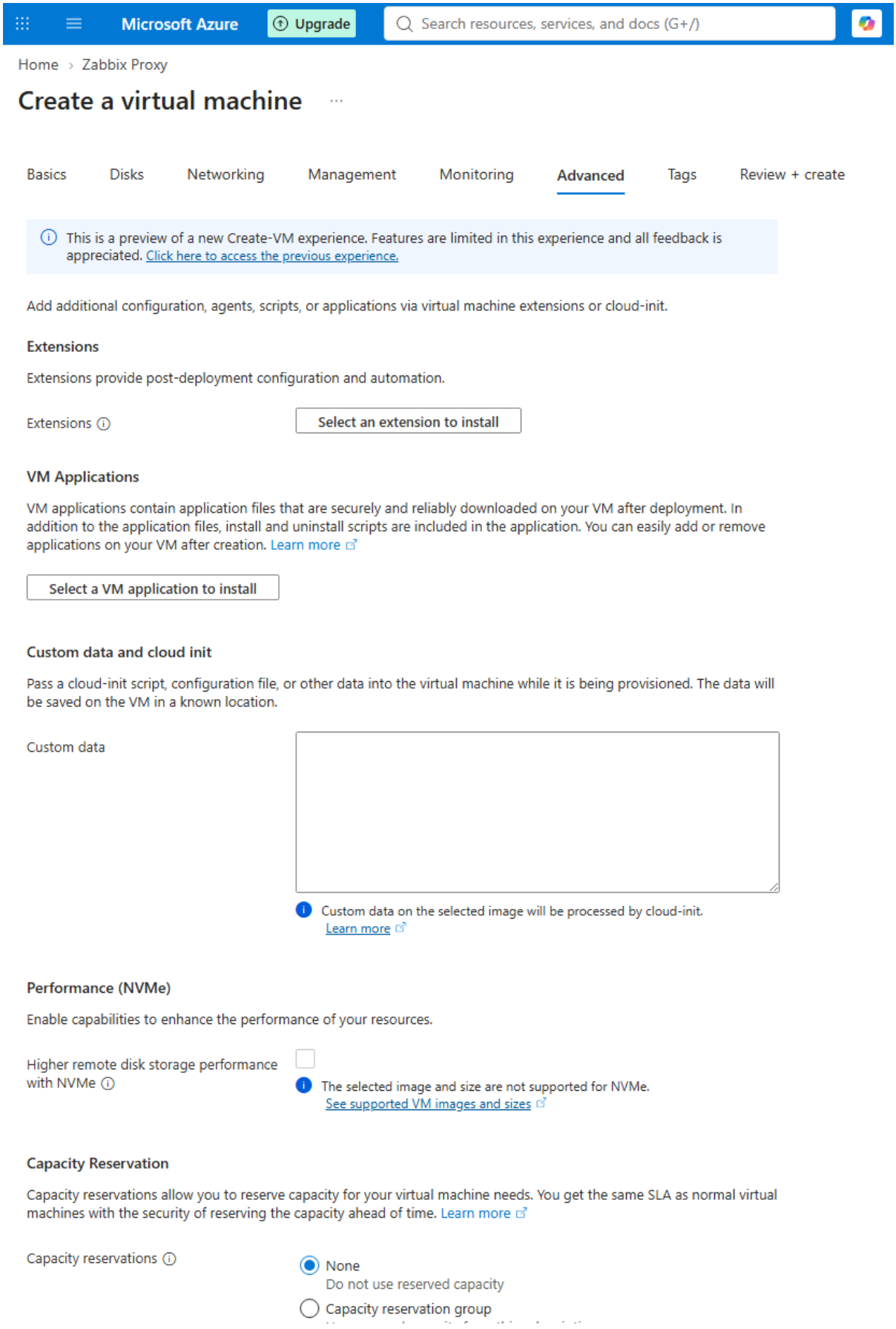
Management 选项卡：参数可以保留默认值。



Monitoring 选项卡：参数可以保留默认值。



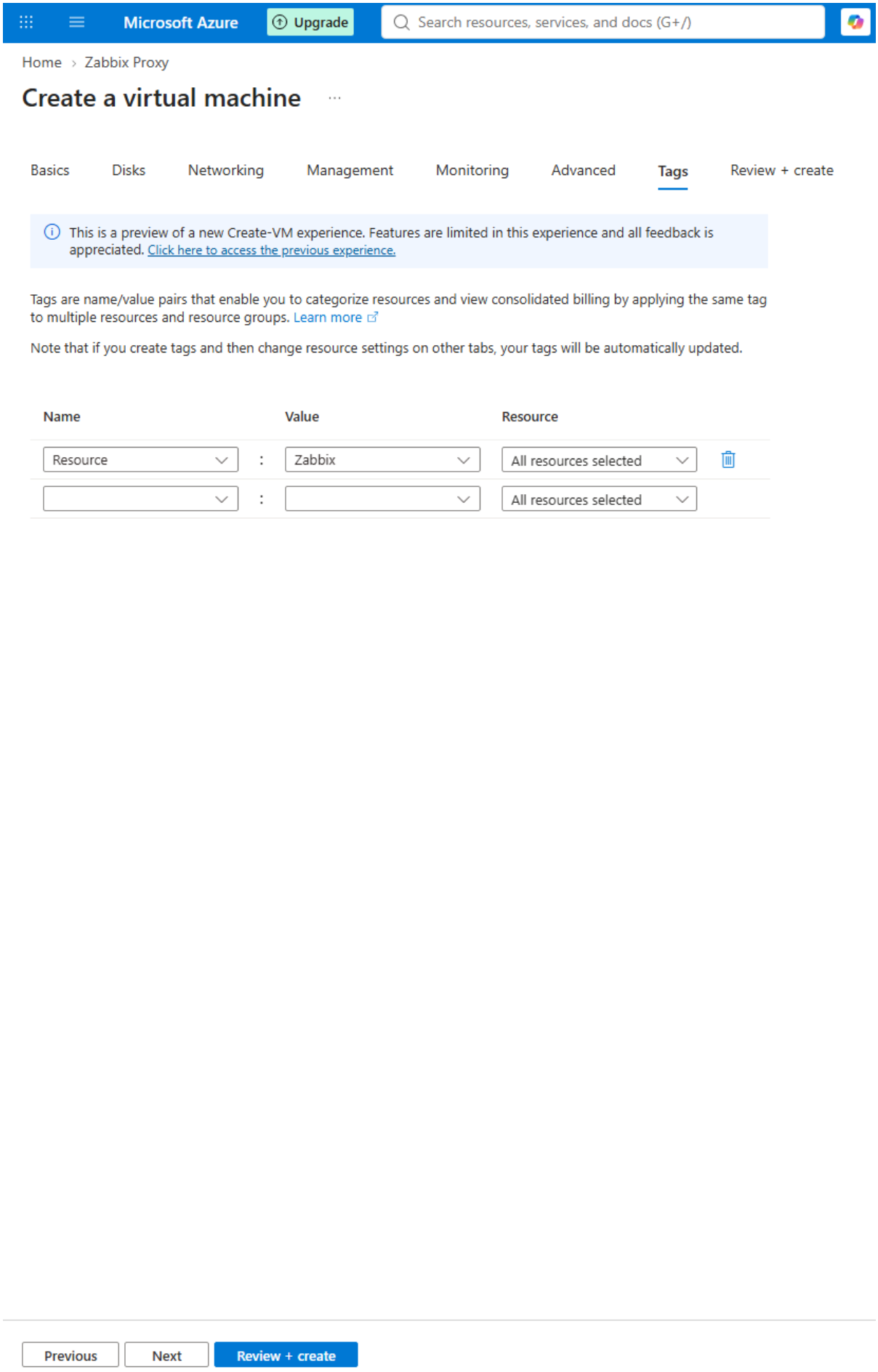
Advanced 选项卡：参数可以保留默认值。



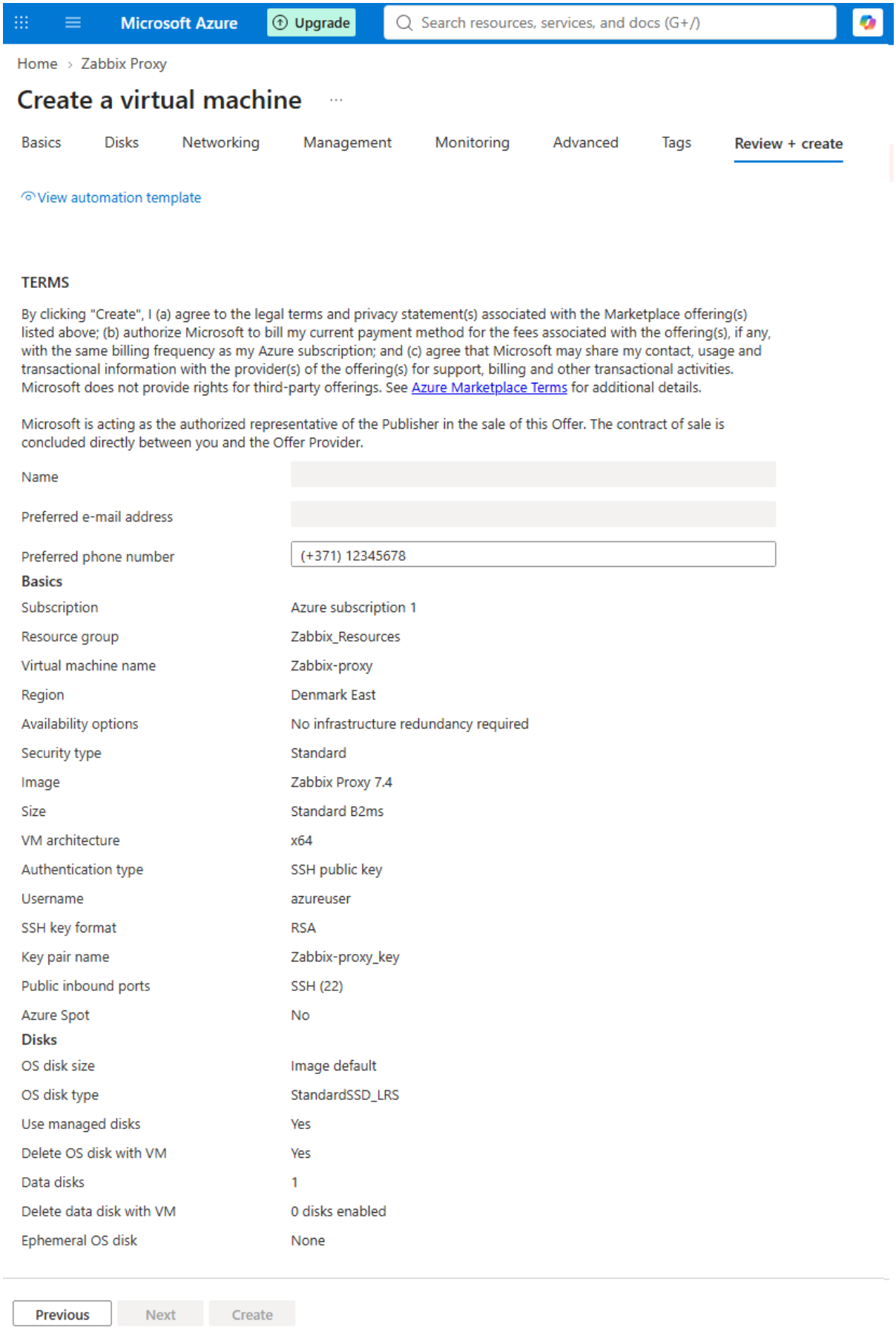
Tags 选项卡：

- *Name*: Resource
- *Value*: Zabbix

- *Resource*: All resource selected



确认 VM 详细信息并创建 VM。



系统会将您重定向到 VM 部署状态页面。等待部署完成。

部署完成后，打开资源并记下其公共 IP 地址。使用 VM 部署期间创建的管理员凭据通过 SSH 连接到 Zabbix proxy，以配置 proxy。

```
*******************************************************************************

To learn about available professional services, including technical suppport and
 training, please visit https://www.zabbix.com/services

Official Zabbix documentation available at https://www.zabbix.com/documentation/
current/

*******************************************************************************
[azureuser@Zabbix-proxy ~]$
```

Zabbix proxy 配置文件位于 `/etc/zabbix/zabbix_proxy.conf`。

初始 proxy 配置包括修改 Zabbix proxy 配置文件中的 `ProxyMode`、`Server` 和 `Hostname` 参数。

可以通过编辑`ProxyMode` 参数来更改 proxy 模式。将 `ProxyMode=0` 设置为主动模式，或将 `ProxyMode=1` 设置为被动模式。

`Server` 参数应包含 Zabbix 服务器的 IP 地址。

`Hostname` 参数应包含 Zabbix proxy 的主机名。此处定义的主机名应与在 Zabbix 前端配置 proxy 时提供的 proxy 名称一致。

```
############ GENERAL PARAMETERS #################

### Option: ProxyMode
#       Proxy operating mode.
#       0 - proxy in the active mode
#       1 - proxy in the passive mode
#
# Mandatory: no
# Default:
# ProxyMode=0

### Option: Server
#       If ProxyMode is set to active mode:
#               IP address or DNS name (address:port) or cluster (address:port;address2:port) of Zabbix server to get
configuration data from and send data to.
#               If port is not specified, default port is used.
#               Cluster nodes need to be separated by semicolon.
#       If ProxyMode is set to passive mode:
#               List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix server.
#               Incoming connections will be accepted only from the addresses listed here.
#               If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally
#               and '::/0' will allow any IPv4 or IPv6 address.
#               '0.0.0.0/0' can be used to allow any IPv4 address.
#               Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
#
# Mandatory: yes
# Default:
# Server=

Server=127.0.0.1
```

```
### Option: Hostname
#       Unique, case sensitive Proxy name. Make sure the Proxy name is known to the server!
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

Hostname=Zabbix proxy
```

打开 Zabbix 前端并转到 *Administration* > *Proxies*。点击 *Create proxy* 按钮。

输入与 proxy 配置文件中 *Hostname* 参数值相同的 proxy 名称。

选择 proxy 模式并提供 proxy IP 地址（主动 proxy 可不提供）。

点击 *Create proxy* 按钮以创建 proxy。

Proxy 数据库位置

SQLite proxy 数据库位于 `/tmp/zabbix_proxy.sqlite3`，由 Zabbix proxy 配置文件中的 `DBName` 参数指定。

Google Cloud 部署指南

概述

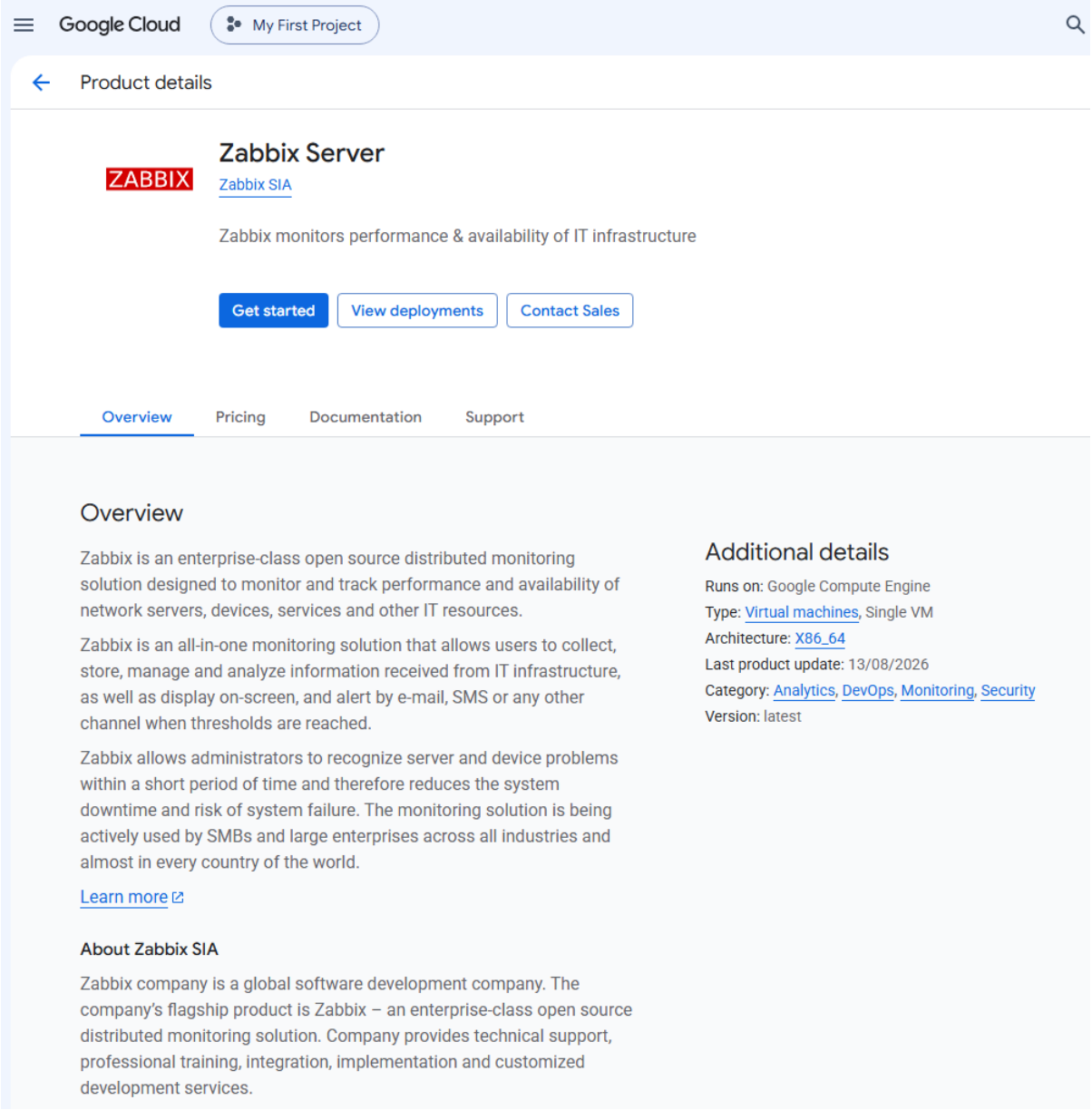
本页面介绍如何使用 Google Cloud Marketplace 部署 Zabbix。

> **Attention:**
> 该平台支持部署多个 Zabbix 版本。请务必阅读与您部署的 Zabbix 版本相同的文档。您可以从版本下拉菜单中选择不同 Zabbix 版本的文档。

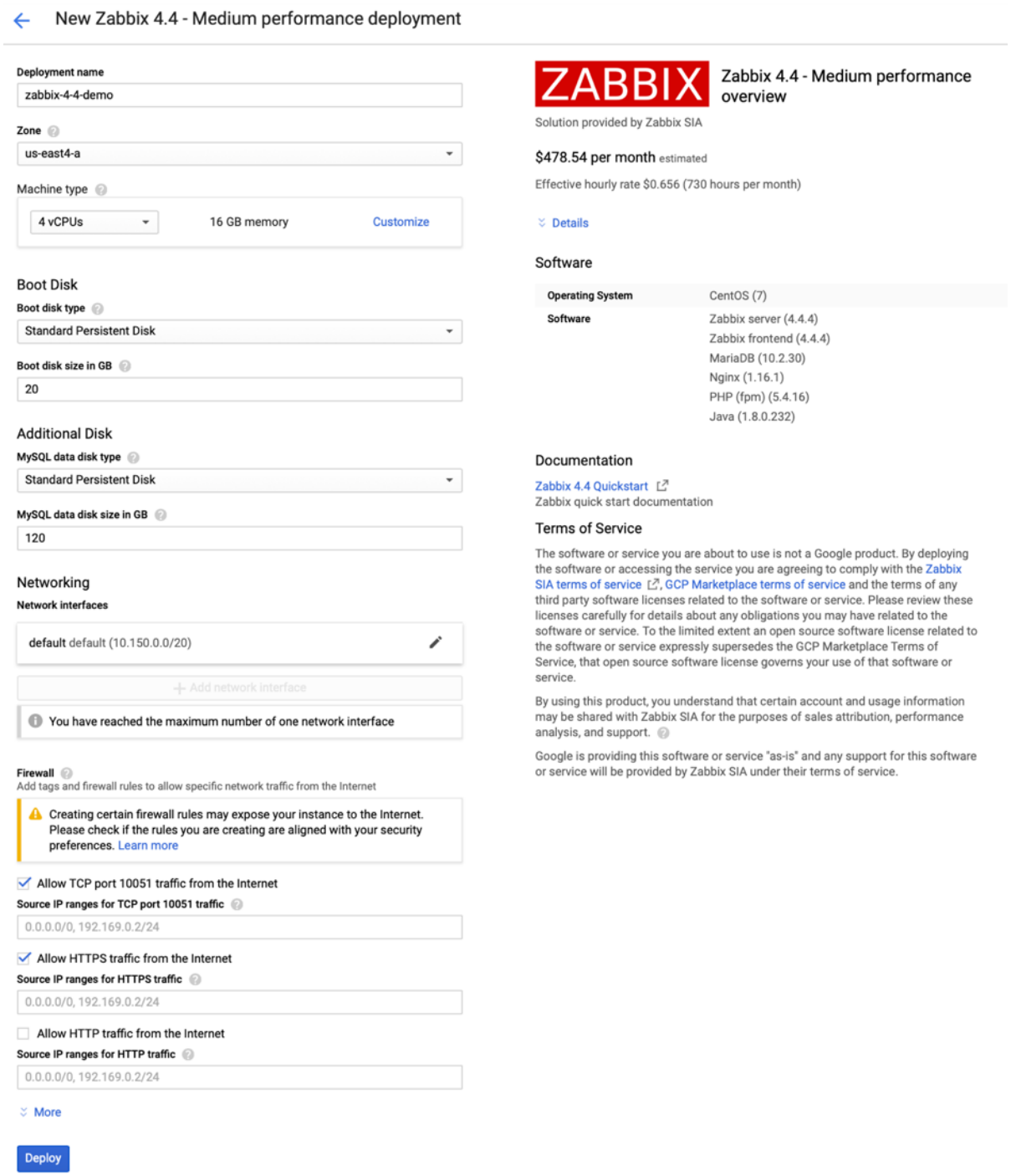
部署 appliance

首先，部署 Zabbix appliance。借助 Google Cloud Marketplace，这非常简单。只需在 Marketplace 中进入 Zabbix Server 产品页面。

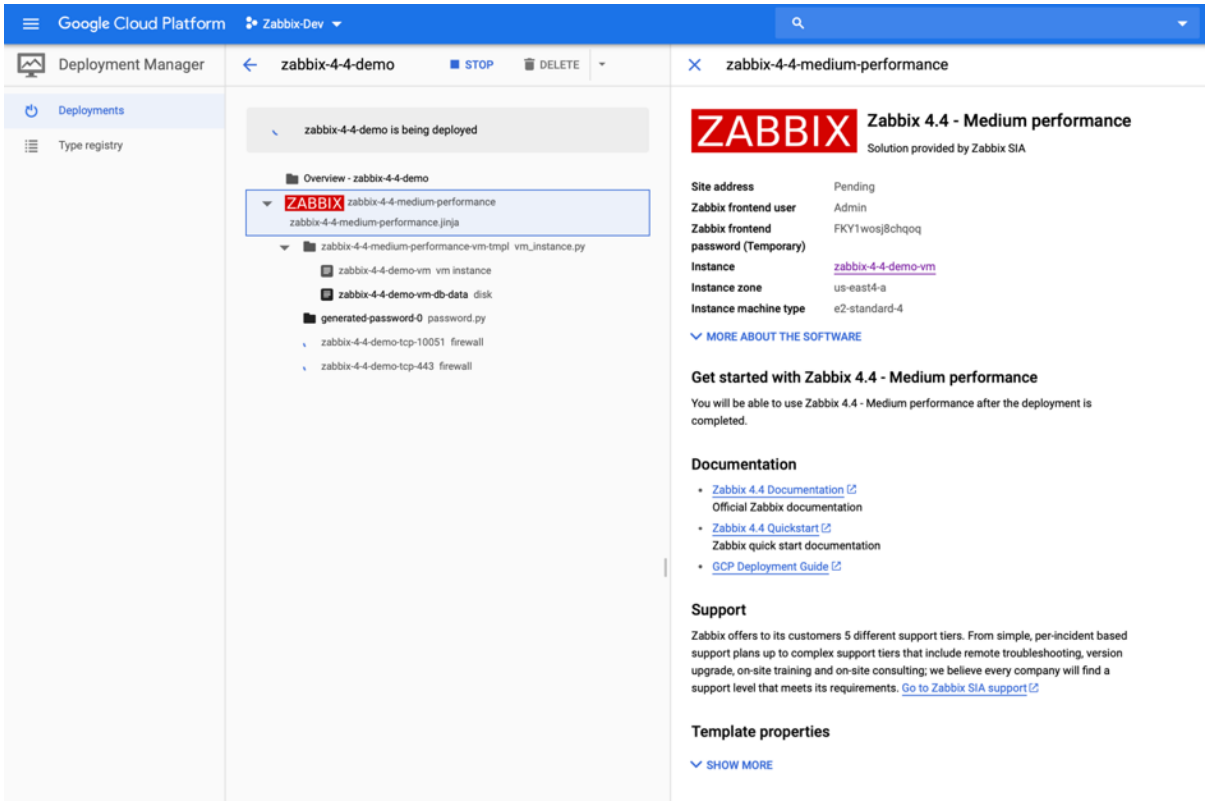
点击 开始使用。



您可以使用默认设置，也可以对其进行自定义。完成后，点击 部署。



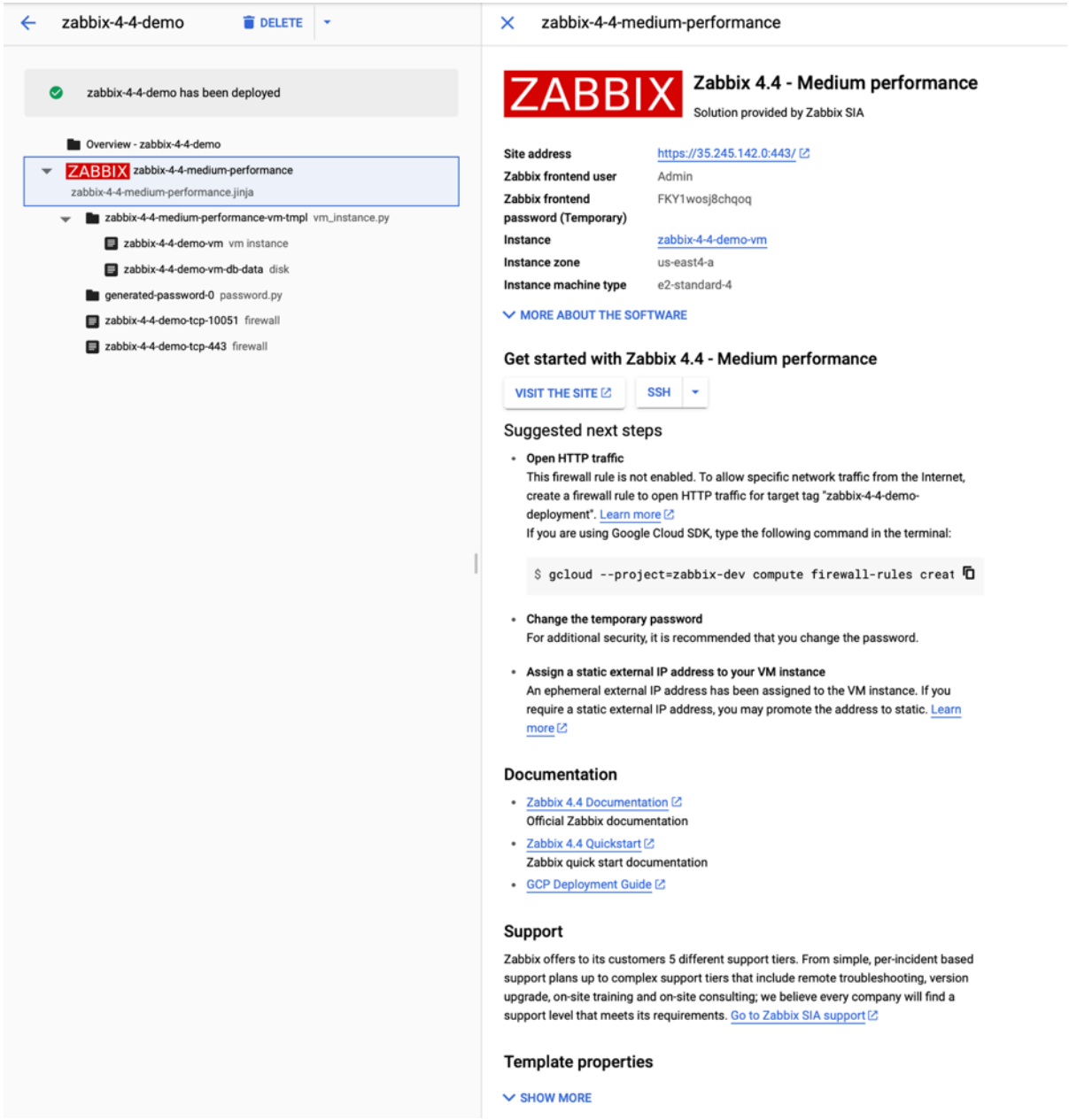
部署现已进行中。请等待其完成。



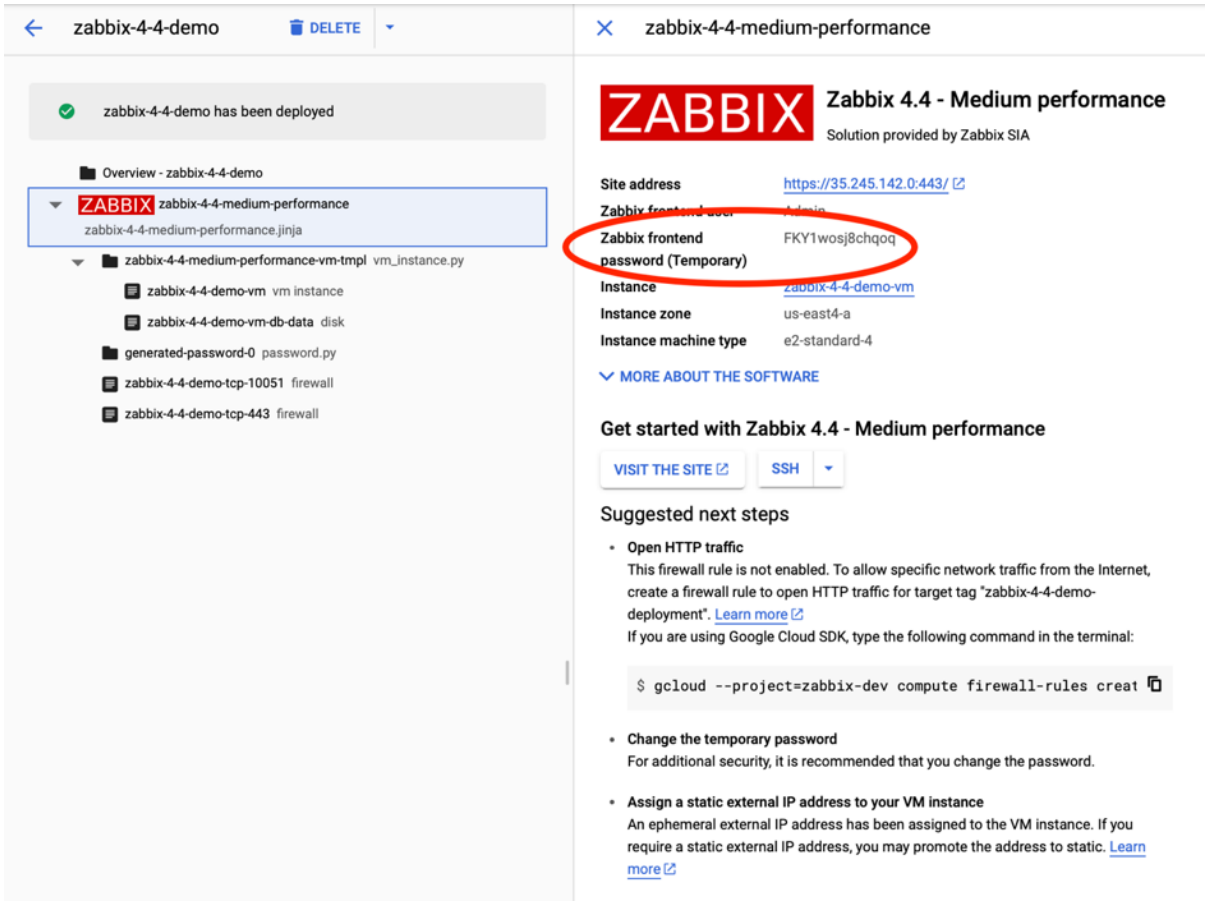
就是这样！部署现已开始。

访问 appliance

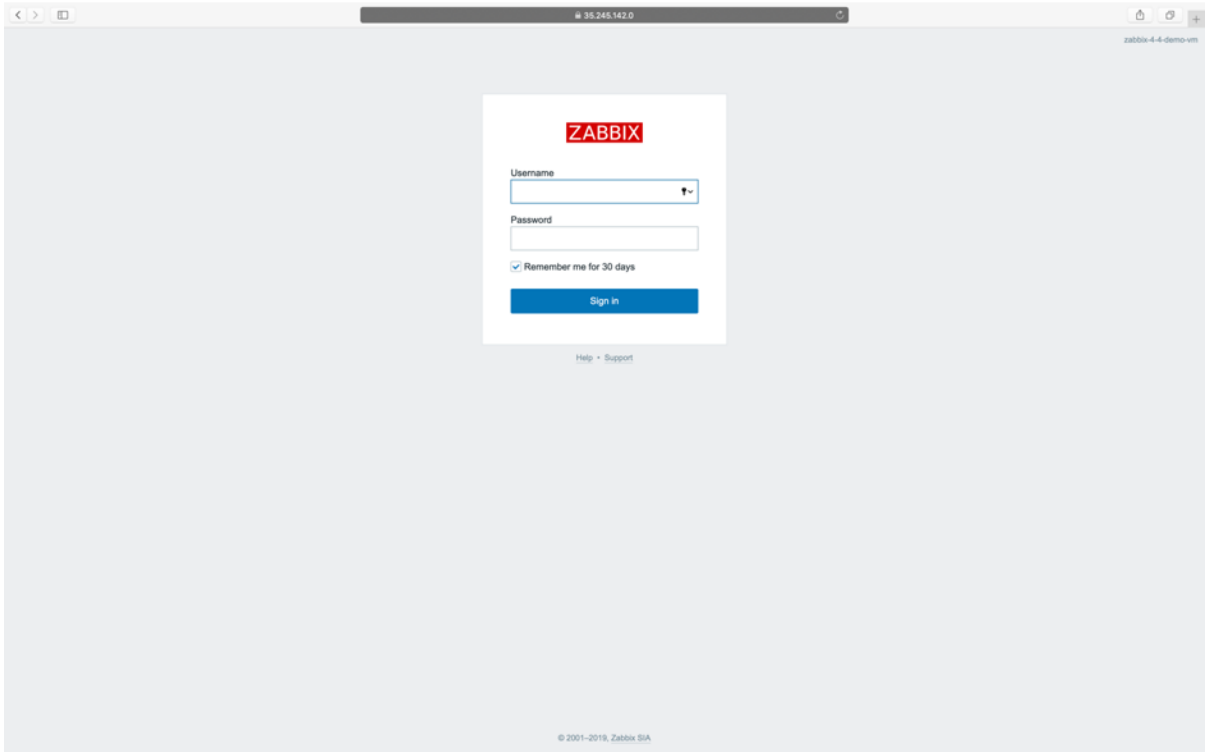
部署完成后，您应该会看到一条消息，提示 appliance 正在部署。

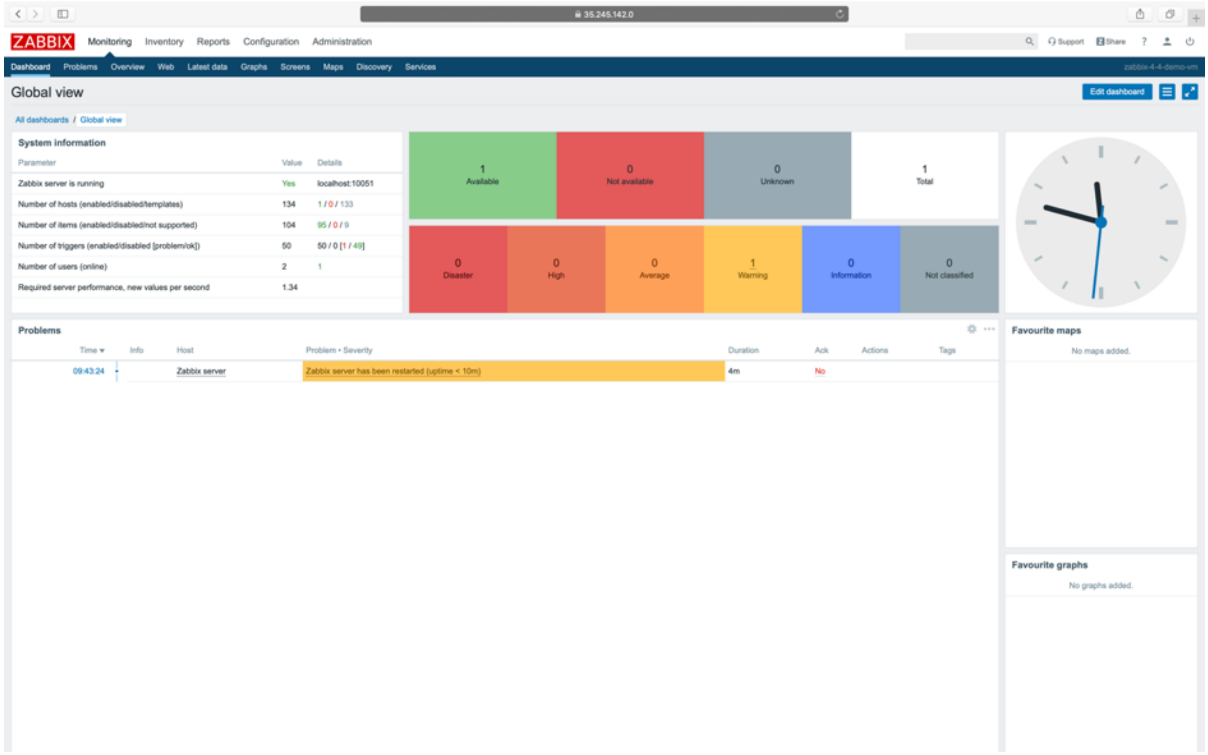


在 Web 浏览器中打开 *Site Address*，即可查看 Zabbix 前端。在此之前，请获取 Zabbix *Admin* 用户的密码，以登录 Zabbix appliance。

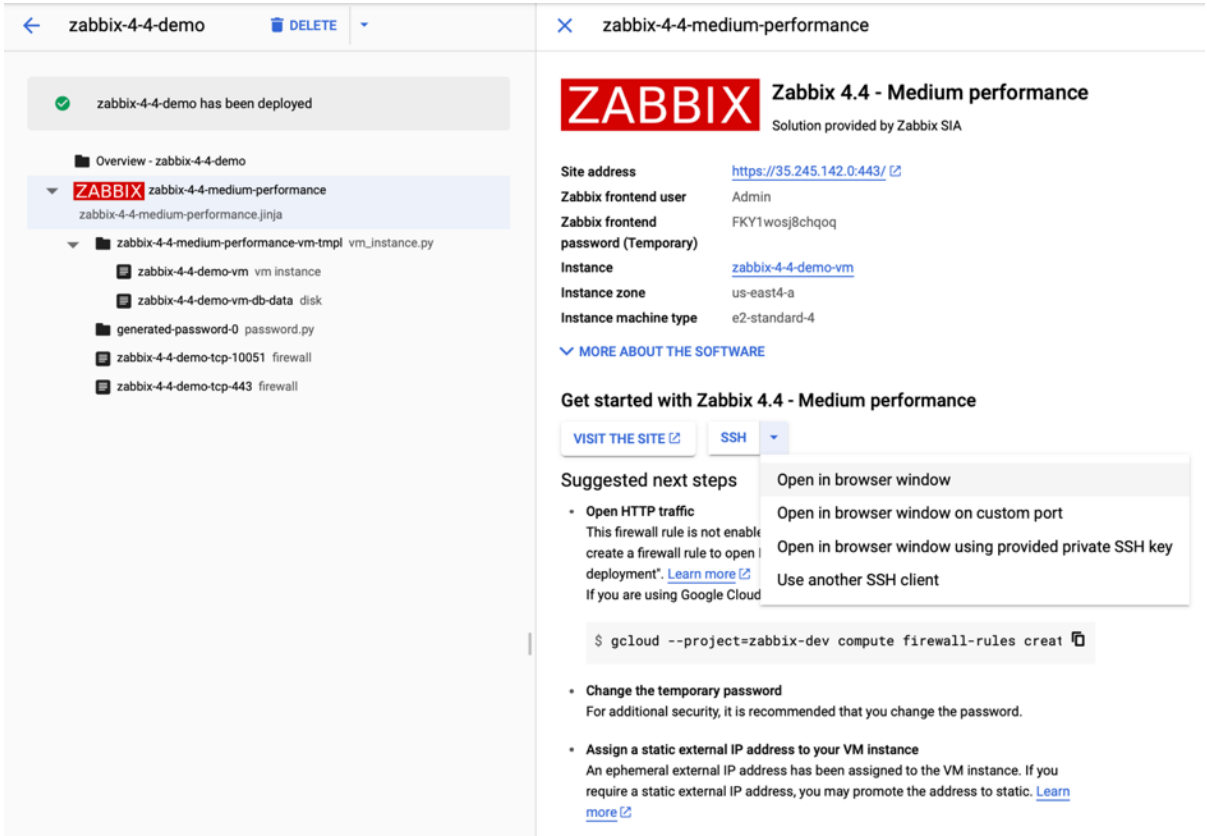


现在，您已经拥有 Zabbix appliance。





您还可以登录实例，与数据库进行交互。为此，请返回 Google 控制台，并按照下图中的红色箭头操作，使用 *Open in browser window* 选项启动 SSH 会话。



MySQL 无需 `root` 密码即可使用。密码存储在 `/root/.my.cnf` 配置文件中。默认情况下，Zabbix appliance 使用 UTC 时区。如果需要更改时区，请编辑 `/etc/php.d/99-zabbix.ini` 中的 `date.timezone` PHP 配置指令，然后重启 php-fpm 进程：`systemctl restart php-fpm`。

升级

在升级之前，强烈建议查阅所有相关的升级说明。

了解哪些组件必须升级也会很有帮助。

概述

以下提供了分步升级说明：

- Red Hat Enterprise Linux（使用软件包）
- Debian/Ubuntu（使用软件包）
- 使用容器升级
- 使用源代码升级

相关说明：

- 对于高可用性（HA）集群中的服务器，请参阅升级 HA 集群
- 对于 TimescaleDB 数据库，请参阅升级 TimescaleDB 架构

要升级的组件

建议升级 Zabbix agent，但并非强制要求。

强烈建议升级 Zabbix proxy。
Zabbix 服务器完全支持与服务器具有相同主版本的 proxy。
Zabbix 服务器也支持版本不早于 Zabbix 服务器上一 LTS 版本的 proxy，但功能受限（数据采集、执行远程命令、立即检查监控项值）。
配置更新也会被禁用，并且过时的 proxy 只能使用旧配置。

> **Attention:**
> 不支持比 Zabbix 服务器上一个 LTS 版本更旧或比 Zabbix 服务器主版本更新的 proxy。Zabbix 服务器将忽略来自不受支持的 proxy 的数据，并且与 Zabbix 服务器的所有通信都将失败并显示警告。有关更多信息，请参阅版本兼容性。

为尽量减少升级期间的停机时间和数据丢失，建议先停止、升级并启动 Zabbix 服务器，然后依次停止、升级并启动各个 Zabbix proxy。在服务器停机期间，正在运行的 proxy 将继续收集数据。服务器启动并运行后，过时的 proxy 会将数据发送到更新版本的服务器（但 proxy 配置不会更新），并将保持部分功能可用。Zabbix 服务器停机期间发生的问题，其所有通知都将在升级后的服务器启动后才生成。

如果 Zabbix proxy 首次启动时 SQLite 数据库文件不存在，proxy 会自动创建该文件。

请注意，如果 Zabbix proxy 使用 SQLite3，并且在启动时检测到现有数据库文件的版本低于所需版本，它将自动删除数据库文件并创建一个新文件。因此，存储在 SQLite 数据库文件中的历史数据将会丢失。如果 Zabbix proxy 的版本低于数据库文件版本，Zabbix 将记录错误并退出。

根据数据库大小，将数据库升级到 8.0 版本可能需要较长时间。

升级说明

支持从 **2.0**.x 版本直接升级到 Zabbix 8.0.x。如需从更早版本升级，请参阅 Zabbix 2.0 及更早版本的文档。

> **Note:**
> 请注意，升级后，Zabbix 中的某些第三方软件集成可能会受到影响，前提是外部软件与升级后的 Zabbix 版本不兼容。

以下升级说明可供参考：

从以下版本升级	阅读完整升级说明	版本之间最重要的变化
7.4.x	适用于： Zabbix 8.0	不再支持 1.6 之前版本的 Zabbix agent。 提高了所需的最低数据库版本。 所需的最低 PHP 版本从 8.0.0 提高到 8.2.0。 已将 % 字符添加到Zabbix agent 和Zabbix agent 2 的 `UnsafeUserParameters` 列表中。
7.2.x	适用于： Zabbix 7.4 Zabbix 8.0	不再支持 PCRE 库。
7.0.x	适用于： Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	不再支持 Oracle DB。

从以下版本升级	阅读完整升级说明	版本之间最重要的变化
6.4.x	适用于： Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	所需的最低 PHP 版本从 7.4.0 提高到 8.0.0。 支持 agent、HTTP agent、SNMP walk[OID] 和 SNMP get[OID] 检查的异步轮询器。 为 proxy 使用单独的数据库表。 更改 Windows agent 配置文件的默认位置。 Oracle DB 已弃用。 不再支持旧的数值（浮点）值类型。
6.2.x	适用于： Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	所需的最低 MySQL 版本从 8.0.0 提高到 8.0.30。 Zabbix 服务器/proxy 需要 `libevent_pthreads` 库。 升级后首次启动时，使用 SQLite3 的 Zabbix proxy 会自动删除旧版本的数据库（包括所有历史数据），并创建新数据库。
6.0.x LTS	适用于： Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	所需的最低 PHP 版本从 7.2.5 提高到 7.4.0。 服务监控经过重大重新设计。 升级期间需要创建确定性触发器。如果 MySQL/MariaDB 启用了二进制日志记录，则需要超级用户权限，或设置变量/配置参数 *log_bin_trust_function_creators = 1*。有关如何设置该变量的说明，请参阅数据库创建脚本。
5.4.x	适用于： Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	提高了所需的最低数据库版本。 如果数据库版本过低，服务器/proxy 将无法启动。 由于数据库结构发生变化，审计日志记录会丢失。
5.2.x	适用于： Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	提高了所需的最低数据库版本。 聚合监控项不再作为单独类型存在。
5.0.x LTS	适用于： Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	所需的最低 PHP 版本从 7.2.0 提高到 7.2.5。 密码哈希算法从 MD5 更改为 bcrypt。
4.4.x	适用于： Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	不再支持 IBM DB2。 所需的最低 PHP 版本从 5.4.0 提高到 7.2.0。 提高了所需的最低数据库版本。 更改了 Zabbix PHP 文件目录。

从以下版本升级	阅读完整升级说明	版本之间最重要的变化
4.2.x	适用于： Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	已移除 Jabber 和 Ez Texting 媒体类型。
4.0.x LTS	适用于： Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	旧版 proxy 无法再向升级后的服务器报告数据。 新版 agent 将无法再与旧版 Zabbix 服务器配合使用。
3.4.x	适用于： Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	现在必须使用 `libpthread` 和 `zlib` 库。 不再支持纯文本协议，且标头为必需项。 不再支持 1.4 之前版本的 Zabbix agent。 被动 proxy 配置中的 Server 参数现在为必需项。
3.2.x	适用于： Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	不再支持将 SQLite 作为 Zabbix 服务器/前端的后端数据库。 支持 Perl 兼容正则表达式（PCRE），取代 POSIX 扩展正则表达式。 Zabbix 服务器必须使用 `libpcre` 和 `libevent` 库。 为用户参数、远程命令、没有 `nowait` 标志的 system.run[] 监控项以及由 Zabbix 服务器执行的脚本添加退出代码检查。 必须升级 Zabbix Java 网关才能支持新功能。

从以下版本升级	阅读完整升级说明	版本之间最重要的变化
3.0.x LTS	适用于： Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	数据库升级可能较慢，具体取决于 history 表的大小。
2.4.x	适用于： Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	所需的最低 PHP 版本从 5.3.0 提高到 5.4.0。 必须指定 LogFile agent 参数。
2.2.x LTS	适用于： Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	已移除基于节点的分布式监控。

从以下版本升级	阅读完整升级说明	版本之间最重要的变化
2.0.x	适用于： Zabbix 2.2 Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4 Zabbix 7.0 Zabbix 7.2 Zabbix 7.4 Zabbix 8.0	所需的最低 PHP 版本从 5.1.6 提高到 5.3.0。 需要区分大小写的 MySQL 数据库才能确保服务器正常运行；Zabbix 服务器要与 MySQL 数据库正常配合使用，必须使用 utf8 字符集和 utf8_bin 排序规则。请参阅数据库创建脚本。 需要使用 `mysqli` PHP 扩展，而不是 `mysql`。

从软件包升级

概述

本节提供了使用 Zabbix 提供的官方 RPM 和 DEB 软件包成功执行升级所需的步骤，适用于：

- Red Hat Enterprise Linux
- Debian/Ubuntu

来自操作系统仓库的 Zabbix 软件包

某些操作系统发行版（特别是基于 Debian 的发行版）提供其自己的 Zabbix 软件包。
这些软件包不受 **Zabbix** 支持，并且可能已过时，或缺少最新功能和错误修复。
建议仅使用来自 Zabbix Official Repository 的官方软件包。

如果您要从操作系统发行版提供的软件包升级（或者曾经安装过这些软件包），请按照以下步骤切换到官方 Zabbix 软件包：

1. 卸载旧软件包。
2. 检查并删除旧安装遗留的所有文件。
3. 按照 Zabbix 提供的安装说明安装官方 Zabbix 软件包。

请勿直接升级，否则可能导致安装损坏。

Debian/Ubuntu

概述

本节提供了使用 Debian/Ubuntu 的 Zabbix 官方软件包，将 Zabbix **7.4.x** 升级到最新 Zabbix **8.0.x** 版本的说明。

Warning:
升级前，请查看相关的升级说明，并确保您的系统满足 Zabbix 8.0 的要求。

Note:
建议在升级期间同时运行两个并行的 SSH 会话：一个用于执行升级步骤，另一个用于监控服务器/proxy 日志。
例如，在第二个会话中运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，以实时查看最新的日志条目和可能出现的错误。
这对于生产环境实例来说可能至关重要。

有关在 Zabbix 8.0.x 次要版本之间升级的说明（例如，从 8.0.1 升级到 8.0.3），请参见次要版本之间升级。

升级程序

1 停止 Zabbix 进程

停止 Zabbix 服务器，以确保不会有新数据写入数据库：

```
systemctl stop zabbix-server
```

如果升级的是 Zabbix proxy、agent 或 agent 2，也请停止这些组件：

```
systemctl stop zabbix-proxy
systemctl stop zabbix-agent
systemctl stop zabbix-agent2
```

2 备份 Zabbix 数据库

备份现有的 Zabbix 数据库，以防升级失败（例如磁盘空间问题、断电或其他意外情况）。

3 备份 Zabbix 配置文件、PHP 文件和 Zabbix 二进制文件

备份现有的 Zabbix 配置文件、PHP 文件和 Zabbix 二进制文件。

对于配置文件，运行：

```
mkdir /opt/zabbix-backup/
cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
##### For Apache:
cp /etc/apache2/conf-enabled/zabbix.conf /opt/zabbix-backup/
##### For Nginx:
cp /etc/nginx/conf.d/zabbix.conf /opt/zabbix-backup/
cp /etc/zabbix/nginx.conf /opt/zabbix-backup/
```

对于 PHP 文件和 Zabbix 二进制文件，运行：

```
cp -R /usr/share/zabbix/ /opt/zabbix-backup/
cp -R /usr/share/zabbix-* /opt/zabbix-backup/
```

4 更新仓库配置包

在继续升级之前，请卸载当前的 Zabbix 仓库包：

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

您可能还需要在下载新包之前，手动删除工作目录中任何旧的 Zabbix 包(例如，`rm zabbix-release_latest+debian12_all.deb`)，以防止包管理器在升级过程中重复使用过时版本。

然后，安装最新的仓库配置包，以确保与最新包兼容，并包含任何近期的安全补丁或错误修复。

在 **Debian 13** 上，运行：

```
wget https://repo.zabbix.com/zabbix/8.0/release/debian/pool/main/z/zabbix-release/zabbix-release_latest+de
dpkg -i zabbix-release_latest+debian13_all.deb
```

在 **Debian 12** 上，运行：

```
wget https://repo.zabbix.com/zabbix/8.0/release/debian/pool/main/z/zabbix-release/zabbix-release_latest+de
dpkg -i zabbix-release_latest+debian12_all.deb
```

> **Note:**
> 对于较旧的 Debian 版本，请将上面的链接替换为 Zabbix repository 中对应的正确链接。但请注意，这些版本的软件包可能不包含所有 Zabbix 组件，如果要通过软件包升级这些组件，请考虑升级您的操作系统。有关包含的组件列表，请参见 Zabbix packages。

在 **Ubuntu 26.04** 上，运行：

```
wget https://repo.zabbix.com/zabbix/8.0/release/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest+ub
dpkg -i zabbix-release_latest+ubuntu26.04_all.deb
```

在 **Ubuntu 24.04** 上，运行：

```
wget https://repo.zabbix.com/zabbix/8.0/release/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest+ub
dpkg -i zabbix-release_latest+ubuntu24.04_all.deb
```

在 **Ubuntu 22.04** 上，运行：

```
wget https://repo.zabbix.com/zabbix/8.0/release/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest+ub
dpkg -i zabbix-release_latest+ubuntu22.04_all.deb
```

Note:
对于较旧的 Ubuntu 版本，请将上面的链接替换为 Zabbix repository 中对应的正确链接。但请注意，这些版本的软件包可能不包含所有 Zabbix 组件，如果要通过软件包升级这些组件，请考虑升级您的操作系统。有关包含的组件列表，请参见 Zabbix packages。

您可能会看到关于 Zabbix 仓库配置的提示：

```
Configuration file '/etc/apt/sources.list.d/zabbix.list'
==> Deleted (by you or by a script) since installation.
==> Package distributor has shipped an updated version.
What would you like to do about it ?  Your options are:
Y or I  : install the package maintainer's version
N or O  : keep your currently-installed version
D       : show the differences between the versions
Z       : start a shell to examine the situation
The default action is to keep your current version.
*** zabbix.list (Y/I/N/O/D/Z) [default=N] ?
```

输入 Y（或 I）以安装软件包维护者提供的 Zabbix 仓库配置版本。

然后，更新仓库信息：

```
apt update
```

5 升级 Zabbix 组件

要升级 Zabbix 组件，请运行：

```
apt install --only-upgrade zabbix-server-mysql zabbix-frontend-php zabbix-agent
```

- 如果使用 PostgreSQL，请在命令中将 `mysql` 替换为 `pgsql`。
- 如果升级 proxy，请在命令中将 `server` 替换为 `proxy`。
- 如果升级 Zabbix agent 2，请在命令中将 `zabbix-agent` 替换为 `zabbix-agent2 zabbix-agent2-plugin-*`。

Attention:
使用 `apt install zabbix-agent2` 命令升级 Zabbix agent 2 可能会导致错误。有关更多信息，请参阅已知问题。

您可能会看到有关 Zabbix 服务器（或 proxy）配置的提示：

```
Configuration file '/etc/zabbix/zabbix_server.conf'
==> Modified (by you or by a script) since installation.
==> Package distributor has shipped an updated version.
What would you like to do about it ?  Your options are:
Y or I  : install the package maintainer's version
N or O  : keep your currently-installed version
D       : show the differences between the versions
Z       : start a shell to examine the situation
The default action is to keep your current version.
*** zabbix_server.conf (Y/I/N/O/D/Z) [default=N] ?
```

请输入最符合您当前情况的选项。例如，输入 D 比较当前配置和新配置，然后决定是否要安装软件包维护者提供的版本（Y 或 I）。

然后，升级 Zabbix 前端并重启 Web 服务器。

对于 Apache，请运行：

```
apt install zabbix-apache-conf
systemctl restart apache2
```

对于 Nginx，请运行：

```
apt install zabbix-nginx-conf
systemctl restart nginx
```

对于 Nginx，您可能会看到有关 `nginx.conf` 配置文件的提示。请输入最符合您当前情况的选项：

```
Configuration file '/etc/zabbix/nginx.conf'
 ==> Modified (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
```

```
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** nginx.conf (Y/I/N/O/D/Z) [default=N] ?
```

6 检查组件配置参数

查看相关的升级说明，确认是否需要对配置参数进行任何更改。

有关新的可选参数，请参见新增内容页面。

如果使用 Nginx 作为 web 服务器，并且在升级过程中重新安装了 `/etc/zabbix/nginx.conf` 配置文件，请检查该文件，并更新 `listen` 和 `server_name` 指令，使其与 Zabbix 前端配置一致。

7 启动 Zabbix 进程

启动已升级的 Zabbix 组件：

```
systemctl start zabbix-server
systemctl start zabbix-proxy
systemctl start zabbix-agent
systemctl start zabbix-agent2
```

8 清除 Web 浏览器 Cookie 和缓存

升级后，您可能需要清除 Web 浏览器的 Cookie 和缓存，以使 Zabbix Web 界面正常工作。

在次要版本之间升级

可以在 Zabbix 8.0.x 的次要版本之间进行升级（例如，从 8.0.1 升级到 8.0.3）。

首先，更新软件仓库信息：

```
apt update
```

然后，要升级所有 Zabbix 组件，请运行：

```
apt install --only-upgrade 'zabbix*'
```

- 要仅升级 Zabbix 服务器，请将命令中的 `'zabbix*'` 替换为 `'zabbix-server*'`。
- 要仅升级 Zabbix proxy，请将命令中的 `'zabbix*'` 替换为 `'zabbix-proxy*'`。
- 要仅升级 Zabbix agent，请将命令中的 `'zabbix*'` 替换为 `'zabbix-agent*'`。
- 要仅升级 Zabbix agent 2，请将命令中的 `'zabbix*'` 替换为 `'zabbix-agent2*'`。

Red Hat Enterprise Linux

概述

本节提供了使用适用于 Red Hat Enterprise Linux 及其衍生版（AlmaLinux、CentOS Stream、Oracle Linux 和 Rocky Linux）的 Zabbix 官方软件包，将 Zabbix **7.4.x** 升级到最新版本 Zabbix **8.0.x** 的说明。

> **Warning:**
> 升级前，请查阅相关的升级说明，并确保您的系统满足 Zabbix 8.0 的要求。

> **Note:**
> 建议在升级过程中同时运行两个并行的 SSH 会话：一个用于执行升级步骤，另一个用于监控服务器/proxy 日志。例如，在第二个会话中运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，以实时查看最新日志条目和可能出现的错误。这对于生产环境实例可能至关重要。

有关在 Zabbix 8.0.x 次要版本之间升级的说明（例如，从 8.0.1 升级到 8.0.3），请参见次要版本之间升级。

升级过程

1 停止 Zabbix 进程

停止 Zabbix 服务器，以确保不会有新数据写入数据库：

```
systemctl stop zabbix-server
```

如果升级 Zabbix proxy、agent 或 agent 2，也请停止这些组件：

```
systemctl stop zabbix-proxy
systemctl stop zabbix-agent
systemctl stop zabbix-agent2
```

2 备份 Zabbix 数据库

备份现有的 Zabbix 数据库，以防升级失败（例如磁盘空间问题、断电或其他意外情况）。

3 备份 Zabbix 配置文件、PHP 文件和 Zabbix 二进制文件

备份现有的 Zabbix 配置文件、PHP 文件和 Zabbix 二进制文件。

对于配置文件，运行：

```
mkdir /opt/zabbix-backup/
cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
cp /etc/httpd/conf.d/zabbix.conf  /opt/zabbix-backup/
```

对于 PHP 文件和 Zabbix 二进制文件，运行：

```
cp -R /usr/share/zabbix/ /opt/zabbix-backup/
cp -R /usr/share/zabbix-* /opt/zabbix-backup/
```

4 更新仓库配置包

在继续升级之前，请将当前的仓库包更新到最新版本，以确保与最新软件包兼容，并包含最近的安全补丁或错误修复。

在 **RHEL 10** 上，运行：

```
rpm -Uvh https://repo.zabbix.com/zabbix/8.0/release/rhel/10/noarch/zabbix-release-latest.el10.noarch.rpm
```

在 **RHEL 9** 上，运行：

```
rpm -Uvh https://repo.zabbix.com/zabbix/8.0/release/rhel/9/noarch/zabbix-release-latest.el9.noarch.rpm
```

> **Note:**
> 对于较旧版本的 RHEL 或其衍生发行版，请将上面的链接替换为 Zabbix 仓库 中对应的正确链接。
> 但请注意，这些版本的软件包可能不包含所有 Zabbix 组件；如果要通过软件包升级这些组件，请考虑升级您的操作系统。
> 有关所包含组件的列表，请参见 Zabbix 软件包。

然后，清理 dnf 包管理器的缓存（包括先前安装或更新期间下载的头文件、元数据和软件包文件）：

```
dnf clean all
```

在下一次执行 dnf 操作时，由于旧的元数据已被清除，dnf 将从仓库重新下载最新的元数据。

另请参见：已知问题，了解如何在 RHEL 上更新仓库配置包。

5 升级 Zabbix 组件

要升级 Zabbix 组件，请运行：

```
dnf install zabbix-server-mysql zabbix-web-mysql zabbix-selinux-policy zabbix-agent
```

- 如果使用 PostgreSQL，请在命令中将 `mysql` 替换为 `pgsql`。
- 如果升级 proxy，请在命令中将 `server` 替换为 `proxy`。
- 如果升级 agent 2，请在命令中将 `zabbix-agent` 替换为 `zabbix-agent2 zabbix-agent2-plugin-*`。
- 如果还安装了其他 Zabbix 软件包，也请考虑将它们一并包含在命令中进行升级。

> **Attention:**
> 使用 `dnf install zabbix-agent2` 命令升级 Zabbix agent 2 可能会导致错误。有关更多信息，请参见已知问题。

然后，要使用 Apache 升级 Zabbix 前端并重启 Apache，请运行：

```
dnf install zabbix-apache-conf
systemctl restart httpd
```

6 审查组件配置参数

请查看相关的升级说明，以确认是否需要对配置参数进行任何更改。

有关新的可选参数，请参见新增内容页面。

7 启动 Zabbix 进程

启动已升级的 Zabbix 组件：

```
systemctl start zabbix-server
systemctl start zabbix-proxy
systemctl start zabbix-agent
systemctl start zabbix-agent2
```

8 清除 Web 浏览器 Cookie 和缓存

升级后，您可能需要清除 Web 浏览器的 Cookie 和缓存，以确保 Zabbix Web 界面正常工作。

在次要版本之间升级

可以在 Zabbix 8.0.x 次要版本之间进行升级（例如，从 8.0.1 升级到 8.0.3）。

要升级所有 Zabbix 组件，请运行：

```
dnf upgrade 'zabbix-*'
```

- 仅升级 Zabbix 服务器时，请将命令中的 'zabbix-*' 替换为 'zabbix-server-*'。
- 仅升级 Zabbix proxy 时，请将命令中的 'zabbix-*' 替换为 'zabbix-proxy-*'。
- 仅升级 Zabbix agent 时，请将命令中的 'zabbix-*' 替换为 'zabbix-agent-*'。
- 仅升级 Zabbix agent 2 时，请将命令中的 'zabbix-*' 替换为 'zabbix-agent2-*'。

从容器升级

概述

本节介绍成功将upgrade 到 Zabbix **8.0**.x 容器所需的步骤。

可分别参考以下说明：

- Zabbix 组件镜像升级
- Docker Compose 文件升级

Warning:
升级前，请务必阅读相关的升级说明！

Attention:
开始升级前，请确认用户对数据库具有升级所需的必要权限。

 对于从 Zabbix 6.0 或更早版本升级的情况，在升级过程中需要创建确定性触发器。如果 MySQL/MariaDB 启用了二进制日志记录，则这需要超级用户权限，或设置变量/配置参数 `log_bin_trust_function_creators = 1`。有关如何设置该变量的说明，请参见数据库创建脚本。

 请注意，如果从控制台执行，该变量只会被临时设置，并会在 Docker 重启后失效。在这种情况下，请保持 SQL 服务运行，仅通过执行 `docker compose down zabbix-server` 停止 `zabbix-server` 服务，然后执行 `docker compose up -d zabbix-server`。

 或者，也可以在配置文件中设置该变量。

根据数据库大小不同，升级数据库到 8.0 版本可能需要较长时间。

Zabbix 组件镜像升级

这些说明将升级用作 Zabbix 组件容器基础的 Docker 镜像。

以下步骤以 Zabbix proxy 为例。请将 `zabbix-proxy-sqlite3` 替换为您组件的容器名称。

1. 检查所有容器，以识别它们的名称和当前镜像版本：

```
docker ps -a

#### CONTAINER ID   IMAGE                                          (...) NAMES
#### 5a0c2e3d02b4   zabbix/zabbix-proxy-sqlite3:alpine-7.4-latest   (...) zabbix-proxy-sqlite3
```

2. 拉取新版本镜像：

```
docker pull zabbix/zabbix-proxy-sqlite3:alpine-8.0-latest
```

3. 停止 Zabbix 组件容器：

```
docker stop zabbix-proxy-sqlite3
```

4. 删除 Zabbix 组件容器：

> **Attention:**
> 在继续之前，请确保您已记录原始 `docker run` 命令中使用的所有配置标志和卷挂载，因为重新创建容器时需要它们。

```
docker rm zabbix-proxy-sqlite3
```

5. 使用更新后的镜像启动新容器：

```
docker run --name zabbix-proxy-sqlite3 \
  -e ZBX_SERVER_HOST=192.0.2.1 \
  -e ZBX_PROXYMODE=0 \
  -e ZBX_HOSTNAME=zabbix-proxy-sqlite3 \
  -v zabbix-proxy-data:/var/lib/zabbix/db_data \
  --init \
  -d \
  zabbix/zabbix-proxy-sqlite3:alpine-8.0-latest
```

> **Note:**
> 有关运行 Zabbix 容器的更多信息，请参见Docker（手动部署）。

6. 通过检查容器日志验证升级是否成功：

```
docker logs -f zabbix-proxy-sqlite3
```

Docker Compose 文件升级

这些说明用于升级部署 Zabbix 组件所使用的 Docker Compose 文件。如果您是使用Docker Compose 安装的 Zabbix，请按照这些说明操作。

1. 检查所有容器，以识别它们的名称和当前镜像版本：

```
docker ps -a

#### CONTAINER ID   IMAGE                                          (...) NAMES
#### d095f486deef   zabbix/zabbix-server-mysql:alpine-7.4-latest   (...) zabbix-docker-zabbix-server-1
#### 70627d833da0   zabbix/zabbix-web-nginx-mysql:alpine-7.4-latest (...) zabbix-docker-zabbix-web-ngina
#### aaaa664cc185   zabbix/zabbix-server-mysql:alpine-7.4-latest   (...) zabbix-docker-server-db-init-1
#### 9ab3e77492ff   mysql:8.0-oracle                               (...) zabbix-docker-mysql-server-1
```

2. 进入您之前克隆的 Zabbix Docker 仓库，拉取最新更新，并切换到 8.0 分支：

> **Attention:**
> 如果您修改了本地仓库中的 `.env`、`compose.yaml` 或其他文件，`git pull` 可能会覆盖这些更改。请在执行此步骤前备份任何本地修改。

```
cd zabbix-docker
git pull
git checkout 8.0
```

3. 使用更新后的 Docker Compose 文件部署 Zabbix 组件：

```
#### 使用 MySQL 作为数据库：
docker compose -f ./compose.yaml up -d

#### 使用 PostgreSQL 作为数据库：
docker compose -f ./compose_pgsql.yaml up -d
```

> **Note:**
> 在所有容器重新创建并启动后，数据采集可能需要 1-3 分钟才会恢复。有关运行 Zabbix 容器的更多信息，请参见Docker Compose。

4. 通过检查容器日志验证升级是否成功（将 `zabbix-docker-zabbix-server-1` 替换为您组件容器的名称）：

```
docker logs -f zabbix-docker-zabbix-server-1
```

从源码升级

概述

本节介绍使用 Zabbix 官方源将 Zabbix **7.4**.x 成功升级到 Zabbix **8.0**.x 所需的步骤。

Warning:
升级前请务必阅读相关的升级说明！

您还可以查看 8.0 的要求。

Note:
升级期间，同时运行两个并行的 SSH 会话可能会很方便：在一个会话中执行升级步骤，在另一个会话中监控服务器/proxy 日志。例如，在第二个 SSH 会话中运行 `tail -f zabbix_server.log` 或 `tail -f zabbix_proxy.log`，即可实时查看最新的日志条目和可能出现的错误。对于生产实例而言，这一点可能至关重要。

服务器升级过程

1 停止服务器

停止 Zabbix 服务器，以确保没有新数据写入数据库。

2 备份现有的 Zabbix 数据库

这是非常重要的一步。请确保已备份数据库。如果升级过程失败（磁盘空间不足、断电或任何意外问题），备份将有所帮助。

3 备份配置文件、PHP 文件和 Zabbix 二进制文件

备份 Zabbix 二进制文件、配置文件和 PHP 文件目录。

4 安装新的服务器二进制文件

使用这些说明从源代码编译 Zabbix 服务器。

5 检查服务器配置参数

请务必查看升级说明，确认是否需要更改配置参数。

有关新的可选参数，请参见新功能页面。

6 启动新的 Zabbix 二进制文件

启动新的二进制文件。
检查日志文件，确认二进制文件是否已成功启动。

Zabbix 服务器将自动升级数据库。
启动时，Zabbix 服务器会报告当前的数据库版本（必需和可选）以及所需的数据库版本。
如果当前必需版本低于所需版本，Zabbix 服务器会自动执行所需的数据库升级补丁。
数据库升级的开始信息和进度（百分比）会写入 Zabbix 服务器日志文件。
升级完成后，日志文件中会写入 “database upgrade fully completed” 消息。
如果任何升级补丁执行失败，Zabbix 服务器将不会启动。
如果当前必需的数据库版本高于所需版本，Zabbix 服务器也不会启动。
只有当当前必需的数据库版本与所需的必需版本一致时，Zabbix 服务器才会启动。

```
8673:20161117:104750.259 current database version (mandatory/optional): 03040000/03040000
8673:20161117:104750.259 required mandatory version: 03040000
```

启动服务器之前：

- 确保数据库用户具有足够的权限（创建表、删除表、创建索引、删除索引）
- 确保有足够的可用磁盘空间。

7 安装新的 Zabbix 前端

所需的最低 PHP 版本为 8.2.0。如有需要，请进行更新，并按照安装说明操作。

8 清除 Web 浏览器 Cookie 和缓存

升级后，您可能需要清除 Web 浏览器的 Cookie 和缓存，Zabbix 前端才能正常工作。

Proxy 升级过程

1 停止 proxy

停止 Zabbix proxy。

2 备份配置文件和 Zabbix Proxy 旧版本程序

备份配置文件和 Zabbix Proxy 旧版本程序。

3 安装新的 proxy 二进制文件

使用这些说明从源代码编译 Zabbix proxy。

4 检查 proxy 配置参数

请务必查看升级说明，确认是否需要更改配置参数。

有关新的可选参数，请参见新增内容 页面。

5 启动新的 Zabbix proxy

启动新的 Zabbix proxy。检查日志文件，确认 proxy 是否已成功启动。

Zabbix proxy 将自动升级数据库。数据库升级过程与启动Zabbix 服务器 时类似。

agent 升级流程

> **Attention:**
> 升级 agent 不是强制性的。只有在需要访问新功能时，才需要升级 agent。

本节介绍的升级过程可用于升级 Zabbix agent 和 Zabbix agent 2。

1 停止 agent

停止 Zabbix agent。

2 备份配置文件和 Zabbix agent 程序文件

备份配置文件和 Zabbix agent 程序文件。

3 安装新的 agent 二进制文件

使用这些说明从源代码编译 Zabbix agent。

或者，您也可以从 Zabbix 下载页面下载预编译的 Zabbix agent。

4 检查 agent 配置参数

请务必查看升级说明，确认是否需要对配置参数进行任何更改。

有关新的可选参数，请参见新功能页面。

5 启动新的 Zabbix agent

启动新的 Zabbix agent。检查日志文件，确认 agent 是否已成功启动。

在次版本之间升级

在 8.0.x 的次版本之间升级时（例如，从 8.0.1 升级到 8.0.3），需要对服务器/proxy/agent 执行与主版本升级期间相同的操作。唯一的区别是，在次版本之间升级时不会对数据库进行任何更改。

升级问题

转义相关　升级后函数参数超出字符限制

在 Zabbix 7.0 中，已为历史函数字符串参数添加了正确的反斜杠转义。

由于从 7.0 之前的 Zabbix 版本升级时会额外添加反斜杠，这会导致参数变长；如果参数长度超过 255 个字符的最大数据大小，可能会造成触发器函数损坏。

为了说明数据库升级期间可能出现的问题、这些问题的影响，以及如何识别和修复它们，下面以一个包含 3 个监控项和 2 个触发器的示例配置为例。

设置

1. Zabbix 6.0 版本。

2. 包含 114 个反斜杠并以 a 结尾的 /tmp/ONE 文件：

```
\\\
```

Trig_228：

```
find(/Zabbix server/vfs.file.contents[/tmp/ONE],1m:now,"regexp","\\\
```

双栈（IPv4/IPv6）环境中的模板兼容性

在双栈环境中（即系统同时配置为支持 IPv4 和 IPv6），主机名 `localhost` 通常会解析为 IPv4 和 IPv6 地址。由于许多操作系统和 DNS 解析器通常会优先使用 IPv6 而不是 IPv4，如果被监控的服务仅配置为监听 IPv4，则 Zabbix 模板可能无法正常工作。

未配置为监听 IPv6 地址的服务可能会变得无法访问，从而导致监控失败。用户可能已经正确配置了 IPv4 访问，但仍会因为默认优先使用 IPv6 的行为而遇到连接问题。

一种变通方法是，确保相关服务（Nginx、Apache、PostgreSQL 等）已配置为同时监听 IPv4 和 IPv6 地址，并允许 Zabbix 服务器/agent 通过 IPv6 访问。此外，在 Zabbix 模板和配置中，应显式使用 `localhost` 而不是 `127.0.0.1`，以确保同时兼容 IPv4 和 IPv6。

例如，在使用 PostgreSQL by Zabbix agent 2 模板监控 PostgreSQL 时，您可能需要编辑 `pg_hba.conf` 文件，以允许 `zbx_monitor` 用户建立连接。如果双栈环境优先使用 IPv6（系统将 localhost 解析为 `::1`），而您配置了 `localhost`，却只添加了一个 IPv4 条目（`127.0.0.1/32`），则连接将失败，因为没有匹配的 IPv6 条目。

以下 `pg_hba.conf` 文件示例可确保 `zbx_monitor` 用户能够从本地机器使用 IPv4 和 IPv6 地址并通过不同的认证方法连接到任意数据库：

```
### TYPE      DATABASE      USER             ADDRESS          METHOD
  host      all           zbx_monitor      localhost        trust
  host      all           zbx_monitor      127.0.0.1/32     md5
  host      all           zbx_monitor      ::1/128          scram-sha-256
```

如有必要，在为连接字符串配置 PostgreSQL by Zabbix agent 2 模板宏时，您也可以直接使用 IPv4 地址（`127.0.0.1`）。

意外安装 EPEL Zabbix 软件包

如果已安装并启用 EPEL 仓库，安装 Zabbix 软件包时可能会拉取 EPEL 版本，而不是官方 Zabbix 软件包。要解决此问题：

1. 删除所有从 EPEL 安装的 Zabbix 软件包：

```
dnf remove zabbix-server-mysql
```

2. 通过在 `/etc/yum.repos.d/epel.repo` 文件中添加以下行，将 EPEL 中的 Zabbix 软件包排除：

```
[epel]
...
excludepkgs=zabbix*
```

3. 重新安装官方 Zabbix 服务器软件包：

```
dnf install zabbix-server-mysql
```

在安装过程中，官方 Zabbix 软件包的版本字符串中包含单词 `release`（例如 `7.0.0-release1.el8`），以此与 EPEL 软件包区分开来。

适用于 Red Hat UBI 环境的 RHEL 上的 Zabbix 软件包

在 Red Hat Universal Base Image 环境中从 Red Hat Enterprise Linux 软件包安装 Zabbix 时，请确保能够访问所需的仓库和依赖项。Zabbix 软件包依赖 `libOpenIPMI.so` 和 `libOpenIPMIposix.so` 库，而 UBI 系统上默认启用的软件包管理器仓库中没有任何软件包提供这些库，这将导致安装失败。

`libOpenIPMI.so` 和 `libOpenIPMIposix.so` 库可在 `OpenIPMI-libs` 软件包中获得，该软件包由 `redhat-#-for-<arch>-appstream-rp` 仓库提供。
对此仓库的访问由订阅控制；在 UBI 环境中，这种访问是通过将 RHEL 主机的仓库配置和 secrets 目录挂载到容器文件系统命名空间中来传递的。

更多信息，请参见 ZBX-24291。

RHEL 软件包的签名密钥已过期

在Red Hat Enterprise Linux 或其衍生发行版上升级 Zabbix 时，您可能会遇到 Zabbix repository 中软件包签名密钥已过期的问题。
当签名密钥过期时，验证软件包签名的尝试将导致错误，提示证书或密钥已不再有效。
例如：

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

要解决此类问题，请为您所使用的 RHEL 具体变体手动重新安装最新的 `zabbix-release` 软件包（将下面的链接替换为 Zabbix repository 中对应的正确链接）。

例如，在 **RHEL 10** 上，运行：

```
rpm -Uvh https://repo.zabbix.com/zabbix/8.0/release/rhel/10/noarch/zabbix-release-latest.el10.noarch.rpm
```

然后，更新仓库信息：

```
dnf update
```

更多信息，请参见 ZBX-24761。

使用 MariaDB 的数据库 TLS 连接

如果使用的是 MariaDB，则 DBTLSConnect 参数 不支持 `verify_ca` 选项。

MySQL/MariaDB 中可能出现的死锁

在高负载运行且涉及多个 LLD worker 的情况下，可能会遇到由与行锁定策略相关的 InnoDB 错误导致的死锁（参见上游缺陷）。该错误已在 MySQL 8.0.29 起修复，但在 MariaDB 中尚未修复。更多详细信息，请参见 ZBX-21506。

全局事件关联

如果第一个和第二个事件之间的时间间隔非常短，即半秒及以下，事件可能无法被正确关联。

NetBSD 8.0 及更新版本

在 NetBSD 8.X 和 9.X 版本上，各种 Zabbix 进程在启动时可能会随机崩溃。这是由于默认栈大小（4MB）过小导致的，必须通过运行以下命令来增大：

```
ulimit -s 10240
```

更多信息请参见相关的问题报告：ZBX-18275。

Zabbix agent 2 中的正则表达式限制

由于标准的 Go regexp 库的限制，Zabbix agent 2 不支持正向预查和反向预查功能。

IPMI 检查

在 Debian 9（stretch）之前以及 Ubuntu 16.04（xenial）之前的版本中，IPMI 检查无法与标准 OpenIPMI 库软件包一起正常工作。
要解决此问题，请按照 ZBX-6139 中的说明，在启用 OpenSSL 的情况下重新编译 OpenIPMI 库。

IPMI — 不受信任的主机可能导致 OpenIPMI 崩溃

Zabbix 用于轮询 IPMI 数据的 OpenIPMI 库存在一个漏洞，不受信任的设备可以通过发送经过特殊构造的响应来触发该漏洞。不受信任的 IPMI 设备可能发送经过构造的数据，导致 OpenIPMI 库崩溃，进而可能导致执行 IPMI 轮询的 Zabbix 服务器进程终止。

SSH 检查

- 某些 Linux 发行版（如 Debian、Ubuntu）如果从软件包安装 libssh2 库，则不支持加密的私钥（带密码短语）。更多详细信息请参见 ZBX-4850。
- 在某些使用 OpenSSH 8 的 Linux 发行版上使用 libssh 0.9.x 时，SSH 检查可能会偶尔报告 “无法从 SSH 服务器读取数据”。这是由 libssh 的一个问题导致的（更详细的报告）。预计该错误已在稳定版 libssh 0.9.5 中修复。另请参见 ZBX-17756 了解详细信息。
- 在 SSH 脚本中使用管道符 “|” 可能会导致 “无法从 SSH 服务器读取数据” 错误。在这种情况下，建议升级 libssh 库版本。另请参见 ZBX-21337 了解详细信息。

ODBC 检查

- 如果可能，不应将 MySQL unixODBC 驱动程序与针对 MariaDB connector library 编译的 Zabbix 服务器或 Zabbix proxy 一起使用，反之亦然；同时，由于一个上游缺陷，最好也避免使用与驱动程序相同的 connector。
  推荐的配置：

  PostgreSQL、SQLite 或 Oracle connector → MariaDB 或 MySQL unixODBC 驱动程序
MariaDB connector → MariaDB unixODBC 驱动程序
MySQL connector → MySQL unixODBC 驱动程序

  更多信息及可用的变通方法，请参见 ZBX-7665。
- 从 Microsoft SQL Server 查询的 XML 数据在 Linux 和 UNIX 系统上可能会以多种方式被截断。
- 已观察到，使用 ODBC 检查通过多个版本的 Oracle Instant Client for Linux 监控 Oracle 数据库时，可能会导致 Zabbix 服务器崩溃。
 另请参见：ZBX-18402、ZBX-20803。
- 如果使用 FreeTDS UnixODBC 驱动程序，则需要在 SQL 查询前添加 `SET NOCOUNT ON` 语句（例如，`SET NOCOUNT ON DECLARE @strsql NVARCHAR(max) SET @strsql = ....`）。
  否则，Zabbix 中的数据库监控项将无法检索信息，并报错
  “SQL query returned empty result”。
 更多信息请参见 ZBX-19917。

监控项中的请求方法参数不正确

仅用于 HTTP 检查的请求方法参数，可能会因从 4.0 之前的 Zabbix 版本升级，而被错误地设置为 “1”，这对所有监控项来说都是一个非默认值。有关如何修复此问题的详细信息，请参见 ZBX-19308。

Web 监控和 HTTP agent

由于某些 Linux 发行版中的一个上游缺陷，当在 Web 场景或 HTTP agent 中启用 “SSL verify peer” 时，Zabbix 服务器会发生内存泄漏。更多信息及可用的变通方法，请参见 ZBX-10486。

简单检查

早于 v3.10 的 **fping** 版本中存在一个 bug，会错误处理重复的 echo replay 数据包。
这可能会导致 `icmpping`、`icmppingloss`、`icmppingsec` 监控项出现非预期结果。
建议使用最新版本的 **fping**。
更多详情请参见 ZBX-11726。

无 root 权限容器中执行 fping 时的错误

当容器以无 root 权限模式运行，或处于具有特定限制的环境中时，在执行 ICMP 检查时，您可能会遇到与 fping 执行相关的错误，例如 `fping: Operation not permitted`，或者所有资源的所有数据包均丢失。

要修复此问题，请在”docker run” 或”podman run” 命令中添加 `--cap-add=net_raw`。

此外，在非 root 环境中执行 fping 可能还需要修改 sysctl，即：

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

其中，“1995” 是 zabbix 的 GID。更多详细信息，请参见 ZBX-22833。

SNMP 检查

如果使用的是 OpenBSD 操作系统，Net-SNMP 库在 5.7.3 及以下版本中的一个 use-after-free 缺陷，可能会在 Zabbix 服务器配置文件中设置了 SourceIP 参数时导致 Zabbix 服务器崩溃。
作为一种临时解决方法，请不要设置 SourceIP 参数。
Linux 也存在同样的问题，但不会导致 Zabbix 服务器停止工作。
针对 OpenBSD 上 net-snmp 软件包的本地补丁已应用，并将随 OpenBSD 6.3 一同发布。

SNMP 数据尖峰

已观察到 SNMP 数据中出现尖峰，这可能与某些物理因素有关，例如市电电压尖峰。更多详情请参见 ZBX-14318。

SNMP trap

SNMP trap 所需的”net-snmp-perl” 软件包已在 RHEL 8.0-8.2 中移除；并在 RHEL 8.3 中重新添加。

因此，如果您使用的是 RHEL 8.0-8.2，最佳解决方案是升级到 RHEL 8.3。

另请参见 ZBX-17192 以获取更多信息。

RHEL 7 中 alerter 进程崩溃

在 RHEL 7 中，已发现 Zabbix 服务器 alerter 进程崩溃的情况。
详情请参见 ZBX-10461。

升级 Zabbix agent 2（6.0.5 或更早版本）

从软件包升级 Zabbix agent 2（版本 6.0.5 或更早）时，可能会出现与插件相关的文件冲突错误。要修复该错误，请备份您的 agent 2 配置（如有必要），卸载 agent 2，然后重新安装。

在基于 RHEL 的系统上，运行：

```
dnf remove zabbix-agent2
dnf install zabbix-agent2
```

在基于 Debian 的系统上，运行：

```
apt remove zabbix-agent2
apt install zabbix-agent2
```

更多信息，请参见 ZBX-23250。

前端语言环境切换

据观察，前端语言环境可能会在没有明显规律的情况下切换，即某些页面（或页面的某些部分）以一种语言显示，而其他页面（或页面的某些部分）则以另一种语言显示。通常，当存在多个用户，且其中一些用户使用一种语言环境、另一些用户使用另一种语言环境时，可能会出现此问题。

一种已知的解决方法是在 PHP 和 Apache 中禁用多线程。

该问题与 PHP 中语言环境设置的工作方式有关：语言环境信息按进程维护，而不是按线程维护。因此，在多线程环境中，当同一个 Apache 进程运行多个项目时，另一个线程可能会更改语言环境，从而影响 Zabbix 线程中数据的处理方式。

有关更多信息，请参阅相关问题报告：

- ZBX-10911（前端语言环境切换问题）
- ZBX-16297（使用 BC Math 函数中的 `bcdiv` 函数处理图形中的数字时出现问题）

图形

图形（经典）问题

如果您遇到经典图形相关问题，建议将 GD 库（libgd）更新到 2.3.3-13 或更高版本，并将 PHP 更新到 8.0.19、8.1.33、8.2.29、8.3.25、8.4.12 或更高版本。

夏令时

夏令时（DST）的变化会导致显示 X 轴标签时出现不规则情况（日期重复、日期缺失等）。

求和聚合

当在图形中对少于一小时的时间段使用求和聚合 时，如果数据来自趋势数据，图形将显示不正确的（被放大了的）值。

文本重叠

对于某些前端语言（例如日语），本地字体可能会导致图形图例中的文本重叠。
为避免这种情况，请使用 PHP GD 扩展 2.3.0（或更高）版本。

日志文件监控

如果文件系统已 100% 满，并且日志文件仍在持续追加内容，则 `log[]` 和 `logrt[]` 监控项会反复从头重新读取日志文件（更多信息请参见 ZBX-10884）。

MySQL 慢查询

当监控项不存在值时，Zabbix 服务器会生成较慢的 SELECT 查询。此问题已知会在 MySQL 5.6/5.7 版本中发生（有关扩展讨论，请参见 ZBX-10652），并且在某些特定情况下，也可能出现在较新的 MySQL 版本中。一种变通方法是在 MySQL 中禁用 `index_condition_pushdown` 或 `prefer_ordering_index` 优化器。但请注意，此变通方法可能无法解决与慢查询相关的所有问题。

来自链接的持久过滤器设置

当打开指向 Zabbix 前端页面且包含过滤器设置（包括时间选择器）的链接时，该过滤器会自动为用户保存到数据库中，并替换此前为该页面保存的过滤器和/或时间选择器设置。
这些设置会保持生效，直到用户手动更新或重置它们。

SNMPv3 trap 中的 IPv6 地址问题

由于 net-snmp 的一个缺陷，在 SNMP trap 中使用 SNMPv3 时，IPv6 地址可能无法正确显示。
有关更多详细信息和可能的变通方法，请参见 ZBX-14541。

登录失败信息中的长 IPv6 IP 地址将被截断

登录失败尝试消息只会显示已存储 IP 地址的前 39 个字符，因为这是数据库字段的字符长度限制。
这意味着，长度超过 39 个字符的 IPv6 IP 地址将无法完整显示。

Windows 上的 Zabbix agent 检查

Zabbix agent 配置文件（`zabbix_agentd.conf`）中 `Server` 参数里不存在的 DNS 条目可能会增加 Windows 上 Zabbix agent 的响应时间。这是因为 Windows DNS 缓存守护进程不会缓存 IPv4 地址的负响应。不过，对于 IPv6 地址，负响应会被缓存，因此一种可能的解决方法是在主机上禁用 IPv4。

YAML 导出/导入

YAML 导出/导入 存在一些已知问题：

- 错误消息不可翻译；
- 带有.yaml 文件扩展名的有效 JSON 有时无法导入；
- 未加引号的人类可读日期会自动转换为 Unix 时间戳。

SUSE 上使用 NGINX 和 php-fpm 的安装向导

前端安装向导无法在 SUSE 上使用 NGINX + php-fpm 保存配置文件。这是由于 `/usr/lib/systemd/system/php-fpm.service` 单元中的一项设置导致的，该设置阻止 Zabbix 向 `/etc` 写入内容（在 PHP 7.4 中引入）。

可用的临时解决方案有两种：

- 将 php-fpm systemd 单元中的 ProtectSystem 选项从'full' 改为'true'。
- 手动保存 `/etc/zabbix/web/zabbix.conf.php` 文件。

Authorization 标头转发

在某些情况下，Apache 或 NGINX 可能会阻止 API 请求中的 Authorization 标头到达 Zabbix。这可能会在使用 Zabbix API 或单点登录（SSO）服务时导致身份验证问题，例如配合 Okta 使用的 SAML。

为解决此问题，请更新 Web 服务器配置。

对于 **Apache**，如果您将其用作反向 proxy（非 CGI 设置），请将以下指令添加到 `/etc/httpd/conf/httpd.conf`（基于 RHEL 的系统）或 `/etc/apache2/apache2.conf`（Debian/Ubuntu）中：

```
SetEnvIfNoCase ^Authorization$ "(.+)" HTTP_AUTHORIZATION=$1
```

如果 Apache 直接执行脚本来处理请求（例如使用 mod_cgi），请改为添加以下指令：

```
CGIPassAuth On
```

相比之下，**NGINX** 会自动处理 Authorization 标头。但是，如果 NGINX 充当反向 proxy，您也可以通过将以下指令添加到 `/etc/nginx/nginx.conf`（针对您的 Zabbix 前端 location）中来显式转发 Authorization 标头：

```
...
location / {
...
    proxy_set_header Authorization $http_authorization;
    proxy_pass http://backend_server;
...
}
```

更新配置后，请重启您的 Web 服务器。

更多详细信息，请参见：

- ZBX-22952
- Apache 2.4 + PHP-FPM and Authorization headers
- SetEnvIfNoCase 和 CGIPassAuth 指令
- NGINX Reverse Proxy

Ubuntu 20 上用于 Zabbix Web 服务的 Chromium

尽管在大多数情况下，Zabbix Web 服务可以使用 Chromium 运行，但在 Ubuntu 20.04 上使用 Chromium 会导致以下错误：

```
Cannot fetch data: chrome failed to start:cmd_run.go:994:
WARNING: cannot create user data directory: cannot create
"/var/lib/zabbix/snap/chromium/1564": mkdir /var/lib/zabbix: permission denied
Sorry, home directories outside of /home are not currently supported. See https://forum.snapcraft.io/t/112
```

出现此错误的原因是，`/var/lib/zabbix` 被用作用户 "zabbix" 的主目录。

MySQL 自定义错误代码

当 Zabbix 检测到后端数据库无法访问时，它会发送通知并继续尝试连接。对于某些数据库引擎，会识别特定的错误代码。在 MySQL 中，这些可识别的错误代码包括：

- CR_CONN_HOST_ERROR
- CR_SERVER_GONE_ERROR
- CR_CONNECTION_ERROR
- CR_SERVER_LOST
- CR_UNKNOWN_HOST
- ER_SERVER_SHUTDOWN
- ER_ACCESS_DENIED_ERROR
- ER_ILLEGAL_GRANT_FOR_TABLE
- ER_TABLEACCESS_DENIED_ERROR
- ER_UNKNOWN_ERROR

此外，在 Azure 上将 Zabbix 与 MySQL 安装一起使用时，Zabbix 日志中可能会出现通用错误消息 *[9002] Some errors occurred*。该消息由数据库发送到 Zabbix 服务器或 proxy。要确定错误原因，请查阅 Azure 日志。

切换到 PCRE2 后无效的正则表达式

在 Zabbix 6.0 中，已添加对 PCRE2 的支持。
尽管仍然支持 PCRE，但适用于 RHEL 7 及更新版本、SLES（所有版本）、Debian 9 及更新版本、Ubuntu 16.04 及更新版本的 Zabbix 安装包已更新为使用 PCRE2。

虽然带来了许多好处，但切换到 PCRE2 可能会导致某些现有的 PCRE 正则表达式模式变为无效，或表现出不同的行为。
特别是，这会影响模式 *[\w-\.]*。为了在不影响语义的情况下使该正则表达式再次有效，请将表达式改为 *[-\w\.]* 。
这是因为 PCRE2 将连字符视为分隔符，从而在字符类中创建了一个范围。

Geomap 小组件错误

如果你从较旧版本的 Zabbix 升级，并且使用了 NGINX，但在升级过程中没有切换到新的 NGINX 配置文件，那么 Geomap 小组件中的地图可能无法正确加载。

要解决此问题，你可以丢弃旧的配置文件，使用当前版本软件包中的配置文件，并按照 下载说明 中 *e.* 为 *Zabbix* 前端配置 *PHP* 一节所述重新进行配置。

或者，你也可以手动编辑现有的 NGINX 配置文件（通常为 `/etc/zabbix/nginx.conf`）。为此，打开该文件并找到以下块：

```
location ~ /(api\/|conf[^\.]|include|locale|vendor) {
        deny            all;
        return          404;
}
```

然后，将此块替换为：

```
location ~ /(api\/|conf[^\.]|include|locale) {
        deny            all;
        return          404;
}

location /vendor {
        deny            all;
        return          404;
}
```

预处理——全局变量不安全

预处理 JavaScript 按请求执行，但对未声明标识符的赋值（例如 `secret = value`）会创建隐式全局变量，这些变量可能在当前执行结束后仍然存在。将敏感数据（令牌、密码等）存储在隐式全局变量中，会增加被后续预处理运行或在同一环境中执行的其他集成意外暴露或复用的风险。

不要依赖隐式全局变量。始终使用 `var` 或 `const` 声明变量，并避免将敏感信息附加到全局对象（例如 `globalThis` 或 `window`）上。在预处理中，没有受支持的方法可以从内部覆盖内置全局对象。

安全示例：

```
var apiToken = payload.token;
var count = 1;
return JSON.stringify({ token: apiToken, calls: count });
```

Windows 上的处理器组

Microsoft 文档指出，逻辑处理器少于 64 个的系统始终只有一个处理器组，即 Group 0。但是，Zabbix 用户报告了一个罕见的 bug：ZBX-20260，即在逻辑处理器数量不超过 64 个的系统上存在两个处理器组。这导致两个处理器组中只有一个处理器组具有"\Processor(n)" 性能计数器。该 bug 的实际根本原因尚不清楚。不过，在 stackoverflow.com 上描述了一个类似案例，其根本原因是 BIOS 与 Windows 之间的交互操作。

使用 utf8mb4 排序规则进行过滤的限制

当过滤器（例如，在 数据采集 > 维护 中）应用于包含某些 Unicode 字符（例如 𝓮、𝔢）的实体时，可能无法正常工作。
此问题源于 MySQL 或 MariaDB 数据库默认的 utf8mb4_bin 排序规则处理 Unicode 字符排序和比较的方式。

为解决此限制，用户可以将数据库列的排序规则更改为 utf8mb4_0900_bin、utf8mb4_0900_ai_ci 或 utf8mb4_unicode_520_ci 等替代方案。
但请注意，更改排序规则可能会导致空格处理以及其他字符的排序和过滤出现意外行为。

有关更改排序规则的更多信息，请参见 MySQL 文档 或 MariaDB 文档。
有关排序规则差异的详细信息，请参见 MySQL 文档中的 Unicode Character Sets。

使用 tcmalloc 分析过高的内存使用

如果您怀疑 Zabbix 安装占用了过多内存，可以使用 tcmalloc 的内存分析功能来调查 Zabbix 服务器/proxy 的内存消耗。

1. 在从源码 安装 Zabbix 时，请配置额外的标志：

```
export CFLAGS="-std=gnu99 -g -O0"
```

-std=gnu99 标志是构建 Zabbix 服务器、Zabbix proxy 或 Zabbix agent 所必需的。
-g 标志会添加额外的调试信息，而 -O0 会禁用优化，因为优化可能会干扰 tcmalloc 的分析。

2. 在启动 Zabbix 服务器之前，设置以下环境变量。
这些变量会告诉 tcmalloc 如何跟踪并报告内存使用情况：

```
LD_PRELOAD="/usr/lib/aarch64-linux-gnu/libtcmalloc.so" \
HEAPPROFILE=./heap_profile \
HEAP_PROFILE_ALLOCATION_INTERVAL=0 \
HEAP_PROFILE_INUSE_INTERVAL=4294967296 \
HEAPPROFILESIGNAL=5 \
MALLOCSTATS=1 \
./sbin/zabbix_server -f -c /etc/zabbix/zabbix_server.conf
```

3. 向目标进程发送信号 5，以触发分析转储。
将 1234 替换为实际的进程 ID（PID）：

```
kill -5 1234
```

4. 打印生成的分析文件：

```
pprof-symbolize -text ./sbin/zabbix_server ./heap_profile.0001.heap

Using local file ./sbin/zabbix_server.
Using local file ./heap_profile.0001.heap.
Total: 1078.1 MB
  1076.8  99.9%  99.9%   1076.8  99.9% zbx_malloc2
     1.0   0.1% 100.0%      1.0   0.1% __GI___strdup
     0.2   0.0% 100.0%      0.2   0.0% CRYPTO_zalloc@@OPENSSL_3.0.0
     0.1   0.0% 100.0%      0.1   0.0% OPENSSL_LH_insert@@OPENSSL_3.0.0
     0.0   0.0% 100.0%      0.0   0.0% zbx_realloc2
     0.0   0.0% 100.0%      0.1   0.0% PKCS7_decrypt@@OPENSSL_3.0.0
     0.0   0.0% 100.0%      0.0   0.0% find_best_tree_node
     0.0   0.0% 100.0%      0.0   0.0% CRYPTO_strndup@@OPENSSL_3.0.0
     ...
     0.0   0.0% 100.0%      0.0   0.0% preprocessing_flush_value
     0.0   0.0% 100.0%   1074.0  99.6% preprocessor_add_request
```

在此示例中，zbx_malloc2 负责了几乎所有内存分配。

另请参见：

- ZBX-25050 和 ZBX-25584，了解相关问题报告。
- GCC Option Summary 中关于编译选项（-std=gnu99、-g、-O0 等）的说明。
- Gperftools Heap Profiler 中关于 tcmalloc 分析环境变量的文档。

MySQL 8.0 组复制在多主模式下

在多主模式下使用 MySQL 8.0 组复制时，您可能会在事务提交期间遇到类似以下的错误：

```
1531697:20250128:064734.697 query [txnlev:1] [update alerts set status=1,retries=0,error='' where alertid=
1531697:20250128:064734.713 query [txnlev:1] [commit;]
1531697:20250128:064734.753 [Z3005] query failed: [3101] Plugin instructed the server to rollback the curr
```

该错误似乎是由涉及外键约束的回滚操作问题触发的。

另请参见：

- 有关相关问题报告，请参见 ZBX-26060。
- 有关上游问题，请参见 MySQL Bug #96758 "Rollbacks with Foreign Keys on single node"。

登录和 systemd

我们建议将创建 的 *zabbix* 用户设置为系统用户，即不允许其登录。一些用户忽略了这一建议，使用同一个账户登录运行 Zabbix 的主机（例如使用 SSH）。这可能会导致用户注销时 Zabbix 守护进程崩溃。在这种情况下，您将在 Zabbix 服务器日志中看到类似以下内容：

```
zabbix_server [27730]: [file:'selfmon.c',line:375] lock failed: [22] Invalid argument
zabbix_server [27716]: [file:'dbconfig.c',line:5266] lock failed: [22] Invalid argument
zabbix_server [27706]: [file:'log.c',line:238] lock failed: [22] Invalid argument
```

在 Zabbix agent 日志中看到：

```
zabbix_agentd [27796]: [file:'log.c',line:238] lock failed: [22] Invalid argument
```

这是因为 /etc/systemd/logind.conf 中配置了 systemd 的默认设置 RemoveIPC=yes。当您注销系统时，之前由 Zabbix 创建的信号量会被删除，从而导致崩溃。

以下是 systemd 文档中的说明：

```
RemoveIPC=

Controls whether System V and POSIX IPC objects belonging to the user shall be removed when the
user fully logs out. Takes a boolean argument. If enabled, the user may not consume IPC resources
after the last of the user's sessions terminated. This covers System V semaphores, shared memory
and message queues, as well as POSIX shared memory and message queues. Note that IPC objects of the
root user and other system users are excluded from the effect of this setting. Defaults to "yes".
```

此问题有两种解决方案：

1. （推荐）不要将 *zabbix* 账户用于 Zabbix 进程以外的任何用途，为其他用途创建专用账户。
2. （不推荐）在 /etc/systemd/logind.conf 中设置 RemoveIPC=no，然后重启系统。请注意，RemoveIPC 是系统范围的参数，更改它将影响整个系统。

要求

硬件

内存

Zabbix 同时需要物理内存和磁盘内存。所需的磁盘内存大小显然取决于被监控的主机数量和参数数量。如果您计划长期保留监控参数的历史记录，则应考虑至少准备几 GB 的空间，以便在数据库中存储这些历史记录。每个 Zabbix 守护进程都需要与数据库服务器建立多个连接。为连接分配的内存量取决于数据库引擎的配置。

> **Note:**
> 物理内存越多，数据库（因此 Zabbix）的运行速度就越快。

CPU

Zabbix，尤其是 Zabbix 数据库，可能会根据监控参数的数量和所选数据库引擎而需要大量 CPU 资源。

其他硬件

在 Zabbix 中使用短信通知支持功能需要串行通信端口和串行 GSM 调制解调器。
USB 转串口转换器也可以使用。

硬件配置示例

下表提供了硬件配置示例，假设使用 **Linux/BSD/Unix** 平台。

以下是用于开始部署的规模和硬件配置示例。每个 Zabbix 安装环境都是独特的。请务必在预发布或开发环境中对 Zabbix 系统的性能进行基准测试，以便在将 Zabbix 安装部署到生产环境之前，充分了解您的需求。

安装规模	监控指标[1]	CPU/vCPU 核心数	内存 (GiB)	数据库	Amazon EC2[2]
小型	1 000	2	8	MySQL Server, Percona Server, MariaDB Server, PostgreSQL	m6i.large/m6g.large
中型	10 000	4	16	MySQL Server, Percona Server, MariaDB Server, PostgreSQL	m6i.xlarge/m6g.xlarge
大型	100 000	16	64	MySQL Server, Percona Server, MariaDB Server, PostgreSQL	m6i.4xlarge/m6g.4xlarge

安装规模	监控指标[1]	CPU/vCPU 核心数	内存 (GiB)	数据库	Amazon EC2[2]
超大型	1 000 000	32	96	MySQL Server, Percona Server, MariaDB Server, PostgreSQL	m6i.8xlarge/m6g.8xlarge

[1] 出于容量规划的目的，1 个监控指标大致可用以下公式表示：1 个指标 = 1 个监控项 + 1 个触发器 + 1 个图形。此规则仅用于估算，并非用于计算监控指标数量的公式。
 [2] 这是使用 ARM64 或 x86_64 架构的 Amazon 通用型 EC2 实例示例。在将 Zabbix 安装到生产环境之前进行安装评估和测试时，应选择适当的实例类型，例如计算优化型、内存优化型或存储优化型实例。

Note:
实际配置在很大程度上取决于活动监控项的数量和刷新频率（详情请参阅本页的数据库大小部分）。对于大型安装，强烈建议在单独的服务器上运行数据库。

支持的平台

由于安全要求以及监控服务器的关键任务性质，只有 UNIX 操作系统才能持续提供所需的性能、容错能力和弹性。Zabbix 可运行于市场主流版本之上。

以下平台提供并经过测试的 Zabbix 组件如下：

平台	服务器	agent	Agent 2	备注
Linux	x	x	x	
Windows	-	x	x	Zabbix agent 支持自 Windows XP（64 位）/Server 2003 起的所有桌面版和服务器版。 Zabbix agent 2 支持自 Windows 10（32 位）/Server 2016 起的所有桌面版和服务器版，因为它仅使用受支持的 Go 版本进行编译，以防止严重的安全漏洞。自 Go 1.21 起，Windows 的最低要求版本已提高，因此 Windows 10/Server 2016 成为 Zabbix agent 2 的最低支持版本。
macOS	x	x	-	
IBM AIX	x	x	-	Zabbix agent 在低于 6.1 TL07 / 7.1 TL01 的 AIX 平台版本上无法运行。
FreeBSD	x	x	-	
OpenBSD	x	x	-	
Solaris	x	x	-	
NetBSD	x	x	-	
HP-UX	x	x	-	

Note:
Zabbix 服务器/agent 也可能在其他类 Unix 操作系统上运行。

Attention:
如果 Zabbix 在启用加密的情况下编译，则会禁用 core dump；如果系统不允许禁用 core dump，则 Zabbix 将无法启动。

所需软件

Zabbix 基于现代 Web 服务器、主流数据库引擎和 PHP 脚本语言构建。

第三方外部配套软件

如果标记为必需，则表示所需的软件/库是严格必要的。
可选软件用于支持某些特定功能。

软件	必需状态	支持的版本	备注
MySQL/Percona	二选一	8.4.0-9.7.X	如果使用 MySQL（或 Percona）作为 Zabbix 后端数据库，则为必需。需要 InnoDB 引擎。 我们建议使用 C API (libmysqlclient) 库构建服务器/proxy。 我们强烈建议仅在生产环境中使用 LTS 版本。不建议使用标记为 “滚动发布”、“早期访问” 或类似版本的发行版。
MariaDB		10.11.00-12.3.X	需要 InnoDB 引擎。 我们建议使用 MariaDB Connector/C 库构建服务器/proxy。 另请参阅：MariaDB 可能发生的死锁和使用 MariaDB 10.5.1-10.5.9 访问 UI 元素。 我们强烈建议仅在生产环境中使用 LTS 版本。不建议使用标记为 “滚动发布”、“早期访问” 或类似版本的发行版。
PostgreSQL		15.0-18.X	如果使用 PostgreSQL 作为 Zabbix 后端数据库，则为必需。 根据安装规模，可能需要增大 PostgreSQL 的 *work_mem* 配置属性（默认值为 4MB），以确保数据库为特定操作使用足够的内存，并避免查询执行耗时过长。
用于 PostgreSQL 的 *TimescaleDB*		2.20.X-2.29.X	如果将 TimescaleDB 用作 PostgreSQL 数据库扩展，则为必需。请确保安装支持压缩的 TimescaleDB Community Edition。 有关 PostgreSQL 和 TimescaleDB 版本兼容性的详细信息，请参阅 TimescaleDB 文档。
Elasticsearch		7.X	如果使用 Elasticsearch 作为 Zabbix 后端数据库，则为必需。 Elasticsearch 只能用于存储历史数据，并且仅支持 Zabbix 服务器。目前对 Elasticsearch 的支持仍处于实验阶段。另请参阅服务器/proxy 所需的软件。
ClickHouse		26.3.0.0-26.9.X.X	如果将 ClickHouse 用作 Zabbix 后端数据库以存储历史数据（仅支持 Zabbix 服务器），或用作遥测数据源，则为必需。 要将 ClickHouse 用作遥测数据源，需要 ClickHouse 26.8.17.4 或更高版本。此外，Zabbix 服务器和 ClickHouse 服务器上的系统时间必须同步；否则，遥测查询可能遗漏数据，或两次返回相同的数据。
SQLite	可选	3.3.5-3.53.X	SQLite 仅支持 Zabbix proxy。如果使用 SQLite 作为 Zabbix proxy 数据库，则为必需。
smartmontools		7.1 或更高版本	Zabbix agent 2 所需。
who			用户计数插件所需。
dpkg			system.sw.packages 插件所需。
pkgtool			system.sw.packages 插件所需。
rpm			system.sw.packages 插件所需。
pacman			system.sw.packages 插件所需。
q applets			在 Gentoo Linux 上，作为 q applets 一部分的 `qlist` 和 `qsize` 是 system.sw.packages 插件所需的。

Note:
尽管 Zabbix 可以使用操作系统中提供的数据库，但为了获得最佳体验，我们建议使用从数据库开发者官方软件仓库安装的数据库。

前端

Zabbix 前端支持的最小屏幕宽度为 1200px。

如果标记为必需，则表示该软件/库是严格必需的。可选软件/库用于支持某些特定功能。

软件	必需状态	支持的版本	备注
PHP	是	8.2.0 - 8.5.X	

软件	必需状态	支持的版本	备注
Apache	二选一	2.4 或更高	
Nginx		1.20 或更高	
MySQL	二选一	参见第三方外部周边软件	
PostgreSQL			
PHP 扩展			
mysqli	是		如果使用 MySQL 作为 Zabbix 后端数据库，则必需。
pgsql			如果使用 PostgreSQL 作为 Zabbix 后端数据库，则必需。
bcmath			php-bcmath (*--enable-bcmath*)
mbstring			php-mbstring (*--enable-mbstring*)
sockets			php-net-socket (*--enable-sockets*)；支持用户脚本时必需。
gd		2.0.28 或更高	php-gd（如果发行版提供单独的软件包）；PHP GD 扩展必须支持 PNG 图像（*--with-png-dir*）、JPEG 图像（*--with-jpeg-dir*）和 FreeType 2（*--with-freetype-dir*）。对于某些前端语言，可能需要 2.3.0 或更高版本，以避免图表中的文本重叠问题。
libxml		2.6.15 或更高	php-xml（如果发行版提供单独的软件包）
xmlwriter			php-xmlwriter（如果发行版提供单独的软件包）
xmlreader			php-xmlreader（如果发行版提供单独的软件包）
ctype			php-ctype (*--enable-ctype*)
session			php-session（如果发行版提供单独的软件包）
ldap	否		php-ldap；LDAP 身份验证所需。
openssl			php-openssl；SAML 身份验证所需。
gettext			php-gettext (*--with-gettext*)；翻译所需。
cURL		7.19.4 或更高	php-curl；Duo Universal Prompt MFA、SMTP 身份验证以及测试遥测数据源所需。

Zabbix 随附的第三方前端库：

库	必需状态	随附版本	备注
jQuery JavaScript Library	是	3.6.0	用于简化跨浏览器开发过程的 JavaScript 库。
jQuery UI		1.12.1	基于 jQuery 构建的一组用户界面交互、效果、控件和主题。
SAML PHP Toolkit		4.3.1	用于为 Zabbix 添加 SAML 2.0 身份验证支持的 PHP 工具包。
Duo Universal PHP library		1.1.2	用于为 Zabbix 添加 Duo Universal Prompt 多因素身份验证支持的 PHP 库。
Symfony Yaml Component		5.1.0	支持以 YAML 格式导出和导入 Zabbix 配置元素。

Note:
Zabbix 也可能可以在较旧版本的 Apache、MySQL 和 PostgreSQL 上运行。

Attention:
对于默认 DejaVu 以外的字体，可能需要 PHP 函数 imagerotate。如果缺少该函数，在显示图表时，这些字体可能无法正确渲染。只有在 PHP 使用内置 GD 进行编译时，该函数才可用；Debian 和其他发行版并非如此。

用于编写和调试 Zabbix 前端代码的第三方库：

库	必需状态	最低版本	描述
Composer	否	2.4.1	用于 PHP 的应用级软件包管理器，为管理 PHP 软件及其所需库的依赖关系提供标准格式。
PHPUnit		8.5.29	用于测试 Zabbix 前端的 PHP 单元测试框架。
SASS		3.4.22	一种预处理脚本语言，可解释并编译为层叠样式表（CSS）。

客户端侧的 Web 浏览器

必须启用 Cookie 和 JavaScript。

支持 Google Chrome、Mozilla Firefox、Microsoft Edge、Apple Safari 和 Opera 的最新稳定版本。

Warning:
已实施 IFrame 的同源策略，这意味着不能将 Zabbix 放置在不同域名的框架中。

 不过，如果放置在框架中的页面与 Zabbix 前端位于同一域名下，则该页面将能够通过 JavaScript 访问 Zabbix 前端。例如，`http://secure-zabbix.com/cms/page.html` 这样的页面，如果被放置到 `http://secure-zabbix.com/zabbix/` 上的仪表板中，将拥有对 Zabbix 的完整 JS 访问权限。

服务器/proxy

如果标记为必需，则所需的软件/库是严格必要的。可选项用于支持某些特定功能。

Requirement	Mandatory status	Description
libpcre2	是	PCRE2 库是支持 Perl 兼容正则表达式（PCRE）所必需的。 支持 PCRE2 v10.x。
libevent		进程间通信所必需。版本 2.0.10 或更高。
libevent-pthreads		进程间通信所必需。
libpthread		互斥锁和读写锁支持所必需（可能是 libc 的一部分）。
libresolv		DNS 解析所必需（可能是 libc 的一部分）。
libiconv		文本编码/格式转换所必需（可能是 libc 的一部分）。Linux 上的 Zabbix 服务器必需。
libz		压缩支持所必需。
libm		数学库。仅 Zabbix 服务器需要。
libmysqlclient	二选一	使用 MySQL 时必需。
libmariadb		使用 MariaDB 时必需。
libpq5		使用 PostgreSQL 时必需；*libpq5* 版本必须与所使用的 PostgreSQL 数据库版本匹配或更高。
libsqlite3		使用 SQLite 时必需。仅 Zabbix proxy 需要。
libOpenIPMI	否	IPMI 支持所必需。仅 Zabbix 服务器需要。
libssh2 或 *libssh*		SSH 检查所必需。版本要求：libssh2 为 1.8.0 或更高；libssh 为 0.9.0 或更高。
libcurl		以下功能所必需： - Web 监控、VMware 监控和HTTP agent 监控项（全部要求版本 7.19.1 或更高）； - Zabbix agent 的web.page.* 监控项（版本 7.19.1 或更高；另请参阅agent 要求）； - SMTP 身份验证（Basic：版本 7.20.0 或更高；OAuth：版本 7.33 或更高；另请参阅前端 要求）； - Elasticsearch（版本 7.28.0 或更高）； - Telemetry 查询 监控项（版本 7.28.0 或更高）。 建议所有功能使用 7.28.0 或更高版本。 要在 `web.page.*` 监控项中使用升级后的 cURL 功能，请重启 Zabbix 服务器/proxy。 对于 SMTP 身份验证，请在运行时使用 `libcurl-full` 软件包。
libxml2		VMware 监控和 XML XPath 预处理中所必需。
net-snmp		SNMP 支持所必需。版本 5.3.0 或更高。 从 net-snmp 库 5.8 开始支持强加密协议（AES192/AES192C、AES256/AES256C）；在基于 RHEL 8 及更高版本的系统上，建议使用 net-snmp 5.8.15 或更高版本。
libunixodbc		数据库监控所必需。
libgnutls 或 *libopenssl*		使用加密时所必需。 最低版本：*libgnutls* - 3.1.18，*libopenssl* - 1.0.1
libldap		LDAP 支持所必需。
fping		ICMP ping 监控项所必需。
c-ares		如果 Zabbix 使用 `--with-ares` 选项进行配置，则异步 DNS 解析所必需。否则将使用 *libevent*。 最低版本：1.16.0。DNS 查询缓存和故障转移功能（默认启用查询缓存）要求 c-ares 1.26.0 或更高版本。在 c-ares 由操作系统软件包提供的系统上，请确保已安装的运行时库提供 DNS 缓存和故障转移所需的解析器功能。
g++		C++ 编译器。使用 `--with-apm` 选项编译 Zabbix proxy 以及实现应用性能监控（APM）时所必需。

Requirement	Mandatory status	Description
gRPC		如果使用 `--with-apm` 选项编译，则 Zabbix proxy 上的APM 所必需。 所需库：*grpc++*、*grpc*、*gpr*。 编译 Zabbix proxy 时还需要 `grpc_cpp_plugin` 工具。 以下操作系统的标准软件包仓库中提供所需版本： - Ubuntu 23.04 或更高版本； - Debian 12（Bookworm）或更高版本； - Alpine Linux 3.15 或更高版本； - RHEL、Rocky Linux、CentOS Stream 9 或更高版本（仅来自 EPEL 仓库）； - Fedora X 或更高版本。 在较旧的操作系统版本上，必须从源代码构建 *gRPC*。
protobuf		如果使用 `--with-apm` 选项编译，则 Zabbix proxy 上的APM 所必需。 编译 Zabbix proxy 时还需要 `protoc` 编译器。
Abseil		如果使用 `--with-apm` 选项编译，则 Zabbix proxy 上的APM 所必需。 所需库：*absl_synchronization*、*absl_base*。 通常与 *gRPC* 一起安装。

Agent

Requirement	Mandatory status	Description
libpcre2	Yes	需要 PCRE2 库以支持 Perl Compatible Regular Expression (PCRE)。 支持 PCRE2 v10.x。 日志监控所必需。在 Windows 上也必需。
libpthread		需要用于支持互斥锁和读写锁（可能是 libc 的一部分）。在 Windows 上不需要。
libresolv		需要用于 DNS 解析（可能是 libc 的一部分）。在 Windows 上不需要。
libiconv		需要用于将日志监控项、文件内容、文件正则和 regmatch 监控项中的文本编码/格式转换为 UTF-8（可能是 libc 的一部分）。在 Windows 上不需要。
libgnutls or *libopenssl*	No	如果使用加密，则需要。 最低版本：*libgnutls* - 3.1.18，*libopenssl* - 1.0.1 在 Microsoft Windows 上需要 OpenSSL 1.1.1 或更高版本。
libldap		如果使用 LDAP，则需要。在 Windows 上不支持。
libcurl		需要用于扩展支持 Zabbix agent web.page.* 监控项。 没有 libcurl 时，可使用基本功能（例如，web.page.get[http://example.com]）。有了 libcurl，agent 还支持其他功能，例如带凭据的 HTTP URL（例如，http://user:password@example.com）以及 HTTPS URL。 需要 7.19.1 或更高版本（建议 7.28.0 或更高版本）。 要使用升级后的 cURL 功能，请重启 Zabbix agent。
libmodbus		仅在使用 Modbus 监控时需要。 版本 3.0 或更高。

Agent 2

Requirement	Mandatory status	Description
Go	Yes	从源代码构建 Zabbix agent 2 及其插件所必需。 支持 Go 1.25 或更高版本。安装说明请参见 go.dev。 Zabbix agent 2 及其插件使用的 Go 库列在 Zabbix Git 仓库中（仓库中标记为 `indirect` 的库是其他必需库的依赖项）： - Zabbix agent 2 - Plugin support - PostgreSQL - MongoDB - MSSQL - Ember+ - NVIDIA GPU - Example plugin

Requirement	Mandatory status	Description
libpcre2	Yes	需要 PCRE2 库以支持 Perl Compatible Regular Expression（PCRE）。 支持 PCRE2 v10.x。 日志监控需要此库。在 Windows 上也需要。
libopenssl	No	使用加密功能时需要。 UNIX 平台需要 OpenSSL 1.0.1 或更高版本。 OpenSSL 库必须启用 PSK 支持。不支持 LibreSSL。 在 Microsoft Windows 系统上需要 OpenSSL 1.1.1 或更高版本。

Web 服务

支持使用最新稳定版本的 Google Chrome 通过 Zabbix Web 服务生成计划报表。

构建 Web 服务所需的 Go 版本与Zabbix agent 2 使用的版本一致。

Java 网关

如果您从源代码仓库或归档文件获取了 Zabbix，则必要的依赖项已经包含在源代码树中。

如果您从发行版的软件包获取了 Zabbix，则必要的依赖项已经由软件包系统提供。

在上述两种情况下，软件都已准备就绪，可以直接使用，无需额外下载。

但是，如果您希望提供这些依赖项的自有版本（例如，您正在为某个 Linux 发行版准备软件包），下面列出了 Java 网关已知可正常使用的库版本。Zabbix 也可能可以与这些库的其他版本配合使用。

下表列出了原始代码中当前随 Java 网关捆绑的 JAR 文件：

库	是否必须	捆绑版本	注释
android-json	是	4.3r1	JSON（JavaScript 对象表示法）是一种轻量级的数据交换格式。这是从 Android SDK 中提取的、与 `org.json` 兼容的 Android 实现。
logback-classic		1.5.25	
logback-core		1.5.25	
slf4j-api		2.0.16	

Java 网关可以使用 Oracle Java 或开源的 OpenJDK（版本 8 或更高版本）构建。Zabbix 提供的软件包使用 OpenJDK 编译。下表列出了各发行版用于构建 Zabbix 软件包的 OpenJDK 软件包：

发行版	OpenJDK 软件包
AlmaLinux 10	`java-21-openjdk-devel`（amd64、arm64：21.0.12.0.8-1.2）
AlmaLinux 9	`java-17-openjdk-devel`（amd64、arm64：17.0.15.0.6-2）
AlmaLinux 8	`java-1.8.0-openjdk-devel`（amd64：1.8.0.332.b09-2；arm64：1.8.0.382.b05-2）
Amazon Linux 2023	`java-22-amazon-corretto-devel`（amd64、arm64：22.0.2+9-1）
CentOS Stream 10	`java-21-openjdk-devel`（amd64、arm64：21.0.11.0.10-2）
CentOS Stream 9	`java-17-openjdk-devel`（amd64、arm64：17.0.15.0.6-2）
CentOS Stream 8	`java-1.8.0-openjdk-devel`（amd64、arm64：1.8.0.362.b08-3）
CentOS 7	`java-1.8.0-openjdk-devel`（amd64：1.8.0.282.b08-1）
Debian 13	`default-jdk-headless`（amd64、arm64：2:1.21-76）
Debian 12	`default-jdk-headless`（amd64、arm64：2:1.17-74）
Debian 11	`default-jdk-headless`（amd64：2:1.11-72）
OpenSUSE Leap 16	`java-21-openjdk-devel`（amd64、arm64：21.0.11.0-160000.2.1）
OpenSUSE Leap 15	`java-17-openjdk-devel`（amd64：17.0.5.0-150400.3.9.3；arm64：17.0.8.0-150400.3.27.1）
Oracle Linux 10	`java-21-openjdk-devel`（amd64、arm64：21.0.11.0.10-2.0.1）
Oracle Linux 9	`java-17-openjdk-devel`（amd64、arm64：17.0.15.0.6-2.0.1）
Oracle Linux 8	`java-1.8.0-openjdk-devel`（amd64：1.8.0.372.b07-4.0.1）； `java-11-openjdk-devel`（arm64：11.0.20.0.8-3.0.1）
Oracle Linux 7	`java-1.8.0-openjdk-devel`（amd64：1.8.0.282.b08-1）
Raspberry Pi OS 13	`default-jdk-headless`（arm64、armhf：2:1.21-76）
Raspberry Pi OS 12	`default-jdk-headless`（arm64、armhf：2:1.17-74）
Raspberry Pi OS 11	`default-jdk-headless`（arm64：2:1.11-72；armhf：2:1.11-72+b4）
RHEL 10	`java-21-openjdk-devel`（amd64、arm64：21.0.12.0.8-1.2）
RHEL 9	`java-17-openjdk-devel`（amd64、arm64：17.0.15.0.6-2）

发行版	OpenJDK 软件包
RHEL 8	`java-1.8.0-openjdk-devel`（amd64：1.8.0.372.b07-4；arm64：1.8.0.382.b05-2）
RHEL 7	`java-1.8.0-openjdk-devel`（amd64：1.8.0.282.b08-1）
Rocky Linux 10	`java-21-openjdk-devel`（amd64、arm64：21.0.12.0.8-1.2）
Rocky Linux 9	`java-17-openjdk-devel`（amd64、arm64：17.0.15.0.6-2）
Rocky Linux 8	`java-1.8.0-openjdk-devel`（amd64、arm64：1.8.0.372.b07-4；arm64：1.8.0.382.b05-2）
SLES 16	`java-21-openjdk-devel`（amd64、arm64：21.0.11.0-160000.1.1）
SLES 15	`java-17-openjdk-devel`（amd64：17.0.5.0-150400.3.9.3；arm64：17.0.8.0-150400.3.27.1）
Ubuntu 26.04	`default-jdk-headless`（amd64、arm64：2:1.25-77）
Ubuntu 24.04	`default-jdk-headless`（amd64、arm64：2:1.21-75+exp1）
Ubuntu 22.04	`default-jdk-headless`（amd64、arm64：2:1.11-72build2）
Ubuntu 20.04	`default-jdk-headless`（amd64、arm64：2:1.11-72）

Bridge adapter

Zabbix bridge adapter 没有特殊要求。

默认端口

以下列表包含 Zabbix 组件在默认配置下可能使用的端口：

Zabbix 组件	协议和端口	发起连接的组件
Zabbix agent	TCP 10050	Zabbix 服务器或 Zabbix proxy（仅被动检查）
Zabbix agent 2	TCP 10050	Zabbix 服务器或 Zabbix proxy（仅被动检查）
Zabbix 服务器	TCP 10051	主动 Zabbix agent、主动 Zabbix proxy、Zabbix sender、Zabbix 前端
Zabbix proxy	TCP 10051	主动 Zabbix agent、Zabbix sender、Zabbix 服务器（仅被动 proxy）
Zabbix Java 网关	TCP 10052	Zabbix 服务器或 proxy
Zabbix web service	TCP 10053	Zabbix 服务器
Zabbix MCP 服务器	TCP 8443	MCP 客户端（AI agent）
Zabbix 前端	TCP 80 TCP 443	Web 浏览器、Zabbix web service（计划报表）、API 客户端
SNMP traps (snmptrapd)	UDP 162	SNMP 设备
Zabbix bridge adapter	TCP 10054	Zabbix 服务器

Warning:
仅开放实际运行的组件所需的端口，并将每条防火墙规则限制为需要连接的特定主机（例如，仅允许来自 Zabbix 服务器或 proxy 的端口 10050 连接，而不是允许来自任意地址的连接）。不要将 Zabbix 端口直接暴露到互联网。
 在开放端口之前，请强化安装：启用组件之间的加密，在组件配置文件中限制允许的主机，并查看安全最佳实践。

数据库大小

Zabbix 配置数据需要固定的磁盘空间，而且不会增长太多。

Zabbix 数据库大小主要取决于以下变量，这些变量定义了已存储历史数据的数量：

- 每秒处理的值数量

这是 Zabbix 服务器每秒接收的新值的平均数量。例如，如果有 3000 个用于监控的监控项，刷新间隔为 60 秒，则每秒值数量计算为 3000/60 = **50**。

这意味着每秒会向 Zabbix 数据库中新增 50 个值。

- 用于历史数据的 housekeeper 设置

Zabbix 会保留固定时间段内的值，通常为数周或数月。每个新值都需要一定的磁盘空间来存储数据和索引。

因此，如果希望保留 30 天的历史数据，并且每秒接收 50 个值，那么总值数量大约为 (**30***24*3600)* **50** = 129.600.000，约为 130M 个值。

根据所使用的数据库引擎、接收值的类型（浮点数、整数、字符串、日志文件等），保存单个值所需的磁盘空间可能从 40 字节到数百字节不等。通常，对于数值型监控项，每个值大约占用 90 字节[2]。在本例中，这意味着 130M 个值将需要 130M * 90 字节 = **10.9GB** 的磁盘空间。

Note:
文本/日志监控项值的大小无法精确预测，但通常可以按每个值约 500 字节估算。

- 用于趋势数据的 housekeeper 设置

Zabbix 会在 **trends** 表中为每个监控项保留 1 小时的 max/min/avg/count 值集合。这些数据用于趋势分析和长期图表。这一小时周期无法自定义。

Zabbix 数据库根据数据库类型，每个总计大约需要 90 字节。假设我们希望保留 5 年的趋势数据。3000 个监控项的值每年将需要 3000*24*365* **90** = **2.2GB**，5 年则为 **11GB**。

- 用于事件的 housekeeper 设置

每个 Zabbix 事件大约需要 250 字节的磁盘空间[1]。很难估算 Zabbix 每天生成的事件数量。在最坏情况下，可以假设 Zabbix 每秒生成一个事件。

对于每个已恢复的事件，都会创建一条 event_recovery 记录。通常大多数事件都会恢复，因此可以假设每个事件对应一条 event_recovery 记录。这意味着每个事件还会额外占用 80 字节。

此外，事件还可以包含标签，每条标签记录大约需要 100 字节的磁盘空间[1]。每个事件的标签数量（#tags）取决于配置。因此，每个事件还需要额外的 #tags * 100 字节磁盘空间。

这意味着，如果我们希望保留 3 年的事件数据，则需要 3*365*24*3600* (250+80+#tags*100) = **~30GB**+#tags*100B 的磁盘空间[2]。

Note:
[1] 当事件名称、标签和值包含非 ASCII 字符时会更多。
 [2] 这些大小估算基于 MySQL，其他数据库可能不同。

下表包含可用于计算 Zabbix 系统所需磁盘空间的公式：

参数	所需磁盘空间公式（字节）
Zabbix 配置	固定大小。通常为 10MB 或更少。
历史数据	days*(items/refresh rate)*24*3600*bytes items：监控项数量 days：保留历史数据的天数 refresh rate：监控项的平均刷新间隔 bytes：保存单个值所需的字节数，取决于数据库引擎，通常约为 90 字节。
趋势数据	days*(items/3600)*24*3600*bytes items：监控项数量 days：保留历史数据的天数 bytes：保存单个趋势值所需的字节数，取决于数据库引擎，通常约为 90 字节。
事件	days*events*24*3600*bytes events：每秒事件数。在最坏情况下为每秒一个（1）事件。 days：保留历史数据的天数 bytes：保存单个事件所需的字节数，取决于数据库引擎，通常约为 ~330 + 每个事件的平均标签数 * 100 字节。

因此，总所需磁盘空间可按以下公式计算：

配置 + 历史数据 + 趋势数据 + 事件

安装 Zabbix 后，磁盘空间不会立即被占满。数据库大小会逐渐增长，然后在某个时点停止增长，这取决于 housekeeper 设置。

时间同步

确保运行 Zabbix 的服务器具有精确的系统时间非常重要。
ntpd 是最常用的守护进程，可将主机时间与其他机器的时间同步。
强烈建议在所有运行 Zabbix 组件的系统上保持系统时间同步。

新增功能和升级说明：**8.0**

Zabbix 8.0.0 基于 Zabbix 7.4.0 继续发展，新增了多项功能和增强功能。

本页面还包括：

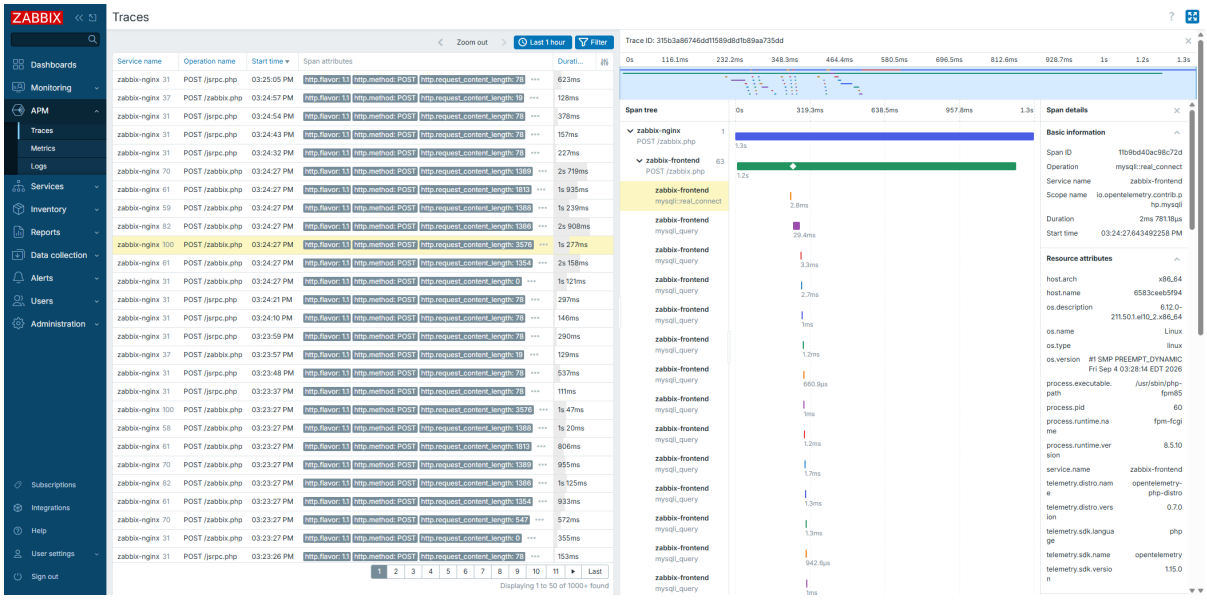
- 模板中的新增功能，包括新模板以及对现有模板的更改。
- 升级说明，包括不兼容的更改。

另请参阅：API 更改。

应用性能监控　Zabbix 现已支持基于 OpenTelemetry 的应用性能监控（APM）。OpenTelemetry 是用于从应用程序和服务收集遥测数据的开放标准。这些数据可帮助您了解请求如何流经应用程序和服务、衡量其性能，并找出操作缓慢或失败的原因。

借助 APM，您可以：

- 使用 Zabbix proxy 收集遥测数据。现在，Zabbix proxy 可以通过 OpenTelemetry 协议（OTLP/gRPC）接收来自应用程序或 OpenTelemetry 收集器的数据，并将其写入数据源；请参阅proxy 配置中的新 *APM* 选项卡。
- 在新的 管理 > 数据源部分指定遥测数据源（ClickHouse），并在新的*APM*菜单中查看数据。
- 使用新的遥测查询监控项类型，在触发器和图形中使用遥测数据。



此外，还新增了以下参数和监控项：

- Zabbix 服务器参数：TelemetryProvider和StartTelemetryQueryPollers。
- Zabbix proxy 参数：TelemetryProvider、StartTelemetryQueryPollers、StartAPMCollectors、APMListenIP、APMListenPort、APMTLSCAFile、APMTLSCertFile和APMTLSKeyFile。
- Zabbix 前端参数：$TELEMETRY_PROVIDERS[]。
- Zabbix proxy 编译选项：--with-apm；官方 Zabbix proxy 软件包使用此选项构建。
- 用于监控遥测数据收集的内部监控项：zabbix[apm,<type>]。

ClickHouse 支持　现在，您可以使用 ClickHouse 作为 Zabbix 后端数据库来存储监控项值历史记录，也可以将其作为应用性能监控的遥测数据源。

有关历史记录存储设置的详细信息和要求，请参阅ClickHouse 设置。如果您希望将现有 Zabbix 数据库（MySQL 或 PostgreSQL）中的历史数据迁移到 ClickHouse，请参阅 ClickHouse 架构和历史记录迁移脚本。

此功能还改变了在 Zabbix 配置文件中设置历史记录存储提供程序的方式。此更改同时影响 ClickHouse 和 Elasticsearch 设置。有关详细信息，请参阅升级说明。

复杂事件处理　复杂事件处理（CEP）对 Zabbix 事件执行基于规则的处理。

例如，您可能希望：

- 将相关事件分组，并将一系列事件中的第一个事件标记为原因（后续事件标记为症状）
- 为事件指定生存时间
- 为特定事件指定最大数量
- 按位置接收相似事件的通知

- 如果某个事件在一段时间内发生过于频繁（或完全未发生），则接收通知

复杂事件处理引入了用于事件分析的滑动时间窗口。
例如，在 10 分钟的窗口中，落入该窗口的所有事件构成一个单独的上下文。
随着窗口向前移动，较早的事件会离开窗口（被“驱逐”），而新事件可能会进入窗口。

时间窗口的概念为执行操作提供了逻辑节点。
例如，事件进入窗口时，可以对其进行转换或“丰富”：提高或降低严重性、添加或调整标签、启用抑制等。

当事件离开窗口或发现某种模式时，也可以执行类似操作。

事件流分析由 CEP 规则定义。
CEP 规则通常由时间窗口、条件和操作组成。
操作与执行节点相关联。

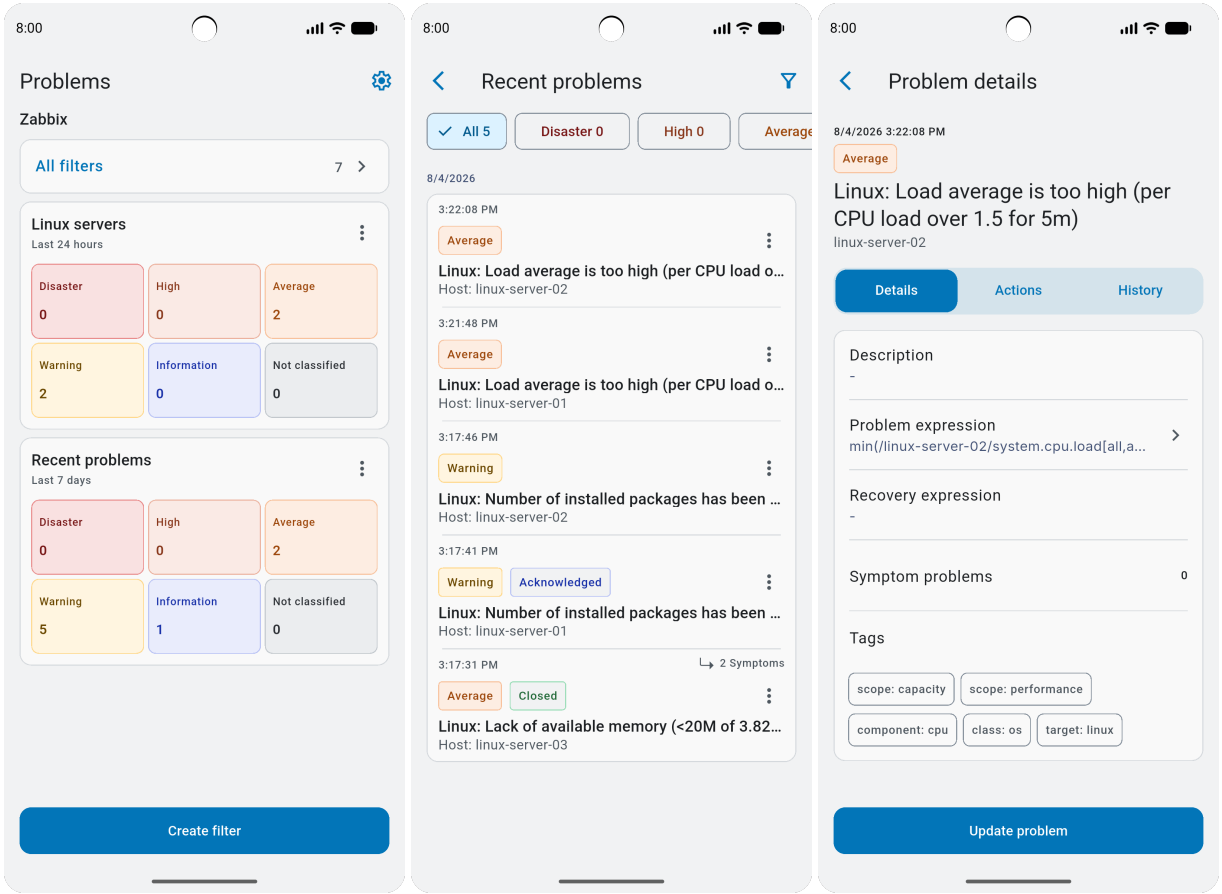
更详细的文档将在稳定版 8.0 发布后提供。

移动应用　Zabbix 移动应用（即将推出）可让您通过移动设备监控 Zabbix 环境。查看问题、检查问题详情、更新问题，并在需要处理某项内容时接收推送通知。

使用该应用后，Zabbix 8.0 新增了以下功能：

- 用于管理移动设备的前端部分：用户 > 设备 和 用户设置 > 设备。
- 用于将问题通知从 Zabbix 服务器发送到移动应用的媒介类型：推送通知。
- 允许 Zabbix 服务器与移动应用通信的组件：Bridge adapter。
- Zabbix 服务器配置参数：BridgeAdapterURL、EnableMobileDevices。

有关如何设置 Zabbix 以配合移动应用使用的详细信息，请参阅移动应用部分。



MCP 服务器　新的Zabbix MCP 服务器 通过 Model Context Protocol（MCP）将 AI agent 连接到 Zabbix。现在，您可以使用自然语言向 AI 助手询问监控相关信息，例如：

- 数据库服务器上有哪些问题尚未确认？
- 为什么 *web-01* 从今天早上开始一直报告问题？
- 今晚 *22:00* 至 *02:00* 将 *Linux servers* 组中的所有主机置于维护状态。

How busy is the host Zabbix server?

Bash(date +%s)
└ 1789042843

Called zabbix 2 times (ctrl+o to expand)

The host is idle — and the agent is reporting again; OS metrics started arriving in the last few minutes.

OS-level (host 10084, "Zabbix server")

Metric	Item	Latest value	Timestamp
CPU utilization	42269	0.73 %	1789042824
Memory utilization	42270	29.67 %	1789042804
Load average (1m)	42249	0.17	1789042809

Zabbix server internal workload

Metric	Item	Latest values
Processed values per second	23277	1.13 → 1.44 → 4.82 vps (rising as the agent's items resume)
History syncer busy	23257	0.27–0.55 %
Poller busy	23264	0 %
Queue / queue > 10 min	23272 / 23271	0 / 0

CPU under 1 %, memory under a third, load well below 1, nothing queued. The host has essentially no load.

The agent data only appeared between my previous check and this one — a minute earlier the same items returned nothing over the whole 6-day window. Worth confirming problem 19 clears now that the agent is back; I can re-check problem_get if you want.

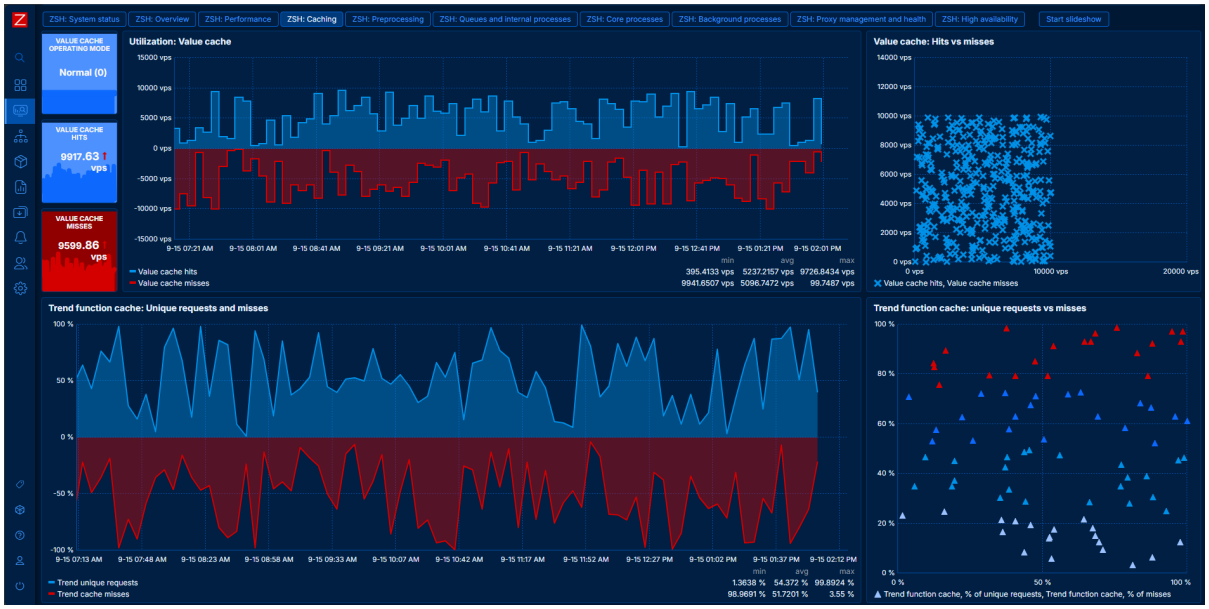
agent 使用 Zabbix API token 进行连接，因此只能访问 token 所有者有权访问的内容，并且所有更改都会以该用户的身份记录在审计日志中。有关可用操作的列表，请参阅工具和资源。

按需为触发器创建维护期　现在可以为触发器或事件名称定义维护期。

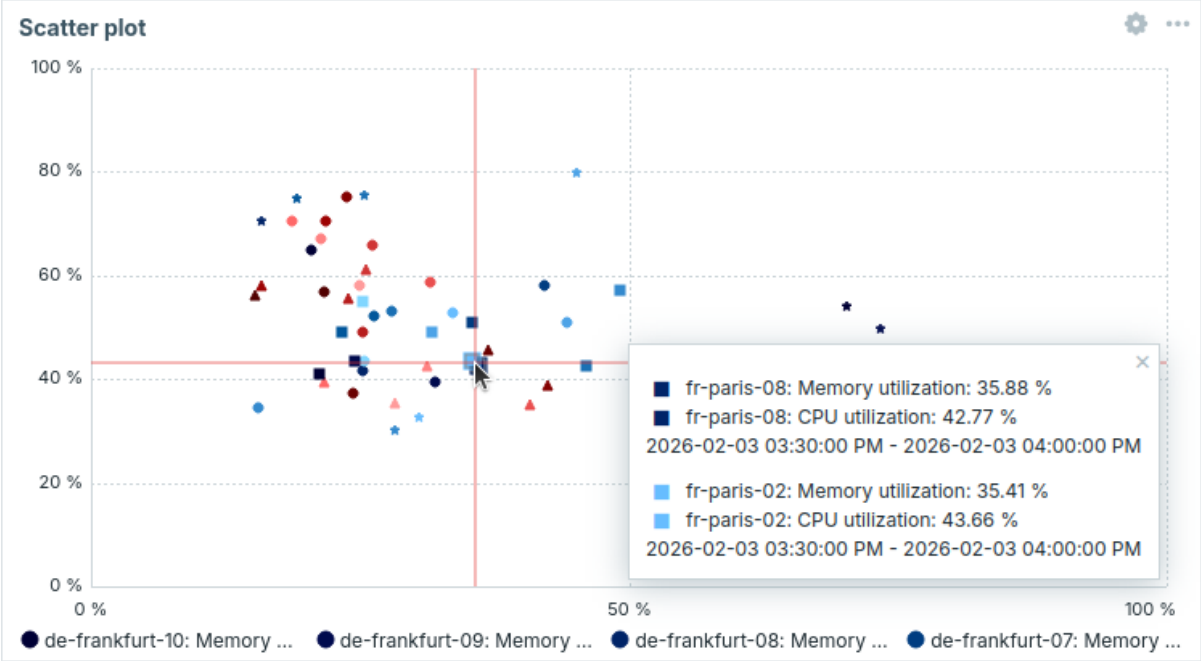
也可以通过事件菜单按需创建维护期。可以针对主机、触发器、事件名称或标签启动按需维护。

为触发器创建维护期的另一种方式是使用问题更新表单。

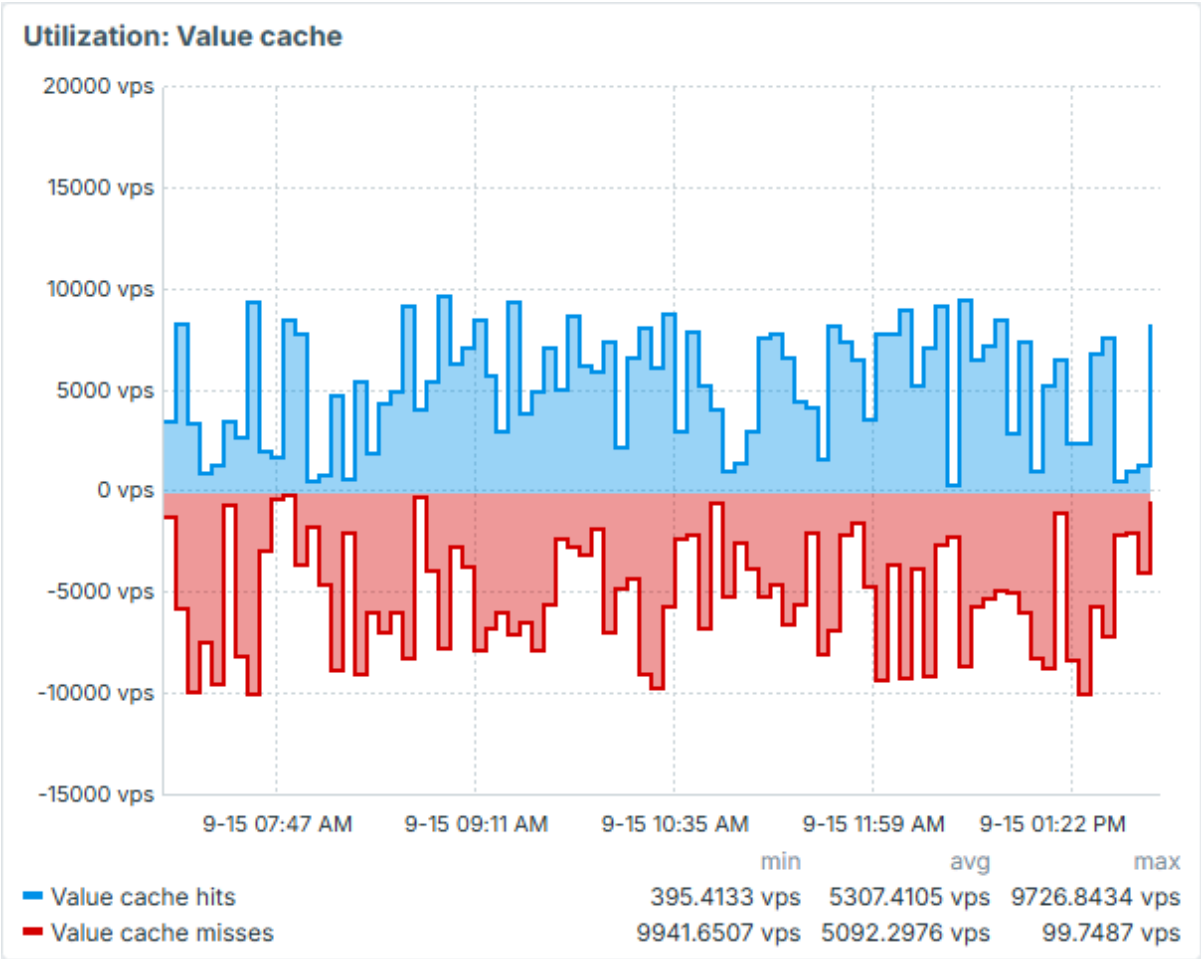
深蓝色主题　Zabbix 前端现提供新的深蓝色主题。您可以在用户配置文件中选择该主题，也可以将其设置为默认主题：



散点图部件　Scatter plot 部件已添加到仪表板部件中。它通过沿 X 轴和 Y 轴绘制单个数据点来显示两个指标之间的关系。这有助于揭示数据集中的模式、聚类、相关性和离群值。



图表小部件的 **Y** 轴反转　图表小部件中的图表现在可以以反转的 Y 轴值显示。
新的 *Invert values* 设置允许你将 Y 轴值乘以 −1，而不会更改原始数据。



另请参见：图表小部件增强功能。

JSON 数据类型　Zabbix 现在支持将 JSON 作为监控项值的数据类型。

此前，JSON 值通过文本监控项采集，并以字符串形式存储，限制为 64KB。
现在，Zabbix 可以原生存储 JSON 值，限制为 128MiB，同时还会拒绝无效的 JSON 值（例如，包含未加引号的键、尾随逗号、括号不匹配等）。

除 *Calculated* 外，所有监控项和监控项原型类型都支持 JSON 数据类型，并且可用于实时数据导出 和connectors。

JSON 值可以存储在所有受支持的数据库 和Elasticsearch 中。
如果你使用的是 TimescaleDB，请参阅升级说明。

请注意，使用 JSON 数据类型的监控项不能用于触发器；不过，你可以通过具有非 JSON 数据类型的依赖监控项 提取 JSON 字段，并将其用于触发器。

返回 JSON 字符串（`net.if.discovery`、`vfs.file.get` 等）的监控项仍然是文本监控项；不过，如有需要，你可以将它们改为 JSON。

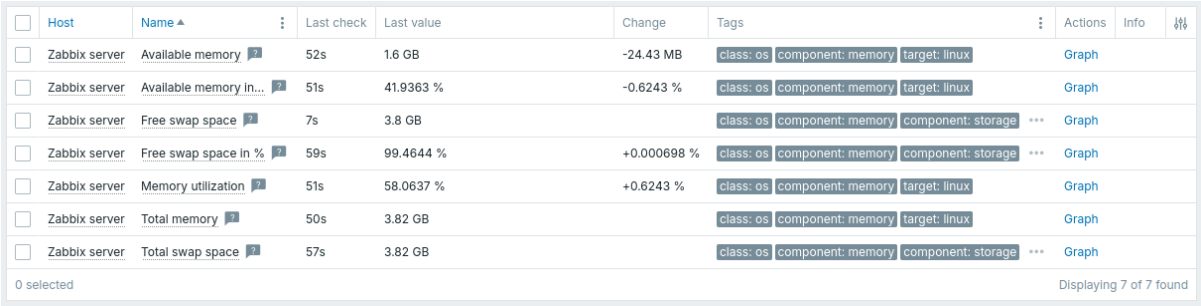
有关更多详细信息和 JSON 数据限制，请参阅监控项配置。

表格的自定义布局　现在，在多个前端表格中，您可以隐藏/显示、重新排序以及调整列宽。此外，还可以添加一个 自定义文本列，其中包含自定义文本、宏值，或两者兼有。

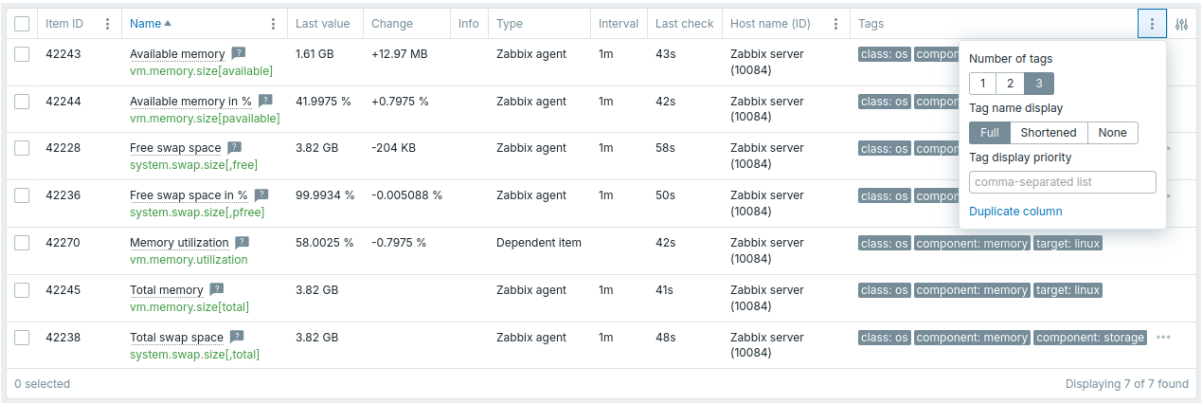
自定义布局可在以下位置使用：

- 监控 > 主机、最新数据 和 问题
- 数据采集 > 模板 和 主机

例如，监控 > 最新数据中的默认表格布局：



自定义布局示例：



某些列还带有 选项按钮，可为该列显示数据的方式提供更多控制。因此，一些筛选器参数已从筛选器选项卡移至列选项中。如果您正在使用收藏筛选器选项卡，请参阅升级说明。

全局正则表达式　全局正则表达式新增了多项易用性改进：

- 全局正则表达式配置表单现在会在模态窗口中打开；
- 表达式类型已重命名。
- 表单现在包含 描述字段。
- 测试不再位于单独的选项卡中。
- 现在可以按名称和描述筛选全局正则表达式列表。
- 现在可以导出和导入全局正则表达式。

模板

新增了模板，并且许多模板已更新。

现在，连接到 HTTPS 端点的模板默认会验证端点的 TLS 证书。这会影响所有云和 API 监控模板，例如 *Claude API by HTTP*、*Cloudflare by HTTP*、*Kubernetes Cluster by HTTP* 等。在升级任何基于 HTTP 的模板之前，请参阅升级说明。

升级 Zabbix 时，现有模板不会自动更新，以避免覆盖自定义修改。如需升级模板或添加新模板，请参阅模板升级。

Note:
请注意，自 Zabbix 6.0 起，所有模板均采用更新后的格式，这可能会影响 6.0 之前版本模板的导入。有关更多信息，请参阅 6.0 中的模板变更。

新模板 **AI** 使用情况监控

- Claude API by HTTP，用于监控 Claude API 使用情况和成本的模板。
- OpenAI Platform by HTTP，用于监控 OpenAI 开发者平台的模板。

云

- AWS by HTTP 模板集新增了模板 *AWS Backup Vault by HTTP*。
- Azure by HTTP 模板集新增了模板 *Azure Sentinel by HTTP* 和 *Azure Container Apps by HTTP*。
- GCP by HTTP 模板集新增了模板 *GCP Cloud Run Service by HTTP*、*GCP Cloud Storage Bucket by HTTP*、*GCP Application Load Balancer by HTTP* 和 *GCP Cost Monitoring by HTTP*。
- Oracle Cloud by HTTP 模板集新增了模板 *Oracle Cloud Load Balancer by HTTP* 和 *Oracle Cloud Costs by HTTP*。

虚拟化和容器

- Kubernetes Cluster by HTTP，用于监控 Kubernetes 集群的模板，可通过专用主机自动发现和监控节点；该模板取代了之前的 Kubernetes 模板。
- Microsoft Hyper-V Failover Cluster by SSH 和 Microsoft Hyper-V Standalone by SSH，用于通过 SSH 监控 Microsoft Hyper-V 集群和独立主机的模板。
- Podman by Zabbix agent、by Zabbix agent active 和 by HTTP：用于通过 Podman API 监控 Podman 容器和服务运行状况的模板。请注意，Podman API 不支持授权方法。

数据库

- MariaDB by ODBC，用于通过 ODBC 监控 MariaDB 数据库的模板。
- MariaDB by Zabbix agent 和 MariaDB by Zabbix agent active，用于通过 Zabbix agent 和 Zabbix agent active 监控 MariaDB 数据库的模板。
- Percona by ODBC，用于通过 ODBC 监控 Percona 数据库的模板。
- Percona by Zabbix agent 和 Percona by Zabbix agent active，用于通过 Zabbix agent 和 Zabbix agent active 监控 Percona 数据库的模板。

网络、安全和电信

- Aruba CX 8300s by SNMP，为 Aruba CX 8300 交换机系列提供基于 SNMP 的监控的模板。
- Ciena 3906 by SNMP，用于监控 Ciena 3906 设备（包括文件系统和 CPU 负载）的模板。
- Cisco Secure Firewall Threat Defense by HTTP，使用 REST API 为 Cisco Secure Firewall Threat Defense 设备提供监控功能的模板。
- Cradlepoint NCM v2 by HTTP，用于通过 HTTP 监控 Cradlepoint NCM v2 及其设备的模板集。
- Huawei AR600 by SNMP，用于监控 Huawei AR600 路由器（包括 CoS 和 NQA 指标）的模板。
- Ribbon SBC Edge by HTTP，用于通过 HTTP 监控 Ribbon SBC Edge（原名 SWe Lite）设备的模板。
- Ribbon SBC SWe Core by HTTP，用于通过 HTTP 监控 SBC SWe Core 设备的模板，以及 Ribbon SBC SWe CE by HTTP，用于通过 HTTP 监控 Ribbon SBC SWe Call Engine (CE) 实例的模板。
- Stormshield SNS by SNMP，用于通过 SNMP 监控 Stormshield Network Security (SNS) 设备的模板。
- VeloCloud SD-WAN Edge by HTTP，用于通过 HTTP 监控 VeloCloud SD-WAN Edge 设备的模板。
- Vyatta Virtual Router by SNMP，用于监控 Vyatta 1908e 虚拟路由器的模板。

应用和服务

- Domain RDAP by HTTP，用于通过 RDAP 协议监控域名注册数据的模板。
- Envoy Proxy by OTLP，用于通过 *Telemetry query* 监控项收集 Envoy Proxy 的 OpenTelemetry 跟踪和日志的模板。
- Generic OpenTelemetry by OTLP，用于通过 *Telemetry query* 监控项收集 OpenTelemetry 跟踪、指标和日志的模板。
- GLPI by HTTP，使用带有 OAuth2 身份验证的 REST API v2，通过 HTTP 监控 GLPI 的模板。
- OpenRouter traces by OTLP，用于收集 OpenRouter 跟踪数据的模板。
- Proxmox VE by OTLP，使用其 OpenTelemetry 指标流，通过 *Telemetry query* 监控项监控 Proxmox VE 节点的模板。

已更新的模板

- 改进了 SNMP 模板中的设备运行时间触发器，以处理 SNMP 运行时间计数器重置并防止产生错误的重启告警。

操作系统

- 在 AIX、FreeBSD、HP-UX、Linux、macOS、OpenBSD、Solaris 和 Windows 模板中，文件系统监控现在使用新的 `vfs.fs.get` 参数（请参阅轻量级文件系统发现）。有关详细信息，请参阅模板变更。
- Linux by Zabbix agent 和 Linux by Zabbix agent active 已更新，改进了仪表板，并增强了对网络接口和块设备的监控。

- Linux by Prom 和 Linux by SNMP 已更新，改进了仪表板。
- Windows by Zabbix agent 和 Windows by Zabbix agent active 模板新增了指标、发现规则、原型、用户宏和仪表板小组件，并支持 Windows 计划任务、Windows 防火墙配置文件以及可加密的 BitLocker 卷。

AI 使用监控

- GitHub organization by HTTP 已更新，新增了用于监控 Microsoft Copilot 的监控项。
- Microsoft 365 reports by HTTP 已更新，新增了用于监控 Microsoft Copilot 的监控项。

云

- AWS by HTTP 模板集中的 *AWS Cost Explorer by HTTP* 和 Azure by HTTP 模板集中的 *Azure Cost Management by HTTP* 已更新，改进了仪表板。

虚拟化

- Proxmox VE by HTTP 已更新，支持嵌套 LLD 功能。此外，为了提高清晰度，显示百分比的监控项的单位格式已进行调整。
- 在 VMware 和 VMware FQDN 模板集中，VMware 告警发现的监控项和触发器原型名称已更新，加入了告警实体名称。

数据库

- MySQL by Zabbix agent、MySQL by Zabbix agent 2、MySQL by Zabbix agent active、MySQL by Zabbix agent 2 active 和 MySQL by ODBC 已更新，同时支持 `SHOW SLAVE STATUS`（旧语法）和 `SHOW REPLICA STATUS`（新语法）。此外，MySQL by Zabbix agent、MySQL by Zabbix agent active 和 MySQL by ODBC 已新增指标、改进仪表板，并扩展了表的发现规则（*MySQL by ODBC* 还包括副本）。
- PostgreSQL by ODBC 已更新，新增了指标、发现规则和仪表板，并全面改进了性能和可观测性。
- PostgreSQL by Zabbix agent 和 PostgreSQL by Zabbix agent active 已更新，新增了指标、发现规则、监控项和触发器原型，并增强了监控功能。
- Redis by Zabbix agent 2 已更新，支持通过用户和密码宏进行身份验证，并改进了监控项描述。

网络、安全和电信

- VeloCloud SD-WAN by HTTP 原名为 *VMWare SD-WAN VeloCloud by HTTP*，现已重命名并更新，以与最新的 VeloCloud SD-WAN 平台保持一致。

存储和备份

- Ceph by Zabbix agent 2 现在支持原生连接模式。有关详细信息，请参阅Ceph plugin。
- Veeam Backup and Replication by HTTP 和 Veeam Backup Enterprise Manager by HTTP 已更新，新增了监控功能，并支持 Veeam Backup & Replication v13.1.1.18，包括 REST API 修订版 1.3-rev2 以及更新后的许可证到期监控。

操作系统模板中的文件系统监控　在 AIX、FreeBSD、HP-UX、Linux、macOS、OpenBSD、Solaris 和 Windows 模板中（包括 Linux 和 Windows 模板的主动检查变体），文件系统监控现在使用新的 `vfs.fs.get` 参数：

- 获取文件系统监控项（`vfs.fs.get`）已移除；
- 已挂载文件系统发现规则现在是一个键为 `vfs.fs.get[short]` 的 Zabbix agent 监控项（更新间隔：1h），不再是依赖型发现规则；
- *FS [{#FSNAME}]*：获取数据监控项原型（Windows 模板中为 *FS [{#FSLABEL}({#FSNAME})]*：获取数据）现在是一个键为 `vfs.fs.get[full,{#FSNAME}]` 的 Zabbix agent 监控项，仅收集一个文件系统的统计信息。其他文件系统监控项原型将其用作主监控项。

因此，每个文件系统都会通过单独的 agent 检查来收集统计信息，新挂载的文件系统将在发现更新间隔内被发现（默认情况下为 1h）。

这些模板中的文件系统监控要求使用 Zabbix agent 8.0 或更高版本。

媒体类型

已添加以下媒体类型：

- IBM Maximo 服务请求 webhook 媒体类型支持将 Zabbix 与 IBM Maximo 关联。

以下媒体类型已更新：

- GLPI webhook 媒体类型现已支持使用 OAuth2 授权的 RESTful API v2（默认启用）、通过配置标志可选支持旧版 REST API v1、为问题事件创建工单，以及由用户配置紧急程度映射。
- MS Teams Workflow webhook 媒体类型现已成为 Microsoft Teams 的推荐集成方式。
  基于 Microsoft 已于 2026 年 5 月 22 日停用的 Office 365 Connector（Incoming Webhook）功能的 MS Teams webhook 媒体类型不再可用。
  现有的 MS Teams webhook 配置需要迁移到 MS Teams Workflow webhook 媒体类型。

此外，连接到 HTTPS 端点的媒体类型现在默认会验证端点的 TLS 证书。
这会影响所有 webhook 媒体类型，例如 *Slack*、*Jira*、*Telegram* 等。
在升级任何 webhook 媒体类型之前，请参阅升级说明。

监控项

trapper 监控项允许的主机默认值　对于Zabbix trapper 监控项以及启用了 *Enable trapping* 的HTTP agent 监控项，*Allowed hosts* 字段现在默认使用用户宏 `{$TRAPPER.ALLOWED_HOSTS}`。如果该值为空，则会拒绝来自所有主机的传入连接。要允许提交数据，请指定一个或多个 IP 地址或 DNS 名称。

趋势函数缓存利用率监控　已向`zabbix[tcache,cache,<parameter>]` 内部监控项添加了 `paccessed` 参数。
它返回过去 24 小时内被访问的趋势函数缓存条目所占的百分比。

web.page.* 监控项中的重定向　Zabbix agent 和 Zabbix agent 2 的web.page.get、web.page.perf 和web.page.regexp 监控项现在可以跟随重定向。重定向的最大次数通过新的可选 `redirect_limit` 参数设置；默认情况下不会跟随重定向。

插件

Ceph 插件　Zabbix agent 2 Ceph 插件 现在以两种模式运行：

- **native** - 此模式使用 go-ceph 库，通过原生 Ceph API（msgr2 协议）直接与 Ceph 集群通信。
  这是现代 Ceph 部署的推荐模式，但仅支持 **Linux**，并且从 Ceph 16 开始支持。
- **restful**（已弃用）- 此模式使用 Ceph RESTful API 进行通信。
  这是为了向后兼容而设置的默认模式，但由于 `mgr/restful` 模块已被移除，因此在 Ceph 20（Tentacle）或更高版本中无法使用。

重要：当使用 Zabbix agent 2 的 Ceph 插件监控 Ceph 20（Tentacle）及更高版本时，请使用 native 模式，因为 restful 模式已不再受支持。

使用哪种模式由 mode 参数的值决定（native/restful）：

- `Plugins.Ceph.Default.Mode=native` - 为插件设置 native 模式
- `Plugins.Ceph.Sessions.<SessionName>.Mode=native` - 为指定会话设置 native 模式

请注意，不同模式使用的用户凭据集不同，彼此不兼容。
`Plugins.Ceph.InsecureSkipVerify` 参数在 native 模式下会被忽略，因为连接安全性由 Ceph 集群端通过 msgr2 协议定义（默认安全）。

另请注意，Zabbix agent 2 的 Ceph 插件现在是一个可加载插件，并且需要额外的安装步骤。
这是由于需要 librados 包（用于 native 模式）。
详情请参见 Ceph 插件 readme。

MongoDB 插件　Zabbix agent 2 MongoDB 插件 提供了多项增强功能：- 完全支持传统的 MongoDB URI 解析，同时支持 `mongodb://` 和 `mongodb+srv://` 两种方案。- 自动发现 MongoDB 副本集节点，增强对 MongoDB 集群的监控能力。- 支持 x509 身份验证，可使用客户端证书安全连接到 MongoDB。

Oracle 插件　Zabbix agent 2 Oracle 插件 现在支持使用 TCPS（TLS）协议与 Oracle 数据库建立加密连接。这增加了通过 TLS 保护的套接字监控 Oracle 实例的能力，从而提升远程监控的安全性。

Redis 插件 **— TLS** 支持和启动时验证　Zabbix agent 2 Redis 插件已添加 TLS 支持。

已实现插件 TLS 配置的启动时验证，并改进了验证/错误消息。无效的配置逻辑（例如：使用连接类型 `verify_full` 但未指定 `TLSCAFile`）可能会导致 Zabbix agent 2 无法启动。

可加载插件的测试执行模式　现在可以使用 `-t`（`--test`）标志并传递监控项键作为参数，以测试模式启动可加载插件。在此模式下，插件会为调试和开发目的而运行，并且会忽略插件配置文件。

发现和自动注册

proxy 组条件　发现和自动注册动作现已支持 proxy 组条件。

低级发现

发现表单中的 **“转换为 JSON”** 复选框　在底层发现规则表单和发现原型表单中新增了转换为 *JSON* 复选框。当在类型下拉列表中选择 “HTTP agent” 时，该复选框会显示。

此选项允许您将检索到的数据标记为在进一步处理前自动转换为 JSON。

嵌套低级别发现中的宏支持　现在，嵌套低级别发现规则已支持低级别发现宏，可用于：

- JSONPath 预处理参数
- 自定义 LLD 宏的 JSONPath 字段

由低级发现创建的触发器的可编辑标签　现在，可以为从触发器原型创建的触发器手动添加标签。从触发器原型继承的标签仍会自动应用。手动添加的标签可以在已发现的触发器上进行修改，并将包含在事件标签数组中，同时可用于支持标签的功能，例如过滤器、仪表板小部件和通知宏。

Linux 和 **AIX** 上的块设备发现　新增了`vfs.dev.get` 监控项键，用于在 Linux 和 AIX 上发现块设备。它可用于块设备的低级别发现。`vfs.dev.discovery` 监控项键仍然可用，以保持兼容性。

Linux 上的网络接口发现　新的`net.if.get` 监控项键已添加到 Linux 上的 Zabbix agent 和 Zabbix agent 2 中。
它可用于网络接口的低级别发现，返回包含详细信息的网络接口列表。
`net.if.discovery` 监控项键仍可用，以保持兼容性。

轻量级文件系统发现　Zabbix agent 和 Zabbix agent 2 的 `vfs.fs.get` 监控项现在支持两个可选参数，使文件系统发现更加轻量且可靠：

- **mode** - 在 `short` 模式下，监控项仅返回已挂载文件系统的列表及其基本属性，不包含磁盘空间和 inode 统计信息。由于不会查询文件系统本身，发现过程不会触发自动挂载，也不会受到无响应网络文件系统的影响。`full` 模式（默认）返回与之前相同的数据。
- **mountpoint** - 仅返回指定挂载点的数据，以便分别收集每个已发现文件系统的统计信息(例如，使用 `vfs.fs.get[full,{#FSNAME}]`)。

操作系统模板现在采用了这种方式；请参阅模板变更。

有关更多信息，请参阅vfs.fs.get 和已挂载文件系统的发现。

数据库

对多个 **PostgreSQL** 数据库主机的支持　Zabbix 现在支持将多个 PostgreSQL 数据库主机指定为以逗号分隔的 `host:port` 值列表，利用 PostgreSQL 连接故障转移支持。系统会按顺序尝试这些主机，直到建立读写连接为止。此行为可在前端安装 中的 *Database host* 字段，以及 `zabbix_server.conf` 和 `zabbix_proxy.conf` 中的DBHost 中使用。

优化的 **Zabbix** 服务器启动　Zabbix 服务器现在的启动速度显著更快。这是通过一次从数据库加载多个监控项的历史数据，而不是逐个加载来实现的。此改进适用于 Zabbix 服务器支持的所有数据库类型。

优化的 **Elasticsearch** 性能　Zabbix 服务器现在与Elasticsearch 的配合更加高效。服务器在检索数据时不再使用 Elasticsearch 的 *scroll* 功能，因此效率更高，并且会重用现有的 Elasticsearch 连接，而不是为每个请求都新建一个连接。此外，服务器在同时向 Elasticsearch 发送多个请求时也不再会卡住（使用 cURL 的 *multi-handle*）。

进程

SNMPv3 EngineID 缓存与复用　Zabbix 现在会缓存 SNMPv3 EngineID → IP 映射，并会在后续的 SNMPv3 检查中尝试复用已缓存的 EngineID，从而减少探测流量并提升轮询器性能。如果复用的 EngineID 没有响应，轮询器会回退到 EngineID 探测；在接口变更或持续失败后，还可能会移除过期条目。

历史缓存恢复期间优化 **proxy** 限流　已优化 proxy 限流逻辑，以提高历史缓存恢复期间的服务器稳定性。
当历史缓存使用率达到限流阈值时，服务器会像以前一样继续停止接受 proxy 数据。
当缓存使用率降至 60% 时，服务器开始处理限流列表，但对于包含非常大批次（大约超过 1 万条记录）的 proxy 上传，仍可能拒绝，直到缓存压力进一步降低。
此更改可在服务器恢复期间降低缓存反复过载的风险。

提高 **zabbix_get** 和 **zabbix_js** 的最大超时时间　zabbix_get 和zabbix_js 命令行工具的 `timeout` 参数的最大值已提高至 600 秒。

优化的 **housekeeping**　Housekeeping 已通过将 housekeeper 任务的创建移至数据库触发器而得到优化。
当删除监控项（包括低级别发现规则）、触发器、服务或网络发现规则时，现在由数据库触发器（而不是显式的应用层调用）将清理任务填充到 `housekeeper` 表中。
有关详细信息，请参见housekeeping procedure。

housekeeper 现在还会删除由随后已被删除的已发现主机或服务生成的网络发现事件，以及与随后已被删除的触发器问题相关的所有事件(此前，仅删除问题本身；相关事件只有在 housekeeper 的 *Trigger data storage period* 到期后才会被删除)。

手动关闭 — 恢复事件继承的触发器标签　在手动关闭 之后创建的恢复事件，除了监控项和主机标签外，还会继承触发器标签。这些标签会出现在事件标签数组中，并可用于通知宏，例如`{EVENT.RECOVERY.TAGS}` 和`{EVENT.RECOVERY.TAGSJSON}`。

Zabbix 守护进程的 **DNS** 查询缓存　当使用 `--with-ares` 构建 Zabbix 时，Zabbix 服务器、Zabbix proxy 和 Zabbix agent 已支持对所有 DNS 请求使用 c-ares 解析器，从而提供 DNS 查询缓存，并改进解析器故障切换能力。
DNS 查询缓存需要 c-ares 1.26.0 或更高版本。

在 **Windows** 上构建对 **c-ares** 的支持　现在可以在 Microsoft Windows 上使用 c-ares 解析器来构建 Zabbix agent。c-ares 可通过 vcpkg 安装，agent 构建支持使用 ARES=<vcpkg prefix>，或分别使用 ARESINCDIR/ARESLIBDIR 路径来指定头文件目录和库目录。

基于计数的触发器函数改进　触发器表达式中基于计数的函数（count()、item_count()、changecount() 等）的运行已得到改进，可更快地处理更多值。

运行时控制目标更新　Zabbix 服务器和 proxy 帮助信息以及manpages 中可用的运行时控制 目标列表已更新，以与受支持的进程类型 保持一致。

已新增 supervisor 目标，同时已移除不受支持的 discovery worker 和 preprocessing worker 目标。

安全性

proxy 和服务器访问控制　用户组配置中的新 *Proxy access list* 选项卡允许您授予用户组对 proxy 的访问权限，可以单独授予，也可以通过 proxy 组授予。

用户角色配置中的新 *Select "Server" for monitoring and discovery* 选项允许您禁止用户创建由服务器监控的主机，以及由服务器执行网络发现规则的主机；同时还可以禁止用户使用服务器测试监控项。

Zabbix agent 和 **Zabbix agent 2** 配置中的新 AllowKeyRegexp 和 DenyKeyRegexp 参数　Zabbix agent 和Zabbix agent 2 配置中新增了 AllowKeyRegexp 和 DenyKeyRegexp 参数。这些参数允许您定义基于正则表达式的模式，以便更精确、更灵活地匹配键及其参数。

身份验证

用于单点登录的可导入 **SAML** 证书　超级管理员用户现在可以直接在前端中导入证书和私钥，用于配置SAML。在 用户 > 身份验证 > *SAML* 设置中新增了三个字段：

- *IdP* 证书 - 身份提供商提供的 X.509 证书
- *SP* 证书 - 用于 SAML 交换的服务提供商证书
- *SP* 私钥 - 与 SP 证书对应的私钥

这些控件允许在 SAML 设置选项卡中添加新值或修改现有值。证书和私钥在保存到所选存储后端之前会先进行验证；无效或格式错误的值将被拒绝，并显示说明性错误信息。

Post-login URL per user role　It is now possible to enforce a specific URL after login based on the user role. This is handy for enforcing a specific page after login for provisioned users.

For more details, see User profile settings in user role configuration.

使用 **AppRole** 方法进行 **HashiCorp vault** 身份验证　HashiCorp vault 身份验证现在支持 AppRole 方法：

- 在服务器/proxy 配置中，使用新的 VaultAppRoleID 和 VaultAppSecretID 参数（参见示例）。
- 在前端 配置中，通过提供身份验证 role ID 和 secret ID 值。

小部件

时钟小组件 **"时区"** 选项的变更　时钟小组件配置表单中的时区选项已从高级配置部分移至小组件配置的主要部分，并重命名为覆盖时区。

此选项适用于本地时间和服务器时间时间类型。

Geomap 主机标记聚类　Geomap 部件现已支持 *Clustering* 参数，用于控制如何将彼此接近的主机标记合并为一个带计数的单个标记。通过为聚类设置地图缩放级别，您可以在大型地图上保持良好的可读性，并在放大时维持精确的可见性。

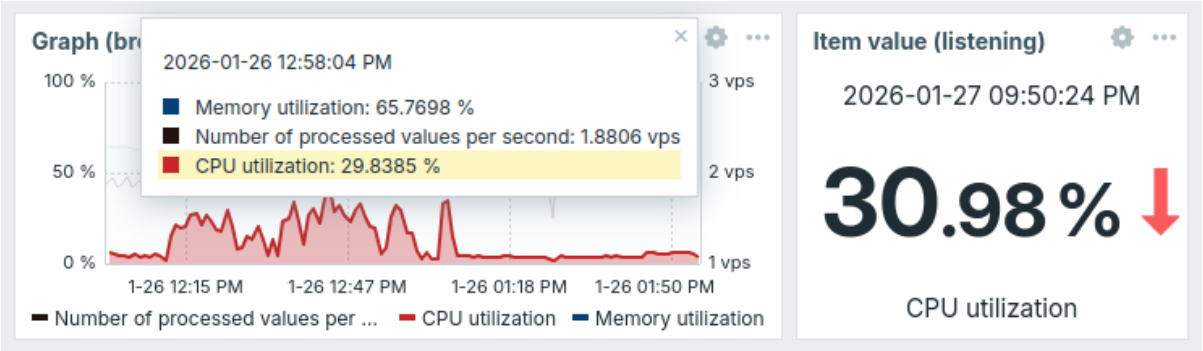
聚合 **"Top items"** 中的列　Top items 小组件现在可以将监控项模式分组到单个聚合列或行中，从而轻松呈现组合指标（例如：主机所有网络接口的总入站流量）。

配置监控项列时新增了三个选项：聚合列、列聚合函数和组合列名称。

图形小组件增强功能　图形小组件提供了多项易用性增强功能：

- 将鼠标悬停在图形上时显示的图形数据工具提示中，监控项按值降序排列。
- 将鼠标悬停在工具提示中的某个监控项上时，该监控项对应的图形会高亮显示，其他图形则变暗。
- 在工具提示中选择某个监控项时，其数据会广播给支持监听该数据的其他小组件。
- 已移除工具提示中显示的监控项数量限制。
- 新增标签中的主机名称设置，可用于选择是否在工具提示和图形图例中显示主机名称。

- 新增反转值设置，可用于翻转图形。



小部件描述字段的字符限制已提高　Gauge 和Item value 小部件中的 *Description* 字段字符限制已从 2048 提高到 65535 个字符。

前端

可导入的仪表板　仪表板仓库中提供了新的可导入仪表板：

- 多云成本仪表板 - 用于监控 AWS、Azure、Google Cloud 和 Oracle Cloud 中的云成本。
- 多云虚拟机仪表板 - 用于监控运行在 AWS、Google Cloud、Oracle Cloud 和 Azure 中的虚拟机。

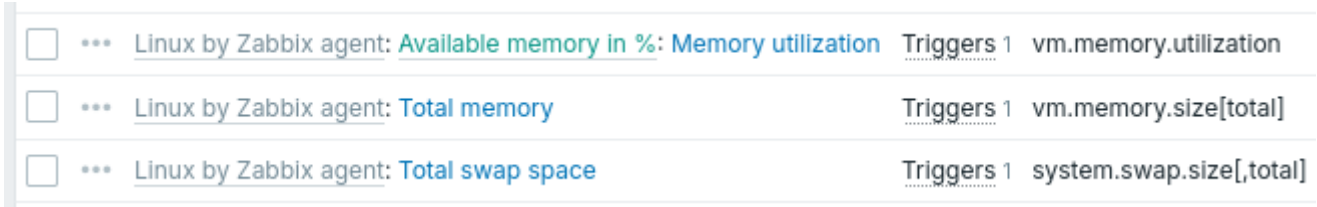
这些仪表板旨在与现有的 Zabbix 集成配合使用，将来自多个主机和监控目标的数据整合到统一的全局仪表板中。

仪表板导出和导入　仪表板现在可以导出和导入，这使得在不同的 Zabbix 实例之间迁移仪表板或在不同环境中重复使用它们更加容易。

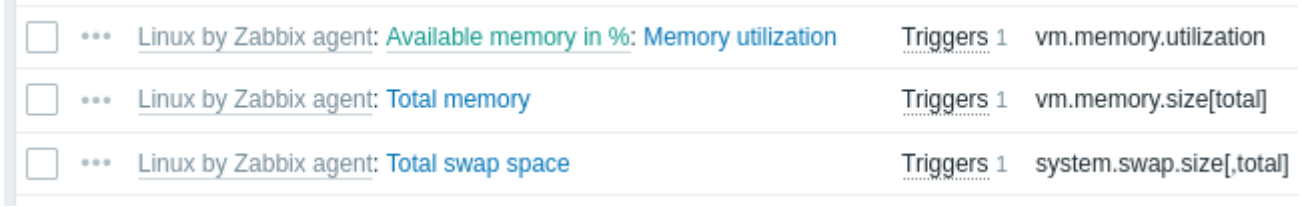
新增的导出选项允许将选定的仪表板保存为 YAML、JSON 或 XML 文件。这些文件之后可以通过导入功能重新导入，以恢复包含其页面、部件和配置的仪表板。

新增和嵌入式字体　Zabbix 现已包含新的嵌入式字体，可提升可读性、加快加载速度，并以尽量减少布局影响的方式渲染更多前端语言。由于这些字体为嵌入式字体，它们还能确保在不同系统之间保持一致的外观。

新字体：



旧版字体：

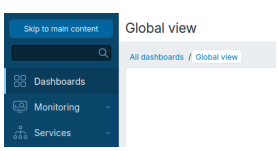


新字体已用于几乎所有主题。如有需要，可使用新添加的 *Blue (classic)* 和 *Dark (classic)* 主题，它们采用旧版字体。

等宽内容和图形所使用的字体族保持不变。

Zabbix UI 增强　Zabbix 用户界面经过现代化改造，提升了可访问性和易用性。

可访问性增强　已添加跳转到主要内容 链接，以改进键盘导航和可访问性。



该链接允许键盘和屏幕阅读器用户跳过重复的侧边栏导航，直接进入主要内容。在正常使用时它保持隐藏，仅在获得焦点时显示，例如在页面加载后按下 Tab 键，或通过屏幕阅读器访问时。

内联验证　前端中的以下表单已添加到支持内联验证的表单集合中：

- API token
- 审计日志
- 身份验证
- 自动注册
- Connector
- 发现原型
- 低级别发现规则
- 网络发现规则
- 事件关联
- 地理地图
- 图形原型
- 自定义图形
- GUI
- 主机组
- 主机原型
- Housekeeping
- 图标映射
- 图像
- 维护
- 媒体类型
- 通知
- 其他配置参数
- 配置文件
- proxy
- proxy 组
- 正则表达式
- 计划报告
- 脚本
- 服务
- SLA
- 模板组
- 测试监控项
- 测试媒体类型
- 超时
- 触发器显示选项
- 触发器表达式构造器
- 更新问题
- 用户组
- 全局用户宏
- 用户角色
- 用户

输入错误会在填写字段后立即显示，从而提升可用性并减少配置错误。

模态表单　以下表单现在会在模态窗口中打开：

- 发现原型 配置表单
- 低级发现规则配置表单
- 主机原型 配置表单
- 全局正则表达式配置表单

主机、模板、监控项和触发器中显示继承的标签　现在，模板、主机、监控项、Web 场景和触发器中的继承标签会以一致的方式显示和返回。

从模板/主机链继承的标签会显示在监测 > 最新数据 中。

凡是支持标签筛选的地方都可以按继承的标签进行筛选，包括 监测 > 最新数据和数据采集 部分，以及所有允许按主机、监控项、触发器或 Web 场景标签进行筛选的仪表板小组件。因此，无论标签定义在哪里，基于标签的选择和子筛选行为都保持一致。

模板、主机和主机原型配置表单中的 标签选项卡现在提供了一个单选控件，用于选择标签的显示方式：模板显示 模板标签 / 继承的标签和模板标签，而主机和主机原型显示 主机标签 / 继承的标签和主机标签。

继承的标签会通过标签名称旁边新增的轮廓文档图标进行视觉区分。

图形和饼图小组件新增了 监控项标签设置。

工具提示定位　现在可以通过拖动工具提示将其重新定位到新位置。例如，这适用于Graph 的工具提示，以及Latest Data 或Problems 部分中带有描述的工具提示。

前端配置中的新 **$ZBX_FEATURE_FLAGS** 变量　前端配置中新增了 $ZBX_FEATURE_FLAGS[] 变量，用于控制访问用户 HTTP 身份验证、模块和媒介类型配置的权限。

zabbix.conf.php 配置文件中新增了以下功能标志：

- $ZBX_FEATURE_FLAGS['http_auth_enabled'] = false 禁止访问用户 HTTP 身份验证配置（替代 $ALLOW_HTTP_AUTH 变量）。
- $ZBX_FEATURE_FLAGS['modules_config_enabled'] = false 禁止访问模块配置。
- $ZBX_FEATURE_FLAGS['media_type_denylist'] = [<value1>, <value2>, ...] 限制编辑使用指定传递方式的媒介类型（'email'、'script'、'sms' 或 'webhook'；可以指定一个或多个值）。

如果 zabbix.conf.php 配置文件存在，则会限制对 setup.php 的访问。

事件详情显示抑制原因　从 *Monitoring* > *Problems* 打开的事件详情 页面，现在会显示某个事件是否因为主机维护而被抑制，或者是否被用户手动抑制。

橙色图标  和  表示与维护相关的抑制/取消抑制，而灰色图标  和  表示手动抑制/取消抑制。

在problem 和event 历史记录中，由已删除用户执行的操作现在显示为 *Inaccessible user*。

系统信息报告增强　系统信息报告现在会显示服务器 ID（UUIDv7）。
 系统信息报告现在也对 Zabbix *Admin* 用户可用。

Zabbix SELinux 策略包的变更　对于 AlmaLinux、CentOS Stream、RHEL、Rocky Linux 和 Oracle Linux 的 10 版本，SELinux 策略由新的 zabbix-selinux-policy-main 包提供，该包包含来自上游 SELinux 仓库的 SELinux 策略。

现有的 zabbix-selinux-policy 包现在依赖于 zabbix-selinux-policy-main 或 selinux-policy-targeted-extra。zabbix-selinux-policy-main 包与 selinux-policy-targeted-extra 冲突；因此，这两个包不能同时安装。

如果您已经安装了 zabbix-selinux-policy，则无需手动更改。在包升级期间，zabbix-selinux-policy-main 将自动安装。如果已经安装了 selinux-policy-targeted-extra，则不会安装任何额外的包。

仪表板幻灯片放映设置　创建仪表板时，开始幻灯片放映复选框默认处于禁用状态，必须明确启用。slideshow URL 参数支持 on 或 off 值，可快速覆盖仪表板幻灯片放映设置。slideshow=1 参数已被弃用，将不再受支持。

链接到 **Zabbix Cloud proxy**　*Administration* > *Proxies* 中的proxy 列表现在包含一个 *Create proxy in Cloud* 按钮。
点击该按钮会在新的浏览器标签页中打开 Zabbix Cloud，您可以在其中部署 proxy，而无需在自己的基础设施上安装它。
有关更多信息，请参阅设置 Zabbix Cloud proxy。

文档

次要版本的整合文档页面　主版本的 Zabbix 次要版本发布文档现在将汇总到一个文档页面中，涵盖新功能 和升级说明。

升级说明

这些说明适用于从 Zabbix 7.4.x 升级到 Zabbix 8.0.0。

所有说明分为以下几类：

- 重大变更 - 可能会破坏现有安装的变更，以及与升级过程相关的其他重要信息
- 其他 - 描述 Zabbix 功能变更的所有其余信息

另请参见：

- 升级步骤，了解从 Zabbix 8.0.0 之前版本升级的所有相关信息；
- 升级 HA 集群，了解在 高可用（HA）集群中升级服务器的说明。

重大变更

从用户角色中移除主机接口 **API** 方法　升级 Zabbix 后，hostinterface.create、hostinterface.update、hostinterface.delete、hostinterface.massadd 和 hostinterface.massremove API 方法将从现有用户角色的 *Access to API* 列表中移除。

自定义主题和媒体类型　Zabbix 新增了三个预定义的前端主题：蓝色（经典）、深色（经典）和深蓝色，以及一种新的推送通知媒体类型。如果您有一个名称为这些名称之一（`blue-classic-theme`、`dark-classic-theme`、`dark-blue-theme`）的自定义主题，或有一个名为 “推送通知” 的媒体类型，请在升级前重命名该主题或媒体类型。否则，升级后：

- 预定义主题将替换自定义主题的样式表，而您自定义的图形颜色仍会继续使用，从而导致两种主题混合使用。
- 现有媒体类型将保持不变，但不会创建新的媒体类型。

使用其他名称的自定义主题和媒体类型不受升级影响。

内部 **proxy** 监控项仅限于服务器监控的主机　内部监控项 `zabbix[proxy,*]` 和 `zabbix[proxy group,*]` 不再支持用于由 proxy 或 proxy group 监控的主机。

不再支持 **1.6** 之前版本的 **Zabbix agent**　由于 Zabbix 服务器/proxy trapper 已移除对旧版纯文本和 XML 协议的支持，因此不再支持 1.6 之前版本的 Zabbix agent。

Logback 库最低版本　Logback 库的最低版本已从 1.5.16 提升至 1.5.25。

内联验证　前端中的以下表单已添加到支持内联验证的表单集合中：

- API token
- 审计日志
- 身份验证
- 自动注册
- Connector
- 事件关联
- 地理地图
- 图形原型
- 自定义图形
- GUI
- 主机组
- 主机原型
- Housekeeping
- 图标映射
- 图像
- 维护
- 媒体类型
- 通知
- 其他配置参数
- 个人资料
- proxy
- proxy 组
- 正则表达式
- 计划报告
- 脚本
- 服务
- SLA
- 模板组
- 测试监控项
- 测试媒体类型
- 超时
- 触发器显示选项
- 触发器表达式构造器
- 更新问题
- 用户组
- 全局用户宏
- 用户角色
- 用户

输入错误会在填写字段后立即显示，从而提升可用性并减少配置错误。

插件　Zabbix agent 2 的 Ceph 插件现在是一个可加载插件，并且需要额外的安装步骤。有关详细信息，请参阅 Ceph 插件的 readme。

MongoDB 插件中的 `tcp://` 方案已被弃用，但为了与现有配置保持向后兼容，仍予以保留。

Oracle 插件配置文件中的 `Plugins.Oracle.CallTimeout` 参数已不再生效。现在，最大等待时间由监控项超时决定。

MongoDB 插件版本支持　此插件版本不再支持 MongoDB 2.6–4.0。它使用 MongoDB Go Driver v2，要求 MongoDB 4.2 或更高版本。

要监控 MongoDB 2.6–4.0，请使用较早版本的插件。较早版本支持 MongoDB 2.6–5.3，并且可以与 Zabbix agent 2 版本 6.0.0 或更高版本配合使用。

最低所需 **PHP** 版本　最低所需的 PHP 版本已从 8.0.0 提升至 8.2.0。

扩展的 **UnsafeUserParameters** 列表　% 字符已添加到Zabbix agent 和Zabbix agent 2 的 `UnsafeUserParameters` 列表中。

已移动的筛选参数　作为可自定义表格功能的一部分，部分筛选参数已从筛选选项卡移至列选项。如果您保存的收藏筛选条件选项卡使用了以下任一筛选参数，则这些选项卡将被重置。升级后，您必须手动重新配置：

- 监控 > 主机：显示已抑制的问题
- 监控 > 最新数据：显示标签、标签名称、标签显示优先级、显示详细信息
- 监控 > 问题：显示标签、标签名称、标签显示优先级、显示运行数据、紧凑视图、显示详细信息、显示时间线、突出显示整行

JavaScript 预处理：内置对象方法现在为只读　预处理 脚本中的内置 JavaScript 对象方法现在为只读。如果您已经在监控项预处理脚本中修改了 JavaScript 原型上的内置方法，可能会遇到问题，因为现在不再允许修改内置原型方法。

MySQL 数据库字符集和排序规则　如果 Zabbix MySQL 数据库是使用 utf8mb3 字符集和 utf8mb3_bin 排序规则创建的，强烈建议在升级 Zabbix 之前将其转换为 utf8mb4 字符集和 utf8mb4_bin 排序规则，如修复 Zabbix 数据库字符集和排序规则 中所述。

其他

默认启用 **SAML** 签名　为提高安全性，在用户 > 身份验证中配置SAML 身份验证 时，所有 SAML 签名选项（消息、断言、*AuthN* 请求、注销请求和注销响应）现在默认启用。因此，除非取消选择签名选项，否则新的 SAML 配置现在需要 SP 私钥和 SP 证书。

升级不会影响现有的 SAML 配置。

已弃用的 **Kubernetes** 模板　Kubernetes 模板已弃用，已由 *Kubernetes Cluster by HTTP* 模板取代。旧模板仍予以保留，以便与现有配置向后兼容。

已弃用宏已移除　以下内置宏 的支持现已移除：

已移除	请改用
{ACK.DATE}	{EVENT.UPDATE.DATE}
{ACK.MESSAGE}	{EVENT.UPDATE.MESSAGE}
{ACK.TIME}	{EVENT.UPDATE.TIME}
{EVENT.ACK.HISTORY}	{EVENT.UPDATE.HISTORY}
{HOSTNAME<1-9>}	{HOST.HOST}
{IPADDRESS<1-9>}	{HOST.IP}
{PROFILE.*}	{INVENTORY.*}
{TRIGGER.COMMENT}	{TRIGGER.DESCRIPTION}
{TRIGGER.KEY}	{ITEM.KEY}
{STATUS}	{TRIGGER.STATUS}
{USER.ALIAS}	{USER.USERNAME}

历史数据存储提供程序的已弃用参数　以下参数现已弃用：

- Zabbix 服务器配置文件：`HistoryStorageDateIndex`、`HistoryStorageURL` 和 `HistoryStorageTypes`。请改用新的`HistoryProvider` 参数。
- Zabbix 前端配置文件：`$HISTORY`。请改用新的`$HISTORY_PROVIDERS` 参数。

这两个新参数都允许您选择历史数据存储提供程序：Elasticsearch、ClickHouse 或 SQL（默认）。请注意，不能同时使用这些新参数和已弃用的参数。

已从 **HTTP agent** 监控项中移除 **JSON** 和 **XML** 验证　HTTP agent 监控项的 *Request body* 字段中已移除 JSON 和 XML 验证。该字段不再检查该值是否为有效的 XML 或 JSON，并且将 *XML data* 选为 *Request body type* 时也不再需要 libxml2 库。

已隐藏受监控主机中的禁用链接　已禁用的 图形、仪表板和 *Web* 链接不再显示在受监控主机列表中。

JSON 数据类型　Zabbix 现在支持将 JSON 作为监控项值的数据类型。

如果您使用 TimescaleDB，则必须手动配置新的 `history_json` 超表（用于存储 JSON 值）。

如果您使用 Elasticsearch，已弃用的`HistoryStorageTypes` Zabbix 服务器配置参数的默认值现在包含 `json`。

新标签过滤器表单中已移除单选按钮　在创建新标签过滤器时，原本需要在 *All tags* 和 *Tag list* 之间进行选择的单选按钮已被移除。

使用 **iframe** 以 **HTML** 形式显示监控项值　监控项历史 小组件中的监控项数据在以 HTML 格式文本显示时，现在已隔离在 iframe 中。

基于 **HTTP** 的模板和媒体类型的 **TLS** 验证　连接到 HTTPS 端点的模板和媒体类型现在默认会验证端点的 TLS 证书。

此更改会影响所有云和 API 监控模板（例如，*Claude API by HTTP*、*Cloudflare by HTTP*、*Kubernetes Cluster by HTTP*）以及所有 webhook 媒体类型（例如，*Slack*、*Jira*、*Telegram*）。您可以使用已添加到这些模板和媒体类型中的新用户宏 `{$HTTP.TLS.VERIFY}` 更改此行为。将宏值设置为 `none` 可禁用验证，设置为 `peer` 可仅验证证书链和有效期，设置为 `full` 可执行完整验证。有关详细信息，请参阅每个HTTP 模板 或webhook 的 README 文件。

升级 Zabbix 时，模板和媒体类型不会自动升级。这样可以避免覆盖您所做的任何自定义更改。如需手动升级或添加模板和媒体类型，请参阅模板升级说明。

模板与主机向导的兼容性　所有开箱即用的模板现在都与主机向导 兼容。要升级它们，请参见模板升级。

配置字段的自动调整大小　现在，UI 中的大多数配置字段都会自动调整大小以适应其内容。

已替换的前端配置变量　前端配置文件(`zabbix.conf.php`)中的 `$ALLOW_HTTP_AUTH` 变量已被 `$ZBX_FEATURE_FLAGS['http_auth_enabl` 功能标志替换。

SNMPv1 和 **SNMPv2** 检查的并发处理　SNMPv1 和 SNMPv2 异步服务检查现在由可用的发现 worker 并发处理。

Zabbix SELinux 策略包的更改　对于 AlmaLinux、CentOS Stream、RHEL、Rocky Linux 和 Oracle Linux 的 10 版本，现有的 `zabbix-selinux-policy` 包现在依赖于 `zabbix-selinux-policy-main` 或 `selinux-policy-targeted-extra`。
 如果您已经安装了 `zabbix-selinux-policy`，则无需手动进行任何更改。在包升级期间，`zabbix-selinux-policy-main` 将自动安装。如果已经安装了 `selinux-policy-targeted-extra`，则不会安装任何额外的软件包。

APM support in Zabbix proxy packages　To use application performance monitoring after upgrading Zabbix proxy, choose the option for your operating system:

- RHEL 10 and its derivatives - install the EPEL repository, enable the CRB repository, and install the Zabbix proxy APM package (see details).
- Debian 12 or later, Raspberry Pi OS (Raspbian) 12 or later, Ubuntu 24.04 or later - upgrade Zabbix proxy packages as usual; APM is included.
- Other systems - compile Zabbix proxy from sources with the `--with-apm` option.

重命名的维护周期状态　维护周期状态已从 *Approaching* 重命名为 *Upcoming*。

已弃用的仪表板幻灯片放映参数　`slideshow=1` 参数的支持将被移除。已为 `slideshow` 参数引入新的值，用于启动和停止幻灯片放映：

- `slideshow=on` - 启动幻灯片放映
- `slideshow=off` - 停止幻灯片放映

全局正则表达式的更改

- 正则表达式配置表单中新增了 *Description* 字段。
- *Expression type* 选项已重命名：

Before	Now
Character string included	*Contains string*
Any character string included	*Contains any substring from list*
Character string not included	*Does not contain string*
Result is TRUE	*Matches regular expression*
Result is FALSE	*Does not match regular expression*

API 的默认访问权限　对于新安装的系统，默认情况下所有角色（包括新创建的角色和默认用户角色）均禁用 API 访问权限。

对于现有安装，默认情况下仅对新创建的角色禁用 API 访问权限。

Empty API allow list　In user roles, an *Allow list* with no API methods specified now denies access to all API methods. Previously, the empty list was ignored, and all API methods were allowed.

To preserve existing access, the upgrade adds the * wildcard (any method) to all user roles that have *Allow list* selected but no methods specified. After the upgrade, review these roles in *Users* > *User roles*: replace * with the methods the role actually needs, or clear the *Enabled* checkbox in the *Access to API* section if the role does not need API access.

新增功能和升级说明：**8.0.x**

本页面介绍 Zabbix 主版本各次次要版本中包含的所有新功能，同时也包括模板变更和升级说明。

另请参阅主版本的新增功能。

Zabbix 8.0.1 有哪些新内容　此版本尚未发布。

有关此版本，请参见升级说明。

升级说明

Zabbix appliance

概述　Zabbix appliance 提供了一种即时部署 Zabbix 服务器和前端的方式，无需手动设置，也无需重新利用现有服务器来运行 Zabbix。

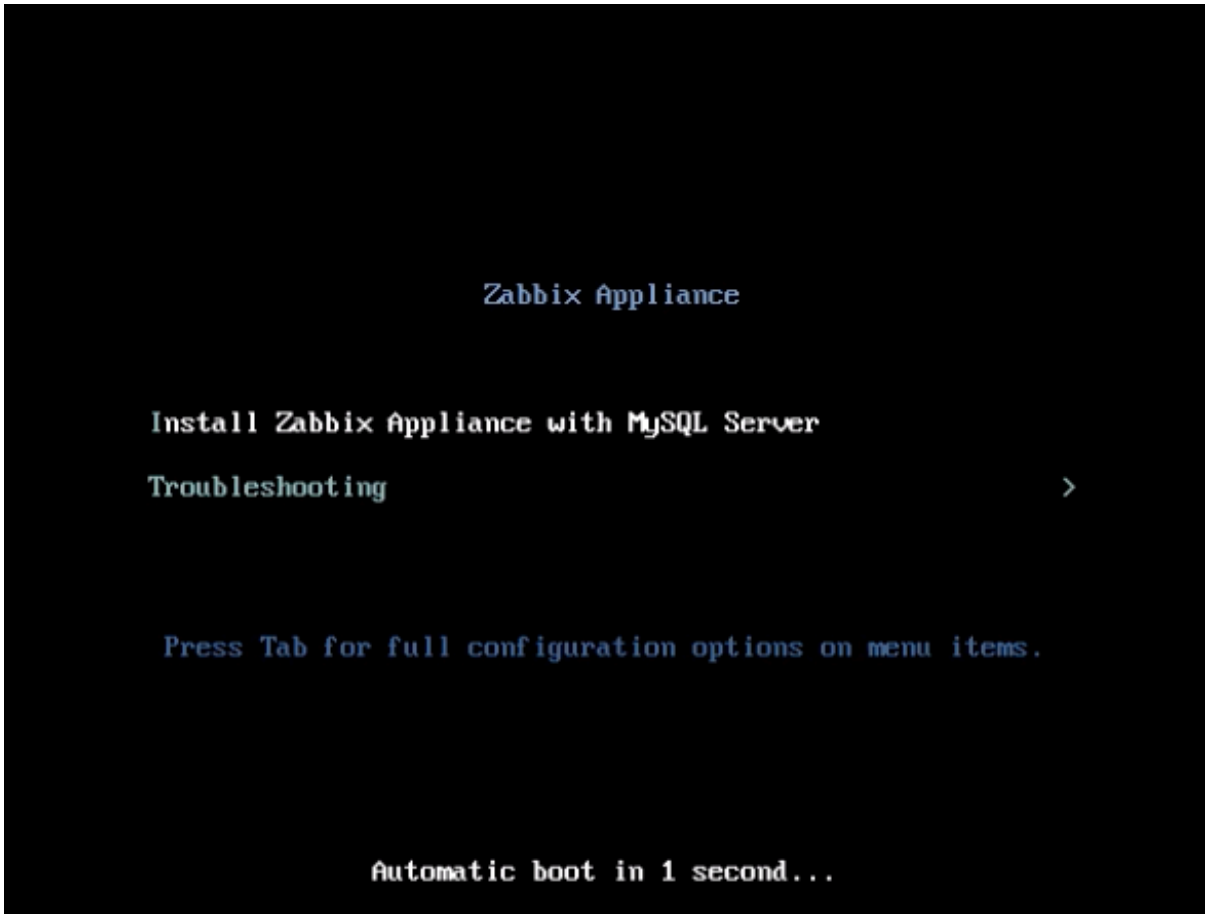
Zabbix 8.0 的 appliance 基于 AlmaLinux 或 Rocky Linux 等 Red Hat Enterprise Linux 10 衍生发行版，并包含预配置的 Zabbix 服务器（运行于 MySQL）以及运行在 NGINX web 服务器上的前端。

> **Attention:**
> 此 appliance 用于评估 Zabbix。不建议在严肃的生产环境中使用。

appliance 镜像可通过以下格式下载：

- 安装 CD/DVD（.iso）
- VMware（.vmx）- 请参阅说明
- 开放虚拟化格式（.ovf）
- Microsoft Hyper-V（.vhd/.vhdx）- 请参阅说明
- KVM、Parallels、QEMU、USB 闪存盘、VirtualBox、Xen（.raw）- 请参阅说明
- KVM、QEMU（.qcow2）

Zabbix 安装 CD/DVD 启动菜单：



快速开始　先决条件

1. 确保主机具备足够的资源，以满足虚拟机的系统要求：

- *RAM*：4 GB
- 磁盘空间：应至少为虚拟机分配 8 GB
- *CPU*：至少 2 个核心

2. 如果尚未安装，请安装用于启动 appliance 镜像的虚拟化软件（例如 VirtualBox）。
3. 下载 与您的虚拟化软件兼容格式的 appliance。
4. 检查网络设置，确保主机上已启用 DHCP。

安装

1. 从下载的镜像启动 appliance 虚拟机。有关说明，请参阅您的虚拟化软件文档，例如，VirtualBox 文档。
2. 配置虚拟机网络设置，以允许来自主机上的浏览器的访问。这可以通过启用 桥接模式来实现。
3. 使用默认系统凭据 登录虚拟机。
4. 要获取 IP 地址，请在虚拟机上运行以下命令：

```
ip addr show
```

5. 在主机上打开浏览器，并访问 appliance 通过 DHCP 获取的 IP 地址。
6. 使用默认凭据 登录 Zabbix，并开始监控。

配置　本节介绍常用的默认配置设置以及可用的自定义选项。

凭据

系统

- 用户名：`root`
- 密码：`zabbix`

Zabbix 前端

- 用户名：`Admin`

- 密码：zabbix

登录后，您可以在用户配置文件设置中更改默认密码，或创建新用户并删除默认用户。

数据库

所有数据库用户的密码都会在安装过程中随机生成。数据库中定义了以下用户：

Root：

- 用户名：root
- 密码：密码存储在 /root/.my.cnf 文件中

在 root 账户下无需输入密码。

Zabbix 服务器：

- 用户名：zabbix_srv
- 密码：密码存储在服务器配置文件中

Zabbix 前端：

- 用户名：zabbix_web
- 密码：密码存储在前端配置文件中

要更改数据库用户密码，请在 MySQL 中修改密码，并同步修改相应的配置文件。

前端访问

可以通过 *http://*< 虚拟机的 *IP*> 访问 Zabbix 前端。

默认情况下，允许从任何位置访问。若要限制访问，请修改 /etc/nginx/conf.d/zabbix.conf。保存编辑后的文件后，通过 SSH 以root 用户身份登录，并执行以下命令重启 NGINX：

```
systemctl restart nginx
```

静态 IP 地址

默认情况下，appliance 使用 DHCP 获取 IP 地址。要设置静态 IP 地址：

1. 以root 用户 身份登录。
2. 运行以下命令，将其中的值替换为您自定义的 IP 地址：

```
nmcli con mod eth0 ipv4.addresses 192.168.1.10/24    # Appliance IP 地址/CIDR 前缀
nmcli con mod eth0 ipv4.gateway 192.168.1.1      # 网关 IP 地址
nmcli con mod eth0 ipv4.dns 8.8.8.8    # DNS 服务器 IP 地址
nmcli con mod eth0 ipv4.method manual
nmcli con up eth0
```

或者，您也可以打开文件 /etc/NetworkManager/system-connections/eth0.nmconnection 并添加以下行：

```
[ipv4]
address1=192.168.1.10/24
dns=8.8.8.8;
gateway=192.168.1.1
method=manual
```

保存编辑后的文件后，运行以下命令：

```
nmcli con reload
nmcli con up eth0
```

要验证网络配置更改是否已应用，请运行以下命令：

```
nmcli dev show eth0
```

防火墙配置

为了管理防火墙设置，该 appliance 使用带有预定义规则的 iptables：

- 开放 SSH 端口（22 TCP）
- 开放 Zabbix agent（10050 TCP）和 Zabbix trapper（10051 TCP）端口
- 开放 HTTP（80 TCP）和 HTTPS（443 TCP）端口
- 开放 SNMP trap 端口（162 UDP）
- 开放到 NTP 端口（123 UDP）的出站连接
- 将 ICMP 数据包限制为每秒 5 个

- 丢弃所有其他传入连接

要开放其他端口，请修改 `/etc/sysconfig/iptables` 文件并重新加载防火墙规则：

```
systemctl reload iptables
```

软件仓库

Zabbix appliance 使用来自 Zabbix repository 的 *zabbix-release* 软件包。软件仓库配置在 `/etc/yum.repos.d/*` 目录中。

时区

默认情况下，appliance 使用 UTC 作为系统时钟。要更改时区，请将相应的文件从 `/usr/share/zoneinfo` 复制到 `/etc/localtime`，例如：

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

> **Note:**
> Zabbix 前端时区 是单独设置的，可以在前端设置中更改。Zabbix 前端的默认时区为 Europe/Riga。

文件位置

- Zabbix 服务器和 agent 配置文件位于 `/etc/zabbix`。
- Zabbix 服务器和 agent 日志文件位于 `/var/log/zabbix`。
- Zabbix 前端配置文件位于 `/etc/zabbix/web`。
- Zabbix 前端位于 `/usr/share/zabbix`。
- *zabbix* 用户的主目录为 `/var/lib/zabbix`。

系统服务

Systemd 服务可用。要查看 Zabbix 服务列表，请在虚拟机上运行以下命令：

```
systemctl list-units zabbix*
```

特定图像格式说明　VMware

- vmdk * 格式的映像可直接用于 VMware Player、Server 和 Workstation 产品。要在 ESX、ESXi 和 vSphere 中使用，必须通过 VMware vCenter Converter 进行转换（下载需要身份验证）。如果使用 VMware vCenter Converter，可能会遇到混合网络适配器的问题。在这种情况下，可以尝试在转换过程中指定 E1000 适配器。或者，转换完成后，删除现有适配器并添加 E1000 适配器。

HDD/闪存镜像（raw）

要启动该镜像，请运行：

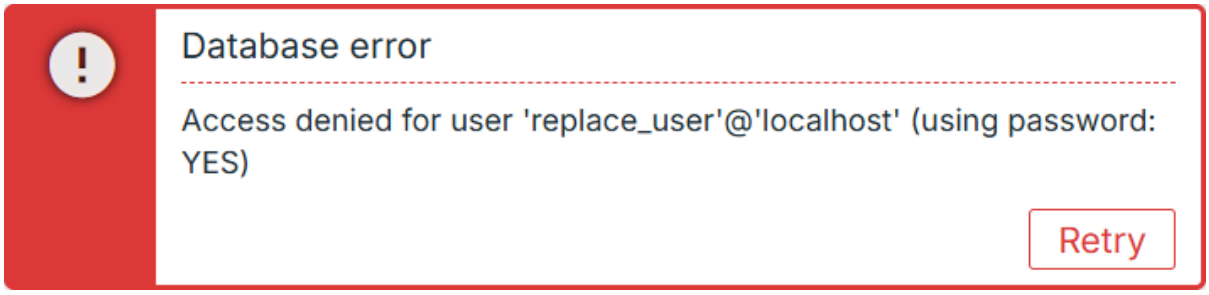
```
dd if=./zabbix_appliance_8.0.0.raw of=/dev/sdc bs=4k conv=fdatasync
```

将 `/dev/sdc` 替换为您的闪存/HDD 磁盘设备路径。

Hyper-V

如果 appliance 无法在 Hyper-V 中启动，请尝试按 **Ctrl+Alt+F2** 切换到 TTY 会话。

故障排除　如果您在尝试登录前端时遇到错误消息 *Access denied for user 'replace_user'@'localhost' (using password: YES)*，这可能表示安装仍在进行中。



如果等待几分钟后错误仍然存在，或者您观察到任何其他异常行为，这很可能意味着安装过程尚未成功完成。在这种情况下，我们建议删除当前 appliance，并按照相同的安装说明重新部署。此步骤通常可以解决问题。

请注意，不建议手动修复损坏的安装，因为这可能会导致进一步的复杂问题。

进程

概述　Zabbix 由多个协同工作的进程组成，用于监控系统并发送通知。每个进程在监控基础架构中都承担着特定的职责。

核心进程

- 服务器 - 执行数据轮询和捕获、计算触发器并向用户发送通知的核心进程。
- agent - 从受监控的主机收集数据。
- Agent 2 - 采用插件架构且功能增强的新一代 agent。
- proxy - 代表 Zabbix 服务器从受监控设备收集监控数据，并在分布式部署中将信息发送到集中式 Zabbix 服务器。
- Java 网关 - 支持监控 Java 应用程序。
- MCP 服务器 - 允许 AI agent 使用模型上下文协议与 Zabbix 协作。

为了从任何位置和任何平台轻松访问 Zabbix，系统提供了基于 Web 的前端。该前端是 Zabbix 服务器的一部分，通常（但并非必须）与运行服务器的物理机器相同。

所有配置数据以及 Zabbix 收集的数据都存储在数据库中。

标题　该头部存在于 Zabbix 组件之间的所有请求和响应消息中。
它用于确定消息长度、是否已压缩，以及是否为大数据包。

Zabbix 通信协议对每个连接的报文大小限制为 1GB。该限制同时适用于接收到的数据包长度和未压缩的数据长度。

在向 Zabbix proxy 发送配置时，数据包大小限制会增加到 4GB，以便同步大型配置。当压缩前的数据长度超过 4GB 时，Zabbix 服务器会自动开始使用大数据包格式（0x04 标志），这会将数据包大小限制提高到 16GB。

请注意，虽然大数据包格式可用于发送任何数据，但目前只有 Zabbix proxy 配置同步器能够处理大于 1GB 的数据包。

头部结构　头部由四个字段组成。头部中的所有数字均采用小端格式。

Field	Size	Size (large packet)	Description
<PROTOCOL>	4	4	"ZBXD" 或 5A 42 58 44
<FLAGS>	1	1	协议标志： 0x01 - Zabbix 通信协议 0x02 - 压缩 0x04 - 大包
<DATALEN>	4	8	数据长度。
<RESERVED>	4	8	使用压缩时（0x02 标志）- 未压缩数据的长度 未使用压缩时 - 00 00 00 00

标头示例　下面是一些代码片段，展示如何将 Zabbix 协议标头添加到你要发送的数据中，以便生成应发送给 Zabbix 的数据包，使其能够被正确解析。这些代码片段假定数据不大于 1GB，因此不会使用大数据包格式。

Python

```
packet = b"ZBXD\1" + struct.pack("<II", len(data), 0) + data
```

或者

```
def zbx_create_header(plain_data_size, compressed_data_size=None):
    protocol = b"ZBXD"
    flags = 0x01
    if compressed_data_size is None:
        datalen = plain_data_size
        reserved = 0
    else:
        flags |= 0x02
        datalen = compressed_data_size
        reserved = plain_data_size
    return protocol + struct.pack("<BII", flags, datalen, reserved)

packet = zbx_create_header(len(data)) + data
```

Perl

```
my $packet = "ZBXD\1" . pack("(II)<", length($data), 0) . $data;
```

或者

```
sub zbx_create_header($;$)
{
    my $plain_data_size = shift;
    my $compressed_data_size = shift;

    my $protocol = "ZBXD";
    my $flags = 0x01;
    my $datalen;
    my $reserved;

    if (!defined($compressed_data_size))
    {
        $datalen = $plain_data_size;
        $reserved = 0;
    }
    else
    {
        $flags |= 0x02;
        $datalen = $compressed_data_size;
        $reserved = $plain_data_size;
    }

    return $protocol . chr($flags) . pack("(II)<", $datalen, $reserved);
}

my $packet = zbx_create_header(length($data)) . $data;
```

PHP

```
$packet = "ZBXD\1" . pack("VV", strlen($data), 0) . $data;
```

或者

```
function zbx_create_header($plain_data_size, $compressed_data_size = null)
{
    $protocol = "ZBXD";
    $flags = 0x01;
    if (is_null($compressed_data_size))
    {
        $datalen = $plain_data_size;
        $reserved = 0;
    }
    else
    {
        $flags |= 0x02;
        $datalen = $compressed_data_size;
        $reserved = $plain_data_size;
    }
    return $protocol . chr($flags) . pack("VV", $datalen, $reserved);
}

$packet = zbx_create_header(strlen($data)) . $data;
```

Bash

```
datalen=$(printf "%08x" ${#data})
datalen="\\x${datalen:6:2}\\x${datalen:4:2}\\x${datalen:2:2}\\x${datalen:0:2}"
printf "ZBXD\1${datalen}\0\0\0\0%s" "$data"
```

概述　Zabbix 服务器是 Zabbix 软件的核心进程。

服务器负责轮询和捕获数据，计算触发器，并向用户发送通知。它是 Zabbix agent 和 proxy 上报系统可用性和完整性数据的中心组件。服务器本身也可以使用简单的服务检查，远程检查网络服务（例如 Web 服务器和邮件服务器）。

服务器是存储所有配置、统计和运行数据的中央存储库，也是 Zabbix 中在任何受监控系统出现问题时会主动向管理员发出告警的实体。

基本 Zabbix 服务器的运行分为三个不同的组件：Zabbix 服务器、前端和数据库存储。

Zabbix 的所有配置信息都存储在数据库中，服务器和前端都会与之交互。例如，当你使用前端（或 API）创建一个新的监控项时，它会被添加到数据库中的 items 表中。然后，大约每分钟一次，Zabbix 服务器会查询 items 表，获取处于活动状态的监控项列表，并将其存储在 Zabbix 服务器内部的缓存中。这就是为什么在 Zabbix 前端中所做的任何更改，最多可能需要两分钟才会显示在最新数据部分。

另请参见：

- Zabbix 服务器高可用性设置
- 配置参数：Zabbix 服务器

运行服务器　如果以软件包形式安装

Zabbix 服务器作为守护进程运行。可以通过执行以下命令启动服务器：

```
systemctl start zabbix-server
```

这在大多数 GNU/Linux 系统上都适用。在其他系统上，您可能需要运行：

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

同样，对于停止/重启/查看状态，请使用以下命令：

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

手动启动

如果上述方法不起作用，则需要手动启动它。找到 zabbix_server 二进制文件的路径并执行：

```
zabbix_server
```

你可以将以下命令行参数与 Zabbix 服务器一起使用：

Parameter	Description
-c --config <config-file>	配置文件的路径。
-f --foreground	以前台方式运行 Zabbix 服务器。
-R --runtime-control <option>	执行管理功能。
-T --test-config	验证配置文件并退出。
-h --help	显示帮助信息并退出。
-V --version	显示版本号。

使用命令行参数运行 Zabbix 服务器的示例：

```
zabbix_server -c /usr/local/etc/zabbix_server.conf
zabbix_server --help
zabbix_server -V
```

运行时控制

运行时控制选项：

选项	描述	目标
config_cache_reload	重新加载配置缓存。如果当前正在加载缓存，则忽略此操作。	

选项	描述	目标
history_cache_clear[=<target>]	清除由其 ID 指定的监控项的历史数据缓存。 影响该监控项的所有值，但第一个值和最后一个值除外。	**target** - 监控项的 ID
diaginfo[=<section>]	在服务器日志文件中收集诊断信息。	historycache - 历史数据缓存统计信息； valuecache - 值缓存统计信息； preprocessing - 预处理管理器统计信息； alerting - 告警管理器统计信息； lld - LLD 管理器统计信息； locks - 互斥锁列表（在 *BSD* 系统上为空）； connector - 队列最长的连接器统计信息。
ha_status	记录高可用性（HA）集群状态。	
ha_remove_node=<target>	移除由其名称或 ID 指定的高可用性（HA）节点。 请注意，无法移除活动/备用节点。	**target** - 节点的名称或 ID（可通过运行 ha_status 获取）。
ha_set_failover_delay=<delay>	设置高可用性（HA）故障转移延迟。 支持时间后缀，例如 10s、1m。	
proxy_config_cache_reload[=<target>]	重新加载 proxy 配置缓存。	**target** - 以逗号分隔的 proxy 名称列表。 如果未指定目标，则重新加载所有 proxy 的配置。
secrets_reload	从 Vault 重新加载机密信息。	
service_cache_reload	重新加载服务管理器缓存。	
snmp_cache_reload	重新加载 SNMP 缓存——清除所有主机的 SNMP 引擎属性（引擎时间、引擎启动次数、引擎 ID、凭据）。排查 SNMP 问题时，可使用此选项强制清除全局缓存。	
housekeeper_execute	启动管家程序。 如果管家程序当前正在运行，则忽略此操作。	
trigger_housekeeper_execute	启动触发器管家程序。 如果触发器管家程序当前正在运行，则忽略此操作。	
log_level_increase[=<target>]	提高日志级别：如果未指定目标，则影响所有进程。 *BSD* 系统不支持此操作。	**process type** - 指定类型的所有进程（例如 poller）。 请参阅所有服务器进程类型。 **process type,N** - 进程类型和编号（例如 poller,3）。 **pid** - 进程标识符（1 到 65535）。对于更大的值，请将目标指定为 “process type,N”。
log_level_decrease[=<target>]	降低日志级别：如果未指定目标，则影响所有进程。 *BSD* 系统不支持此操作。	
prof_enable[=<target>]	启用性能分析。 如果未指定目标，则影响所有进程。 启用性能分析后，将按函数名称提供所有读写锁/互斥锁的详细信息。	**process type** - 指定类型的所有进程（例如 history syncer） 可作为性能分析目标的进程类型：alerter、alert manager、availability manager、configuration syncer、discovery manager、escalator、history poller、history syncer、housekeeper、http poller、icmp pinger、ipmi manager、ipmi poller、java poller、lld manager、lld worker、odbc poller、poller、preprocessing manager、proxy poller、self-monitoring、service manager、snmp trapper、task manager、timer、trapper、unreachable poller、vmware collector。 **process type,N** - 进程类型和编号（例如 history syncer,1）。 **pid** - 进程标识符（1 到 65535）。对于更大的值，请将目标指定为 “process type,N”。 **scope** - rwlock、mutex、processing 可与进程类型和编号一起使用（例如 history syncer,1,processing），也可与指定类型的所有进程一起使用（例如 history syncer,rwlock）。

选项	描述	目标
prof_disable[=<target>]	禁用性能分析。 如果未指定目标，则影响所有进程。	**process type** - 指定类型的所有进程（例如 `history syncer`）。 可作为性能分析目标的进程类型：请参阅 `prof_enable`。 **process type,N** - 进程类型和编号（例如 `history syncer,1`）。 **pid** - 进程标识符（1 到 65535）。对于更大的值，请将目标指定为 “process type,N”。

当将某个进程类型指定为运行时控制的目标时，该命令仅适用于该类型的正在运行的进程。如果当前没有指定类型的进程在运行，命令将返回错误：

```
zabbix_server -R log_level_increase='discovery manager'
Cannot redirect signal: "discovery manager" process does not exist
```

使用运行时控制重新加载服务器配置缓存的示例：

```
zabbix_server -c /usr/local/etc/zabbix_server.conf -R config_cache_reload
```

使用运行时控制重新加载 proxy 配置的示例：

```
### 重新加载所有 proxy 的配置：
zabbix_server -R proxy_config_cache_reload

### 重新加载 Proxy1 和 Proxy2 的配置：
zabbix_server -R proxy_config_cache_reload=Proxy1,Proxy2
```

使用运行时控制清除某个监控项历史缓存的示例：

```
zabbix_server -c /usr/local/etc/zabbix_server.conf -R history_cache_clear=42243
```

使用运行时控制收集诊断信息的示例：

```
### 将所有可用的诊断信息收集到服务器日志文件中：
zabbix_server -R diaginfo

### 将历史缓存统计信息收集到服务器日志文件中：
zabbix_server -R diaginfo=historycache
```

使用运行时控制重新加载 SNMP 缓存的示例：

```
zabbix_server -R snmp_cache_reload
```

Attention:
当通过 Zabbix 前端更新 SNMPv3 接口时，在大多数情况下，Zabbix 会自动为该接口重新加载新的 SNMPv3 凭据；仅当更改凭据后轮询仍然失败时（例如，由于 engineBoots/engineID 不一致或非 RFC 设备），或者当您需要强制清空全局 SNMP 缓存以进行故障排查时，才使用 `-R snmp_cache_reload`。

使用运行时控制触发 housekeeper 执行的示例：

```
zabbix_server -c /usr/local/etc/zabbix_server.conf -R housekeeper_execute
```

使用运行时控制更改日志级别的示例：

```
### 提高所有进程的日志级别：
zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase

### 提高第二个 poller 进程的日志级别：
zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=poller,2

### 提高 PID 为 1234 的进程的日志级别：
zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=1234

### 降低所有 http poller 进程的日志级别：
zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_decrease="http poller"
```

将 HA 故障转移延迟设置为最小 10 秒的示例：

```
zabbix_server -R ha_set_failover_delay=10s
```

进程用户

Zabbix 服务器设计为以非 root 用户运行。它将以启动它的任意非 root 用户身份运行。因此，您可以以任何非 root 用户运行服务器，而不会有任何问题。

如果您尝试以'root' 运行，它将切换到硬编码的'zabbix' 用户，该用户必须在您的系统上存在。只有在相应修改服务器配置文件中的'AllowRoot' 参数后，您才能以'root' 运行服务器。

如果 Zabbix 服务器和agent 运行在同一台机器上，建议用于运行服务器的用户与用于运行 agent 的用户不同。否则，如果两者以同一用户运行，agent 可以访问服务器配置文件，并且 Zabbix 中任何 Admin 级别用户都可以相当容易地获取例如数据库密码。

配置文件

服务器配置文件的默认路径取决于操作系统和 Zabbix 的安装方式：

- 从源代码安装：`/usr/local/etc/zabbix_server.conf`，或由编译时变量 `--sysconfdir` 或 `--prefix` 设置的路径。
- 从软件包和 appliance 安装：`/etc/zabbix/zabbix_server.conf`。

有关配置 `zabbix_server` 的详细信息，请参阅配置文件选项。

启动脚本

这些脚本用于在系统启动/关闭期间自动启动/停止 Zabbix 进程。
这些脚本位于 misc/init.d 目录下。

服务器进程类型和线程

- `agent poller` - 用于被动检查的异步轮询进程，包含一个工作线程；
- `alert manager` - 告警队列管理器；
- `alert syncer` - 告警数据库写入器；
- `alerter` - 用于发送通知的进程；
- `availability manager` - 用于更新主机可用性的进程；
- `browser poller` - 用于浏览器监控项检查的轮询器；
- `configuration syncer` - 用于管理配置数据内存缓存的进程；
- `configuration syncer worker` - 用于解析和同步监控项名称中用户宏值的进程；
- `connector manager` - 连接器管理进程；
- `connector worker` - 用于处理来自连接器管理器请求的进程；
- `discovery manager` - 用于发现设备的管理进程；
- `discovery worker` - 用于处理来自发现管理器的发现任务的进程；
- `escalator` - 用于升级操作的进程；
- `ha manager` - 用于管理高可用性的进程；
- `history poller` - 用于处理需要数据库连接的计算检查的进程；
- `history syncer` - 历史数据数据库写入器；
- `housekeeper` - 用于删除过期数据（监控项历史数据和趋势数据、用户会话、事件等）以及删除对象遗留数据的进程；
- `http agent poller` - 用于 HTTP 检查的异步轮询进程，包含一个工作线程；
- `http poller` - Web 监控轮询器；
- `icmp pinger` - 用于 icmpping 检查的轮询器；
- `internal poller` - 用于内部检查的轮询器；
- `ipmi manager` - IPMI 轮询器管理器；
- `ipmi poller` - 用于 IPMI 检查的轮询器；
- `java poller` - 用于 Java 检查的轮询器；
- `lld manager` - 低级别发现任务的管理进程；
- `lld worker` - 低级别发现任务的工作进程；
- `odbc poller` - 用于 ODBC 检查的轮询器；
- `poller` - 用于被动检查的普通轮询器；
- `preprocessing manager` - 预处理任务管理器，包含预处理工作线程；
- `preprocessing worker` - 用于数据预处理的线程；
- `proxy poller` - 用于被动 proxy 的轮询器；
- `proxy group manager` - proxy 负载均衡和高可用性管理器；
- `report manager`- 计划报表生成任务管理器；
- `report writer` - 用于生成计划报表的进程；
- `self-monitoring` - 用于收集服务器内部统计信息的进程；
- `service manager` - 通过从历史数据同步器、任务管理器和告警管理器接收有关问题、问题标签和问题恢复的信息来管理服务的进程；
- `snmp poller` - 用于 SNMP 检查的异步轮询进程，包含一个工作线程（仅适用于 `walk[OID]` 和 `get[OID]` 监控项）；

- snmp trapper - 用于 SNMP 陷阱的 trapper；
- supervisor - 协调服务器进程和线程的启动与关闭，并监控其就绪状态；
- task manager - 用于远程执行其他组件请求的任务的进程（例如，关闭问题、确认问题、立即检查监控项值、远程命令功能）；
- telemetry query poller - 用于遥测查询监控项检查的轮询器；
- timer - 用于处理维护期的计时器；
- trapper - 用于主动检查、陷阱和 proxy 通信的 trapper；
- trigger housekeeper - 用于删除由后来已删除的触发器生成的问题和事件的进程；
- unreachable poller - 用于不可达设备的轮询器；
- vmware collector - VMware 数据收集器，负责从 VMware 服务收集数据。

可以使用服务器日志文件观察这些进程类型。

服务器日志文件创建时，仅文件所有者具有读写权限。此外，文件所有者所属的组也具有读取权限。其他所有权限均被拒绝。

可以使用 zabbix[process,<type>,<mode>,<state>] 内部监控项监控各种类型的 Zabbix 服务器进程。

历史同步器事务统计

history syncer 进程标题会显示有关 history syncer 事务的详细统计信息：

```
205182 ?        S      0:00  zabbix_server: history syncer #2 [processed 0 values, 0+0 triggers in 0.00002
205183 ?        S      0:00  zabbix_server: history syncer #3 [processed 18 values, 7+0 triggers in 0.0026
205184 ?        S      0:00  zabbix_server: history syncer #4 [processed 0 values, 0+0 triggers in 0.00002
```

在 "A+B triggers" 中：

- A - 因历史值而处理的触发器；
- B - 因定时器而处理的触发器。

processed...in N (<timings>) sec 中的各项耗时如下：

- 将监控项值写入数据库所花费的时间；
- 更新监控项数据（状态、错误、主机库存等）所花费的时间；
- 将趋势数据刷新到数据库所花费的时间；
- 计算触发器所花费的时间；
- 处理事件和动作所花费的时间。

Housekeeping 过程

housekeeper 进程会定期删除过期数据（监控项历史数据和趋势数据、用户会话、事件等），以及已删除对象遗留的数据。它按周期运行，其执行频率和每个周期的删除上限分别由HousekeepingFrequency 和MaxHousekeeperDelete 决定。在一个周期中未删除的任何数据都会延续到下一个周期。可在 *Administration* > *Housekeeping* 中启用并配置自动 housekeeping。

当 housekeeper 进程删除已删除对象遗留的数据时，它依赖于 housekeeper 表中的任务；该表会在对象被删除时填充。例如，当你删除一个主机时，Zabbix 会删除其监控项，但不会删除这些监控项的历史数据、趋势数据或问题。相反，数据库触发器会向 housekeeper 表填充由以下字段组成的任务：

- housekeeperid - 任务 ID
- object - 对象类型（0 - 监控项；1 - 触发器；2 - service；3 - 已发现主机；4 - 已发现 service）
- objectid - 对象 ID（帮助 housekeeper 查找与对象相关的数据）

例如，删除一个包含两个监控项和一个触发器的主机后，housekeeper 表会如下所示：

```
+---------------+--------+-----------+
| housekeeperid | object | objectid |
+---------------+--------+-----------+
|             1 |      1 |    28724 |
|             2 |      0 |    59396 |
|             3 |      0 |    59397 |
+---------------+--------+-----------+
```

数据库触发器会向 housekeeper 表填充任务，而不会检查对象相关数据；该检查由 housekeeper 进程在删除数据时完成：

- 对于已删除的监控项（包括 LLD 规则）- 会从所有历史和趋势表（history、history_str、history_log、history_uint、history_text、history_bin、history_json、trends、trends_uint）中删除其数据。还会从 problem 表中删除其内部事件。
- 对于已删除的触发器 - 会从与事件相关的表（problem、event_symptom、events；数据库本身也会从 event_recovery 表中删除事件）中删除其事件。还会通知 service manager 进程有关已删除事件的信息。

Note:
单独的 trigger housekeeper 进程负责一个更窄的任务——删除没有已知源触发器的问题和事件。其运行频率由ProblemHousekeepingFrequency 控制。
在触发器 housekeeping 过程启动之前，由后来已被删除的触发器创建的问题，可能仍然会生成 service 问题并将其分配给 service。如果你的环境中有许多基于频繁发现/取消发现触发器的 service 状态计算规则，请考虑通过调整服务器配置参数ProblemHousekeepingFrequency 来提高 housekeeping 过程的频率。

- 对于已删除的 service - 会从 problem 表中删除其问题；其在 events 表中的事件仅会在配置的数据保留期 之后删除。
- 对于已删除的已发现主机/service（网络发现）- 会从 events 表中删除其事件。

housekeeper 进程只会删除那些未与问题关联的事件。例如，如果一个过期的问题/恢复事件与一个未关闭的问题相关联，则不会被删除。当 housekeeper 删除过期实体时，会先删除问题，再删除事件。

Note:
使用 partition 模式（TimescaleDB 分区表）的表会被跳过；只会处理使用 regular 模式的表。

支持的平台　由于安全要求以及服务器运行的关键任务性质，UNIX 是唯一能够持续提供所需性能、容错能力和弹性的操作系统。Zabbix 运行于市场领先版本。

Zabbix 服务器已在以下平台上进行测试：

- Linux
- Solaris
- AIX
- HP-UX
- Mac OS X
- FreeBSD
- OpenBSD
- NetBSD
- SCO Open Server

Note:
Zabbix 也可能在其他类 Unix 操作系统上正常运行。

区域设置　请注意，服务器需要 UTF-8 区域设置，以便正确解析某些文本项。大多数现代类 Unix 系统默认都使用 UTF-8 区域设置，不过在某些系统上，可能需要专门进行设置。

Zabbix 服务器高可用性设置

概述

高可用性（HA）通常是关键基础设施所必需的，这类基础设施几乎不能容忍停机。因此，对于任何可能发生故障的服务，都必须提供故障切换选项，以便在当前服务发生故障时接管。

Zabbix 提供了一个原生高可用性解决方案，易于设置，并且不需要任何先前的 HA 经验。原生 Zabbix HA 可用于为 Zabbix 服务器的软件/硬件故障提供额外一层保护，或减少因维护导致的停机时间。

在 Zabbix 高可用模式下，多个 Zabbix 服务器作为集群中的节点运行。当集群中的一台 Zabbix 服务器处于活动状态时，其他服务器处于待机状态，随时准备在必要时接管。



切换到 Zabbix HA 不具有约束性。您可以在任何时候切换回单机运行模式。

另请参见：实现细节

启用高可用集群

将 Zabbix 服务器作为集群节点启动

要将 Zabbix 服务器作为集群节点启动，需要在服务器配置中设置两个参数：

- 必须为每个将作为 HA 集群节点的 Zabbix 服务器指定 HANodeName 参数。

这是一个唯一的节点标识符（例如 zabbix-node-01），服务器将在 agent 和 proxy 配置中通过该标识符进行引用。如果未指定 HANodeName，服务器将以独立模式启动。

- 必须为每个节点指定 NodeAddress 参数。

Zabbix 前端将使用 NodeAddress 参数（address:port）连接到活动服务器节点。NodeAddress 必须与相应 Zabbix 服务器的 IP 或 FQDN 名称匹配。

修改配置文件后，重启所有 Zabbix 服务器。此时，它们将作为集群节点启动。可以在报表 > 系统信息中查看服务器的新状态，也可以运行以下命令：

```
zabbix_server -R ha_status
```

此运行时命令会将当前 HA 集群状态记录到 Zabbix 服务器日志中（同时输出到 stdout）：

```
Failover delay: 60 seconds
Cluster status:
   #  ID                        Name                 Address                 Status      Last Access
   1. ckzxxqg7u0001lsropenyzh3m zabbix-node-01       64.227.66.193:10051     standby     0s
   2. ckzxyqo1k00013frpq539e1jp zabbix-node-02       64.227.74.25:10051      active      3s
```

准备前端

确保 Zabbix 前端配置文件（zabbix.conf.php）中未定义 Zabbix 服务器的 address:port。

```
// Uncomment and set to desired values to override Zabbix hostname/IP and port.
// $ZBX_SERVER                  = '';
// $ZBX_SERVER_PORT             = '';
```

Zabbix 前端将通过读取 Zabbix 数据库中 nodes 表的设置，自动检测活动节点。活动节点的节点地址将用作 Zabbix 服务器地址。

proxy 配置

HA 集群节点（服务器）必须列在被动或主动 Zabbix proxy 的配置中。

对于被动 proxy，节点名称必须列在 proxy 的 Server 参数 中，并以 逗号分隔。

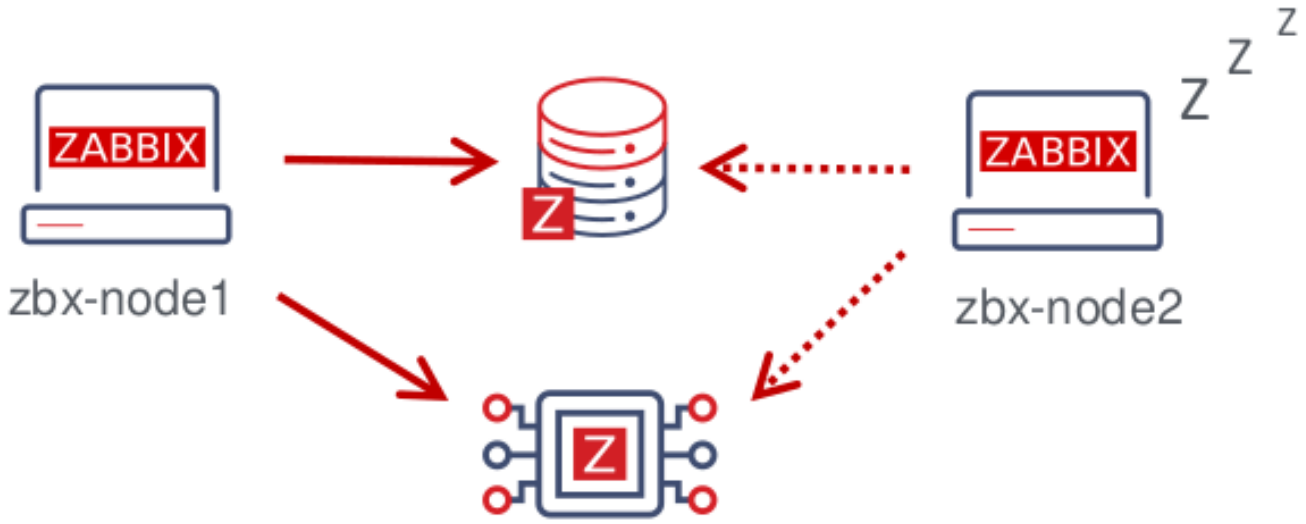
```
Server=zabbix-node-01,zabbix-node-02
```

对于主动 proxy，节点名称必须列在 proxy 的 Server 参数 中，并以 分号分隔。

```
Server=zabbix-node-01;zabbix-node-02
```

Agent 配置

HA 集群节点（服务器）必须列在 Zabbix agent 或 Zabbix agent 2 的配置中。



要启用被动检查，节点名称必须列在 Server 参数 中，并以 逗号分隔。

```
Server=zabbix-node-01,zabbix-node-02
```

要启用主动检查，节点名称必须列在 ServerActive 参数 中。请注意，对于主动检查，节点与其他服务器之间必须用逗号分隔，而节点彼此之间必须用 分号分隔，例如：

```
ServerActive=zabbix-node-01;zabbix-node-02
```

故障切换到备用节点

如果活动节点停止，Zabbix 将自动故障切换到另一个节点。
要发生故障切换，必须至少有一个节点处于备用状态。

故障切换有多快？
所有节点都会每 5 秒更新一次其最后访问时间（如果状态发生变化，也会更新状态）。
因此：

- 如果活动节点关闭并成功将其状态报告为 “stopped”，另一个节点会在 **5** 秒内接管。
- 如果活动节点关闭/变得不可用，但无法更新其状态，备用节点会等待 故障切换延迟 + 5 秒后再接管。

故障切换延迟是可配置的，支持范围为 10 秒到 15 分钟（默认 1 分钟）。
要更改故障切换延迟，可以运行：

```
zabbix_server -R ha_set_failover_delay=5m
```

管理 HA 集群

可以使用专用的运行时控制选项管理 HA 集群的当前状态：

- `ha_status` - 将 HA 集群状态记录到 Zabbix 服务器日志（以及标准输出）中；
- `ha_remove_node=target` - 移除由其 `<target>` 标识的 HA 节点，其中 `<target>` 为节点的名称或 ID（名称/ID 可从运行 `ha_status` 的输出中获取），例如：

```
zabbix_server -R ha_remove_node=zabbix-node-02
```

请注意，无法移除活动/备用节点。

- `ha_set_failover_delay=delay` - 设置 HA 故障转移延迟（介于 10 秒和 15 分钟之间；支持时间后缀（例如 30s、1m））。

可以通过以下方式监控节点状态：

- 在报表 > 系统信息中。
- 在系统信息仪表板小组件中。
- 使用服务器的 `ha_status` 运行时控制选项（见上文）。

内部监控项 `zabbix[cluster,discovery,nodes]` 可用于节点发现，因为它会返回包含高可用性节点信息的 JSON。

禁用 HA 集群

要禁用高可用性集群：

1. 备份配置文件。
2. 停止备用节点。

3. 从活动主服务器中删除 HANodeName 参数。
4. 重启主服务器（它将以独立模式启动）。

升级 HA 集群

要对 HA 节点执行主版本升级：

1. 停止所有节点。
2. 创建完整的数据库备份。
3. 如果数据库使用复制，请确保所有节点都已同步且没有问题。如果复制已损坏，请不要升级。
4. 选择一个将执行数据库升级的单个节点，通过注释掉 HANodeName 将其配置更改为单机模式，并对其进行升级。
5. 确保数据库升级已完全完成（系统信息应显示 Zabbix 服务器正在运行）。
6. 以 HA 模式重启该节点。
7. 升级并启动其余节点（由于此时数据库已经升级，因此无需将它们切换为单机模式）。

在次版本升级中，只需升级第一个节点，确认其已完成升级并正在运行，然后再开始升级下一个节点。

实现细节

高可用性（HA）集群是一种可选方案，支持 Zabbix 服务器。原生 HA 方案的设计目标是使用简单，可跨站点工作，并且对 Zabbix 可识别的数据库没有特定要求。用户可以根据其环境中的高可用性需求，自由选择使用原生 Zabbix HA 方案，或第三方 HA 方案。

该方案由多个 `zabbix_server` 实例或节点组成。每个节点：

- 单独配置。
- 使用同一个数据库。
- 可能有几种模式：active、standby、unavailable、stopped。

同一时间只能有一个节点处于 active（工作）状态。standby 节点只运行一个进程 - HA manager。standby 节点不进行数据采集、处理或其他常规服务器活动；它们不监听端口；数据库连接数最少。

active 节点和 standby 节点都会每 5 秒更新一次各自的最后访问时间。每个 standby 节点都会监控 active 节点的最后访问时间。如果 active 节点的最后访问时间超过了 “failover delay” 秒，standby 节点会切换为 active 节点，并将之前 active 的节点标记为 unavailable 状态。

active 节点会监控自身的数据库连接状态 - 如果连接丢失超过 `failover delay-5` 秒，则必须停止所有处理并切换到 standby 模式。active 节点还会监控 standby 节点的状态 - 如果某个 standby 节点的最后访问时间超过了 `failover delay` 秒，则该 standby 节点会被标记为 `unavailable` 状态。

这些节点的设计可兼容 Zabbix 的次要版本之间的变化。

配置参数：**Zabbix** 服务器

概述

配置参数可用于自定义 Zabbix 服务器行为。

在 Zabbix 服务器配置文件（`zabbix_server.conf`）中指定参数值。除非明确说明参数为必需参数，否则所有参数均为可选参数。

本页面结构如下：

- 参数概览（要查看参数的完整详细信息，请单击参数名称）
- 参数详细信息

参数	描述
AlertScriptsPath	指定告警脚本位置。
AllowRoot	允许服务器以 `root` 身份运行。
AllowSoftwareUpdateCheck	允许从 `zabbix.com` 接收软件更新信息。
AllowUnsupportedDBVersions	允许服务器使用不受支持的数据库版本。
BridgeAdapterURL	设置 Zabbix bridge adapter HTTP(S) URL。
CacheSize	设置配置缓存大小（以字节为单位）。
CacheUpdateFrequency	设置配置缓存更新频率（以秒为单位）。
DBHost	指定数据库主机（或套接字目录）。
DBName	指定数据库名称。必需。
DBPassword	指定数据库密码。
DBPort	指定数据库主机的端口。
DBSchema	指定数据库架构名称。用于 PostgreSQL。
DBSocket	指定 MySQL 套接字文件的路径。
DBUser	指定数据库用户。
DBTLSConnect	指定一个值以强制与数据库建立 TLS 连接。

参数	描述
DBTLSCAFile	指定包含用于数据库证书验证的顶级 CA 证书的文件的完整路径名。
DBTLSCertFile	指定包含用于向数据库进行身份验证的 Zabbix 服务器证书的文件的完整路径名。
DBTLSKeyFile	指定包含用于向数据库进行身份验证的私钥的文件的完整路径名。
DBTLSCipher	指定 Zabbix 服务器允许用于 TLS v1.2 及更低版本 TLS 协议的加密密码列表。仅 MySQL 支持。
DBTLSCipher13	指定 Zabbix 服务器允许用于 TLS v1.3 协议的加密密码套件列表。仅 MySQL 支持，且从 8.0.16 版本开始支持。
DebugLevel	设置调试级别。
EnableGlobalScripts	在 Zabbix 服务器上启用全局脚本。
EnableMobileDevices	在 Zabbix 服务器上启用移动设备支持。
ExportDir	指定目录以启用事件、历史数据和趋势数据的实时导出。
ExportFileSize	设置导出文件的最大大小（以字节为单位）。
ExportType	指定实时导出的实体类型（事件、历史数据、趋势数据）。
ExternalScripts	指定外部脚本位置。
Fping6Location	指定 fping6 的位置。
FpingLocation	指定 fping 的位置。
FrontendAllowedIP	指定允许从前端连接的 IP 地址或 CIDR 范围。
HANodeName	指定高可用集群中的节点名称。
HistoryCacheSize	设置历史数据缓存大小。
HistoryIndexCacheSize	设置历史数据索引缓存大小。
HistoryProvider	设置历史数据存储提供程序（Elasticsearch、ClickHouse 或 SQL）。
HistoryStorageDateIndex	此参数已弃用（请改用 HistoryProvider）。 启用根据日期将历史数据值存储在不同索引中。
HistoryStorageURL	此参数已弃用（请改用 HistoryProvider）。 指定历史数据存储 URL。
HistoryStorageTypes	此参数已弃用（请改用 HistoryProvider）。 指定要发送到历史数据存储的值类型。
HousekeepingFrequency	设置管家程序运行频率（以小时为单位）。
Include	指定要包含在配置文件中的目录或单个文件。
JavaGateway	指定 Zabbix Java 网关 IP 地址（或主机名）。
JavaGatewayPort	指定 Zabbix Java 网关端口。
ListenBacklog	设置 TCP 队列中待处理连接的最大数量。
ListenIP	指定 trapper 监听 IP 地址。
ListenPort	指定 trapper 监听端口。
LoadModule	指定服务器启动时要加载的模块。
LoadModulePath	指定服务器模块位置的完整路径。
LogFile	指定服务器日志文件。
LogFileSize	设置日志文件的最大大小。
LogSlowQueries	设置数据库查询在被记录前允许执行的最长时间（以毫秒为单位）。
LogType	指定日志输出类型。
MaxConcurrentChecksPerPoller	设置每个 HTTP agent poller、agent poller 或 SNMP poller 可同时执行的异步监控项检查的最大数量。
MaxHousekeeperDelete	设置一个管家周期内每个任务可删除的最大行数。
NodeAddress	指定 IP 或主机名以及可选端口，以覆盖前端连接服务器的方式。
PidFile	指定 PID 文件。
ProblemHousekeepingFrequency	设置删除已删除触发器所产生问题的频率。
ProxyConfigFrequency	设置服务器向 Zabbix proxy 发送配置数据的频率。
ProxyDataFrequency	设置服务器从 Zabbix proxy 请求历史数据的频率。
ServiceManagerSyncFrequency	设置同步服务管理器配置的频率。
SMSDevices	指定允许 Zabbix 服务器使用的调制解调器文件。
SNMPTrapperFile	指定用于将数据从 SNMP trap 守护进程传递到服务器的临时文件。
SocketDir	指定用于存储 Zabbix 内部服务所使用 IPC 套接字的目录。
SourceIP	指定源 IP 地址。
SSHKeyLocation	指定 SSH 检查和操作所使用的公钥和私钥的位置。
SSLCertLocation	指定用于客户端身份验证的 SSL 客户端证书文件的位置。
SSLKeyLocation	指定用于客户端身份验证的 SSL 私钥文件的位置。
SSLCALocation	指定用于 SSL 服务器证书验证的证书颁发机构（CA）文件的位置。
StartAgentPollers	设置预先派生的异步 Zabbix agent poller 实例数量。
StartAlerters	设置预先派生的告警器实例数量。
StartBrowserPollers	设置预先派生的浏览器监控项 poller 实例数量。
StartConnectors	设置预先派生的连接器工作进程实例数量。
StartDBSyncers	设置预先派生的历史数据同步器实例数量。
StartDiscoverers	设置预先派生的发现工作进程实例数量。

参数	描述
StartEscalators	设置预先派生的升级器实例数量。
StartHistoryPollers	设置预先派生的历史数据 poller 实例数量。
StartHTTPAgentPollers	设置预先派生的异步 HTTP agent poller 实例数量。
StartHTTPPollers	设置预先派生的 HTTP poller 实例数量。
StartIPMIPollers	设置预先派生的 IPMI poller 实例数量。
StartJavaPollers	设置预先派生的 Java poller 实例数量。
StartLLDProcessors	设置预先派生的低级别发现（LLD）工作进程实例数量。
StartODBCPollers	设置预先派生的 ODBC poller 实例数量。
StartPingers	设置预先派生的 ICMP pinger 实例数量。
StartPollersUnreachable	设置预先派生的不可达主机 poller 实例数量（包括 IPMI 和 Java）。
StartPollers	设置预先派生的 poller 实例数量。
StartPreprocessors	设置预先启动的预处理工作进程实例数量。
StartProxyPollers	设置预先派生的被动 proxy poller 实例数量。
StartReportWriters	设置预先派生的报告编写器实例数量。
StartSNMPPollers	设置预先派生的异步 SNMP poller 实例数量。
StartSNMPTrapper	设置为”1” 以启动 SNMP trapper 进程。
StartTelemetryQueryPollers	设置预先派生的异步 Telemetry 查询 poller 实例数量。
StartTimers	设置预先派生的计时器实例数量。
StartTrappers	设置预先派生的 trapper 实例数量。
StartVMwareCollectors	设置预先派生的 VMware collector 实例数量。
StatsAllowedIP	指定允许请求统计信息的外部 Zabbix 实例的 IP 地址或 DNS 名称。
TelemetryProvider	设置遥测数据源。
Timeout	指定与 Zabbix proxy、agent、Web 服务建立连接并交换数据，以及执行 SNMP 检查时的等待时间（以秒为单位；SNMP `walk[OID]` 和 `get[OID]` 监控项除外）。
TLSCAFile	指定包含顶级 CA 证书的文件的完整路径名，用于对等证书验证以及 Zabbix 组件之间的加密通信。
TLSCertFile	指定包含服务器证书或证书链的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSCipherAll	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherAll13	指定 TLS 1.3 中 OpenSSL 1.1.1 或更高版本使用的密码字符串。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherCert	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherCert13	指定 TLS 1.3 中 OpenSSL 1.1.1 或更高版本使用的密码字符串。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherPSK	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherPSK13	指定 TLS 1.3 中 OpenSSL 1.1.1 或更高版本使用的密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。
TLSCRLFile	指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。
TLSFrontendAccept	指定来自前端的传入连接的加密级别。
TLSFrontendCertIssuer	指定允许的前端证书颁发者。
TLSFrontendCertSubject	指定允许的前端证书主题。
TLSKeyFile	指定包含服务器私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSListen	要求 trapper 套接字仅接受 TLS 连接。
TmpDir	指定临时目录。
TrapperTimeout	指定 trapper 超时时间（以秒为单位），适用于： - 从 Zabbix proxy 获取历史数据； - 向 Zabbix proxy 发送配置数据； - 在 Zabbix 服务器上执行全局脚本或远程命令。
TrendCacheSize	设置趋势数据缓存大小。
TrendFunctionCacheSize	设置趋势函数缓存大小。
UnavailableDelay	设置在不可用期间检查主机可用性的频率。
UnreachableDelay	设置在不可达期间检查主机可用性的频率。
UnreachablePeriod	设置主机在不可达多少秒后被视为不可用。
User	将权限降级为系统中指定的现有用户。
ValueCacheSize	设置历史数据值缓存大小。
Vault	指定 vault 提供程序。
VaultAppRoleID	指定 HashiCorp vault AppRole ID。
VaultAppSecretID	指定 HashiCorp vault AppRole secret ID。
VaultDBPath	指定按键检索数据库凭据的位置。
VaultPrefix	指定 vault 路径或查询的自定义前缀。

参数	描述
VaultTLSCertFile	指定用于客户端身份验证的 SSL 证书文件名称。
VaultTLSKeyFile	指定用于客户端身份验证的 SSL 私钥文件名称。
VaultToken	指定 HashiCorp vault 身份验证令牌。
VaultURL	指定 vault 服务器 URL。
VMwareCacheSize	设置用于存储 VMware 数据的共享内存大小。
VMwareFrequency	设置从单个 VMware 服务收集数据的频率（以秒为单位）。
VMwarePerfFrequency	设置从单个 VMware 服务获取性能计数器统计信息的频率（以秒为单位）。
VMwareTimeout	设置 `vmware collector` 等待 VMware 服务响应的最大秒数。
WebDriverURL	指定 WebDriver 接口 URL。
WebServiceURL	以 `http[s]://host:port/report` 格式指定 Web 服务 URL。

请注意：

- 默认值反映守护进程默认值，而不是随附配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持使用 UTF-8 编码且不带 BOM 的配置文件。
- 以 # 开头的注释仅支持出现在行首。

Warning:
进程过多可能会耗尽预处理管理器的每进程文件描述符限制。

此类进程包括由 poller、不可达 poller、ODBC poller、HTTP poller、Java poller、pinger、trapper 和 proxypoller 执行的数据采集进程，以及 IPMI 管理器、SNMP trapper、预处理工作进程和发现工作进程。

耗尽文件描述符限制将导致 Zabbix 服务器停止，通常发生在启动后不久，但有时可能需要更长时间。

为避免此类问题，请查看Zabbix 服务器配置文件，以优化并发检查和进程数量。此外，如有必要，请检查并调整系统限制，确保文件描述符限制设置得足够高。

参数详情

AlertScriptsPath

指定alert script 的位置。取决于编译时安装变量 `datadir`。

默认值：`/usr/local/share/zabbix/alertscripts`

AllowRoot

允许服务器以 `root` 身份运行。如果禁用该选项且服务器由 `root` 启动，服务器将尝试切换为 `zabbix` 用户。如果在普通用户下启动，则无效。

默认值：0
 取值：0 - 不允许；1 - 允许

AllowSoftwareUpdateCheck

允许 Zabbix 前端接收来自 `zabbix.com` 的软件更新信息。

默认值：1
 取值：0 - 不允许；1 - 允许

AllowUnsupportedDBVersions

允许服务器使用不受支持的数据库版本工作。

默认值: 0
 取值: 0 - 不允许；1 - 允许

BridgeAdapterURL

设置Zabbix bridge adapter 的 HTTP(S) URL。

示例：

```
BridgeAdapterURL=https://bridge-adapter.example.com:10054/rpc
```

CacheSize

设置配置缓存大小（以字节为单位）。用于存储主机、监控项和触发器数据的共享内存大小。

默认值：32M
 范围：128K-64G

CacheUpdateFrequency

设置配置缓存更新频率（以秒为单位）。另请参见runtime control 选项。

默认值：10
 范围：1-3600

DBHost

指定数据库主机（或套接字目录）。

对于 MySQL：

- localhost 或空字符串将使用默认的 UNIX 域套接字。

对于 PostgreSQL：

- localhost 通过 DNS 解析（通常解析为 127.0.0.1）。
- 空字符串将使用默认的 UNIX 域套接字。
- 路径（例如 /var/run/pgbouncer）将使用该路径上的 UNIX 套接字。
- 以逗号分隔的列表可以包含多个 host:port 值。将按顺序尝试每个主机，直到建立读写连接。

示例：DBHost=localhost:5431,127.0.0.1:20051,zabbix.domain,[::1]:30051,[12fc::1]

默认值：localhost

DBName

指定数据库名称。

必填：是

DBPassword

指定数据库密码。如果未使用密码，请注释掉这一行。

DBPort

指定 Zabbix 数据库主机的端口（参见DBHost）。
 DBSocket 和 DBPort 在服务器配置中互斥。请只指定其中一个，或将两者都留空。

MySQL 的默认值：3306

PostgreSQL 的默认值：5432

范围：1024-65535

DBSchema

指定数据库模式名称。用于 PostgreSQL。

DBSocket

指定 MySQL socket 文件的路径。
 DBSocket 和 DBPort 在服务器配置中互斥。请仅指定其中一个，或者将两者都留空。

DBUser

指定数据库用户。

DBTLSConnect

指定一个值以强制与数据库建立 TLS 连接：- *required* - 使用 TLS 连接；- *verify_ca* - 使用 TLS 连接并验证证书；- *verify_full* - 使用 TLS 连接，验证证书，并验证 DBHost 指定的数据库标识是否与其证书匹配。

对于 MySQL，从 5.7.11 开始，以及 PostgreSQL，支持以下值：required、verify_ca、verify_full。
 对于 MariaDB，从 10.2.6 版本开始，支持 required 和 verify_full 值。
 默认情况下不设置任何选项，具体行为取决于数据库配置。

DBTLSCAFile

指定包含用于数据库证书验证的顶级 CA 证书的文件的完整路径名。

必填：否（如果 DBTLSConnect 设置为 verify_ca 或 verify_full，则为是）

DBTLSCertFile

指定包含用于对数据库进行身份验证的 Zabbix 服务器证书的文件的完整路径名。

DBTLSKeyFile

指定包含用于数据库身份验证的私钥文件的完整路径名。

DBTLSCipher

指定 Zabbix 服务器允许用于 TLS 协议（最高至 TLS v1.2）的加密套件列表。仅支持 MySQL。

DBTLSCipher13

指定 Zabbix 服务器允许用于 TLS v1.3 协议的加密套件列表。仅支持 MySQL，自 8.0.16 版本起可用。

DebugLevel

设置调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。
 另请参见运行时控制 选项。

默认值：3
 范围：0-5

EnableGlobalScripts

在 Zabbix 服务器上启用全局脚本。
 注意：全局脚本执行默认已启用。但是，对于从 Zabbix 7.0 开始的新安装，EnableGlobalScripts 会显式设置为 0（禁用）。

默认值：1
 取值：0 - 禁用；1 - 启用

EnableMobileDevices

在 Zabbix 服务器上启用移动设备支持。

启用后，Zabbix 服务器将处理添加或移除移动设备的请求，并向这些设备发送推送通知。

默认值：1
 值：0 - 禁用；1 - 启用

ExportDir

指定一个目录以启用事件、历史数据和趋势的实时导出。

ExportFileSize

设置最大导出文件大小（以字节为单位）。当设置了 `ExportDir` 时，用于轮转。

默认值：1G
 范围：1M-1G

ExportType

为实时导出指定实体类型（事件、历史、趋势）。使用逗号分隔的列表。默认导出所有类型。仅当设置了 `ExportDir` 时才有效。
 注意，如果指定了 `ExportType`，但未指定 `ExportDir`，则这是一个配置错误，服务器将无法启动。

历史和趋势导出的示例：

```
ExportType=history,trends
```

仅导出事件的示例：

```
ExportType=events
```

外部脚本

指定外部脚本的位置。取决于编译时安装变量 `datadir`。

默认值：`/usr/local/share/zabbix/externalscripts`

Fping6Location

指定 fping6 的位置。请确保 fping6 二进制文件的所有者为 root，并已设置 SUID 标志。如果您的 fping 工具能够处理 IPv6 地址，请留空（`Fping6Location=`）。

默认值：`/usr/sbin/fping6`

FpingLocation

指定 fping 的位置。请确保 fping 二进制文件的所有者为 root，并已设置 SUID 标志。

默认值：`/usr/sbin/fping`

FrontendAllowedIP

指定允许从前端连接的 IP 地址或 CIDR 范围。使用逗号分隔的列表。默认情况下，前端请求接受所有连接。如果启用了 IPv6 支持，则 `127.0.0.1`、`::127.0.0.1`、`::ffff:127.0.0.1` 会被视为相同，且 `::/0` 将允许任何 IPv4 或 IPv6 地址。`0.0.0.0/0` 可用于允许任何 IPv4 地址。

示例：

```
FrontendAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

HANodeName

指定高可用集群中的节点名称。当该值为空时，服务器以独立模式运行，并会创建一个名称为空的节点。

HistoryCacheSize

设置历史缓存大小（以字节为单位）。用于存储历史数据的共享内存大小。

默认值：16M
 范围：128K-16G

HistoryIndexCacheSize

设置 history-index 缓存大小（以字节为单位）。用于为存储在 history cache 中的历史数据建立索引的共享内存大小。索引缓存大小大约需要 100 字节来缓存一个监控项。

默认值：4M
 范围：128K-16G

HistoryProvider

使用 `<provider>[;<options>]` 格式设置历史数据存储 provider，其中：

- `<provider>` - 支持的历史数据存储 provider 之一：`sql`（默认）、`elasticsearch` 或 `clickhouse`。
- `<options>` - 以逗号分隔的 `key=value` 对。值可以加引号。

所有历史数据存储 provider 的选项：

- `value_types` - 设置要存储的值类型（`uint`、`dbl`、`str`、`log`、`text`、`json`）。每种类型只能使用一次，并且只能用于单个 provider。
- `precache` - 设置为 1（默认），以便将历史数据存储 provider 中的数据批量预加载到值缓存中。这可以显著改善重启后的值缓存预热速度。

Elasticsearch 和ClickHouse 的选项：

- `url` - 设置用于连接历史数据存储 provider 的 HTTP 或 HTTPS URL。
- `log_slow_queries` - 设置历史数据存储 provider 查询在被记录之前允许的最大持续时间（毫秒）。此选项从 `DebugLevel=3` 开始启用。
- `source_ip` - 设置用于向历史数据存储 provider 发起出站连接的源 IP 地址。

Elasticsearch 的选项：

- `date_index` - 设置为 1 以启用按日期将历史值存储到不同索引中。

ClickHouse 的选项：

- `db` - 设置数据库名称。
- `username` - 设置用于登录数据库的用户名。
- `password` - 设置用于登录数据库的密码。
- `ssl_cert_file`、`ssl_key_file`、`ssl_key_password`、`ssl_verify_peer`、`ssl_verify_host`、`ssl_ca_location`、`ssl_cert_location`、`ssl_key_location` - 设置这些选项以配置到数据库的安全（TLS/SSL）连接。

ClickHouse 示例：

```
HistoryProvider=clickhouse;value_types="uint,dbl,str,log,text,json",url=http://localhost:8123,db=zabbix,us
```

Elasticsearch 示例：

```
HistoryProvider=elasticsearch;value_types="uint,dbl,str,log,text,json",url=http://localhost:9200,date_inde
```

ClickHouse 和 Elasticsearch 示例：

```
HistoryProvider=clickhouse;value_types="uint,dbl,str",url=http://localhost:8123,db=zabbix,username=zabbix,
HistoryProvider=elasticsearch;value_types="log,text,json",url=http://localhost:9200,date_index=0
```

HistoryStorageDateIndex

此参数已弃用（请改用HistoryProvider）。
 启用根据日期将历史值存储到不同索引中。此参数用于Elasticsearch 配置。

默认值：0
 取值：0 - 禁用；1 - 启用

HistoryStorageURL

此参数已弃用（请改用HistoryProvider）。
 指定历史数据存储 URL。此参数用于Elasticsearch 的设置。

HistoryStorageTypes

此参数已弃用（请改用HistoryProvider）。
 指定要发送到历史存储的值类型。使用逗号分隔的列表。此参数用于Elasticsearch 配置。

默认值：`uint,dbl,str,log,text,json`

HousekeepingFrequency

设置 housekeeping 过程的频率（以小时为单位）。Housekeeping 是从数据库中删除过期信息。
 注意：为防止 housekeeper 负载过高（例如，当历史和趋势保留周期大幅缩短时），在每个 housekeeping 周期中，每个监控项最多只会删除 HousekeepingFrequency 小时的 4 倍过期信息。因此，如果 HousekeepingFrequency 为 1，则每个周期最多只会删除 4 小时的过期信息（从最旧的记录开始）。
 注意：为降低服务器启动时的负载，housekeeping 会在服务器启动后延迟 30 分钟执行。因此，如果 HousekeepingFrequency 为 1，服务器启动后的第一次 housekeeping 过程将在 30 分钟后运行，之后将以 1 小时的间隔重复执行。
 可以通过将 HousekeepingFrequency 设置为 0 来禁用自动 housekeeping。在这种情况下，housekeeping 过程只能通过 housekeeper_execute 运行时控制选项启动，并且在一个 housekeeping 周期中删除的过期信息时长为自上次 housekeeping 周期以来时间的 4 倍，但不少于 4 小时且不超过 4 天。
 另请参见运行时控制 选项以及housekeeping 过程 的详细说明。

默认值：1
 范围：0-24

Include

指定要包含到配置文件中的目录或单个文件。
 在 Zabbix 安装期间，除非你在编译时更改此参数，否则该参数会被设置为 /usr/local/etc。

Include 参数的一般规则：

- 路径可以是相对于 zabbix_server.conf 文件位置的相对路径。
- 你可以在一个配置文件中指定多个 Include 参数。
- Zabbix 会在读取到 Include 参数后立即处理它，因此建议将其放在配置文件末尾。

包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，该组件将无法启动。

包含目录或文件模式的规则：

- 不要假定包含顺序有任何规律（例如，文件不会按字母顺序包含）。
- 不要在多个已包含文件中定义同一个参数（例如，用特定设置覆盖通用设置），因为你无法预测最终会使用哪个值。
- 注意某些文本编辑器会自动创建的备份副本(例如，编辑 include/my_specific.conf 可能会创建 include/my_specific.conf.BAK)，因为这两个文件都会被包含。

若只想从目录中包含相关文件，请使用通配符（*）进行模式匹配。

示例：

```
Include=/absolute/path/to/config/files/*.conf
```

JavaGateway

指定 Zabbix Java 网关的 IP 地址或主机名。仅在启动 Java poller 时才需要。

JavaGatewayPort

指定 Zabbix Java 网关端口。

默认值：10052
 范围：1024-32767

ListenBacklog

设置 TCP 队列中待处理连接的最大数量。
 默认值是一个硬编码常量，具体取决于系统。
 支持的最大值也取决于系统，过高的值可能会被静默截断为 “implementation-specified maximum”。

默认值: SOMAXCONN
 范围: 0 - INT_MAX

ListenIP

指定 trapper 监听的 IP 地址。以逗号分隔的列表。
 如果未指定此参数，trapper 将监听所有网络接口。

默认值：0.0.0.0

ListenPort

指定 trapper 监听端口。

默认值：10051
 范围：1024-32767

LoadModule

指定服务器启动时要加载的模块。模块用于扩展服务器的功能。模块必须位于 LoadModulePath 指定的目录中，或者在模块名称前指定路径。如果前置路径为绝对路径（以 / 开头），则会忽略 LoadModulePath。
 格式：

- LoadModule=<module.so>
- LoadModule=<path/module.so>

- LoadModule=</abs_path/module.so>

可以指定多个 LoadModule 参数。

LoadModulePath

指定服务器模块所在位置的完整路径。默认值取决于编译选项。

LogFile

指定服务器日志文件。

必填：是，如果 LogType 设置为 file；否则为否

LogFileSize

设置最大日志文件大小（单位：MB）。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始写入。

默认值：1
 范围：0-1024
 必填：是，若 LogType 设置为 file；否则不是

LogSlowQueries

设置数据库查询在被记录之前允许的最长执行时间（以毫秒为单位）。
0 - 不记录慢查询。
 此选项从 DebugLevel=3 开始启用。

默认值: 0
 范围: 0-3600000

LogType

指定日志输出类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *system* - 将日志写入 syslog；- *console* - 将日志写入标准输出。

默认值：file

MaxConcurrentChecksPerPoller

设置每个 HTTP agent poller、agent poller 或 SNMP poller 一次可执行的异步检查的最大数量。请参见StartHTTPAgentPollers、StartAgentPollers 和StartSNMPPollers。

默认值：1000
 范围：1-1000

MaxHousekeeperDelete

设置在一次 housekeeping 周期中，每个任务从 history、history_str、history_log、history_uint、history_text、history_bin、history_json、trends、trends_uint 和 problem 表中删除的最大行数。
housekeeping 任务存在于 housekeeper 表中，格式为：[housekeeperid]、[object]、[objectid]。
 此参数仅适用于删除已被删除的监控项遗留的数据。
 如果设置为 0，则完全不使用限制。
 在这种情况下，您必须清楚自己在做什么，以免数据库过载。
 当删除大量监控项时，会增加数据库负载，因为 housekeeper 需要移除这些监控项曾经产生的所有历史数据。
 例如，如果我们只需要从模板中删除 1 个监控项原型，但该模板关联了 50 个主机，并且每个主机上的原型都会展开为 100 个实际监控项，那么总共需要删除 5000 个监控项（1*50*100）。
 如果为 MaxHousekeeperDelete 设置 500（MaxHousekeeperDelete=500），则 housekeeper 进程在一个周期内需要从 history 和 trends 表中为已删除的监控项最多删除 2500000 个值（5000*500）。

另请参见housekeeping procedure 的详细信息。

默认值：5000
 范围：0-1000000

NodeAddress

指定带可选端口的 IP 或主机名，以覆盖前端连接到服务器的方式。
 格式：<address>[:<port>]

 如果未设置 IP 或主机名，则将使用 ListenIP 的值。如果未设置 ListenIP，则将使用 localhost 的值。
 如果未设置端口，则将使用 ListenPort 的值。如果未设置 ListenPort，则将使用 10051 的值。

 此选项可被前端配置中指定的地址覆盖。

 另请参见：HANodeName 参数；启用高可用性。

默认值：localhost:10051

PidFile

指定 PID 文件。

默认值：/tmp/zabbix_server.pid

ProblemHousekeepingFrequency

设置删除已删除触发器的问题的频率（以秒为单位）。
 另请参阅housekeeping procedure 的详细信息。

默认值：60
 范围：1-3600

ProxyConfigFrequency

设置服务器向 Zabbix proxy 发送配置数据的频率。仅用于被动模式下的 proxy。

默认值：10
 范围：1-604800

ProxyDataFrequency

设置服务器从 Zabbix proxy 请求历史数据的频率。仅用于被动模式下的 proxy。

默认值：1
 范围：1-3600

ServiceManagerSyncFrequency

设置服务管理器配置同步的频率（以秒为单位）。

默认值：60
 范围：1-3600

SMSDevices

指定允许 Zabbix 服务器使用的调制解调器文件。以逗号分隔的列表。
 如果未设置此参数，则无法发送 SMS。

示例：

```
SMSDevices=/dev/ttyUSB0,/dev/ttyUSB1
```

SNMPTrapperFile

指定用于将数据从 SNMP trap 守护进程传递到服务器的临时文件。
 必须与 `zabbix_trap_receiver.pl` 或 SNMPTT 配置文件中的设置相同。

默认值：`/tmp/zabbix_traps.tmp`

SocketDir

指定用于存储内部 Zabbix 服务所使用的 IPC 套接字的目录。

默认值：`/tmp`

SourceIP

为以下连接指定源 IP 地址：

- 到 Zabbix proxy 和 Zabbix agent 的出站连接
- 无 agent 连接（VMware、SSH、JMX、SNMP、Telnet 和简单检查）
- HTTP agent 连接
- 脚本监控项的 JavaScript HTTP 请求
- 预处理 JavaScript HTTP 请求
- 发送通知邮件（连接到 SMTP 服务器）
- webhook 通知（JavaScript HTTP 连接）
- 连接到 Vault

SSHKeyLocation

指定用于 SSH 检查和操作的公钥和私钥位置。

SSLCertLocation

指定用于客户端身份验证的 SSL 客户端证书文件位置。
 此参数用于 Web 监控、与 Vault 通信、历史数据存储提供程序 ClickHouse，以及遥测数据源 ClickHouse。

SSLKeyLocation

指定用于客户端身份验证的 SSL 私钥文件位置。
 此参数用于 Web 监控、与 Vault 通信、历史数据存储提供程序 ClickHouse，以及遥测数据源 ClickHouse。

SSLCALocation

指定用于验证 SSL 服务器证书的证书颁发机构（CA）文件位置。如果未设置，则使用系统范围的目录。
 请注意，此参数的值将设置为 libcurl 选项 `CURLOPT_CAPATH`。对于 7.42.0 之前的 libcurl 版本，只有在 libcurl 编译为使用 OpenSSL 时，此参数才会生效。有关更多信息，请参阅 cURL 网页。
 此参数用于 Web 监控、SMTP 身份验证、HTTP agent 监控项、与 Vault 的通信、历史数据存储提供程序 ClickHouse，以及遥测数据源 ClickHouse。

StartAgentPollers

设置 Zabbix agent pollers 的预 fork 实例数量。请参见MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartAlerters

设置alerters 的预 fork 实例数量。

默认值：3
 范围：1-100

StartBrowserPollers

设置 browser 监控项pollers 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartConnectors

设置connector workers 的预 fork 实例数量。当启动 connector worker 时，connector manager 进程会自动启动。

默认值：0
 范围：0-1000

StartDBSyncers

设置history syncers 的预 fork 实例数量。
 注意：更改此值时请谨慎，增加它可能弊大于利。大致来说，默认值应足以处理最多 4000 NVPS。

默认值：4
 范围：1-100

StartDiscoverers

设置discovery workers[1] 的预 fork 实例数量。

默认值：5
 范围：0-1000

StartEscalators

设置escalators 的预 fork 实例数量。

默认值：1
 范围：1-100

StartHistoryPollers

设置history pollers 的预 fork 实例数量。
 仅在计算型检查中需要。

默认值：5
 范围：0-1000

StartHTTPAgentPollers

设置 HTTP agent pollers 的预 fork 实例数量。请参见MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartHTTPPollers

设置HTTP pollers[1] 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartIPMIPollers

设置IPMI pollers 的预 fork 实例数量。

默认值：0
 范围：0-1000

StartJavaPollers

设置Java pollers[1] 的预 fork 实例数量。

默认值：0
 范围：0-1000

StartLLDProcessors

设置低级别发现（LLD）worker[1] 的预 fork 实例数量。
 当启动 LLD worker 时，LLD 管理器进程会自动启动。

默认值：2
 范围：1-100

StartODBCPollers

设置ODBC pollers[1] 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartPingers

设置ICMP pingers[1] 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartPollersUnreachable

设置用于不可达主机的 poller 的预 fork 实例数量（包括 IPMI 和 Java）[1]。
 如果已启动常规 poller、IPMI poller 或 Java poller，则必须至少运行一个用于不可达主机的 poller。

默认值：1
 范围：0-1000

StartPollers

设置pollers[1] 的预 fork 实例数量。

默认值：5
 范围：0-1000

StartPreprocessors

设置预启动的预处理worker[1] 线程数量，应不少于可用的 CPU 核心数。如果预处理不受 CPU 限制且包含大量网络请求，则应设置更多 worker。

默认值：16
 范围：1-1000

StartProxyPollers

设置用于被动 proxy 的 poller[1] 的预 fork 实例数量。

默认值：1
 范围：0-250

StartReportWriters

设置report writers 的预 fork 实例数量。
 如果设置为 0，则会禁用计划报告生成。
 当启动 report writer 时，report manager 进程会自动启动。

默认值：0
 范围：0-100

StartSNMPPollers

设置 SNMP pollers 的预 fork 实例数量。参见`MaxConcurrentChecksPerPoller`。

默认值：1
 范围：0-1000

StartSNMPTrapper

设置为 1 以启动SNMP trapper 进程。

默认值：0
 范围：0-1

StartTelemetryQueryPollers

设置异步 Telemetry 查询poller 的预派生实例数。请参阅MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartTimers

设置timers 的预 fork 实例数量。
Timers 进程处理维护周期。

默认值：1
 范围：1-1000

StartTrappers

设置trappers[1] 的预 fork 实例数量。
 Trappers 接受来自 Zabbix sender、active agent 和 active proxy 的传入连接。

默认值：5
 范围：0-1000

StartVMwareCollectors

设置预先 fork 的VMware collector 实例数量。

默认值：0
 范围：0-250

StatsAllowedIP

指定允许请求统计信息的外部 Zabbix 实例的 IP 地址（可选使用 CIDR 表示法）或 DNS 名称。以逗号分隔的列表。
 如果未设置此参数，将不接受任何统计信息请求。
 如果启用了 IPv6 支持，则 `127.0.0.1`、`::127.0.0.1`、`::ffff:127.0.0.1` 会被视为相同，且 `::/0` 将允许任何 IPv4 或 IPv6 地址。`0.0.0.0/0` 可用于允许任何 IPv4 地址。

示例：

```
StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

TelemetryProvider

使用 `<provider>[;<options>]` 格式设置遥测数据源，其中：

- `<provider>` - 受支持的遥测数据源之一：`clickhouse`

- <options> - 以逗号分隔的 key=value 对。值可以使用引号括起来。

ClickHouse 的选项：

- url - 设置用于连接遥测数据源的 HTTP 或 HTTPS URL。
- db - 设置数据库名称。
- username - 设置用于登录数据库的用户名。
- password - 设置用于登录数据库的密码。
- ssl_cert_file、ssl_key_file、ssl_key_password、ssl_verify_peer、ssl_verify_host、ssl_ca_location、ssl_cert_location、ssl_key_location - 设置这些选项以配置与数据库的安全（TLS/SSL）连接。
- source_ip - 设置用于连接遥测数据源的出站连接源 IP 地址。
- vault_path - 设置保存 username 和 password 值的 Vault 密钥路径。请勿与 username 和 password 选项一起使用。

ClickHouse 示例：

```
TelemetryProvider=clickhouse;url=http://localhost:8123,db=zabbix,username=zabbix,password=zabbix
```

使用 Vault 的 ClickHouse 示例：

```
TelemetryProvider=clickhouse;url=http://localhost:8123,db=zabbix,vault_path=clickhouse
```

超时

指定与 Zabbix proxy、agent、web service 以及旧版 SNMP 检查（单个 OID 编号或字符串）建立连接并交换数据时的等待时间（秒）。

此参数定义了多种通信操作的持续时间：

- 在 Zabbix agent 上执行远程命令
- 执行 SSH/Telnet 命令
- 向 Zabbix web service 发起请求
- mediatype 测试请求以及 history.push 方法的通信超时
- 当 IPMI 接口变为不可用时重新调度监控项
- 因权限或加密问题导致数据交换失败时，向 Zabbix proxy 发送响应
- IPC 异步套接字和运行时控制选项的截止时间
- JMX 连接
- 从远程 Zabbix proxy 或服务器获取统计信息
- 向 Zabbix 前端发送响应
- 异步 poller 的 DNS 请求
- active check 心跳的响应
- 从 active agent 获取 Zabbix agent 数据（值）
- 从 Zabbix sender 获取数据
- trapper 无法处理请求时的响应
- 向 Zabbix agent 发送 active check 列表

对于在前端中配置了flexible timeout 设置的检查（在全局、proxy 或单个监控项级别），不会使用此超时。例如，SNMP walk[OID] 和 get[OID] 监控项使用前端中配置的超时；旧版 SNMP 检查仍使用服务器超时值。

默认值：3
 范围：1-30

TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件完整路径名，该证书用于 Zabbix 组件之间的加密通信。

TLSCertFile

指定包含服务器证书或证书链的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSCipherAll

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书和 PSK 的加密的默认密码套件选择条件。

示例：

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherAll13

为 OpenSSL 1.1.1 或更新版本在 TLS 1.3 中指定密码字符串。覆盖基于证书和基于 PSK 的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL
```

OpenSSL 示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherCert

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码套件字符串。覆盖基于证书的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

OpenSSL 示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

TLSCipherCert13

为 TLS 1.3 指定适用于 OpenSSL 1.1.1 或更新版本的密码字符串。覆盖基于证书的加密的默认密码套件选择标准。

TLSCipherPSK

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码套件字符串。
覆盖基于 PSK 的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

OpenSSL 示例：

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

为 TLS 1.3 中的 OpenSSL 1.1.1 或更新版本指定密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。

示例：

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCRLFile

指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。

TLSFrontendAccept

指定来自前端的入站连接的加密级别。

可以指定多个值，并用逗号分隔：

- unencrypted - 接受未加密的连接。
- cert - 接受使用 TLS 和证书保护的连接。

默认值：`unencrypted`

TLSFrontendCertIssuer

指定允许的前端证书颁发者。

TLSFrontendCertSubject

指定允许的前端证书主题。

TLSKeyFile

指定包含服务器私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSListen

在 trapper 套接字上仅允许 TLS 连接。

支持的值：

- required - 仅接受 TLS 连接

TmpDir

指定临时目录。

默认值：`/tmp`

TrapperTimeout

为以下操作指定 trapper 超时时间（秒）：

- 从 Zabbix proxy 检索历史数据
- 向 Zabbix proxy 发送配置数据
- 在 Zabbix 服务器上执行全局脚本或远程命令

默认值：300
 范围：1-300

TrendCacheSize

设置趋势缓存大小，单位为字节。
 用于存储趋势数据的共享内存大小。

默认值：4M
 范围：128K-16G

TrendFunctionCacheSize

设置趋势函数缓存大小，单位为字节。
 用于缓存已计算趋势函数数据的共享内存大小。

默认值：4M
 范围：128K-2G

UnavailableDelay

设置在不可用 期间检查主机可用性的频率，单位为秒。

默认值：60
 范围：1-3600

UnreachableDelay

设置在unreachability 期间检查主机可用性的频率，单位为秒。

默认值：15
 范围：1-3600

UnreachablePeriod

设置在多少秒的不可达后将主机视为不可用。

默认值：45
 范围：1-3600

用户

将权限降级为系统中某个特定的、已存在的用户。
 仅在以'root' 运行且 AllowRoot 被禁用时生效。

默认值：zabbix

ValueCacheSize

设置历史值缓存的大小（以字节为单位）。
 用于缓存监控项历史数据请求的共享内存大小。
 设置为 0 将禁用值缓存（不推荐）。
 当值缓存耗尽共享内存时，服务器日志每 5 分钟记录一条警告消息。

默认值：8M
 范围：0、128K-64G

Vault

指定 vault 提供程序：- *HashiCorp* - HashiCorp KV Secrets Engine 版本 2；- *CyberArk* - CyberArk Central Credential Provider
 必须与前端中设置的 vault 提供程序一致。

默认值：HashiCorp

VaultAppRoleID

指定 HashiCorp vault 的 AppRole ID。
 如果设置，将启用 AppRole 身份验证。不能与 VaultToken 一起使用。
 仅支持 HashiCorp vault。

VaultAppSecretID

指定 HashiCorp vault AppRole secret ID。
 仅支持 HashiCorp vault。

VaultDBPath

指定用于按键检索数据库凭据的位置。
它是一个 Vault 路径或查询，具体取决于所使用的 Vault。

HashiCorp 使用的键为 password 和 username。

带有 VaultPrefix=/v1/secret/data/zabbix/ 的路径示例：

```
database
```

不带 VaultPrefix 的路径示例：

```
secret/zabbix/database
```

CyberArk 使用的键为 Content 和 UserName。

示例：

```
AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_database
```

仅当未指定 DBUser 和 DBPassword 时，才能使用此选项。

VaultPrefix

为 vault 路径或查询指定自定义前缀，具体取决于 Vault。如果未指定，将使用最合适的默认值。
 请注意，如果未指定 VaultPrefix，对于 HashiCorp，data 会在挂载点之后自动追加。

HashiCorp 的前缀示例：

```
/v1/secret/data/zabbix/
```

Cyberark 的前缀示例：

```
/AIMWebService/api/Accounts?
```

VaultTLSCertFile

指定用于客户端身份验证的 SSL 证书文件名称。证书文件必须为 PEM1 格式。
 如果证书文件中也包含私钥，请将 SSL 密钥文件字段留空。
 包含此文件的目录由配置参数 SSLCertLocation 指定。
 此选项可以省略，但建议用于 CyberArkCCP vault。

VaultTLSKeyFile

指定用于客户端身份验证的 SSL 私钥文件名称。私钥文件必须采用 PEM1 格式。
 包含该文件的目录由配置参数 SSLKeyLocation 指定。
 此选项可以省略，但建议用于 CyberArkCCP vault。

VaultToken

指定 HashiCorp Vault 认证令牌。HashiCorp Vault 认证令牌应仅为 Zabbix 服务器生成，并且对Vault 宏 中指定的路径以及可选的 VaultDBPath 配置参数所指定的路径具有只读权限。
 如果同时定义了 VaultToken 和 VAULT_TOKEN 环境变量，则会报错。
 不能与 VaultAppRoleID 一起使用。
 仅适用于 HashiCorp Vault。

VaultURL

指定 vault 服务器 URL。如果未指定 SSLCALocation，则将使用系统范围的 CA 证书目录。

默认值：https://127.0.0.1:8200

VMwareCacheSize

设置用于存储 VMware 数据的共享内存大小。可使用 VMware 内部检查 zabbix[vmware,buffer,...] 来监控 VMware 缓存使用情况（参见内部检查）。
 请注意，如果未配置任何 vmware collector 实例启动，则不会分配共享内存。

默认值：8M
 范围：256K-2G

VMwareFrequency

设置从单个 VMware 服务收集数据的频率（以秒为单位）。此频率应设置为任何 VMware 监控项中最小的更新间隔。

默认值：60
 范围：10-86400

VMwarePerfFrequency

设置从单个 VMware 服务检索性能计数器统计信息的频率（以秒为单位）。
 此频率应设置为任何使用 VMware 性能计数器的 VMware 监控监控项 的最小更新间隔。

默认值：60
 范围：10-86400

VMwareTimeout

设置 vmware collector 等待来自 VMware 服务（vCenter 或 ESX hypervisor）响应的最长秒数。

默认值：10
 范围：1-300

WebDriverURL

指定 WebDriver 接口 URL。

示例（与 Selenium WebDriver standalone server 一起使用）：

```
WebDriverURL=http://localhost:4444
```

WebServiceURL

以 `http[s]://host:port/report` 的格式指定 Web 服务 URL。

示例：

```
WebServiceURL=http://localhost:10053/report
```

仅在非 TLS（HTTP）连接中可以省略 `http://` 协议；配置了 TLS 时，必须使用 `https://`。

脚注

[1] 请注意，过多的数据采集进程（pollers、unreachable pollers、ODBC pollers、HTTP pollers、Java pollers、pingers、trappers、proxypollers）以及 IPMI manager、SNMP trapper、preprocessing workers 和 discovery workers，可能会耗尽 preprocessing manager 的每进程文件描述符限制。

> **Warning:**
> 耗尽文件描述符限制会导致 Zabbix 服务器停止，通常会在启动后不久发生，但有时也可能需要更长时间。为避免此类问题，请查看Zabbix 服务器配置文件，以优化并发检查和进程数量。此外，如有必要，请通过检查和调整系统限制，确保文件描述符限制设置得足够高。

proxy

概述　Zabbix proxy 是一种进程，可以从一个或多个受监控设备收集监控数据，并将这些信息发送到 Zabbix 服务器，本质上代表服务器执行工作。所有收集的数据都会在本地进行缓冲，然后传输到该 proxy 所属的 Zabbix 服务器。

部署 proxy 是可选的，但对于分担单个 Zabbix 服务器的负载可能非常有益。如果仅由 proxy 收集数据，服务器上的处理将减少对 CPU 和磁盘 I/O 的占用。

Zabbix proxy 是集中监控远程位置、分支机构以及没有本地管理员的网络的理想解决方案。

Zabbix proxy 需要单独的数据库。

> **Attention:**
> 请注意，Zabbix proxy 支持的数据库包括 SQLite、MySQL 和 PostgreSQL。

另请参阅：

- 使用 proxy 进行分布式监控
- Proxy 负载均衡和高可用性
- 监控配置同步
- 服务器与 proxy 的数据交换协议
- 配置参数：Zabbix proxy
- 设置 Zabbix Cloud proxy

运行 **proxy**　如果作为软件包安装

Zabbix proxy 作为守护进程运行。可以通过执行以下命令启动 proxy：

```
systemctl start zabbix-proxy
```

这适用于大多数 GNU/Linux 系统。在其他系统上，您可能需要运行：

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

类似地，要停止/重启/查看 Zabbix proxy 的状态，请使用以下命令：

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

手动启动

如果上述方法不起作用，您需要手动启动它。找到 `zabbix_proxy` 二进制文件的路径并执行：

```
zabbix_proxy
```

您可以将以下命令行参数与 Zabbix proxy 一起使用：

Parameter	Description
`-c --config <config-file>`	配置文件的路径。
`-f --foreground`	以前台方式运行 Zabbix proxy。
`-R --runtime-control <option>`	执行管理功能。
`-T --test-config`	验证配置文件并退出。
`-h --help`	打印帮助信息并退出。
`-V --version`	显示版本号。

使用命令行参数运行 Zabbix proxy 的示例：

```
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf
zabbix_proxy --help
zabbix_proxy -V
```

运行时控制

运行时控制选项：

Option	Description	Target
`config_cache_reload`	重新加载配置缓存。如果缓存当前正在加载，则忽略。 活动的 Zabbix proxy 将连接到 Zabbix 服务器并请求配置数据。 被动的 Zabbix proxy 将在服务器下次连接到 proxy 时向 Zabbix 服务器请求配置数据。	
`history_cache_clear[<itemid>]`	清除由其 ID 指定的监控项的历史缓存。 影响该监控项的所有值，但不包括第一个和最后一个值。	**target** - 监控项的 ID。
`diaginfo[=<section>]`	在 proxy 日志文件中收集诊断信息。	`historycache` - 历史缓存统计信息； `preprocessing` - 预处理管理器统计信息； `locks` - mutex 列表（在 *BSD* 系统上为空）。
`snmp_cache_reload`	重新加载 SNMP 缓存 - 清除所有主机的 SNMP 引擎属性（engine time、engine boots、engine id、credentials）。用于在排查 SNMP 问题时强制全局清除缓存。	
`housekeeper_execute`	启动 housekeeping 过程。如果 housekeeping 过程当前正在进行，则忽略。	
`log_level_increase[=<target>]`	提高日志级别 - 如果未指定 target，则影响所有进程。 不支持 *BSD* 系统。	**process type** - 指定类型的所有进程（例如 `poller`）。 查看所有proxy 进程类型。 **process type,N** - 进程类型和编号（例如 `poller,3`） **pid** - 进程标识符（1 到 65535）。对于更大的值，请将 target 指定为'process type,N'。
`log_level_decrease[=<target>]`	降低日志级别 - 如果未指定 target，则影响所有进程。 不支持 *BSD* 系统。	
`prof_enable[=<target>]`	启用性能分析。 如果未指定 target，则影响所有进程。 启用性能分析后，会按函数名提供所有 rwlocks/mutex 的详细信息。	**process type** - 指定类型的所有进程（例如 `history syncer`）。 查看所有proxy 进程类型。 **process type,N** - 进程类型和编号（例如 `history syncer,1`）。 **pid** - 进程标识符（1 到 65535）。对于更大的值，请将 target 指定为'process type,N'。 **scope** - `rwlock`、`mutex`、`processing` 可与进程类型和编号一起使用（例如 `history syncer,1,processing`），也可与该类型的所有进程一起使用（例如 `history syncer,rwlock`）。

Option	Description	Target
prof_disable[=<target>]	禁用性能分析。 如果未指定 target，则影响所有进程。	**process type** - 指定类型的所有进程（例如 `history syncer`）。 查看所有proxy 进程类型。 **process type,N** - 进程类型和编号（例如 `history syncer,1`）。 **pid** - 进程标识符（1 到 65535）。对于更大的值，请将 target 指定为'process type,N'。

当将某个进程类型指定为运行时控制的 target 时，该命令仅适用于该类型的正在运行的进程。如果当前没有该类型的进程在运行，命令将返回错误：

```
zabbix_proxy -R log_level_increase='data sender'
Cannot redirect signal: "data sender" process does not exist
```

使用运行时控制重新加载 proxy 配置缓存的示例：

```
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R config_cache_reload
```

使用运行时控制清除某个监控项历史缓存的示例：

```
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R history_cache_clear=42243
```

使用运行时控制收集诊断信息的示例：

```
### 在 proxy 日志文件中收集所有可用的诊断信息：
zabbix_proxy -R diaginfo

### 在 proxy 日志文件中收集历史缓存统计信息：
zabbix_proxy -R diaginfo=historycache
```

使用运行时控制重新加载 SNMP 缓存的示例：

```
zabbix_proxy -R snmp_cache_reload
```

Attention:
当通过 Zabbix 前端更新 SNMPv3 接口时，在大多数情况下，Zabbix 会自动为该接口重新加载新的 SNMPv3 credentials；仅当更改 credentials 后轮询仍然失败时（例如由于 engineBoots/engineID 不一致或非 RFC 设备），或者当您需要强制全局清除 SNMP 缓存以进行故障排查时，才使用 `-R snmp_cache_reload`。

使用运行时控制触发执行 housekeeper 的示例：

```
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R housekeeper_execute
```

使用运行时控制更改日志级别的示例：

```
### 提高所有进程的日志级别：
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase

### 提高第二个 poller 进程的日志级别：
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=poller,2

### 提高 PID 为 1234 的进程的日志级别：
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=1234

### 降低所有 http poller 进程的日志级别：
zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_decrease="http poller"
```

运行用户

Zabbix proxy 设计为以非 root 用户运行。它将以启动时所使用的非 root 用户身份运行。因此，您可以让 proxy 以任意非 root 用户身份运行，不会有任何问题。

如果您尝试以 "root" 身份运行它，它将切换为硬编码的 "zabbix" 用户，该用户必须存在于您的系统中。只有相应修改 proxy 配置文件中的AllowRoot 参数后，您才能以 "root" 身份运行 proxy。

配置文件

配置文件的默认路径取决于操作系统和 Zabbix 的安装方式：

- 对于从源代码安装：/usr/local/etc/zabbix_proxy.conf，或者由编译时 变量 --sysconfdir 或 --prefix 设置。
- 对于从软件包安装：/etc/zabbix/zabbix_proxy.conf。

有关如何配置 zabbix_proxy 的详细信息，请参见配置文件 选项。

Proxy 进程类型和线程

- agent poller - 用于被动检查的异步轮询进程，包含一个工作线程；
- apm manager - 应用性能监控（APM）管理进程；
- apm worker - 处理来自 apm manager 的应用性能监控（APM）任务的进程；
- availability manager - 负责更新主机可用性的进程；
- browser poller - 用于浏览器监控项检查的轮询进程；
- configuration syncer - 负责管理内存中配置数据缓存的进程；
- data sender - proxy 数据发送进程；
- discovery manager - 负责设备发现的管理进程；
- discovery worker - 处理来自 discovery manager 的发现任务的进程；
- history syncer - 历史数据数据库写入进程；
- housekeeper - 负责删除过期监控项历史数据的进程；
- http agent poller - 用于 HTTP 检查的异步轮询进程，包含一个工作线程；
- http poller - Web 监控轮询进程；
- icmp pinger - 用于 icmpping 检查的轮询进程；
- internal poller - 用于内部检查的轮询进程；
- ipmi manager - IPMI 轮询管理进程；
- ipmi poller - 用于 IPMI 检查的轮询进程；
- java poller - 用于 Java 检查的轮询进程；
- odbc poller - 用于 ODBC 检查的轮询进程；
- poller - 用于被动检查的普通轮询进程；
- preprocessing manager - 预处理任务管理进程，包含预处理工作线程；
- preprocessing worker - 用于数据预处理的线程；
- self-monitoring - 用于收集内部服务器统计信息的进程；
- snmp poller - 用于 SNMP 检查的异步轮询进程，包含一个工作线程（仅适用于 walk[OID] 和 get[OID] 监控项）；
- snmp trapper - SNMP trap 接收进程；
- supervisor - 协调服务器进程和线程的启动与关闭，并监控其就绪状态；
- task manager - 用于远程执行其他组件请求的任务的进程（例如，关闭问题、确认问题、立即检查监控项值以及远程命令功能）；
- telemetry query poller - 用于遥测查询监控项检查的轮询进程；
- trapper - 用于主动检查、trap 和 proxy 通信的接收进程；
- unreachable poller - 用于不可达设备的轮询进程；
- vmware collector - VMware 数据收集器，负责从 VMware 服务收集数据。

可以使用 proxy 日志文件观察这些进程类型。

proxy 日志文件仅为文件所有者创建读写权限。此外，文件所属组中的用户也具有读取权限。其他所有权限均被拒绝。

可以使用 zabbix[process,<type>,<mode>,<state>] 内部监控项监控各种类型的 Zabbix proxy 进程。

历史同步器事务统计

history syncer 进程标题会显示有关 history syncer 事务的详细统计信息。

```
205276 ?        S      0:00  zabbix_proxy: history syncer #1 [processed 1 values in 0.001179 (0.001167,0.0
205277 ?        S      0:00  zabbix_proxy: history syncer #2 [processed 0 values in 0.000022 (0.000000,0.0
```

processed...in N (<timings>) sec 中的计时包括：

- 将监控项值写入数据库所花费的时间。
- 更新监控项数据（状态、错误）所花费的时间。

清理过程

Zabbix proxy 有一个 housekeeper 进程，用于删除过期的监控项历史数据和趋势数据。它按周期运行，运行频率由HousekeepingFrequency 决定，而每个周期的删除上限由ProxyLocalBuffer 和ProxyOfflineBuffer 决定。与 Zabbix 服务器的清理过程 不同，proxy 的 housekeeper 进程不使用 housekeeper 表——它会在每个清理周期一次性删除所有过期数据。

支持的平台　Zabbix proxy 运行在与 Zabbix 服务器相同的支持的平台 列表上。

内存缓冲区　内存缓冲区允许您将新数据（监控项值、网络发现、主机自动注册）存储在缓冲区中，并在不访问数据库的情况下将其上传到 Zabbix 服务器。Zabbix 7.0 为 proxy 引入了内存缓冲区。

在 Zabbix 7.0 之前的安装中，采集的数据会先存储在数据库中，然后再上传到 Zabbix 服务器。对于这些安装，升级到 Zabbix 7.0 后，这仍然是默认行为。

为优化性能，建议在 proxy 上配置使用内存缓冲区。可以通过将ProxyBufferMode 的值从 disk（现有安装的硬编码默认值）修改为 hybrid（推荐）或 memory 来实现。此外，还必须设置内存缓冲区大小（ProxyMemoryBufferSize 参数）。

在混合模式下，如果 proxy 停止、缓冲区已满或数据过旧，缓冲区会将未发送的数据刷新到数据库，从而避免数据丢失。当所有值都已刷新到数据库后，proxy 会恢复使用内存缓冲区。

在内存模式下，将使用内存缓冲区，但不会防止数据丢失。如果 proxy 停止，或内存缓冲区被填满，未发送的数据将被丢弃。

自 Zabbix 7.0 起，所有新安装默认使用混合模式（ProxyBufferMode=hybrid）。

其他参数（例如ProxyMemoryBufferSize 和ProxyMemoryBufferAge）分别定义内存缓冲区大小和缓冲区中数据的最大存留时间。

请注意，如果配置存在冲突，proxy 将输出错误并启动失败，例如：

- ProxyBufferMode 设置为 hybrid 或 memory，且 ProxyMemoryBufferSize 为 0。
- ProxyBufferMode 设置为 hybrid 或 memory，且 ProxyLocalBuffer 不为 0。

区域设置　请注意，proxy 需要使用 UTF-8 区域设置，以便正确解析某些文本项。大多数现代类 Unix 系统默认都使用 UTF-8 区域设置，不过在某些系统上，可能需要专门进行设置。

维护期间队列的计算

> **Attention:**
> Zabbix proxy 不会感知维护周期；有关详细信息，请参见维护期间队列的计算。

在 **proxy** 后使用 **Zabbix** 前端　如果 Zabbix 前端运行在 proxy 服务器后面，则需要重写 proxy 配置文件中的 cookie 路径，使其与反向 proxy 的路径匹配。请参见以下示例。如果未重写 cookie 路径，用户在尝试登录 Zabbix 前端时可能会遇到授权问题。

NGINX 配置示例

```
### ..
location / {
### ..
proxy_cookie_path /zabbix /;
proxy_pass http://192.168.0.94/zabbix/;
### ..
```

Apache 配置示例

```
### ..
ProxyPass "/" http://host/zabbix/
ProxyPassReverse "/" http://host/zabbix/
ProxyPassReverseCookiePath /zabbix /
ProxyPassReverseCookieDomain host zabbix.example.com
### ..
```

使用 **proxy** 进行分布式监控

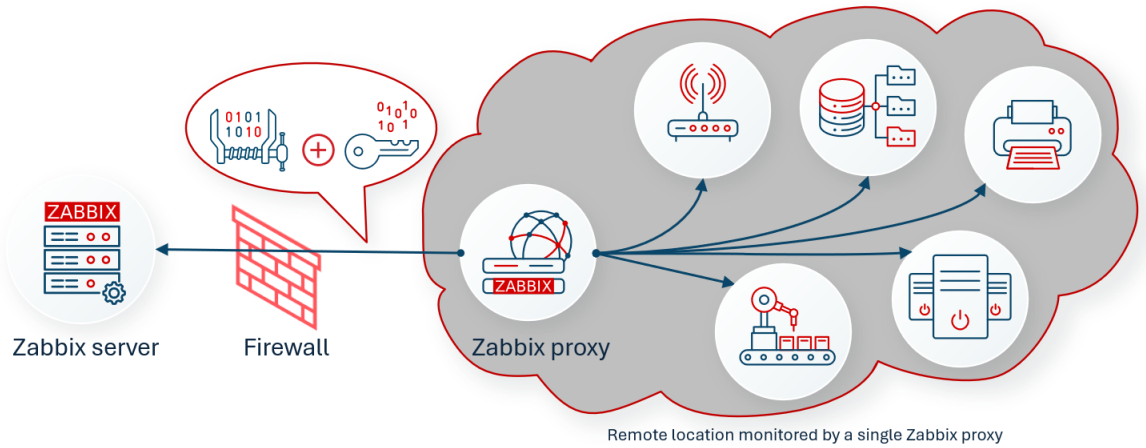
概述

Zabbix proxy 可用于构建高效且可靠的分布式监控环境。

proxy 部署在本地，用于收集性能和可用性数据。然后将数据报告给集中式服务器。

Zabbix proxy 可用于：

- 监控远程位置
- 监控通信不可靠的位置
- 在监控数千台设备时减轻 Zabbix 服务器的负载
- 收集用于应用性能监控（APM）的遥测数据



proxy 只需要与 Zabbix 服务器建立一个 TCP 连接。这样可以更轻松地绕过防火墙，因为您只需配置一条防火墙规则。

Attention:
Zabbix proxy 必须使用独立的数据库。将其指向 Zabbix 服务器数据库会破坏配置。

proxy 收集的所有数据都会先存储在本地，然后再传输到服务器。这样，即使与服务器之间出现临时通信问题，也不会丢失数据。proxy 配置文件中的 `ProxyLocalBuffer` 和 `ProxyOfflineBuffer` 参数控制数据在本地保留的时长。

Attention:
proxy 直接从 Zabbix 服务器数据库接收最新配置更改，因此其配置可能比 Zabbix 服务器更新得更及时；而 Zabbix 服务器的配置更新速度可能受CacheUpdateFrequency 值的影响。因此，proxy 可能会开始收集数据并将其发送给 Zabbix 服务器，而 Zabbix 服务器会忽略这些数据。

Zabbix proxy 是数据收集器。它不会计算触发器、处理事件或发送告警。有关 proxy 支持的功能，请参阅下表：

功能		proxy 支持
监控项		
	Zabbix agent 检查	是
	Zabbix agent 检查（主动式）	是 [1]
	简单检查	是
	Trapper 监控项	是
	SNMP 检查	是
	SNMP 陷阱	是
	IPMI 检查	是
	JMX 检查	是
	日志文件监控	是
	内部检查	是
	SSH 检查	是
	Telnet 检查	是
	外部检查	是
	依赖监控项	是
	脚本监控项	是
	浏览器监控项	是
	遥测查询监控项	是
内置 Web 监控		是
监控项值预处理		是
网络发现		是
主动式 agent 自动注册		是
低级别发现		是 [2]
远程命令		是
遥测数据收集（APM）		是 [3]
计算触发器		否
处理事件		否
事件关联		否
发送告警		否

Note:
[1] 要确保 agent 向 proxy（而不是服务器）请求主动式检查，必须在 agent 配置文件的 **ServerActive** 参数中列出该 proxy。
 [2] 对于 LLD，Zabbix proxy 只收集和预处理数据，然后将其发送给 Zabbix 服务器进行进一步处理。
 [3] 只有在 proxy 上启用 APM 时，才会收集遥测数据（请参阅proxy 配置中的 *APM* 选项卡）；Zabbix 服务器无法接收遥测数据。

proxy 功能

在选择是否使用 proxy 时，需要考虑多个因素。

	Proxy
轻量级	是
GUI	否
独立运行	是
易于维护	是
自动创建数据库	是[1]
本地管理	否
适用于嵌入式硬件	是
单向 *TCP* 连接	是
集中式配置	是
生成通知	否

[1] 自动创建数据库功能仅适用于 SQLite。其他受支持的数据库需要手动设置。

Attention:
Zabbix proxy 不会感知维护周期；有关详细信息，请参见维护期间队列的计算。

防止过载

如果 Zabbix 服务器曾经停机一段时间，而 proxy 已收集了大量数据，随后服务器启动时，可能会发生过载（history cache 使用率会在一段时间内保持在 95-100%）。这种过载可能会导致性能下降，使检查的处理速度低于应有水平。为避免由于 history cache 过载而引发的问题，已实现对此场景的保护机制。

当 Zabbix 服务器 history cache 已满时，会对 history cache 的写入访问进行限流，从而阻塞服务器的数据采集进程。最常见的 history cache 过载场景是服务器停机后，proxy 正在上传已采集的数据。为避免这种情况，已添加 proxy 限流机制（当前无法禁用）。

当 history cache 使用率达到 80% 时，Zabbix 服务器进入限流模式。在限流模式下，服务器仅在 history cache 使用率低于 60% 时接受 proxy 数据，并轮换接收的 proxy。一旦 history cache 使用率降至 20% 以下，服务器会切换回正常模式。

此外，在正常模式下，如果 history cache 使用率超过 60%，Zabbix 服务器会对发送非常大数据包（10,000+ 条记录）的单个 proxy 进行限流。此决策是在服务器评估 proxy 上传的瞬间应用的，因此可能不会立即反映在 history cache 使用率图表中（`zabbix[wcache,history,pused]` 内部监控项及其更新间隔可能会错过短暂的峰值）。

此限流模式将持续，直到缓存使用率再次达到 80%、降至 20%，或者限流列表为空。在第一种情况下，服务器将再次停止接受 proxy 数据。在另外两种情况下，服务器将恢复正常工作，接受来自所有 proxy 的数据。

上述信息可在下表中说明：

History write cache usage	Zabbix server mode	Zabbix server action
达到 80%	等待	停止接受 proxy 数据，但保留一个 *throttling list*（稍后联系的优先级 proxy 列表）。
达到 60%	正常，但已准备进入限流	在决定是否接受数据时，可能拒绝非常大的 proxy 上传（超过 1 万条记录）；继续接受其他 proxy 数据。
降至 20%	正常	清除限流列表，并开始正常接受 proxy 数据。

您可以使用zabbix[wcache,history,pused] 内部监控项，将 Zabbix 服务器的这种行为与某个指标进行关联。

配置

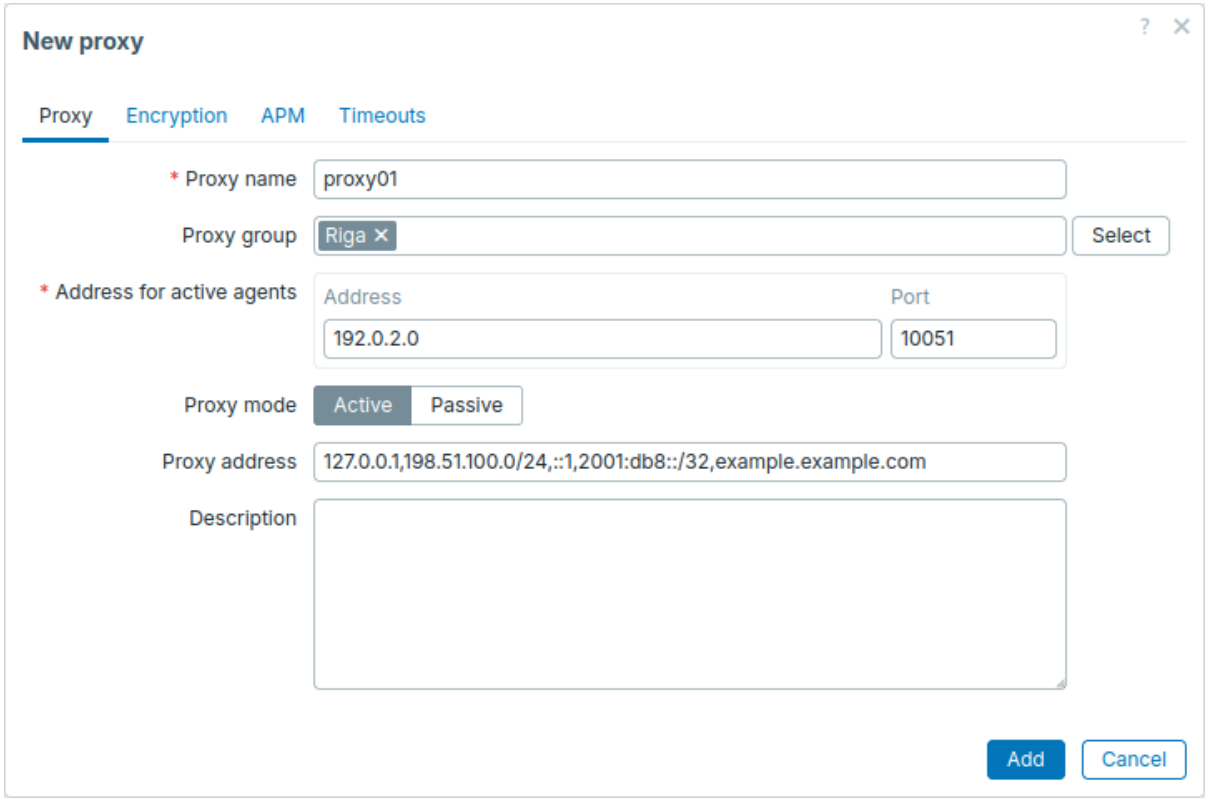
在你已经安装并配置了 proxy 之后，就可以在 Zabbix 前端中对其进行配置了。

添加 proxy

Note:
您可以在 Zabbix Cloud 中部署 proxy，而不是在自己的基础设施上安装 proxy。为此，请在管理 > *Proxy* 中点击在 *Cloud* 中创建 *proxy*；Zabbix Cloud 将在新的浏览器标签页中打开。在那里创建 proxy 节点后，按照下文所述在 Zabbix 前端中配置 proxy。有关更多信息，请参阅设置 Zabbix Cloud proxy。

要在 Zabbix 前端中配置 proxy：

1. 转到管理 > *Proxy*。
2. 点击创建 *proxy*。
3. 在表单中输入 proxy 参数。



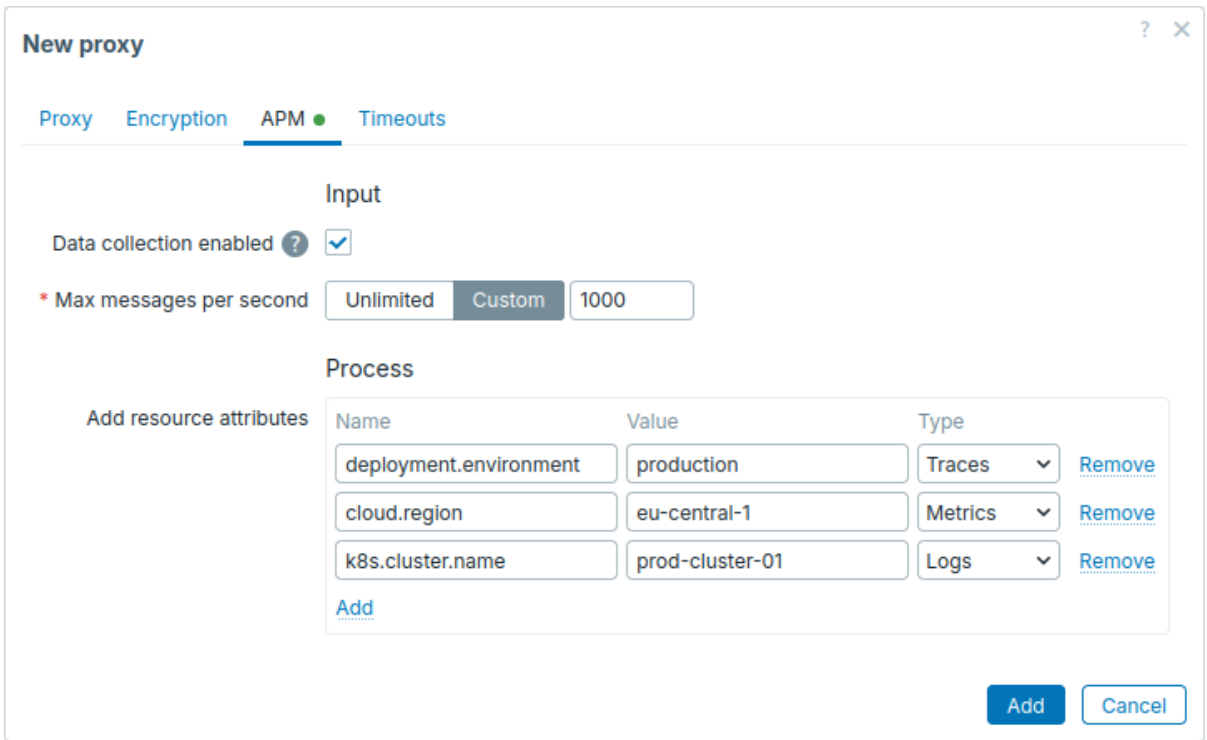
参数	描述
Proxy 名称	输入 proxy 名称。该名称必须与 proxy 配置文件中 *Hostname* 参数的名称相同。
Proxy 组	选择一个 proxy 组，用于 proxy 负载均衡/高可用。
活动 *agent* 的地址	输入受监控的活动 agent 或发送方必须连接到的地址。仅支持 Zabbix 7.0 agent 或更高版本。 此地址用于连接活动和被动 proxy。仅当在 *Proxy* 组字段中选择了 proxy 组时，此字段才可用。
地址	要连接到的 IP 地址/DNS 名称。
端口	要连接到的 TCP 端口号（默认为 10051）。支持用户宏。
Proxy 模式	选择 proxy 模式。 活动 - proxy 将连接到 Zabbix 服务器并请求配置数据 被动 - Zabbix 服务器连接到 proxy 注意，在使用活动 proxy 时，如果未进行身份验证，或未在 *Proxy* 地址字段中限制 proxy 地址，则未加密的通信（敏感）proxy 配置数据可能会被能够访问 Zabbix 服务器 trapper 端口的各方获取。这是因为任何人都可以伪装成活动 proxy，并在未进行身份验证或未限制 proxy 地址的情况下请求配置数据。
Proxy 地址	如果指定，则仅接受来自此逗号分隔的 IP 地址列表（可选择使用 CIDR 表示法）或活动 Zabbix proxy 的 DNS 名称的活动 proxy 请求。 仅当在 *Proxy* 模式字段中选择了活动 proxy 时，此字段才可用。不支持宏。

参数		描述
接口		输入被动 proxy 的接口详细信息。 仅当在 *Proxy* 模式字段中选择了被动 proxy 时，此字段才可用。
	地址	被动 proxy 的 IP 地址/DNS 名称。
	端口	被动 proxy 的 TCP 端口号（默认为 10051）。支持用户宏。
描述		输入 proxy 描述。

加密选项卡允许您要求与 proxy 建立加密连接。

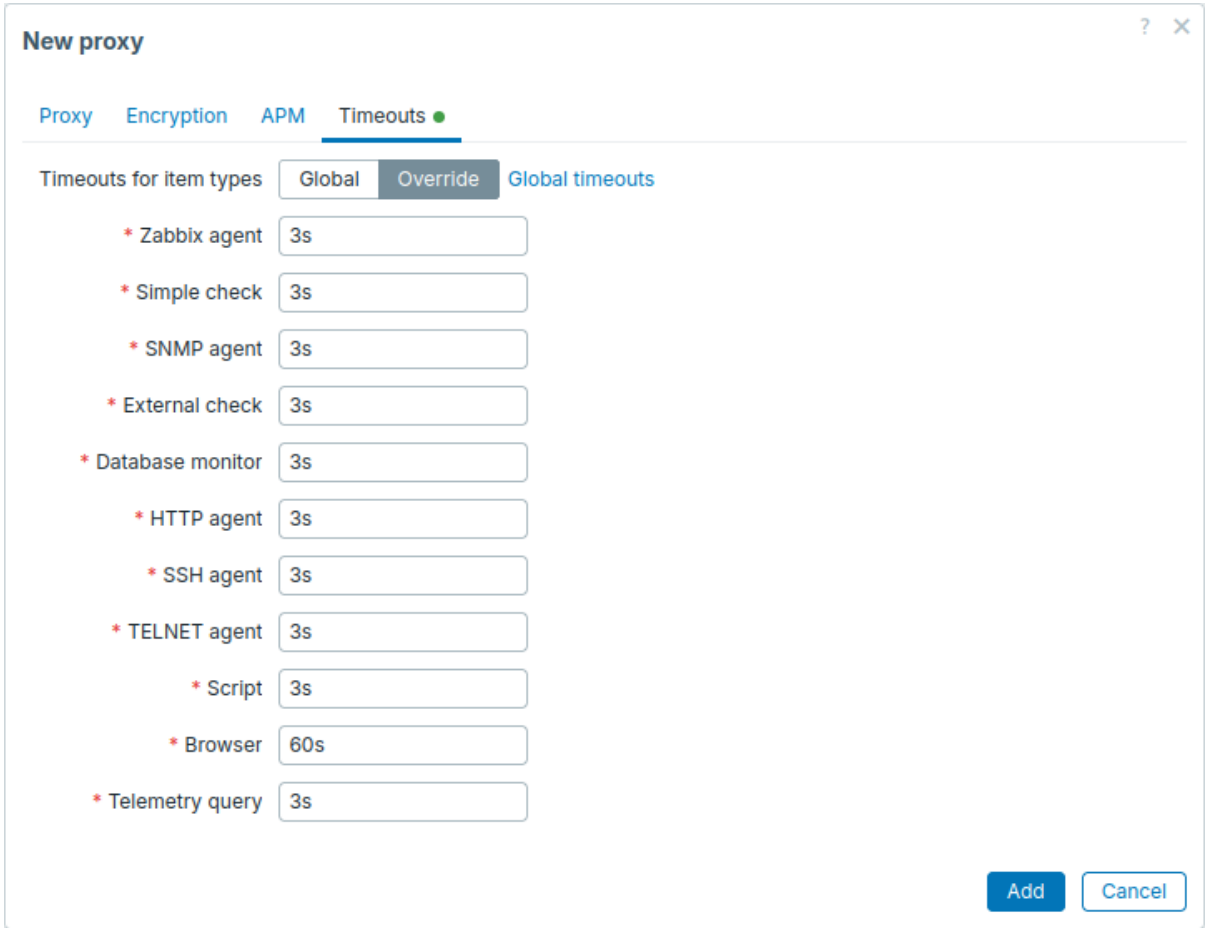
参数	描述
到 *proxy* 的连接	服务器连接到被动 proxy 的方式：不加密（默认）、使用 PSK（预共享密钥）或证书。
来自 *proxy* 的连接	选择允许来自活动 proxy 的连接类型。可以同时选择多种连接类型（便于测试并切换到其他连接类型）。默认为 “不加密”。
颁发者	允许的证书颁发者。首先使用 CA（证书颁发机构）验证证书。如果证书有效且由 CA 签名，则可以使用颁发者字段进一步限制允许的 CA。此字段为可选字段，适用于 Zabbix 安装使用来自多个 CA 的证书的情况。
主题	允许的证书主题。首先使用 CA 验证证书。如果证书有效且由 CA 签名，则可以使用主题字段仅允许一个主题字符串值。如果此字段为空，则接受由已配置 CA 签名的任何有效证书。
PSK 标识	预共享密钥标识字符串。 请勿在 PSK 标识中放入敏感信息，因为该标识会通过网络以未加密方式传输，用于告知接收方应使用哪个 PSK。
PSK	预共享密钥（十六进制字符串）。如果 Zabbix 使用 GnuTLS 或 OpenSSL 库，最大长度为 512 个十六进制数字（256 字节 PSK）；如果 Zabbix 使用 mbed TLS (PolarSSL) 库，最大长度为 64 个十六进制数字 (32 字节 PSK)。示例： 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

APM 选项卡允许您在 proxy 上配置应用性能监控 (APM)。proxy 可以通过 OpenTelemetry 协议 (OTLP/gRPC) 收集应用或 OpenTelemetry collector 发送的遥测数据（追踪、指标和日志），并将数据写入遥测数据源。



参数		描述
启用数据收集		选中复选框以启用 proxy 上的遥测数据收集。 启用遥测数据收集前，请在 proxy 配置文件中将`StartAPMCollectors`参数设置为 1。
每秒最大消息数		设置 proxy 接受的每秒最大消息数： 无限制 - 接受所有传入消息。 自定义 - 每秒最多接受指定数量的消息。 消息是由应用、OpenTelemetry collector 或其他遥测源发送的 OTLP 请求。消息大小取决于发送方；例如，OpenTelemetry collector 可能会将数千个 span 批量放入一条消息，而简单应用可能发送较小的消息（例如每条消息包含一个 trace）。 如果超过限制，proxy 将返回 gRPC 状态码 `UNAVAILABLE`（临时错误），因此发送方可能会根据其配置稍后重试发送消息。
添加资源属性		指定要添加到所选类型传入遥测数据中的资源属性（例如，使用环境或区域标记此 proxy 接收的所有数据）。随后可以使用这些属性筛选遥测数据，例如，在遥测查询监控项条件中使用。
	名称	输入属性名称（例如，`deployment.environment`、`cloud.region`）。对于所选类型必须唯一。
	值	输入属性值（例如，`production`、`eu-central-1`）。可以为空。
	类型	选择属性适用的传入数据类型：追踪、指标或日志。

超时选项卡允许您覆盖支持此功能的监控项类型的全局超时。



参数	描述
监控项类型的超时	设置监控项超时（基于其类型）： 全局 - 使用全局超时（显示在每种监控项类型对应的灰显超时字段中）； 覆盖 - 设置自定义超时（在每种监控项类型的超时字段中设置）。允许范围：1 - 600 秒（默认值：继承自全局超时）。支持时间后缀，例如 30s、1m，以及用户宏。 点击全局超时链接可以配置全局超时。请注意，只有具有超级管理员类型且拥有管理 > 常规前端部分权限的用户才能看到全局超时链接。 请注意，虽然 proxy 级别的超时会覆盖全局超时，但如果为单个监控项配置了超时，则单个监控项的超时会覆盖 proxy 级别的超时。

Note:
如果 proxy 主版本与服务器主版本不匹配，则会在监控项类型的超时旁显示 图标，并显示悬停消息 “由于 proxy 和服务器版本不匹配，超时已禁用”。在这种情况下，proxy 将使用 proxy 配置文件中的`Timeout`参数。

现有 proxy 的编辑表单还包含以下按钮：

- 刷新配置 - 刷新 proxy 的配置；
- 克隆 - 根据现有 proxy 的属性创建新的 proxy；
- 删除 - 删除 proxy。

主机配置

您可以在主机配置 表单中，通过 *Monitored by* 字段指定某个单独的主机应由 proxy 或 proxy group 监控。



主机批量更新 也是指定主机应由 proxy 或 proxy group 监控的另一种方式。

proxy 负载均衡和高可用性

概述

Zabbix proxy 可以组织到 proxy 组中，以实现 proxy 负载均衡和高可用性。

Proxy 负载均衡和高可用性是指在 proxy 组内自动重新分配主机到各个 proxy：

- 如果某个 proxy 离线，其主机会被移动到其他 proxy，从而保持较高的 proxy 可用性。
- 如果某个 proxy 的主机数量明显高于/低于其他 proxy，其主机会被移动到其他 proxy，以平衡 proxy 负载。

主机重新分配仅在满足以下条件的组内 proxy 之间生效：

- proxy 运行的是 Zabbix 7.0 或更高版本。
- Proxy 版本 与 Zabbix 服务器版本匹配。如果使用 Zabbix agent（被动模式），proxy 版本必须与 agent 版本匹配。主动 agent 只要求 Zabbix 7.0 或更高版本。
- proxy 组处于在线状态。
- 主机配置为由 proxy 组监控，而不是由单个 proxy 监控。

Note:
任何分配到 proxy 组的主机都可以通过内部检查 监控 proxy 组的健康状态。但是，要监控组中单个 proxy 的健康状态，请将主机分配给该 proxy；否则，结果可能不一致。

主机重新分配

proxy 负载均衡和高可用性由 Zabbix 服务器通过proxy group manager 管理，它会持续监控每个 proxy 组中所有 proxy 的状态及其主机分布。

组内的 proxy 高可用性通过 proxy 故障切换来保证：当某个 proxy 离线时，其主机会立即重新分配到其他 proxy。同时也会发生 proxy 负载均衡，因为主机会被重新分配到已分配主机数量最少的 proxy。

此外，当 proxy 的主机数量与组平均值相差至少 10 台且相差 2 倍时（主机过多或主机不足），会触发 proxy 负载均衡。如果在宽限期（10 x failover delay）之后不平衡仍然存在，则会将该 proxy 组加入主机重新分配队列。

proxy group manager 使用以下逻辑重新分配主机：

1. 计算每个 proxy 的平均主机数量。
2. 对于主机过多的 proxy——将多余的主机移到未分配的 proxy 池中。
3. 对于主机不足的 proxy——计算达到平衡所需的主机数量。
4. 从主机数量最多的 proxy 中移除所需数量的主机。
5. 将未分配的主机移动到主机数量最少的 proxy。

主机重新分配示例：

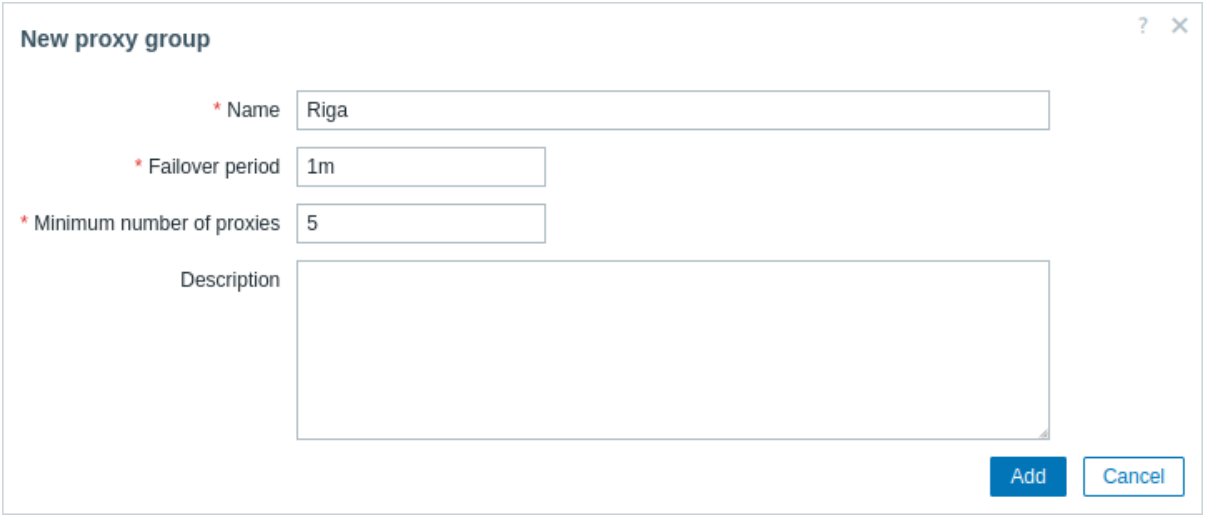
proxy 上的主机数	组平均值	主机重新分配
100	50	是
60	50	否
40	50	否
25	50	是
15	5	是
10	5	否

Attention:
由少于 10 台监控 的主机组成的 proxy 组，可能会导致组内各 proxy 之间的主机分布不均。

配置 proxy 组

要在 Zabbix 前端中配置 proxy 组：

1. 转到 管理 > *Proxy* 组。
2. 单击 创建 *proxy* 组。
3. 在表单中输入 proxy 组参数。



参数	描述
名称	proxy 组的名称。
故障转移周期	proxy 组中的 proxy 必须与 Zabbix 服务器通信、才能被视为在线的时间段（默认值：1m；范围：10s-15m）。如果 proxy 未在此时间段内进行通信，其状态将更改为 离线，并且其主机会立即重新分配给其他 proxy。proxy 负载均衡将在经过该时间段的 10 倍时间后开始。 支持时间后缀（例如 30s、1m）和用户宏。
proxy 最小数量	保持proxy 组在线所需的在线 proxy的最小数量（默认值：1；范围：1-1000）。 支持用户宏。 此值应小于组中 proxy 的总数。例如，在包含 10 个 proxy 的组中，将最小数量设置为 10 会导致任意 proxy 发生故障时该组转为离线。请注意，离线组中的在线 proxy 仍会正常运行，但不会进行负载均衡/高可用。
描述	proxy 组的描述。

参数	描述
Proxy	编辑包含 proxy 的组时，显示最多五个 proxy 的列表（根据用户对 proxy 的权限，以链接或纯文本形式显示）。

配置 proxy 负载均衡

要使用 proxy 负载均衡，您需要在 Zabbix 前端中配置一个 proxy 组（见上文），并确保主机是由proxy 组 监控，而不是由单个 proxy 监控（您可以使用主机批量更新 将主机从各个 proxy 迁移到 proxy 组）。

如果使用 Zabbix agent，还需要按如下方式进行配置：

- 对于被动检查，请在Server 参数中列出 proxy 组中的所有 proxy。
- 对于主动检查，建议在ServerActive 参数中列出 proxy 组中的所有 proxy 或 Zabbix 服务器。请注意，只有 Zabbix agent 7.0（或更高版本）才能在主动模式下与 proxy 组配合使用。

如果 `ServerActive` 参数中只包含 proxy 组中的一个 proxy（或 Zabbix 服务器），agent 仍然能够连接到正确的 proxy。当 agent 服务启动并连接到指定的 proxy 时，agent 将接收并缓存 proxy 组中所有 proxy 的完整 IP 列表及其当前负载。随后，主动检查将根据 proxy 组内当前的 proxy-主机分配关系，重定向到该主机对应的在线 proxy。

Warning:
如果 Zabbix agent 的 `ServerActive` 参数中只指定了单个 proxy，那么当 agent 启动/重启且该指定 proxy 处于离线状态时，可能会导致监控数据丢失。

Note:
使用Zabbix sender 时，数据请求也会根据 proxy 组内当前的 proxy-主机分配关系，重定向到该主机对应的在线 proxy。但是，如果您要从输入文件中发送多个主机的值，请使用`-g` 选项，以防止将数据发送到错误的 proxy。

Zabbix agent 还必须能够通过防火墙连接到 proxy 组中的所有 proxy。否则，在重定向或故障切换期间，主动检查可能会挂起或失败。比如：

- 在主动检查期间，proxy 可能会将 agent 重定向到另一个 proxy。如果该 proxy 被防火墙阻止，通信将在等待响应时挂起。
- 在稳定的高可用性部署中，如果近期没有进行重新均衡，agent 可能永远不会联系到备用 proxy。如果防火墙规则已更改但未经过测试，故障切换可能会失败。

测试 proxy 负载均衡

要测试 proxy 负载均衡：

1. 配置一个 proxy 组。
2. 确保该 proxy 组处于在线状态。
3. 确保主机是由 proxy 组监控，而不是由单个 proxy 监控（您可以使用主机批量更新将主机从 proxy 移动到 proxy 组）。
4. 等待几秒钟，让配置更新以及主机在 proxy 组内各个 proxy 之间完成分配。通过刷新 *Administration* > *Proxies* 中的主机列表来观察变化。

重要说明

- SNMP trap 不受 proxy group 中的 proxy 支持。
- 依赖外部配置的检查（例如，external check 的脚本或database check 的 ODBC 配置）必须在 proxy group 中的所有 proxy 上保持相同配置。
- Database check 需要对数据库对象/服务器具有扩展权限。
- 由 proxy group 中的 proxy 监控 的 VMware 主机会在该组内的各个 proxy 之间随机分配。这会导致每个 proxy 缓存所有 VMware 数据，从而给 vCenter 带来额外负载。
- 基于来自 proxy group 中某个 proxy 的自动注册数据创建的主机，会被设置为由该 proxy group 监控。但是，基于来自 proxy group 中某个 proxy 的网络发现数据创建的主机，会被设置为由该 proxy 监控。

监控配置的同步

概述

本页提供有关 proxy 的监控配置更新的详细信息，即服务器上的监控配置更改如何同步到 proxy。

增量更新

proxy 配置更新是增量进行的。在配置同步期间，仅更新已修改的实体（因此，如果没有实体被修改，则不会发送任何内容）。这种方式可以节省资源，并将 proxy 配置更新间隔设置得更短（几乎即时）。

proxy 配置更改通过修订号进行跟踪。只有修订号大于 proxy 配置修订号的实体，才会包含在发送到 proxy 的配置数据中。

配置同步涉及的实体如下：

实体	详细信息
自动注册 *TLS* 数据	所有自动注册 TLS 数据。
表达式	所有表达式（正则表达式、表达式表）。
全局配置	在'config' 表中定义的全局配置
主机	主机的所有属性、接口、资产清单、监控项、监控项预处理、监控项参数和 Web 场景。
主机宏	主机上定义的所有宏，以及与其关联的所有模板 ID。
proxy 发现规则	分配给 proxy 的发现规则和检查。

这意味着：

- 如果主机上的监控项发生更改，则该主机的所有配置都会同步。
- 如果正则表达式发生更改，则所有正则表达式都会同步。

例外情况是：只要主机上的任何内容发生更改，主机宏也会被发送。

proxy 上的 `-R config_cache_reload` 命令也会启动增量更新。

请注意，在 proxy 启动/重启、HA 故障转移、会话令牌发生更改时，或者 proxy 上的配置更新失败时（例如，在接收配置数据期间连接中断），都会在 proxy 上执行完整的配置同步。

配置参数

ProxyConfigFrequency 参数决定 proxy 配置与服务器同步的频率（默认每 10 秒一次）。

请注意，ProxyConfigFrequency 是：

- passive proxy 的服务器参数
- active proxy 的 proxy 参数

在 active proxy 上，ProxyConfigFrequency 是 Zabbix 6.4 中新增的参数，必须用它替代现已弃用的 ConfigFrequency。

Attention:
如果同时使用 ProxyConfigFrequency 和 ConfigFrequency，proxy 将记录错误并终止。

维护期间队列的计算

Attention:
Zabbix proxy 不会感知维护周期；有关详细信息，请参见维护期间队列的计算。

服务器**-proxy** 数据交换协议

服务器 - proxy 数据交换基于 JSON 格式。

请求和响应消息必须以header and data length 开头。

被动 proxy

配置请求

服务器首先会发送一个空的 `proxy config` 请求。此请求每隔 `ProxyConfigFrequency`（服务器配置参数）秒发送一次。

proxy 会响应当前 proxy 版本、会话令牌和配置修订版本。服务器会响应需要更新的配置数据。

name	value type	description
server→proxy:		
request	*string*	'proxy config'
proxy→server:		
version	*string*	proxy 版本（<major>.<minor>.<build>）。
session	*string*	proxy 配置会话令牌。
config_revision	*number*	proxy 配置修订版本。
server→proxy:		
full_sync	*number*	如果发送完整配置数据，则为 1；否则不存在（可选）。

name	value type	description
data	*array*	表数据对象。如果配置未发生更改，则不存在（可选）。
<table>	*object*	一个或多个包含 <table> 数据的对象（可选，具体取决于更改情况）。
fields	*array*	字段名称数组。
-	*string*	字段名称。
data	*array*	行数组。
-	*array*	列数组。
-	*string,number*	列值，其类型取决于数据库架构中的列类型。
macro.secrets	*object*	宏的机密信息；如果保管库宏没有更改，则不存在（可选）。
config_revision	*number*	配置缓存修订版本 - 随配置数据发送（可选）。
del_hostids	*array*	已移除主机 ID 的数组（可选）。
-	*number*	主机标识符。
del_macro_hostids	*array*	所有宏均已移除的主机 ID 数组（可选）。
-	*number*	主机标识符。
proxy→server:		
response	*string*	请求成功信息（'success' 或'failed'）。
version	*string*	proxy 版本（<major>.<minor>.<build>）。

示例：

服务器 →proxy：

```
{
  "request":"proxy config"
}
```

proxy→ 服务器：

```
{
  "version": "8.0.0",
  "session": "0033124949800811e5686dbfd9bcea98",
  "config_revision": 0
}
```

服务器 →proxy：

```
{
    "full_sync": 1,
    "data": {
        "hosts": {
            "fields": ["hostid", "host", "status", "ipmi_authtype", "ipmi_privilege", "ipmi_username", "ip
            "data": [
                [10084, "Zabbix server", 0, -1, 2, "", "", "Zabbix server", 1, 1, "", "", "", ""]
            ]
        },
        "interface": {
            "fields": ["interfaceid", "hostid", "main", "type", "useip", "ip", "dns", "port", "available"]
            "data": [
                [1, 10084, 1, 1, 1, "127.0.0.1", "", "10053", 1]
            ]
        },
        "interface_snmp": {
            "fields": ["interfaceid", "version", "bulk", "community", "securityname", "securitylevel", "au
            "data": []
        },
        "host_inventory": {
            "fields": ["hostid", "type", "type_full", "name", "alias", "os", "os_full", "os_short", "seria
            "data": [
                [10084, "", "", ..., "56.95387", "24.22067", ..., ""]
            ]
        },
        "items": {
```

```
            "fields": ["itemid", "type", "snmp_oid", "hostid", "key_", "delay", "history", "status", "valu
            "data": [
                [44161, 7, "", 10084, "agent.hostmetadata", "10s", "90d", 0, 1, "", "", "", "", 0, "", "",
                [44162, 0, "", 10084, "agent.ping", "10s", "90d", 0, 3, "", "", "", "", 0, "", "", "", "",
            ]
        },
        "item_rtdata": {
            "fields": ["itemid", "lastlogsize", "mtime"],
            "data": [
                [44161, 0, 0],
                [44162, 0, 0]
            ]
        },
        "item_preproc": {
            "fields": ["item_preprocid", "itemid", "step", "type", "params", "error_handler", "error_handl
            "data": []
        },
        "item_parameter": {
            "fields": ["item_parameterid", "itemid", "name", "value"],
            "data": []
        },
        "globalmacro": {
            "fields": ["globalmacroid", "macro", "value", "type"],
            "data": [
                [2, "{$SNMP_COMMUNITY}", "public", 0]
            ]
        },
        "hosts_templates": {
            "fields": ["hosttemplateid", "hostid", "templateid", "link_type"],
            "data": []
        },
        "hostmacro": {
            "fields": ["hostmacroid", "hostid", "macro", "value", "type", "automatic"],
            "data": [
                [5676, 10084, "{$M}", "AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix:Content",
            ]
        },
        "drules": {
            "fields": ["druleid", "name", "iprange", "delay"],
            "data": [
                [2, "Local network", "127.0.0.1", "10s"]
            ]
        },
        "dchecks": {
            "fields": ["dcheckid", "druleid", "type", "key_", "snmp_community", "ports", "snmpv3_securityn
            "data": [
                [2, 2, 9, "system.uname", "", "10052", "", 0, "", "", 0, 0, 0, "", 1, 0]
            ]
        },
        "regexps": {
            "fields": ["regexpid", "name"],
            "data": [
                [1, "File systems for discovery"],
                [2, "Network interfaces for discovery"],
                [3, "Storage devices for SNMP discovery"],
                [4, "Windows service names for discovery"],
                [5, "Windows service startup states for discovery"]
            ]
        },
        "expressions": {
            "fields": ["expressionid", "regexpid", "expression", "expression_type", "exp_delimiter", "case
            "data": [
```

```
                [1, 1, "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|z
                [3, 3, "^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$", 4, ",
                [5, 4, "^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.303
                [6, 5, "^(automatic|automatic delayed)$", 3, ",", 1],
                [7, 2, "^Software Loopback Interface", 4, ",", 1],
                [8, 2, "^(In)?[Ll]oop[Bb]ack[0-9._]*$", 4, ",", 1],
                [9, 2, "^NULL[0-9.]*$", 4, ",", 1],
                [10, 2, "^[Ll]o[0-9.]*$", 4, ",", 1],
                [11, 2, "^[Ss]ystem$", 4, ",", 1],
                [12, 2, "^Nu[0-9.]*$", 4, ",", 1]
            ]
        },
        "settings": {
            "fields": ["name", "type", "value_str", "value_int"],
            "data": [
                ["autoreg_tls_accept", 2, "", 1],
                ["hk_history_global", 2, "", 0],
                ["snmptrap_logging", 2, "", 1],
                ["proxy_secrets_provider", 2, "", 0],
                ["hk_history", 1, "31d", 0],
                ["timeout_db_monitor", 1, "3s", 0],
                ["timeout_external_check", 1, "3s", 0],
                ["timeout_http_agent", 1, "3s", 0],
                ["timeout_simple_check", 1, "3s", 0],
                ["timeout_snmp_agent", 1, "3s", 0],
                ["timeout_ssh_agent", 1, "3s", 0],
                ["timeout_telnet_agent", 1, "3s", 0],
                ["timeout_zabbix_agent", 1, "3s", 0],
                ["timeout_browser", 1, "30s", 0]
            ]
        },
        "httptest": {
            "fields": ["httptestid", "name", "delay", "agent", "authentication", "http_user", "http_passwo
            "data": []
        },
        "httptestitem": {
            "fields": ["httptestitemid", "httptestid", "itemid", "type"],
            "data": []
        },
        "httptest_field": {
            "fields": ["httptest_fieldid", "httptestid", "type", "name", "value"],
            "data": []
        },
        "httpstep": {
            "fields": ["httpstepid", "httptestid", "name", "no", "url", "timeout", "posts", "required", "s
            "data": []
        },
        "httpstepitem": {
            "fields": ["httpstepitemid", "httpstepid", "itemid", "type"],
            "data": []
        },
        "httpstep_field": {
            "fields": ["httpstep_fieldid", "httpstepid", "type", "name", "value"],
            "data": []
        },
        "config_autoreg_tls": {
            "fields": ["autoreg_tlsid", "tls_psk_identity", "tls_psk"],
            "data": [
                [1, "", ""]
            ]
        }
    },
```

```
    "macro.secrets": {
        "AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix": {
            "Content": "738"
        }
    },
    "config_revision": 2
}
```

proxy→ 服务器：

```
{
  "response": "success",
  "version": "8.0.0"
}
```

数据请求

`proxy data` 请求用于从 proxy 获取主机接口可用性、历史、发现和自动注册数据。此请求每隔 `ProxyDataFrequency`（服务器配置参数）秒发送一次。

name	value type	description
server→proxy:		
request	*string*	'proxy data'
proxy→server:		
session	*string*	数据会话令牌。
interface availability	*array*	*(optional)* 接口可用性数据对象数组。
interfaceid	*number*	接口标识符。
available	*number*	接口可用性： **0**, *INTERFACE_AVAILABLE_UNKNOWN* - 未知 **1**, *INTERFACE_AVAILABLE_TRUE* - 可用 **2**, *INTERFACE_AVAILABLE_FALSE* - 不可用
error	*string*	接口错误消息，或空字符串。
history data	*array*	*(optional)* 历史数据对象数组。
itemid	*number*	监控项标识符。
clock	*number*	监控项值时间戳（秒）。
ns	*number*	监控项值时间戳（纳秒）。
value	*string*	*(optional)* 监控项值。
id	*number*	值标识符（递增计数器，在一个数据会话内唯一）。
timestamp	*number*	*(optional)* 日志类型监控项的时间戳。
source	*string*	*(optional)* Eventlog 监控项 source 值。
severity	*number*	*(optional)* Eventlog 监控项 severity 值。
eventid	*number*	*(optional)* Eventlog 监控项 eventid 值。
state	*string*	*(optional)* 监控项状态： **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
lastlogsize	*number*	*(optional)* 日志类型监控项的最后日志大小。
mtime	*number*	*(optional)* 日志类型监控项的修改时间。
discovery data	*array*	*(optional)* 发现数据对象数组。
clock	*number*	发现数据时间戳。
druleid	*number*	发现规则标识符。
dcheckid	*number*	发现检查标识符，或在发现规则数据中为 null。

name	value type	description
type	*number*	发现检查类型： **-1** 发现规则数据 **0**, *SVC_SSH* - SSH 服务检查 **1**, *SVC_LDAP* - LDAP 服务检查 **2**, *SVC_SMTP* - SMTP 服务检查 **3**, *SVC_FTP* - FTP 服务检查 **4**, *SVC_HTTP* - HTTP 服务检查 **5**, *SVC_POP* - POP 服务检查 **6**, *SVC_NNTP* - NNTP 服务检查 **7**, *SVC_IMAP* - IMAP 服务检查 **8**, *SVC_TCP* - TCP 端口可用性检查 **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS 服务检查 **15**, *SVC_TELNET* - Telnet 可用性检查
ip	*string*	主机 IP 地址。
dns	*string*	主机 DNS 名称。
port	*number*	*(optional)* 服务端口号。
key_	*string*	*(optional)* 类型为 **9** *SVC_AGENT* 的发现检查所用的监控项键值
value	*string*	*(optional)* 从服务接收到的值，对大多数服务可以为空。
status	*number*	*(optional)* 服务状态： **0**, *DOBJECT_STATUS_UP* - 服务正常 **1**, *DOBJECT_STATUS_DOWN* - 服务故障
auto reg- is- tra- tion	*array*	*(optional)* 自动注册数据对象数组。
clock	*number*	自动注册数据时间戳。
host	*string*	主机名。
ip	*string*	*(optional)* 主机 IP 地址。
dns	*string*	*(optional)* 从 IP 地址解析得到的 DNS 名称。
port	*string*	*(optional)* 主机端口。
host_metadata	*string*	*(optional)* agent 发送的主机元数据（基于 HostMetadata 或 HostMetadataItem agent 配置参数）。
tasks	*array*	*(optional)* 任务数组。
type	*number*	任务类型： **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - 远程命令结果
status	*number*	远程命令执行状态： **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - 远程命令已成功完成 **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - 远程命令失败
error	*string*	*(optional)* 错误消息。
parent_taskid	*number*	父任务 ID。
more	*number*	*(optional)* 1 - 还有更多历史数据需要发送。
clock	*number*	*(optional)* 数据传输时间戳（秒）。
ns	*number*	*(optional)* 数据传输时间戳（纳秒）。
version	*string*	proxy 版本（<major>.<minor>.<build>）。
server→proxy:		
response	*string*	请求成功信息（'success' 或'failed'）。
tasks	*array*	*(optional)* 任务数组。
type	*number*	任务类型： **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - 远程命令
clock	*number*	任务创建时间。

name	value type	description
ttl	*number*	任务过期前的秒数。
commandtype	*number*	远程命令类型： **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - 使用自定义脚本 **1**, *ZBX_SCRIPT_TYPE_IPMI* - 使用 IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - 使用 SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - 使用 Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - 使用全局脚本（当前功能上等同于自定义脚本）
command	*string*	要执行的远程命令。
execute_on	*number*	自定义脚本的执行目标： **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - 在 agent 上执行脚本 **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - 在服务器上执行脚本 **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - 在 proxy 上执行脚本
port	*number*	*(optional)* Telnet 和 SSH 命令的端口。
authtype	*number*	*(optional)* SSH 命令的认证类型。
username	*string*	*(optional)* Telnet 和 SSH 命令的用户名。
password	*string*	*(optional)* Telnet 和 SSH 命令的密码。
publickey	*string*	*(optional)* SSH 命令的公钥。
privatekey	*string*	*(optional)* SSH 命令的私钥。
parent_taskid	*number*	父任务 ID。
hostid	*number*	目标主机 ID。

示例：

server→proxy:

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

proxy→server:

```
{
    "session": "12345678901234567890123456789012"
    "interface availability": [
        {
            "interfaceid": 1,
            "available": 1,
            "error": ""
    },
        {
            "interfaceid": 2,
            "available": 2,
            "error": "Get value from agent failed: cannot connect to [[127.0.0.1]:10049]: [111] Connection
    },
        {
            "interfaceid": 3,
            "available": 1,
            "error": ""
    },
        {
            "interfaceid": 4,
            "available": 1,
            "error": ""
    }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
```

```
            "ns":332510044,
            "value":"52956612",
            "id": 1
        },
        {
            "itemid":"12346",
            "clock":1478609647,
            "ns":330690279,
            "state":1,
            "value":"Cannot find information for this network interface in /proc/net/dev.",
            "id": 2
        }
    ],
    "discovery data":[
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":3,
            "type":12,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        },
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":null,
            "type":-1,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        }
    ],
    "auto registration":[
        {
            "clock":1478608371,
            "host":"Logger1",
            "ip":"10.3.0.1",
            "dns":"localhost",
            "port":"10050"
        },
        {
            "clock":1478608381,
            "host":"Logger2",
            "ip":"10.3.0.2",
            "dns":"localhost",
            "port":"10050"
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
```

```
    "version":"8.0.0"
}
```

server→proxy:

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

Tasks request

`proxy tasks` 交换用于在服务器和被动 proxy 之间传递远程命令任务的交付与确认。当服务器按 ProxyDataFrequency 定义的间隔轮询 proxy 以获取任务结果时，会发送一个空的 `proxy tasks` 请求。proxy 会返回其当前版本以及任何待处理的任务结果（包括时间戳）。随后，服务器通过返回一个 `response` 来确认已收到，并且可能会为 proxy 提供需要执行的新任务。

name	value type	description
server→proxy:		
request	*string*	'proxy tasks'
proxy→server:		
version	*string*	Proxy 版本（<major>.<minor>.<build>）。
clock	*number*	*(optional)* 数据传输时间戳（秒）。
ns		*number*
server→proxy:		
response	*string*	请求成功信息（'success' 或'failed'）。

示例：

server→proxy:

```
{
  "request":"proxy tasks"
}
```

proxy→server:

```
{
  "version":"7.0.0",
  "clock":1721059872,
  "ns":848141522
}
```

server→proxy:

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

活动 proxy

配置请求

`proxy config` 请求由主动 proxy 发送，用于获取 proxy 配置数据。此请求每隔 `ProxyConfigFrequency`（proxy 配置参数）秒发送一次。

name	value type	description
proxy→server:		
request	*string*	'proxy config'
host	*string *	Proxy 名称。
version	*string*	Proxy 版本（<major>.<minor>.<build>）。
session	*string*	Proxy 配置会话令牌。
config_revision	*number*	Proxy 配置修订版本。
hostmap_revision	*number*	当前主机到 proxy 映射的修订版本。
server→proxy:		
full_sync	*number*	如果发送完整配置数据，则为 1；否则不存在（可选）。
data	*array*	表数据对象。如果配置未发生更改，则不存在（可选）。
<table>	*object*	一个或多个包含 <table> 数据的对象（可选，取决于更改情况）。
fields	*array*	字段名称数组。
-	*string*	字段名称。
data	*array*	行数组。
-	*array*	列数组。
-	*string,number*	列值，其类型取决于数据库架构中的列类型。
macro.secrets	*object*	Secret 宏信息；如果保管库宏没有更改，则不存在（可选）。
proxy_group	*string*	Proxy 所属 proxy 组的名称。
config_revision	*number*	配置缓存修订版本 - 随配置数据发送（可选）。
del_hostids	*array*	已移除 hostid 的数组（可选）。
-	*number*	主机标识符。
del_macro_hostids	*array*	所有宏均已移除的 hostid 数组（可选）。
-	*number*	主机标识符。

示例：

proxy→server:

```
{
  "request": "proxy config",
  "host": "Zabbix proxy",
  "version":"8.0.0",
  "session": "fd59a09ff4e9d1fb447de1f04599bcf6",
  "config_revision": 0
}
```

server→proxy:

```
{
"full_sync": 1,
"data": {
"hosts": {
"fields": ["hostid", "host", "status", "ipmi_authtype", "ipmi_privilege", "ipmi_username", "ipmi_password"
"data": [
[10084, "Zabbix server", 0, -1, 2, "", "", "Zabbix server", 1, 1, "", "", "", ""]
]
},
"interface": {
"fields": ["interfaceid", "hostid", "main", "type", "useip", "ip", "dns", "port", "available"],
"data": [
[1, 10084, 1, 1, 1, "127.0.0.1", "", "10053", 1]
]
},
"interface_snmp": {
"fields": ["interfaceid", "version", "bulk", "community", "securityname", "securitylevel", "authpassphrase
"data": []
},
"host_inventory": {
"fields": ["hostid", "type", "type_full", "name", "alias", "os", "os_full", "os_short", "serialno_a", "ser
"data": [
[10084, "", "", ..., "56.95387", "24.22067", ..., ""]
]
},
"items": {
"fields": ["itemid", "type", "snmp_oid", "hostid", "key_", "delay", "history", "status", "value_type", "tr
"data": [
[44161, 7, "", 10084, "agent.hostmetadata", "10s", "90d", 0, 1, "", "", "", "", 0, "", "", "", "", 0, null
[44162, 0, "", 10084, "agent.ping", "10s", "90d", 0, 3, "", "", "", "", 0, "", "", "", "", 0, 1, 0, "", nu
]
},
"item_rtdata": {
"fields": ["itemid", "lastlogsize", "mtime"],
"data": [
[44161, 0, 0],
[44162, 0, 0]
]
},
"item_preproc": {
"fields": ["item_preprocid", "itemid", "step", "type", "params", "error_handler", "error_handler_params"],
"data": []
},
"item_parameter": {
"fields": ["item_parameterid", "itemid", "name", "value"],
"data": []
},
"globalmacro": {
"fields": ["globalmacroid", "macro", "value", "type"],
"data": [
[2, "{$SNMP_COMMUNITY}", "public", 0]
]
},
"hosts_templates": {
"fields": ["hosttemplateid", "hostid", "templateid", "link_type"],
"data": []
},
"hostmacro": {
"fields": ["hostmacroid", "hostid", "macro", "value", "type", "automatic"],
"data": [
[5676, 10084, "{$M}", "AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix:Content", 2, 0]
]
```

```
},
"drules": {
"fields": ["druleid", "name", "iprange", "delay"],
"data": [
[2, "Local network", "127.0.0.1", "10s"]
]
},
"dchecks": {
"fields": ["dcheckid", "druleid", "type", "key_", "snmp_community", "ports", "snmpv3_securityname", "snmpv
"data": [
[2, 2, 9, "system.uname", "", "10052", "", 0, "", "", 0, 0, 0, "", 1, 0]
]
},
"regexps": {
"fields": ["regexpid", "name"],
"data": [
[1, "File systems for discovery"],
[2, "Network interfaces for discovery"],
[3, "Storage devices for SNMP discovery"],
[4, "Windows service names for discovery"],
[5, "Windows service startup states for discovery"]
]
},
"expressions": {
"fields": ["expressionid", "regexpid", "expression", "expression_type", "exp_delimiter", "case_sensitive"]
"data": [
[1, 1, "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$", 3, ",", 0
[3, 3, "^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$", 4, ",", 1],
[5, 4, "^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$", 4, ","
[6, 5, "^(automatic|automatic delayed)$", 3, ",", 1],
[7, 2, "^Software Loopback Interface", 4, ",", 1],
[8, 2, "^(In)?[Ll]oop[Bb]ack[0-9._]*$", 4, ",", 1],
[9, 2, "^NULL[0-9.]*$", 4, ",", 1],
[10, 2, "^[Ll]o[0-9.]*$", 4, ",", 1],
[11, 2, "^[Ss]ystem$", 4, ",", 1],
[12, 2, "^Nu[0-9.]*$", 4, ",", 1]
]
},
"settings": {
"fields": ["name", "type", "value_str", "value_int"],
"data": [
["autoreg_tls_accept", 2, "", 1],
["hk_history_global", 2, "", 0],
["snmptrap_logging", 2, "", 1],
["proxy_secrets_provider", 2, "", 0],
["hk_history", 1, "31d", 0],
["timeout_db_monitor", 1, "3s", 0],
["timeout_external_check", 1, "3s", 0],
["timeout_http_agent", 1, "3s", 0],
["timeout_simple_check", 1, "3s", 0],
["timeout_snmp_agent", 1, "3s", 0],
["timeout_ssh_agent", 1, "3s", 0],
["timeout_telnet_agent", 1, "3s", 0],
["timeout_zabbix_agent", 1, "3s", 0],
["timeout_browser", 1, "30s", 0]
]
},
"httptest": {
"fields": ["httptestid", "name", "delay", "agent", "authentication", "http_user", "http_password", "hostid
"data": []
},
"httptestitem": {
```

```
"fields": ["httptestitemid", "httptestid", "itemid", "type"],
"data": []
},
"httptest_field": {
"fields": ["httptest_fieldid", "httptestid", "type", "name", "value"],
"data": []
},
"httpstep": {
"fields": ["httpstepid", "httptestid", "name", "no", "url", "timeout", "posts", "required", "status_codes"
"data": []
},
"httpstepitem": {
"fields": ["httpstepitemid", "httpstepid", "itemid", "type"],
"data": []
},
"httpstep_field": {
"fields": ["httpstep_fieldid", "httpstepid", "type", "name", "value"],
"data": []
},
"config_autoreg_tls": {
"fields": ["autoreg_tlsid", "tls_psk_identity", "tls_psk"],
"data": [
[1, "", ""]
]
}
},
"macro.secrets": {
"AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix": {
"Content": "738"
}
},
"config_revision": 2
}
```


数据请求

`proxy data` 请求由 proxy 发送，用于提供主机接口可用性、历史数据、发现和自动注册数据。该请求每隔 `DataSenderFrequency`（proxy 配置参数）秒发送一次。请注意，主动模式 proxy 仍会每秒轮询 Zabbix 服务器以获取远程命令任务（使用空的 `proxy data` 请求）。

name	value type	description
proxy→server:		
request	*string*	'proxy data'
host	*string*	proxy 名称。
session	*string*	数据会话令牌。
interface availability	*array*	*(可选)* 接口可用性数据对象数组。
interfaceid	*number*	接口标识符。
available	*number*	接口可用性： **0**, *INTERFACE_AVAILABLE_UNKNOWN* - 未知 **1**, *INTERFACE_AVAILABLE_TRUE* - 可用 **2**, *INTERFACE_AVAILABLE_FALSE* - 不可用
error	*string*	接口错误消息，或空字符串。
history data	*array*	*(可选)* 历史数据对象数组。
itemid	*number*	监控项标识符。
clock	*number*	监控项值时间戳（秒）。
ns	*number*	监控项值时间戳（纳秒）。
value	*string*	*(可选)* 监控项值。
id	*number*	值标识符（递增计数器，在一个数据会话内唯一）。

name	value type	description
timestamp	*number*	*(可选)* 日志类型监控项的时间戳。
source	*string*	*(可选)* Eventlog 监控项来源值。
severity	*number*	*(可选)* Eventlog 监控项严重性值。
eventid	*number*	*(可选)* Eventlog 监控项 eventid 值。
state	*string*	*(可选)* 监控项状态： **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
lastlogsize	*number*	*(可选)* 日志类型监控项的上次日志大小。
mtime	*number*	*(可选)* 日志类型监控项的修改时间。
discovery	*array*	*(可选)* 发现数据对象数组。
data		
clock	*number*	发现数据时间戳。
druleid	*number*	发现规则标识符。
dcheckid	*number*	发现检查标识符，或在发现规则数据中为 null。
type	*number*	发现检查类型： **-1** 发现规则数据 **0**, *SVC_SSH* - SSH 服务检查 **1**, *SVC_LDAP* - LDAP 服务检查 **2**, *SVC_SMTP* - SMTP 服务检查 **3**, *SVC_FTP* - FTP 服务检查 **4**, *SVC_HTTP* - HTTP 服务检查 **5**, *SVC_POP* - POP 服务检查 **6**, *SVC_NNTP* - NNTP 服务检查 **7**, *SVC_IMAP* - IMAP 服务检查 **8**, *SVC_TCP* - TCP 端口可用性检查 **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS 服务检查 **15**, *SVC_TELNET* - Telnet 可用性检查
ip	*string*	主机 IP 地址。
dns	*string*	主机 DNS 名称。
port	*number*	*(可选)* 服务端口号。
key_	*string*	*(可选)* 类型为 **9** *SVC_AGENT* 的发现检查监控项键值。
value	*string*	*(可选)* 从服务接收到的值，对大多数服务可以为空。
status	*number*	*(可选)* 服务状态： **0**, *DOBJECT_STATUS_UP* - 服务正常 **1**, *DOBJECT_STATUS_DOWN* - 服务异常
autoregistration	*array*	*(可选)* 自动注册数据对象数组。
clock	*number*	自动注册数据时间戳。
host	*string*	主机名。
ip	*string*	*(可选)* 主机 IP 地址。
dns	*string*	*(可选)* 从 IP 地址解析得到的 DNS 名称。
port	*string*	*(可选)* 主机端口。
host_metadata	*string*	*(可选)* agent 发送的主机元数据（基于 HostMetadata 或 HostMetadataItem agent 配置参数）。
tasks	*array*	*(可选)* 任务数组。
type	*number*	任务类型： **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - 远程命令结果
status	*number*	远程命令执行状态： **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - 远程命令已成功完成 **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - 远程命令失败
error	*string*	*(可选)* 错误消息。
parent_taskid	*number*	父任务 ID。
more	*number*	*(可选)* 1 - 还有更多历史数据需要发送

name	value type	description
clock	*number*	*(可选)* 数据传输时间戳（秒）。
ns	*number*	*(可选)* 数据传输时间戳（纳秒）。
version	*string*	proxy 版本（<major>.<minor>.<build>）。
server→proxy:		
response	*string*	请求成功信息（'success' 或'failed'）。
upload	*string*	历史数据上传控制（历史数据、自动注册、主机可用性、网络发现）。 可能的值： **enabled** - 正常运行 **disabled** - 服务器不接受数据（可能由于内部缓存超限）
tasks	*array*	*(可选)* 任务数组。
type	*number*	任务类型： **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - 远程命令
clock	*number*	任务创建时间。
ttl	*number*	任务过期前的秒数。
commandtype	*number*	远程命令类型： **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - 使用自定义脚本 **1**, *ZBX_SCRIPT_TYPE_IPMI* - 使用 IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - 使用 SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - 使用 Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - 使用全局脚本（当前功能上等同于自定义脚本）
command	*string*	要执行的远程命令。
execute_on	*number*	自定义脚本的执行目标： **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - 在 agent 上执行脚本 **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - 在服务器上执行脚本 **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - 在 proxy 上执行脚本
port	*number*	*(可选)* Telnet 和 SSH 命令的端口。
authtype	*number*	*(可选)* SSH 命令的认证类型。
username	*string*	*(可选)* Telnet 和 SSH 命令的用户名。
password	*string*	*(可选)* Telnet 和 SSH 命令的密码。
publickey	*string*	*(可选)* SSH 命令的公钥。
privatekey	*string*	*(可选)* SSH 命令的私钥。
parent_taskid	*number*	父任务 ID。
hostid	*number*	目标主机 ID。

示例：

proxy→server:


```
{
"request": "proxy data",
"host": "Zabbix proxy",
"session": "818cdd1b537bdc5e50c09ed4969235b6",
"interface availability": [{
"interfaceid": 1,
"available": 1,
"error": ""
}],
"history data": [{
"id": 1114,
"itemid": 44162,
"clock": 1665730632,
"ns": 798953105,
"value": "1"
}, {
"id": 1115,
"itemid": 44161,
"clock": 1665730633,
```

```
"ns": 811684663,
"value": "58"
}],
"auto registration": [{
"clock": 1665730633,
"host": "Zabbix server",
"ip": "127.0.0.1",
"dns": "localhost",
"port": "10053",
"host_metadata": "58",
"tls_accepted": 1
}],
"discovery data": [{
"clock": 1665732232,
"drule": 2,
"dcheck": 2,
"ip": "127.0.0.1",
"dns": "localhost",
"port": 10052,
"status": 1
}, {
"clock": 1665732232,
"drule": 2,
"dcheck": null,
"ip": "127.0.0.1",
"dns": "localhost",
"status": 1
}],
"host data": [{
"hostid": 10084,
"active_status": 1
}],
"tasks": [{
"type": 3,
"clock": 1665730985,
"ttl": 0,
"status": -1,
"info": "Remote commands are not enabled",
"parent_taskid": 3
}],
"version": "8.0.0",
"clock": 1665730643,
"ns": 65389964
}
```


server→proxy:


```
{
"upload": "enabled",
"response": "success",
"tasks": [{
"type": 2,
"clock": 1665730986,
"ttl": 600,
"commandtype": 0,
"command": "ping -c 3 127.0.0.1; case $? in [01]) true;; *) false;; esac",
"execute_on": 2,
"port": 0,
"authtype": 0,
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
```

```
"alertid": 0,
"parent_taskid": 4,
"hostid": 10084
}]
}
```

配置参数：**Zabbix proxy**

概述

配置参数允许自定义 Zabbix proxy 的行为。

在 Zabbix proxy 配置文件（zabbix_proxy.conf）中指定参数值。除非明确说明参数为必需参数，否则所有参数均为可选。

本页面的结构如下：

- 参数概览（要查看参数的完整详细信息，请单击其名称）
- 参数详细信息

参数	描述
AllowRoot	允许 proxy 以 root 身份运行。
AllowUnsupportedDBVersions	允许 proxy 使用不受支持的数据库版本。
APMListenIP	指定 APM 收集器用于监听传入遥测数据的 IP 地址。
APMListenPort	指定 APM 收集器用于监听传入遥测数据的端口。
APMTLSCAFile	指定包含顶级 CA 证书的文件的完整路径名，用于验证向 APM 收集器发送数据的应用程序或 OpenTelemetry 收集器的证书。
APMTLSCertFile	指定包含 APM 收集器证书或证书链的文件的完整路径名。
APMTLSKeyFile	指定包含 APM 收集器私钥的文件的完整路径名。
CacheSize	设置配置缓存大小。
ConfigFrequency	此参数已弃用（请使用 ProxyConfigFrequency）。 设置从 Zabbix 服务器获取配置数据的频率。
DataSenderFrequency	设置向 Zabbix 服务器发送所收集数据的频率。
DBHost	指定数据库主机（或套接字目录）。
DBName	指定数据库名称，或指定 SQLite3 数据库文件的路径。
DBPassword	指定数据库密码。
DBPort	指定 Zabbix 数据库主机的端口。
DBSchema	指定数据库架构名称。用于 PostgreSQL。
DBSocket	指定 MySQL 套接字文件的路径。
DBUser	指定数据库用户。
DBTLSConnect	指定一个值以强制与数据库建立 TLS 连接。
DBTLSCAFile	指定包含顶级 CA 证书的文件的完整路径名，用于验证数据库证书。
DBTLSCertFile	指定包含 Zabbix proxy 证书的文件的完整路径名，用于向数据库进行身份验证。
DBTLSKeyFile	指定包含私钥的文件的完整路径名，用于向数据库进行身份验证。
DBTLSCipher	指定 Zabbix proxy 允许用于 TLS v1.2 及更低版本 TLS 协议的加密密码列表。仅支持 MySQL。
DBTLSCipher13	指定 Zabbix proxy 允许用于 TLS v1.3 协议的加密密码套件列表。仅支持 MySQL，且从 8.0.16 版本开始支持。
DebugLevel	设置调试级别。
EnableRemoteCommands	允许来自 Zabbix 服务器的远程命令。
ExternalScripts	指定外部脚本位置。
Fping6Location	指定 fping6 的位置。
FpingLocation	指定 fping 的位置。
HistoryCacheSize	设置历史数据缓存大小。
HistoryIndexCacheSize	设置历史数据索引缓存大小。
Hostname	指定 proxy 名称。
HostnameItem	指定用于设置 Hostname 的监控项（如果未定义）。
HousekeepingFrequency	设置内务处理过程的频率（小时）。
Include	指定要包含在配置文件中的目录或单个文件。
JavaGateway	指定 Zabbix Java 网关的 IP 地址（或主机名）。
JavaGatewayPort	指定 Zabbix Java 网关端口。
ListenBacklog	设置 TCP 队列中待处理连接的最大数量。
ListenIP	指定 trapper 监听 IP 地址。
ListenPort	指定 trapper 监听端口。
LoadModule	指定 proxy 启动时要加载的模块。

参数	描述
LoadModulePath	指定 proxy 模块位置的完整路径。
LogFile	指定 proxy 日志文件。
LogFileSize	设置日志文件的最大大小。
LogRemoteCommands	启用将已执行的 shell 命令作为警告记录到日志中。
LogSlowQueries	设置数据库查询在记录日志前允许执行的最长时间（毫秒）。
LogType	指定日志输出类型。
MaxConcurrentChecksPerPoller	设置每个 HTTP agent 轮询器、agent 轮询器或 SNMP 轮询器可同时执行的异步检查的最大数量。
PidFile	指定 PID 文件。
ProxyBufferMode	指定历史数据、发现数据和自动注册数据的存储机制（磁盘/内存/混合）。
ProxyConfigFrequency	设置从 Zabbix 服务器获取配置数据的频率（秒）。
ProxyLocalBuffer	设置在本地保留数据的时长（小时），即使数据已经与服务器同步。
ProxyMemoryBufferAge	设置 proxy 内存缓冲区中数据的最大存留时间（秒）。
ProxyMemoryBufferSize	设置用于存储所收集历史数据、发现数据和自动注册数据的共享内存缓存大小。
ProxyMode	设置 proxy 运行模式（主动/被动）。
ProxyOfflineBuffer	设置与 Zabbix 服务器失去连接时保留数据的时长（小时）。
Server	如果 ProxyMode 设置为 `active`：指定用于获取配置数据和发送数据的 Zabbix 服务器 IP 地址或 DNS 名称（address:port），或集群（address:port;address2:port）。 如果 ProxyMode 设置为 `passive`：指定 Zabbix 服务器的 IP 地址或 DNS 名称。
SNMPTrapperFile	指定用于将数据从 SNMP trap 守护进程传递到 proxy 的临时文件。
SocketDir	指定用于存储 Zabbix 内部服务所使用 IPC 套接字的目录。
SourceIP	指定源 IP 地址。
SSHKeyLocation	指定 SSH 检查和操作所使用的公钥和私钥的位置。
SSLCertLocation	指定用于客户端身份验证的 SSL 客户端证书文件的位置。
SSLKeyLocation	指定用于客户端身份验证的 SSL 私钥文件的位置。
SSLCALocation	指定用于验证 SSL 服务器证书的证书颁发机构（CA）文件的位置。
StartAgentPollers	设置预先派生的异步 Zabbix agent 轮询器实例数量。
StartAPMCollectors	设置为 1 以启动 APM 收集器。
StartBrowserPollers	设置预先派生的浏览器监控项轮询器实例数量。
StartDBSyncers	设置预先派生的历史数据同步器实例数量。
StartDiscoverers	设置预先派生的发现工作进程实例数量。
StartHTTPAgentPollers	设置预先派生的异步 HTTP agent 轮询器实例数量。
StartHTTPPollers	设置预先派生的 HTTP 轮询器实例数量。
StartIPMIPollers	设置预先派生的 IPMI 轮询器实例数量。
StartJavaPollers	设置预先派生的 Java 轮询器实例数量。
StartODBCPollers	设置预先派生的 ODBC 轮询器实例数量。
StartPingers	设置预先派生的 ICMP ping 进程实例数量。
StartPollersUnreachable	设置预先派生的不可达主机轮询器实例数量（包括 IPMI 和 Java）。
StartPollers	设置预先派生的轮询器实例数量。
StartPreprocessors	设置预先启动的预处理工作进程实例数量。
StartSNMPPollers	设置预先派生的异步 SNMP 轮询器实例数量。
StartSNMPTrapper	设置为”1” 以启动 SNMP trapper 进程。
StartTelemetryQueryPollers	设置预先派生的异步 Telemetry 查询轮询器实例数量。
StartTrappers	设置预先派生的 trapper 实例数量。
StartVMwareCollectors	设置预先派生的 VMware 收集器实例数量。
StatsAllowedIP	指定允许请求统计信息的外部 Zabbix 实例的 IP 地址或 DNS 名称。
TelemetryProvider	设置遥测数据源。
Timeout	指定建立连接以及与 Zabbix proxy、agent、Web 服务交换数据时的等待时间（秒），以及 SNMP 检查的等待时间（不包括 SNMP `walk[OID]` 和 `get[OID]` 监控项）。
TLSAccept	指定来自 Zabbix 服务器的传入连接所需的加密级别。
TLSCAFile	指定包含顶级 CA 证书的文件的完整路径名，用于验证对等证书，并用于 Zabbix 组件之间的加密通信。
TLSCertFile	指定包含服务器证书或证书链的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSCipherAll	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书和 PSK 的加密的默认密码套件选择条件。
TLSCipherAll13	指定 TLS 1.3 中适用于 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖基于证书和 PSK 的加密的默认密码套件选择条件。
TLSCipherCert	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书的加密的默认密码套件选择条件。
TLSCipherCert13	指定 TLS 1.3 中适用于 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖基于证书的加密的默认密码套件选择条件。
TLSCipherPSK	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于 PSK 的加密的默认密码套件选择条件。

参数	描述
TLSCipherPSK13	指定 TLS 1.3 中适用于 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖基于 PSK 的加密的默认密码套件选择条件。
TLSConnect	指定与 Zabbix 服务器建立连接所需的加密级别。
TLSCRLFile	指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。
TLSKeyFile	指定包含 proxy 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSListen	要求 trapper 套接字仅使用 TLS 连接。
TLSPSKFile	指定包含 proxy 预共享密钥的文件的完整路径名，用于与 Zabbix 服务器进行加密通信。
TLSPSKIdentity	指定预共享密钥身份字符串，用于与 Zabbix 服务器进行加密通信。
TLSServerCertIssuer	指定允许的服务器证书颁发者。
TLSServerCertSubject	指定允许的服务器证书主题。
TmpDir	指定临时目录。
TrapperTimeout	指定以下操作的超时时间（秒）： - 从 Zabbix 服务器获取配置数据； - 执行全局脚本或远程命令。
UnavailableDelay	设置在不可用期间检查主机可用性的频率。
UnreachableDelay	设置在不可达期间检查主机可用性的频率。
UnreachablePeriod	设置主机在不可达多少秒后被视为不可用。
User	将权限降级为系统中指定的现有用户。
Vault	指定 vault 提供程序。
VaultAppRoleID	指定 HashiCorp vault AppRole ID。
VaultAppSecretID	指定 HashiCorp vault AppRole secret ID。
VaultDBPath	指定按键获取数据库凭据的位置。
VaultPrefix	指定 vault 路径或查询的自定义前缀。
VaultTLSCertFile	指定用于客户端身份验证的 SSL 证书文件名称。
VaultTLSKeyFile	指定用于客户端身份验证的 SSL 私钥文件名称。
VaultToken	指定 HashiCorp vault 身份验证令牌。
VaultURL	指定 vault 服务器 URL。
VMwareCacheSize	设置用于存储 VMware 数据的共享内存大小。
VMwareFrequency	设置从单个 VMware 服务收集数据的频率（秒）。
VMwarePerfFrequency	设置从单个 VMware 服务获取性能计数器统计信息的频率（秒）。
VMwareTimeout	设置 `vmware collector` 等待 VMware 服务响应的最长时间（秒）。
WebDriverURL	指定 WebDriver 接口 URL。

请注意：

- 默认值反映的是守护进程默认值，而不是随附配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持使用 UTF-8 编码且不带 BOM 的配置文件。
- 以 # 开头的注释仅支持出现在行首。

参数详情

AllowRoot

允许 proxy 以 `root` 身份运行。如果禁用且 proxy 由 `root` 启动，proxy 将尝试切换为 `zabbix` 用户。如果在普通用户下启动，则无效。

默认值：0
 取值：0 - 不允许；1 - 允许

AllowUnsupportedDBVersions

允许 proxy 使用不受支持的数据库版本运行。

默认值：0
 取值：0 - 不允许；1 - 允许

APMListenIP

指定APM collector 应监听以接收传入遥测数据的 IP 地址。
 如果未指定，APM collector 将监听所有网络接口。

默认值：`0.0.0.0`

APMListenPort

指定APM collector 监听传入遥测数据的端口。
 默认端口为标准 OpenTelemetry (OTLP/gRPC) 端口，OpenTelemetry SDK 和 collector 默认使用该端口。

默认值：4317
 范围：1024-32767

APMTLSCAFile

指定包含顶级 CA 证书的文件的完整路径名，用于验证向APM collector 发送数据的应用程序或 OpenTelemetry collector 的证书。
 如果设置了此参数，APM collector 仅接受加密连接；否则，仅接受未加密连接。

APMTLSCertFile

指定包含APM collector 证书或证书链的文件的完整路径名。
 此参数必须与APMTLSKeyFile 一起设置，且仅当设置了APMTLSCAFile 时才可设置。

APMTLSKeyFile

指定包含APM collector 私钥的文件的完整路径名。
 此参数必须与APMTLSCertFile 一起设置，并且仅在设置了APMTLSCAFile 时设置。

CacheSize

设置配置缓存大小，单位为字节。用于存储主机和监控项数据的共享内存大小。

默认值：32M
 范围：128K-64G

ConfigFrequency

此参数已弃用（请改用ProxyConfigFrequency）。
 设置从 Zabbix 服务器检索配置数据的频率（以秒为单位）。
 主动 proxy 参数。对被动 proxy 忽略（请参见ProxyMode 参数）。

默认值：3600
 范围：1-604800

DataSenderFrequency

设置从 proxy 向 Zabbix 服务器发送已收集数据的频率。请注意，主动 proxy 仍会每秒轮询 Zabbix 服务器以获取远程命令任务。
 主动 proxy 参数。对被动 proxy 忽略（请参见 ProxyMode 参数）。

默认值：1
 范围：1-3600

DBHost

指定数据库主机（或套接字目录）。

对于 MySQL：

- localhost 或空字符串将使用默认的 UNIX 域套接字。

对于 PostgreSQL：

- localhost 会通过 DNS 解析（通常解析为 127.0.0.1）。
- 空字符串将使用默认的 UNIX 域套接字。
- 路径（例如 /var/run/pgbouncer）会使用该路径下的 UNIX 套接字。
- 以逗号分隔的列表可以包含多个 host:port 值。系统会按顺序尝试每个主机，直到建立读写连接。示例：DBHost=localhost:5431,127.0

默认值：localhost

DBName

指定数据库名称，或指定 SQLite3 数据库文件的路径（Zabbix 的多进程架构不允许使用内存数据库，例如 :memory:、file::memory:?cache=shared 或 file:memdb1?mode=memory&cache=shared）。
 警告：请勿尝试使用 Zabbix 服务器正在使用的同一个数据库。

必填：是

DBPassword

指定数据库密码。如果未使用密码，请注释掉此行。SQLite 会忽略此项。

DBPort

指定 Zabbix 数据库主机的端口（参见DBHost）。对于 SQLite 将被忽略。
 在 proxy 配置中，DBSocket 和 DBPort 互斥。请仅指定其中一个，或将两者都留空。

MySQL 的默认值：3306

PostgreSQL 的默认值：5432

范围：1024-65535

DBSchema

指定数据库模式名称。用于 PostgreSQL。

DBSocket

指定 MySQL socket 文件的路径。
 在 proxy 配置中，DBSocket 和 DBPort 互斥。请仅指定其中一个，或将两者都留空。

DBUser

指定数据库用户。SQLite 会忽略此项。

DBTLSConnect

指定一个值以强制与数据库建立 TLS 连接：- *required* - 使用 TLS 连接；- *verify_ca* - 使用 TLS 连接并验证证书；- *verify_full* - 使用 TLS 连接，验证证书，并验证由 DBHost 指定的数据库标识是否与其证书匹配
 在 MySQL 5.7.11 及更高版本和 PostgreSQL 中，支持以下值：”required”、”verify”、”verify_full”。
 在 MariaDB 10.2.6 及更高版本中，支持”required” 和”verify_full” 值。
 默认情况下不设置任何选项，具体行为取决于数据库配置。

DBTLSCAFile

指定包含用于数据库证书验证的顶级 CA 证书的文件的完整路径名。

必需：否（如果将 DBTLSConnect 设置为 *verify_ca* 或 *verify_full*，则为是）

DBTLSCertFile

指定包含用于数据库身份验证的 Zabbix proxy 证书的文件的完整路径名。

DBTLSKeyFile

指定包含用于对数据库进行身份验证的私钥文件的完整路径名。

DBTLSCipher

指定 Zabbix proxy 允许用于 TLS 协议（最高至 TLS v1.2）的加密套件列表。仅支持 MySQL。

DBTLSCipher13

指定 Zabbix proxy 允许用于 TLS v1.3 协议的加密套件列表。仅支持 MySQL，从 8.0.16 版本开始。

DebugLevel

设置调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。

默认值：3
 范围：0-5

EnableRemoteCommands

允许来自 Zabbix 服务器的远程命令。

默认值: 0
 取值: 0 - 不允许；1 - 允许

外部脚本

指定外部脚本的位置（取决于 `datadir` 编译时安装变量）。

默认值：`/usr/local/share/zabbix/externalscripts`

Fping6Location

指定 fping6 的位置。请确保 fping6 二进制文件的所有者为 root，并已设置 SUID 标志。如果你的 `fping` 工具能够处理 IPv6 地址，请留空（`Fping6Location=`）。

默认值：`/usr/sbin/fping6`

FpingLocation

指定 fping 的位置。请确保 fping 二进制文件的所有者为 root，并已设置 SUID 标志。

默认值：`/usr/sbin/fping`

HistoryCacheSize

设置历史缓存大小，单位为字节。用于存储历史数据的共享内存大小。

默认值：16M
 范围：128K-16G

HistoryIndexCacheSize

设置历史索引缓存大小，单位为字节。用于对存储在 history cache 中的历史数据进行索引的共享内存大小。索引缓存大小大约需要 100 字节来缓存一个监控项。

默认值：4M
 范围：128K-16G

主机名

指定一个唯一的、区分大小写的 proxy 名称。请确保服务器已知该 proxy 名称。
 允许的字符：字母数字、'.'、' '、'_' 和'-'。最大长度：128

默认值：由 HostnameItem 设置

HostnameItem

指定用于设置 Hostname 的监控项（如果未定义）。
 该监控项将在 proxy 上运行，方式与在 agent 上类似。
 如果已设置 Hostname，则忽略此项。
 不支持用户参数、性能计数器或别名；支持 system.run[]。

默认值：system.hostname

HousekeepingFrequency

设置 housekeeping 过程的频率（以小时为单位）。housekeeping 是从数据库中删除过期信息。
 注意：为降低 proxy 启动时的负载，housekeeping 会在 proxy 启动后延迟 30 分钟执行。因此，如果 HousekeepingFrequency 为 1，则 proxy 启动后的第一次 housekeeping 过程将在 30 分钟后运行，之后每小时重复一次。
 可以通过将 HousekeepingFrequency 设置为 0 来禁用自动 housekeeping。在这种情况下，housekeeping 过程只能通过 housekeeper_execute 运行时控制选项启动。
 另请参阅运行时控制 选项以及housekeeping 过程 的详细信息。

默认值：1
 范围：0-24

Include

指定要包含到配置文件中的目录或单个文件。
 在 Zabbix 安装期间，除非你在编译时更改此设置，否则该参数会被设置为 /usr/local/etc。

Include 参数的一般规则：

- 路径可以是相对于 zabbix_proxy.conf 文件位置的相对路径。
- 你可以在一个配置文件中指定多个 Include 参数。
- Zabbix 会在读取到 Include 参数后立即处理它，因此建议将其放在配置文件末尾。

包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，该组件将无法启动。

包含目录或文件模式的规则：

- 不要假设包含顺序有任何规律（例如，文件不会按字母顺序包含）。
- 不要在多个已包含文件中定义同一个参数（例如，用特定设置覆盖通用设置），因为你无法预测最终会使用哪个值。
- 注意某些文本编辑器会自动创建的备份副本(例如，编辑 include/my_specific.conf 时可能会创建 include/my_specific.conf.BA
  因为这两个文件都会被包含。

如果只想从目录中包含相关文件，请使用通配符（*）进行模式匹配。

示例：

```
Include=/absolute/path/to/config/files/*.conf
```

JavaGateway

指定 Zabbix Java 网关的 IP 地址或主机名。仅在启动 Java poller 时才需要。

JavaGatewayPort

指定 Zabbix Java 网关端口。

默认值：10052
 范围：1024-32767

ListenBacklog

设置 TCP 队列中待处理连接的最大数量。
 默认值是一个硬编码常量，具体取决于系统。
 支持的最大值也取决于系统，过高的值可能会被静默截断为 “implementation-specified maximum”。

默认值: SOMAXCONN
 范围: 0 - INT_MAX

ListenIP

指定 trapper 监听的 IP 地址。使用逗号分隔的列表。
 如果未指定此参数，trapper 将监听所有网络接口。

默认值：0.0.0.0

ListenPort

指定 trapper 监听端口。

默认值：10051
 范围：1024-32767

LoadModule

指定 proxy 启动时要加载的模块。模块用于扩展 proxy 的功能。模块必须位于 LoadModulePath 指定的目录中，或者必须在模块名称前指定路径。如果前置路径是绝对路径（以 / 开头），则会忽略 LoadModulePath。
 格式：- LoadModule=<module.so> - LoadModule=<path/module.so> - LoadModule=</abs_path/module.so>

可以指定多个 LoadModule 参数。

LoadModulePath

proxy 模块所在位置的完整路径。默认值取决于编译选项。

LogFile

指定 proxy 日志文件。

必填：是，如果 LogType 设置为 file；否则不是

LogFileSize

设置最大日志文件大小（单位：MB）。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始。

默认值：1
 范围：0-1024

LogRemoteCommands

启用将已执行的 shell 命令记录为警告。

默认值：0
 取值：0 - 禁用，1 - 启用

LogSlowQueries

设置数据库查询在被记录到日志之前允许的最长执行时间（以毫秒为单位）。
0 - 不记录慢查询。
 此选项从 DebugLevel=3 开始启用。

默认值：0
 范围：0-3600000

LogType

指定日志输出类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *system* - 将日志写入 syslog；- *console* - 将日志写入标准输出。

默认值：file

MaxConcurrentChecksPerPoller

设置每个 HTTP agent poller、agent poller 或 SNMP poller 一次可执行的异步检查的最大数量。请参见StartHTTPAgentPollers、StartAgentPollers 和StartSNMPPollers。

默认值：1000
 范围：1-1000

PidFile

指定 PID 文件。

默认值：/tmp/zabbix_proxy.pid

ProxyBufferMode

指定历史数据、发现和自动注册数据的存储机制：- *disk* - 数据存储在数据库中，并从数据库上传；- *memory* - 数据存储在内存中，并从内存上传。如果缓冲区内存耗尽，旧数据将被丢弃。关闭时，缓冲区将被丢弃。- *hybrid* - proxy 缓冲区通常像内存模式一样工作，直到内存耗尽或最旧记录超过配置的年龄。如果发生这种情况，缓冲区会刷新到数据库，并像磁盘模式一样工作，直到所有数据都已上传，然后再次开始使用内存工作。关闭时，内存缓冲区会刷新到数据库。

另请参见：Proxy memory buffer。

默认值：disk
 取值：disk; memory; hybrid

ProxyConfigFrequency

设置从 Zabbix 服务器检索配置数据的频率，单位为秒。
 主动 proxy 参数。对被动 proxy 忽略（参见ProxyMode 参数）。

默认值：10
 范围：1-604800

ProxyLocalBuffer

设置本地保留数据的时长（以小时为单位），即使这些数据已经与服务器同步过。
 如果本地数据将被第三方应用程序使用，则可以使用此参数。

默认值：0
 范围：0-720

ProxyMemoryBufferAge

设置 proxy 内存缓冲区中数据的最大保留时间（秒）。
 启用后（不为零），如果 proxy 内存缓冲区中的记录已超过该时间，则会强制 proxy 缓冲区切换到数据库模式，直到所有记录都上传到服务器。此参数必须小于或等于 `ProxyOfflineBuffer` 参数。

默认值：0
 范围：0;600-864000

ProxyMemoryBufferSize

设置用于存储已收集的历史数据、发现数据和自动注册数据的共享内存缓存大小（以字节为单位）。如果启用（不为零），proxy 将在内存中保留历史数据、发现数据和自动注册数据，直到缓存已满，或者存储的记录早于定义的 `ProxyMemoryBufferAge`。此参数不能与 `ProxyLocalBuffer` 参数同时使用。

默认值：0
 范围：0;128K-2G

ProxyMode

设置 proxy 的运行模式。
0 - active 模式下的 proxy
1 - passive 模式下的 proxy
 注意，当使用 active proxy 时，(敏感的) proxy 配置数据可能会被能够访问 Zabbix 服务器 trapper 端口的第三方获取。这是因为任何人都可以伪装成 active proxy 并请求配置数据；此过程不会进行身份验证。

默认值: 0
 范围: 0-1

ProxyOfflineBuffer

设置在无法与 Zabbix 服务器连接时保留数据的时长（以小时为单位）。
 较旧的数据将会丢失。

默认值：1
 范围：1-720

服务器

如果 ProxyMode 设置为 `active`：指定 Zabbix 服务器 IP 地址或 DNS 名称（address:port），或者cluster（address:port;address2:port），用于获取配置数据并向其发送数据。
 如果未指定端口，则使用默认端口。
 集群节点必须用分号分隔。

 如果 ProxyMode 设置为 `passive`：指定 Zabbix 服务器的 IP 地址（可选使用 CIDR 表示法）或 DNS 名称。以逗号分隔的列表。
 传入连接只会接受此处列出的地址。如果启用了 IPv6 支持，则 `127.0.0.1`、`::127.0.0.1`、`::ffff:127.0.0.1` 会被视为相同。
`::/0` 将允许任何 IPv4 或 IPv6 地址。`0.0.0.0/0` 可用于允许任何 IPv4 地址。

示例：

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

必填：是

SNMPTrapperFile

指定用于将数据从 SNMP trap 守护进程传递到 proxy 的临时文件。
 必须与 `zabbix_trap_receiver.pl` 或 SNMPTT 配置文件中的设置相同。

默认值：`/tmp/zabbix_traps.tmp`

SocketDir

指定用于存储内部 Zabbix 服务所使用的 IPC 套接字的目录。

默认值：`/tmp`

SourceIP

为以下连接指定源 IP 地址：

- 到 Zabbix 服务器的出站连接
- 无 agent 连接（VMware、SSH、JMX、SNMP、Telnet 和简单检查）
- HTTP agent 连接
- 脚本监控项的 JavaScript HTTP 请求
- 预处理的 JavaScript HTTP 请求
- 到 Vault 的连接

SSHKeyLocation

指定用于 SSH 检查和操作的公钥和私钥位置。

SSLCertLocation

指定用于客户端身份验证的 SSL 客户端证书文件位置。
 此参数用于 Web 监控、与 Vault 的通信以及遥测数据源 ClickHouse。

SSLKeyLocation

指定用于客户端身份验证的 SSL 私钥文件位置。
 此参数用于 Web 监控、与 Vault 的通信以及遥测数据源 ClickHouse。

SSLCALocation

指定用于验证 SSL 服务器证书的证书颁发机构（CA）文件位置。
 请注意，此参数的值将设置为 CURLOPT_CAPATH libcurl 选项。对于 7.42.0 之前的 libcurl 版本，仅当 libcurl 编译为使用 OpenSSL 时，此参数才会生效。有关更多信息，请参阅 cURL 网页。
 此参数用于 Web 监控、SMTP 身份验证、HTTP agent 监控项、与 Vault 的通信以及遥测数据源 ClickHouse。

StartAgentPollers

设置 Zabbix agent pollers 的预 fork 实例数量。请参见MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartAPMCollectors

设置为 1 可启动 APM collector 的预派生实例（apm manager 和apm worker 进程）。

默认值：0
 范围：0-1

StartBrowserPollers

设置 browser 监控项pollers 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartDBSyncers

设置history syncers 的预 fork 实例数量。
 注意: 更改此值时请谨慎，增加该值可能弊大于利。

默认值: 4
 范围: 1-100

StartDiscoverers

设置discovery workers 的预先 fork 的实例数量。

默认值：5
 范围：0-1000

StartHTTPAgentPollers

设置HTTP agent pollers 的预 fork 实例数量。参见MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartHTTPPollers

设置HTTP pollers 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartIPMIPollers

设置IPMI pollers 的预先 fork 的实例数量。

默认值：0
 范围：0-1000

StartJavaPollers

设置Java pollers 的预 fork 实例数量。

默认值：0
 范围：0-1000

StartODBCPollers

设置ODBC pollers 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartPingers

设置ICMP pingers 的预 fork 实例数量。

默认值：1
 范围：0-1000

StartPollersUnreachable

设置用于不可达主机的 poller 的预 fork 实例数量（包括 IPMI 和 Java）。如果已启动常规、IPMI 或 Java poller，则必须至少运行一个用于不可达主机的 poller。

默认值：1
 范围：0-1000

StartPollers

设置pollers 的预 fork 实例数量。

默认值：5
 范围：0-1000

预启动预处理器

应将预启动的预处理worker 线程实例数量设置为不小于可用 CPU 核心数。如果预处理不受 CPU 限制且包含大量网络请求，则应设置更多 worker。

默认值：16
 范围：1-1000

StartSNMPPollers

设置SNMP pollers 的预 fork 实例数量。请参见MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartTelemetryQueryPollers

设置异步 Telemetry 查询poller 的预分叉实例数。请参阅MaxConcurrentChecksPerPoller。

默认值：1
 范围：0-1000

StartSNMPTrapper

设置为"1" 以启动一个SNMP trapper 进程。

默认值：0
 范围：0-1

StartTrappers

设置trappers 的预先 fork 的实例数量。
Trappers 接受来自 Zabbix sender 和 active agent 的传入连接。

默认值：5
 范围：0-1000

StartVMwareCollectors

设置预先 fork 的vmware collector 实例数量。

默认值：0
 范围：0-250

StatsAllowedIP

指定允许请求统计信息的外部 Zabbix 实例的 IP 地址（可选使用 CIDR 表示法）或 DNS 名称。使用逗号分隔的列表。
 如果未设置此参数，将不接受任何统计信息请求。
 如果启用了 IPv6 支持，则 127.0.0.1、::127.0.0.1、::ffff:127.0.0.1 会被视为相同，且 ::/0 将允许任何 IPv4 或 IPv6 地址。0.0.0.0/0 可用于允许任何 IPv4 地址。

示例：

```
StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

TelemetryProvider

使用 <provider>[;<options>] 格式设置遥测数据源，其中：

- <provider> - 受支持的遥测数据源之一：clickhouse
- <options> - 以逗号分隔的 key=value 对。值可以使用引号括起来。

ClickHouse 的选项：

- url - 设置用于连接遥测数据源的 HTTP 或 HTTPS URL。
- db - 设置数据库名称。
- username - 设置用于登录数据库的用户名。
- password - 设置用于登录数据库的密码。
- ssl_cert_file、ssl_key_file、ssl_key_password、ssl_verify_peer、ssl_verify_host、ssl_ca_location、ssl_cert_location、ssl_key_location - 设置这些选项以配置与数据库的安全（TLS/SSL）连接。
- source_ip - 设置用于向遥测数据源发起出站连接的源 IP 地址。
- vault_path - 设置保存 username 和 password 值的 Vault 密钥路径。不要与 username 和 password 选项一起使用。

ClickHouse 示例：

```
TelemetryProvider=clickhouse;url=http://localhost:8123,db=zabbix,username=zabbix,password=zabbix
```

使用 Vault 的 ClickHouse 示例：

```
TelemetryProvider=clickhouse;url=http://localhost:8123,db=zabbix,vault_path=clickhouse
```

超时

指定与 Zabbix 服务器、agent、web service 以及旧版 SNMP 检查（单个 OID 编号或字符串）建立连接并交换数据时的等待时间（秒）。

此参数定义了多种通信操作的持续时间：

- 在 Zabbix agent 上执行远程命令
- 执行 SSH / Telnet 命令
- 当 IPMI 接口变为不可用时重新调度监控项
- 在因权限或加密问题导致数据交换失败时，向 Zabbix 服务器发送响应
- IPC 异步套接字和运行时控制选项的截止时间
- 异步轮询器 DNS 请求
- 主动检查心跳的响应
- 从主动 agent 获取 Zabbix agent 数据（值）
- 从 Zabbix sender 获取数据
- 向 Zabbix agent 发送主动检查列表

对于在前端中配置了flexible timeout 设置的检查（在全局、proxy 或单个监控项级别），不会使用此超时。
例如，SNMP `walk[OID]` 和 `get[OID]` 监控项使用前端中配置的超时；旧版 SNMP 检查仍然使用服务器超时值。

默认值：3
 范围：1-30

TLSAccept

指定来自 Zabbix 服务器的传入连接所需的加密级别。用于被动 proxy，在主动 proxy 上忽略。可以指定多个值，使用逗号分隔：- *unencrypted* - 接受未加密连接（默认）；- *psk* - 接受使用 TLS 和预共享密钥（PSK）的连接；- *cert* - 接受使用 TLS 和证书的连接。

必填：对于被动 proxy，如果定义了 TLS 证书或 PSK 参数，则为必填（即使是 *unencrypted* 连接）；否则为否

TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，该文件用于 Zabbix 组件之间的加密通信。

TLSCertFile

指定包含 proxy 证书或证书链的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSCipherAll

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书和基于 PSK 的加密的默认密码套件选择标准。

示例：

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherAll13

为 OpenSSL 1.1.1 或更高版本指定 TLS 1.3 的密码套件字符串。覆盖基于证书和基于 PSK 的加密的默认密码套件选择条件。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL
```

OpenSSL 示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherCert

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码套件字符串。覆盖基于证书的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

OpenSSL 示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

TLSCipherCert13

为 TLS 1.3 指定适用于 OpenSSL 1.1.1 或更新版本的密码字符串。覆盖基于证书的加密的默认密码套件选择标准。

TLSCipherPSK

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于 PSK 的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

OpenSSL 示例：

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

为 TLS 1.3 中的 OpenSSL 1.1.1 或更新版本指定密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择条件。

示例：

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSConnect

指定 proxy 连接到 Zabbix 服务器所需的加密级别。用于主动 proxy，在被动 proxy 上忽略。只能指定一个值：- *unencrypted* - 不使用加密连接（默认）；- *psk* - 使用 TLS 和预共享密钥（PSK）连接；- *cert* - 使用 TLS 和证书连接。

必填：对于主动 proxy，如果定义了 TLS 证书或 PSK 参数，则为必填（即使是 *unencrypted* 连接）；否则为否

TLSCRLFile

指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。

TLSKeyFile

指定包含 proxy 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSListen

在 trapper 套接字上仅允许 TLS 连接。

支持的值：

- required - 仅接受 TLS 连接

TLSPSKFile

指定包含 proxy 预共享密钥的文件的完整路径名，用于与 Zabbix 服务器进行加密通信。

TLSPSKIdentity

指定预共享密钥标识字符串，用于与 Zabbix 服务器进行加密通信。

TLSServerCertIssuer

指定允许的服务器证书颁发者。

TLSServerCertSubject

指定允许的服务器证书主题。

TmpDir

指定临时目录。

默认值：`/tmp`

TrapperTimeout

指定以下操作的超时时间（秒）：
 - 从 Zabbix 服务器检索配置数据；
 - 执行全局脚本或远程命令。

默认值：300
 范围：1-300

UnavailableDelay

设置在unavailability 期间检查主机可用性的频率，单位为秒。

默认值：60
 范围：1-3600

UnreachableDelay

设置在unreachability 期间检查主机可用性的频率，单位为秒。

默认值: 15
 范围: 1-3600

UnreachablePeriod

设置在多少秒的不可达后将主机视为不可用。

默认值：45
 范围：1-3600

用户

将权限降级为系统中某个已存在的特定用户。
 仅在以 `root` 运行且 `AllowRoot` 已禁用时生效。

默认值：zabbix

Vault

指定 vault 提供程序：- *HashiCorp* - HashiCorp KV Secrets Engine 版本 2；- *CyberArk* - CyberArk Central Credential Provider
 必须与前端中设置的 vault 提供程序一致。

默认值：HashiCorp

VaultAppRoleID

指定 HashiCorp vault 的 AppRole ID。
 如果设置，将启用 AppRole 身份验证。不能与 VaultToken 一起使用。
 仅支持 HashiCorp vault。

VaultAppSecretID

指定 HashiCorp vault AppRole secret ID。
 仅支持 HashiCorp vault。

VaultDBPath

指定用于按键检索数据库凭据的位置。根据 Vault 的不同，它可以是 Vault 路径或查询。

HashiCorp 使用的键为 password 和 username。

带有 VaultPrefix=/v1/secret/data/zabbix/ 的路径示例：

```
database
```

不带 VaultPrefix 的路径示例：

```
secret/zabbix/database
```

CyberArk 使用的键为 Content 和 UserName。

示例：

```
AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_proxy_database
```

仅当未指定 DBUser 和 DBPassword 时，才能使用此选项。

VaultPrefix

为 vault 路径或查询指定自定义前缀，具体取决于 Vault。如果未指定，将使用最合适的默认值。
 请注意，如果未指定 VaultPrefix，对于 HashiCorp，data 会在挂载点之后自动追加。
 请注意，如果未指定 VaultPrefix，对于 HashiCorp，data 会在挂载点之后自动追加。

HashiCorp 的前缀示例：

```
/v1/secret/data/zabbix/
```

Cyberark 的前缀示例：

```
/AIMWebService/api/Accounts?
```

VaultTLSCertFile

指定用于客户端身份验证的 SSL 证书文件名称。证书文件必须为 PEM1 格式。
 如果证书文件中还包含私钥，请将 SSL 密钥文件字段留空。
 包含此文件的目录由 SSLCertLocation 配置参数指定。
 此选项可以省略，但建议用于 CyberArkCCP vault。

VaultTLSKeyFile

指定用于客户端身份验证的 SSL 私钥文件名称。私钥文件必须为 PEM1 格式。
 包含该文件的目录由 SSLKeyLocation 配置参数指定。
 此选项可以省略，但建议用于 CyberArkCCP vault。

VaultToken

指定应专门为 Zabbix proxy 生成的 HashiCorp vault 身份验证令牌，该令牌应仅对可选 VaultDBPath 配置参数中指定的路径具有只读权限。
 如果 VaultToken 和 VAULT_TOKEN 环境变量同时定义，则会报错。
 不能与 VaultAppRoleID 一起使用。
 仅支持 HashiCorp vault。

VaultURL

指定 vault 服务器 URL。如果未指定 SSLCALocation，则将使用系统范围的 CA 证书目录。

默认值：https://127.0.0.1:8200

VMwareCacheSize

设置用于存储 VMware 数据的共享内存大小。
 可使用 VMware 内部检查 zabbix[vmware,buffer,...] 来监控 VMware 缓存使用情况（参见内部检查）。
 请注意，如果未配置任何 vmware 收集器实例启动，则不会分配共享内存。

默认值：8M
 范围：256K-2G

VMwareFrequency

设置从单个 VMware 服务收集数据的频率（以秒为单位）。
 此频率应设置为任何 VMware 监控项中最小的更新间隔。

默认值：60
 范围：10-86400

VMwarePerfFrequency

设置从单个 VMware 服务检索性能计数器统计信息的频率（以秒为单位）。
 此频率应设置为任何使用 VMware 性能计数器的 VMware 监控监控项 的最小更新间隔。

默认值：60
 范围：10-86400

VMwareTimeout

设置 `vmware collector` 等待来自 VMware 服务（vCenter 或 ESX hypervisor）响应的最长秒数。

默认值：10
 范围：1-300

WebDriverURL

指定 WebDriver 接口 URL。

示例（与 Selenium WebDriver standalone server 一起使用）：

```
WebDriverURL=http://localhost:4444
```

agent

概述　Zabbix agent 部署在监控目标上，用于主动监控本地资源和应用程序（硬盘、内存、处理器统计信息等）。

agent 在本地收集运行信息，并将数据报告给 Zabbix 服务器进行进一步处理。如果发生故障（例如硬盘空间已满或服务进程崩溃），Zabbix 服务器可以主动向报告故障的特定机器的管理员发出警报。

由于使用本机系统调用来收集统计信息，Zabbix agent 具有很高的效率。

可以在一个主机上运行多个 agent 实例。

另请参阅：

- 被动和主动检查协议
- 以 root 身份运行 agent
- 配置参数：Zabbix agent
- 配置参数：Zabbix agent（Windows）
- Microsoft Windows 上的 Zabbix agent
- IBM AIX 上的 Zabbix agent

被动和主动检查

Zabbix agent 可以执行被动检查和主动检查：

- 被动检查 - Zabbix agent 响应来自 Zabbix 服务器（或 proxy）的请求。例如，服务器请求数据（如 CPU 负载），agent 返回结果。
- 主动检查 - Zabbix agent 在不等待来自 Zabbix 服务器（或 proxy）请求的情况下收集并发送数据。首先，它从服务器获取监控项列表（CPU 负载、可用内存等），然后收集所需数据，并定期将新值发送回服务器。

agent 检查类型通过选择相应的监控监控项类型 进行配置。Zabbix agent 处理类型为 *Zabbix agent* 或 *Zabbix agent (active)* 的监控项。

支持的平台　有关支持的平台，请参阅Requirements 页面。

Unix-like 系统上的 agent　Unix-like 系统上的 Zabbix agent 运行在被监控的主机上。

安装

Zabbix agent 可以使用以下方法之一安装在基于 Linux 的系统上：

- Zabbix 软件包 - 选择 Agent 组件（选择 Zabbix 版本、操作系统发行版和操作系统版本后），然后按照提供的说明操作；
- Zabbix 源代码 - 下载源文件，并通过使用 `--enable-agent` 选项配置源代码来编译 Zabbix agent。

Attention:
通常，32 位 Zabbix agent 可以在 64 位系统上运行，但在某些情况下可能会失败。

Note:
适用于 macOS、IBM AIX、FreeBSD、OpenBSD 和 Solaris 的预编译 Zabbix agent 二进制文件可供下载。与当前 Zabbix 服务器/proxy 版本兼容的旧版二进制文件可用于 NetBSD 和 HP-UX。有关在 IBM AIX 上安装和运行 agent 的信息，请参阅IBM AIX 上的 Zabbix agent。

如果作为软件包安装

Zabbix agent 以守护进程方式运行。
可以通过执行以下命令启动 agent：

```
systemctl start zabbix-agent
```

这适用于大多数 GNU/Linux 系统。
在其他系统上，您可能需要运行：

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

要停止、重启或检查 Zabbix agent 的状态，请使用以下命令：

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

如果从二进制压缩包安装

解压压缩包后，找到 zabbix_agentd 二进制文件和 zabbix_agentd.conf 配置文件。它们在压缩包中的位置取决于平台（例如，sbin/ 和 conf/，或 usr/sbin/ 和 etc/zabbix/）。直接从解压后的目录运行二进制文件，并使用 -c 选项指定配置文件。例如：

```
/opt/zabbix/sbin/zabbix_agentd -c /opt/zabbix/etc/zabbix/zabbix_agentd.conf
```

在类 UNIX 系统上，Zabbix agent 设计为以非 root 用户身份运行。如果以 root 身份启动，除非在配置文件中设置了 [AllowRoot=1](<PARAMS>#allowroot)，否则它会切换到 zabbix 用户（或 [User](<PARAMS>#user) 参数指定的用户）。该用户必须存在于系统中，否则 agent 不会启动。

在 IBM AIX 上，还可以通过系统资源控制器（SRC）运行 agent，并在系统启动时启动；请参阅IBM AIX 上的 Zabbix agent。

手动启动

您可以直接运行 zabbix_agentd 二进制文件来启动 Zabbix agent，例如：

```
zabbix_agentd
```

当包含 zabbix_agentd 的目录已包含在 PATH 中时，此方法有效。

Windows 系统上的 agent　Windows 上的 Zabbix agent 作为 Windows 服务运行。

安装

Zabbix agent 可以使用以下方法之一在 Windows 上安装：

- 预编译的 Zabbix agent 二进制文件 - 下载 Zabbix agent MSI 安装包，并按照从 MSI 安装 Windows agent 页面中的说明进行操作；
- Zabbix 源码 - 下载源文件，并按照在 Windows 上编译 Zabbix agent 页面中的说明进行操作。

有关将 Zabbix agent（来自 ZIP 压缩包）作为 Windows 服务安装的更多详细信息，请参见Microsoft Windows 上的 Zabbix agent 页面。

选项　以下命令行参数可与 Zabbix agent 一起使用：

参数	描述
UNIX 和 Windows agent	
-c --config <config-file>	配置文件的路径。 您可以使用此选项指定非默认配置文件。
-f --foreground	在前台运行 Zabbix agent（默认值：true）。
-p --print	打印已知监控项并退出。 请注意，如需同时返回用户参数结果，必须指定配置文件（如果该文件不在默认位置）。

参数	描述
-t --test <item key>	测试指定的监控项并退出。 请注意，如需同时返回用户参数结果，必须指定配置文件（如果该文件不在默认位置）。
-T --test-config	验证配置文件并退出。
-h --help	显示帮助信息。
-V --version	显示版本号。
仅限 **UNIX agent**	
-R --runtime-control <option>	执行管理功能。请参阅运行时控制。
仅限 **Windows agent**	
-m --multiple-agents	使用多个 agent 实例（配合 -i、-d、-s、-x 选项）。 为区分各实例的服务名称，每个服务名称都会包含指定配置文件中的 Hostname 值。
-S --startup-type <value>	设置 Zabbix Windows agent 服务的启动类型。允许的值： automatic - （默认）在 Windows 启动时自动启动服务； delayed - 在自动启动的服务完成启动后延迟启动该服务（适用于 Windows Server 2008/Vista 及更高版本）； manual - 手动启动服务（由用户或应用程序启动）； disabled - 禁用服务，使其无法由用户或应用程序启动。 您可以将此选项与 -i 选项结合使用，也可以单独使用此选项来修改已安装服务的启动类型。
-i --install	将 Zabbix Windows agent 安装为服务。
-d --uninstall	卸载 Zabbix Windows agent 服务。
-s --start	启动 Zabbix Windows agent 服务。
-x --stop	停止 Zabbix Windows agent 服务。

使用命令行参数的具体示例：

- 打印所有内置 agent 监控项及其值。
- 使用指定配置文件中定义了 mysql.ping 键的用户参数进行测试。
- 使用默认路径下的配置文件为 Windows 安装 Zabbix Agent 服务。
- 使用与 agent 可执行文件位于同一文件夹中的配置文件为 Windows 安装 Zabbix Agent [Hostname] 服务，并通过在服务名称中添加配置文件中的Hostname 值使服务名称保持唯一。
- 使用与 agent 可执行文件位于同一文件夹中的配置文件修改已安装的 Windows Zabbix Agent 服务的启动类型。

```
zabbix_agentd --print
zabbix_agentd -t "mysql.ping" -c /etc/zabbix/zabbix_agentd.conf
zabbix_agentd.exe -i
zabbix_agentd.exe -i -m -c zabbix_agentd.conf
zabbix_agentd.exe -c zabbix_agentd.conf -S delayed
```

运行时控制

使用运行时控制选项，您可以更改 agent 进程的日志级别。

选项	描述	目标
log_level_increase[<target>]	提高日志级别。 如果未指定目标，则会影响所有进程。	目标可以指定为： 进程类型 - 指定类型的所有进程（例如 listener） 请参阅所有agent 进程类型。 进程类型,**N** - 进程类型和编号（例如 listener,3） **pid** - 进程标识符（1 至 65535）。对于更大的值，请将目标指定为 “进程类型,N”。
log_level_decrease[<target>]	降低日志级别。 如果未指定目标，则会影响所有进程。	
userparameter_reload	从当前配置文件重新加载UserParameter 和Include 选项的值。	

示例：

- 提高所有进程的日志级别
- 提高第三个 listener 进程的日志级别
- 提高 PID 为 1234 的进程的日志级别
- 降低所有主动检查进程的日志级别

```
zabbix_agentd -R log_level_increase
zabbix_agentd -R log_level_increase=listener,3
zabbix_agentd -R log_level_increase=1234
zabbix_agentd -R log_level_decrease="active checks"
```

> **Note:**
> OpenBSD、NetBSD 和 Windows 不支持运行时控制。

agent 进程类型　agent 进程类型如下：

- `active checks` - 用于执行主动检查的进程；
- `collector` - 用于数据收集的进程；
- `listener` - 用于监听被动检查的进程。

可以使用 agent 日志文件来观察这些进程类型。

agent 日志文件仅为文件所有者创建读写权限。此外，该文件对所有者组可读。其他所有权限均被拒绝。

进程用户　UNIX 上的 Zabbix agent 设计为以非 root 用户运行。它将以启动时所使用的非 root 用户身份运行。因此，您可以让 agent 以任意非 root 用户身份运行，不会有任何问题。

如果您尝试以 “root” 身份运行它，它将切换为硬编码的 “zabbix” 用户，该用户必须存在于您的系统中。只有相应修改 agent 配置文件中的`AllowRoot` 参数后，您才能让 agent 以 “root” 身份运行。

配置文件　配置文件的默认路径取决于操作系统和 Zabbix 的安装方式：

- 在 UNIX 上，默认路径为：
  - 从源代码安装：`/usr/local/etc/zabbix_agentd.conf`，或由编译时变量 `--sysconfdir` 或 `--prefix` 设置的路径。
  - 从软件包安装或使用 appliance 安装：`/etc/zabbix/zabbix_agentd.conf`。
- 在 Windows 上，默认路径为 `C:\Program Files\Zabbix Agent\zabbix_agentd.conf`。

如果一台主机上运行单个 agent 实例，则可以使用默认配置文件，或使用在命令行中指定的配置文件。如果运行多个实例，则每个 agent 实例都必须拥有自己的配置文件（其中一个实例可以使用默认配置文件）。

有关配置 Zabbix agent 的详细信息，请参阅Unix Zabbix agent 或Windows Zabbix agent 的配置文件选项。

区域设置　请注意，agent 需要 UTF-8 区域设置，这样某些文本类 agent 监控项才能返回预期内容。大多数现代类 Unix 系统默认都使用 UTF-8 区域设置，不过在某些系统上，可能需要专门进行设置。

退出代码　Zabbix agent 在成功退出时返回 0，在失败时返回 1。

被动和主动检查协议

本节提供Zabbix agent 和Zabbix agent 2 执行的被动检查和主动检查的协议细节。

Zabbix 使用基于 JSON 的通信协议与 agent 进行通信。

自 Zabbix 7.0 起，Zabbix agent 和 Zabbix agent 2 的协议已统一。Zabbix agent 和 Zabbix agent 2 的请求/响应之间的差异通过”variant” 标签值来体现。

被动检查

被动检查是一种简单的数据请求。Zabbix 服务器或 proxy 会请求某些数据（例如 CPU 负载），然后 Zabbix agent 将结果返回给服务器。

被动检查是异步执行的 - 在开始其他检查之前，不需要先接收某个请求的响应。DNS 解析同样是异步的。

agent poller 会尝试连接 DNS 查询返回的所有地址。这样可以确保如果某个 IP 地址不可达，poller 会尝试下一个可用地址，从而提高连接成功的可能性。此增强适用于 Zabbix 服务器和 proxy。

异步检查的最大并发数为 1000（由MaxConcurrentChecksPerPoller 定义）。

异步 agent poller 的数量由StartAgentPollers 参数定义。

服务器请求

有关报头和数据长度的定义，请参阅协议详情。

```
{
  "request": "passive checks",
  "data": [
    {
      "key": "agent.version",
      "timeout": 3
    }
  ]
}
```

Field		Type	Mandatory	Value
request		*string*	yes	"passive checks"
data		*array of object*	yes	被动检查监控项。
	key	*string*	yes	展开宏后的监控项键值。
	timeout	*number*	yes	通信超时。

agent 响应

```
{
  "version": "8.0.0",
  "variant": 2,
  "data": [
    {
      "value": "8.0.0"
    }
  ]
}
```

Field		Type	Mandatory	Value
version		*string*	yes	agent 版本号。
variant		*number*	yes	agent 变体（*1* - Zabbix agent，*2* - Zabbix agent 2）。
data		*array of object*	yes	包含检查结果。
	value	*string*	no	如果检查成功，则为监控项值。
	error	*string*	no	如果检查不成功，则为错误消息。

例如，对于受支持的监控项：

1. 服务器打开一个 TCP 连接
2. 服务器发送 **<HEADER><DATALEN>{”request”:”passive checks”,”data”:[{”key”:”agent.ping”,”timeout”:3}]}**
3. agent 读取请求并响应：**<HEADER><DATALEN>{”version”:”8.0.0”,”variant”:2,”data”:[{”value”:1}]}**
4. 服务器处理数据以获取值，在本例中为’1’
5. TCP 连接关闭

对于不受支持的监控项：

1. 服务器打开一个 TCP 连接
2. 服务器发送 **<HEADER><DATALEN>{”request”:”passive checks”,”data”:[{”key”:”vfs.fs.size[/nono]”,”timeout”:3}]}**
3. agent 读取请求并响应：**<HEADER><DATALEN>{”version”:”8.0.0”,”variant”:2,”data”:[{”error”:”Unsupported item key.”}]}**
4. 服务器处理数据，将监控项状态更改为不受支持，并附带指定的错误消息
5. TCP 连接关闭

回退到旧协议

为确保 Zabbix 服务器或 proxy 能与 7.2 之前版本的 agent 配合工作（这些版本使用明文协议），系统实现了对旧协议的回退机制。

在重启后或接口配置发生更改时，被动检查会使用 JSON 协议（7.0 及更高版本）执行。
如果响应中未收到有效的 JSON（agent 返回”ZBX_NOTSUPPORTED”），Zabbix 会将该接口缓存为旧协议，并通过仅发送监控项键值来重试检查。

请注意，Zabbix 服务器/proxy 每小时都会再次尝试让所有接口使用新协议；如有需要，则回退到旧协议。

主动检查

主动检查需要更复杂的处理流程。agent 必须先从服务器/proxy 获取监控项和/或远程命令列表，然后独立进行处理。

用于获取主动检查的服务器/proxy 列在 agent 配置文件的ServerActive 参数中。请求这些检查的频率由同一配置文件中的RefreshActiveChecks 参数设置。但是，如果刷新主动检查失败，则会在固定的 60 秒后重试。

Note:
自 Zabbix 6.4 起，agent（主动模式）不再每两分钟（默认）从服务器/proxy 接收一次完整的配置副本。相反，为了减少网络流量和资源占用，系统会每 5 秒（默认）执行一次增量配置同步；在此过程中，只有当 agent 还未收到配置副本，或者主机配置、全局宏或全局正则表达式发生变化时，服务器/proxy 才会提供完整的配置副本。

agent 随后会定期将新值发送到服务器。如果 agent 收到了任何要执行的远程命令，执行结果也会一并发送。请注意，自 Zabbix agent 7.0 起，活动 agent 支持远程命令执行。

Note:
如果 agent 位于防火墙后面，您可以考虑仅使用主动检查，因为在这种情况下，您无需修改防火墙以允许初始入站连接。

获取监控项列表

Agent 请求

active checks 请求用于获取由 agent 处理的 active checks。
该请求由 agent 在启动时发送，之后会按照RefreshActiveChecks 间隔发送。

```
{
  "request": "active checks",
  "host": "Zabbix server",
  "host_metadata": "mysql,nginx",
  "interface": "zabbix.server.lan",
  "ip": "159.168.1.1",
  "port": 12050,
  "version": "8.0.0",
  "variant": 2,
  "config_revision": 1,
  "session": "e3dcbd9ace2c9694e1d7bbd030eeef6e"
}
```

Field	Type	Mandatory	Value
request	*string*	yes	`active checks`
host	*string*	yes	主机名。
host_metadata	*string*	no	配置参数 HostMetadata 或 HostMetadataItem 的指标值。
interface	*string*	no	配置参数 HostInterface 或 HostInterfaceItem 的指标值。
ip	*string*	no	配置参数 ListenIP 的第一个 IP（如果已设置）。
port	*number*	no	配置参数 ListenPort 的值（如果已设置且不是默认的 agent 监听端口）。
version	*string*	yes	agent 版本号。
variant	*number*	yes	agent 变体（*1* - Zabbix agent，*2* - Zabbix agent 2）。
config_revision	*number*	no	用于增量配置同步的配置标识符。
session	*string*	no	用于增量配置同步的会话标识符。

服务器响应

active checks 响应由服务器在处理 active checks 请求后返回给 agent。

```
{
  "response": "success",
  "config_revision": 2,
  "data": [
    {
      "key": "system.uptime",
      "itemid": 1234,
      "delay": "10s",
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "agent.version",
```

```
      "itemid": 5678,
      "delay": "10m",
      "lastlogsize": 0,
      "mtime": 0,
      "timeout": "30s"
    }
  ],
  "commands": [
    {
      "command": "df -h --output=source,size / | awk 'NR>1 {print $2}'",
      "id": 1324,
      "wait": 1
    }
  ]
}
```

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	no	失败时的错误信息。
data	*array of objects*	no	active check 监控项。如果主机配置未更改，则省略。
key	*string*	no	展开宏后的监控项键值。
itemid	*number*	no	监控项标识符。
delay	*string*	no	监控项更新间隔。 自 Zabbix 7.0 起，Zabbix agent 和 Zabbix agent 2 都支持灵活/计划间隔。
lastlogsize	*number*	no	监控项 lastlogsize。
mtime	*number*	no	监控项 mtime。
timeout	*string*	no	监控项超时时间。
refresh_unsupported	*number*	no	不受支持的监控项刷新间隔。
regexp	*array of objects*	no	全局正则表达式。
name	*string*	no	全局正则表达式名称。
expression	*string*	no	全局正则表达式。
expression_type	*number*	no	全局正则表达式类型。
exp_delimiter	*string*	no	全局正则表达式分隔符。
case_sensitive	*number*	no	全局正则表达式大小写敏感设置。
commands	*array of objects*	no	要执行的远程命令。如果由某个动作的操作或手动脚本执行触发了远程命令执行，则会包含此项。请注意，自 Zabbix agent 7.0 起，活动 agent 支持远程命令执行。较旧的活动 agent 会忽略 active checks 服务器响应中包含的任何远程命令。
command	*string*	no	远程命令。
id	*number*	no	远程命令标识符。
wait	*number*	no	远程命令执行模式（来自动作操作的命令为"0"（nowait）；来自手动脚本执行的命令为"1"（wait））。
timeout	*number*	no	服务器/proxy 配置中的远程命令执行超时。
config_revision	*number*	no	用于增量配置同步的配置标识符。如果主机配置未更改，则省略。若主机配置发生更改，则递增。

服务器必须返回 success。

例如：

1. Agent 打开一个 TCP 连接
2. Agent 请求监控项列表
3. 服务器返回监控项列表和要执行的远程命令
4. Agent 解析响应
5. TCP 连接关闭
6. Agent 开始周期性采集数据并执行远程命令（自 Zabbix agent 7.0 起支持）

Attention:
请注意，使用 active check 时，具有访问 Zabbix 服务器 trapper 端口权限的人员可能会获取（敏感的）配置数据。
这是因为任何人都可以伪装成活动 agent 并请求监控项配置数据；除非使用加密选项，否则不会进行身份验证。

发送收集到的数据

Agent 发送

agent 数据请求包含已收集的监控项值以及已执行远程命令的值（如果有）。

```
{
  "request": "agent data",
  "data": [
    {
      "id": 1,
      "itemid": 5678,
      "value": "7.0.0",
      "clock": 1712830783,
      "ns": 76808644
    },
    {
      "id": 2,
      "itemid": 1234,
      "value": "69672",
      "clock": 1712830783,
      "ns": 77053975
    }
  ],
  "commands": [
    {
      "id": 1324,
      "value": "16G"
    }
  ],
  "session": "8495cd52070e6ca52b371f29c8574165",
  "host": "Zabbix server",
  "version": "8.0.0",
  "variant": 2
}
```

Field	Type	Mandatory	Value
request	*string*	yes	`agent data`
data	*array of objects*	yes	监控项值。
id	*number*	yes	值标识符（递增计数器，用于在网络问题情况下检查重复值）。
itemid	*number*	yes	监控项标识符。
value	*string*	no	监控项值。
lastlogsize	*number*	no	监控项 lastlogsize。
mtime	*number*	no	监控项 mtime。
state	*number*	no	监控项状态。
source	*string*	no	值事件日志来源。
eventid	*number*	no	值事件日志 eventid。
severity	*number*	no	值事件日志严重性。
timestamp	*number*	no	值事件日志时间戳。
clock	*number*	yes	值时间戳（自 Epoch 起的秒数）。
ns	*number*	yes	值时间戳的纳秒数。
commands	*array of objects*	no	远程命令执行结果。请注意，自 Zabbix agent 7.0 起，活动 agent 支持远程命令执行。较早版本的活动 agent 会忽略活动检查服务器响应中包含的任何远程命令。
id	*number*	no	远程命令标识符。
value	*string*	no	如果执行成功，则为远程命令执行结果。
error	*string*	no	如果执行失败，则为远程命令执行错误消息。

Field	Type	Mandatory	Value
session	*string*	yes	每次 agent 启动时生成的唯一会话标识符（32 位十六进制数字）。
host	*string*	yes	主机名。
version	*string*	yes	agent 版本号。
variant	*number*	yes	agent 变体（*1* - Zabbix agent，*2* - Zabbix agent 2）。

每个值都会分配一个虚拟 ID。值 ID 是一个简单的递增计数器，在一个数据会话内唯一（由会话令牌标识）。该 ID 用于丢弃在网络连接不佳的环境中可能发送的重复值。

服务器响应

agent data 响应由服务器在处理 agent data 请求后返回给 agent。

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

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	yes	监控项处理结果。

Attention:
如果服务器上某些值发送失败（例如，主机或监控项已被禁用或删除），agent 不会重试发送这些值。

例如：

1. Agent 打开一个 TCP 连接
2. Agent 发送一个值列表
3. 服务器处理数据并返回状态
4. TCP 连接关闭

Attention:
错误消息将在服务器端被截断为 2048 个字符。

心跳消息

agent 发送

心跳消息由 active agent 每隔 `HeartbeatFrequency` 秒发送到 Zabbix 服务器/proxy（在Zabbix agent 或Zabbix agent 2 配置文件中配置）。

它用于监控主动检查的可用性。

```
{
  "request": "active check heartbeat",
  "host": "Zabbix server",
  "heartbeat_freq": 60,
  "version": "8.0.0",
  "variant": 2
}
```

字段	类型	必填	值
request	*string*	是	`active check heartbeat`
host	*string*	是	主机名称。
heartbeat_freq	*number*	是	agent 心跳频率（HeartbeatFrequency 配置参数）。
version	*string*	是	agent 版本号。
variant	*number*	是	agent 变体（*1* - Zabbix agent，*2* - Zabbix agent 2）。

重定向响应

当主机被重新分配后，服务器可能会指示 agent 将其心跳消息（以及后续的主动检查）重定向到另一个 proxy 或服务器实例。

```
{
  "response": "failed",
  "redirect": {
    "revision": 2,
    "address": "192.0.2.1:10055"
  }
}
```

字段		类型	必填	值
response		*string*	是	`success` | `failed`
redirect		*object*	是	重定向指令。
	revision	*number*	是	配置修订标识符。
	address	*string*	是	目标服务器/proxy 地址。

较旧的 XML 协议

Note:
Zabbix 将接受最大 16 MB 的 XML Base64 编码数据，但单个解码后的值不应超过 64 KB；否则，在解码时将被截断为 64 KB。

以 **root** 身份运行 **agent**

自 Zabbix **5.0.0** 起，官方软件包中的 Zabbix agent 的 systemd 服务文件会显式包含 `User` 和 `Group` 指令。
这两个指令都设置为 `zabbix`。

现在已经无法通过 `zabbix_agentd.conf` 文件来配置 Zabbix agent 以哪个用户身份运行，
因为 agent 会忽略此配置，并以 systemd 服务文件中指定的用户身份运行。
若要让 Zabbix agent 以 root 身份运行，需要进行下面所述的修改。

Zabbix agent

要覆盖 Zabbix agent 的默认用户和组，请运行：

```
systemctl edit zabbix-agent
```

然后添加以下内容：

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

重新加载守护进程并重启 zabbix-agent 服务：

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

对于 **Zabbix agent**，这将重新启用在agent 配置文件 中配置用户的功能。现在，您需要在配置文件中设置 `User=root` 和 `AllowRoot=1` 配置参数。

Zabbix agent 2

要覆盖 Zabbix agent 2 的默认用户和组，请运行：

```
systemctl edit zabbix-agent2
```

然后，添加以下内容：

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

重新加载守护进程并重启 zabbix-agent2 服务：

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

对于 **Zabbix agent2**，这将完全决定其运行所使用的用户。无需进行其他修改。

配置参数：**Zabbix agent**

概述

配置参数可用于自定义 Zabbix agent 的行为。

在 Zabbix agent 配置文件（`zabbix_agentd.conf`）中指定参数值。除非明确说明参数为必需参数，否则所有参数均为可选。

本页面的结构如下：

- 参数概览（要查看参数的完整详细信息，请单击参数名称）
- 参数详细信息

参数	描述
Alias	为监控项键设置别名。
AllowKey	允许执行与某个模式匹配的监控项键。
AllowKeyRegexp	允许执行与基于正则表达式的模式匹配的监控项键。
AllowRoot	允许 agent 以 `root` 身份运行。
BufferSend	设置将缓冲区中的值发送到 Zabbix 服务器/proxy 的频率。
BufferSize	设置内存缓冲区中的最大值数量。
DebugLevel	设置调试级别。
DenyKey	拒绝执行与某个模式匹配的监控项键。
DenyKeyRegexp	拒绝执行与基于正则表达式的模式匹配的监控项键。
EnableRemoteCommands	允许来自 Zabbix 服务器的远程命令。此参数已弃用；请改用 `AllowKey`、`AllowKeyRegexp`、`DenyKey` 或 `DenyKeyRegexp`。
HeartbeatFrequency	设置心跳消息的发送频率（秒）。
HostInterface	指定主机接口。
HostInterfaceItem	指定用于获取主机接口的监控项。
HostMetadata	指定主机元数据。
HostMetadataItem	指定用于获取主机元数据的 Zabbix agent 监控项。
Hostname	指定主机名。
HostnameItem	指定用于获取主机名的 Zabbix agent 监控项。
Include	指定要包含在配置文件中的目录或单个文件。
ListenBacklog	设置 TCP 队列中待处理连接的最大数量。
ListenIP	指定用于接收来自服务器/proxy 连接的 agent 监听 IP 地址。
ListenPort	指定 agent 监听端口。
LoadModule	指定 agent 启动时要加载的模块。
LoadModulePath	指定 agent 模块所在位置的完整路径。
LogFile	指定 agent 日志文件。
LogFileSize	设置日志文件的最大大小。
LogRemoteCommands	启用将已执行的 shell 命令作为警告进行记录。
LogType	指定日志输出类型。
MaxLinesPerSecond	处理 `log` 或 `logrt` 主动检查时，设置每秒发送到 Zabbix 服务器/proxy 的新行最大数量。
PidFile	指定 PID 文件。
RefreshActiveChecks	指定刷新主动检查列表的频率。
Server	指定 Zabbix 服务器/proxy 的 IP 地址或 DNS 名称。
ServerActive	指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
SourceIP	指定源 IP 地址。
StartAgents	设置用于处理被动检查的预派生 `zabbix_agentd` 实例数量。
Timeout	指定建立连接以及与 Zabbix proxy 或服务器交换数据时的等待时间（秒）。
TLSAccept	指定传入连接所需的加密级别。
TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，该文件用于 Zabbix 组件之间的加密通信。
TLSCertFile	指定包含 agent 证书或证书链的文件的完整路径名，该文件用于 Zabbix 组件之间的加密通信。
TLSCipherAll	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherAll13	指定 TLS 1.3 中适用于 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherCert	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherCert13	指定 TLS 1.3 中适用于 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherPSK	指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。

参数	描述
TLSCipherPSK13	指定 TLS 1.3 中适用于 OpenSSL 1.1.1 或更高版本的密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。
TLSConnect	指定连接到 Zabbix 服务器或 proxy 时所需的加密级别。
TLSCRLFile	指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。
TLSKeyFile	指定包含 agent 私钥的文件的完整路径名，该私钥用于 Zabbix 组件之间的加密通信。
TLSPSKFile	指定包含 agent 预共享密钥的文件的完整路径名，该密钥用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSPSKIdentity	指定预共享密钥标识字符串，该字符串用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSServerCertIssuer	指定允许的服务器（proxy）证书颁发者。
TLSServerCertSubject	指定允许的服务器（proxy）证书主题。
UnsafeUserParameters	允许将所有字符传递到用户定义参数的参数中。
User	将权限降级为系统中指定的现有用户。
UserParameter	指定要监控的用户定义参数。
UserParameterDir	指定 UserParameter 命令的默认搜索路径。

请注意：

- 默认值反映的是守护进程默认值，而不是随附配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持使用 UTF-8 编码且不带 BOM 的配置文件。
- 以 # 开头的注释仅支持出现在行首。

参数详情

别名

为监控项键设置别名。可使用别名将较长且复杂的监控项键替换为更短、更简单的键。
 可以存在多个 Alias 参数。不允许存在多个具有相同 Alias 键的参数。
 不同的 Alias 键可以引用同一个监控项键。
 别名可以在 HostMetadataItem 中使用，但不能在 HostnameItem 参数中使用。

示例 1：获取用户 zabbix 的 ID。

```
Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)",,,,\1]
```

现在可以使用 zabbix.userid 这一简写键来获取数据。

示例 2：使用默认参数和自定义参数获取 CPU 利用率。

```
Alias=cpu.util:system.cpu.util
Alias=cpu.util[*]:system.cpu.util[*]
```

这样，您可以使用 cpu.util 键，在使用默认参数的情况下获取 CPU 利用率百分比；也可以使用 cpu.util[all, idle, avg15] 获取有关 CPU 利用率的特定数据。

示例 3：运行多个处理相同发现监控项的低级别发现规则。

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

现在可以使用 vfs.fs.discovery 设置多个发现规则，并为每个规则使用不同的参数，例如 vfs.fs.discovery[foo]、vfs.fs.discovery[bar] 等。

AllowKey

允许执行与某个模式匹配的监控项键值。该键值模式是一个通配符表达式，支持使用 * 字符匹配任意数量的任意字符。
 可以将多个键值匹配规则与 DenyKey、AllowKeyRegexp 和 DenyKeyRegexp 组合定义。这些参数会按照它们出现的顺序逐个处理。该顺序决定了哪个匹配规则具有优先级。另请参见：限制 agent 检查。

AllowKeyRegexp

允许执行与模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 DenyKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

示例：

```
AllowKeyRegexp=^system\.run\[(?:/usr)?/bin/hostname(?: -[fsd])?\]$
```

AllowRoot

允许 agent 以 root 身份运行。如果禁用且 agent 由 root 启动，agent 将尝试切换到用户 zabbix。如果在普通用户下启动，则无效。

默认值：0
 取值：0 - 不允许；1 - 允许

BufferSend

设置将缓冲区中的值发送到 Zabbix 服务器/proxy 的频率（单位：秒）。

默认值：5
 范围：1-3600

BufferSize

设置内存缓冲区中值的最大数量。
 如果缓冲区已满，agent 将把所有已收集的数据发送到 Zabbix 服务器或 proxy。

默认值：100
 范围：2-65535

DebugLevel

设置调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。

默认值：3
 范围：0-5

DenyKey

拒绝执行与模式匹配的监控项键值。键值模式是通配符表达式，支持使用 * 字符匹配任意数量的任意字符。
 可以结合 AllowKey、AllowKeyRegexp 和 DenyKeyRegexp 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

默认值：DenyKey=system.run[*]

DenyKeyRegexp

拒绝执行与某个模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 AllowKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现顺序逐一处理。该顺序决定哪个匹配规则具有优先级。
 默认情况下，所有 system.run[] 监控项均被拒绝（DenyKey=system.run[*]），除非指定了其他 system.run[*] 规则。

另请参阅：限制 agent 检查。

EnableRemoteCommands

允许来自 Zabbix 服务器的远程命令。
 此参数已弃用，请改用 AllowKey=system.run[*]、AllowKeyRegexp=^system\.run\[.*\]$、DenyKey=system.run[*] 或 DenyKeyRegexp=^system\.run\[.*\]$。
 根据取值，它是 AllowKey、AllowKeyRegexp、DenyKey 和 DenyKeyRegexp 参数的内部别名：- 0 - DenyKey=system.run[*]（DenyKeyRegexp=^system\.run\[.*\]$）- 1 - AllowKey=system.run[*]（AllowKeyRegexp=^system\.run\[.*\]$）

默认值：0
 取值：0 - 不允许，1 - 允许

HeartbeatFrequency

设置心跳消息的发送频率，单位为秒。用于监控主动检查的可用性。
0 - 禁用心跳消息。

默认值：60
 范围：0-3600

HostInterface

指定在主机autoregistration 期间使用的主机接口（IP 地址或 DNS 名称）。此值将用于填充新创建的主机中的接口，并允许显式配置 IP 或 DNS 地址。有关更多详细信息，请参见Using DNS as default interface。

如果未定义，则该值将从 HostInterfaceItem 获取。

如果该值超过 255 个字符的限制，agent 将报错且不会启动。

范围：0-255 个字符

HostInterfaceItem

指定一个监控项，用于在主机自动注册期间确定主机接口（IP 地址或 DNS 名称）。仅当未定义 HostInterface 时，才会使用此值。有关更多详细信息，请参阅使用 DNS 作为默认接口。

在自动注册请求期间，如果指定监控项返回的值超过 255 个字符，agent 将记录一条警告消息。

无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

HostMetadata

指定用于在主机自动注册期间（主动 agent）识别或区分主机的元数据。
HostMetadata 可用于在主机名之外进一步区分主机。

如果未定义该参数，将从 HostMetadataItem 获取其值。

如果指定的值超过 2034 字节或不是 UTF-8 字符串，agent 将报错并且不会启动。
当参数要求使用 IP 地址或 DNS 名称时，虽然值是有效的 UTF-8 字符串，但如果不是有效的 IP 地址或 DNS 名称，也会被拒绝并报告为无效。

不支持多行元数据——输出将在第一个换行符处被截断。

范围：0-2034 字节

HostMetadataItem

指定一个 Zabbix agent 监控项，以获取主机元数据。仅当未定义 HostMetadata 时，才会使用此选项。

每次尝试自动注册时都会获取 HostMetadataItem 的值，并且仅在主机自动注册过程中使用（主动 agent）。HostMetadataItem 可用于根据主机名以外的信息区分主机。

支持用户参数和别名。无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

在自动注册请求期间，如果指定监控项返回的值超过 65535 个 UTF-8 码点，agent 将记录警告消息。监控项返回的值必须是 UTF-8 字符串，否则将被忽略。当参数要求 IP 地址或 DNS 名称时，虽然值是有效的 UTF-8 字符串，但如果不是有效的 IP 地址或 DNS 名称，也会被拒绝并报告为无效。

不支持多行元数据——输出将在第一个换行符处被截断。

主机名

指定唯一且区分大小写的主机名。以逗号分隔的列表。
 用于主动检查，并且必须与服务器上配置的主机名匹配。
 如果未定义，则从 HostnameItem 获取该值。
 允许的字符：字母数字字符、'.'、' '、'_' 和'-'。每个主机名的最大长度为 128 个字符，整行最大长度为 2048 个字符。

默认值：由 HostnameItem 设置

HostnameItem

指定一个用于获取主机名的 Zabbix agent 监控项。
 仅当未定义 Hostname 时使用此选项。
 不支持用户参数或别名，但无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 的值如何，均支持system.run[] 监控项。

默认值：system.hostname

包含

指定要包含到配置文件中的目录或单个文件。
 在 Zabbix 安装期间，除非你在编译时更改此参数，否则该参数会被设置为 /usr/local/etc。

Include 参数的一般规则：

- 路径可以是相对于 zabbix_agentd.conf 文件位置的相对路径。
- 你可以在一个配置文件中指定多个 Include 参数。
- Zabbix 会在读取到 Include 参数后立即处理它，因此建议将其放在配置文件末尾。

被包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，该组件将无法启动。

包含目录或文件模式的规则：

- 不要假设包含顺序是固定的（例如，文件不会按字母顺序包含）。
- 不要在多个被包含文件中定义同一个参数（例如，用特定设置覆盖通用设置），因为你无法预测最终会使用哪个值。
- 注意某些文本编辑器会自动创建的备份副本(例如，编辑 include/my_specific.conf 可能会创建 include/my_specific.conf.BAK)
  因为这两个文件都会被包含。

若只想从目录中包含相关文件，请使用通配符（*）进行模式匹配。

示例：

```
Include=/absolute/path/to/config/files/*.conf
```

ListenBacklog

设置 TCP 队列中待处理连接的最大数量。
 默认值是一个硬编码常量，具体取决于系统。
 支持的最大值也取决于系统，过高的值可能会被静默截断为 “implementation-specified maximum”。

默认值: SOMAXCONN
 范围: 0 - INT_MAX

ListenIP

指定 agent 用于接收来自服务器/proxy 连接的监听 IP 地址。使用逗号分隔的列表。

默认值：0.0.0.0

ListenPort

指定 agent 用于接收来自服务器/proxy 连接的监听端口。

默认值：10050
 范围：1024-32767

LoadModule

指定在 agent 启动时加载的模块。
 模块用于扩展 agent 的功能。
 该模块必须位于 LoadModulePath 指定的目录中，或者路径必须位于模块名称之前。如果前面的路径是绝对路径（以 / 开头），则会忽略 LoadModulePath。
 格式：- LoadModule=<module.so> - LoadModule=<path/module.so> - LoadModule=</abs_path/module.so>

允许包含多个 LoadModule 参数。

LoadModulePath

指定 agent 模块所在位置的完整路径。
 默认值取决于编译选项。

LogFile

指定 agent 日志文件。

必填：是，如果 LogType 设置为 file；否则不是

LogFileSize

设置最大日志文件大小（以 MB 为单位）。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始。

默认值：1
 范围：0-1024

LogRemoteCommands

启用将已执行的 shell 命令记录为警告。
 仅当命令是远程执行时才会记录。
 如果 system.run[] 由 HostMetadataItem、HostInterfaceItem 或 HostnameItem 参数在本地启动，则不会创建日志条目。

默认值：0
 取值：0 - 禁用，1 - 启用

LogType

指定日志输出类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *system* - 将日志写入 syslog；- *console* - 将日志写入标准输出。

默认值：file

MaxLinesPerSecond

设置在处理 log 或 logrt 主动检查时，每秒发送到 Zabbix 服务器/proxy 的最大新行数。
 所提供的值将被 log 或 logrt 监控项键中的 maxlines 参数覆盖。
 注意: Zabbix 处理的新行数将是 MaxLinesPerSecond 中设置值的 10 倍，以便在日志监控项中查找所需字符串。

默认值: 20
 范围: 1-1000

PidFile

指定 PID 文件。

默认值：/tmp/zabbix_agentd.pid

RefreshActiveChecks

指定刷新主动检查列表的频率，单位为秒。
 请注意，在刷新主动检查失败后，下一次刷新将于 60 秒后尝试。

默认值：5
 范围：1-86400

服务器

指定 Zabbix 服务器/proxy 的 IP 地址，必要时可使用 CIDR 表示法，或 DNS 名称。以逗号分隔的列表。
 仅接受来自此处列出的主机的传入连接。如果启用了 IPv6 支持，则 127.0.0.1、::127.0.0.1、::ffff:127.0.0.1 会被视为相同，::/0 将允许任何 IPv4 或 IPv6 地址。0.0.0.0/0 可用于允许任何 IPv4 地址。请注意，“IPv4-compatible IPv6 addresses”（0000::/96 前缀）受支持，但已被 RFC4291 弃用。允许使用空格。

示例：

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

必填：是，如果未显式将 StartAgents 设置为 0

ServerActive

指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
 服务器/proxy 地址是 IP 地址或 DNS 名称，并可选端口，二者之间用冒号分隔。
 集群配置是一个或多个服务器或 proxy 组成员地址，地址之间用分号分隔。可以指定多个 Zabbix 服务器/集群和 Zabbix proxy，彼此之间用逗号分隔。除非使用 proxy 组，否则每个 Zabbix 服务器/集群不应指定多个 Zabbix proxy。如果指定了 Zabbix proxy，则不应再指定该 proxy 对应的 Zabbix 服务器/集群。
 可以提供多个以逗号分隔的地址，以并行使用多个独立的 Zabbix 服务器。允许使用空格。
 如果未指定端口，则使用默认端口。
 如果为该主机指定了端口，则 IPv6 地址必须用方括号括起来。如果未指定端口，则 IPv6 地址的方括号可选。
 如果未指定此参数，则会禁用主动检查。

Zabbix proxy 示例：

```
ServerActive=127.0.0.1:10051
```

Zabbix proxy 组示例：

```
ServerActive=proxy1.example.com;proxy2.example.com;proxy3.example.com;proxy4.example.com;proxy5.example.co
```

多个服务器示例：

```
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
```

高可用性示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

包含两个集群和一个服务器的高可用性示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

SourceIP

为以下情况指定源 IP 地址：

- 到 Zabbix 服务器或 Zabbix proxy 的出站连接；
- 在执行某些监控项（`web.page.get`、`net.tcp.port` 等）时建立连接。

StartAgents

设置用于处理被动检查的 `zabbix_agentd` 预先 fork 的实例数量。
 如果设置为 0，则会禁用被动检查，agent 将不会监听任何 TCP 端口。

默认值：10
 范围：0-100

超时

指定建立连接以及与 Zabbix proxy 或服务器交换数据时的等待时间（秒）。

此参数定义了各种通信操作的持续时间，包括：

- 等待来自 Zabbix 服务器的响应；
- 向 Zabbix 服务器发送请求，包括监控项配置请求以及主动检查 中的监控项数据；
- 通过 logfile 获取日志数据；
- 发送心跳消息；
- `vfs.*` 检查的最长持续时间；
- 供 Zabbix agent 模块使用；
- 在服务器或 proxy 版本早于 7.0 且发送不带超时的检查时，作为回退值。

对于那些在前端中可配置超时设置的 agent 检查（全局、proxy 或单个监控项级别），此超时不会被使用。

默认值：3
 范围：1-30

TLSAccept

指定传入连接所需的加密级别。
 用于被动检查。
 可以指定多个值，并用逗号分隔：- *unencrypted* - 接受未加密连接（默认）；- *psk* - 接受使用 TLS 和预共享密钥（PSK）的连接；- *cert* - 接受使用 TLS 和证书的连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，该证书用于 Zabbix 组件之间的加密通信。

TLSCertFile

指定包含 agent 证书或证书链的文件的完整路径名，用于与 Zabbix 组件之间的加密通信。

TLSCipherAll

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。

示例：

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherAll13

为 TLS 1.3 中的 OpenSSL 1.1.1 或更新版本指定密码字符串。覆盖基于证书和基于 PSK 的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL
```

OpenSSL 示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherCert

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于证书的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

OpenSSL 示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

TLSCipherCert13

为 TLS 1.3 指定 OpenSSL 1.1.1 或更新版本的密码字符串。覆盖基于证书的加密的默认密码套件选择标准。

TLSCipherPSK

指定 GnuTLS 优先级字符串或 OpenSSL（TLS 1.2）密码字符串。覆盖基于 PSK 的加密的默认密码套件选择标准。

GnuTLS 示例：

```
NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

OpenSSL 示例：

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

为 TLS 1.3 中的 OpenSSL 1.1.1 或更新版本指定密码字符串。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。

示例：

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSConnect

指定 agent 连接到 Zabbix 服务器或 proxy 所需的加密级别。
 用于主动监控项。
 只能指定一个值：- *unencrypted* - 以未加密方式连接（默认）；- *psk* - 使用 TLS 和预共享密钥（PSK）连接；- *cert* - 使用 TLS 和证书连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCRLFile

指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。

TLSKeyFile

指定包含 agent 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSPSKFile

指定包含 agent 预共享密钥的文件的完整路径名，该密钥用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSPSKIdentity

指定预共享密钥标识字符串，用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSServerCertIssuer

指定允许的服务器（proxy）证书颁发者。

TLSServerCertSubject

指定允许的服务器（proxy）证书主题。

UnsafeUserParameters

允许在参数中传递所有字符到用户自定义参数。以下字符不允许使用：\ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ %
 此外，不允许使用换行符。

默认值：0
 取值：0 - 不允许，1 - 允许

用户

将权限降级为系统中某个已存在的特定用户。
 仅在以 root 运行且 AllowRoot 已禁用时生效。

默认值：zabbix

UserParameter

指定一个用户自定义参数用于监控。可以有多个用户自定义参数。
 格式：UserParameter=<key>,<shell command>
 请注意，shell 命令不能只返回空字符串或仅返回 EOL。如果指定了 UserParameterDir 参数，shell 命令可以使用相对路径。

示例：

```
UserParameter=system.test,who|wc -l
UserParameter=check_cpu,./custom_script.sh
```

UserParameterDir

指定 UserParameter 命令的默认搜索路径。如果使用此参数，agent 会在执行命令之前将其工作目录更改为此处指定的目录。因此，UserParameter 命令可以使用相对的 ./ 前缀，而不必使用完整路径。
 只允许一条配置项。

示例：

```
UserParameterDir=/opt/myscripts
```

另请参阅

1. 从 2.0.0 版本开始，Zabbix agent 的主动和被动检查配置差异

配置参数：**Zabbix agent**（**Windows**）

概述

配置参数可用于自定义 Windows 上 Zabbix agent 的行为。

在 Zabbix agent 配置文件（zabbix_agentd.conf）中指定参数值。除非明确说明某个参数为必需参数，否则所有参数均为可选参数。

本页面结构如下：

- 参数概览（单击参数名称可查看完整详细信息）。
- 参数详细信息。

参数	描述
Alias	为监控项键设置别名。
AllowKey	允许执行与某个模式匹配的监控项键。
AllowKeyRegexp	允许执行与基于正则表达式的模式匹配的监控项键。
BufferSend	设置将缓冲区中的值发送到 Zabbix 服务器/proxy 的频率。
BufferSize	设置内存缓冲区中的最大值数量。
DebugLevel	设置调试级别。
DenyKey	拒绝执行与某个模式匹配的监控项键。
DenyKeyRegexp	拒绝执行与基于正则表达式的模式匹配的监控项键。
EnableRemoteCommands	允许来自 Zabbix 服务器的远程命令。此参数已弃用；请改用 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp。
HeartbeatFrequency	设置心跳消息的发送频率（秒）。
HostInterface	指定主机接口。
HostInterfaceItem	指定用于获取主机接口的监控项。
HostMetadata	指定主机元数据。
HostMetadataItem	指定用于获取主机元数据的 Zabbix agent 监控项。
Hostname	指定主机名。
HostnameItem	指定用于获取主机名的 Zabbix agent 监控项。
Include	指定要包含在配置文件中的目录或单个文件。
ListenBacklog	设置 TCP 队列中待处理连接的最大数量。
ListenIP	指定用于接收来自服务器/proxy 的连接的 agent 监听 IP 地址。
ListenPort	指定 agent 监听端口。

参数	描述
LogFile	指定 agent 日志文件。
LogFileSize	设置日志文件的最大大小。
LogRemoteCommands	启用将已执行的 shell 命令记录为警告。
LogType	指定日志输出类型。
MaxLinesPerSecond	在处理 log、logrt 或 eventlog 主动检查时，设置每秒发送到 Zabbix 服务器/proxy 的新行最大数量。
PerfCounter	定义一个新参数 <parameter_name>，其值为指定时间段 <period>（秒）内系统性能计数器 <perf_counter_path> 的平均值。
PerfCounterEn	定义一个新参数 <parameter_name>，其值为指定时间段 <period>（秒）内系统性能计数器 <perf_counter_path> 的平均值。与 PerfCounter 相比，性能计数器路径必须使用英文。
RefreshActiveChecks	指定刷新主动检查列表的频率。
Server	指定 Zabbix 服务器/proxy 的 IP 地址或 DNS 名称。
ServerActive	指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
SourceIP	指定源 IP 地址。
StartAgents	设置用于处理被动检查的 zabbix_agentd 预派生实例数量。
Timeout	指定建立连接以及与 Zabbix 服务器或 proxy 交换数据时的等待时间（秒）。
TLSAccept	指定传入连接所需的加密级别。
TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，该文件用于 Zabbix 组件之间的加密通信。
TLSCertFile	指定包含 agent 证书或证书链的文件的完整路径名，该文件用于 Zabbix 组件之间的加密通信。
TLSConnect	指定连接到 Zabbix 服务器或 proxy 时所需的加密级别。
TLSCRLFile	指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。
TLSKeyFile	指定包含 agent 私钥的文件的完整路径名，该文件用于 Zabbix 组件之间的加密通信。
TLSPSKFile	指定包含 agent 预共享密钥的文件的完整路径名，该文件用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSPSKIdentity	指定预共享密钥标识字符串，该字符串用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSServerCertIssuer	指定允许的服务器（proxy）证书颁发者。
TLSServerCertSubject	指定允许的服务器（proxy）证书主题。
UnsafeUserParameters	允许将所有字符传递到用户定义参数的参数中。
UserParameter	指定要监控的用户定义参数。
UserParameterDir	指定 UserParameter 命令的默认搜索路径。

请注意：

- 默认值反映的是守护进程默认值，而不是随附配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持使用 UTF-8 编码且不带 BOM 的配置文件。
- 以 # 开头的注释仅支持出现在行首。

参数详情

别名

为监控项键设置别名。可使用别名将较长且复杂的监控项键替换为更短、更简单的键。
 可以存在多个 Alias 参数。不允许存在多个具有相同 Alias 键的参数。
 不同的 Alias 键可以引用同一个监控项键。
 别名可以在 HostMetadataItem 中使用，但不能在 HostnameItem 或 PerfCounter 参数中使用。

示例 1：从服务器获取分页文件使用率（百分比）。

```
Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage]
```

现在可以使用简写键 pg_usage 获取数据。

示例 2：使用默认参数和自定义参数获取 CPU 负载。

```
Alias=cpu.load:system.cpu.load
Alias=cpu.load[*]:system.cpu.load[*]
```

这样，您可以使用 cpu.load 键获取使用默认参数的 CPU 负载，也可以使用 cpu.load[percpu,avg15] 获取有关 CPU 负载的特定数据。

示例 3：运行多个处理相同发现监控项的低级别发现规则。

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

现在可以使用 vfs.fs.discovery 设置多个发现规则，并为每个规则使用不同的参数，例如 vfs.fs.discovery[foo]、vfs.fs.discovery[bar] 等。

AllowKey

允许执行与某个模式匹配的监控项键值。该键值模式是一个通配符表达式，支持使用 * 字符匹配任意数量的任意字符。
 可以将多个键值匹配规则与 DenyKey、AllowKeyRegexp 和 DenyKeyRegexp 组合定义。这些参数会按照它们出现的顺序逐一处理。该顺序决定了哪个匹配规则具有优先级。另请参见：限制 agent 检查。

AllowKeyRegexp

允许执行与模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 DenyKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

示例：

```
AllowKeyRegexp=^system\.run\[(?:/usr)?/bin/hostname(?: -[fsd])?\]$
```

BufferSend

设置将缓冲区中的值发送到 Zabbix 服务器/proxy 的频率（单位：秒）。

默认值：5
 范围：1-3600

BufferSize

设置内存缓冲区中的最大值数量。
 如果缓冲区已满，agent 将把所有已收集的数据发送到 Zabbix 服务器或 proxy。

默认值：100
 范围：2-65535

DebugLevel

设置调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。

默认值：3
 范围：0-5

DenyKey

拒绝执行与模式匹配的监控项键值。键值模式是支持使用 * 字符匹配任意数量任意字符的通配符表达式。
 可以结合 AllowKey、AllowKeyRegexp 和 DenyKeyRegexp 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定了哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

默认值：DenyKey=system.run[*]

DenyKeyRegexp

拒绝执行与某个模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 AllowKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。
 默认情况下，所有 system.run[] 监控项都会被拒绝（DenyKey=system.run[*]），除非指定了其他 system.run[*] 规则。

另请参阅：限制 agent 检查。

EnableRemoteCommands

允许来自 Zabbix 服务器的远程命令。
 此参数已弃用，请改用 AllowKey=system.run[*]、AllowKeyRegexp=^system\.run\[.*\]$、DenyKey=system.run[*] 或 DenyKeyRegexp=^system\.run\[.*\]$。
 根据取值的不同，它是 AllowKey、AllowKeyRegexp、DenyKey 和 DenyKeyRegexp 参数的内部别名：- 0 - DenyKey=system.run[*]（DenyKeyRegexp=^system\.run\[.*\]$）- 1 - AllowKey=system.run[*]（AllowKeyRegexp=^system\.run\[.*\]$）

默认值：0
 取值：0 - 不允许，1 - 允许

HeartbeatFrequency

设置心跳消息的发送频率，单位为秒。用于监控主动检查的可用性。
0 - 禁用心跳消息。

默认值：60
 范围：0-3600

HostInterface

指定在主机自动注册 期间使用的主机接口（IP 地址或 DNS 名称）。此值将用于填充新创建主机中的接口，并允许显式配置 IP 或 DNS 地址。有关更多详细信息，请参见将 DNS 用作默认接口。

如果未定义，则该值将从 HostInterfaceItem 获取。

如果该值超过 255 个字符的限制，agent 将报错且不会启动。

范围：0-255 个字符

HostInterfaceItem

指定一个监控项，用于在主机自动注册期间确定主机接口（IP 地址或 DNS 名称）。仅当未定义 HostInterface 时才会使用此值。有关更多详细信息，请参阅使用 DNS 作为默认接口。

在自动注册请求期间，如果指定监控项返回的值超过 255 个字符，agent 将记录一条警告消息。

无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

HostMetadata

指定用于在主机自动注册（主动 agent）期间识别或区分主机的元数据。
HostMetadata 可在主机名之外进一步区分主机。

如果未定义该参数，将从 HostMetadataItem 获取其值。

如果指定的值超过 2034 字节或不是 UTF-8 字符串，agent 将报错并且不会启动。
当参数要求使用 IP 地址或 DNS 名称时，虽然是有效的 UTF-8 字符串但不是有效 IP 地址或 DNS 名称的值也会被拒绝，并报告为无效值。

不支持多行元数据——输出将在第一个换行符处被截断。

范围：0-2034 字节

HostMetadataItem

指定一个 Zabbix agent 监控项，以获取主机元数据。仅当未定义 HostMetadata 时，才会使用此选项。

每次尝试自动注册时都会获取 HostMetadataItem 的值，并且该值仅用于主机自动注册过程（主动 agent）。HostMetadataItem 可用于根据主机名以外的信息区分主机。

支持用户参数、性能计数器和别名。无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

在自动注册请求期间，如果指定监控项返回的值超过 65535 个 UTF-8 代码点，agent 将记录警告消息。监控项返回的值必须是 UTF-8 字符串，否则将被忽略。当参数要求 IP 地址或 DNS 名称时，虽然值是有效的 UTF-8，但如果不是有效的 IP 地址或 DNS 名称，也会被拒绝并报告为无效。

不支持多行元数据——输出将在第一个换行符处被截断。

主机名

指定唯一且区分大小写的主机名。以逗号分隔的列表。
 主动检查必需，并且必须与服务器上配置的主机名匹配。如果未定义，则从 HostnameItem 获取该值。
 允许的字符：字母数字、'.'、' '、'_' 和'-'。每个主机名的最大长度为 128 个字符，整行最大长度为 2048 个字符。

默认值：由 HostnameItem 设置

HostnameItem

指定一个用于获取主机名的 Zabbix agent 监控项。
 仅当未定义 Hostname 时使用此选项。
 不支持用户参数、性能计数器或别名，但无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 的值如何，均支持system.run[] 监控项。
 另请参阅更详细的说明。

默认值：system.hostname

Include

指定要包含在配置文件中的目录或单个文件（如果使用 Windows MSI 安装程序包安装 Zabbix agent，配置文件默认位于 C:\Program Files\Zabbix Agent；如果将 Zabbix agent 安装为 zip 归档文件，配置文件位于安装期间指定的文件夹中）。

Include 参数的一般规则：

- 路径可以是相对于 zabbix_agentd.conf 文件位置的相对路径。
- 可以在一个配置文件中指定多个 Include 参数。
- Zabbix 在读取 Include 参数后会立即对其进行处理，因此请考虑将其放在配置文件末尾。

所包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，组件将无法启动。

包含目录或文件模式的规则：

- 不要假定包含顺序（例如，文件不会按字母顺序包含）。
- 不要在多个所包含文件中定义相同的参数（例如，使用特定设置覆盖常规设置），因为无法预测将使用哪个值。
- 注意某些文本编辑器会自动创建备份副本(例如，编辑 include/my_specific.conf 可能会创建 include/my_specific.conf.BAK)，因为这两个文件都会被包含。

要仅包含目录中的相关文件，请使用通配符（*）进行模式匹配。

示例：

```
Include=C:\Program Files\Zabbix Agent\zabbix_agentd.d\*.conf
```

ListenBacklog

设置 TCP 队列中待处理连接的最大数量。
 默认值是一个硬编码常量，具体取决于系统。
 支持的最大值也取决于系统，过高的值可能会被静默截断为 “implementation-specified maximum”。

默认值: SOMAXCONN
 范围: 0 - INT_MAX

ListenIP

指定 agent 用于接收来自服务器/proxy 连接的监听 IP 地址。使用逗号分隔的列表。

默认值：0.0.0.0

ListenPort

指定 agent 用于接收来自服务器/proxy 连接的监听端口。

默认值：10050
 范围：1024-32767

LogFile

指定 agent 日志文件。

默认值：c:\zabbix_agentd.log
 是否必需：是，如果 LogType 设置为 file；否则不是

LogFileSize

设置最大日志文件大小（单位：MB）。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始。

默认值：1
 范围：0-1024

LogRemoteCommands

将已执行的 shell 命令作为警告进行记录。
 仅当命令是远程执行时才会记录。
 如果 system.run[] 由 HostMetadataItem、HostInterfaceItem 或 HostnameItem 参数在本地启动，则不会创建日志条目。

默认值：0
 取值：0 - 禁用，1 - 启用

LogType

指定日志输出类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *system* - 将日志写入 Windows 事件日志；- *console* - 将日志写入标准输出。

默认值：file

MaxLinesPerSecond

设置在处理 log、logrt 或 eventlog 主动检查时，每秒发送到 Zabbix 服务器/proxy 的最大新行数。
 所提供的值将被 log、logrt 或 eventlog 监控项键中的 maxlines 参数覆盖。
 注意：Zabbix 处理的新行数将是 MaxLinesPerSecond 中设置值的 10 倍，以便在日志监控项中查找所需字符串。

默认值：20
 范围：1-1000

PerfCounter

定义一个新参数 <parameter_name>，它表示指定时间段 <period>（以秒为单位）内系统性能计数器 <perf_counter_path> 的平均值。
 语法：<parameter_name>,”<perf_counter_path>”,<period>

例如，如果您希望获取最近一分钟内每秒处理器中断的平均次数，可以将新参数”interrupts” 定义如下：

```
PerfCounter = interrupts,"\Processor(0)\Interrupts/sec",60
```

请注意性能计数器路径周围的双引号。参数名（interrupts）在创建监控项时将用作监控项键值。用于计算平均值的样本将每秒采集一次。
 您可以运行 typeperf -qx 来获取 Windows 中可用的所有性能计数器列表。

PerfCounterEn

定义一个新参数 <parameter_name>，它表示指定时间段 <period>（以秒为单位）内系统性能计数器 <perf_counter_path> 的平均值。与 PerfCounter 相比，perfcounter 路径必须使用英文。仅支持 **Windows Server 2008/Vista** 及更高版本。
 语法：<parameter_name>,”<perf_counter_path>”,<period>

例如，如果您希望获取最近一分钟内每秒处理器中断的平均次数，可以按如下方式定义一个新参数”interrupts”：

```
PerfCounterEn = interrupts,"\Processor(0)\Interrupts/sec",60
```

请注意性能计数器路径周围的双引号。参数名（interrupts）在创建监控项时将用作 item key。用于计算平均值的样本将每秒采集一次。
 您可以通过查看以下注册表项来获取英文字符串列表：HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009。

RefreshActiveChecks

指定刷新主动检查列表的频率，单位为秒。
 请注意，在刷新主动检查失败后，下一次刷新将在 60 秒后尝试。

默认值：5
 范围：1-86400

服务器

指定 Zabbix 服务器/proxy 的 IP 地址（可选使用 CIDR 表示法）或 DNS 名称。使用逗号分隔的列表。
 仅接受来自此处列出的主机的传入连接。如果启用了 IPv6 支持，则 127.0.0.1、::127.0.0.1、::ffff:127.0.0.1 会被视为相同，且 ::/0 将允许任何 IPv4 或 IPv6 地址。0.0.0.0/0 可用于允许任何 IPv4 地址。请注意，“IPv4 兼容的 IPv6 地址”（0000::/96 前缀）受支持，但已被 RFC4291 弃用。允许使用空格。

示例：

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

必填：是，如果未明确将 StartAgents 设置为 0

ServerActive

指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
 服务器/proxy 地址是 IP 地址或 DNS 名称，后跟可选端口，二者之间用冒号分隔。
 集群配置是一个或多个服务器或 proxy 组成员地址，地址之间用分号分隔。可以指定多个 Zabbix 服务器/集群和 Zabbix proxy，之间用逗号分隔。除非使用 proxy 组，否则每个 Zabbix 服务器/集群不应指定多个 Zabbix proxy。如果指定了 Zabbix proxy，则不应再为该 proxy 指定 Zabbix 服务器/集群。
 可以提供多个以逗号分隔的地址，以并行使用多个独立的 Zabbix 服务器。允许使用空格。
 如果未指定端口，则使用默认端口。
 如果为该主机指定了端口，则 IPv6 地址必须用方括号括起来。如果未指定端口，则 IPv6 地址的方括号为可选。
 如果未指定此参数，则会禁用主动检查。

Zabbix proxy 示例：

```
ServerActive=127.0.0.1:10051
```

Zabbix proxy 组示例：

```
ServerActive=proxy1.example.com;proxy2.example.com;proxy3.example.com;proxy4.example.com;proxy5.example.co
```

多个服务器示例：

```
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
```

高可用示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

包含两个集群和一台服务器的高可用示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

范围：ServerActive 中列出的主动服务器数量，加上 StartAgents 中指定的被动检查预 fork 实例数量，必须小于 64。

SourceIP

指定以下连接的源 IP 地址：

- 到 Zabbix 服务器或 Zabbix proxy 的出站连接；
- 在执行某些监控项（web.page.get、net.tcp.port 等）时建立的连接。

StartAgents

设置用于处理被动检查的 zabbix_agentd 预先 fork 的实例数量。
 如果设置为 0，则会禁用被动检查，agent 将不会监听任何 TCP 端口。

默认值：10
 范围：0-100 (*)

超时

指定与 Zabbix proxy 或服务器建立连接并交换数据时的等待时间（秒）。

此参数定义了各种通信操作的持续时间，包括：

- 等待来自 Zabbix 服务器的响应；

- 向 Zabbix 服务器发送请求，包括监控项配置请求以及active checks 中的监控项数据；
- 通过 logfile 或 Windows 事件日志监控检索日志数据；
- 发送心跳消息；
- `vfs.*` 检查的最大持续时间；
- 由 Zabbix agent 模块使用；
- 在服务器或 proxy 低于 7.0 版本且发送不带超时的检查时，作为回退值。

对于那些在前端中具有可配置超时设置的 agent 检查（全局、proxy 级别或按监控项级别），此超时将不会被使用。

默认值：3
 范围：1-30

TLSAccept

指定传入连接所需的加密级别。
 用于被动检查。
 可以指定多个值，并用逗号分隔：- *unencrypted* - 接受未加密连接（默认）；- *psk* - 接受使用 TLS 和预共享密钥（PSK）的连接；- *cert* - 接受使用 TLS 和证书的连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSCertFile

指定包含 agent 证书或证书链的文件的完整路径名，用于与 Zabbix 组件之间的加密通信。

TLSConnect

指定 agent 连接到 Zabbix 服务器或 proxy 所需的加密级别。
 用于主动检查。
 只能指定一个值：- *unencrypted* - 不使用加密进行连接（默认）；- *psk* - 使用 TLS 和预共享密钥（PSK）进行连接；- *cert* - 使用 TLS 和证书进行连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCRLFile

指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。

TLSKeyFile

指定包含 agent 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSPSKFile

指定包含 agent 预共享密钥的文件的完整路径名，用于与 Zabbix 服务器或 proxy 之间的加密通信。

TLSPSKIdentity

指定预共享密钥标识字符串，用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSServerCertIssuer

指定允许的服务器（proxy）证书颁发者。

TLSServerCertSubject

指定允许的服务器（proxy）证书主题。

UnsafeUserParameters

允许在参数中传递所有字符给用户自定义参数。以下字符不允许使用：\ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ %
 此外，不允许使用换行符。

默认值：0
 取值：0 - 不允许，1 - 允许

UserParameter

指定一个用户自定义参数用于监控。可以有多个用户自定义参数。
 格式：`UserParameter=<key>,<shell command>`
 请注意，shell 命令不能只返回空字符串或仅返回 EOL。如果指定了 `UserParameterDir` 参数，shell 命令可以使用相对路径。

示例：

```
UserParameter=system.test,dir /b | find /c /v ""
UserParameter=check_cpu,.\custom\_script.bat
```

UserParameterDir

指定 `UserParameter` 命令的默认搜索路径。如果使用此参数，agent 会在执行命令之前将其工作目录更改为此处指定的目录。因此，`UserParameter` 命令可以使用相对的 `./` 前缀，而不必使用完整路径。只允许一条配置项。

示例：

```
UserParameterDir=C:\opt\myscripts
```

Microsoft Windows 上的 Zabbix agent

配置 agent

两代 Zabbix agent 都作为 Windows 服务运行。
对于 Zabbix agent 2，请在以下说明中将 agentd 替换为 agent2。

您可以在 Microsoft Windows 主机上运行单个 Zabbix agent 实例，也可以运行多个 agent 实例。
单个实例可以使用以下任一配置：

- 默认配置文件，该文件位于 agent 二进制文件所在的同一目录中；
- 在命令行中指定的配置文件。

如果运行多个实例，则每个 agent 实例都必须拥有自己的配置文件（其中一个实例可以使用默认配置文件）。

Zabbix 源代码归档中提供了示例配置文件：

- Zabbix agent 使用 `conf/zabbix_agentd.conf`；
- Zabbix agent2 使用 `conf/zabbix_agent2.conf`。

如果您希望从归档中将适用于 Windows 的 Zabbix agent/agent 2 安装为服务，而不显式指定配置文件，则在安装 agent 之前：

- 应手动将 `conf/zabbix_agentd.conf` 复制到要安装 `zabbix_agentd.exe` 的目录；
- 应手动将 `conf/zabbix_agent2.conf` 和 `conf/zabbix_agent2.d` 目录复制到要安装 `zabbix_agent2.exe` 的目录。

有关配置 Zabbix Windows agent 的详细信息，请参阅 Zabbix agent 配置文件或 Zabbix agent 2 配置文件选项。

Hostname 参数

要在主机上执行主动检查，Zabbix agent 必须定义主机名。此外，agent 端设置的主机名值应与前端中为该主机配置的主机名完全匹配。

agent 端的主机名值可以通过agent 配置文件中的 `Hostname` 或 `HostnameItem` 参数定义；如果未指定这些参数中的任何一个，则使用默认值。

`HostnameItem` 参数的默认值是 `system.hostname` agent 键返回的值。对于 Windows，该键返回 `gethostname()` 函数的结果；该函数会查询命名空间提供程序，以确定本地主机名。如果没有命名空间提供程序响应，则返回 NetBIOS 名称。

`Hostname` 的默认值是 `HostnameItem` 参数返回的值。因此，实际上，如果这两个参数均未指定，实际主机名将是主机的 NetBIOS 名称；Zabbix agent 将使用 NetBIOS 主机名从 Zabbix 服务器获取主动检查列表，并将结果发送到服务器。

`system.hostname` 键支持两个可选参数 - *type* 和 *transform*。

Type 决定监控项应返回的名称类型：

- *netbios*（默认）- 返回 NetBIOS 主机名，长度限制为 15 个字符，且仅包含大写字母；
- *host* - 区分大小写，返回完整的真实 Windows 主机名（不含域名）；
- *shorthost* - 返回主机名中第一个点号之前的部分。如果名称不包含点号，则返回完整字符串。
- *fqdn* - 返回完全限定域名（不含末尾的点号）。

Transform 允许为主机名指定其他转换规则：

- *none*（默认）- 使用原始字母大小写；
- *lower* - 将文本转换为小写。

因此，为简化 `zabbix_agentd.conf` 文件的配置并使其保持统一，可以采用以下不同方法：

1. 不定义 `Hostname` 或 `HostnameItem` 参数，Zabbix agent 将使用 NetBIOS 主机名作为主机名。
2. 不定义 `Hostname` 参数，并按如下方式定义 `HostnameItem`：
    - `HostnameItem=system.hostname[host]` - 让 Zabbix agent 使用完整的真实 Windows 主机名（区分大小写）作为主机名。
    - `HostnameItem=system.hostname[shorthost,lower]` - 让 Zabbix agent 仅使用主机名中第一个点号之前的部分，并将其转换为小写。
    - `HostnameItem=system.hostname[fqdn]` - 让 Zabbix agent 使用完全限定域名作为主机名。

主机名还用作 Windows 服务名称的一部分，该服务名称用于安装、启动、停止和卸载 Windows 服务。例如，如果 Zabbix agent 配置文件指定了 `Hostname=Windows_db_server`，则该 agent 将作为 Windows 服务 `Zabbix Agent [Windows_db_server]` 安装。因此，要为每个 Zabbix agent 实例使用不同的 Windows 服务名称，每个实例必须使用不同的主机名。

将 agent 安装为 Windows 服务

在安装 agent 之前，请先将 `conf/zabbix_agentd.conf` 手动复制到 zabbix_agentd.exe 将要安装的目录中。

要使用默认配置文件安装单个 Zabbix agent 实例：

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

> **Attention:**
> 在 64 位系统上，所有与运行 64 位进程相关的检查要正常工作，都需要使用 64 位版本的 Zabbix agent。

如果您希望使用默认配置文件以外的其他配置文件，则应使用以下命令安装服务：

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

应指定配置文件的完整路径。

可以像这样将多个 Zabbix agent 实例安装为服务：

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

现在，已安装的服务应可在控制面板中看到。

启动 agent

要启动 agent 服务，您可以使用控制面板，也可以通过命令行执行。

要使用默认配置文件启动一个 Zabbix agent 实例：

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

要使用其他配置文件启动一个 Zabbix agent 实例：

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

要启动多个 Zabbix agent 实例中的某一个：

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

停止 agent

要停止 agent 服务，您可以使用控制面板，也可以通过命令行执行。

要停止使用默认配置文件启动的单个 Zabbix agent 实例：

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

要停止使用其他配置文件启动的单个 Zabbix agent 实例：

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

要停止多个 Zabbix agent 实例中的某一个：

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

卸载 agent Windows 服务

要使用默认配置文件卸载单个 Zabbix agent 实例：

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

要使用非默认配置文件卸载单个 Zabbix agent 实例：

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

要从 Windows 服务中卸载多个 Zabbix agent 实例：

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

限制

Windows 版 Zabbix agent 不支持 CPU 在 NUMA 节点之间非均匀分布的非标准 Windows 配置。如果逻辑 CPU 的分布不均匀，则某些 CPU 可能无法获取 CPU 性能指标。例如，如果有 72 个逻辑 CPU 和 2 个 NUMA 节点，则两个节点都必须各有 36 个 CPU。

IBM AIX 上的 Zabbix agent

概述

可从下载页面获取适用于 IBM AIX 的预编译 Zabbix agent 二进制文件，文件格式为 .tar.gz 压缩包。本页面介绍如何从压缩包安装 agent、手动运行 agent 或通过系统资源控制器（SRC）运行 agent、在系统启动时启动 agent，以及从 SRC 中移除 agent。

安装 agent

AIX 的 tar 命令不支持 -z 选项，因此请使用 gunzip 解压归档文件。请使用 tar 的 -o 选项，以便提取的文件归属于执行提取操作的用户，而不是归属于归档中存储的数字用户 ID 和组 ID。例如，要将 agent 放置在 /opt/zabbix 中（以 root 身份运行；/opt/zabbix 必须尚不存在）：

```
gunzip -c zabbix_agent-7.4.7-aix-7.2-powerpc-openssl.tar.gz | tar -xof -
mv zabbix_agent-7.4.7-aix-7.2-powerpc-openssl /opt/zabbix
```

在 AIX 归档文件中，agent 二进制文件位于 usr/sbin/，配置文件位于 etc/zabbix/。启动 agent 前，请根据需要调整 /opt/zabbix/etc/zabbix/zabbix_agentd.conf（至少调整Server、ServerActive 和Hostname 参数）。

如果以 root 身份启动，agent 将切换到 zabbix 用户（或切换到User 参数指定的用户），除非在配置文件中设置了AllowRoot=1。该用户必须存在于系统中，否则 agent 将无法启动。如果 zabbix 用户尚不存在，请同时创建同名组：

```
mkgroup zabbix
mkuser pgrp=zabbix login=false rlogin=false zabbix
```

login=false 和 rlogin=false 属性可防止使用该账户登录；但 agent 仍可作为该用户运行。

手动启动 agent

可以手动启动 agent：

```
/opt/zabbix/usr/sbin/zabbix_agentd -c /opt/zabbix/etc/zabbix/zabbix_agentd.conf
```

使用 SRC 运行 agent

注册 agent

agent 也可以向系统资源控制器（SRC）注册。通过 SRC 启动时，agent 必须以前台模式运行（使用 -f 选项）。否则，SRC 会失去对 agent 进程的跟踪，因而无法停止该进程。例如（以 root 身份运行）：

```
mkssys \
  -s zabbix_agentd \
  -p /opt/zabbix/usr/sbin/zabbix_agentd \
  -a "-c /opt/zabbix/etc/zabbix/zabbix_agentd.conf -f" \
  -u zabbix \
  -G tcpip \
  -S -n 15 -f 9 \
  -R
```

在此示例中：

- -s 设置与 SRC 命令配合使用的子系统名称；
- -p 设置 agent 二进制文件的完整路径；
- -a 将 agent 命令行参数设置为一个带引号的字符串；SRC 启动 agent 时会将此字符串拆分为单独的参数（只接受一个 -a 选项）；
- -u 设置 agent 运行时所使用的用户；创建子系统时，用户名会转换为数字用户 ID，因此该用户必须已存在；
- -G 将 agent 添加到 tcpip 子系统组中，因此组命令也会应用于该 agent（例如，stopsrc -g tcpip 也会停止该 agent）；
- -S -n 15 -f 9 使 SRC 通过信号控制 agent：stopsrc 在正常停止时发送 SIGTERM（15），在强制停止时（stopsrc -f）发送 SIGKILL（9）；
- -R 使 SRC 在 agent 异常终止时重新启动它（例如，其某个进程崩溃时）；执行 stopsrc 后，agent 不会重新启动。

由于用户 ID 存储在子系统定义中，如果重新创建 zabbix 用户时使用了不同的用户 ID，请更新该定义：

```
chssys -s zabbix_agentd -u zabbix
```

启动和停止 agent

可以使用 startsrc、lssrc 和 stopsrc 启动、查询和停止 agent：

```
startsrc -s zabbix_agentd
lssrc -s zabbix_agentd
stopsrc -s zabbix_agentd
```

请避免使用强制停止命令（stopsrc -f）。该命令只会终止主 agent 进程；其他 agent 进程仍会继续运行并保持监听端口处于开放状态，因此在手动终止这些进程之前（例如使用 kill -9），无法再次启动 agent。

在系统启动时启动 agent

将 agent 注册到 SRC 并不会使其在系统启动时启动。要在启动时自动启动 agent，请在 /etc/inittab 中添加以下条目：

```
mkitab "zabbix_agentd:2:once:/usr/bin/startsrc -s zabbix_agentd >/dev/console 2>&1"
```

从 SRC 中移除 agent

要从系统启动项和 SRC 中移除 agent：

```
stopsrc -s zabbix_agentd
rmitab zabbix_agentd
rmssys -s zabbix_agentd
```

Agent 2

概述　Zabbix agent 2 是Zabbix agent 的新一代版本，使用 Go 编写（其中部分 C 代码复用了 Zabbix agent 的代码）。其设计目标是：

- 减少 TCP 连接数量；
- 提供更好的检查并发性；
- 通过插件轻松扩展，插件可提供仅需少量代码的简单检查，也可支持由长时间运行的脚本组成的复杂检查，以及带有周期性报告的独立数据采集；
- 作为 Zabbix agent 的替代方案，支持所有先前功能。

Zabbix agent 2 中的所有指标都由插件采集。

这些功能是插件特有的：

- 被动检查和主动检查都支持计划和灵活的间隔
- 考虑计划和任务并发性的任务队列管理
- 插件级超时
- Zabbix agent 2 及其插件在启动时的兼容性检查

插件有两种类型：

- 内置插件
- 可加载插件

你也可以创建自己的插件。

自定义创建的插件不必内置到 agent 2 中。你可以将它们作为可加载插件添加，从而简化新指标插件的创建。

可加载插件可以在运行时添加，无需重新编译 Zabbix agent 2。

另请参见：

- Agent 与 agent 2 对比
- 配置参数：Zabbix agent 2
- 配置参数：Zabbix agent 2（Windows）
- 配置参数：插件
- Zabbix agent 2 插件协议

被动检查和主动检查

Zabbix agent 2 支持被动检查和主动检查，与 Zabbix agent 类似。
此外，Zabbix agent 2 的主动检查还支持单个主动服务器内的灵活/调度间隔和检查并发。

> **Note:**
> 默认情况下，重启后，Zabbix agent 2 会在监控项更新间隔内的一个条件随机时间安排主动检查的首次数据采集，以防止资源使用量出现峰值。
> 若要在 agent 重启后立即执行没有 *Scheduling* 更新间隔的主动检查，请在配置文件中设置ForceActiveChecksOnStart 参数（全局级别），或设置 Plugins.<Plugin name>.System.ForceActiveChecksOnStart 插件参数（仅影响特定插件的检查）。
> 如果设置了插件级别的参数，则该参数会覆盖全局参数。

检查并发

来自不同插件的检查可以并发执行。单个插件内并发检查的数量受插件容量设置限制。每个插件可能具有硬编码的容量设置（默认值为1000），可通过 Plugins.<PluginName>.System.Capacity=N 设置在 Plugins 配置参数中降低。

支持的平台　有关支持的平台，请参阅Requirements 页面。

Unix-like 系统上的 Agent 2　Unix-like 系统上的 Zabbix agent 2 运行在被监控的主机上。

安装

Zabbix agent 2 可以使用以下方法之一安装在基于 Linux 的系统上：

- Zabbix 软件包 - 选择 Agent 2 组件（在选择 Zabbix 版本、OS 发行版和 OS 版本之后），然后按照说明操作；
- Zabbix 源码 - 下载源文件，并通过使用 --enable-agent2 选项进行配置来编译 agent。

Note:
Zabbix agent 2 的监控能力可以通过可加载插件进行扩展，这些插件可单独获取。有关详细信息，请参见可加载插件。

如果以软件包形式安装

Zabbix agent 2 作为前台进程运行，并依赖外部服务管理器（例如 systemd）在后台执行。Zabbix agent 2 在 Linux 上不支持内置的守护进程化。

可通过执行以下命令启动 agent：

```
systemctl start zabbix-agent2
```

要停止、重启或检查 Zabbix agent 2 的状态，请使用以下命令：

```
systemctl stop zabbix-agent2
systemctl restart zabbix-agent2
systemctl status zabbix-agent2
```

手动启动

您可以通过找到 zabbix_agent2 二进制文件并直接运行它来启动 Zabbix agent；例如：

```
zabbix_agent2
```

Windows 系统上的 Agent 2　Zabbix agent 2 作为独立进程运行；不过，它也可以作为 Windows 服务运行。

安装

Zabbix agent 2 可以使用以下方法之一安装在 Windows 上：

- 预编译的 Zabbix agent 二进制文件 - 下载 agent MSI 安装包，并按照从 MSI 安装 Windows agent 页面中的说明进行操作；
- Zabbix 源代码 - 下载源文件，并按照在 Windows 上构建 Zabbix agent 2 页面中的说明进行操作。

Note:
Zabbix agent 2 的监控能力可以通过可加载插件进行扩展，这些插件需单独获取。有关详细信息，请参见可加载插件。

有关将 Zabbix agent 2（从 ZIP 压缩包）作为 Windows 服务安装的更多详细信息，请参见Microsoft Windows 上的 Zabbix agent 页面。

选项　以下命令行参数可与 Zabbix agent 2 一起使用：

参数	描述
UNIX 和 Windows agent	
-c --config <config-file>	配置文件的路径。 您可以使用此选项指定非默认配置文件。
-f --foreground	在前台运行 Zabbix agent（默认值：true）。
-p --print	打印已知监控项并退出。 请注意，如需同时返回用户参数结果，必须指定配置文件（如果该文件不在默认位置）。
-t --test <item key>	测试指定的监控项并退出。 请注意，如需同时返回用户参数结果，必须指定配置文件（如果该文件不在默认位置）。
-T --test-config	验证配置文件并退出。
-h --help	打印帮助信息并退出。
-v --verbose	打印调试信息。将此选项与 -p 和 -t 选项一起使用。

参数	描述
-V --version	打印 agent 版本和许可证信息。
-R --runtime-control <option>	执行管理功能。请参阅运行时控制。
仅限 **Windows agent**	
-m --multiple-agents	使用多个 agent 实例（与 -i、-d、-s、-x 选项一起使用）。 为区分各实例的服务名称，每个服务名称都将包含指定配置文件中的 Hostname 值。
-S --startup-type <value>	设置 Zabbix Windows agent 服务的启动类型。允许的值： automatic - （默认）在 Windows 启动时自动启动服务； delayed - 延迟启动服务，直到自动启动的服务完成启动； manual - 手动启动服务（由用户或应用程序启动）； disabled - 禁用服务，使其无法由用户或应用程序启动。 您可以将此选项与 -i 选项一起使用，也可以单独使用此选项来修改已安装服务的启动类型。
-i --install	将 Zabbix Windows agent 安装为服务。
-d --uninstall	卸载 Zabbix Windows agent 服务。
-s --start	启动 Zabbix Windows agent 服务。
-x --stop	停止 Zabbix Windows agent 服务。

以下是使用命令行参数的具体示例：

- 打印所有内置 agent 监控项及其值。
- 使用指定配置文件中定义的 mysql.ping 监控项键测试用户参数。
- 使用配置文件的默认路径为 Windows 安装 Zabbix Agent 服务。
- 使用与 agent 可执行文件位于同一文件夹中的配置文件，修改已安装的 Windows Zabbix Agent 服务的启动类型。

```
zabbix_agent2 --print
zabbix_agent2 -t "mysql.ping" -c /etc/zabbix/zabbix_agent2.conf
zabbix_agent2.exe -i
zabbix_agent2.exe -c zabbix_agent2.conf -S delayed
```

运行时控制

运行时控制提供了一些远程控制选项。

选项	描述
log_level_increase	提高日志级别。
log_level_decrease	降低日志级别。
metrics	列出可用的指标。
periodic_prof_enable	启用定期性能分析。 除 CPU 数据外，所有指标都会即时收集数据并将其转储。定期 CPU 数据会在一段时间内收集到隐藏的临时文件中。间隔结束后，该文件会作为 CPU 性能分析文件发布，并保留配置数量的已完成文件。 临时文件不计入 ProfilerMaxFilesPerProfile。
periodic_prof_disable	禁用定期性能分析。
periodic_prof_execute	在采样 5 秒后写入性能分析文件。 启用定期性能分析后，此命令会结束当前 CPU 间隔，写入单独的 5 秒 CPU 性能分析文件，并开始新的定期 CPU 文件。 命令完成时，显式请求的性能分析文件会根据 ProfilerMaxFilesPerProfile 进行轮换。 此命令收集性能分析数据期间，所有性能分析器运行时命令都会被拒绝，并显示 “profiler is busy”。
periodic_prof_set_interval	设置性能分析间隔。
version	显示 agent 版本。
userparameter_reload	从当前配置文件重新加载UserParameter 和Include 选项的值。
help	显示运行时控制的帮助信息。

示例：

- 提高 agent 2 的日志级别。
- 打印运行时控制选项。

```
zabbix_agent2 -R log_level_increase
zabbix_agent2 -R help
```

配置文件　配置文件的默认路径取决于操作系统和 Zabbix 的安装方式：

- 在 UNIX 上，默认路径为：
  - 从源代码安装：/usr/local/etc/zabbix_agent2.conf，或由编译时变量 --sysconfdir 或 --prefix 设置的路径。
  - 从软件包安装：/etc/zabbix/zabbix_agent2.conf。
- 在 Windows 上，默认路径为 C:\Program Files\Zabbix Agent 2\zabbix_agent2.conf。

如果主机上运行的是单个 agent 实例，则可以使用默认配置文件，或使用在命令行中指定的配置文件。如果运行多个实例，则每个 agent 实例都必须拥有自己的配置文件（其中一个实例可以使用默认配置文件）。

Zabbix agent 2 的配置参数大多与 Zabbix agent 的配置参数相同，但存在以下例外：

参数	描述
Zabbix agent 2 特有的配置参数	
ControlSocket	运行时控制套接字路径。Zabbix agent 2 使用控制套接字执行运行时命令。
EnablePersistentBuffer、PersistentBufferFile、PersistentBufferPeriod	这些参数用于为 Zabbix agent 2 上的 active 监控项配置持久化存储。
ForceActiveChecksOnStart	确定 agent 是应在重启后立即执行 active 检查，还是随时间均匀分散执行。
Plugins	插件可能拥有自己的参数，格式为 Plugins.<Plugin name>.<Parameter>=<value>。常见的插件参数是 System.Capacity，用于设置可同时执行的检查数量上限。
StatusPort	Zabbix agent 2 监听 HTTP 状态请求以及显示已配置插件和部分内部参数的端口。
Zabbix agent 特有但 Zabbix agent 2 不支持的配置参数	
AllowRoot、User	Zabbix agent 2 不支持这些参数，因为它不支持守护进程化。
EnableRemoteCommands	已弃用；在两个 agent 中，应改用 AllowKey、AllowKeyRegexp、DenyKey 和 DenyKeyRegexp 参数。
ListenBacklog	在 Zabbix agent 中用于定义 TCP 队列中待处理连接的最大数量。Zabbix agent 2 不支持此参数。
LoadModule、LoadModulePath	Zabbix agent 2 不支持可加载模块。
MaxLinesPerSecond、LogRemoteCommands	在 Zabbix agent 2 中，应改用 Plugins.Log.MaxLinesPerSecond 和 Plugins.SystemRun.LogRemoteCommands。
StartAgents	此参数在 Zabbix agent 中用于增加 passive 检查的并发数或禁用 passive 检查。在 Zabbix agent 2 中，并发数在插件级别进行配置，并可通过容量设置进行限制。如果未指定 Server 参数，Zabbix Agent 2 将禁用 passive 检查。

有关更多详细信息，请参阅Zabbix agent2 的配置文件选项。

退出代码　Zabbix agent 2 也可以使用较旧的 OpenSSL 版本（1.0.1、1.0.2）进行编译。

在这种情况下，Zabbix 会提供用于在 OpenSSL 中加锁的互斥锁。如果互斥锁加锁或解锁失败，则会将错误消息打印到标准错误流（STDERR），并且 Agent 2 分别以返回码 2 或 3 退出。

Agent 与 agent 2 对比

本节描述 Zabbix agent 与 Zabbix agent 2 之间的差异。

Parameter	Zabbix agent	Zabbix agent 2
Programming language	C	Go with some parts in C
Daemonization	yes	by systemd only (yes on Windows)
Supported extensions	Custom loadable modules in C.	Custom plugins in Go.
Requirements		
Supported platforms	Linux, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, Mac OS X, Solaris: 9, 10, 11, Windows: all desktop and server versions since XP	Linux, Windows: all desktop and server versions, on which a supported Go version can be installed.
Supported crypto libraries	GnuTLS 3.1.18 and newer. OpenSSL 1.0.1, 1.0.2, 1.1.0, 1.1.1, 3.0.x. LibreSSL - tested with versions 2.7.4, 2.8.2 (certain limitations apply, see the Encryption page for details).	Linux: OpenSSL 1.0.1 and later. MS Windows: OpenSSL 1.1.1 or later. The OpenSSL library must have PSK support enabled. LibreSSL is not supported.
Monitoring processes		
Processes	A separate active check process for each server/proxy record.	Single process with automatically created threads. The maximum number of threads is determined by the `GOMAXPROCS` environment variable.
Metrics	**UNIX**: see a list of supported items. **Windows**: see a list of additional Windows-specific items.	**UNIX**: All metrics supported by Zabbix agent. Additionally, the agent 2 provides Zabbix-native monitoring solution for: Docker, Memcached, MySQL, PostgreSQL, Redis, systemd, and other monitoring targets - see a full list of agent 2 specific items. **Windows**: All metrics supported by Zabbix agent, and also `net.tcp.service*` checks of HTTPS, LDAP. Additionally, the agent 2 provides Zabbix-native monitoring solution for: PostgreSQL, Redis.
Concurrency	Active checks for single server are executed sequentially.	Checks from different plugins or multiple checks within one plugin can be executed concurrently.
Third-party traps	no	yes
Additional features		
Persistent storage	no	yes
Persistent files for `log*[]` metrics	yes (only on Unix)	no
Log data upload	Can be performed during log gathering to free the buffer.	Log gathering is stopped when the buffer is full, therefore the `BufferSize` parameter must be at least `MaxLinesPerSecond` x 2.
Changes user at runtime	yes (Unix-like systems only)	no (controlled by systemd)
User-configurable ciphersuites	yes	no

另请参见：

- *Zabbix* 进程说明：Zabbix agent, Zabbix agent 2
- 配置参数：Zabbix agent UNIX / Windows, Zabbix agent 2 UNIX / Windows

内置插件

这些插件内置于 Zabbix agent 2 中，可直接开箱使用。单击插件名称可跳转到插件仓库，查看更多信息。

插件名称	描述	支持的监控项键	备注
Agent	正在使用的 Zabbix agent 的指标。	agent.hostname, agent.ping, agent.version	支持的键与 Zabbix agent 键具有相同参数。
CPU	系统 CPU 监控（CPU 数量/CPU 核心数、已发现的 CPU、利用率百分比）。	system.cpu.discovery, system.cpu.num, system.cpu.util	支持的键与 Zabbix agent 键具有相同参数。
Docker	Docker 容器监控。	docker.container_info, docker.container_stats, docker.containers, docker.containers.discovery, docker.data_usage, docker.images, docker.images.discovery, docker.info, docker.ping	另请参见： 配置参数
File	文件指标采集。	vfs.file.cksum, vfs.file.contents, vfs.file.exists, vfs.file.md5sum, vfs.file.regexp, vfs.file.regmatch, vfs.file.size, vfs.file.time	支持的键与 Zabbix agent 键具有相同参数。
Kernel	内核监控。	kernel.maxfiles, kernel.maxproc	支持的键与 Zabbix agent 键具有相同参数。
Log	日志文件监控。	log, log.count, logrt, logrt.count	支持的键与 Zabbix agent 键具有相同参数。 另请参见： 插件配置参数（Unix/Windows）
Memcached	Memcached 服务器监控。	memcached.ping, memcached.stats	
Modbus	读取 Modbus 数据。	modbus.get	支持的键与 Zabbix agent 键具有相同参数。
MQTT	接收已发布的 MQTT 主题值。	mqtt.get	要配置到 MQTT broker 的加密连接，请在 agent 配置文件中将 TLS 参数指定为命名会话或默认参数。目前 TLS 参数不能作为监控项键参数传递。
MySQL	MySQL 及其分支版本监控。	mysql.custom.query, mysql.db.discovery, mysql.db.size, mysql.get_status_variables, mysql.ping, mysql.replication.discovery, mysql.replication.get_slave_status, mysql.version	要配置到数据库的加密连接，请在 agent 配置文件中将 TLS 参数指定为命名会话或默认参数。目前 TLS 参数不能作为监控项键参数传递。
NetIf	网络接口监控。	net.if.collisions, net.if.discovery, net.if.get, net.if.in, net.if.out, net.if.total	支持的键与 Zabbix agent 键具有相同参数。

插件名称	描述	支持的监控项键	备注
Oracle	Oracle Database 监控。	oracle.diskgroups.stats, oracle.diskgroups.discovery, oracle.archive.info, oracle.archive.discovery, oracle.cdb.info, oracle.custom.query, oracle.datafiles.stats, oracle.db.discovery, oracle.fra.stats, oracle.instance.info, oracle.pdb.info, oracle.pdb.discovery, oracle.pga.stats, oracle.ping, oracle.proc.stats, oracle.redolog.info, oracle.sga.stats, oracle.sessions.stats, oracle.sys.metrics, oracle.sys.params, oracle.ts.stats, oracle.ts.discovery, oracle.user.info, oracle.version	使用该插件前请安装 Oracle Instant Client。
Proc	进程 CPU 利用率百分比。	proc.cpu.util	支持的键与 Zabbix agent 键具有相同参数。
Redis	Redis 服务器监控。	redis.config, redis.info, redis.ping, redis.slowlog.count	要配置到 Redis 的加密连接，请在 agent 配置文件中将 TLS 参数指定为命名会话或默认参数。TLS 参数不能作为监控项键参数传递。请注意，不正确或无效的 TLS 配置可能会导致 Zabbix agent 2 无法启动，因此在启用 TLS 之前，请先验证证书文件、权限和路径。

Smart	S.M.A.R.T. 监控。	smart.attribute.discovery, smart.disk.discovery, smart.disk.get	所需的 smartctl 最低版本为 7.1。
			运行 Zabbix agent 2 的用户需要具备 smartctl 的 sudo/root 访问权限。该插件仅使用以下命令：
			`/usr/sbin/smartctl -a *`
			`/usr/sbin/smartctl --scan *`
			`/usr/sbin/smartctl -j -V`
			
			支持的键 仅可在 Linux/Windows 上与 Zabbix agent 2 一起使用，既可作为被动检查，也可作为主动检查。
			另请参见：
			配置参数

SW	已安装软件包列表。	system.sw.packages, system.sw.packages.get	支持的键与 Zabbix agent 键具有相同参数。
Swap	以字节/百分比表示的交换空间大小。	system.swap.size	支持的键与 Zabbix agent 键具有相同参数。
SystemRun	运行指定命令。	system.run	支持的键与 Zabbix agent 键具有相同参数。 另请参见： 插件配置参数（Unix/Windows）
Systemd	systemd 服务监控。	systemd.unit.discovery, systemd.unit.get, systemd.unit.info	
TCP	TCP 连接可用性检查。	net.tcp.port	支持的键与 Zabbix agent 键具有相同参数。
UDP	UDP 服务可用性和性能监控。	net.udp.service, net.udp.service.perf	支持的键与 Zabbix agent 键具有相同参数。

插件名称	描述	支持的监控项键	备注
Uname	获取系统信息。	system.hostname, system.sw.arch, system.uname	支持的键与 Zabbix agent 键具有相同参数。
Uptime	系统运行时间指标采集。	system.uptime	支持的键与 Zabbix agent 键具有相同参数。
VFSDev	VFS 指标采集。	vfs.dev.discovery, vfs.dev.get, vfs.dev.read, vfs.dev.write	支持的键与 Zabbix agent 键具有相同参数。
WebCertificate	TLS/SSL 网站证书监控。	web.certificate.get	
WebPage	网页监控。	web.page.get, web.page.perf, web.page.regexp	支持的键与 Zabbix agent 键具有相同参数。
ZabbixAsync	异步指标采集。	net.tcp.listen, net.udp.listen, sensor, system.boottime, system.cpu.intr, system.cpu.load, system.cpu.switches, system.hw.cpu, system.hw.macaddr, system.localtime, system.sw.os, system.swap.in, system.swap.out, vfs.fs.discovery	支持的键与 Zabbix agent 键具有相同参数。
ZabbixStats	Zabbix 服务器/proxy 内部指标或队列中延迟监控项的数量。	zabbix.stats	支持的键与 Zabbix agent 键具有相同参数。
ZabbixSync	同步指标采集。	net.dns, net.dns.record, net.tcp.service, net.tcp.service.perf, proc.mem, proc.num, system.hw.chassis, system.hw.devices, system.sw.packages, system.users.num, vfs.dir.count, vfs.dir.size, vfs.fs.get, vfs.fs.inode, vfs.fs.size, vm.memory.size.	支持的键与 Zabbix agent 键具有相同参数。

可加载插件

Zabbix agent 2 默认不提供可加载插件，需要单独安装：

- 在 Linux 上，可以使用软件包（例如，在 Ubuntu 上使用 `apt install zabbix-agent2-plugin-ember-plus` 安装 Ember+），或构建插件
- 在 Windows 上，可以从 MSI 安装，或构建插件

Attention:
安装插件前，请检查其 README 文件。其中可能包含特定要求和安装说明。

点击插件名称可转到插件仓库，其中包含插件 README 文件及其他信息。

插件名称	描述	支持的监控项键	备注
Ceph	Ceph 监控。	ceph.df.details, ceph.osd.stats, ceph.osd.discovery, ceph.osd.dump, ceph.ping, ceph.pool.discovery, ceph.status	另请参阅Ceph 插件配置参数。
Ember+	Ember+ 监控。	ember.get	另请参阅Ember+ 插件配置参数。
MongoDB	MongoDB 服务器和集群监控（基于文档的分布式数据库）。	mongodb.cfg.discovery, mon- godb.collection.stats, mon- godb.collections.discovery, mon- godb.collections.usage, mon- godb.connpool.stats, mongodb.db.stats, mon- godb.db.discovery, mon- godb.jumbo_chunks.count, mongodb.oplog.stats, mongodb.ping, mongodb.rs.config, mongodb.rs.status, mon- godb.server.status, mongodb.sh.discovery, mongodb.version	要配置与数据库的加密连接，请在 agent 配置文件中将 TLS 参数指定为命名会话参数。 目前无法将 TLS 参数作为监控项键参数传递。 另请参阅MongoDB 插件配置参数。
MSSQL	MSSQL 数据库监控。	mssql.availability.group.get, mssql.custom.query, mssql.db.get, mssql.job.status.get, mssql.last.backup.get, mssql.local.db.get, mssql.mirroring.get, mssql.nonlocal.db.get, mssql.perfcounter.get, mssql.ping, mssql.quorum.get, mssql.quorum.member.get, mssql.replica.get, mssql.version	要配置与数据库的加密连接，请在 agent 配置文件中将 TLS 参数指定为命名会话参数或默认参数。目前无法将 TLS 参数作为监控项键参数传递。 另请参阅MSSQL 插件配置参数。

插件名称	描述	支持的监控项键	备注
NVIDIA GPU	NVIDIA GPU 监控。	nvml.device.count, nvml.device.decoder.utilization, nvml.device.ecc.mode, nvml.device.encoder.stats.get, nvml.device.encoder.utilization, nvml.device.energy.consumption, nvml.device.errors.memory, nvml.device.errors.register, nvml.device.fan.speed.avg, nvml.device.get, nvml.device.graphics.frequency, nvml.device.memory.bar1.get, nvml.device.memory.fb.get, nvml.device.memory.frequency, nvml.device.pci.utilization, nvml.device.performance.state, nvml.device.power.limit, nvml.device.power.usage, nvml.device.serial, nvml.device.sm.frequency, nvml.device.temperature, nvml.device.utilization, nvml.device.video.frequency, nvml.system.driver.version, nvml.version	另请参阅NVIDIA GPU 插件配置参数。
PostgreSQL	PostgreSQL 及其分支监控。	pgsql.autovacuum.count, pgsql.archive, pgsql.bgwriter, pgsql.cache.hit, pgsql.connections, pgsql.custom.query, pgsql.dbstat, pgsql.dbstat.sum, pgsql.db.age, pgsql.db.bloating_tables, pgsql.db.discovery, pgsql.db.size, pgsql.locks, pgsql.oldest.xid, pgsql.ping, pgsql.queries, pgsql.replication.count, pgsql.replication.process, pgsql.replication.process.discovery, pgsql.replication.recovery_role, pgsql.replication.status, pgsql.replication_lag.b, pgsql.replication_lag.sec, pgsql.uptime, pgsql.version, pgsql.wal.stat	要配置与数据库的加密连接，请在 agent 配置文件中将 TLS 参数指定为命名会话参数或默认参数。 目前无法将 TLS 参数作为监控项键参数传递。 另请参阅PostgreSQL 插件配置参数。

Note:
可加载插件启动时使用以下选项：
 - `-V --version` - 输出插件版本和许可证信息；
 - `-h --help` - 输出帮助信息。
 - `-t, --test <item key>` - 启动插件进行测试（忽略插件配置）。

配置参数：**Zabbix agent 2**

概述

配置参数可用于自定义 Zabbix agent 2 的行为。

Zabbix agent 2 是新一代 Zabbix agent，可替代 Zabbix agent 使用。

在 Zabbix agent 2 配置文件（zabbix_agent2.conf）中指定参数值。除非明确说明参数为必需参数，否则所有参数均为可选参数。

本页面结构如下：

- 参数概览（要查看参数的完整详细信息，请点击其名称）
- 参数详细信息

参数	描述
Alias	为监控项键设置别名。
AllowKey	允许执行与某个模式匹配的监控项键。
AllowKeyRegexp	允许执行与基于正则表达式的模式匹配的监控项键。
BufferSend	设置将值从缓冲区发送到 Zabbix 服务器/proxy 的频率。
BufferSize	设置内存缓冲区中的最大值数量。
ControlSocket	指定用于通过 “-R” 选项发送运行时命令的控制套接字。
DebugLevel	设置调试级别。
DenyKey	拒绝执行与某个模式匹配的监控项键。
DenyKeyRegexp	拒绝执行与基于正则表达式的模式匹配的监控项键。
EnablePersistentBuffer	启用对 active 监控项使用本地持久化存储。
EnableProfiler	启用 agent 启动时定期进行 Go 性能分析。
ForceActiveChecksOnStart	针对首次接收的配置，在重启后立即执行 active 检查。
HeartbeatFrequency	设置心跳消息的发送频率（秒）。
HostInterface	指定主机接口。
HostInterfaceItem	指定用于获取主机接口的监控项。
HostMetadata	指定主机元数据。
HostMetadataItem	指定用于获取主机元数据的 Zabbix agent 监控项。
Hostname	指定主机名。
HostnameItem	指定用于获取主机名的 Zabbix agent 监控项。
Include	指定要包含在配置文件中的目录或单个文件。
ListenIP	指定用于接收来自服务器/proxy 连接的 agent 监听 IP 地址。
ListenPort	指定 agent 监听端口。
LogFile	指定 agent 日志文件。
LogFileSize	设置日志文件的最大大小。
LogType	指定日志输出类型。
PersistentBufferFile	指定用于保存 SQLite 数据库的文件。
PersistentBufferPeriod	设置在无法连接到 Zabbix 服务器或 proxy 时存储数据的时间段。
PidFile	指定 PID 文件。
Plugins.<PluginName>.System.Capacity	设置每个插件可同时执行的检查数量上限。
Plugins.Log.MaxLinesPerSecond	处理 log 或 logrt active 检查时，设置每秒发送到 Zabbix 服务器/proxy 的新行数上限。
Plugins.SystemRun.LogRemoteCommands	启用将已执行的 shell 命令作为警告记录到日志中。
PluginSocket	指定用于可加载插件通信的 UNIX 套接字路径。
PluginTimeout	设置与可加载插件建立连接的超时时间。
ProfilerDir	写入性能分析数据的目录。
ProfilerMaxFilesPerProfile	每种性能分析类型保留的文件数量上限。
ProfilerInterval	性能分析数据的写入间隔。
RefreshActiveChecks	指定刷新 active 检查列表的频率。
Server	指定 Zabbix 服务器/proxy 的 IP 地址或 DNS 名称。
ServerActive	指定用于获取 active 检查的 Zabbix 服务器/proxy 地址或集群配置。
SourceIP	指定源 IP 地址。
StatusPort	指定用于监听 HTTP 状态请求的端口（http://localhost:<port>/status）。
Timeout	指定建立连接以及与 Zabbix 服务器或 proxy 交换数据时的等待时间（秒）。
TLSAccept	指定传入连接所需的加密级别。
TLSCAFile	指定包含顶级 CA 证书的文件的完整路径名，用于对等证书验证以及 Zabbix 组件之间的加密通信。
TLSCertFile	指定包含 agent 证书或证书链的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSCipherAll	指定 OpenSSL 密码字符串（TLS 1.2）。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherAll13	为 OpenSSL 1.1.1 或更高版本指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherCert	指定 OpenSSL 密码字符串（TLS 1.2）。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherCert13	为 OpenSSL 1.1.1 或更高版本指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherPSK	指定 OpenSSL 密码字符串（TLS 1.2）。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。

参数	描述
TLSCipherPSK13	为 OpenSSL 1.1.1 或更高版本指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。
TLSConnect	指定连接到 Zabbix 服务器或 proxy 时所需的加密级别。
TLSCRLFile	指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。
TLSKeyFile	指定包含 agent 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSPSKFile	指定包含 agent 预共享密钥的文件的完整路径名，用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSPSKIdentity	指定预共享密钥标识字符串，用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSServerCertIssuer	指定允许的服务器（proxy）证书颁发者。
TLSServerCertSubject	指定允许的服务器（proxy）证书主题。
UnsafeUserParameters	允许将所有字符传递到用户定义参数的参数中。
UserParameter	指定要监控的用户定义参数。
UserParameterDir	指定 UserParameter 命令的默认搜索路径。

请注意：

- 默认值反映的是进程默认值，而不是随附配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持使用 UTF-8 编码且不含 BOM 的配置文件。
- 以 # 开头的注释仅支持出现在行首。

参数详情

别名

为监控项键设置别名。可使用别名将较长且复杂的监控项键替换为更短、更简单的键。
 可以存在多个 Alias 参数。不允许存在多个具有相同 Alias 键的参数。
 不同的 Alias 键可以引用同一个监控项键。
 别名可以在 HostMetadataItem 中使用，但不能在 HostnameItem 参数中使用。

示例 1：获取用户 zabbix 的 ID。

```
Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)",,,,\1]
```

现在可以使用 zabbix.userid 这一简写键来获取数据。

示例 2：使用默认参数和自定义参数获取 CPU 利用率。

```
Alias=cpu.util:system.cpu.util
Alias=cpu.util[*]:system.cpu.util[*]
```

这样，您可以使用 cpu.util 键获取使用默认参数的 CPU 利用率百分比，也可以使用 cpu.util[all, idle, avg15] 获取有关 CPU 利用率的特定数据。

示例 3：运行多个处理相同发现监控项的低级别发现规则。

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

现在可以使用 vfs.fs.discovery 设置多个发现规则，并为每个规则使用不同的参数，例如 vfs.fs.discovery[foo]、vfs.fs.discovery[bar] 等。

AllowKey

允许执行与模式匹配的监控项键值。该键模式是一个通配符表达式，支持 * 字符来匹配任意数量的任意字符。
 多个键匹配规则可以与 DenyKey 结合定义。多个键匹配规则可以与 DenyKey、AllowKeyRegexp 和 DenyKeyRegexp 结合定义。参数会按照其出现顺序逐个处理。该顺序决定了哪个匹配规则具有优先级。另请参见：限制 agent 检查。

AllowKeyRegexp

允许执行与模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 DenyKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现顺序逐一处理。顺序决定哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

示例：

```
AllowKeyRegexp=^system\.run\[(?:/usr)?/bin/hostname(?: -[fsd])?\]$
```

BufferSend

设置将缓冲区中的值发送到 Zabbix 服务器/proxy 的频率（单位：秒）。
 请注意，如果缓冲区已满，数据会更快发送。

默认值：5
 范围：1-3600

BufferSize

设置内存缓冲区中的最大值数量。
 如果缓冲区已满，agent 将把所有已收集的数据发送到 Zabbix 服务器或 proxy。仅在禁用持久缓冲区（EnablePersistentBuffer=0）时才应使用此参数。

默认值：1000
 范围：2-65535

ControlSocket

指定用于通过 -R 选项发送运行时命令的控制套接字。

默认值：/tmp/agent.sock

DebugLevel

设置调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键性信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。

默认值：3
 范围：0-5

DenyKey

拒绝执行与模式匹配的监控项键。
 键模式是通配符表达式，支持使用 * 字符匹配任意数量的任意字符。
 可以结合 AllowKey 定义多个键匹配规则。可以结合 AllowKey、AllowKeyRegexp 和 DenyKeyRegexp 定义多个键匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

默认值：DenyKey=system.run[*]

DenyKeyRegexp

拒绝执行与某个模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 AllowKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。
 默认情况下，所有 system.run[] 监控项都会被拒绝（DenyKey=system.run[*]），除非指定了其他 system.run[*] 规则。

另请参阅：限制 agent 检查。

示例：

```
DenyKeyRegexp=^system\.run\[.*(?:rm -rf|poweroff|reboot).*\]$
```

EnablePersistentBuffer

为主动监控项启用本地持久化存储的使用。
 如果禁用持久化存储，将使用内存缓冲区。

默认值：0
 取值：0 - 禁用，1 - 启用

EnableProfiler

在 agent 启动时启用定期 Go 性能分析。也可以在运行时使用 -R 选项启用或禁用性能分析。在 periodic_prof_execute 收集性能分析数据期间，所有性能分析器运行时命令都会因忙碌而被拒绝。

默认值：0
 值：0 - 禁用，1 - 启用

ForceActiveChecksOnStart

在重启后立即执行主动检查，以获取首次接收到的配置。
 也可作为按插件配置参数使用，例如：Plugins.Uptime.System.ForceActiveCl

默认值：0
 取值：0 - 禁用，1 - 启用

HeartbeatFrequency

设置心跳消息的发送频率，单位为秒。
 用于监控主动检查的可用性。
0 - 禁用心跳消息。

默认值：60
 范围：0-3600

HostInterface

指定在主机自动注册期间使用的主机接口（IP 地址或 DNS 名称）。
 此值将用于填充新创建主机中的接口，并允许显式配置 IP 或 DNS 地址。
 有关更多详细信息，请参见将 DNS 用作默认接口。

如果未定义，则该值将从 HostInterfaceItem 获取。

如果该值超过 255 个字符的限制，agent 将报错且不会启动。

范围：0-255 个字符

HostInterfaceItem

指定一个监控项，用于在主机自动注册期间确定主机接口（IP 地址或 DNS 名称）。
 仅当未定义 HostInterface 时，才会使用此值。
 有关详细信息，请参阅使用 DNS 作为默认接口。

在自动注册请求期间，如果指定监控项返回的值超过 255 个字符，agent 将记录一条警告消息。

无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

HostMetadata

指定用于在主机自动注册期间识别或区分主机的元数据。
 HostMetadata 可用于根据主机名以外的信息区分主机。

如果未定义，将从 HostMetadataItem 获取该值。

如果指定值超过 2034 字节或不是 UTF-8 字符串，agent 将报错并且不会启动。当参数要求 IP 地址或 DNS 名称时，虽然是有效 UTF-8 字符串但不是有效 IP 地址或 DNS 名称的值也会被拒绝，并报告为无效值。

不支持多行元数据——输出将在第一个换行符处被截断。

范围：0-2034 字节

HostMetadataItem

指定一个用于获取主机元数据的监控项。
 仅当未定义 HostMetadata 时，此选项才会生效。

每次尝试自动注册时，都会获取 HostMetadataItem 的值，并且该值仅用于主机自动注册过程。HostMetadataItem 可用于根据主机名以外的信息区分主机。

支持用户参数和别名。无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

在自动注册请求期间，如果指定监控项返回的值超过 65535 个 UTF-8 代码点，agent 将记录警告消息。监控项返回的值必须是 UTF-8 字符串，否则将被忽略。当参数要求 IP 地址或 DNS 名称时，虽然值是有效的 UTF-8 字符串，但如果不是有效的 IP 地址或 DNS 名称，也会被拒绝并报告为无效。

不支持多行元数据——输出将在第一个换行符处被截断。

主机名

指定唯一且区分大小写的主机名。以逗号分隔的列表。
 用于主动检查，并且必须与服务器上配置的主机名一致。
 如果未定义，则从 HostnameItem 获取该值。
 允许的字符：字母数字字符、'.'、' '、'_' 和'-'。每个主机名的最大长度为 128 个字符，整行最大长度为 2048 个字符。

默认值：由 HostnameItem 设置

HostnameItem

指定一个用于获取主机名的监控项。
 仅当未定义 Hostname 时使用此选项。
 不支持用户参数或别名，但无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 的值如何，均支持system.run[] 监控项。

默认值：system.hostname

Include

指定要包含到配置文件中的目录或单个文件。
 在 Zabbix 安装期间，除非你在编译时更改此设置，否则该参数会被设置为 /usr/local/etc。

Include 参数的一般规则：

- 路径可以是相对于 zabbix_agent2.conf 文件位置的相对路径。
- 你可以在一个配置文件中指定多个 Include 参数。
- Zabbix 会在读取到 Include 参数后立即处理它，因此建议将其放在配置文件末尾。

包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，该组件将无法启动。

包含目录或文件模式的规则：

- 不要假设包含顺序是固定的（例如，文件不会按字母顺序包含）。
- 不要在多个已包含文件中定义同一个参数（例如，用特定设置覆盖通用设置），因为你无法预测最终会使用哪个值。
- 注意某些文本编辑器会自动创建的备份副本(例如，编辑 include/my_specific.conf 可能会创建 include/my_specific.conf.BAK)，因为这两个文件都会被包含。

若只想从目录中包含相关文件，请使用通配符（*）进行模式匹配。

示例：

```
Include=/absolute/path/to/config/files/*.conf
```

ListenIP

指定 agent 用于接收来自服务器/proxy 连接的监听 IP 地址。使用逗号分隔的列表。
 如果连接到 Zabbix 服务器，则会将第一个 IP 地址发送给 Zabbix 服务器，以获取主动检查列表。

默认值：0.0.0.0

ListenPort

指定 agent 用于接收来自服务器/proxy 连接的监听端口。

默认值：10050
 范围：1024-32767

LogFile

指定 agent 日志文件。

默认：/tmp/zabbix_agent2.log
 必填：是，如果 LogType 设置为 file；否则不是

LogFileSize

设置最大日志文件大小（以 MB 为单位）。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始写入。

默认值：1
 范围：0-1024

LogType

指定日志输出类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *system* - 将日志写入 syslog；- *console* - 将日志写入标准输出。

默认值：file

PersistentBufferFile

指定用于保存 SQLite 数据库的文件。必须是完整文件名。
 仅在启用持久缓冲区时使用此参数（EnablePersistentBuffer=1）。

PersistentBufferPeriod

设置在无法连接到 Zabbix 服务器或 proxy 时用于存储数据的时间段。较早的数据将会丢失。
 日志数据将被保留。仅在启用持久缓冲区时使用此参数（EnablePersistentBuffer=1）。

默认值：1h
 范围：1m-365d

PidFile

指定 PID 文件。

默认值：/tmp/zabbix_agent2.pid

插件。<PluginName>.System.Capacity

设置 <PluginName> 插件可同时执行的检查数量上限。

默认值：1000 范围：1-1000

Plugins.Log.MaxLinesPerSecond

设置在处理 log 或 logrt 主动检查时，每秒发送到 Zabbix 服务器/proxy 的最大新行数。
 所提供的值将被 log 和 logrt 监控项键中的 maxlines 参数覆盖。
 注意: Zabbix 处理的新行数将是 MaxLinesPerSecond 设置值的 10 倍，以便在日志监控项中查找所需字符串。

默认值: 20
 范围: 1-1000

Plugins.SystemRun.LogRemoteCommands

将已执行的 shell 命令记录为警告。
 仅当命令通过远程执行时才会记录。
 如果 system.run[] 由 HostMetadataItem、HostInterfaceItem 或 HostnameItem 参数在本地启动，则不会创建日志条目。

默认值：0
 取值：0 - 禁用，1 - 启用

PluginSocket

指定用于可加载插件通信的 UNIX socket 路径。

默认值：/tmp/agent.plugin.sock

PluginTimeout

指定可加载插件连接的超时时间，单位为秒。

默认值：Timeout
 范围：1-30

ProfilerDir

写入性能分析数据的目录。

默认值：/tmp/zabbix_agent2_profiles

ProfilerMaxFilesPerProfile

每种 profile 类型保留的文件最大数量。该限制会在每次 profile 转储后应用于已完成的文件。活动的 CPU profile 会收集到隐藏的临时文件中，不计入该限制。periodic_prof_execute 创建的文件同样受此限制约束。

默认值：10
 范围：1-100

ProfilerInterval

性能分析数据写入间隔，以秒为单位。

默认值：3600
 范围：1-86400

RefreshActiveChecks

指定刷新主动检查列表的频率，单位为秒。
 请注意，在刷新主动检查失败后，下一次刷新将在 60 秒后尝试。

默认值：5
 范围：1-86400

服务器

指定 Zabbix 服务器/proxy 的 IP 地址（可选使用 CIDR 表示法）或 DNS 名称。以逗号分隔的列表。
 仅会接受来自此处列出的主机的传入连接。如果启用了 IPv6 支持，则 127.0.0.1、::127.0.0.1、::ffff:127.0.0.1 会被视为相同，::/0 将允许任何 IPv4 或 IPv6 地址。0.0.0.0/0 可用于允许任何 IPv4 地址。允许使用空格。如果未指定此参数，将禁用被动检查，且 agent 不会监听任何 TCP 端口。

示例：

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

ServerActive

指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
 服务器/proxy 地址是 IP 地址或 DNS 名称，端口可选，使用冒号分隔。
 集群配置是一个或多个服务器或 proxy 组成员地址，使用分号分隔。可以指定多个 Zabbix 服务器/集群和 Zabbix proxy，并用逗号分隔。除非使用 proxy 组，否则不应为每个 Zabbix 服务器/集群指定多个 Zabbix proxy。如果指定了 Zabbix proxy，则不应再指定该 proxy 对应的 Zabbix 服务器/集群。
 可以提供多个以逗号分隔的地址，以并行使用多个独立的 Zabbix 服务器。允许使用空格。
 如果未指定端口，则使用默认端口。
 如果为该主机指定了端口，则 IPv6 地址必须用方括号括起来。如果未指定端口，则 IPv6 地址的方括号可选。
 如果未指定此参数，则会禁用主动检查。

Zabbix proxy 示例：

```
ServerActive=127.0.0.1:10051
```

Zabbix proxy 组示例：

```
ServerActive=proxy1.example.com;proxy2.example.com;proxy3.example.com;proxy4.example.com;proxy5.example.co
```

多个服务器示例：

```
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
```

高可用示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

包含两个集群和一个服务器的高可用示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

SourceIP

为以下连接指定源 IP 地址：

- 到 Zabbix 服务器或 Zabbix proxy 的出站连接。
- 在执行某些监控项（web.page.get、net.tcp.port 等）时建立连接。

StatusPort

指定用于监听 HTTP 状态请求的端口（http://localhost:<port>/status）。

范围：1024-32767

超时

指定建立连接并与 Zabbix proxy 或服务器交换数据时的等待时间（以秒为单位）。

此参数定义了各种通信操作的持续时间，包括：

- 等待来自 Zabbix 服务器的响应；
- 向 Zabbix 服务器发送请求，包括监控项配置请求以及active checks 中的监控项数据；
- 通过 logfile 获取日志数据；
- 发送心跳消息；
- `vfs.*` 检查的最长持续时间；
- 在服务器或 proxy 版本早于 7.0 且发送不带超时的检查的场景中，作为回退机制。

对于那些在前端中具有可配置超时设置的 agent 检查（全局、proxy 或单个监控项级别），此超时不会被使用。

默认值：3
 范围：1-30

TLSAccept

指定传入连接所需的加密级别。
 用于被动检查。
 可以指定多个值，并用逗号分隔：- *unencrypted* - 接受未加密连接（默认）；- *psk* - 接受使用 TLS 和预共享密钥（PSK）的连接；- *cert* - 接受使用 TLS 和证书的连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，该证书用于 Zabbix 组件之间的加密通信。

TLSCertFile

指定包含 agent 证书或证书链的文件的完整路径名，用于与 Zabbix 组件之间的加密通信。

TLSCipherAll

指定 OpenSSL 密码字符串（TLS 1.2）。覆盖基于证书和基于 PSK 的加密所使用的默认密码套件选择标准。

示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherAll13

为 OpenSSL 1.1.1 或更新版本指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书和基于 PSK 的加密的默认密码套件选择标准。

示例：

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherCert

指定 OpenSSL 密码字符串（TLS 1.2）。覆盖基于证书加密的默认密码套件选择条件。

示例：

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

请注意，此参数不能与 `TLSAccept=cert,psk` 同时使用；对于证书连接（`TLSConnect=cert`），请改用 `TLSCipherAll`。

TLSCipherCert13

为 OpenSSL 1.1.1 或更新版本指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书的加密所使用的默认密码套件选择条件。

请注意，此参数不能与 `TLSAccept=cert,psk` 一起使用；对于证书连接（`TLSConnect=cert`），请改用 `TLSCipherAll13`。

TLSCipherPSK

指定 OpenSSL 密码字符串（TLS 1.2）。覆盖基于 PSK 的加密所使用的默认密码套件选择条件。

示例：

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

为 OpenSSL 1.1.1 或更新版本指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于 PSK 的加密所使用的默认密码套件选择条件。

示例：

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSConnect

指定 agent 连接到 Zabbix 服务器或 proxy 所需的加密级别。
 用于主动检查。
 只能指定一个值：- *unencrypted* - 连接时不使用加密（默认）；- *psk* - 使用 TLS 和预共享密钥（PSK）连接；- *cert* - 使用 TLS 和证书连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCRLFile

指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。

TLSKeyFile

指定包含 agent 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSPSKFile

指定包含 agent 预共享密钥的文件的完整路径名，该密钥用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSPSKIdentity

指定预共享密钥标识字符串，用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSServerCertIssuer

指定允许的服务器（proxy）证书颁发者。

TLSServerCertSubject

指定允许的服务器（proxy）证书主题。

UnsafeUserParameters

允许在参数中传递所有字符到用户自定义参数。以下字符不允许使用：\ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ %
 此外，不允许使用换行符。

默认值：0
 取值：0 - 不允许，1 - 允许

UserParameter

指定一个用户自定义参数用于监控。可以有多个用户自定义参数。
 格式：UserParameter=<key>,<shell command>
 请注意，shell 命令不能只返回空字符串或仅返回行尾符。如果指定了 UserParameterDir 参数，shell 命令可以使用相对路径。

示例：

```
UserParameter=system.test,who|wc -l
UserParameter=check_cpu,./custom_script.sh
```

UserParameterDir

指定 UserParameter 命令的默认搜索路径。如果使用此参数，agent 会在执行命令之前将其工作目录更改为此处指定的目录。因此，UserParameter 命令可以使用相对的 ./ 前缀，而不是完整路径。
 只允许一条配置项。

示例：

```
UserParameterDir=/opt/myscripts
```

配置参数：**Zabbix agent 2**（**Windows**）

概述

配置参数可用于自定义 Windows 上 Zabbix agent 2 的行为。

在 Zabbix agent 2 配置文件（zabbix_agent2.conf）中指定参数值。除非明确说明参数为必需参数，否则所有参数均为可选。

本页面结构如下：

- 参数概览（要查看参数的完整详细信息，请单击其名称）
- 参数详细信息

参数	描述
Alias	为监控项键设置别名。
AllowKey	允许执行与某个模式匹配的监控项键。
AllowKeyRegexp	允许执行与基于正则表达式的模式匹配的监控项键。
BufferSend	设置将值从缓冲区发送到 Zabbix 服务器的频率。
BufferSize	设置内存缓冲区中的最大值数量。
ControlSocket	指定控制套接字，用于通过 -R 选项发送运行时命令。
DebugLevel	设置调试级别。
DenyKey	拒绝执行与某个模式匹配的监控项键。
DenyKeyRegexp	拒绝执行与基于正则表达式的模式匹配的监控项键。

参数	描述
EnablePersistentBuffer	启用对活动监控项使用本地持久化存储。
EnableProfiler	启用 agent 启动时定期进行 Go 性能分析。
ForceActiveChecksOnStart	在重启后收到第一个配置时立即执行主动检查。
HeartbeatFrequency	设置心跳消息的频率（秒）。
HostInterface	指定主机接口。
HostInterfaceItem	指定用于获取主机接口的监控项。
HostMetadata	指定主机元数据。
HostMetadataItem	指定用于获取主机元数据的 Zabbix agent 监控项。
Hostname	指定主机名。
HostnameItem	指定用于获取主机名的 Zabbix agent 监控项。
Include	指定要包含在配置文件中的目录或单个文件。
ListenIP	指定 agent 用于接收来自服务器/proxy 的连接的监听 IP 地址。
ListenPort	指定 agent 监听端口。
LogFile	指定 agent 日志文件。
LogFileSize	设置日志文件的最大大小。
LogType	指定日志输出类型。
PerfCounter	指定新参数 <parameter_name>，该参数表示在指定时间段 <period>（秒）内系统性能计数器 <perf_counter_path> 的平均值。
PerfCounterEn	指定新参数 <parameter_name>，该参数表示在指定时间段 <period>（秒）内系统性能计数器 <perf_counter_path> 的平均值。与 PerfCounter 相比，性能计数器路径必须使用英文。
PersistentBufferFile	指定 Zabbix agent 2 保存 SQLite 数据库的文件。
PersistentBufferPeriod	设置在无法连接服务器或 proxy 时保存数据的持续时间。
Plugins.<PluginName>.System.Capacity	设置每个插件可同时执行的检查数量上限。
Plugins.Log.MaxLinesPerSecond	在处理 `log`、`logrt` 和 `eventlog` 主动检查时，设置 agent 每秒向 Zabbix 服务器或 proxy 发送的新行数上限。
Plugins.SystemRun.LogRemoteCommands	启用将已执行的 shell 命令作为警告记录到日志中。
PluginSocket	指定用于可加载插件通信的 Windows 命名管道路径。
PluginTimeout	设置与可加载插件建立连接的超时时间（秒）。
ProfilerDir	写入性能分析数据的目录。
ProfilerMaxFilesPerProfile	设置每种性能分析类型要保留的最大文件数。
ProfilerInterval	设置性能分析数据写入间隔。
RefreshActiveChecks	指定刷新主动检查列表的频率。
Server	指定 Zabbix 服务器/proxy 的 IP 地址或 DNS 名称。
ServerActive	指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
SourceIP	指定源 IP 地址。
StatusPort	设置 agent 监听 HTTP 状态请求的端口（`http://localhost:<port>/status`）。
Timeout	指定建立连接以及与 Zabbix 服务器或 proxy 交换数据时的等待时间（秒）。
TLSAccept	指定传入连接所需的加密级别。
TLSCAFile	指定包含顶级 CA 证书的文件的完整路径名，用于对对等证书进行验证，并用于 Zabbix 组件之间的加密通信。
TLSCertFile	指定包含 agent 证书或证书链的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSCipherAll13	指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。
TLSCipherCert13	指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书的加密所使用的默认密码套件选择标准。
TLSCipherPSK13	指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于 PSK 的加密所使用的默认密码套件选择标准。
TLSConnect	指定连接到 Zabbix 服务器或 proxy 时所需的加密级别。
TLSCRLFile	指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。
TLSKeyFile	指定包含 agent 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。
TLSPSKFile	指定包含 agent 预共享密钥的文件的完整路径名，用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSPSKIdentity	指定预共享密钥标识字符串，用于与 Zabbix 服务器或 proxy 进行加密通信。
TLSServerCertIssuer	指定允许的服务器（proxy）证书颁发者。
TLSServerCertSubject	指定允许的服务器（proxy）证书主题。
UnsafeUserParameters	允许将所有字符传递到用户自定义参数的参数中。
UserParameter	指定要监控的用户自定义参数。
UserParameterDir	指定 `UserParameter` 命令的默认搜索路径。

请注意：

- 默认值反映的是进程默认值，而不是随附配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持使用 UTF-8 编码且不带 BOM 的配置文件；
- 以 # 开头的注释仅支持出现在行首。

参数详情

别名

为监控项键设置别名。可使用别名将较长且复杂的监控项键替换为更短、更简单的键。
 可以存在多个 Alias 参数。不允许存在多个具有相同 Alias 键的参数。
 不同的 Alias 键可以引用同一个监控项键。
 别名可以在 HostMetadataItem 中使用，但不能在 HostnameItem 参数中使用。

示例 1：从服务器获取分页文件使用率（百分比）。

```
Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage]
```

现在可以使用简写键 pg_usage 获取数据。

示例 2：使用默认参数和自定义参数获取 CPU 负载。

```
Alias=cpu.load:system.cpu.load
Alias=cpu.load[*]:system.cpu.load[*]
```

这样，您可以使用 cpu.load 键获取使用默认参数的 CPU 负载，也可以使用 cpu.load[percpu,avg15] 获取有关 CPU 负载的特定数据。

示例 3：运行多个处理相同发现监控项的低级别发现规则。

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

现在可以使用 vfs.fs.discovery 设置多个发现规则，并为每个规则指定不同的参数，例如 vfs.fs.discovery[foo]、vfs.fs.discovery[bar] 等。

AllowKey

允许执行与模式匹配的监控项键。键模式是一个通配符表达式，支持 * 字符来匹配任意数量的任意字符。
 可以将多个键匹配规则与 DenyKey、AllowKeyRegexp 和 DenyKeyRegexp 组合定义。这些参数会按照它们出现的顺序逐一处理。该顺序决定了哪个匹配规则具有优先级。另请参见：限制 agent 检查。

AllowKeyRegexp

允许执行与模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 DenyKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。另请参阅：限制 agent 检查。

示例：

```
AllowKeyRegexp=^system\.run\[(?:/usr)?/bin/hostname(?: -[fsd])?\]$
```

BufferSend

设置将缓冲区中的值发送到 Zabbix 服务器/proxy 的频率（以秒为单位）。
 请注意，如果缓冲区已满，数据将更早发送。

默认值：5
 范围：1-3600

BufferSize

设置内存缓冲区中的最大值数量。
 如果缓冲区已满，agent 将把所有已收集的数据发送到 Zabbix 服务器或 proxy。
 仅在禁用持久缓冲区时才应使用此参数（EnablePersistentBuffer=0）。

默认值：1000
 范围：2-65535

ControlSocket

指定控制套接字，用于通过 -R 选项发送运行时命令。

默认值：\\.\pipe\agent.sock

DebugLevel

设置调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。

默认值：3
 范围：0-5

DenyKey

拒绝执行与模式匹配的监控项键值。键值模式是通配符表达式，支持使用 * 字符匹配任意数量的任意字符。
 可以结合 AllowKey、AllowKeyRegexp 和 DenyKeyRegexp 定义多个键值匹配规则。参数将按照其出现的顺序逐一处理。顺序决定哪个匹配规则具有优先级。

另请参阅：限制 agent 检查。

默认值：DenyKey=system.run[*]

DenyKeyRegexp

拒绝执行与某个模式匹配的监控项键值。键值模式是一个正则表达式。
 可以结合 AllowKeyRegexp、AllowKey 和 DenyKey 定义多个键值匹配规则。参数会按照其出现顺序逐一处理。该顺序决定哪个匹配规则具有优先级。
 默认情况下，所有 system.run[] 监控项都会被拒绝（DenyKey=system.run[*]），除非指定了其他 system.run[*] 规则。

另请参阅：限制 agent 检查。

EnablePersistentBuffer

为主动监控项启用本地持久化存储。如果禁用持久化存储，将使用内存缓冲区。

默认值：0
 取值：0 - 禁用，1 - 启用

EnableProfiler

在 agent 启动时启用定期 Go 性能分析。也可以在运行时使用 -R 选项启用或禁用性能分析。在 periodic_prof_execute 收集性能分析数据期间，所有性能分析器运行时命令都会因忙碌而被拒绝。

默认值：0
 值：0 - 禁用，1 - 启用

ForceActiveChecksOnStart

在重启后立即执行主动检查，以获取首次接收到的配置。也可作为按插件配置参数使用，例如：Plugins.Uptime.System.ForceActiveChecks

默认值：0
 取值：0 - 禁用，1 - 启用

HeartbeatFrequency

设置心跳消息的发送频率，单位为秒。用于监控主动检查的可用性。
0 - 禁用心跳消息。

默认值：60
 范围：0-3600

HostInterface

指定在主机自动注册 期间使用的主机接口（IP 地址或 DNS 名称）。此值将用于填充新创建主机中的接口，并允许显式配置 IP 或 DNS 地址。有关更多详细信息，请参见将 DNS 用作默认接口。

如果未定义，则该值将从 HostInterfaceItem 获取。

如果该值超过 255 个字符的限制，agent 将报错且不会启动。

范围：0-255 个字符

HostInterfaceItem

指定一个监控项，用于在主机自动注册期间确定主机接口（IP 地址或 DNS 名称）。仅当未定义 HostInterface 时才使用此值。有关更多详细信息，请参阅使用 DNS 作为默认接口。

在自动注册请求期间，如果指定监控项返回的值超过 255 个字符，agent 将记录一条警告消息。

无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

HostMetadata

指定用于在主机自动注册（主动 agent）期间识别或区分主机的元数据。
HostMetadata 可用于在主机名之外进一步区分主机。

如果未定义，将从 HostMetadataItem 获取该值。

如果指定的值超过 2034 字节或不是 UTF-8 字符串，agent 将报错并且不会启动。
当参数要求 IP 地址或 DNS 名称时，虽然是有效 UTF-8 字符串但不是有效 IP 地址或 DNS 名称的值也会被拒绝，并报告为无效。

不支持多行元数据——输出将在第一个换行符处被截断。

范围：0-2034 字节

HostMetadataItem

指定一个 Zabbix agent 监控项，以获取主机元数据。仅当未定义 HostMetadata 时，才会使用此选项。

每次尝试自动注册时都会获取 HostMetadataItem 的值，并且该值仅用于主机自动注册过程（主动 agent）。HostMetadataItem 可用于在主机名之外进一步区分主机。

支持用户参数和别名。无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 设置如何，均支持system.run[] 监控项。

在自动注册请求期间，如果指定监控项返回的值超过 65535 个 UTF-8 代码点，agent 将记录警告消息。监控项返回的值必须是 UTF-8 字符串，否则将被忽略。当参数要求 IP 地址或 DNS 名称时，虽然值是有效的 UTF-8 字符串，但如果不是有效的 IP 地址或 DNS 名称，也会被拒绝并报告为无效。

不支持多行元数据——输出将在第一个换行符处被截断。

主机名

指定唯一、区分大小写的主机名。以逗号分隔的列表。
 用于主动检查，并且必须与服务器上配置的主机名匹配。如果未定义，则从 HostnameItem 获取该值。
 允许的字符：字母数字字符、'.'、' '、'_' 和'-'。最大长度：每个主机名 128 个字符，整行 2048 个字符。

默认值：由 HostnameItem 设置

HostnameItem

指定一个用于获取主机名的监控项。仅当未定义 Hostname 时使用此选项。不支持用户参数或别名，但无论 AllowKey、AllowKeyRegexp、DenyKey 或 DenyKeyRegexp 的值如何，均支持system.run[] 监控项。

默认值：system.hostname

Include

指定要包含在配置文件中的目录或单个文件（如果使用 Windows MSI 安装程序包安装 Zabbix agent，配置文件默认位于 C:\Program Files\Zabbix Agent 2；如果将 Zabbix agent 安装为 zip 压缩包，则配置文件位于安装期间指定的文件夹中）。在安装 Zabbix 期间，此参数设置为 /usr/local/etc，除非您在编译时更改了该设置。

Include 参数的一般规则：

- 路径可以相对于 zabbix_agent2.conf 文件的位置（例如，Include=.\zabbix_agent2.d\plugins.d*.conf）。
- 可以在一个配置文件中指定多个 Include 参数。
- Zabbix 在读取 Include 参数后会立即对其进行处理，因此请考虑将其放置在配置文件末尾。

所包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，组件将无法启动。

包含目录或文件模式的规则：

- 不要假定包含文件的顺序（例如，文件不会按字母顺序包含）。
- 不要在多个所包含文件中定义相同的参数（例如，使用特定设置覆盖常规设置），因为无法预测将使用哪个值。
- 注意某些文本编辑器会自动创建备份副本(例如，编辑 include/my_specific.conf 可能会创建 include/my_specific.conf.BAK)，因为这两个文件都会被包含。

要仅包含目录中的相关文件，请使用通配符 (*) 进行模式匹配。

示例：

```
Include=C:\Program Files\Zabbix Agent2\zabbix_agent2.d\*.conf
```

ListenIP

指定 agent 用于接收来自服务器/proxy 连接的监听 IP 地址。使用逗号分隔的列表。
 如果连接到 Zabbix 服务器，则会将第一个 IP 地址发送给 Zabbix 服务器，以获取主动检查列表。

默认值：0.0.0.0

ListenPort

指定 agent 用于接收来自服务器/proxy 连接的监听端口。

默认值：10050
 范围：1024-32767

LogFile

指定 agent 日志文件。

默认值：c:\zabbix_agent2.log
 必填：是，如果 LogType 设置为 file；否则为否

LogFileSize

设置最大日志文件大小（单位：MB）。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始。

默认值：1
 范围：0-1024

LogType

指定日志输出类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *console* - 将日志写入标准输出。

默认值：file

PerfCounter

定义一个新参数 <parameter_name>，它是系统性能计数器 <perf_counter_path> 在指定时间段 <period>（以秒为单位）内的平均值。
 语法：<parameter_name>,"<perf_counter_path>",<period>

例如，如果您希望获取最近一分钟内每秒处理器中断的平均次数，可以将新参数"interrupts" 定义如下：

```
PerfCounter = interrupts,"\Processor(0)\Interrupts/sec",60
```

请注意性能计数器路径周围的双引号。参数名称（interrupts）在创建监控项时将用作监控项键。用于计算平均值的样本将每秒采集一次。
 您可以运行 typeperf -qx 来获取 Windows 中可用的所有性能计数器列表。

PerfCounterEn

定义一个新参数 <parameter_name>，它是指定时间段 <period>（以秒为单位）内系统性能计数器 <perf_counter_path> 的平均值。与 PerfCounter 相比，perfcounter 路径必须使用英文。仅支持 **Windows Server 2008/Vista** 及更高版本。
 语法：<parameter_name>,"<perf_counter_path>",<period>

例如，如果您希望获取最近一分钟内每秒处理器中断的平均次数，可以按如下方式定义一个新参数"interrupts"：

```
PerfCounterEn = interrupts,"\Processor(0)\Interrupts/sec",60
```

请注意性能计数器路径周围的双引号。参数名称（interrupts）在创建监控项时将用作监控项键值。用于计算平均值的样本将每秒采集一次。
 您可以通过查看以下注册表项找到英文字符串列表：HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009。

PersistentBufferFile

指定 Zabbix agent 2 应保存 SQLite 数据库的文件。必须是完整的文件名。
 仅在启用持久缓冲区时使用此参数(EnablePersistentBuffer=1)。

PersistentBufferPeriod

设置在无法连接到服务器或 proxy 时数据的存储时长。较旧的数据将丢失。日志数据将被保留。
 仅在启用持久缓冲区时使用此参数(EnablePersistentBuffer=1)。

默认值：1h
 范围：1m-365d

插件.<PluginName>.System.Capacity

设置 <PluginName> 插件可同时执行的检查数量上限。

默认值：1000 范围：1-1000

Plugins.Log.MaxLinesPerSecond

设置 agent 在处理 log、logrt 和 eventlog 主动检查时，每秒发送给 Zabbix 服务器或 proxy 的最大新行数。所提供的值将被 log、logrt 或 eventlog 监控项键中的 maxlines 参数覆盖。
 注意: Zabbix 处理的新行数将是 MaxLinesPerSecond 设置值的 10 倍，以便在日志监控项中查找所需字符串。

默认值: 20
 范围: 1-1000

Plugins.SystemRun.LogRemoteCommands

启用将已执行的 shell 命令记录为警告。仅当命令是远程执行时，才会记录。如果 system.run[] 由 HostMetadataItem、HostInterfaceItem 或 HostnameItem 参数在本地启动，则不会创建日志条目。

默认值：0
 取值：0 - 禁用，1 - 启用

PluginSocket

指定用于可加载插件通信的 Windows 命名管道路径。

默认值：\\.\pipe\agent.plugin.sock

PluginTimeout

设置与可加载插件的连接超时时间，单位为秒。

默认值：Timeout
 范围：1-30

ProfilerDir

写入性能分析数据的目录。

默认值：c:\zabbix_pprof

ProfilerMaxFilesPerProfile

每种 profile 类型保留的文件最大数量。该限制会在每次 profile 转储后应用于已完成的文件。活动的 CPU profile 会收集到隐藏的临时文件中，不计入该限制。`periodic_prof_execute` 创建的文件同样受此限制约束。

默认值：10
 范围：1-100

ProfilerInterval

性能分析数据写入间隔，以秒为单位。

默认值：3600
 范围：1-86400

RefreshActiveChecks

指定刷新 active checks 列表的频率，单位为秒。
 请注意，在刷新 active checks 失败后，下一次刷新将在 60 秒后尝试。

默认值：5
 范围：1-86400

服务器

指定 Zabbix 服务器/proxy 的 IP 地址（可选使用 CIDR 表示法）或 DNS 名称。以逗号分隔的列表。
 如果未指定该参数，则会禁用被动检查，且 agent 不会监听任何 TCP 端口。
 传入连接只会接受来自此处列出的主机。如果启用了 IPv6 支持，则 `127.0.0.1`、`::127.0.0.1`、`::ffff:127.0.0.1` 会被视为相同，且 `::/0` 将允许任何 IPv4 或 IPv6 地址。`0.0.0.0/0` 可用于允许任何 IPv4 地址。允许使用空格。

示例：

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

ServerActive

指定用于获取主动检查的 Zabbix 服务器/proxy 地址或集群配置。
 如果未指定该参数，则会禁用主动检查。

服务器/proxy 地址可以是 IP 地址或 DNS 名称，并可选指定端口，端口通过冒号分隔。如果未指定端口，则使用默认端口。
 如果为该主机指定了端口，则 IPv6 地址必须用方括号括起来。如果未指定端口，则 IPv6 地址的方括号可选。

集群配置是一个或多个服务器或 proxy 组成员地址，地址之间用分号分隔。

可以指定多个 Zabbix 服务器/集群和 Zabbix proxy，之间用逗号分隔。

除非使用 proxy 组，否则每个 Zabbix 服务器/集群不应指定多个 Zabbix proxy。如果指定了 Zabbix proxy，则不应再为该 proxy 指定 Zabbix 服务器/集群。

可以提供多个以逗号分隔的地址，以并行使用多个独立的 Zabbix 服务器。允许包含空格。

Zabbix proxy 示例：

```
ServerActive=127.0.0.1:10051
```

Zabbix proxy 组示例：

```
ServerActive=proxy1.example.com;proxy2.example.com;proxy3.example.com;proxy4.example.com;proxy5.example.co
```

多个服务器示例：

```
ServerActive=127.0.0.1:20051,zabbix.domain,\[::1\]:30051,::1,\[12fc::1\]
```

高可用示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

包含两个集群和一个服务器的高可用示例：

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

SourceIP

为以下连接指定源 IP 地址：

- 到 Zabbix 服务器或 Zabbix proxy 的出站连接；
- 在执行某些监控项（`web.page.get`、`net.tcp.port` 等）时建立连接。

StatusPort

设置 agent 监听 HTTP 状态请求的端口（`http://localhost:<port>/status`）。

范围：1024-32767

超时

指定建立连接以及与 Zabbix proxy 或服务器交换数据时的等待时间（秒）。

此参数定义了各种通信操作的持续时间，包括：

- 等待来自 Zabbix 服务器的响应；
- 向 Zabbix 服务器发送请求，包括监控项配置请求以及主动检查 中的监控项数据；
- 通过 logfile 或 Windows 事件日志监控检索日志数据；
- 发送心跳消息；
- `vfs.*` 检查的最长持续时间；
- 在服务器或 proxy 版本早于 7.0 且发送不带超时的检查时，作为回退机制。

对于那些在前端中可配置超时设置的 agent 检查（全局、proxy 或单个监控项级别），此超时不会被使用。

默认值：3
 范围：1-30

TLSAccept

指定传入连接所需的加密级别。
 用于被动检查。

可以指定多个值，并用逗号分隔：

- *unencrypted* - 接受未加密连接（默认）；
- *psk* - 接受使用 TLS 和预共享密钥（PSK）的连接；
- *cert* - 接受使用 TLS 和证书的连接。

必填：是，如果已定义 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSCertFile

指定包含 agent 证书或证书链的文件的完整路径名，用于与 Zabbix 组件之间的加密通信。

TLSCipherAll13

指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书和 PSK 的加密所使用的默认密码套件选择标准。

示例：

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherCert13

指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于证书的加密所使用的默认密码套件选择条件。

请注意，此参数不能与 `TLSAccept=cert,psk` 同时使用；对于证书连接（`TLSConnect=cert`），请改用 `TLSCipherAll13`。

TLSCipherPSK13

指定 OpenSSL 密码字符串（TLS 1.3）。覆盖基于 PSK 的加密的默认密码套件选择标准。

示例：

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSConnect

指定 agent 连接到 Zabbix 服务器或 proxy 所需的加密级别。
 用于主动检查。
 只能指定一个值：- *unencrypted* - 连接时不使用加密（默认）；- *psk* - 使用 TLS 和预共享密钥（PSK）连接；- *cert* - 使用 TLS 和证书连接。

必填：是，如果定义了 TLS 证书或 PSK 参数（即使是 *unencrypted* 连接）；否则为否

TLSCRLFile

指定包含已吊销证书的文件的完整路径名。此参数用于 Zabbix 组件之间的加密通信。

TLSKeyFile

指定包含 agent 私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

TLSPSKFile

指定包含 agent 预共享密钥的文件的完整路径名，用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSPSKIdentity

指定预共享密钥标识字符串，用于与 Zabbix 服务器或 proxy 进行加密通信。

TLSServerCertIssuer

指定允许的服务器（proxy）证书颁发者。

TLSServerCertSubject

指定允许的服务器（proxy）证书主题。

UnsafeUserParameters

允许在参数中传递所有字符到用户自定义参数。以下字符不允许使用：\ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ %
 此外，不允许使用换行符。

默认值：0
 取值：0 - 不允许，1 - 允许

UserParameter

指定一个用户自定义参数用于监控。可以有多个用户自定义参数。
 格式：UserParameter=<key>,<shell command>
 请注意，shell 命令不能只返回空字符串或仅返回 EOL。如果指定了 UserParameterDir 参数，shell 命令可以使用相对路径。

示例：

```
UserParameter=system.test,dir /b | find /c /v ""
UserParameter=check_cpu,.\custom\_script.bat
```

UserParameterDir

指定 UserParameter 命令的默认搜索路径。如果使用此参数，agent 会在执行命令之前将其工作目录切换到此处指定的目录。因此，UserParameter 命令可以使用相对的 .\ 前缀，而不是完整路径。
 仅允许一条配置项。

示例：

```
UserParameterDir=C:\opt\myscripts
```

配置参数：插件

请参阅 Zabbix agent 2 插件的配置参数：

- Ceph 插件
- Docker 插件
- Ember+ 插件
- Memcached 插件
- Modbus 插件
- MongoDB 插件
- MQTT 插件
- MSSQL 插件
- MySQL 插件
- NVIDIA GPU 插件
- Oracle 插件
- PostgreSQL 插件
- Redis 插件
- S.M.A.R.T. 插件

插件配置原则

本页面介绍插件配置原则和最佳实践。

所有插件都使用 Plugins.* 参数进行配置，该参数可以位于 Zabbix agent 配置文件中，也可以位于插件配置文件中。如果插件使用单独的配置文件，则应在 Zabbix agent 2 配置文件的Include 参数中指定该文件的路径。

典型的插件参数具有以下结构：

Plugins.<PluginName>.<Parameter>=<Value>

此外，还有两组特定的参数：

- Plugins.<PluginName>.Default.<Parameter>=<Value> 用于定义默认参数值。
- Plugins.<PluginName>.<SessionName>.<Parameter>=<Value> 用于通过命名会话为不同的监控目标定义单独的参数集。

所有参数名称都应符合以下要求：

- 建议将插件名称首字母大写。
- 参数名称首字母必须大写。
- 不允许使用特殊字符。
- 嵌套层级不受最大级别限制。

- 参数数量不受限制。

例如，要仅对 Uptime 插件执行主动检查，并且在 agent 重启后立即执行不具有 *Scheduling* 更新间隔的主动检查，请在配置文件中设置 `Plugins.Uptime.System.ForceActiveChecksOnStart=1`。同样，要为 CPU 插件设置并发检查的自定义限制，请在配置文件中设置 `Plugins.CPU.System.Capacity=N`。

默认值

您可以在配置文件中为连接相关参数（URI、用户名、密码等）设置默认值，格式如下：

```
Plugins.<PluginName>.Default.<Parameter>=<Value>
```

例如，`Plugins.Mysql.Default.Username=zabbix`、`Plugins.MongoDB.Default.Uri=tcp://127.0.0.1:27017` 等。

如果监控项键或命名会话参数中未提供此类参数的值，插件将使用默认值。如果默认参数也未定义，则使用硬编码的默认值。

> **Note:**
> 如果监控项键不包含任何参数，Zabbix agent 2 将尝试使用默认参数部分中定义的值采集指标。

命名会话

命名会话表示插件参数的一个额外层级，可用于为被监控的每个实例指定一组单独的认证参数。
每个命名会话参数应具有以下结构：

```
Plugins.<PluginName>.Sessions.<SessionName>.<Parameter>=<Value>
```

会话名称可用作 connString 监控项键参数，而无需分别指定 URI、用户名和/或密码。

在监控项键中，第一个参数可以是 connString 或 URI。
如果第一个键参数与任何会话名称都不匹配，则会将其视为 URI。
请注意，监控项键中不支持传递嵌入式 URI 凭据，请改用命名会话参数。

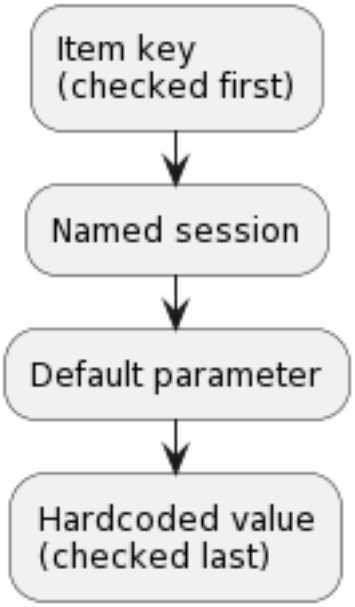
可用的命名会话参数 列表取决于插件。

可以通过在监控项键参数中指定新值来覆盖会话参数（请参见示例）。

如果某个参数未为命名会话定义，Zabbix agent 2 将使用默认插件参数 中定义的值。

参数优先级

Zabbix agent 2 插件会按以下顺序搜索与连接相关的参数值：



1. 第一个监控项键值参数会与会话名称进行比较。
   如果未找到匹配项，则将其视为实际值；在这种情况下，将跳过步骤 3。
   如果找到匹配项，则必须在指定名称的会话中定义该参数值（通常是 URI）。
2. 其他参数如果已定义，则会从监控项键值中获取。
3. 如果某个监控项键值参数（例如 password）为空，插件将查找对应的指定会话参数。
4. 如果会话参数也未指定，则将使用对应默认参数中定义的值。
5. 如果以上方法都失败，插件将使用硬编码的默认值。

示例 1

对两个实例"MySQL1" 和"MySQL2" 进行监控。

配置参数：

```
Plugins.Mysql.Sessions.MySQL1.Uri=tcp://127.0.0.1:3306
Plugins.Mysql.Sessions.MySQL1.User=mysql1_user
Plugins.Mysql.Sessions.MySQL1.Password=unique_password
Plugins.Mysql.Sessions.MySQL2.Uri=tcp://192.0.2.1:3306
Plugins.Mysql.Sessions.MySQL2.User=mysql2_user
Plugins.Mysql.Sessions.MySQL2.Password=different_password
```

由于此配置，每个会话名称都可以在监控项键值 中用作 connString，例如 `mysql.ping[MySQL1]` 或 `mysql.ping[MySQL2]`。

示例 2

在监控项键中提供部分参数。

配置参数：

```
Plugins.PostgreSQL.Sessions.Session1.Uri=tcp://192.0.2.234:5432
Plugins.PostgreSQL.Sessions.Session1.User=old_username
Plugins.PostgreSQL.Sessions.Session1.Password=session_password
```

监控项键: `pgsql.ping[session1,new_username,,postgres]`

由于此配置，agent 将使用以下参数连接到 PostgreSQL：

- 来自会话参数的 URI：`192.0.2.234:5432`
- 来自监控项键的用户名：`new_username`
- 来自会话参数的密码（因为在监控项键中省略了该参数）：`session_password`
- 来自监控项键的数据库名：`postgres`

示例 3

使用默认配置参数采集监控项。

配置参数：

```
Plugins.PostgreSQL.Default.Uri=tcp://192.0.2.234:5432
Plugins.PostgreSQL.Default.User=zabbix
Plugins.PostgreSQL.Default.Password=password
```

监控项键值: `pgsql.ping[,,,postgres]`

由于此配置，agent 将使用以下参数连接到 PostgreSQL：

- 默认 URI: `192.0.2.234:5432`
- 默认用户名: `zabbix`
- 默认密码: `password`
- 来自监控项键值的数据库名称: `postgres`

连接

某些插件支持同时从多个实例收集指标。本地和远程实例都可以被监控。支持 TCP 和 Unix 套接字连接。

建议将插件配置为保持与实例的连接处于打开状态。这样做的好处是，由于连接数量更少，可以减少网络拥塞、延迟以及 CPU 和内存使用。客户端库会负责处理这一点。

> **Note:**
> 未使用连接应保持打开状态的时间段可以通过 `Plugins.<PluginName>.KeepAlive` 参数确定。示例：`Plugins.Memcached.KeepAlive`

Ceph 插件

概述

配置参数可用于自定义 Ceph plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `ceph.conf`），而不是放在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数是必需的，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.Ceph.Default.ApiKey	指定用于连接 Ceph 的默认 API 密钥。
Plugins.Ceph.Default.Mode	指定用于连接 Ceph 的默认模式。
Plugins.Ceph.Default.URI	指定用于连接 Ceph 的默认 URI。
Plugins.Ceph.Default.User	指定用于连接 Ceph 的默认用户名。
Plugins.Ceph.InsecureSkipVerify	指定 HTTP 客户端是否应验证服务器的证书链和主机名。
Plugins.Ceph.KeepAlive	设置未使用的 plugin 连接在关闭前的最长等待时间。
Plugins.Ceph.Sessions.<SessionName>.ApiKey	指定用于连接 Ceph 的命名会话 API 密钥。
Plugins.Ceph.Sessions.<SessionName>.Mode	指定用于连接 Ceph 的命名会话模式。
Plugins.Ceph.Sessions.<SessionName>.URI	指定用于连接 Ceph 的命名会话 URI。
Plugins.Ceph.Sessions.<SessionName>.User	指定用于连接 Ceph 的命名会话用户名。
Plugins.Ceph.Timeout	设置请求执行超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以”#” 开头的注释仅支持位于行首。

参数详情

Plugins.Ceph.Default.ApiKey

指定用于连接 Ceph 的默认 API 密钥。
 当监控项键或命名会话中未指定值时使用。

Plugins.Ceph.Default.Mode

指定连接到 Ceph 的默认模式。
 原生模式仅在 Linux 上受支持，并且从 Ceph 16 开始支持。restful 模式在 Ceph 20 或更新版本中将无法工作。
 请注意，不同模式所需的用户凭据集不同，且彼此不兼容。

默认值：`restful`（已弃用）
 范围：native/restful

Plugins.Ceph.Default.URI

指定用于连接 Ceph 的默认 URI。
 当监控项键或命名会话中未指定值时使用。

 不应包含嵌入式凭据（它们将被忽略）。
 必须符合 URI 格式。
 仅支持 `https` 协议；协议可以省略。
 端口可以省略（默认值 =8003）。

默认值：`https://localhost:8003`

示例：

```
https://127.0.0.1:8003
localhost
```

Plugins.Ceph.Default.User

指定用于连接 Ceph 的默认用户名。
 当监控项键或命名会话中未指定值时使用。

Plugins.Ceph.InsecureSkipVerify

指定 HTTP 客户端是否应验证服务器的证书链和主机名。
 如果为 *true*，TLS 将接受服务器提供的任何证书以及该证书中的任何主机名。在此模式下，TLS 容易受到中间人攻击（仅用于测试）。

默认值：`false`
 范围：false/true

Plugins.Ceph.KeepAlive

设置在关闭未使用的插件连接之前的最大等待时间（以秒为单位）。

默认值：300
 范围：60-900

Plugins.Ceph.Sessions.<SessionName>.ApiKey

指定用于连接 Ceph 的命名会话 API 密钥。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.Ceph.Sessions.<SessionName>.Mode

指定用于连接 Ceph 的命名会话模式。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 native 模式仅在 Linux 上受支持，并且从 Ceph 16 开始支持。restful 模式在 Ceph 20 或更高版本中将无法工作。
 请注意，不同模式使用的用户凭据集不同，彼此之间不兼容。

默认值：`restful`（已弃用）
 范围：native/restful

Plugins.Ceph.Sessions.<SessionName>.URI

指定用于连接 Ceph 的命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 不应包含嵌入式凭据（它们将被忽略）。
 必须符合 URI 格式。
 仅支持 `https` 协议；协议可以省略。
 端口可以省略（默认值 =8003）。

默认值：`https://localhost:8003`

示例：

```
https://127.0.0.1:8003
localhost
```

Plugins.Ceph.Sessions.<SessionName>.User

指定用于连接到 Ceph 的命名会话用户名。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.Ceph.Timeout

设置请求执行超时时间（即在关闭请求之前等待其完成的时长），单位为秒。

默认值：`global timeout`
 范围：1-30

另请参阅

- 配置plugins 的说明
- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)

Docker 插件

概述

配置参数可用于自定义 Docker 插件的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `docker.conf`），而不是放在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看参数的完整详情，请单击其名称）
- 参数详情

Parameter	Description
Plugins.Docker.Endpoint	指定 Docker 守护进程 UNIX 套接字位置。
Plugins.Docker.Timeout	指定请求执行超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不含 BOM 的配置文件；
- 以”#” 开头的注释仅在行首支持。

参数详情

Plugins.Docker.Endpoint

指定 Docker 守护进程的 UNIX 套接字位置。
 必须包含协议（仅支持 `unix://`）。

默认值：`unix:///var/run/docker.sock`

Plugins.Docker.Timeout

指定请求执行超时时间（即在关闭请求之前等待其完成的时长，单位为秒）。

默认值：`global timeout`
 范围：1-30

另请参见

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

Ember+ 插件

概述

配置参数可用于自定义 Ember+ plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `emberplus.conf`），而不是放在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

Ember+ plugin 是一个可加载插件，可在 Ember+ plugin repository 中获取并查看完整说明。

该插件目前只能从源代码构建（适用于 Unix 和 Windows）。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.EmberPlus.Default.URI	指定用于连接 Ember+ 的默认 URI。
Plugins.EmberPlus.KeepAlive	设置在关闭未使用的 plugin 连接之前的最大等待时间。
Plugins.EmberPlus.Sessions.<SessionName>.URI	指定用于连接 Ember+ 的命名会话 URI。
Plugins.EmberPlus.System.Path	指定 Ember+ plugin 可执行文件的路径。
Plugins.EmberPlus.Timeout	设置请求执行超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值。
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码的配置文件，不支持带 BOM 的文件。
- 以”#” 开头的注释仅在行首支持。

参数详情

Plugins.EmberPlus.Default.Uri

指定用于连接 Ember+ 的默认 URI。
 唯一支持的 schema 是 `tcp://`。schema 可以省略。嵌入的凭据将被忽略。

默认值：`tcp://localhost:9998`

Plugins.EmberPlus.KeepAlive

设置在关闭未使用的插件连接之前的最长等待时间（秒）。

默认值：300
 范围：60-900

Plugins.EmberPlus.Sessions.<SessionName>.Uri

指定用于连接 Ember+ 的命名会话 URI。
 唯一支持的 schema 是 `tcp://`。可以省略 schema。嵌入的凭据将被忽略。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

默认值：`tcp://localhost:9998`

Plugins.EmberPlus.System.Path

指定 Ember+ 插件可执行文件的路径。

示例：`Plugins.EmberPlus.System.Path=/usr/libexec/zabbix/zabbix-agent2-plugin-ember-plus`

Plugins.EmberPlus.Timeout

设置等待服务器响应的时长（以秒为单位），适用于首次连接以及会话中的后续操作。

默认值：`global timeout`
 范围：1-30

选项

Parameter	Description
-V --version	打印插件版本和许可信息。
-h --help	打印帮助信息（简写）。
-t, --test <item key>	启动插件进行测试（忽略插件配置）。

另请参见：

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

Memcached 插件

概述

配置参数可用于自定义 Memcached plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `memcached.conf`），而不是写在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数为必填项，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.Memcached.Default.Password	指定用于连接 Memcached 的默认密码。
Plugins.Memcached.Default.Uri	指定用于连接 Memcached 的默认 URI。
Plugins.Memcached.Default.User	指定用于连接 Memcached 的默认用户名。
Plugins.Memcached.KeepAlive	设置关闭未使用的 plugin 连接之前的最大等待时间。
Plugins.Memcached.Sessions.<SessionName>.Password	指定用于连接 Memcached 的命名会话密码。
Plugins.Memcached.Sessions.<SessionName>.Uri	指定用于连接 Memcached 的命名会话 URI。
Plugins.Memcached.Sessions.<SessionName>.User	指定用于连接 Memcached 的命名会话用户名。
Plugins.Memcached.Timeout	设置请求执行超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以”#” 开头的注释仅在行首支持。

参数详情

Plugins.Memcached.Default.Password

指定用于连接 Memcached 的默认密码。
 当监控项键或命名会话中未指定值时使用。

Plugins.Memcached.Default.Uri

指定用于连接 Memcached 的默认 URI。
 当监控项键值或命名会话中未指定值时使用。
 嵌入的凭据将被忽略。
 必须符合 URI 格式。支持的 scheme：`tcp`、`unix`。
 可以省略 scheme。
 可以省略端口（默认值 =11211）。

默认值：`tcp://localhost:11211`

示例：

```
tcp://localhost:11211
localhost
unix:/var/run/memcached.sock
```

Plugins.Memcached.Default.User

指定用于连接 Memcached 的默认用户名。
 当监控项键值或命名会话中未指定值时使用。

Plugins.Memcached.KeepAlive

设置在关闭未使用的插件连接之前的最长等待时间（秒）。

默认值：300
 范围：60-900

Plugins.Memcached.Sessions.<SessionName>.Password

指定用于连接 Memcached 的命名会话密码。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.Memcached.Sessions.<SessionName>.Uri

指定用于连接 Memcached 的命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 嵌入式凭据将被忽略。
 必须符合 URI 格式。支持的方案：`tcp`、`unix`。
 可以省略方案。
 可以省略端口（默认值 =11211）。

示例：

```
tcp://localhost:11211
localhost
unix:/var/run/memcached.sock
```

Plugins.Memcached.Sessions.<SessionName>.User

指定用于连接到 Memcached 的命名会话用户名。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.Memcached.Timeout

设置请求执行超时时间（即在关闭请求之前等待其完成的时长，单位为秒）。

默认值：`global timeout`
 范围：1-30

另请参阅：

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

Modbus 插件

概述

配置参数可用于自定义 Modbus 插件的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `modbus.conf`），而不是写在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.Modbus.Sessions.<SessionName>.Endpoint	指定用于连接 Modbus 的命名会话端点。
Plugins.Modbus.Sessions.<SessionName>.SlaveID	指定用于连接 Modbus 的命名会话从站 ID。
Plugins.Modbus.Sessions.<SessionName>.Timeout	以秒为单位设置命名会话超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以”#” 开头的注释仅支持位于行首。

参数详情

Plugins.Modbus.Sessions.<SessionName>.Endpoint

指定用于连接 Modbus 的命名会话端点。
 连接字符串由协议方案、主机地址以及端口或串口名称和属性组成。
 <SessionName> - 定义一个会话名称，用于在监控项键中使用。

Plugins.Modbus.Sessions.<SessionName>.SlaveID

指定用于连接 Modbus 的命名会话从站 ID。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

示例：`Plugins.Modbus.Sessions.MB1.SlaveID=20`

注意，仅当监控项键值 中提供的 slave ID 参数值为空时，才会检查此命名会话参数。

Plugins.Modbus.Sessions.<SessionName>.Timeout

以秒为单位设置指定会话的超时时间。
 <SessionName> - 定义用于监控项键值中的会话名称。

示例：Plugins.Modbus.Sessions.MB1.Timeout=2

如果需要设置请求执行超时（即在关闭请求之前等待请求完成的时间），请使用监控项配置 表单。

另请参见

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

MongoDB 插件

概述

配置参数可用于自定义 MongoDB plugin 的行为。

建议将这些参数单独放在各自的配置文件中（例如 mongodb.conf），而不是放在 agent 2 配置文件中。然后使用Include 指令将该文件包含到 Zabbix agent 2 配置中。

MongoDB plugin 是一个可加载的 plugin，可在 MongoDB plugin repository 中获取并查看完整说明。

除非明确说明该参数为必填，否则所有参数均为可选。

连接 MongoDB 时，plugin 会按特定顺序检查其配置，以确定使用哪个值。从高到低的优先级顺序如下：

1. 连接 URI 中指定的参数（例如，在 Plugins.MongoDB.Sessions.<session_name>.Uri 中 ? 后定义的查询参数优先级最高）。
2. Zabbix agent 2 监控项键中的参数（例如，mongodb.collection.stats）。
3. 在命名会话中定义的参数(例如 Plugins.MongoDB.Sessions.<SessionName>.User、Plugins.MongoDB.Sessions.<SessionNa
4. 默认值（例如 Plugins.MongoDB.Default.User、Plugins.MongoDB.Default.Password）。

例如：

- 如果您在 *.<SessionName>.Uri 参数中指定了用于对 MongoDB 服务器连接进行身份验证的数据库(例如 Plugins.MongoDB.Sessions 那么即使 *.<SessionName>.AuthSource 参数设置为其他值，plugin 也会将 *.<SessionName>.AuthSource 参数视为已设置为 admin。
- 如果您在 *.<SessionName>.Uri 参数中指定了 URI(例如 Plugins.MongoDB.Sessions.<session_name>.Uri=mongodb://127. 但在 Zabbix agent 2 监控项键中提供了用户名和密码，那么即使已配置 Plugins.MongoDB.Sessions.<SessionName>.User 和 Plugins.MongoDB.Sessions.<SessionName>.Password，plugin 仍会使用监控项键中的用户名和密码。

Attention:
如果使用 mongodb+srv:// 方案，URI 可能会从 DNS TXT 记录中获取参数。从该 TXT 记录获取的参数会覆盖所有其他配置，包括 URI 参数、Zabbix Agent 2 监控项键参数以及会话参数。
 用于连接字符串的主机名只能存在一个 TXT 记录。

本页结构如下：

- 参数概览（要查看参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.MongoDB.Default.AuthMechanism	指定用于连接 MongoDB 服务器的默认认证机制。
Plugins.MongoDB.Default.AuthSource	指定用于对 MongoDB 服务器连接进行身份验证的默认数据库。
Plugins.MongoDB.Default.Disable	禁用 MongoDB 副本集节点的自动发现。
Plugins.MongoDB.Default.Password	指定默认密码。
Plugins.MongoDB.Default.ReplicaSet	指定默认副本集名称，用于 MongoDB 副本集节点的自动发现。
Plugins.MongoDB.Default.ReadPreference	指定连接 MongoDB 副本集节点或回退时的默认顺序。
Plugins.MongoDB.Default.TLSConnect	指定 Zabbix agent 2 与被监控数据库之间通信所需的加密级别。
Plugins.MongoDB.Default.TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.MongoDB.Default.TLSCertFile	指定包含 agent 证书或证书链的文件完整路径。
Plugins.MongoDB.Default.TLSKeyFile	指定包含数据库私钥的文件完整路径。
Plugins.MongoDB.Default.Uri	指定默认 URI。
Plugins.MongoDB.Default.User	指定默认用户名。
Plugins.MongoDB.KeepAlive	设置在关闭未使用的 plugin 连接之前的最长等待时间。
Plugins.MongoDB.Sessions.<SessionName>.AuthMechanism	指定用于连接 MongoDB 服务器的命名会话认证机制。
Plugins.MongoDB.Sessions.<SessionName>.AuthSource	指定用于对 MongoDB 服务器连接进行身份验证的命名会话数据库。

Parameter	Description
Plugins.MongoDB.Sessions.<SessionName>.Direct	禁用命名会话的 MongoDB 副本集节点自动发现。
Plugins.MongoDB.Sessions.<SessionName>.Password	指定命名会话密码。
Plugins.MongoDB.Sessions.<SessionName>.ReplicaSet	指定命名会话副本集名称，用于 MongoDB 副本集节点的自动发现。
Plugins.MongoDB.Sessions.<SessionName>.ReadPreference	指定命名会话连接 MongoDB 副本集节点或回退时的顺序。
Plugins.MongoDB.Sessions.<SessionName>.TLSConnect	指定命名会话中 Zabbix agent 2 与被监控数据库之间通信所需的加密级别。
Plugins.MongoDB.Sessions.<SessionName>.TLSCAFile	指定命名会话中包含用于对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile	指定命名会话中包含 agent 证书或证书链的文件完整路径。
Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile	指定命名会话中包含数据库私钥的文件完整路径。
Plugins.MongoDB.Sessions.<SessionName>.Uri	指定命名会话 URI。
Plugins.MongoDB.Sessions.<SessionName>.User	指定命名会话用户名。
Plugins.MongoDB.System.Path	指定 plugin 可执行文件的路径。
Plugins.MongoDB.Timeout	设置请求执行超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件。
- 以 "\#" 开头的注释仅在行首受支持。

参数详情

Plugins.MongoDB.Default.AuthMechanism

指定用于连接 MongoDB 服务器的默认身份验证机制。
 <SessionName> - 定义用于监控项键中的会话名称。

范围：SCRAM-SHA-1、SCRAM-SHA-256 或 MONGODB-X509

Plugins.MongoDB.Default.AuthSource

指定用于连接 MongoDB 服务器进行身份验证的默认数据库。
 <SessionName> - 定义用于监控项键中的会话名称。

默认值：admin

如果使用 MONGODB-X509 身份验证机制，则此参数会自动设置为 $external；将其修改为任何其他值都会在启动时导致错误。

Plugins.MongoDB.Default.Direct

禁用 MongoDB 副本集节点的自动发现。
 <SessionName> - 定义用于监控项键中的会话名称。

默认值: false
 范围: true - 仅连接到指定的服务器，忽略副本集中的其他成员，false - 自动发现副本集中的所有节点。

如果使用 tcp://（已弃用）方案，则该参数默认设置为 true。

Plugins.MongoDB.Default.Password

指定默认密码。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MongoDB.Default.ReplicaSet

指定默认副本集名称，用于自动发现 MongoDB 副本集节点。
 <SessionName> - 定义用于监控项键中的会话名称。
 如果此参数与 mongodb+srv:// 方案一起使用，其行为将与提供了多个 URL 时相同；DNS 服务器可能包含一个 TXT 记录，可覆盖此参数（或任何其他参数）。
 当 Plugins.MongoDB.Sessions.<SessionName>.Direct 参数设置为 true 时不可用；agent 将无法启动。

Plugins.MongoDB.Default.ReadPreference

指定连接 MongoDB 副本集节点或回退时的默认顺序。
 <SessionName> - 定义用于监控项键的会话名称。
 如果 Plugins.MongoDB.Sessions.<SessionName>.Direct 参数设置为 true，则不可用；agent 将无法启动。

取值范围：primary、primaryPreferred、secondary、secondaryPreferred、nearest、PrimaryMode、PrimaryPreferredMode、SecondaryMode、SecondaryPreferredMode 或 NearestMode

推荐使用 primary、primaryPreferred、secondary、secondaryPreferred 和 nearest。以 Mode 结尾的值作为旧版兼容别名受到支持。

Plugins.MongoDB.Default.TLSConnect

指定 Zabbix agent 2 与被监控数据库之间通信所需的加密级别。
 <SessionName> - 定义用于监控项键中的会话名称。
 自插件版本 1.2.1 起支持。

范围：required - 要求 TLS 连接，verify_ca - 验证证书，verify_full - 验证证书和 IP 地址

Plugins.MongoDB.Default.TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件完整路径名，用于 Zabbix agent 2 与被监控数据库之间的加密通信。
 <Session-Name> - 定义用于监控项键中的会话名称。

必填：yes，如果 Plugins.MongoDB.Sessions.<SessionName>.TLSConnect 设置为 verify_ca 或 verify_full

Plugins.MongoDB.Default.TLSCertFile

指定包含 agent 证书或证书链的文件完整路径，用于 Zabbix agent 2 与受监控数据库之间的加密通信。
 <SessionName> - 定义用于监控项键中的会话名称。

必填：yes，如果指定了 Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile

Plugins.MongoDB.Default.TLSKeyFile

指定包含数据库私钥的文件的完整路径名，用于 Zabbix agent 2 与被监控数据库之间的加密通信。
 <SessionName> - 定义用于监控项键中的会话名称。

必填：yes，如果已指定 Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile

Plugins.MongoDB.Default.Uri

指定默认 URI。
 <SessionName> - 定义用于监控项键值中的会话名称。
 必须符合 URI 格式。
 支持的方案：mongodb:// 和 mongodb+srv://；tcp:// 方案已弃用，仅为兼容现有配置而保留。
 可以省略端口（默认值 =27017）。
 示例：“

默认值：tcp://127.0.0.1:27017

示例：

```
mongodb://127.0.0.1:27017
mongodb+srv://example.com
localhost
```

Plugins.MongoDB.Default.User

指定默认用户名。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MongoDB.KeepAlive

设置在关闭未使用的插件连接之前的最长等待时间（秒）。

默认值：300
 范围：60-900

Plugins.MongoDB.Sessions.<SessionName>.AuthMechanism

指定用于连接到 MongoDB 服务器的命名会话认证机制。
 <SessionName> - 定义用于监控项键中的会话名称。

范围：SCRAM-SHA-1、SCRAM-SHA-256 或 MONGODB-X509

Plugins.MongoDB.Sessions.<SessionName>.AuthSource

指定用于认证连接到 MongoDB 服务器的命名会话数据库。
 <SessionName> - 定义用于监控项键中的会话名称。

默认值：admin

如果使用 MONGODB-X509 认证机制，此参数会自动设置为 $external；将其修改为其他任何值都会在启动时导致错误。

Plugins.MongoDB.Sessions.<SessionName>.Direct

禁用对命名会话的 MongoDB 副本集节点自动发现。
 <SessionName> - 定义一个会话名称，用于监控项键中。

默认值: false
 范围: true - 仅连接到指定的服务器，忽略副本集中的其他成员，false - 自动发现副本集中的所有节点。

如果使用 tcp://（已弃用）方案，则该参数默认设置为 true。

Plugins.MongoDB.Sessions.<SessionName>.Password

指定命名会话的密码。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MongoDB.Sessions.<SessionName>.ReplicaSet

指定命名会话的副本集名称，用于自动发现 MongoDB 副本集节点。
 <SessionName> - 定义会话名称，用于监控项键值中。
 如果此参数与 mongodb+srv:// 方案一起使用，其行为将与提供了多个 URL 时相同；DNS 服务器可能包含一个 TXT 记录，可覆盖此（或任何其他）参数。
 当 Plugins.MongoDB.Sessions.<SessionName>.Direct 参数设置为 true 时不可用；agent 将无法启动。

Plugins.MongoDB.Sessions.<SessionName>.ReadPreference

指定连接 MongoDB 副本集节点或回退时使用的命名会话顺序。
 <SessionName> - 定义会话名称，以便在监控项键中使用。
 如果 Plugins.MongoDB.Sessions.<SessionName>.Direct 参数设置为 true，则此参数不可用；agent 将无法启动。

取值范围：primary、primaryPreferred、secondary、secondaryPreferred、nearest、PrimaryMode、PrimaryPreferredMode、SecondaryMode、SecondaryPreferredMode 或 NearestMode

推荐使用 primary、primaryPreferred、secondary、secondaryPreferred 和 nearest。以 Mode 结尾的值作为旧版兼容别名受支持。

Plugins.MongoDB.Sessions.<SessionName>.TLSConnect

指定命名会话中 Zabbix agent 2 与被监控数据库之间通信所需的加密级别。
 <SessionName> - 定义用于监控项键中的会话名称。
 自插件版本 1.2.1 起支持。

范围：required - 要求 TLS 连接，verify_ca - 验证证书，verify_full - 验证证书和 IP 地址

Plugins.MongoDB.Sessions.<SessionName>.TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件完整路径，用于 Zabbix agent 2 与所监控数据库之间针对指定会话的加密通信。
 <SessionName> - 定义会话名称，用于监控项键值中。

必填：yes，如果 Plugins.MongoDB.Sessions.<SessionName>.TLSConnect 设置为 verify_ca 或 verify_full

Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile

指定包含 agent 证书或证书链的文件完整路径，用于 Zabbix agent 2 与所监控数据库之间的加密通信，适用于指定会话。
 <SessionName> - 定义会话名称，用于监控项键值中。

必填：yes，如果已指定 Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile

Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile

指定包含数据库私钥的文件的完整路径名，用于 Zabbix agent 2 与命名会话所监控的数据库之间的加密通信。
 <SessionName> - 定义用于监控项键中的会话名称。

必填：yes，如果已指定 Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile

Plugins.MongoDB.Sessions.<SessionName>.Uri

指定命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 必须与 URI 格式匹配。
 支持的方案：mongodb:// 和 mongodb+srv://；tcp:// 方案已弃用，仅为与现有配置保持向后兼容而保留。
 可以省略端口（默认值 =27017）。
 示例：“

默认值：tcp://127.0.0.1:27017

示例：

```
mongodb://127.0.0.1:27017
mongodb+srv://example.com
localhost
```

Plugins.MongoDB.Sessions.<SessionName>.User

指定命名会话的用户名。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MongoDB.System.Path

指定 MongoDB 插件可执行文件的路径。

示例用法：Plugins.MongoDB.System.Path=/usr/libexec/zabbix/zabbix-agent2-plugin-mongodb

Plugins.MongoDB.Timeout

设置请求执行超时（即在关闭请求之前等待其完成的时长，单位为秒）。

默认值：global timeout
 范围：1-30

选项

Parameter	Description
-V --version	打印插件版本和许可证信息。
-h --help	打印帮助信息（简写）。
-t, --test <item key>	启动插件进行测试（忽略插件配置）。

另请参阅

- 常规 Zabbix agent 2 配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

MQTT 插件

概述

配置参数可用于自定义 MQTT plugin 的行为。

建议将这些参数单独放在自己的配置文件中（例如 `mqtt.conf`），而不是放在 agent 2 配置文件中。然后使用Include 指令将该文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.MQTT.Default.Password	指定用于连接 MQTT 的默认密码。
Plugins.MQTT.Default.TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.MQTT.Default.TLSCertFile	指定包含 agent 证书或证书链的文件完整路径。
Plugins.MQTT.Default.TLSKeyFile	指定包含 MQTT 私钥的文件完整路径。
Plugins.MQTT.Default.Topic	指定 MQTT 订阅的默认主题。
Plugins.MQTT.Default.Url	指定默认的 MQTT broker 连接字符串。
Plugins.MQTT.Default.User	指定用于连接 MQTT 的默认用户名。
Plugins.MQTT.Sessions.<SessionName>.Password	指定用于连接 MQTT 的命名会话密码。
Plugins.MQTT.Sessions.<SessionName>.TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件完整路径，适用于命名会话。
Plugins.MQTT.Sessions.<SessionName>.TLSCertFile	指定包含 agent 证书或证书链的文件完整路径，适用于命名会话。
Plugins.MQTT.Sessions.<SessionName>.TLSKeyFile	指定包含 MQTT 私钥的文件完整路径，适用于命名会话。
Plugins.MQTT.Sessions.<SessionName>.Topic	指定用于 MQTT 订阅的命名会话主题。
Plugins.MQTT.Sessions.<SessionName>.Url	指定命名会话的 MQTT broker 连接字符串。
Plugins.MQTT.Sessions.<SessionName>.User	指定用于连接 MQTT 的命名会话用户名。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以”#” 开头的注释仅在行首支持。

参数详情

Plugins.MQTT.Default.Password

指定用于连接 MQTT 的默认密码。
 当监控项键值或命名会话中未指定值时使用。

Plugins.MQTT.Default.TLSCAFile

指定包含用于 Zabbix agent 2 与 MQTT broker 之间加密通信的对等证书验证所需顶级 CA 证书的文件完整路径名。
 在命名会话中未指定值时使用。

Plugins.MQTT.Default.TLSCertFile

指定包含 agent 证书或证书链的文件完整路径，用于 Zabbix agent 2 与 MQTT broker 之间的加密通信。
 在命名会话中未指定值时使用。

Plugins.MQTT.Default.TLSKeyFile

指定包含 MQTT 私钥文件的完整路径名，用于 Zabbix agent 2 与 MQTT broker 之间的加密通信。
 在命名会话中未指定值时使用。

Plugins.MQTT.Default.Topic

指定 MQTT 订阅的默认主题。
 当监控项键值或命名会话中未指定值时使用。
 主题可以包含通配符（“+”、“#”）。

示例：

```
path/to/file
path/to/#
path/+/topic
```

Plugins.MQTT.Default.Url

指定默认的 MQTT broker 连接字符串。
 当监控项键值或命名会话中未指定任何值时使用。
 不应包含查询参数。
 必须符合 URL 格式。
 支持的 scheme：tcp（默认）、ws、tls
 可以省略 scheme。
 可以省略端口（默认 =1883）

默认值：tcp://localhost:1883

示例：

```
tcp://host:1883
localhost
ws://host:8080
```

Plugins.MQTT.Default.User

指定用于连接 MQTT 的默认用户名。
 当监控项键或命名会话中未指定值时使用。

Plugins.MQTT.Sessions.<SessionName>.Password

指定用于连接到 MQTT 的命名会话密码。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.MQTT.Sessions.<SessionName>.TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件完整路径名，适用于指定会话，用于 Zabbix agent 2 与 MQTT broker 之间的加密通信。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.MQTT.Sessions.<SessionName>.TLSCertFile

指定包含 agent 证书或证书链的文件完整路径，用于指定会话，在 Zabbix agent 2 与 MQTT broker 之间的加密通信中使用。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

Plugins.MQTT.Sessions.<SessionName>.TLSKeyFile

指定包含 MQTT 私钥的文件的完整路径名，用于命名会话，供 Zabbix agent 2 与 MQTT broker 之间的加密通信使用。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.MQTT.Sessions.<SessionName>.Topic

指定用于 MQTT 订阅的命名会话主题。
 <SessionName> - 定义会话名称，用于监控项键值中。
 该主题可以包含通配符（”+”、”#”）。

示例：

```
path/to/file
path/to/#
path/+/topic
```

Plugins.MQTT.Sessions.<SessionName>.Url

指定命名会话的 MQTT broker 连接字符串。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 不应包含查询参数。
 必须符合 URL 格式。
 支持的协议：tcp（默认）、ws、tls
 可以省略协议。
 可以省略端口（默认 =1883）

默认：tcp://localhost:1883

示例：

```
tcp://host:1883
localhost
ws://host:8080
```

Plugins.MQTT.Sessions.<SessionName>.User

指定用于连接到 MQTT 的命名会话用户名。
 <SessionName> - 定义用于监控项键中的会话名称。

如果需要设置请求执行超时（即在关闭请求之前等待其完成的时间），请使用监控项配置 表单。

另请参见

- 常规 Zabbix agent 2 配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

MSSQL 插件

概述

配置参数可用于自定义 MSSQL plugin 的行为。

建议将这些参数指定在单独的配置文件中（例如 mssql.conf），而不是写入 agent 2 配置文件。然后使用Include 指令将该文件包含到 Zabbix agent 2 配置中。

MSSQL plugin 是一个可加载插件，可在 MSSQL plugin repository 中获取并查看完整说明。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.MSSQL.CustomQueriesDir	指定包含用户自定义 .sql 文件的目录路径，这些文件中包含 plugin 可执行的自定义查询。
Plugins.MSSQL.CustomQueriesEnabled	启用 mssql.custom.query 监控项键的执行。
Plugins.MSSQL.Default.CACertPath	指定签发 MSSQL 服务器证书的证书颁发机构（CA）公钥证书的默认文件路径。
Plugins.MSSQL.Default.Database	指定默认数据库名称。
Plugins.MSSQL.Default.Encrypt	指定默认加密级别。
Plugins.MSSQL.Default.HostnameInCertificate	默认指定 MSSQL 服务器证书的通用名称（CN）。
Plugins.MSSQL.Default.Password	指定默认密码。
Plugins.MSSQL.Default.TLSMinVersion	默认指定最低 TLS 版本。
Plugins.MSSQL.Default.TrustServerCertificate	默认信任服务器证书而不进行验证。
Plugins.MSSQL.Default.Uri	指定默认 URI。
Plugins.MSSQL.Default.User	指定默认用户名。
Plugins.MSSQL.KeepAlive	指定在关闭未使用的 plugin 连接之前的最长等待时间（秒）。
Plugins.MSSQL.Sessions.<SessionName>.CACertPath	指定命名会话文件路径。该路径指向签发 MSSQL 服务器证书的证书颁发机构（CA）公钥证书。
Plugins.MSSQL.Sessions.<SessionName>.Database	指定命名会话数据库名称。
Plugins.MSSQL.Sessions.<SessionName>.Encrypt	指定命名会话的加密级别。
Plugins.MSSQL.Sessions.<SessionName>.HostnameInCertificate	指定命名会话的 MSSQL 服务器证书通用名称（CN）。
Plugins.MSSQL.Sessions.<SessionName>.Password	指定命名会话密码。
Plugins.MSSQL.Sessions.<SessionName>.TLSMinVersion	指定命名会话的最低 TLS 版本。
Plugins.MSSQL.Sessions.<SessionName>.TrustServerCertificate	信任命名会话的服务器证书而不进行验证。
Plugins.MSSQL.Sessions.<SessionName>.Uri	指定命名会话 URI。
Plugins.MSSQL.Sessions.<SessionName>.User	指定命名会话用户名。
Plugins.MSSQL.System.Path	指定 MSSQL plugin 可执行文件的路径。
Plugins.MSSQL.Timeout	设置首次连接以及会话后续操作时等待服务器响应的时长（秒）。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值。
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件。
- 以 "\#" 开头的注释仅在行首支持。

参数详情

Plugins.MSSQL.CustomQueriesDir

指定包含用户定义的 .sql 文件的目录路径，这些文件包含插件可以执行的自定义查询。
 插件会在启动时加载配置目录中所有可用的.sql 文件。这意味着，对自定义查询文件所做的任何更改都不会生效，直到插件重新启动。插件会与 Zabbix agent 2 一起启动和停止。

默认值：
/usr/local/share/zabbix/custom-queries/mssql - 适用于 Unix 系统，
*:\Program Files\Zabbix Agent 2\Custom Queries\MSSQL - 适用于 Windows 系统（其中 * 是来自 ProgramFiles 环境变量的驱动器名称）

Plugins.MSSQL.CustomQueriesEnabled

启用 mssql.custom.query 监控项键的执行。
 如果禁用，则不会从自定义查询目录路径加载任何查询。

默认值：false
 范围：true/false

Plugins.MSSQL.Default.CACertPath

指定颁发 MSSQL 服务器证书的证书颁发机构（CA）公钥证书的默认文件路径。
 证书必须采用 PEM 格式。

Plugins.MSSQL.Default.Database

指定默认数据库名称。

Plugins.MSSQL.Default.Encrypt

指定默认加密级别。

范围：
true - 插件与服务器之间发送的数据已加密；
false - 插件与服务器之间发送的数据除登录数据包外不加密；
strict - 插件与服务器之间发送的数据使用 TDS8 进行端到端加密；
disable - 插件与服务器之间发送的数据不加密。

Plugins.MSSQL.Default.HostNameInCertificate

默认指定 MSSQL 服务器证书的通用名称（CN）。

Plugins.MSSQL.Default.Password

指定受保护的 MSSQL 服务器的默认密码。

Plugins.MSSQL.Default.TLSMinVersion

默认指定最小 TLS 版本。

范围：`1.0`、`1.1`、`1.2`、`1.3`

Plugins.MSSQL.Default.TrustServerCertificate

默认情况下信任服务器证书，而不对其进行验证。

范围：`true`、`false`

Plugins.MSSQL.Default.Uri

指定用于连接 MSSQL 服务器的默认 URI。
 嵌入的凭据将被忽略。
 唯一支持的 schema 是 `sqlserver://`。
 可以省略该 schema。

默认值：`sqlserver://localhost:1433`

Plugins.MSSQL.Default.User

指定受保护的 MSSQL 服务器的默认用户名。

Plugins.MSSQL.KeepAlive

指定在关闭未使用的插件连接之前的最长等待时间（以秒为单位）。

默认值：300
 范围：60–900

Plugins.MSSQL.Sessions.<SessionName>.CACertPath

指定命名会话文件路径，指向签发 MSSQL 服务器证书的证书颁发机构（CA）的公钥证书。
 <SessionName> - 定义用于监控项键中的会话名称。
 证书必须采用 PEM 格式。

Plugins.MSSQL.Sessions.<SessionName>.Database

指定命名会话的数据库名称。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MSSQL.Sessions.<SessionName>.Encrypt

指定命名会话加密级别。
 <SessionName> - 定义用于监控项键值中的会话名称。

范围：
true - 插件与服务器之间发送的数据已加密；
false - 插件与服务器之间发送的数据除登录数据包外不加密；
strict - 插件与服务器之间发送的数据使用 TDS8 进行端到端加密；
disable - 插件与服务器之间发送的数据不加密。

Plugins.MSSQL.Sessions.<SessionName>.HostNameInCertificate

指定命名会话所使用的 MSSQL 服务器证书的通用名称（CN）。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MSSQL.Sessions.<SessionName>.Password

为受保护的 MSSQL 服务器指定命名会话密码。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MSSQL.Sessions.<SessionName>.TLSMinVersion

指定命名会话的最低 TLS 版本。
 <SessionName> - 定义用于监控项键中的会话名称。

范围：`1.0`、`1.1`、`1.2`、`1.3`

Plugins.MSSQL.Sessions.<SessionName>.TrustServerCertificate

信任指定会话的服务器证书，而不进行验证。
 <SessionName> - 定义用于监控项键中的会话名称。

范围：`true`、`false`

Plugins.MSSQL.Sessions.<SessionName>.Uri

指定用于连接 MSSQL 服务器的命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 嵌入的凭据将被忽略。
 唯一支持的 schema 是 `sqlserver://`。
 可以省略该 schema。

默认值: `sqlserver://localhost:1433`

Plugins.MSSQL.Sessions.<SessionName>.User

为受保护的 MSSQL 服务器指定命名会话的用户名。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.MSSQL.System.Path

指定 MSSQL 插件可执行文件的路径。
 MSSQL 插件的全局设置。适用于所有连接。

示例：`Plugins.MSSQL.System.Path=/usr/libexec/zabbix/zabbix-agent2-plugin-mssql`

Plugins.MSSQL.Timeout

设置等待服务器响应的时长（以秒为单位），适用于首次连接以及会话中的后续操作。

默认值：`global timeout`
 范围：1-30

选项

Parameter	Description
-V --version	打印插件版本和许可信息。
-h --help	打印帮助信息（简写）。
-t, --test <item key>	启动插件进行测试（忽略插件配置）。

另请参阅

- 常规 Zabbix agent 2 配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置plugins 的说明

MySQL 插件

概述

配置参数可用于自定义 MySQL plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `mysql.conf`），而不是写入 agent 2 配置文件。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数是必需的，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.Mysql.CallTimeout	指定等待请求完成的最长时间（秒）。
Plugins.Mysql.CustomQueriesPath	指定包含自定义查询 `.sql` 文件的目录完整路径。
Plugins.Mysql.CustomQueriesEnabled	启用 `mysql.custom.query` 监控项键的执行。
Plugins.Mysql.Default.Password	指定默认密码。
Plugins.Mysql.Default.TLSCipher	指定 Zabbix agent 2 与被监控数据库之间通信的默认加密级别。
Plugins.Mysql.Default.TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.Mysql.Default.TLSCertFile	指定包含 agent 证书或证书链的文件完整路径。
Plugins.Mysql.Default.TLSKeyFile	指定包含数据库私钥的文件完整路径。
Plugins.Mysql.Default.Uri	指定默认 URI。
Plugins.Mysql.Default.User	指定默认用户名。
Plugins.Mysql.KeepAlive	指定在关闭未使用的 plugin 连接之前的最长等待时间（秒）。
Plugins.Mysql.Sessions.<SessionName>.Password	指定命名会话密码。
Plugins.Mysql.Sessions.<SessionName>.TLSCipher	指定命名会话中 Zabbix agent 2 与被监控数据库之间通信的加密级别。
Plugins.Mysql.Sessions.<SessionName>.TLSCAFile	指定包含用于命名会话对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.Mysql.Sessions.<SessionName>.TLSCertFile	指定包含命名会话 agent 证书或证书链的文件完整路径。
Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile	指定包含命名会话数据库私钥的文件完整路径。
Plugins.Mysql.Sessions.<SessionName>.Uri	指定命名会话 URI。
Plugins.Mysql.Sessions.<SessionName>.User	指定命名会话用户名。
Plugins.Mysql.Timeout	设置建立连接时的最长等待时间（秒）。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以"#" 开头的注释仅支持在行首。

参数详情

Plugins.Mysql.CallTimeout

指定等待请求完成的最长时间（秒）。

默认值：`global timeout`
 范围：1-30

Plugins.Mysql.CustomQueriesPath

指定包含自定义查询 `.sql` 文件的目录的完整路径名。

默认值：
`/usr/local/share/zabbix/custom-queries/mysql` - 适用于 Unix 系统，
`*:\Program Files\Zabbix Agent 2\Custom Queries\Mysql` - 适用于 Windows 系统（其中 `*` 是来自 `ProgramFiles` 环境变量的驱动器名称）

Plugins.Mysql.CustomQueriesEnabled

启用 `mysql.custom.query` 监控项键的执行。
 如果禁用，则不会从自定义查询目录路径加载任何查询。

默认值：`false`
 范围：`true/false`

Plugins.Mysql.Default.Password

指定受保护的 MySQL 服务器的默认密码。

Plugins.Mysql.Default.TLSConnect

指定 Zabbix agent 2 与受监控数据库之间通信的默认加密级别。

范围：
required - 要求 TLS 连接；
verify_ca - 验证证书；
verify_full - 验证证书和 IP 地址

Plugins.Mysql.Default.TLSCAFile

指定包含用于对等证书验证的顶级 CA 证书的文件完整路径名，用于 Zabbix agent 2 与受监控数据库之间的加密通信。

必填：`yes`，如果 `Plugins.Mysql.Default.TLSConnect` 设置为 `verify_ca` 或 `verify_full`

Plugins.Mysql.Default.TLSCertFile

指定包含 agent 证书或证书链的文件完整路径，用于 Zabbix agent 2 与受监控数据库之间的加密通信。

必填：`yes`，如果已指定 `Plugins.Mysql.Default.TLSKeyFile`

Plugins.Mysql.Default.TLSKeyFile

指定包含数据库私钥的文件的完整路径名，用于 Zabbix agent 2 与被监控数据库之间的加密通信。

必填：如果指定了 `Plugins.Mysql.Default.TLSCertFile`，则为 `yes`

Plugins.Mysql.Default.Uri

指定用于连接 MySQL 服务器的默认 URI。
 嵌入的凭据将被忽略。
 支持的方案：`tcp`、`unix`
 可以省略 schema。
 可以省略端口（默认值 =3306）。

示例：

```
tcp://localhost:3306
localhost
unix:/var/run/mysql.sock
```

Plugins.Mysql.Default.User

指定受保护的 MySQL 服务器的默认用户名。

Plugins.Mysql.KeepAlive

指定在关闭未使用的插件连接之前的最长等待时间（秒）。

默认值：300
 范围：60-900

Plugins.Mysql.Sessions.<SessionName>.Password

为受保护的 MySQL 服务器指定命名会话密码。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.Mysql.Sessions.<SessionName>.TLSConnect

指定 Zabbix agent 2 与被监控数据库之间通信所使用的命名会话加密级别。
 <SessionName> - 定义用于监控项键中的会话名称。

范围：
required - 要求 TLS 连接；
verify_ca - 验证证书；
verify_full - 验证证书和 IP 地址

Plugins.Mysql.Sessions.<SessionName>.TLSCAFile

指定包含顶级 CA 证书的文件完整路径名，用于命名会话的对等证书验证，适用于 Zabbix agent 2 与被监控数据库之间的加密通信。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

必填：`yes`，如果 `Plugins.Mysql.Sessions.<SessionName>.TLSConnect` 设置为 `verify_ca` 或 `verify_full`

Plugins.Mysql.Sessions.<SessionName>.TLSCertFile

指定包含命名会话 agent 证书或证书链的文件完整路径，用于 Zabbix agent 2 与受监控数据库之间的加密通信。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

必需：`yes`，如果指定了 `Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile`

Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile

指定包含命名会话数据库私钥的文件完整路径，用于 Zabbix agent 2 与受监控数据库之间的加密通信。
 <SessionName> - 定义一个会话名称，用于监控项键中。

必填：`yes`，如果已指定 `Plugins.Mysql.Sessions.<SessionName>.TLSCertFile`

Plugins.Mysql.Sessions.<SessionName>.Uri

指定用于连接 MySQL 服务器的命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 嵌入的凭据将被忽略。
 支持的 scheme：`tcp`、`unix`
 可以省略 schema。
 可以省略端口（默认值 =3306）。

示例：

```
tcp://localhost:3306
localhost
unix:/var/run/mysql.sock
```

Plugins.Mysql.Sessions.<SessionName>.User

为受保护的 MySQL 服务器指定命名会话的用户名。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.Mysql.Timeout

设置建立连接时的最长等待时间（秒）。

默认值：`global timeout`
 范围：1-30

另请参阅

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

NVIDIA GPU 插件

概述

配置参数可用于自定义 NVIDIA GPU plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `nvidiagpu.conf`），而不是放在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

NVIDIA GPU plugin 是一个可加载插件，可在 NVIDIA GPU plugin repository 中获取并查看完整说明。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.NVIDIA.System.Path	指定 NVIDIA GPU plugin 可执行文件的路径。
Plugins.NVIDIA.Timeout	指定在连接尝试以及会话后续操作期间，等待服务器响应的最长时间（秒）。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以”#” 开头的注释仅支持出现在行首。

参数详情

Plugins.NVIDIA.System.Path

指定 NVIDIA GPU 插件可执行文件的路径。

示例：

```
Plugins.NVIDIA.System.Path=/path/to/executable/nvidia
```

Plugins.NVIDIA.Timeout

指定在连接尝试以及会话中的后续操作期间，等待服务器响应的最长时间（以秒为单位）。
 如果全局监控项类型超时或单个监控项超时更大，则会覆盖此值。

默认值：`global timeout`
 范围：1-30

另请参见

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

Oracle 插件

概述

配置参数可用于自定义 Oracle plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 `oracle.conf`），而不是放在 agent 2 配置文件中。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数是必需的，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.Oracle.CallTimeout	请求完成的最大等待时间（秒）。此参数当前不起作用；改为使用item timeout 设置。
Plugins.Oracle.ConnectTimeout	指定建立连接的最大等待时间（秒）。
Plugins.Oracle.CustomQueriesPath	指定包含自定义查询 `.sql` 文件的目录完整路径。
Plugins.Oracle.CustomQueriesEnabled	启用 `oracle.custom.query` 监控项键的执行。
Plugins.Oracle.Default.Password	指定默认密码。
Plugins.Oracle.Default.Service	指定默认服务名。
Plugins.Oracle.Default.URI	指定默认 URI。
Plugins.Oracle.Default.User	指定默认用户名。
Plugins.Oracle.KeepAlive	设置在关闭未使用的 plugin 连接之前的最大等待时间（秒）。
Plugins.Oracle.ResolveTNS	指定如何解释用于连接 Oracle 服务器的连接字符串（ConnString）。
Plugins.Oracle.Sessions.<SessionName>.Password	指定命名会话的密码。
Plugins.Oracle.Sessions.<SessionName>.Service	指定命名会话的服务名。
Plugins.Oracle.Sessions.<SessionName>.URI	指定命名会话的 URI。
Plugins.Oracle.Sessions.<SessionName>.User	指定命名会话的用户名。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件。
- 以 \# 开头的注释仅在行首支持。

参数详情

Plugins.Oracle.CallTimeout

请求完成的最长等待时间（秒）。
 此参数当前不起作用。最长等待时间由监控项超时 决定。

默认值：Timeout[1]
 范围：1-30

Plugins.Oracle.ConnectTimeout

指定建立连接的最长等待时间（秒）。

默认值：Timeout[1]
 范围：1-30

Plugins.Oracle.CustomQueriesPath

指定包含自定义查询 .sql 文件的目录完整路径。

默认值：
/usr/local/share/zabbix/custom-queries/oracle - 适用于 Unix 系统，
*:\Program Files\Zabbix Agent 2\Custom Queries\Oracle - 适用于 Windows 系统（其中 * 是 ProgramFiles 环境变量中的驱动器名称）

示例：

```
/etc/zabbix/oracle/sql
```

Plugins.Oracle.CustomQueriesEnabled

启用 oracle.custom.query 监控项键的执行。
 如果禁用，则不会从自定义查询目录路径加载任何查询。

默认值：false
 范围：true/false

Plugins.Oracle.Default.Password

指定用于连接 Oracle 的默认密码。
 当监控项键或命名会话中未指定值时使用。

插件.Oracle.Default.Service

指定用于连接 Oracle 的默认服务名称（不支持 SID）。
 当监控项键或命名会话中未指定值时使用。

Plugins.Oracle.Default.Uri

指定用于连接 Oracle 的默认 URI。
 当监控项键值或命名会话中未指定值时使用。
 嵌入的凭据将被忽略。
 必须符合 URI 格式。
 支持的 scheme：tcp、tcps
 可以省略 schema。
 可以省略端口（默认值 =1521）。
 也可以将 TNS key 或 TNS value 作为连接字符串指定。TNS value 必须在不包含空白字符的情况下组成。

示例：

```
tcp://127.0.0.1:1521
localhost
zbx_tns_example (TNS key)
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=xe))) (TNS val
```

插件.Oracle.Default.User

指定用于连接 Oracle 的默认用户名。
 当监控项键或命名会话中未指定值时使用。

Plugins.Oracle.KeepAlive

设置在关闭未使用的插件连接之前的最长等待时间（秒）。

默认值：300
 范围：60-900

Plugins.Oracle.ResolveTNS

指定如何解释用于连接 Oracle 服务器的连接字符串（ConnString）。
 如果将其设置为 false，则会向 ConnString 添加默认协议和端口（如果缺失）。
 如果将该选项设置为 true，则会省略默认协议和端口（除非在 ConnString 中明确指定），并且 ConnString 将按原样传递给 Oracle 客户端。如果 Oracle 客户端在 tnsnames.ora 文件中找到 ConnString，则将使用找到的连接描述来连接 Oracle 服务器。

默认值：true
 范围：true/false

Plugins.Oracle.Sessions.<SessionName>.Password

指定用于连接 Oracle 的命名会话密码。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.Oracle.Sessions.<SessionName>.Service

指定用于连接 Oracle 的命名会话服务名（不支持 SID）。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.Oracle.Sessions.<SessionName>.Uri

指定用于连接 Oracle 的命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 嵌入的凭据将被忽略。
 必须符合 URI 格式。
 支持的 scheme：tcp、tcps
 可以省略 schema。
 可以省略端口（默认值 =1521）。
 也可以将 TNS key 或 TNS value 作为连接字符串指定。TNS value 必须在不包含空白字符的情况下组成。

示例：

```
tcp://127.0.0.1:1521
localhost
zbx_tns_example (TNS key)
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=xe))) (TNS val
```

Plugins.Oracle.Sessions.<SessionName>.User

指定用于连接 Oracle 的命名会话用户名。
 <SessionName> - 定义一个会话名称，用于监控项键中。

脚注

[1] 超时时间由 Zabbix agent 2 (UNIX) 或 Zabbix agent 2 (Windows) 配置文件（zabbix_agent2.conf）中的 Timeout 选项定义。

另请参阅

- 常规 Zabbix agent 2 配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

PostgreSQL 插件

概述

配置参数可用于自定义 PostgreSQL plugin 的行为。

建议将这些参数指定在它们自己的配置文件中（例如 postgresql.conf），而不是放在 agent 2 配置文件中。然后使用Include 指令将该文件包含到 Zabbix agent 2 配置中。

PostgreSQL plugin 是一个可加载的 plugin，可在 PostgreSQL plugin repository 中查看并获得完整说明。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.PostgreSQL.CallTimeout	设置请求完成前允许等待的最长时间（秒）。
Plugins.PostgreSQL.CustomQueriesPath	指定包含自定义查询 .sql 文件的目录完整路径。
Plugins.PostgreSQL.CustomQueriesEnabled	启用 postgresql.custom.query 监控项键的执行。
Plugins.PostgreSQL.Default.CachingMode	指定 PostgreSQL 连接的默认缓存模式。
Plugins.PostgreSQL.Default.Database	指定默认数据库名称。
Plugins.PostgreSQL.Default.Password	指定默认密码。
Plugins.PostgreSQL.Default.TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.PostgreSQL.Default.TLSCertFile	指定包含 PostgreSQL 证书或证书链的文件完整路径。
Plugins.PostgreSQL.Default.TLSConnect	指定 Zabbix agent 2 与被监控数据库之间通信的默认加密级别。
Plugins.PostgreSQL.Default.TLSKeyFile	指定包含 PostgreSQL 私钥的文件完整路径。
Plugins.PostgreSQL.Default.Uri	指定默认 URI。
Plugins.PostgreSQL.Default.User	指定默认用户名。
Plugins.PostgreSQL.KeepAlive	设置在关闭未使用的 plugin 连接之前的最长等待时间（秒）。
Plugins.PostgreSQL.Sessions.<SessionName>.CachingMode	指定命名会话的 PostgreSQL 连接缓存模式。
Plugins.PostgreSQL.Sessions.<SessionName>.Database	指定命名会话的数据库名称。
Plugins.PostgreSQL.Sessions.<SessionName>.Password	指定命名会话的密码。
Plugins.PostgreSQL.Sessions.<SessionName>.TLSCAFile	指定包含用于对等证书验证的顶级 CA 证书的文件完整路径。
Plugins.PostgreSQL.Sessions.<SessionName>.TLSCertFile	指定包含 PostgreSQL 证书或证书链的文件完整路径。
Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect	指定命名会话中 Zabbix agent 2 与被监控数据库之间通信的加密级别。
Plugins.PostgreSQL.Sessions.<SessionName>.TLSKeyFile	指定包含 PostgreSQL 私钥的文件完整路径。
Plugins.PostgreSQL.Sessions.<SessionName>.Uri	指定命名会话的 URI。
Plugins.PostgreSQL.Sessions.<SessionName>.User	指定命名会话的用户名。

|Plugins.PostgreSQL.System.Path| 指定 PostgreSQL plugin 可执行文件的路径。 | |Plugins.PostgreSQL.Timeout| 设置请求执行超时时间。|

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值。
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件。
- 以"#" 开头的注释仅在行首支持。

参数详情

Plugins.PostgreSQL.CallTimeout

设置请求完成的最大等待时间（秒）。

默认值：`global timeout`
 范围：1-30

Plugins.PostgreSQL.CustomQueriesPath

指定包含自定义查询 `.sql` 文件的目录的完整路径名。

默认值：
`/usr/local/share/zabbix/custom-queries/postgreSQL` - 适用于 Unix 系统，
`*:\Program Files\Zabbix Agent 2\Custom Queries\PostgreSQL` - 适用于 Windows 系统（其中 * 是来自 ProgramFiles 环境变量的驱动器名称）

示例：

```
/etc/zabbix/oracle/sql
```

Plugins.PostgreSQL.CustomQueriesEnabled

启用 `postgresql.custom.query` 监控项键的执行。
 如果禁用，则不会从自定义查询目录路径加载任何查询。

默认值：`false`
 范围：`true/false`

Plugins.PostgreSQL.Default.CacheMode

指定 PostgreSQL 连接的默认缓存模式。

默认值：`prepare`
 范围：
`prepare` - 在 PostgreSQL 服务器上创建预处理语句；
`describe` - 使用匿名预处理语句来描述语句，而不在服务器上创建语句。

请注意，`describe` 主要适用于环境不允许使用预处理语句的情况，例如运行 PgBouncer 之类的连接池时。

Plugins.PostgreSQL.Default.Database

指定用于连接 PostgreSQL 的默认数据库。
 当监控项键或命名会话中未指定值时使用。

Plugins.PostgreSQL.Default.Password

指定用于连接 PostgreSQL 的默认密码。
 当监控项键或命名会话中未指定值时使用。

Plugins.PostgreSQL.Default.TLSCAFile

指定包含用于验证 Zabbix agent 2 与被监控数据库之间加密通信的对等证书的顶级 CA 证书的文件完整路径。
 如果未在命名会话中指定值，则使用此项。

必填：`yes`，如果 `Plugins.PostgreSQL.Default.TLSConnect` 设置为 `verify_ca` 或 `verify_full`

Plugins.PostgreSQL.Default.TLSCertFile

指定包含 PostgreSQL 证书或证书链的文件的完整路径名，用于 Zabbix agent 2 与被监控数据库之间的加密通信。
 如果未在命名会话中指定值，则使用此项。

必填：`yes`，如果已指定 `Plugins.PostgreSQL.Default.TLSKeyFile`

Plugins.PostgreSQL.Default.TLSConnect

指定 Zabbix agent 2 与被监控数据库之间通信的默认加密级别。
 当命名会话中未指定值时使用。

范围：
`required` - 使用 TLS 作为传输模式进行连接，不进行身份检查；
`verify_ca` - 使用 TLS 连接并验证证书；
`verify_full` - 使用 TLS 连接，验证证书，并验证 DBHost 指定的数据库标识（CN）是否与其证书匹配。

未定义的加密类型表示未加密连接。

Plugins.PostgreSQL.Default.TLSKeyFile

指定包含 PostgreSQL 私钥的文件的完整路径名，用于 Zabbix agent 2 与受监控数据库之间的加密通信。
 如果未在命名会话中指定值，则使用此项。

必填：`yes`，如果已指定 `Plugins.PostgreSQL.Default.TLSCertFile`

Plugins.PostgreSQL.Default.Uri

指定用于连接 PostgreSQL 的默认 URI。
 当监控项键值或命名会话中未指定值时使用。
 嵌入的凭据将被忽略。
 必须符合 URI 格式。
 支持的方案：tcp、unix

示例：

```
tcp://127.0.0.1:5432
tcp://localhost
unix:/var/run/postgresql/.s.PGSQL.5432
```

Plugins.PostgreSQL.Default.User

指定用于连接 PostgreSQL 的默认用户名。
 当监控项键或命名会话中未指定值时使用。

Plugins.PostgreSQL.KeepAlive

设置在关闭未使用的插件连接之前的最长等待时间（秒）。

Plugins.PostgreSQL.Sessions.<SessionName>.CacheMode

指定 PostgreSQL 连接的命名会话缓存模式。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

默认值：prepare
 范围：
prepare - 在 PostgreSQL 服务器上创建预编译语句；
describe - 使用匿名预编译语句来描述语句，而不在服务器上创建语句。

请注意，describe 主要适用于环境不允许使用预编译语句的情况，例如在使用 PgBouncer 之类的连接池时。

Plugins.PostgreSQL.Sessions.<SessionName>.Database

指定用于连接 PostgreSQL 的命名会话数据库。
 <SessionName> - 定义用于监控项键中的会话名称。

Plugins.PostgreSQL.Sessions.<SessionName>.Password

指定用于连接 PostgreSQL 的命名会话密码。
 <SessionName> - 定义用于监控项键中的会话名称。
 必须符合密码格式。

Plugins.PostgreSQL.Sessions.<SessionName>.TLSCAFile

指定包含用于对 Zabbix agent 2 与受监控数据库之间加密通信进行对等证书验证的顶级 CA 证书的文件完整路径。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

必填：yes，如果 Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect 设置为 verify_ca 或 verify_full

Plugins.PostgreSQL.Sessions.<SessionName>.TLSCertFile

指定包含 PostgreSQL 证书或证书链的文件完整路径，用于 Zabbix agent 2 与受监控数据库之间的加密通信。
 <SessionName> - 定义用于监控项键中的会话名称。

必填：yes，如果指定了 Plugins.PostgreSQL.Sessions.<SessionName>.TLSKeyFile

Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect

指定会话级别的加密方式，用于 Zabbix agent 2 与受监控数据库之间的通信。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

范围：
required - 使用 TLS 作为传输模式进行连接，不进行身份检查；
verify_ca - 使用 TLS 连接并验证证书；
verify_full - 使用 TLS 连接，验证证书，并验证 DBHost 指定的数据库标识（CN）与其证书匹配。

未定义加密类型表示未加密连接。

Plugins.PostgreSQL.Sessions.<SessionName>.TLSKeyFile

指定包含 PostgreSQL 私钥的文件完整路径，用于 Zabbix agent 2 与被监控数据库之间的加密通信。
 <SessionName> - 定义一个会话名称，用于监控项键值中。

必填：yes，如果已指定 Plugins.PostgreSQL.Sessions.<SessionName>.TLSCertFile。

Plugins.PostgreSQL.Sessions.<SessionName>.Uri

指定用于连接 PostgreSQL 的命名会话 URI。
 <SessionName> - 定义一个会话名称，用于监控项键值中。
 嵌入的凭据将被忽略。
 必须符合 URI 格式。
 支持的 scheme：tcp、unix

示例：

```
tcp://127.0.0.1:5432
tcp://localhost
unix:/var/run/postgresql/.s.PGSQL.5432
```

Plugins.PostgreSQL.Sessions.<SessionName>.User

指定用于连接 PostgreSQL 的命名会话用户名。
 <SessionName> - 定义一个会话名称，用于监控项键中。

Plugins.PostgreSQL.System.Path

指定 PostgreSQL 插件可执行文件的路径。

必填：yes

示例：

```
Plugins.PostgreSQL.System.Path=/usr/libexec/zabbix/zabbix-agent2-plugin-postgresql
```

Plugins.PostgreSQL.Timeout

设置请求执行超时时间（即在关闭请求之前等待其完成的时长，单位为秒）。

默认值：global timeout
 范围：1-30

选项

Parameter	Description
-V --version	打印插件版本和许可证信息。
-h --help	打印帮助信息（简写）。
-t, --test <item key>	启动插件进行测试（忽略插件配置）。

另请参阅

- 常规 Zabbix agent 2 配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

Redis 插件

概述

配置参数可用于自定义 Redis 插件的行为。

建议将这些参数指定在它们自己的配置文件中（例如 redis.conf），而不是写入 agent 2 配置文件。然后使用Include 指令将此文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请单击其名称）
- 参数详情

Parameter	Description
Plugins.Redis.Default.Password	指定用于连接 Redis 的默认密码。
Plugins.Redis.Default.TLSConnect	指定 Zabbix agent 2 通信所需的加密级别。
Plugins.Redis.Default.TLSCAFile	指定包含顶级 CA 证书的文件完整路径名。
Plugins.Redis.Default.TLSCertFile	指定包含 agent 证书的文件完整路径名。
Plugins.Redis.Default.TLSKeyFile	指定包含 agent 私钥的文件完整路径名。
Plugins.Redis.Default.Uri	指定用于连接 Redis 的默认 URI。
Plugins.Redis.Default.User	指定用于连接 Redis 的默认用户名。
Plugins.Redis.KeepAlive	设置在关闭未使用的插件连接之前的最长等待时间（秒）。
Plugins.Redis.Sessions.<Session name>.Password	指定用于连接 Redis 的命名会话密码。
Plugins.Redis.Sessions.<Session name>.TLSConnect	指定命名会话的 Zabbix agent 2 通信所需的加密级别。
Plugins.Redis.Sessions.<Session name>.TLSCAFile	指定命名会话中包含顶级 CA 证书的文件完整路径名。
Plugins.Redis.Sessions.<Session name>.TLSCertFile	指定命名会话中包含 agent 证书的文件完整路径名。
Plugins.Redis.Sessions.<Session name>.TLSKeyFile	指定命名会话中包含 agent 私钥的文件完整路径名。
Plugins.Redis.Sessions.<Session name>.Uri	指定用于连接 Redis 的命名会话 URI。
Plugins.Redis.Sessions.<Session name>.User	指定用于连接 Redis 的命名会话用户名。
Plugins.Redis.Timeout	设置请求执行超时时间。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件。

- 以”#” 开头的注释仅在行首支持。

参数详情

Plugins.Redis.Default.Password

指定用于连接 Redis 的默认密码。
 当监控项键或命名会话中未指定值时使用。

Plugins.Redis.Default.TLSConnect

指定 Zabbix agent 2 通信所需的加密级别。
 空值或未设置的值将被视为不使用 TLS。

范围：`required` - 加密，未验证（仅用于测试），`verify_ca` - 加密，验证服务器证书，`verify_full` - 加密，验证服务器证书和通过 SAN 验证服务器名称

Plugins.Redis.Default.TLSCAFile

指定包含顶级 CA 证书的文件的完整路径名。

Plugins.Redis.Default.TLSCertFile

指定包含 agent 证书的文件的完整路径名。

Plugins.Redis.Default.TLSKeyFile

指定包含 agent 私钥的文件的完整路径名。

Plugins.Redis.Default.Uri

指定用于连接 Redis 的默认 URI。
 如果在监控项键或命名会话中未指定值，则使用此项。
 嵌入的凭据将被忽略。必须符合 URI 格式。
 支持的 scheme：`tcp`、`unix`
 可以省略 scheme。
 可以省略端口（默认值 =6379）。

默认值：`tcp://localhost:6379`

示例：

```
tcp://localhost:6379
localhost
unix:/var/run/redis.sock
```

Plugins.Redis.Default.User

指定用于连接 Redis 的默认用户名。
 当监控项键值或命名会话中未指定值时使用。

默认值：`default`

Plugins.Redis.KeepAlive

设置在关闭未使用的插件连接之前的最长等待时间（秒）。

默认值：300
 范围：60-900

Plugins.Redis.Sessions.<SessionName>.Password

指定用于连接 Redis 的命名会话密码。
 <SessionName> - 定义要在监控项键中使用的会话名称。

Plugins.Redis.Sessions.<SessionName>.TLSConnect

指定命名会话的 Zabbix agent 2 通信所需的加密级别。
 <SessionName> - 定义要在监控项键中使用的会话名称。
 空值或未设置的值将被视为不使用 TLS。

范围：`required` - 加密，未验证（仅用于测试），`verify_ca` - 加密，验证服务器证书，`verify_full` - 加密，验证服务器证书和服务器名称，并通过 SAN 进行验证

Plugins.Redis.Sessions.<SessionName>.TLSCAFile

指定包含该命名会话的顶级 CA 证书的文件完整路径。
 <SessionName> - 定义要在监控项键中使用的会话名称。

必填：`yes`，如果 `Plugins.Redis.Sessions.<SessionName>.TLSConnect` 设置为 `verify_ca` 或 `verify_full`

Plugins.Redis.Sessions.<SessionName>.TLSCertFile

指定包含该 agent 证书的文件完整路径，用于指定会话。
 <SessionName> - 定义要在监控项键中使用的会话名称。

Plugins.Redis.Sessions.<SessionName>.TLSKeyFile

指定包含该 session 的 agent 私钥的文件完整路径。
 <SessionName> - 定义要在监控项键中使用的 session 名称。

Plugins.Redis.Sessions.<SessionName>.Uri

指定用于连接 Redis 的命名会话 URI。
 <SessionName> - 定义要在监控项键中使用的会话名称。
 嵌入式凭据将被忽略。必须符合 URI 格式。
 支持的 scheme：tcp、unix
 可以省略 scheme。
 可以省略端口（默认值 =6379）。

默认值：tcp://localhost:6379

示例：

```
tcp://localhost:6379
localhost
unix:/var/run/redis.sock
```

Plugins.Redis.Sessions.<SessionName>.User

指定用于连接 Redis 的命名会话用户名。
 <SessionName> - 定义要在监控项键中使用的会话名称。

Plugins.Redis.Timeout

设置请求执行超时时间（即在关闭请求之前等待其完成的时长，单位为秒）。

默认值：global timeout
 范围：1-30

另请参见

- 常规 Zabbix agent 2 配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

SMART 插件

概述

配置参数可用于自定义 SMART 插件的行为。

建议将这些参数指定在它们自己的配置文件中（例如 smart.conf），而不是写在 agent 2 配置文件中。然后使用Include 指令将该文件包含到 Zabbix agent 2 配置中。

除非明确说明该参数为必填，否则所有参数均为可选。

本页结构如下：

- 参数概览（要查看某个参数的完整详情，请点击其名称）
- 参数详情

Parameter	Description
Plugins.Smart.Path	指定 smartctl 可执行文件的路径。
Plugins.Smart.Timeout	设置请求执行超时时间（即在关闭请求之前等待其完成的时长，单位为秒）。

请注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- smartctl 可执行文件的路径必须通过将其目录添加到系统的 PATH 环境变量中，或通过配置 Plugins.Smart.Path 来设置；这同样适用于 Linux 和 Windows；
- Zabbix 仅支持 UTF-8 编码且不带 BOM 的配置文件；
- 以”#” 开头的注释仅在行首支持。

参数详情

Plugins.Smart.Path

指定 smartctl 可执行文件的路径。

默认值：smartctl

Plugins.Smart.Timeout

设置请求执行超时时间（即在关闭请求之前等待请求完成的时长，单位为秒）。

默认值：global timeout
 范围：1-30

另请参阅

- Zabbix agent 2 通用配置参数说明：Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- 配置插件 的说明

Zabbix agent 2 插件协议

Zabbix agent 2 协议基于代码、大小和数据模型。

代码

Type	Size	Comments
Byte	4	有效负载类型，目前仅支持 JSON。

大小

Type	Size	Comments
Byte	4	当前有效负载的字节大小。

负载数据

Type	Size	Comments
Byte	由 *Size* 字段定义	JSON 格式的数据。

负载数据定义

通用数据

以下参数存在于所有请求/响应中：

Name	Type	Comments
id	uint32	对于请求 - 用于将请求与响应关联起来的递增标识符。在单个请求方向内唯一（即从 agent 到插件，或从插件到 agent）。 对于响应 - 对应请求的 ID。
type	uint32	请求类型。

日志请求

由插件发送的请求，用于将日志消息写入 agent 日志文件。

direction	plugin → agent
response	no

日志请求特有的参数：

Name	Type	Comments
severity	uint32	消息严重级别（日志级别）。
message	string	要记录的消息。

示例：

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

注册请求

agent 在启动阶段发送的请求，用于获取所提供的监控项并注册插件。

direction	agent → plugin
response	yes

注册请求特有的参数：

Name	Type	Comments
version	string	协议版本 <major>.<minor>

示例：

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

注册响应

插件对注册请求的响应。

direction	plugin → agent
response	n/a

注册响应特有的参数：

Name	Type	Comments
name	string	插件名称。
metrics	array of strings (optional)	插件中使用的带描述的监控项。返回 RegisterMetrics()。如果返回错误，则不存在。
interfaces	uint32 (optional)	插件支持的接口位掩码。如果返回错误，则不存在。
error	string (optional)	如果插件无法启动，则返回的错误消息。如果返回监控项，则不存在。

示例：

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

或

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

启动请求

用于执行 Runner 接口的 Start 函数的请求。

direction	agent → plugin
response	no

该请求没有特定参数，只包含common data 参数。

示例：

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

终止请求

由 agent 发送给插件以关闭插件的请求。

direction	agent → plugin
response	no

该请求没有特定参数，只包含common data 参数。

示例：

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

导出请求

用于执行 Exporter 接口的 Export 函数的请求。

direction	agent → plugin

response	no

导出请求特有的参数：

Name	Type	Comments
key	string	插件键。
parameters	array of strings (optional)	Export 函数的参数。

示例：

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

导出响应

来自 Exporter 接口的 Export 函数的响应。

direction	plugin → agent
response	n/a

导出响应特有的参数：

Name	Type	Comments
value	string (optional)	来自 Export 函数的响应值。如果返回错误，则不存在。
error	string (optional)	如果 Export 函数未成功执行，则为错误消息。如果返回 value，则不存在。

示例：

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

或

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

配置请求

用于执行 *Configurator* 接口的 *Configure* 函数的请求。

direction	agent → plugin
response	n/a

Configure 请求的特定参数：

Name	Type	Comments
global_options	JSON object	包含全局 agent 配置选项的 JSON 对象。
private_options	JSON object (optional)	包含私有插件配置选项的 JSON 对象（如果提供）。

示例：

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

验证请求

用于执行 *Configurator* 接口的 *Validate* 函数的请求。

direction	agent → plugin
response	yes

Validate 请求的特定参数：

Name	Type	Comments
private_options	JSON object (optional)	包含私有插件配置选项的 JSON 对象（如果提供）。

示例：

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

验证响应

来自 *Configurator* 接口的 *Validate* 函数的响应。

direction	plugin → agent
response	n/a

Validate 响应特有的参数：

Name	Type	Comments
error	string (optional)	如果 Validate 函数未成功执行，则返回的错误消息。若执行成功，则不存在。

示例：

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

或

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

Java 网关

概述　Zabbix Java 网关可以从源代码或软件包安装。

Zabbix 原生支持以名为 “Zabbix Java 网关” 的 Zabbix 守护进程监控 JMX 应用程序。Zabbix Java 网关是使用 Java 编写的守护进程。要获取主机上特定 JMX 计数器的值，Zabbix 服务器会查询 Zabbix Java 网关，而后者使用 JMX 管理 API 远程查询目标应用程序。应用程序无需安装任何其他软件，只需在命令行中使用 `-Dcom.sun.management.jmxremote` 选项启动即可。

Java 网关接受来自 Zabbix 服务器或 proxy 的传入连接，并且只能作为 “被动 proxy” 使用。与 Zabbix proxy 不同，它也可以由 Zabbix proxy 使用（Zabbix proxy 不能进行链式连接）。每个 Java 网关的访问权限都直接在 Zabbix 服务器或 Zabbix proxy配置文件中进行配置，因此每个 Zabbix 服务器或 Zabbix proxy 只能配置一个 Java 网关。如果某个主机包含 *JMX agent* 类型的监控项和其他类型的监控项，则只有 *JMX agent* 监控项会被传递给 Java 网关进行获取。

当监控项需要通过 Java 网关进行更新时，Zabbix 服务器或 proxy 会连接到 Java 网关并请求值，Java 网关随后获取该值并将其传回服务器或 proxy。因此，Java 网关不会缓存任何值。

Zabbix 服务器或 proxy 有一类专门用于连接 Java 网关的进程，其数量由 `StartJavaPollers` 选项控制。在内部，Java 网关会启动多个线程，其数量由 `START_POLLERS` 选项控制。在服务器端，如果连接耗时超过 “Timeout” 秒，连接将被终止，但 Java 网关可能仍在忙于从 JMX 计数器获取值。为解决此问题，Java 网关提供了 `TIMEOUT` 选项，可用于设置 JMX 网络操作的超时时间。

另请参阅：

- 配置参数：Zabbix Java 网关
- 从 Debian/Ubuntu 软件包进行设置
- 从 RHEL 软件包进行设置
- 从源代码进行设置

Zabbix 服务器或 proxy 会尽可能将针对单个 JMX 目标的请求进行汇总（受监控项间隔影响），并通过单个连接发送到 Java 网关，以获得更好的性能。

建议将 `StartJavaPollers` 设置为小于或等于 `START_POLLERS`；否则，可能会出现 Java 网关中没有可用线程来处理传入请求的情况。在这种情况下，Java 网关会使用 `ThreadPoolExecutor.CallerRunsPolicy`，这意味着主线程将处理传入请求，并且在此期间不会接受任何新请求。

如果您尝试使用 Zabbix Java 网关监控基于 Wildfly 的 Java 应用程序，请安装 Wildfly 下载页面 上提供的最新 `jboss-client.jar`。

配置参数：**Zabbix Java** 网关

如果您使用 `startup.sh` 和 `shutdown.sh` 脚本来启动Zabbix Java 网关，则可以在 `settings.sh` 文件中指定所需的配置参数。启动和关闭脚本会读取该设置文件，并负责将 shell 变量（列在第一列）转换为 Java 属性（列在第二列）。

如果您通过直接运行 `java` 手动启动 Zabbix Java 网关，则需要在命令行中指定相应的 Java 属性。

Variable	Property	Mandatory	Range	Default	Description
LISTEN_IP	zabbix.listenIP	no		0.0.0.0	要监听的 IP 地址。
LISTEN_PORT	zabbix.listenPort	no	1024-32767	10052	要监听的端口。
PID_FILE	zabbix.pidFile	no		/tmp/zabbix_java.pid	PID 文件名。如果省略，Zabbix Java 网关将作为控制台应用程序启动。
PROPERTIES_FILE	zabbix.propertiesFile	no			属性文件名。可用于以键值格式设置额外属性，使其不会在命令行中显示，或覆盖现有属性。 例如： ”javax.net.ssl.trustStorePassword=<p
START_POLLERS	zabbix.startPollers	no	1-1000	5	要启动的工作线程数量。
TIMEOUT	zabbix.timeout	no	1-30	3	网络操作的等待时间（秒）。

Warning:
端口 10052 未在 IANA 注册。

从 **Debian/Ubuntu** 软件包进行设置

概述

如果是通过软件包安装，以下信息将帮助您设置 Zabbix Java 网关。

配置并运行 Java 网关

Java 网关配置可以在以下文件中进行调整：

```
/etc/zabbix/zabbix_java_gateway.conf
```

有关更多详细信息，请参阅 Zabbix Java 网关配置参数。

要启动 Zabbix Java 网关：

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

要在启动时自动启动 Zabbix Java 网关：

```
systemctl enable zabbix-java-gateway
```

配置服务器以使用 Java 网关

在 Java 网关启动并运行后，您需要告诉 Zabbix 服务器在哪里查找 Zabbix Java 网关。这可以通过在服务器配置文件中指定 `JavaGateway` 和 `JavaGatewayPort` 参数来完成。如果运行 JMX 应用程序的主机由 Zabbix proxy 监控，则应改为在proxy 配置文件中指定连接参数。

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

默认情况下，服务器不会启动任何与 JMX 监控相关的进程。不过，如果您希望使用它，就必须指定预先 fork 的 Java poller 实例数量。您可以用与指定普通 poller 和 trapper 相同的方式来完成此操作。

```
StartJavaPollers=5
```

完成配置后，不要忘记重启服务器或 proxy。

调试 Java 网关

Zabbix Java 网关日志文件为：

```
/var/log/zabbix/zabbix_java_gateway.log
```

如果您想增加日志详细程度，请编辑以下文件：

```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

并将 `level="info"` 中的 `info` 改为 `debug`，甚至改为 `trace`（用于深入排查问题）：

```
<configuration scan="true" scanPeriod="15 seconds">
[...]
      <root level="info">
              <appender-ref ref="FILE" />
      </root>

</configuration>
```

JMX 监控

有关更多详细信息，请参阅JMX monitoring 页面。

从 **RHEL** 软件包进行设置

概述

如果是通过软件包安装，以下信息将帮助您设置 Zabbix Java 网关。

配置和运行 Java 网关

Zabbix Java 网关的配置参数可以在以下文件中调整：

```
/etc/zabbix/zabbix_java_gateway.conf
```

有关更多详细信息，请参阅 Zabbix Java 网关配置参数。

要启动 Zabbix Java 网关：

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

要在启动时自动启动 Zabbix Java 网关：

RHEL 7 及更高版本：

```
systemctl enable zabbix-java-gateway
```

RHEL 7 之前的版本：

```
chkconfig --level 12345 zabbix-java-gateway on
```

配置服务器以使用 Java 网关

在 Java 网关启动并运行后，您需要告诉 Zabbix 服务器在哪里查找 Zabbix Java 网关。
这可以通过在服务器配置文件中指定 `JavaGateway` 和 `JavaGatewayPort` 参数来完成。
如果运行 JMX 应用程序的主机由 Zabbix proxy 监控，则应改为在proxy 配置文件中指定连接参数。

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

默认情况下，服务器不会启动任何与 JMX 监控相关的进程。
不过，如果您希望使用该功能，则必须指定预先 fork 的 Java poller 实例数量。
其设置方式与指定普通 poller 和 trapper 相同。

```
StartJavaPollers=5
```

完成配置后，不要忘记重启服务器或 proxy。

调试 Java 网关

Zabbix Java 网关日志文件为：

```
/var/log/zabbix/zabbix_java_gateway.log
```

如果您想增加日志详细程度，请编辑以下文件：

```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

并将 `level="info"` 中的 `info` 改为 `debug`，甚至改为 `trace`（用于深入排查问题）：

```
<configuration scan="true" scanPeriod="15 seconds">
[...]
      <root level="info">
              <appender-ref ref="FILE" />
      </root>

</configuration>
```

JMX 监控

有关更多详细信息，请参阅JMX monitoring 页面。

从源代码安装

概述

如果是从源代码安装的，以下信息将帮助你设置 Zabbix Java 网关。

文件概览

如果您从源代码获取了 Java 网关，最终应会在 $PREFIX/sbin/zabbix_java 下得到一组 shell 脚本、JAR 文件和配置文件。这些文件的作用总结如下。

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

Java 网关本身的 JAR 文件。

```
lib/logback-core-1.5.25.jar
lib/logback-classic-1.5.25.jar
lib/slf4j-api-2.0.16.jar
lib/android-json-4.3_r3.1.jar
```

Java 网关的依赖项：Logback、SLF4J 和 Android JSON 库。

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

Logback 的配置文件。

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

用于启动和停止 Java 网关的便捷脚本。

`settings.sh`

由上述启动和停止脚本加载的配置文件。

配置和运行 Java 网关

默认情况下，Java 网关监听 10052 端口。如果您计划让 Java 网关运行在其他端口上，可以在 `settings.sh` 脚本中进行指定。
有关如何指定此项以及其他选项，请参见Java 网关配置文件的说明。

> **Warning:**
> 10052 端口未在 IANA 注册。

当您确认这些设置无误后，可以通过运行 `startup.sh` 脚本来启动 Java 网关：

```
./startup.sh
```

同样地，当您不再需要 Java 网关时，运行 `shutdown.sh` 脚本即可将其停止：

```
./shutdown.sh
```

请注意，与服务器或 proxy 不同，Java 网关是轻量级的，不需要数据库。

配置服务器以使用 Java 网关

在 Java 网关启动并运行后，您需要告诉 Zabbix 服务器在哪里查找 Zabbix Java 网关。这可以通过在服务器配置文件中指定 JavaGateway 和 JavaGatewayPort 参数来完成。如果运行 JMX 应用程序的主机由 Zabbix proxy 监控，则应改为在proxy 配置文件中指定连接参数。

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

默认情况下，服务器不会启动任何与 JMX 监控相关的进程。不过，如果您希望使用它，就必须指定预先 fork 的 Java poller 实例数量。您可以用与指定普通 poller 和 trapper 相同的方式来完成此操作。

```
StartJavaPollers=5
```

完成配置后，不要忘记重启服务器或 proxy。

Java 网关调试

如果 Java 网关存在任何问题，或者您在前端看到的关于某个监控项的错误信息不够详细，您可能需要查看 Java 网关日志文件。

默认情况下，Java 网关会将活动记录到 `/tmp/zabbix_java.log` 文件中，日志级别为"info"。有时这些信息还不够，此时需要将日志级别提高到"debug"。要提高日志级别，请修改 `lib/logback.xml` 文件，并将 `<root>` 标签的 level 属性改为 debug：

```
<root level="debug">
  <appender-ref ref="FILE" />
</root>
```

请注意，与 Zabbix 服务器或 Zabbix proxy 不同，在更改 `logback.xml` 文件后无需重启 Zabbix Java 网关 - `logback.xml` 中的更改会自动生效。调试完成后，您可以将日志级别恢复为"info"。

如果您希望将日志输出到其他文件，或者完全不同的介质（例如数据库），请调整 `logback.xml` 文件以满足您的需求。有关更多详细信息，请参阅 Logback Manual。

有时为了调试，使用控制台应用程序而不是守护进程来启动 Java 网关会很有帮助。为此，请在 `settings.sh` 中注释掉 `PID_FILE` 变量。如果省略 `PID_FILE`，`startup.sh` 脚本会将 Java 网关作为控制台应用程序启动，并让 Logback 改用 `lib/logback-console.xml` 文件；该文件不仅会将日志输出到控制台，还会启用"debug" 日志级别。

最后，请注意，由于 Java 网关使用 SLF4J 进行日志记录，您可以通过在 lib 目录中放置合适的 JAR 文件，将 Logback 替换为您选择的框架。有关更多详细信息，请参阅 SLF4J Manual。

JMX 监控

有关更多详细信息，请参阅JMX monitoring 页面。

发送器

概述　Zabbix sender 是一个命令行实用程序，可用于将性能数据发送到 Zabbix 服务器或 proxy 进行处理。

该实用程序通常用于长时间运行的用户脚本，以定期发送可用性和性能数据。

若要将结果直接发送到 Zabbix 服务器或 proxy，必须配置*Zabbix trapper* 监控项类型。

另请参阅：

- Zabbix sender 协议
- 适用于 Windows 的 Zabbix sender 动态链接库
- Zabbix 的 Python 库，该库内置了模拟 Zabbix sender 工作的功能

运行 **Zabbix sender**　运行 Zabbix UNIX sender 的示例：

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

其中：

- `z` - Zabbix 服务器主机（也可以使用 IP 地址）；
- `s` - 受监控主机的技术名称（在 Zabbix 前端中注册的名称）；
- `k` - 监控项键值；
- `o` - 要发送的值。

> **Attention:**
> 包含空格的选项必须使用双引号括起来。

Zabbix sender 可用于从输入文件发送多个值。有关更多信息，请参阅Zabbix sender 手册页。

如果指定了配置文件，Zabbix sender 将使用 agent `ServerActive` 配置参数中定义的所有地址发送数据。如果向某个地址发送失败，sender 会尝试向其他地址发送。如果向某个地址发送批量数据失败，则不会再向该地址发送后续批次的数据。

Zabbix sender 接受 UTF-8 编码的字符串（适用于类 Unix 系统和 Windows），文件开头不得包含字节顺序标记（BOM）。

Windows 上的 Zabbix sender 可以按如下方式运行：

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

zabbix_sender 的实时发送场景会收集短时间内连续传递给它的多个值，并通过单个连接将这些值发送到服务器。与前一个值的间隔不超过 0.2 秒的值可以放入同一堆栈中，但最大轮询时间仍为 1 秒。

Note:
如果指定的配置文件中存在无效的参数条目（未遵循 parameter=value 表示法），Zabbix sender 将终止运行。

使用 **Zabbix sender** 进行低级别发现　以下是使用 Zabbix sender 发送 JSON 格式值以进行低级别发现的示例：

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

要使其生效，低级别发现规则必须使用 *Zabbix trapper* 监控项类型（在本示例中，使用 trapper.discovery.item 键）。

Zabbix sender 协议

Zabbix 服务器和 Zabbix proxy 使用基于 JSON 的通信协议从 Zabbix sender 接收数据。
可以通过trapper 监控项 或启用 trapping 的HTTP agent 监控项 来接收数据。

请求和响应消息必须以header and data length 开头。

Zabbix sender 请求

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

Zabbix 服务器响应

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

带时间戳的 Zabbix sender 请求

或者，Zabbix sender 可以发送包含时间戳和纳秒的请求。

```
{
    "request": "sender data",
    "data": [
        {
            "host": "<hostname>",
            "key": "trap",
            "value": "test value",
            "clock": 1516710794,
            "ns": 592397170
        },
        {
            "host": "<hostname>",
            "key": "trap",
            "value": "test value",
            "clock": 1516710795,
            "ns": 192399456
        }
    ],
    "clock": 1516712029,
```

```
    "ns": 873386094
}
```

Zabbix 服务器响应

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

Windows 版 Zabbix sender 动态链接库

在 Windows 环境中，应用程序可以通过使用 Zabbix sender 动态链接库（zabbix_sender.dll）向 Zabbix 服务器/proxy 发送数据，而无需启动外部进程（zabbix_sender.exe）。

编译使用 zabbix_sender.dll 的用户应用程序时，需要 zabbix_sender.h 和 zabbix_sender.lib。

获取方式

获取 zabbix_sender.dll 有两种方式。

1. 以 ZIP 压缩包形式下载 zabbix_sender.h、zabbix_sender.lib 和 zabbix_sender.dll 文件。

在选择下载选项时，请确保在 *Encryption* 下选择"No encryption"，并在 *Packaging* 下选择"Archive"。然后下载 Zabbix agent（不要下载 Zabbix agent 2）。

zabbix_sender.h、zabbix_sender.lib 和 zabbix_sender.dll 文件位于下载的 ZIP 压缩包中的 bin\dev 目录下。请将这些文件解压到所需位置。

2. 从源代码构建 zabbix_sender.dll（请参见说明）。

包含开发文件的动态链接库位于 bin\winXX\dev 目录下。使用时，请包含 zabbix_sender.h 头文件，并链接 zabbix_sender.lib 库。

另请参见

- 示例：一个使用 Zabbix sender 动态链接库实现的简单 Zabbix sender 工具示例，用于说明该库的用法；
- zabbix_sender.h 文件：Zabbix sender 动态链接库的接口函数。该文件包含文档，说明每个接口函数的用途、参数和返回值。

获取

概述　Zabbix get 是一个命令行工具，可用于与 Zabbix agent 通信并从 agent 获取所需信息。

该工具通常用于排查 Zabbix agent 的故障。

另请参阅Python library for Zabbix，它内置了模拟 Zabbix get 的功能。

运行 **Zabbix get**　以下是在 UNIX 下运行 Zabbix get 以从 agent 获取处理器负载值的示例：

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

以下是运行 Zabbix get 以从网站抓取字符串的另一个示例：

```
cd bin
./zabbix_get -s 192.168.1.1 -p 10050 -k "web.page.regexp[www.example.com,,,\"USA: ([a-zA-Z0-9.-]+)\",,\1]"
```

请注意，这里的监控项键包含空格，因此使用引号将监控项键传递给 shell。引号不是监控项键的一部分；它们会被 shell 去除，不会传递给 Zabbix agent。

如果监控项键不受支持，Zabbix get 将返回退出代码 1。

Zabbix get 接受以下命令行参数：

```
-s --host <host name or IP>             指定主机的主机名或 IP 地址
-p --port <port number>                 指定主机上运行的 agent 的端口号（默认值：10050）
-I --source-address <IP address>        指定源 IP 地址
-t --timeout <seconds>                  指定超时时间。有效范围：1-600 秒（默认值：30 秒）
-k --key <item key>                     指定要检索其值的监控项键
-P --protocol <value>                   与 agent 通信所使用的协议。可选值：
```

```
                                        auto - 使用 JSON 协议连接，失败后回退并重试明文协议（默认）
                                        json - 使用 JSON 协议连接
                                        plaintext - 使用明文协议连接，仅发送监控项键（6.4.x 及更早版本）
-h --help                               显示此帮助信息
-V --version                            显示版本号

--tls-connect <value>                   如何连接到 agent。可选值：
                                        unencrypted - 不加密连接（默认）
                                        psk - 使用 TLS 和预共享密钥连接
                                        cert - 使用 TLS 和证书连接
--tls-ca-file <CA file>                 包含用于对等证书验证的顶级 CA 证书的文件完整路径
--tls-crl-file <CRL file>               包含已吊销证书的文件完整路径
--tls-agent-cert-issuer <cert issuer>   允许的 agent 证书颁发者
--tls-agent-cert-subject <cert subject> 允许的 agent 证书主题
--tls-cert-file <cert file>             包含证书或证书链的文件完整路径
--tls-key-file <key file>               包含私钥的文件完整路径
--tls-psk-identity <PSK-identity>       用于标识预共享密钥的唯一、区分大小写的字符串
--tls-psk-file <PSK-file>               包含预共享密钥的文件完整路径
--tls-cipher13 <cipher-string>          适用于 OpenSSL 1.1.1 或更新版本的 TLS 1.3 密码字符串。覆盖默认的密码套件选择
--tls-cipher <cipher-string>            GnuTLS 优先级字符串（适用于 TLS 1.2 及以上）或 OpenSSL 密码字符串（仅适用于
```

更多信息请参见Zabbix get 手册页。

在 Windows 上也可以类似地运行 Zabbix get：

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

JS

概述 zabbix_js 是一个命令行工具，可用于嵌入式脚本测试。

该工具会使用字符串参数执行用户脚本，并打印结果。脚本通过嵌入式 Zabbix 脚本引擎执行。

如果发生编译或执行错误，zabbix_js 会将错误输出到 stderr，并以代码 1 退出。

```
zabbix_js -s script-file -p input-param [-l log-level] [-t timeout]
zabbix_js -s script-file -i input-file [-l log-level] [-t timeout]
zabbix_js -h
zabbix_js -V
```

用法 zabbix_js 接受以下命令行参数：

```
-s, --script script-file            指定要执行的脚本文件名
                                    如果将文件名指定为 "-"，则脚本将从 stdin 读取
-i, --input input-file              指定输入信息的文件名
                                    其内容将作为值传递给脚本
                                    与 -p 互斥
                                    如果将文件名指定为 "-"，则脚本将从 stdin 读取
-p, --param input-param             指定输入参数：要作为值传递给脚本的变量
                                    与 -i 互斥
-l, --loglevel log-level            指定日志级别
                                    范围：0-5
-t, --timeout timeout               指定超时时间（秒）
                                    有效范围：1-600 秒（默认值：10 秒）
-h, --help                          显示帮助信息
-V, --version                       显示版本号
-w <webdriver url>                  启用浏览器监控
```

示例： 示例 1：使用包含值的变量执行脚本

脚本（script-file.js）：

```
return value;
```

用法：

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

输出：example_value

示例 2：使用包含输入信息的文件执行脚本

脚本文件（script-file.js）：

```
return value;
```

输入信息文件（example.txt）：

```
Example of input information from the file
```

用法：

```
zabbix_js -s script-file.js -i example.txt
```

输出：Example of input information from the file

示例 3：使用从 stdin 读取的输入执行脚本

脚本文件（script-file.js）：

```
return value;
```

用法：

```
zabbix_js -s script-file.js -i -
```

输入（stdin）：

```
Example of input from stdin
```

输出：Example of input from stdin

Web 服务

概述　Zabbix web service 是一个用于与外部 web 服务通信的进程。目前，Zabbix web service 用于生成和发送计划报告，未来计划添加更多功能。

Zabbix 服务器通过 HTTP(S) 连接到该 web service。Zabbix web service 要求在同一主机上安装Google Chrome；在某些发行版上，该服务也可能与 Chromium 一起工作（请参见已知问题）。

安装　官方的 zabbix-web-service 软件包可在 Zabbix 软件源中获取。

要从源代码编译 Zabbix web service，请指定 `--enable-webservice` 配置选项。

要配置 Zabbix web service，请更新web service 配置文件中的参数。

> **Attention:**
> 强烈建议使用证书在 Zabbix 服务器与 Zabbix web service 之间设置加密。默认情况下，Zabbix 服务器与 Zabbix web service 之间传输的数据未加密，这可能导致未经授权的访问。

配置参数：**Zabbix web service**

概述

Zabbix web service 是一个用于与外部 web 服务进行通信的进程。

本节列出了 Zabbix web service 配置文件（zabbix_web_service.conf）支持的参数。

这些参数仅列出名称，不附加其他信息。单击参数可查看完整详情。

Parameter	Description
AllowedIP	以逗号分隔的 IP 地址列表，可选使用 CIDR 表示法，或 Zabbix 服务器和 Zabbix proxy 的 DNS 名称。
DebugLevel	调试级别。
IgnoreURLCertErrors	指定访问前端 URL 时 TLS 证书验证错误的处理方式。
Include	你可以在配置文件中包含单个文件或目录中的所有文件。
ListenPort	服务将监听此端口，等待来自服务器的连接。
LogFile	日志文件名。
LogFileSize	日志文件的最大大小。
LogType	日志输出类型。
Timeout	格式化仪表板的 PDF report 所花费的最长时间（秒）。
TLSAccept	接受哪些传入连接。
TLSCAFile	包含用于对等证书验证的顶级 CA 证书的文件完整路径名，用于 Zabbix 组件之间的加密通信。
TLSCertFile	包含服务证书或证书链的文件完整路径名，用于 Zabbix 组件之间的加密通信。
TLSKeyFile	包含服务私钥的文件完整路径名，用于 Zabbix 组件之间的加密通信。

除非明确说明某个参数为必填，否则所有参数均为非必填。

注意：

- 默认值反映的是进程默认值，而不是随软件包提供的配置文件中的值；
- 值支持环境变量；
- Zabbix 仅支持 UTF-8 编码且不带BOM的配置文件；
- 以 # 开头的注释仅在行首支持。

参数详情

AllowedIP

以逗号分隔的 IP 地址列表，可选使用 CIDR 表示法，或 Zabbix 服务器和 Zabbix proxy 的 DNS 名称。仅接受来自此处列出的主机的传入连接。
 如果启用了 IPv6 支持，则 `127.0.0.1`、`::127.0.0.1`、`::ffff:127.0.0.1` 会被视为相同，且 `::/0` 将允许任何 IPv4 或 IPv6 地址。`0.0.0.0/0` 可用于允许任何 IPv4 地址。

示例：

`127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com`

必填：是

DebugLevel

指定调试级别：- *0* - 关于 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（会产生大量信息）；- *5* - 扩展调试（会产生更多信息）。

默认值：3
 范围：0-5

IgnoreURLCertErrors

指定访问前端 URL 时的 TLS 证书验证错误处理方式：
0 - 不忽略证书错误；
1 - 忽略证书错误。

默认值：0
 范围：0-1

包含

指定要包含到配置文件中的目录或单个文件。
 在 Zabbix 安装期间，除非你在编译时更改此参数，否则该参数会被设置为 `/usr/local/etc`。

`Include` 参数的一般规则：

- 路径可以相对于 `zabbix_web_service.conf` 文件所在位置。
- 你可以在一个配置文件中指定多个 `Include` 参数。
- Zabbix 会在读取到 `Include` 参数后立即处理它，因此建议将其放在配置文件末尾。

包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，该组件将无法启动。

包含目录或文件模式的规则：

- 不要假定包含顺序是固定的（例如，文件不会按字母顺序包含）。
- 不要在多个已包含文件中定义同一个参数（例如，用特定设置覆盖通用设置），因为你无法预测最终会使用哪个值。
- 注意某些文本编辑器会自动创建的备份副本(例如，编辑 `include/my_specific.conf` 可能会创建 `include/my_specific.conf.BAK`)，因为这两个文件都会被包含。

若要仅从目录中包含相关文件，请使用通配符（*）进行模式匹配。

示例：

```
Include=/absolute/path/to/config/files/*.conf
```

ListenPort

该服务将监听此端口，以接收来自服务器的连接。

默认值：10053
 范围：1024-32767

LogFile

日志文件的名称。

示例：

`/tmp/zabbix_web_service.log`

必需：是，如果 LogType 设置为 `file`；否则为否

LogFileSize

日志文件的最大大小，单位为 MB。
0 - 禁用自动日志轮转。
 注意：如果达到日志文件大小限制且文件轮转因任何原因失败，现有日志文件将被截断并重新开始。

默认值：1
 范围：0-1024

LogType

日志输出类型：- *file* - 将日志写入由 LogFile 参数指定的文件；- *system* - 将日志写入 syslog；- *console* - 将日志写入标准输出。

默认值：`file`

超时

用于格式化仪表板的 PDF 报告 所花费的最长时间（以秒为单位）。

默认值：10
 范围：1-30

TLSAccept

接受哪些传入连接：- *unencrypted* - 接受未加密连接（默认）；- *cert* - 接受使用 TLS 和证书的连接。

默认：`unencrypted`

TLSCAFile

用于 Zabbix 组件之间加密通信的、包含用于对等证书验证的顶级 CA 证书的文件完整路径。

TLSCertFile

包含服务证书或证书链的文件的完整路径名，用于与 Zabbix 组件之间的加密通信。

TLSKeyFile

包含服务私钥的文件的完整路径名，用于 Zabbix 组件之间的加密通信。

Bridge adapter

Zabbix bridge adapter 可让您的 Zabbix 服务器与Zabbix 移动应用进行通信。

Zabbix 移动应用不会直接连接到您的 Zabbix 服务器。相反，两个组件会在它们之间传递数据：

- Zabbix Bridge - 一个集中托管的服务，用于与移动设备上的 Zabbix 移动应用交换数据。
- Zabbix bridge adapter - 您需要安装并运行的组件。它将您的 Zabbix 服务器连接到 Zabbix Bridge。

您可以使用软件包将 bridge adapter 安装在与 Zabbix 服务器相同的系统上，也可以安装在其他系统上。

要配置 Zabbix bridge adapter，请更新其配置文件（默认路径：`/etc/zabbix/zabbix_bridge_adapter.conf`）。

要确保 Zabbix bridge adapter 的安全性，请参阅安全的移动访问。

数据流　使用 Zabbix 移动应用时，数据传输过程如下：

1. 您在 Zabbix 移动应用中执行操作，应用随后向 Zabbix Bridge 发送查询请求。
2. Zabbix Bridge 将查询作为 JSON-RPC 请求传递给 Zabbix bridge adapter。
3. Zabbix bridge adapter 检查请求的 Zabbix API 方法是否被允许。
   如果不允许该方法，bridge adapter 将拒绝请求。
   如果允许该方法，bridge adapter 将请求发送到 Zabbix API。
4. Zabbix API 处理请求，并将响应返回给 Zabbix bridge adapter。
5. Zabbix bridge adapter 将响应返回给 Zabbix Bridge。
6. Zabbix Bridge 将响应传递给 Zabbix 移动应用，应用会在您的移动设备上显示数据。

配置参数：**Zabbix bridge** 适配器

概述

配置参数可用于自定义 Zabbix bridge adapter 的行为。

请在配置文件（`zabbix_bridge_adapter.conf`）中指定参数值。除非明确说明参数为必需参数，否则所有参数均为可选参数。

本页面结构如下：

- 参数概览（要查看参数的完整详细信息，请点击其名称）
- 参数详细信息

参数	描述
AllowedAPIMethods	指定可通过 Zabbix bridge adapter 调用的 Zabbix API 方法。
DataDir	指定创建和存储 bridge adapter 状态文件及加密密钥的目录。
DebugLevel	设置调试级别。
Include	指定要包含在配置文件中的目录或单个文件。
ListenIP	指定 bridge adapter 监听来自 Zabbix 服务器的传入 HTTP(S) 连接时使用的 IP 地址。
ListenHealthPort	指定 bridge adapter 监听传入 HTTP 健康检查连接时使用的端口。
ListenPort	指定 bridge adapter 监听来自 Zabbix 服务器的传入 HTTP(S) 连接时使用的端口。
LogFile	指定 bridge adapter 日志文件的完整路径。
LogFileSize	设置日志文件的最大大小（以 MB 为单位）。
LogType	指定日志输出类型。
PidFile	指定 PID 文件。
TLSConfig.AllowedCN	指定允许的客户端 Common Name 的逗号分隔列表（仅适用于 `mTLS`）。
TLSConfig.CAFile	指定包含用于在 `mTLS` 模式下验证客户端证书的顶级 CA 证书的文件的完整路径。
TLSConfig.CertFile	指定包含证书或证书链的文件的完整路径。
TLSConfig.KeyFile	指定包含私钥的文件的完整路径。
TLSConfig.Mode	设置 `ListenPort` 上传入连接的 TLS 模式。
ServerId	指定 Zabbix 服务器的唯一标识符。必需。
ZabbixAPITLSCAFile	指定包含顶级 CA 证书的文件的完整路径，bridge adapter 使用这些证书验证 Zabbix API 证书。
ZabbixAPITLSConnect	设置与 Zabbix API 建立出站连接时使用的 TLS 模式。
ZabbixAPIURL	指定 Zabbix API 端点的 URL。必需。
ZabbixBridgeURL	指定用于获取 Zabbix Bridge 配置的 Zabbix Bridge URL（在 bridge adapter 启动时获取）。

请注意：

- 默认值反映的是进程默认值，而不是随附配置文件中的值。
- 值支持环境变量。
- Zabbix 仅支持使用 UTF-8 编码且不带 BOM 的配置文件；
- 以 # 开头的注释仅支持出现在行首。

参数详细信息

AllowedAPIMethods

指定可通过 Zabbix bridge adapter 调用的 Zabbix API 方法。
 对于多个方法，请重复此参数，每行指定一个方法。

每个参数值必须是精确的方法名称（例如 `problem.get`），或仅替换完整 API 名称（例如 `*.get`）或完整方法名称（例如 `problem.*`）的通配符模式。

任何其他值均无效，并会阻止 Zabbix bridge adapter 启动。这包括单独的通配符（*）、部分通配符模式（host* 或 host*.*）以及空值（AllowedAPIMethods=）。

示例：

```
AllowedAPIMethods=apiinfo.version
AllowedAPIMethods=problem.*
AllowedAPIMethods=*.get
```

Zabbix 移动应用所需的最小方法集：

```
AllowedAPIMethods=apiinfo.version
AllowedAPIMethods=device.onboard
AllowedAPIMethods=device.offboard
AllowedAPIMethods=user.get
AllowedAPIMethods=role.get
AllowedAPIMethods=problem.get
AllowedAPIMethods=event.get
AllowedAPIMethods=event.acknowledge
AllowedAPIMethods=trigger.get
AllowedAPIMethods=alert.get
AllowedAPIMethods=host.get
AllowedAPIMethods=hostgroup.get
```

默认值：未设置（拒绝所有 Zabbix API 方法）

DataDir

指定创建和存储 bridge adapter 状态文件及加密密钥的目录。
 该目录必须具有 0700 权限，其中存储的所有文件必须具有 0600 权限。

默认值：/var/lib/zabbix-bridge-adapter

DebugLevel

设置调试级别：

- 0 - 有关 Zabbix 进程启动和停止的基本信息
- 1 - 关键信息
- 2 - 错误信息
- 3 - 警告
- 4 - 用于调试（生成大量信息）
- 5 - 扩展调试（生成更多信息）

默认值：3
 范围：0-5

Include

指定要包含在配置文件中的目录或单个文件。
 安装 Zabbix 时，此参数默认设置为 /usr/local/etc，除非您在编译时更改了该设置。

Include 参数的一般规则：

- 路径可以是相对于 zabbix_bridge_adapter.conf 文件所在位置的相对路径。
- 可以在一个配置文件中指定多个 Include 参数。
- Zabbix 在读取 Include 参数后会立即对其进行处理，因此请考虑将其放置在配置文件末尾。

所包含文件的规则：

- 每个文件都必须可读。
- 每个文件都必须具有正确的语法。否则，组件将无法启动。

包含目录或文件模式的规则：

- 不要假定包含顺序（例如，文件不会按字母顺序包含）。
- 不要在多个所包含文件中定义相同的参数（例如，使用特定设置覆盖常规设置），因为无法预测最终使用哪个值。
- 注意某些文本编辑器会自动创建备份副本(例如，编辑 include/my_specific.conf 时可能会创建 include/my_specific.conf.BAK)，因为这两个文件都会被包含。

若要仅包含目录中的相关文件，请使用通配符 (*) 进行模式匹配。

示例：

```
Include=/absolute/path/to/config/files/*.conf
```

ListenIP

指定用于接收来自 Zabbix 服务器的 HTTP(S) 连接的桥接适配器监听 IP 地址。
 如果未指定，桥接适配器将在所有 IP 地址上监听。

ListenHealthPort

指定用于接收传入 HTTP 健康检查连接的 bridge 适配器监听端口。
 提供 `/health` 和 `/ready` 端点。
 如果未指定，则不会启动健康检查服务。

默认值：8081
 范围：1024-32767

ListenPort

指定 bridge adapter 用于接收来自 Zabbix 服务器的 HTTP(S) 连接的监听端口。

默认值：10054
 范围：1024-32767

LogFile

指定 bridge adapter 日志文件的完整路径。

必填：如果LogType 设置为 `file`，则为 是

默认值：`/tmp/zabbix_bridge_adapter.log`

LogFileSize

设置日志文件的最大大小（以 MB 为单位）。
 0 - 禁用自动日志轮换。

默认值：1
 范围：0-1024

LogType

指定日志输出类型：

- `file` - 将日志写入由 `LogFile` 参数指定的文件。
- `system` - 将日志写入 syslog。
- `console` - 将日志写入标准输出。

默认值：`file`

PidFile

指定 PID 文件。

默认值：`/tmp/zabbix_bridge_adapter.pid`

ServerId

指定 Zabbix 服务器的唯一标识符。
 可在 Zabbix 用户界面的报表 > 系统信息中查看。

必填：是

TLSConfig.AllowedCN

指定允许的客户端通用名称列表（仅适用于 `mTLS`），多个名称之间使用逗号分隔。
 如果未指定，bridge 适配器不会检查客户端通用名称。

TLSConfig.CAFile

指定包含用于在 `mTLS` 模式下验证客户端证书的顶级 CA 证书的文件完整路径。

TLSConfig.CertFile

指定包含证书或证书链的文件的完整路径。

TLSConfig.KeyFile

指定包含私钥的文件的完整路径。

TLSConfig.Mode

设置 `ListenPort` 上传入连接的 TLS 模式：

- `none` - 接受未加密的连接。
- `tls` - 使用最低版本为 v1.2 的标准 TLS。
- `mtls` - 在 TLS 的基础上启用双向身份验证。

默认值：none

ZabbixAPITLSCAFile

指定一个文件的完整路径名，该文件包含 bridge adapter 用于验证 Zabbix API 证书的顶级 CA 证书。

必填：如果ZabbixAPITLSConnect 设置为 verify_ca 或 verify_full，则为 是。

ZabbixAPITLSConnect

设置到 Zabbix API 的出站连接的 TLS 模式：

- disabled（或为空）- 使用普通 HTTP；ZabbixAPIURL 必须使用 http://。
- required - 使用 HTTPS，但不验证证书（仅用于测试）。
- verify_ca - 使用 HTTPS 并验证 Zabbix API 证书。
- verify_full - 使用 HTTPS，并验证 Zabbix API 证书和主机名。

ZabbixAPIURL

指定 Zabbix API 端点的 URL。

必填：是

示例：

```
ZabbixAPIURL=https://example.com/zabbix/api_jsonrpc.php
```

ZabbixBridgeURL

指定用于检索 Zabbix Bridge 配置的 Zabbix bridge URL（在 bridge 适配器启动时检索）。

默认值：ZabbixBridgeURL=https://test.aws.zabbix-lab.win/.well-known/bridge-configuration

MCP 服务器

概述　Zabbix MCP 服务器是一个允许 AI agent 使用模型上下文协议（MCP）与 Zabbix 协作的进程。MCP 服务器与 MCP 客户端和Zabbix API 通信。

使用 MCP 服务器，AI agent 可以：

- 获取主机组、模板组、主机、模板、监控项、LLD 规则、接口、宏、触发器、问题、维护周期和监控项历史数据（不包括二进制监控项）；
- 执行问题更新：添加评论、确认或取消确认、更改严重性、抑制或取消抑制，或关闭问题；
- 创建、更新和删除维护周期。

Zabbix API 令牌用于授权；令牌会随每个请求发送。访问权限受令牌所有者的权限限制：用户只能查看其有权访问的主机组中的数据，并且用户无权执行的操作将被拒绝。

MCP 服务器是无状态的：请求之间不会保留会话。

> **Note:**
> 虽然 Zabbix MCP 服务器可能适用于较旧或较新的 Zabbix 版本，但其兼容性仅正式支持与 MCP 服务器自身相同的 Zabbix 版本。

另请参阅：

- 配置参数：MCP 服务器
- 工具和资源

安装　官方的 zabbix-mcp-server 软件包可在 Zabbix 软件源中获取。

要从源代码安装 Zabbix MCP server，请使用 make 进行构建。

要配置 Zabbix MCP server，请更新MCP server 配置文件中的参数。至少必须设置 FrontendURL 和 AllowedIP 参数。

默认配置文件（zabbix_mcp_server.conf）位于 /etc/zabbix/，其中禁用了最常暴露敏感数据的工具。要启用某个工具，请删除相应的 DenyTool 行。

另请参阅：安全最佳实践

运行　要启动 MCP 服务器，请运行：

```
zabbix_mcp_server -c /etc/zabbix/zabbix_mcp_server.conf
```

要检查配置文件并退出而不启动服务器，请使用 -T 选项：

```
zabbix_mcp_server -c /etc/zabbix/zabbix_mcp_server.conf -T
```

支持以下命令行选项：

选项	描述
-c <file>	配置文件的路径。 默认值为 /etc/zabbix/zabbix_mcp_server.conf。
-T	验证配置文件并退出。
-h	显示帮助信息。
-V	显示版本号。

从源代码安装 MCP 服务器时，必须手动创建 systemd 单元，例如：

```
[Unit]
Description=Zabbix MCP server
After=network-online.target

[Service]
User=zabbix
ExecStart=/usr/local/sbin/zabbix_mcp_server -c /etc/zabbix/zabbix_mcp_server.conf
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

连接 **MCP** 客户端　任何支持 Streamable HTTP 传输的 MCP 客户端都可以连接到 MCP 服务器。需要配置以下设置：

设置	值
传输	Streamable HTTP
URL	https://<host>:<ListenPort>/mcp
标头	Authorization: Bearer <Zabbix API token>，每个请求都必须发送

每个用户都必须配置自己的 API token。

要验证设置，请向 MCP 服务器发送单个请求：

```
curl -sS https://mcp.example.com:8443/mcp \
  -H "Authorization: Bearer $ZABBIX_API_TOKEN" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json, text/event-stream" \
  -d '{"jsonrpc":"2.0","id":1,"method":"tools/list"}'
```

HTTP 状态代码 200 以及响应中可用工具的列表，表示 MCP 服务器可访问并接受该 token；该列表显示已公开的工具。其他状态代码表示以下问题：

- 400 - Accept 标头未同时包含传输所需的两种媒体类型；
- 401 - 缺少包含 token 的 Authorization 标头；
- 403 - 客户端地址未列在 AllowedIP 参数中。

审批和工具访问　如果 MCP 客户端支持表单引导，则每项更改在发送到 Zabbix 之前都必须获得用户批准。MCP 客户端会向用户显示将要进行的更改，只有在用户确认后才会应用更改；如果用户拒绝或关闭请求，Zabbix 中不会发生任何更改。批准有效期为五分钟，且仅适用于所显示的确切更改；在此期间可以重复使用该批准。

如果 MCP 客户端不支持表单引导（包括仅支持 URL 引导的客户端），则会立即应用更改，无需批准。在这种情况下，请限制写入工具和 Zabbix API 权限。

审批尤其重要，因为工具输出可能包含来自受监控系统的不可信文本，这些文本可能会影响 agent。

使用 AllowTool、DenyTool、AllowToolRegexp 和 DenyToolRegexp 控制 MCP 服务器公开哪些工具。被拒绝的工具无法调用或列出，其关联资源也无法访问。

Warning:
AllowTool、DenyTool、AllowToolRegexp 和 DenyToolRegexp 规则会限制 MCP 访问，但不会更改 Zabbix token 的权限。如果前端地址暴露给 AI agent，该 agent 可能会尝试直接发送 API 请求。请确保 Zabbix 前端位置和 MCP 服务器配置对 agent 不可访问，以便请求必须通过 MCP 服务器及其安全控制措施发送。

安全最佳实践　API 令牌

身份验证和授权完全由 Zabbix 处理。MCP 客户端在每个请求中发送 Zabbix API 令牌；MCP 服务器在不存储或记录该令牌的情况下将其转发，而 Zabbix 负责验证令牌、强制执行权限，并将操作记录在审计日志中。

为每个 agent 和用户使用单独的、遵循最小权限原则且会过期的令牌。将令牌存储在安全位置，并定期轮换 API 令牌。

TLS 加密

为防止令牌以明文形式发送，请设置 TLSAccept=cert，并指定证书和密钥（TLSCertFile 和 TLSKeyFile）；或者在受信任的 TLS 终止反向 proxy 后运行 MCP 服务器，并设置 TLSAccept=unencrypted。仅在受信任的网络中才可以接受未加密的 HTTP。

访问限制

将 AllowedIP 参数限制为受信任 MCP 客户端的地址，并使用 Zabbix 权限限制 API 令牌可以执行的操作。

MCP 服务器不提供请求速率限制。每个 MCP 请求通常会产生一次 Zabbix API 调用，但在循环中运行的 agent 仍可能产生较大的负载。

敏感数据保护

监控项键可能包含凭据（例如，在 SSH、JMX、ODBC 和 HTTP agent 监控项中），宏值可能包含密码和团体字符串，接口会暴露内部地址，而日志或文本监控项值可能包含任何数据。因此，在随附的配置文件中，item_get、lld_get、interface_get、macro_get 和 history_get 工具默认被拒绝。

为避免连接的 agent 的 LLM 提供商获取敏感宏值，请使用机密宏；其值永远不会由 API 返回。

工具和资源

概述

Zabbix MCP 服务器将 Zabbix 功能以工具和资源的形式提供给 MCP 客户端。

工具名称用于描述相应的活动。如果设置了ToolPrefix 参数，则此前缀会添加到每个工具名称中；工具访问规则匹配以下列出的原始名称。

读取工具

工具	Zabbix API 方法	参数	排序依据	资源
hostgroup_get	hostgroup.get	groupids、names、search、limit	名称	zabbix://hostgroup/<i
templategroup_get	templategroup.get	groupids、names、search、limit	名称	zabbix://templategrou
host_get	host.get	hostids、groupids、search、limit	名称	zabbix://host/<id>
template_get	template.get	templateids、groupids、search、limit	名称	zabbix://template/<id
item_get	item.get	hostids、itemids、search、limit	名称	zabbix://item/<id>
lld_get	discoveryrule.get	hostids、ruleids、limit	名称	-
interface_get	hostinterface.get	hostids、interfaceids、limit	接口 ID	-
macro_get	usermacro.get	hostids 或 global、macro_names、limit	宏	-
problem_get	problem.get	eventids、groupids、hostids、severities、acknowledged、suppressed、recent、limit	事件 ID，降序	-
maintenance_get	maintenance.get	maintenanceids、groupids、hostids、limit	名称	-
history_get	history.get	itemids、time_from、time_till、type、limit	时钟，降序	-
trigger_get	trigger.get	triggerids、hostids、search、limit	描述	zabbix://trigger/<id>

请注意：

- host_get 和 template_get 还会返回每个对象的触发器（字段集合有限）、标签和关联的模板。监控项、发现规则、接口和宏由相应的工具返回。
- history_get 要求时间范围不超过 7 天，且开始时间不得早于 7 天前。type 参数必须是 float、unsigned、character、text、log 或 json 之一，并且必须与监控项的值类型匹配；如果类型错误，将返回空结果，且无法将其与没有数据的监控项区分开来。无法通过 MCP 服务器检索二进制监控项（值类型 5）的历史数据。
- 读取工具请求固定的字段列表；不会请求所有字段（output: extend）。
- 包含资源的工具的结果包含 uri 字段，因此 agent 可以从列表继续定位到单个实体。

响应格式

所有读取工具均以相同格式返回响应：

```
{
  "results": [ ... ],
  "count": 2,
  "truncated": true,
  "truncated_by": "count",
  "limit": 2,
  "note": " 已达到结果数量限制 ..."
}
```

响应受 limit 参数（默认值：100，最大值：1000）以及响应序列化大小（MaxResponseSize，默认值：64K）的限制。达到任一限制时，响应将停止。响应仅会在结果之间截断，不会在结果内部截断；截断的响应会指明达到的是哪个限制。每个工具都有固定的排序顺序，因此截断的响应始终是完整结果集的开头部分。不支持分页；如需获取其他结果，请缩小查询范围。

问题工具

所有问题工具都会使用相应的操作标志调用event.acknowledge。eventids 参数为必填项；可以使用 problem_get 获取事件 ID。任何问题工具都可以添加消息（评论）。除 problem_annotate 外，message 参数均为可选；对于 problem_annotate，该参数为必填项。如果指定了该参数，消息会在同一次 Zabbix API 调用中记录。

工具	效果	其他参数
problem_acknowledge	确认问题。	-
problem_unacknowledge	移除确认。	-
problem_close	手动关闭问题（仅适用于允许手动关闭的触发器）。	-
problem_annotate	记录消息。	message（必填）
problem_change_severity	更改问题的严重性。	severity（0-5）
problem_suppress	将问题抑制至指定时间，或无限期抑制。	suppress_until
problem_unsuppress	移除抑制。	-

无法创建或删除问题；所有问题工具都会更新现有问题。

维护工具

工具	Zabbix API 方法	参数
maintenance_create	maintenance.create	name、groupids 和/或 hostids、active_since、active_till、timeperiods、description、tags、tags_evaltype
maintenance_update	maintenance.update	maintenanceid 以及上述任意参数。列表参数会替换整个值集合；空列表会清空该集合。
maintenance_delete	maintenance.delete	maintenanceids

更新维护周期时，列表参数（例如 hostids、groupids 和 timeperiods）会替换现有值，而不是将值添加到现有集合中。例如，如果某个维护周期涵盖主机 A 和主机 B，而使用仅包含主机 C 的 hostids 调用 maintenance_update，则该维护周期将仅涵盖主机 C。

maintenance_create 是唯一不具备幂等性的操作，而 maintenance_delete 是唯一无法撤销的操作。若要禁止使用这些操作，请通过DenyTool 参数拒绝这些工具。

写入工具的行为

问题和维护工具属于写入工具。其工作方式如下：

- 在调用任何 Zabbix API 之前，会先验证参数；如果验证失败，错误信息会指出未满足的约束条件。
- 如果客户端支持，只有在用户批准后才会应用更改。
- 更改由 Zabbix 使用调用方的令牌执行。Zabbix 会检查权限，并将更改以令牌所有者的身份记录在审计日志中。
- Zabbix 错误（包括权限错误）将作为包含 code、message 和 data 字段的错误对象原样返回。
- 写入工具会直接返回 Zabbix 结果；读取工具的响应限制不适用。
- MCP 服务器绝不会重试写入调用。

资源

资源使用 zabbix://<kind>/<id> URI 方案。
resources/list 响应中仅包含 zabbix://about；无法枚举实体 ID，实体 ID 可从工具的结果中获取。
resources/templates/list 响应列出了实体的种类。

URI	内容
zabbix://about	API URL、前端 URL、Zabbix 版本、调用方令牌解析到的身份，以及已公开工具的列表。
zabbix://host/<hostid>	不包含监控项、触发器、接口和宏的主机记录。
zabbix://hostgroup/<groupid>	不包含成员列表的主机组记录。
zabbix://template/<templateid>	不包含监控项、触发器和发现规则的模板记录。
zabbix://templategroup/<groupid>	不包含成员列表的模板组记录。
zabbix://item/<itemid>	不包含当前值的监控项定义。
zabbix://trigger/<triggerid>	不包含主机、依赖关系、函数和标签的触发器定义。

如果相应工具被拒绝访问，则对应资源不存在。
不存在的实体，以及令牌所有者无权访问的实体，都会报告为 *resource not found*；这些情况经过有意设计，无法相互区分。
zabbix://about 资源不受工具访问规则影响。

配置参数：**Zabbix MCP** 服务器

概述

Zabbix MCP 服务器是一个允许 AI agent 使用模型上下文协议（MCP）与 Zabbix 协作的进程。

本节列出了 Zabbix MCP 服务器配置文件（zabbix_mcp_server.conf）支持的参数。

下方列出了参数及其简要信息。单击参数可查看完整详细信息。

参数	描述
AllowedIP	MCP 客户端的 IP 地址列表（以逗号分隔，可选择使用 CIDR 表示法）或 DNS 名称列表。
AllowTool	允许某个工具的通配符规则。
AllowToolRegexp	允许工具的正则表达式规则。
DebugLevel	调试级别。
DenyTool	拒绝某个工具的通配符规则。
DenyToolRegexp	拒绝工具的正则表达式规则。
FrontendURL	Zabbix API 端点。
IgnoreURLCertErrors	指定访问 Zabbix API 时 TLS 证书验证错误的处理方式。
ListenIP	要监听的 IP 地址。
ListenPort	要监听的端口。
LogFile	日志文件的名称。
LogFileSize	日志文件的最大大小。
LogType	日志输出的类型。
MaxResponseSize	读取工具返回的响应的最大大小。
PidFile	PID 文件的名称。
Timeout	等待 Zabbix API 响应的最长时间（以秒为单位）。
TLSAccept	要接受的传入连接类型。
TLSCertFile	包含服务器证书或证书链的文件的完整路径名。
TLSKeyFile	包含服务器私钥的文件的完整路径名。
ToolPrefix	添加到所有工具名称前的前缀。

除非明确说明某个参数为必需参数，否则所有参数均为非必需参数。

请注意：

- 默认值反映的是进程默认值，而不是随附配置文件中的值；
- 值支持使用 ${NAME} 格式的环境变量；
- Zabbix 仅支持使用 UTF-8 编码且不含 BOM 的配置文件；
- 以 # 开头的注释仅支持出现在行首。

参数详细信息

AllowedIP

以逗号分隔的 IP 地址列表（可选择使用 CIDR 表示法），或 MCP 客户端的 DNS 名称。仅接受来自此处列出的主机的传入连接。

示例：

```
AllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,mcp-client.example.com
```

必填：是

AllowTool

允许工具的通配符规则。该参数可以多次指定。
 通配符模式使用监控项键语法，其中 * 匹配任意字符。该规则匹配不带ToolPrefix的工具名称。
 所有 AllowTool、DenyTool、AllowToolRegexp 和 DenyToolRegexp 规则按照指定顺序组成一个列表；由第一个匹配的规则决定。未匹配任何规则的工具将被允许。

示例：

```
AllowTool=host_get
```

AllowToolRegexp

允许工具的正则表达式规则。该参数可以多次指定。
 正则表达式使用 RE2 语法；除非使用锚点，否则会匹配纯工具名称的任意部分。

示例：

```
AllowToolRegexp=^(host|hostgroup|trigger|problem)_get$
```

DebugLevel

指定调试级别：- *0* - 有关 Zabbix 进程启动和停止的基本信息；- *1* - 关键信息；- *2* - 错误信息；- *3* - 警告；- *4* - 用于调试（生成大量信息）；- *5* - 扩展调试（生成更多信息）。

默认值：3
 范围：0-5

DenyTool

拒绝工具的通配符规则。该参数可以多次指定。
 被拒绝的工具不会包含在工具列表中，也无法调用。有关详细信息，请参阅AllowTool。

示例：

```
DenyTool=maintenance_delete
```

DenyToolRegexp

拒绝工具的正则表达式规则。该参数可以指定多次。
 有关详细信息，请参阅AllowToolRegexp。

示例：

```
DenyToolRegexp=^problem_(close|suppress)$
```

前端 URL

Zabbix API 端点：以 api_jsonrpc.php 结尾的 Zabbix 前端 URL。

示例：

```
FrontendURL=https://zabbix.example.com/api_jsonrpc.php
```

必填：是

IgnoreURLCertErrors

指定访问 Zabbix API 时 TLS 证书验证错误的处理方式：
 false - 不忽略证书错误；
 true - 忽略证书错误（仅在开发环境中使用）。

默认值：false

ListenIP

要监听的 IP 地址。

默认值：0.0.0.0

ListenPort

用于监听的端口。MCP 端点路径为 /mcp。

默认值：8443
 范围：1024-32767

LogFile

日志文件的名称。

默认值：/tmp/zabbix_mcp_server.log

必填：否

LogFileSize

日志文件的最大大小，单位为 MB。
0 - 禁用自动日志轮换。

默认值：1
 范围：0-1024

LogType

日志输出的类型：- *file* - 将日志写入 LogFile 参数指定的文件；- *system* - 将日志写入 syslog；- *console* - 将日志写入标准输出。

默认值：`file`

MaxResponseSize

读取工具返回的响应的最大大小。达到此限制或工具的 `limit` 参数中的较小值时，响应将停止。
 支持单位后缀。

默认值：64K
 范围：16K-256K

PidFile

PID 文件的名称。

默认值：`/tmp/zabbix_mcp_server.pid`

超时

等待 Zabbix API 响应的最长时间（以秒为单位）。

默认值：10
 范围：3-30

TLSAccept

接受哪些传入连接：- 未加密 - 接受未加密的连接（默认）；- 证书 - 接受使用 TLS 和证书的连接。

默认值：`unencrypted`

TLSCertFile

包含服务器证书或证书链的文件的完整路径名。

必填：是，如果将 `TLSAccept` 设置为 `cert`；否则为否

TLSKeyFile

包含服务器私钥的文件的完整路径名。

必填：是，如果 `TLSAccept` 设置为 `cert`；否则为否

ToolPrefix

添加到所有工具名称前的前缀，例如，用于避免与其他 MCP 服务器的工具发生名称冲突。例如，设置 `ToolPrefix=zbx_` 后，`host_get` 工具将以 `zbx_host_get` 的名称公开。
 允许使用的字符：字母、数字、下划线 (_)、连字符 (-) 和点号 (.)。
 工具访问规则与不带前缀的工具名称匹配。

示例：

```
ToolPrefix=zbx_
```

环境变量

概述　环境变量允许在不将值硬编码到配置文件中的情况下配置 Zabbix 组件。这使得在动态环境中（例如 Docker）更容易管理配置，因为可以在运行时传递变量，以适应不同的设置。

在最简单的情况下，只需将 Zabbix 服务器DebugLevel 配置参数的值设置为一个环境变量，然后就可以在启动时用它来配置服务器：

```
### Zabbix server 配置文件:
DebugLevel=${NEW_DEBUG_LEVEL}

### 启动 Zabbix server:
NEW_DEBUG_LEVEL=5 /usr/sbin/zabbix_server
```

以下 Zabbix 组件支持环境变量：

- Server
- Proxy
- Agent (UNIX 或Windows)
- Agent 2 (UNIX 或Windows)，包括plugins

- Web service
- Zabbix sender（使用-c, --config option 时）

重要说明

- 当某个配置参数设置为环境变量，而在运行该组件时未指定该环境变量时，将使用该参数的默认值。
- 使用运行时命令 时（例如，提高 agent 日志级别），必须指定之前使用过的任何环境变量。这是因为 Zabbix 组件会使用其配置文件来执行运行时命令；如果省略环境变量，则会使用配置参数的默认值。请参见示例。
- `userparameter_reload` 运行时命令 不支持重新加载环境变量。在重新加载期间，变量会被忽略，只会重新加载具有常规值的参数。
- 在 Zabbix 组件启动后，配置文件中使用的当前进程环境变量会被清除。这样可以确保子进程（例如，Zabbix 执行的远程脚本）无法访问这些变量。不过请注意，进程的初始变量仍然可以被检索到（例如，通过 `/proc/<PID>/environ` 文件）。

语法　环境变量必须使用以下语法：`${字母数字/下划线}`。

变量名只能包含字母（a-z、A-Z）、下划线（_）和数字（0-9），且不能以数字开头。

不符合所需语法的变量，或与普通值组合使用的变量，将被视为普通值，这可能会导致错误。

正确的变量语法：

```
DebugLevel=${NEW_DEBUG_LEVEL}
Hostname=${ZBX_HOSTNAME}
LogFile=${LogFile_001}
```

错误的变量语法：

```
DebugLevel=${5_DebugLevel}
Hostname=${ZBX.HOSTNAME 1}
LogFile=/${HOME}/zabbix/zabbix_server.log
```

> **Note:**
> 在 Windows 中，环境变量名不区分大小写。

示例　以下示例展示了如何在 Zabbix 组件中配置和使用环境变量。

示例 1：配置和测试 Zabbix agent

1. 在agent 配置文件 中设置环境变量：

```
Hostname=${ZBX_HOSTNAME}
ServerActive=${ServerActive}
```

2. 测试配置文件：

```
ZBX_HOSTNAME="New Zabbix agent" ServerActive=127.0.0.1 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agent
```

3. 使用环境变量启动 agent：

```
ZBX_HOSTNAME="New Zabbix agent" ServerActive=127.0.0.1 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agent
```

使用运行时命令时（例如，提高 agent 日志级别），必须指定之前使用的所有环境变量：

```
ZBX_HOSTNAME="New Zabbix agent" ServerActive=127.0.0.1 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agent
```

这是因为 agent 使用其配置文件执行运行时命令；如果省略环境变量，将使用配置参数的默认值。

或者，在 agent 配置文件中设置环境变量后，可以使这些变量对进程可用（例如，使用 `export` 命令）。这样可以降低因变量缺失或设置不正确而导致意外行为的风险。

```
export ZBX_HOSTNAME="New Zabbix agent"
export ServerActive=127.0.0.1
/usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf --test-config
/usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
/usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf -R log_level_increase
```

示例 2：为容器配置 Zabbix agent

如果您要为 Zabbix 组件（例如 Zabbix agent）创建和配置自定义镜像，可以使用环境变量定义配置参数，然后使用这些变量启动容器。

1. 准备镜像时，在agent 配置文件中设置环境变量：

```
Hostname=${ZBX_HOSTNAME}
BufferSize=${BUFSZ}
ListenPort=${LISTENPORT}
UserParameter=${_UsrPar01}
UserParameter=${_UsrPar02}
```

2. 构建容器镜像后，使用环境变量启动 agent 容器（例如 Docker）：

```
docker run --name my-zabbix-agent -e ZBX_HOSTNAME="new-hostname" -e BUFSZ=1000 -e LISTENPORT=20050 -e _Usr
```

3. 使用运行时命令时（例如提高 agent 日志级别），访问容器 shell 并执行运行时命令：

```
docker exec -it <containerid> sh
/usr/sbin/zabbix_agentd -R log_level_increase
```

> **Attention:**
> userparameter_reload 运行时命令不支持重新加载环境变量。重新加载期间，变量会被忽略，只有具有常规值的参数会被重新加载。

配置

概述　Zabbix 中的配置包括指定要监控哪些主机和系统、定义要收集哪些数据，以及在出现问题时如何发送通知。

请使用侧边栏导航到相关的配置项。

关键配置任务

- 设置新的主机或主机组 → 主机和主机组
- 定义要从主机收集哪些数据 → 监控项
- 创建用于检测问题的规则 → 使用触发器进行问题检测
- 配置通知的发送方式 → 事件通知
- 通过仪表板、图形和地图显示监控数据 → 可视化
- 在多个主机之间复用配置 → 模板和模板组, 开箱即用的模板
- 控制用户访问和权限 → 用户和用户组

配置最佳实践　配置选择会影响性能和可维护性。虽然每个 Zabbix 部署都不同，但这些最佳实践可以作为基础，并根据需要进行调整。

为每个监控实例单独创建主机

在 Zabbix 中，主机是一个逻辑实体，而不是物理机器或设备。
请为每个独立实例单独创建一个主机，例如虚拟机、数据库、容器或网络交换机。

这种做法可带来以下优势：

1. 监控数据更清晰，每个主机都有独立的监控项、触发器和告警通知。
2. 通过user-roles 实现更精细的用户访问控制，并符合最小权限原则。

对相似主机使用模板和批量更新

对于类似 *Network switch 1* 和 *Network switch 2* 这样的主机，您可以创建一个包含所需监控项的模板（*Network switch template*），并将这些主机分组到一个主机组（*Network switches*）中。

在添加新交换机时，您可以在 *Data Collection* > *Hosts* 中使用批量更新来关联该模板，而不是手动克隆主机。

配置主监控项和依赖监控项以最小化请求

为了尽量减少对目标实体的请求，请配置一个主监控项，在单次请求中收集数据，然后使用带有预处理的依赖监控项来提取特定值。

例如，主监控项可以收集包含多个指标的 JSON 或 XML 响应，或者执行一个返回多列的数据库查询（例如，打开的连接数、已中止的连接数、允许的最大同时连接数，以及自启动以来的累计连接总数），而依赖监控项将分别解析并存储每个值。

在收集后丢弃主监控项的历史数据，只保留依赖监控项的数据。

在需要时部署 Zabbix proxy

如果所有主机都与 Zabbix 服务器位于同一本地网络中，且不存在可扩展性或性能方面的顾虑，则可能不需要 proxy。在规模较大或更复杂的环境中，proxy 可以实现更均衡的负载分配。

在以下情况下添加 Zabbix proxy：

1. 通过防火墙使用各种指标收集方法监控多个主机。proxy 从主机收集数据并将其转发到 Zabbix 服务器，从而减少需要开放的防火墙端口数量。
2. 监控远程位置、分支机构或网络。位于远程位置的 Zabbix proxy 会在网络中断期间继续收集数据，并在连接恢复后将数据发送回来。
3. 管理大规模部署。如果您拥有大量主机和/或每秒收集大量值，请部署 proxy 以降低服务器负载并提高性能。

配置独立的 secret 宏检索

您可以将secret 用户宏用作 secret 文本或 secret vault 宏。

默认情况下，secret 宏值由 Zabbix 服务器检索并传播到 proxy。为了增强 secret vault 宏的安全性，请配置宏值由 Zabbix 服务器和 proxy 独立检索。

配置模板

概述

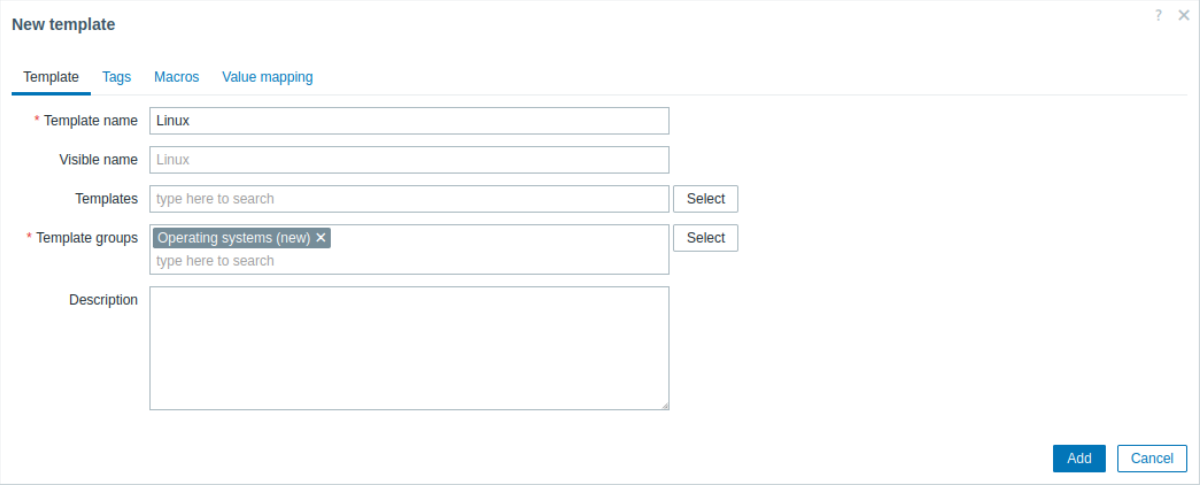
配置模板时，首先需要通过定义其常规参数来创建模板，然后再向其中添加实体（监控项、触发器、图形等）。

创建模板

要创建模板，请执行以下操作：

1. 转到 数据采集 > 模板。
2. 单击 创建模板。
3. 编辑模板属性。

模板选项卡包含常规模板属性。



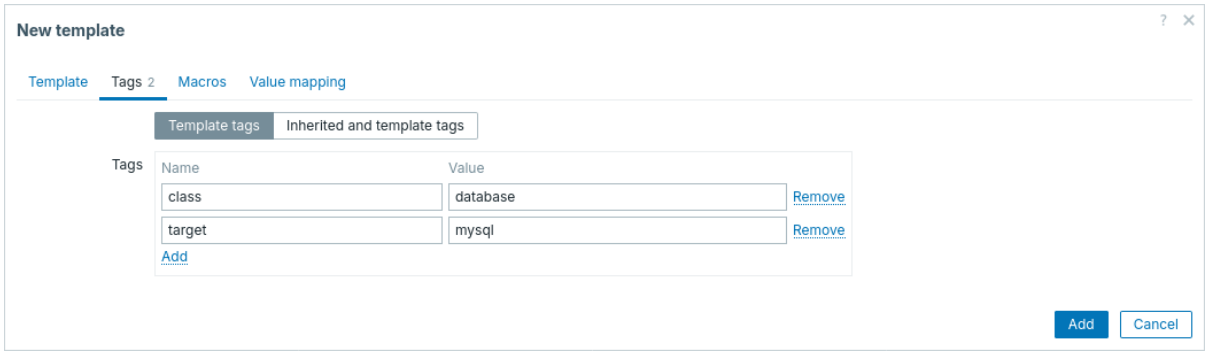
所有必填字段均以红色星号标记。

模板属性：

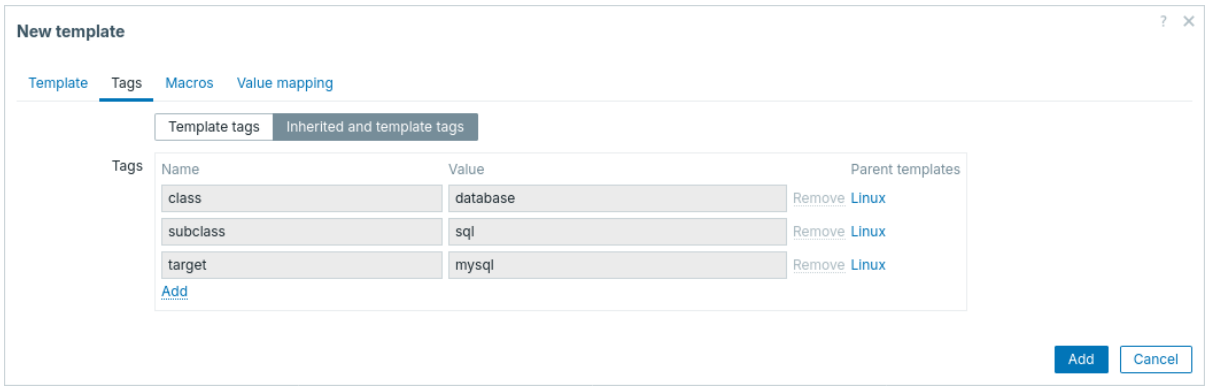
参数	描述
模板名称	唯一的模板名称。 允许使用字母数字字符、空格、点、短横线和下划线。 不允许使用开头和结尾的空格。
可见名称	如果设置此名称，它将显示在列表、拓扑图等位置。
模板	将一个或多个模板链接到此模板。所有实体（监控项、触发器等）都将从链接的模板继承。 要链接新模板，请在 模板字段中输入其名称，这将显示匹配模板的下拉列表。或者，单击 选择打开 模板弹出窗口，该窗口初始不显示任何模板。要填充列表，请在 模板组字段中输入模板组名称（并从建议列表中选择一个），或者单击 模板组字段旁的 选择按钮，打开 模板组弹出窗口。选择模板组后，模板弹出窗口将更新为显示属于该组的模板。保存或更新配置时，所选模板将被链接。 要取消链接模板，请使用 模板区块中的以下两个选项之一： 取消链接 - 取消链接模板，但保留其实体（监控项、触发器等）； 取消链接并清除 - 取消链接模板并删除其所有实体（监控项、触发器等）。
模板组	模板所属的模板组。
描述	模板描述。

参数	描述
供应商和版本	模板供应商和版本；仅在更新现有模板时显示（如果模板配置包含此信息），包括 Zabbix 提供的开箱即用模板、导入的模板或通过模板 API修改的模板。 无法在 Zabbix 前端中修改。 对于开箱即用模板，版本显示格式如下：Zabbix 的主版本号、分隔符（“-”）、修订号（每次模板发布新版本时递增，并在 Zabbix 每个主版本发布时重置）。例如：7.0-0、7.0-5、8.0-0、8.0-3。

标签选项卡允许您定义模板级别的标签。
 链接到此模板的主机的所有问题都将使用此处输入的值进行标记。



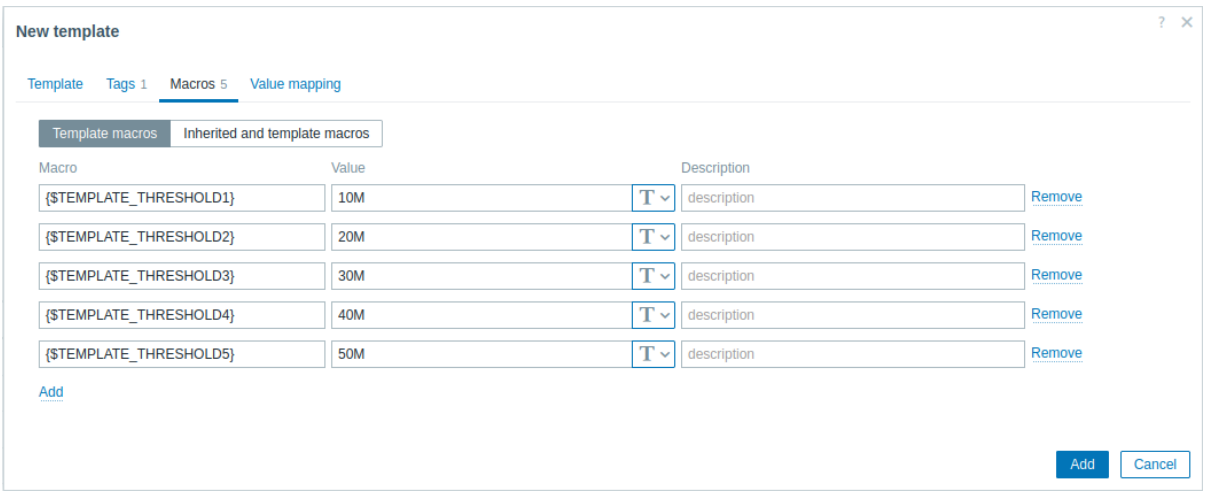
如果选择 继承的和模板标签选项，还可以查看链接模板中的标签，这些标签将由此模板继承。



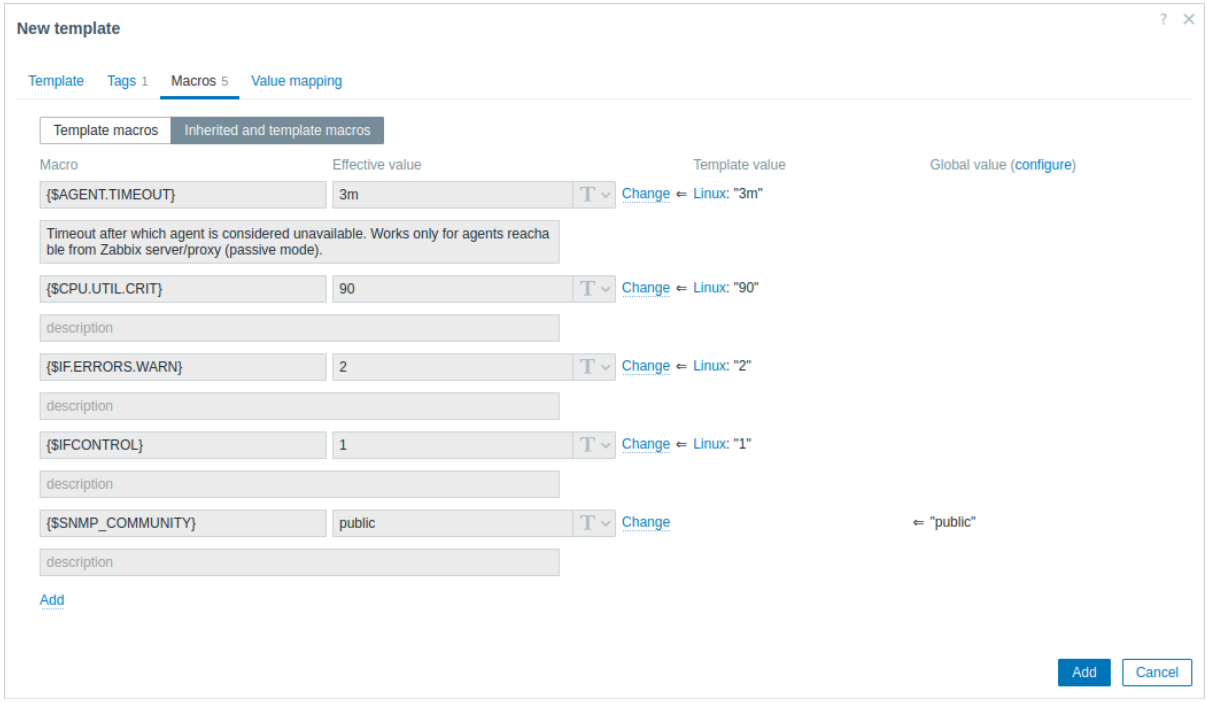
为方便起见，此处提供了指向相应模板的链接。

标签支持用户宏、{INVENTORY.*} 宏、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 宏。

宏选项卡允许您以名称-值对的形式定义模板级别的用户宏。请注意，宏值可以保存为纯文本、机密文本或 Vault 机密。还支持添加描述。



如果选择 继承的和模板宏选项，还可以查看链接模板中的宏和模板将继承的全局宏，以及宏将解析为的值。



为方便起见，此处提供了指向相应模板的链接，以及指向全局宏配置的链接。还可以在模板级别编辑链接模板宏或全局宏，从而在该模板上创建宏的副本。

值映射选项卡允许您在值映射中配置监控项数据的易读表示形式。

按钮：

Add	添加模板。添加的模板应显示在列表中。
Update	更新现有模板的属性。
Clone	根据当前模板的属性创建另一个模板。这会包含从链接模板继承的实体（监控项、触发器等）以及直接附加到当前模板的实体，但会排除当前模板的供应商和版本，以便克隆模板与原模板区分开来。
Delete	删除模板；模板的实体（监控项、触发器等）将保留在链接的主机中。
Delete and clear	删除模板及其在链接主机中的所有实体。
Cancel	取消编辑模板属性。

添加监控项、触发器、图形

> **Attention:**
> 监控项必须先添加到模板中。没有相应的监控项，无法添加触发器和图形。

您可以将现有监控项添加到模板，或创建新的监控项。

要将监控项添加到模板：

1. 转到 数据采集 > 主机（或 模板）。
2. 在所需主机/模板所在行中点击 监控项。
3. 勾选要添加到模板的监控项复选框。
4. 点击监控项列表下方的 复制。
5. 选择应将监控项复制到的模板（或模板组），然后点击 复制。
 所有选中的监控项都应复制到模板中。

添加触发器和图形的方式类似（分别从触发器和图形列表中进行），同样需要注意，只有先添加所需的监控项后，才能添加它们。

添加仪表板

要在 数据采集 > 模板中向模板添加仪表板，请执行以下操作：

1. 单击模板所在行中的 仪表板。
2. 按照配置仪表板 的指南配置仪表板。

Attention:
在模板仪表板上配置小部件时（而不是在全局仪表板上），与主机相关的参数不可用，并且某些参数的标签会有所不同。这是因为模板仪表板只显示与该模板关联的主机的数据。例如，问题 小部件中的 主机组、排除主机组和 主机参数不可用，主机可用性 小部件中的 主机组参数不可用，并且 在维护中显示主机参数已重命名为 在维护中显示数据，等等。有关模板仪表板小部件中参数可用性的更多信息，请参见各个仪表板小部件 的具体参数。

Note:
有关访问从模板仪表板创建的主机仪表板的详细信息，请参见主机仪表板 部分。

配置低级发现规则

请参阅手册中的低级发现章节。

添加 web 场景

要在 数据采集 > 模板中向模板添加 web 场景，请执行以下操作：

1. 单击模板所在行中的 *Web*。
2. 按照常规方法配置web 场景。

配置模板组

概述

模板组用于对模板进行逻辑分组，并为其分配用户权限。

每个模板必须至少分配一个模板组。一个模板可以属于多个模板组，而每个模板组也可以包含多个模板。

请注意，在 Zabbix 中，所有权限都基于组：用户组、主机组 和模板组。因此，即使某个单独用户需要访问某个单独模板，也需要先将该用户添加到用户组，然后为包含该模板的模板组分配该用户组的权限。

配置

Attention:
只有 Super admin 用户可以创建模板组。

在 Zabbix 前端中，有两种创建模板组的方式。

方式一：

1. 转到 *Data collection* > *Template groups*。
2. 单击屏幕右上角的 *Create template group*。
3. 在表单中输入组名。



方式二：在配置模板时，在 *Template groups* 输入字段中输入一个不存在的组名。

创建模板组后，您可以在 *Data collection* > *Template groups* 下的列表中单击模板名称，以编辑组名、克隆该组或删除该组。

删除模板组只会删除逻辑组，不会删除组中的模板。对于任何现有模板，如果某个模板组是其唯一所属组，则无法删除该模板组。

创建模板子组

模板子组（或嵌套模板组）是包含它的父模板组的子级。

通过在组名称输入字段中使用正斜杠 / 来表示其与父组的关系，即可创建子组。例如：

- 输入 `Linux servers/Databases` 会在父组 `Linux servers` 下创建 `Linux servers/Databases` 子组。
- 输入 `Linux servers/Databases/MySQL/Tokyo` 会在嵌套的父组 `Linux servers`、`Linux servers/Databases`、`Linux servers/Databases/MySQL` 中创建相应的子组。

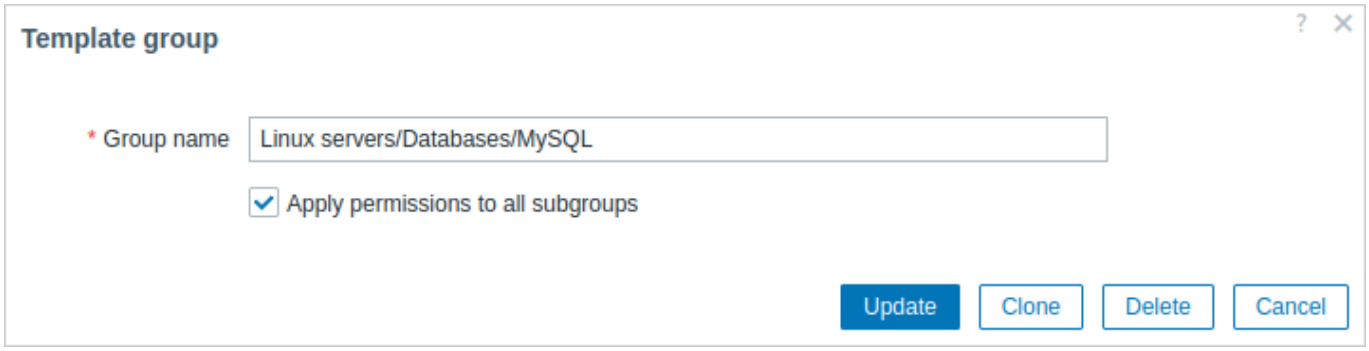
创建子组时，不允许使用前导或尾随斜杠，也不允许连续使用多个斜杠。不支持对 / 进行转义。

创建子组之前，不要求预先创建任何父模板组。您可以选择先创建子组（例如 Linux servers/Databases），也可以先创建任意父模板组（在本例中为 Linux servers）。如果您先创建子组，则不会自动创建父模板组。

模板组的权限

- 当为现有的父模板组创建子组时（例如，在 Linux servers 已存在的情况下创建 Linux servers/Databases），用户组 对该子组的权限将从父组继承。
- 当为现有的子组创建父模板组时（例如，在 Linux servers/Databases 已存在的情况下创建 Linux servers），不会为父组设置任何权限。

在编辑任何模板组时，您还可以设置一个附加选项，将权限应用于所有子组。



勾选此复选框并点击 更新后，将把相同级别的权限应用于正在编辑的模板组的所有当前和未来子组。

因此，如果某些用户组已经对正在编辑的模板组的子组授予了不同的权限，勾选该复选框后，将为所有当前和未来子组授予与正在编辑的组相同的用户权限。

请注意，此选项不会保存在数据库中，并且会覆盖现有权限。通过此选项所做的任何更改只能手动撤销。

关联/取消关联

概述

链接是将模板应用于主机的过程，而取消链接将从主机中删除与模板的关联。

链接模板

要将模板链接到主机，请执行以下操作：

1. 转到 数据采集 > 主机。
2. 单击所需的主机。
3. 在 模板字段中开始输入模板名称。系统将显示匹配的模板列表；向下滚动以进行选择。
   或者，您可以单击字段旁边的 选择，然后在弹出窗口的列表中选择一个或多个模板。
4. 在主机属性表单中单击 添加/更新。

现在，主机将拥有模板中的所有实体。
这包括监控项、触发器、图形、低级别发现规则、Web 场景以及仪表板。

Attention:
如果多个模板包含具有相同监控项键值的监控项，则无法将这些模板链接到同一主机。
此外，由于触发器和图形使用监控项，如果使用相同的监控项键值，则也无法从多个模板将它们链接到同一主机。

从模板添加实体（监控项、触发器等）时：

- 主机上之前已存在的相同实体将更新为模板的实体，并且对该实体已有的任何主机级自定义设置都将丢失；
- 模板中的实体将被添加；
- 在链接模板之前仅直接链接到主机的实体将保持不变。

在列表中，模板中的所有实体现在都会以模板名称作为前缀，表示这些实体属于特定模板。
模板名称本身（以灰色显示）是一个链接，允许您访问模板级别的这些实体列表。

Note:
对于某些监控项，例如外部检查、HTTP agent 检查、简单检查、SSH 检查和Telnet 检查，主机接口是可选的。
如果在链接模板时主机没有定义接口，则这些监控项将不带接口添加。
如果之后添加主机接口，该接口不会自动分配给已存在的监控项。
要一次性将新添加的主机接口分配给所有模板监控项，请先从主机取消链接模板，然后重新将其链接回来。
要保留监控项历史记录，请使用 取消链接选项，不要使用 取消链接并清除。

如果某个实体没有以模板名称作为前缀，则表示该实体在模板链接之前已存在于主机上，并非由模板添加。

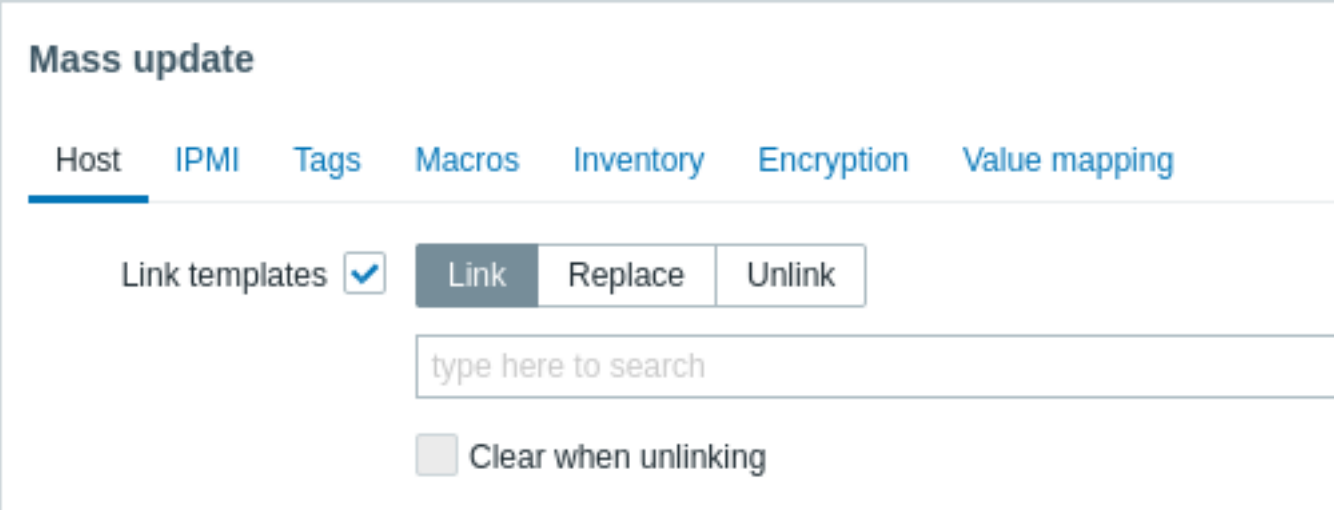
实体唯一性标准

从模板中添加实体（监控项，触发器等）时，重要的是要知道这些实体已经存在于主机上并需要更新，哪些实体有所不同。决定同一性/差异的唯一性标准是：

- 对于监控项 - 监控项键
- 对于触发器 - 触发器名称和表达式
- 对于自定义图形 - 图形名称及其监控项

将模板链接到多个主机

要更新多个主机的模板链接，请在数据采集 > 主机中选中一些主机的复选框，然后点击列表下方的批量更新，接着选择链接模板：



要链接其他模板，请在自动完成字段中开始输入模板名称，直到出现包含匹配模板的下拉列表。向下滚动以选择要链接的模板。

替换选项允许您链接新模板，同时取消链接之前已链接到这些主机的任何模板。取消链接选项允许您指定要取消链接的模板。取消链接时清除选项不仅允许您取消链接之前链接的模板，还可以删除从这些模板继承的所有实体（监控项、触发器等）。

Note:
Zabbix 提供了一组规模较大的预定义模板。您可以将其用作参考，但请注意不要在生产环境中直接使用未修改的模板，因为其中可能包含过多监控项，并且数据采集频率过高。如果您希望使用这些模板，请根据实际需求进行调整。

编辑链接实体

如果您尝试编辑从模板链接的监控项或触发器，您可能会发现许多关键选项无法进行编辑。这是因为模板的初衷是在模板级别以一键式方式编辑内容。但是，您仍然可以，例如，在单个主机上启用/禁用监控项并设置更新间隔、历史长度和一些其他参数。

如果你想完全的编辑这些实体，你必须在模板级别编辑它（模板级别的快捷方式显示在表单名称中），请记住这些更改将影响所有链接到此模板的主机。

Attention:
对在模板级别实现的实体的任何自定义将覆盖先前在主机级别对实体的自定义。

取消关联模板

要从主机取消关联模板，请执行以下操作：

1. 转到 数据采集 > 主机。
2. 单击所需的主机并找到 模板字段。
3. 单击模板旁边的 取消关联或 取消关联并清除，以取消关联该模板。
4. 在主机属性表单中单击 更新。

选择 取消关联选项只会移除与模板的关联，同时保留该模板的所有实体与主机的关联。这包括监控项、触发器、图形、低级别发现规则和 web 场景，但不包括仪表板。请注意，从已关联模板继承的值映射和标签也会被移除。

选择 取消关联并清除选项将同时移除与模板的关联以及其所有实体（监控项、触发器等）。

嵌套

概述

嵌套是一个模板包含一个或多个其它模板的一种方式。

有时，针对不同的服务，应用程序，我们会需要将模板上各种实体进行分割，因此您最终可能会得到相当多的模板，这些模板都可能需要链接到许多的主机。为了简化该操作，可以将一些模板链接到一个模板中。

嵌套的好处是您只需将一个模板（“嵌套”，父模板）链接到主机，主机将自动继承链接模板（“嵌套”，子模板）的所有实体。例如，如果我们将模板 *T1* 和 *T2* 链接到模板 *T3*，那么 *T3* 就有了所有来自于 *T1* 和 *T2*, 的实体，但反之则不然。如果我们将模板 *T1* 链接到模板 *T2* 和 *T3*，那么 *T2* 和 *T3* 就有了所有来自于 *T1* 的实体。

配置嵌套模板

要链接模板，你需要先使用现有模板（或创建一个新模板），然后：

1. 打开模板配置表单。
2. 找到 *Templates* 字段。
3. 点击 *Select* 打开 *Templates* 弹出窗口。
4. 在弹出窗口中，选择所需的模板，然后点击 *Select* 将模板添加到列表中。
5. 在模板配置表单中点击 *Add* 或 *Update*。

这样，已配置模板的所有实体，以及所有已链接模板的实体，都会显示在模板配置中。这包括监控项、触发器、图形、低级别发现规则和 web 场景，但不包括仪表板。不过，已链接模板的仪表板仍然会被主机继承。

要取消链接任一已链接模板，请在模板配置表单中点击 *Unlink* 或 *Unlink and clear*，然后点击 *Update*。

Unlink 选项只会移除与已链接模板的关联，而不会删除其任何实体（监控项、触发器等）。

Unlink and clear 选项会同时移除与已链接模板的关联，以及其所有实体（监控项、触发器等）。

批量更新

概览

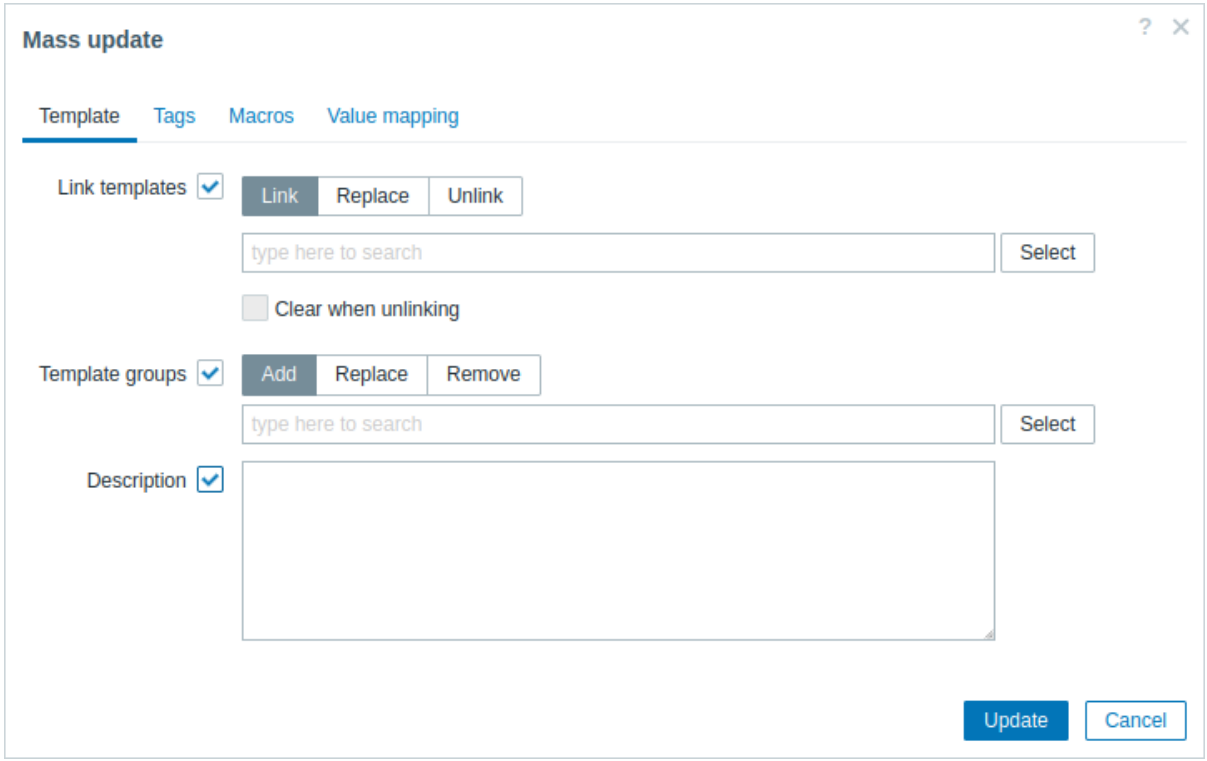
有时您可能想要同时更改多个模板的某些属性。您可以使用批量更新功能，无需打开每个模板进行独立编辑。

使用批量更新

要批量更新某些模板，请执行以下操作：

1. 勾选您要更新的模板列表 中对应模板前面的复选框。
2. 单击列表下方的 批量更新。
3. 切换到包含所需属性的选项卡（模板、标签、宏或 值映射）。
4. 勾选要更新的任意属性对应的复选框，并为其输入新值。

模板选项卡包含模板批量更新选项。



为链接模板更新选择相应按钮时，可用以下选项：

- 链接 - 指定要链接的其他模板；
- 替换 - 指定要链接的模板，同时取消链接之前已链接的所有模板；
- 取消链接 - 指定要取消链接的模板。

要指定要链接或取消链接的模板，请在自动完成字段中开始输入模板名称，直到出现下拉列表，其中会显示匹配的模板。向下滚动以选择所需的模板。

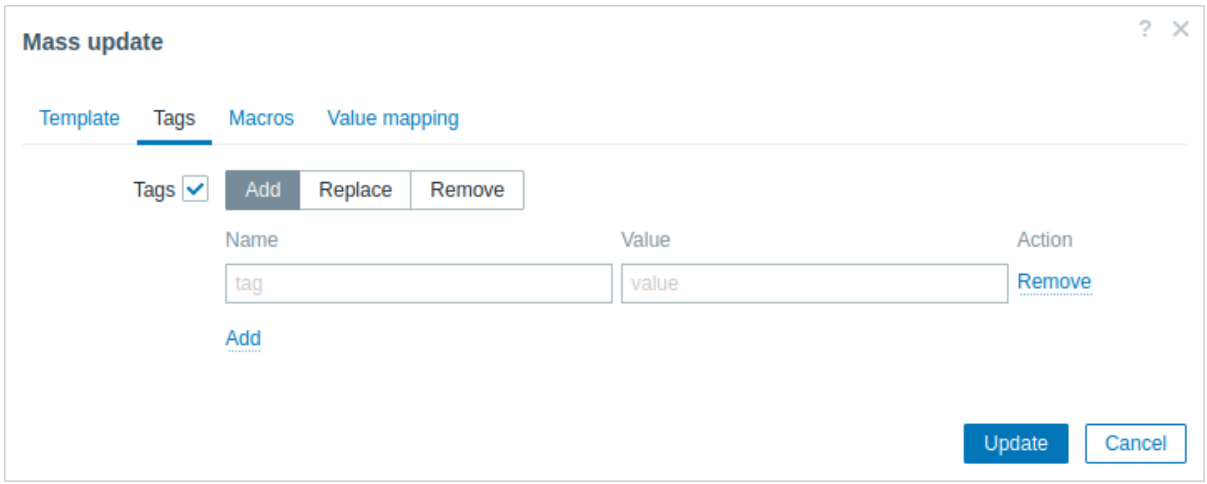
取消链接时清除选项不仅允许您取消链接之前已链接的模板，还可以删除从这些模板继承的所有实体（监控项、触发器、图形等）。

为模板组更新选择相应按钮时，可用以下选项：

- 添加 - 从现有模板组中指定其他模板组，或为模板输入全新的模板组；
- 替换 - 从所有现有模板组中移除模板，并将其替换为此字段中指定的模板组（现有模板组或新模板组）；
- 移除 - 从模板中移除指定的模板组。

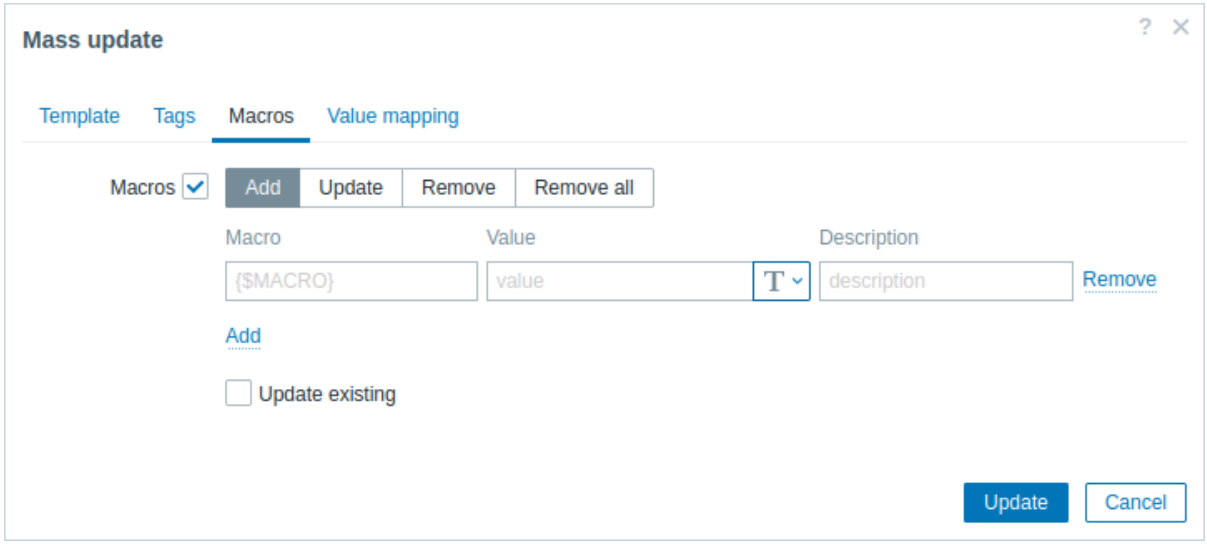
这些字段支持自动完成：开始输入内容后，会显示包含匹配模板组的下拉列表。如果模板组是新的，它也会出现在下拉列表中，并在字符串后标记为 *(新建)*。向下滚动以进行选择。

标签选项卡允许你批量更新模板级别的标签



用户宏，{INVENTORY.*} 宏, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} 和 {HOST.ID} 宏在标签中被支持。注意：具有相同名称但不同值的标签不被认为是“重复”，可以添加到相同的模板中。

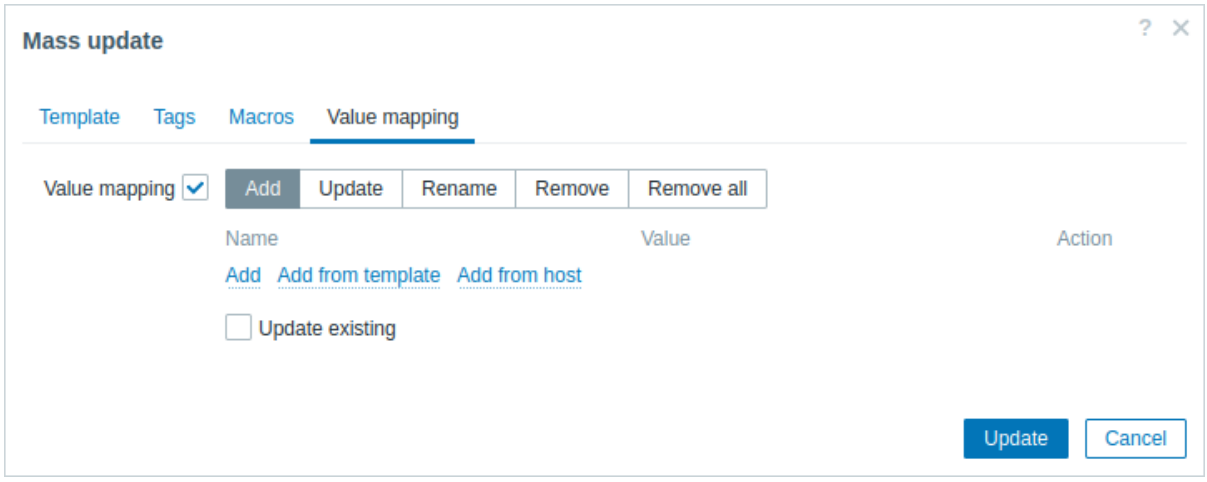
宏选项卡允许批量更新模板级宏。



选择相应的宏更新按钮时，可用以下选项：

- 添加 - 为模板指定其他用户宏。如果选中更新现有宏复选框，则会更新指定宏名称的值、类型和描述。如果未选中该复选框，并且模板中已存在同名宏，则不会更新该宏。
- 更新 - 替换此列表中指定宏的值、类型和描述。如果选中添加缺失宏复选框，则会将模板中之前不存在的宏作为新宏添加。如果未选中该复选框，则只会更新模板中已存在的宏。
- 移除 - 从模板中移除指定的宏。如果选中排除所选项复选框，则会移除列表中未指定的所有宏。如果未选中该复选框，则只会移除列表中指定的宏。
- 全部移除 - 从模板中移除所有用户宏。如果未选中我确认移除所有宏复选框，则会打开新的弹出窗口，要求确认移除所有宏。

值映射选项卡允许批量更新值映射。



值映射更新可使用以下选项：

- 添加 - 向模板添加值映射。如果勾选更新现有值映射，则会更新具有此名称的值映射的所有属性。否则，如果已存在同名值映射，则不会更新该值映射。
- 更新 - 更新现有值映射。如果勾选添加缺失值映射，则会将模板中之前不存在的值映射作为新值映射添加。否则，仅更新模板中已存在的值映射。
- 重命名 - 为现有值映射指定新名称。
- 移除 - 从模板中移除指定的值映射。如果勾选排除选定项，则会移除所有值映射，但指定的值映射除外。
- 全部移除 - 从模板中移除所有值映射。如果未勾选我确认移除所有值映射复选框，则会打开新的弹出窗口，要求确认移除操作。

值映射添加/更新操作提供了从模板添加和从主机添加选项。
这些选项分别允许您从模板或主机中选择值映射。

完成所有必需的更改后，单击 *Update*。这些属性将相应地更新到所有选定的模板中。

主机和主机组

什么是“主机”？

在 Zabbix 中，“主机” 是指任何物理或虚拟设备、应用程序、服务，或任何其他在逻辑上相关的受监控参数集合。

创建主机是 Zabbix 中最初的监控任务之一。例如，如果您想监控服务器 “x” 上的某些参数，您必须先创建一个名为 “Server X” 的主机，然后才能为其添加监控项。

主机被组织到主机群组中。

继续阅读：

- 主机向导，了解用于创建和配置新的或现有主机的引导式分步界面。
- 配置主机，了解创建和配置主机的经典方法。

主机向导

概述

Host Wizard 是一个引导式、逐步操作的界面，用于在 Zabbix 中设置您的监控目标（设备、应用程序、服务等）。它会引导您完成以下步骤：

- 选择模板
- 创建或选择主机
- 安装 Zabbix agent 或 agent 2
- 添加主机接口
- 将附加配置应用到您的监控目标或 Zabbix 主机（如果模板要求）

要在 Zabbix 前端中访问 Host Wizard，请执行以下操作：

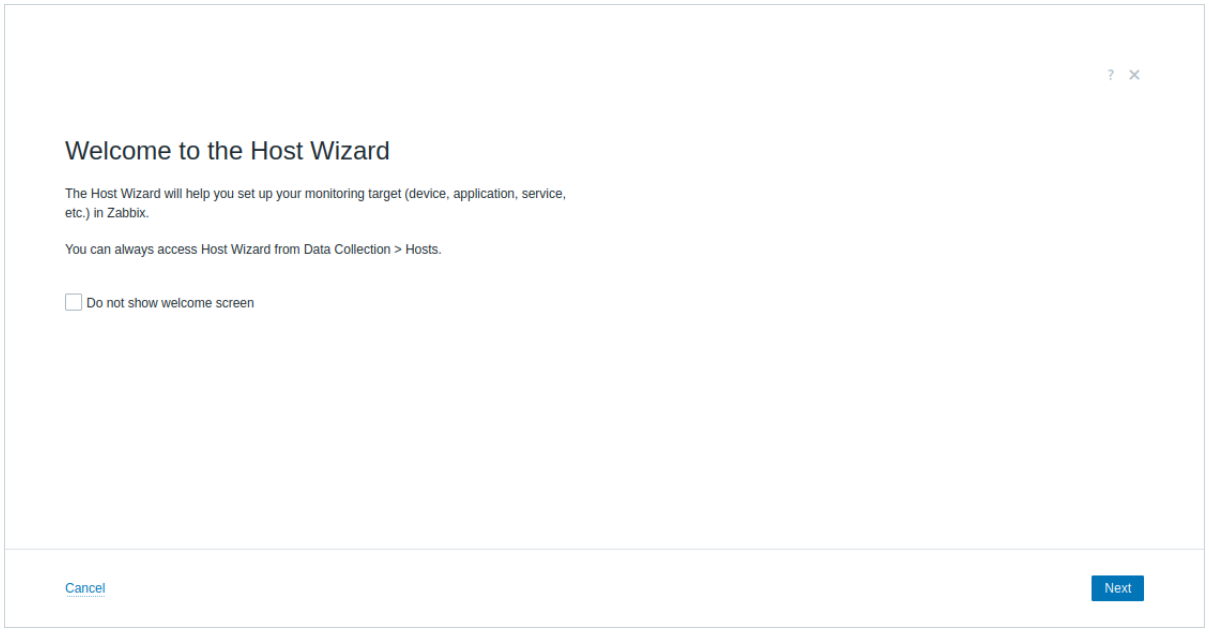
1. 转到：*Data collection* > *Hosts*。
2. 点击屏幕右上角的 *Host Wizard*。
3. 按照向导中的说明进行操作。

您也可以通过在前端的各个部分中单击现有主机的名称（或其旁边的三点图标），从主机菜单 访问 Host Wizard 来编辑该主机。请注意，如果现有主机已经包含所选模板所需的配置，则在编辑该主机时会跳过某些步骤。

另外，您也可以使用经典方式来创建和配置主机。

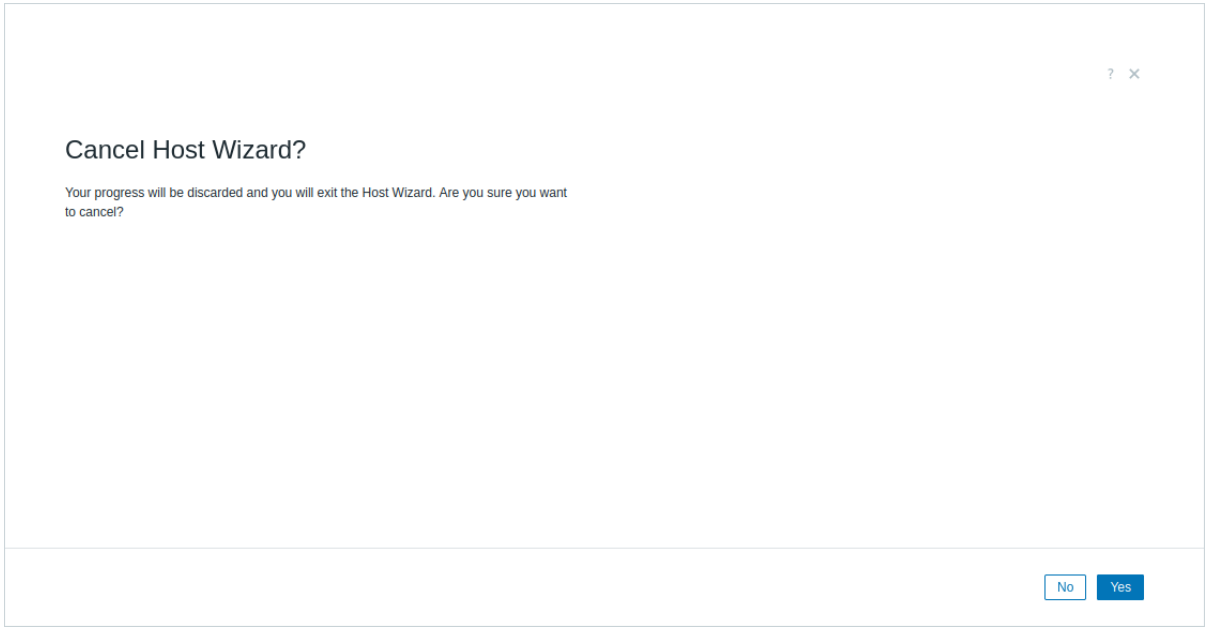
欢迎使用主机向导

当您启动主机向导时，会显示欢迎界面：



如果您希望在今后的会话中跳过此界面，可以勾选 不显示欢迎界面复选框，然后点击 下一步。

如果您在主机向导的后续步骤中开始配置设置，并尝试通过点击关闭图标或按 Esc 关闭向导，将会显示确认界面：



点击 是将退出主机向导，而不会保存您的进度。点击 否、Esc 或关闭图标将返回上一步。

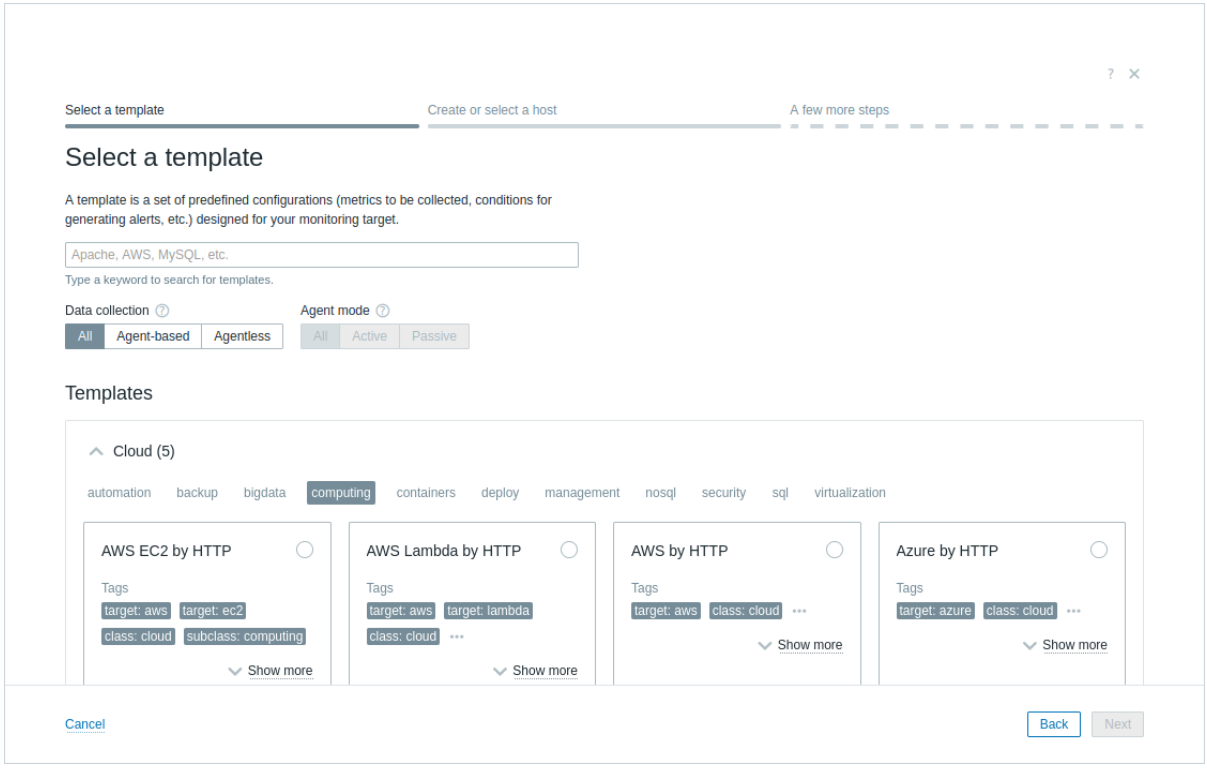
选择模板

设置监控目标的第一步是选择一个模板。模板是一组预定义配置（要收集的指标、生成告警的条件等），专为您的监控目标设计。

Zabbix开箱即用模板为操作系统、应用程序、数据库、网络设备、服务等提供了各种预定义的监控配置。

在此步骤中，您可以：

- 按类别（云、数据库、网络等）和子类别（例如自动化、备份）浏览模板，这些类别和子类别基于模板标签。
- 按模板名称中的关键字，或模板标签的名称和值搜索模板。
- 按数据采集方式筛选模板（基于 agent 或无 agent；基于 agent 的模板至少包含一个Zabbix agent 监控项）。
- 按 agent 模式筛选模板（主动或被动；主动模板至少包含一个 Zabbix agent（主动式）监控项，而被动模板至少包含一个被动式监控项；有关详细信息，请参阅被动和主动 agent 检查）。
- 使用主机向导配置现有主机时，还可以按已链接到该主机的模板进行筛选。



Attention:
选择模板时，您可能会看到以下消息：某些模板 *(n)* 与主机向导不兼容。了解如何更新这些模板。不支持自定义模板。这表示某些开箱即用模板尚不兼容主机向导，需要进行升级；请参阅模板升级说明。

您选择的模板将决定主机向导中的后续步骤。例如，如果您选择使用 Zabbix agent 进行数据采集的 *MySQL by Zabbix agent* 模板，向导将引导您完成 agent 安装过程。

创建或选择主机

模板必须关联到一个主机，主机是 Zabbix 中代表监控目标的实体。关联后，主机会接收模板中的所有实体，例如监控项（要收集的指标）和触发器（生成告警的条件），这些实体是执行监控所必需的。

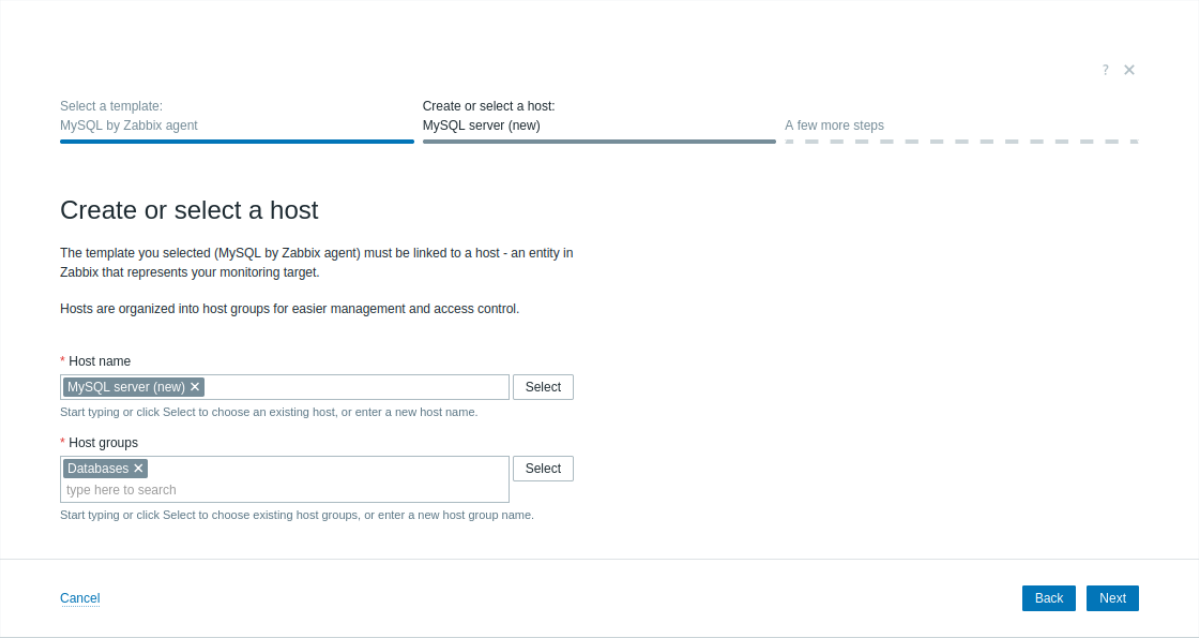
每个主机还必须至少属于一个主机组，主机组用于组织主机以及为用户分配主机访问权限。

在此步骤中，您可以：

- 输入主机和主机组的名称，创建新的主机和主机组。
- 输入主机名称创建新主机，并选择一个现有主机组，将新主机分配到该主机组。
- 选择现有主机；您还可以将该主机分配到其他主机组，而不会将其从当前所属的任何主机组中移除。

Note:
选择现有主机不会移除或覆盖其当前配置，除非后续步骤中明确说明。无法选择已发现的主机。

例如，您可以创建一个名为 *MySQL server* 的新主机，用于表示要监控的本地已安装 MySQL 服务器。之前选择的 *MySQL by Zabbix agent* 模板将关联到此主机。此外，您可以选择现有的 *Databases* 主机组（或创建一个新的主机组，例如 *MySQL servers*），以便将该主机与其他数据库相关主机进行组织，并在以后简化权限管理。



安装 Zabbix agent

基于 agent 的模板（例如 *MySQL by Zabbix agent*）要求安装 Zabbix agent 或agent 2——这是部署在监控目标上的进程，用于主动监控本地资源和应用程序（如果模板有此要求）。

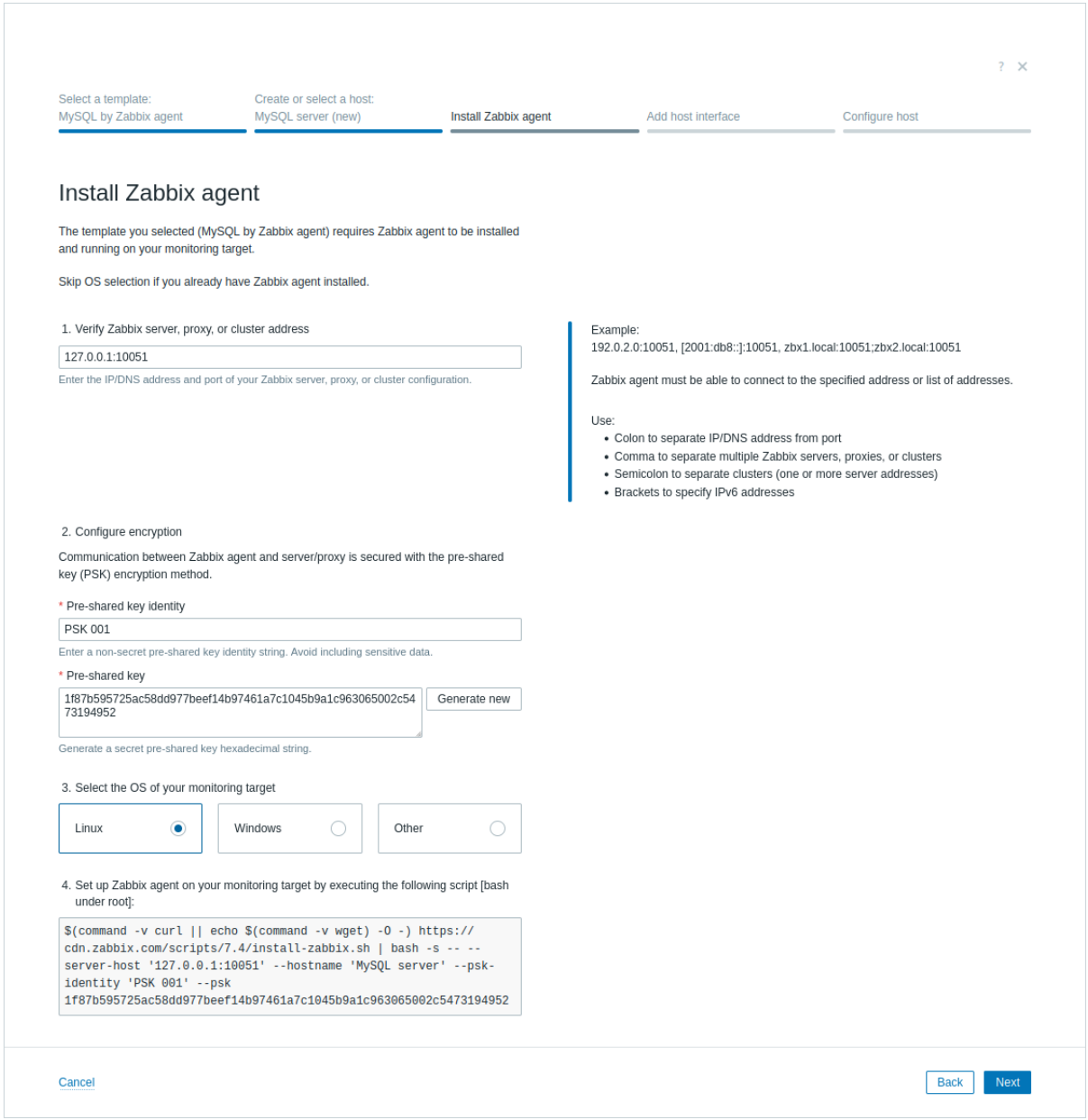
在此步骤中：

- 验证 Zabbix 服务器、proxy 或集群地址（例如 192.0.2.1:10051）。此值将用于配置 Zabbix agent 的Server 和ServerActive 参数。
- 输入新的、唯一且非机密的预共享密钥标识（例如 *PSK 001* 或 *mysql-agent-psk1*），并生成预共享密钥。这些值将用于为 Zabbix 主机（例如 *MySQL server*）和 Zabbix agent 配置预共享密钥 (PSK)加密。

Attention:
编辑现有主机时，PSK 配置将覆盖该主机上现有的所有加密设置。

- 选择承载监控目标的计算机的操作系统。
- 通过在该系统上运行提供的脚本，或按照安装说明，在该系统上安装 Zabbix agent。

例如，您可以将预共享密钥标识设置为 *PSK 001*，并生成新的预共享密钥。然后，您可以选择 *Linux* 作为操作系统，并在该系统上运行提供的脚本。安装 Zabbix agent 后，返回主机向导。

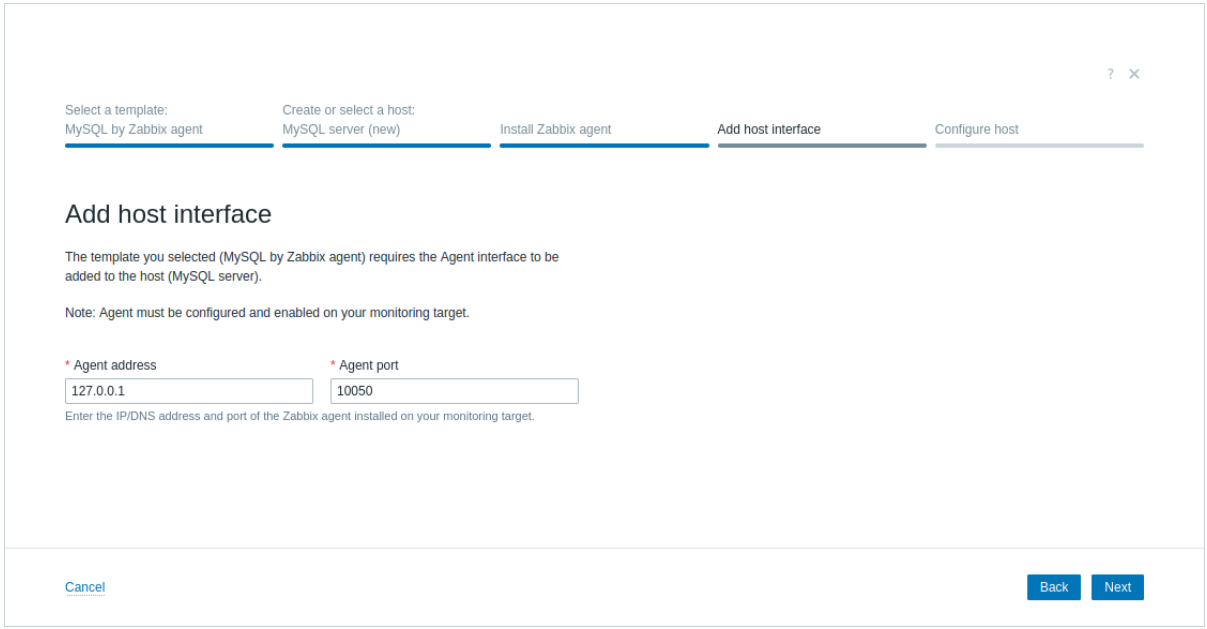


添加主机接口

主机接口定义了 Zabbix 服务器如何通过网络连接到您的监控目标。它定义了连接参数，例如 IP 地址、DNS 名称、端口和接口类型（Agent、SNMP、JMX 或 IPMI），具体取决于所选模板所需的数据采集方法。

在此步骤中，请输入所选模板所需的主机接口详细信息。

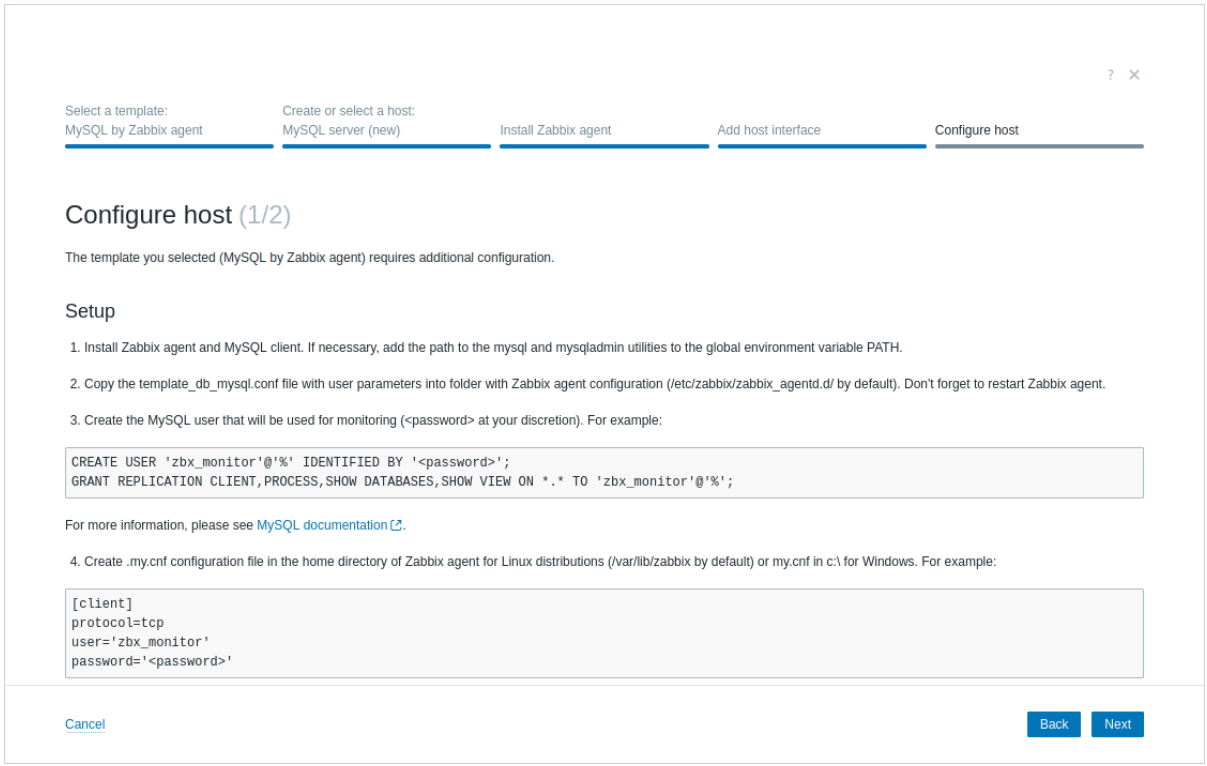
例如，如果 Zabbix 服务器、Zabbix agent 和 MySQL 服务器都在同一台机器上本地运行，您可以使用默认的 agent 地址（127.0.0.1）和端口（10050）。



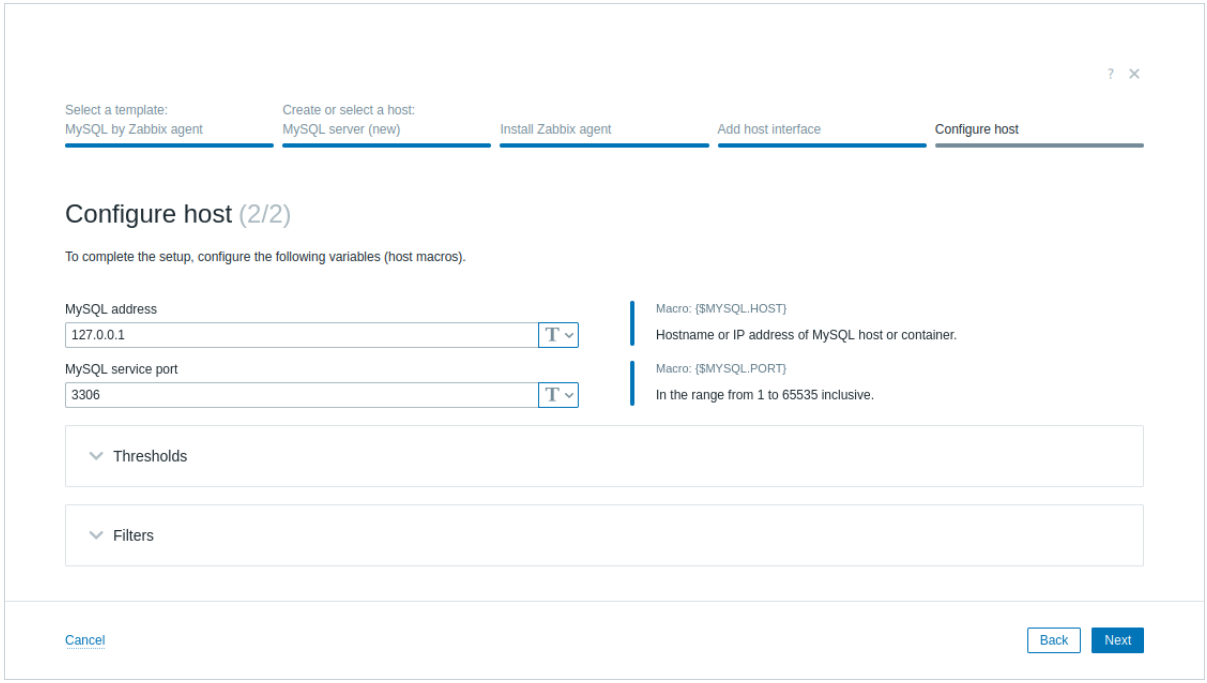
配置主机

某些模板需要在创建主机之前进行额外配置。这些配置可能包括：

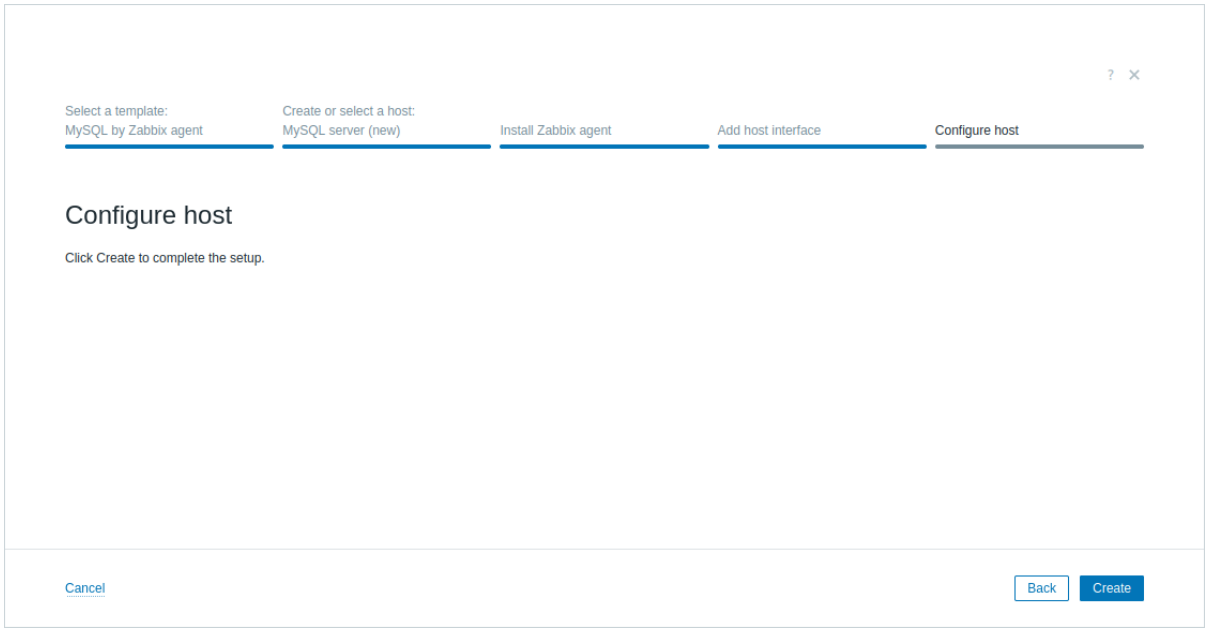
- 手动配置监控目标（例如，启用特定服务、创建服务用户或授予权限）：



- 定义主机宏 - 用于控制监控项行为、连接设置、身份验证凭据和其他参数的变量：



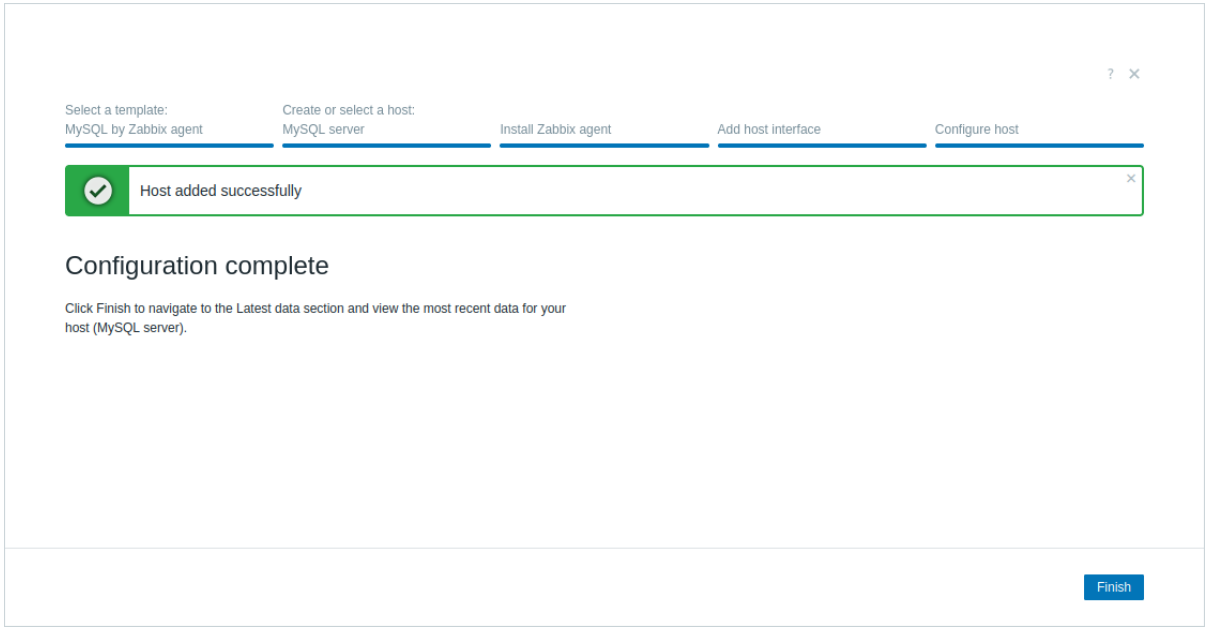
完成所有必需的配置步骤后，单击 创建将主机添加到 Zabbix（或单击 更新以更新现有主机的配置）：



配置完成

此时，Zabbix 已经开始监控您的目标。

对于新的主机，单击 完成以导航到最新数据 部分，并查看您的主机的最新数据。
对于现有主机，单击 完成以关闭主机向导并返回到打开它的界面。



配置主机

概述

要在 Zabbix 前端中配置主机，请执行以下操作：

1. 转到 数据采集 > 主机或 监控 > 主机。
2. 单击屏幕右上角的 创建主机（或单击主机名称以编辑现有主机）。
3. 在表单中输入主机参数。

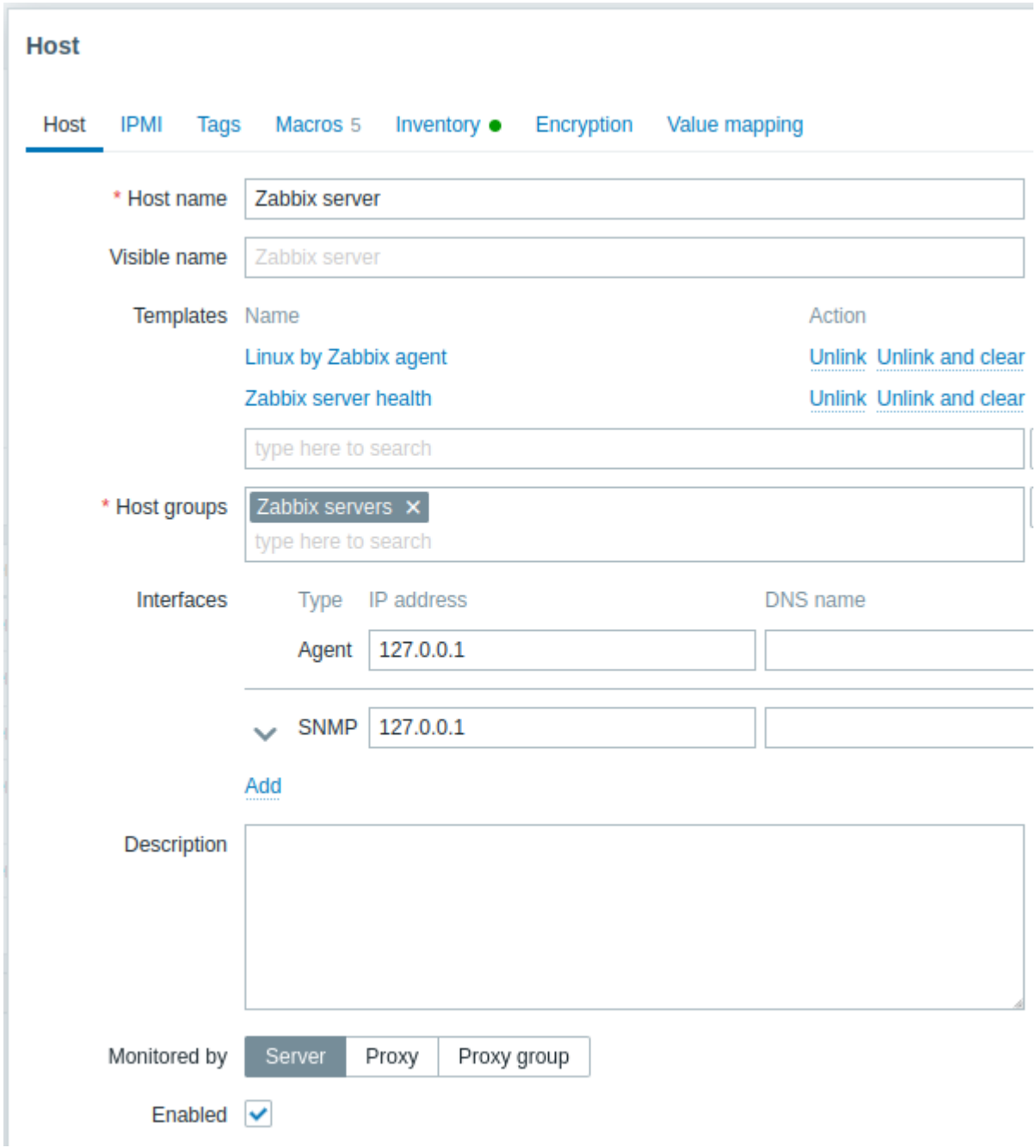
您也可以在现有主机的配置表单中使用 克隆按钮来创建新主机。
该主机将具有现有主机的所有属性，包括关联的模板、这些模板中的实体（监控项、触发器等），以及直接附加到现有主机的实体。

请注意，当主机被克隆时，它将保留所有模板实体，并保持其在模板中的原始状态。
对这些实体在现有主机级别所做的任何更改（例如更改监控项间隔、修改正则表达式或向低级别发现规则添加原型）都不会克隆到新主机；新主机中的这些内容将与模板中的一致。

或者，您也可以使用主机向导 通过引导式的分步界面来配置主机。

配置

主机选项卡包含常规主机属性：



所有必填输入字段均以红色星号标记。

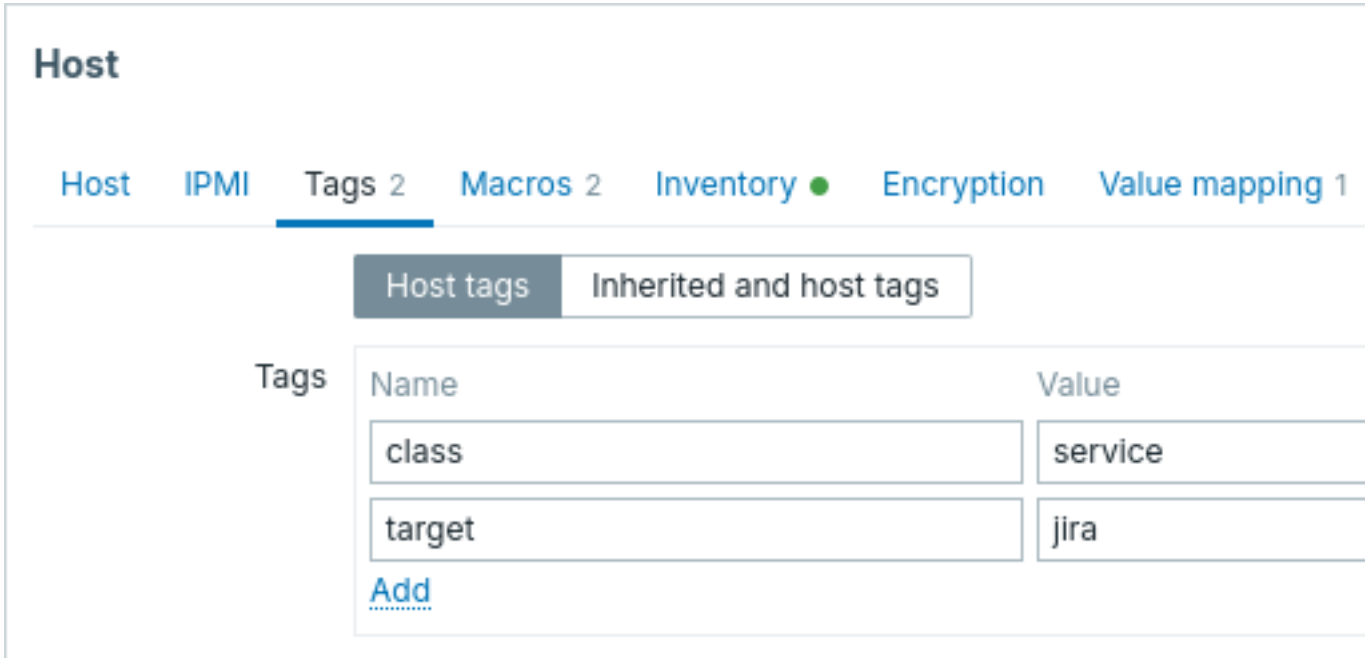
参数	描述
主机名	输入唯一的主机名。允许使用字母数字字符、空格、点、短横线和下划线。但不允许使用开头和结尾空格。 注意：如果您要配置的主机上运行着 Zabbix agent，则agent 配置文件中的 *Hostname* 参数必须与此处输入的主机名相同。处理主动检查时需要使用该参数中的名称。
可见名称	输入主机的唯一可见名称。如果设置了此名称，则列表、映射等位置将显示该名称，而不是技术主机名。此属性支持 UTF-8。
模板	将模板链接到主机。所有实体（监控项、触发器等）都将从模板继承。 要链接新模板，请在文本输入字段中开始输入模板名称。系统将显示匹配的模板列表；向下滚动以进行选择。或者，可以单击字段旁边的选择，然后在弹出窗口的列表中选择模板。保存或更新主机配置表单时，所有选定的模板都将链接到主机。 要取消链接模板，请使用已链接的模板区域中的以下两个选项之一： 取消链接 - 取消链接模板，但保留其实体（监控项、触发器等）； 取消链接并清除 - 取消链接模板并删除其所有实体（监控项、触发器等）。 列出的模板名称是可点击的链接，可跳转到模板配置表单。

参数		描述
主机群组		选择主机所属的主机群组。主机必须至少属于一个主机群组。通过添加不存在的群组名称，可以创建新群组并将其链接到主机。
接口		主机支持多种主机接口类型：*Agent*、*SNMP*、*JMX* 和 *IPMI*。 默认情况下未定义接口。要添加新接口，请在接口区域中单击添加，选择接口类型并输入 *IP/DNS*、连接到和端口信息。 注意：任何监控项正在使用的接口都无法删除，其删除链接将显示为灰色。 SNMP 接口中的 “IP” 或 “DNS” 也用于SNMP 陷阱。 匹配时，仅使用主机接口中选定的 “IP” 或 “DNS”。 有关配置 SNMP 接口（v1、v2 和 v3）的更多详细信息，请参阅配置 SNMP 监控。
	IP 地址	主机 IP 地址（可选）。
	DNS 名称	主机 DNS 名称（可选）。
	连接到	单击相应按钮将告知 Zabbix 服务器从 agent 获取数据时使用什么信息： **IP** - 连接到主机 IP 地址（推荐） **DNS** - 连接到主机 DNS 名称
	端口	TCP/UDP 端口号。默认值为：Zabbix agent 使用 10050，SNMP agent 使用 161，JMX 使用 12345，IPMI 使用 623。
	默认	选中单选按钮以设置默认接口。
描述		输入主机描述。
监控方		选择监控主机的对象： 服务器 - 主机由 Zabbix 服务器监控； **Proxy** - 主机由独立 proxy 监控； **Proxy** 组 - 主机由 proxy 组监控。
Proxy		显示已分配的 proxy 名称（仅当 Zabbix 服务器已从选定的 proxy 组中为其分配 proxy 时）。 仅当主机由 proxy 组监控时才会显示此字段。
启用		选中复选框后，主机将启用并准备进行监控。 取消选中复选框后，主机将禁用并停止监控： 对于由 Zabbix 服务器/proxy 发起的被动数据请求（例如，Zabbix agent、SNMP agent、简单检查），配置同步后将禁用监控。与主机关联的触发器和动作也只有在重新加载配置后才会禁用。 对于 Zabbix agent 的主动检查，在 Zabbix agent 收到主机已禁用的信息后的时间范围内（约 5 秒）停止监控。在这段短暂时间内，主机仍会在本地收集主动检查的数据，并尝试将其发送到服务器/proxy；但是，由于主机被标记为已禁用，服务器/proxy 将拒绝这些数据。 禁用主机时，其监控项会立即从历史缓存中移除（其最后值除外，最后值会为日志保留）。

IPMI 选项卡包含 IPMI 管理属性。

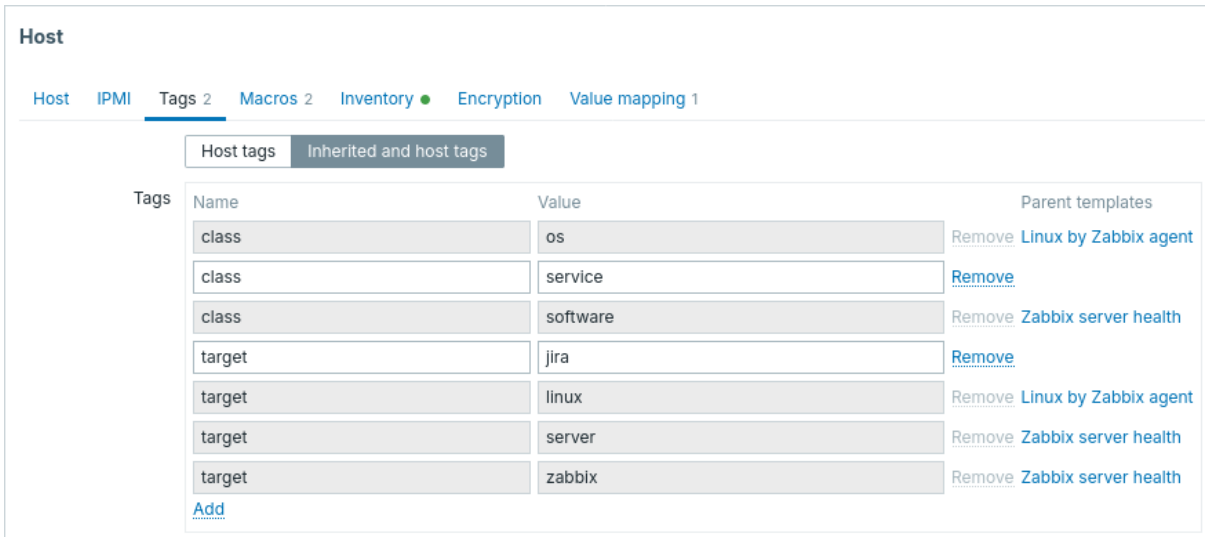
参数	描述
认证算法	选择认证算法。
权限级别	选择权限级别。
用户名	用于认证的用户名。可以使用用户宏。
密码	用于认证的密码。可以使用用户宏。

标签选项卡允许您定义主机级别的标签。该主机的所有问题都将使用此处输入的值进行标记。



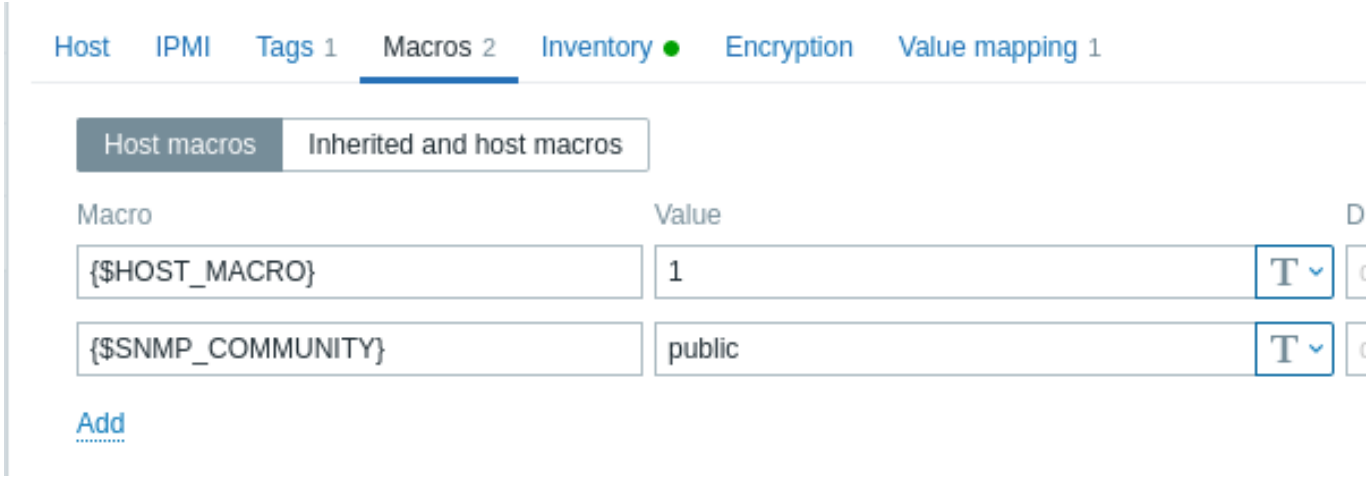
标签支持用户宏、{INVENTORY.*} 宏、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 宏。

如果选择继承的和主机标签选项，还可以在此处查看模板级别的标签。此处将显示为主机定义的所有用户标签及其来源。

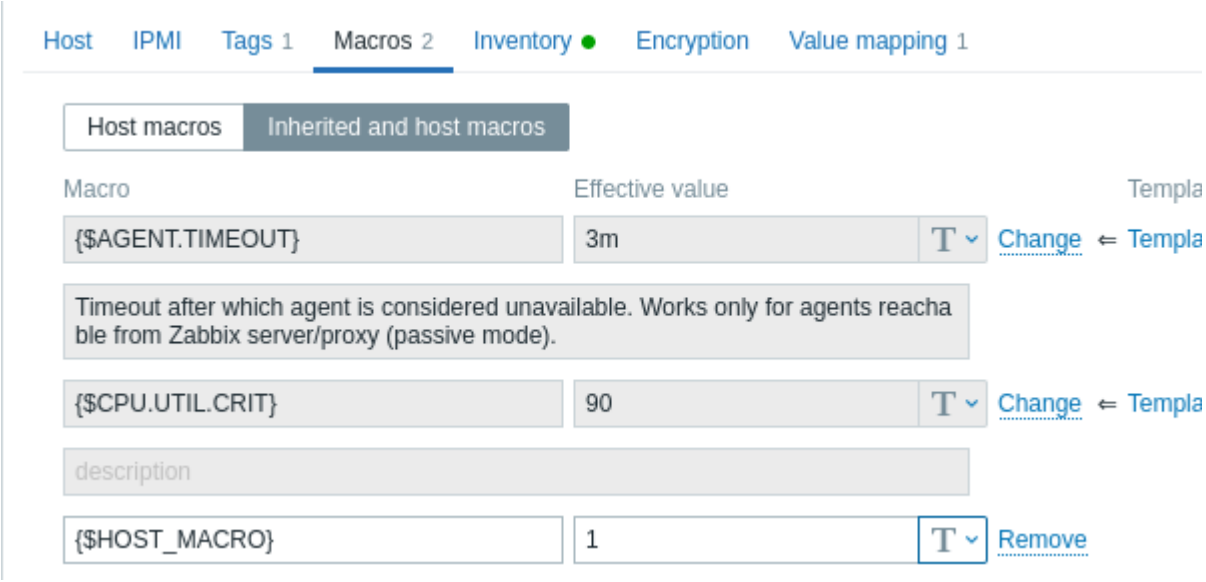


为方便起见，系统提供了指向相应模板的链接。

宏选项卡允许您以名称-值对的形式定义主机级别的用户宏。请注意，宏值可以保存为纯文本、机密文本或 Vault 机密。同时支持添加描述。

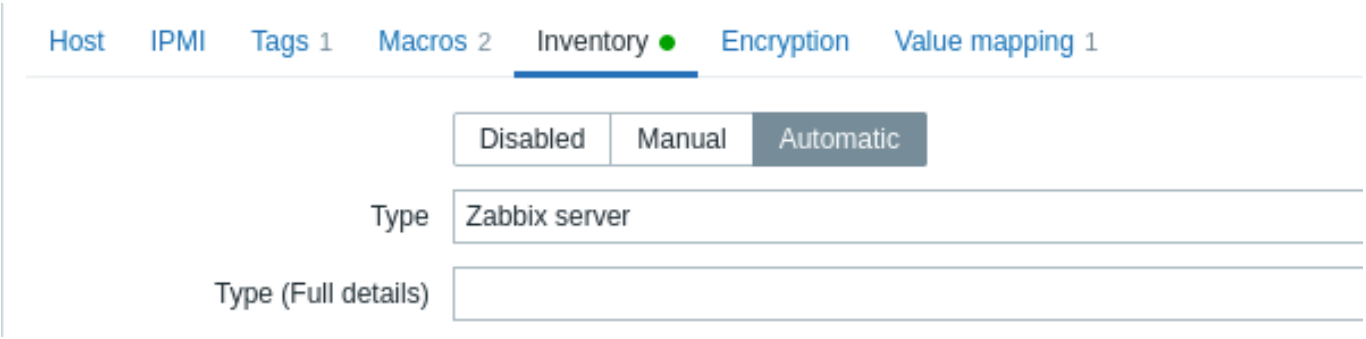


如果选择继承的和主机宏选项，还可以在此处查看模板级别和全局用户宏。此处将显示为主机定义的所有用户宏、其解析后的值以及来源。



为方便起见，系统提供了指向相应模板和全局宏配置的链接。也可以在主机级别编辑模板/全局宏，这实际上会在主机上创建该宏的副本。

资产清单选项卡允许您手动输入主机的资产清单信息。您还可以选择启用自动填充资产清单，或禁用此主机的资产清单填充。



如果启用了资产清单（手动或自动），选项卡名称旁会显示一个绿色圆点。

加密

加密选项卡允许您要求与主机建立加密连接。

Parameter	Description
Connections to host *Connections from host*	服务器或 proxy 如何连接到主机上的 Zabbix agent：不加密（默认）、使用 PSK（预共享密钥）或证书。 选择允许从主机发起的连接类型（即来自 Zabbix agent 和 Zabbix sender 的连接）。可以同时选择多种连接类型（适用于测试以及切换到其他连接类型）。默认值为 “不加密”。
Issuer	允许的证书颁发者。证书首先会通过 CA（证书颁发机构）验证。如果证书有效且由 CA 签名，则可使用 *Issuer* 字段进一步限制允许的 CA。此字段适用于您的 Zabbix 安装使用来自多个 CA 的证书的情况。如果此字段为空，则接受任何 CA。
Subject	允许的证书主题。证书首先会通过 CA 验证。如果证书有效且由 CA 签名，则可使用 *Subject* 字段仅允许一个 *Subject* 字符串值。如果此字段为空，则接受任何由已配置 CA 签名的有效证书。
PSK identity	预共享密钥标识字符串。 不要在 PSK identity 中放入敏感信息，因为它会以未加密方式通过网络传输，以告知接收方应使用哪个 PSK。
PSK	预共享密钥（十六进制字符串）。最大长度：如果 Zabbix 使用 GnuTLS 或 OpenSSL 库，则为 512 个十六进制数字（256 字节 PSK）；如果 Zabbix 使用 mbed TLS（PolarSSL）库，则为 64 个十六进制数字（32 字节 PSK）。示例： 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

值映射

值映射选项卡允许您在值映射中配置监控项数据的易读表示形式。

配置主机组

概述

主机组用于对主机进行逻辑分组，并为其分配用户权限。

每个主机必须至少分配一个主机组。一个主机可以属于多个主机组，而每个主机组也可以包含多个主机。

请注意，在 Zabbix 中，所有权限都基于组：用户组、主机组和模板组。因此，即使某个单独用户只需要访问一台主机，也需要先将该用户添加到一个用户组中，然后为包含该主机的主机组分配该用户组的权限。

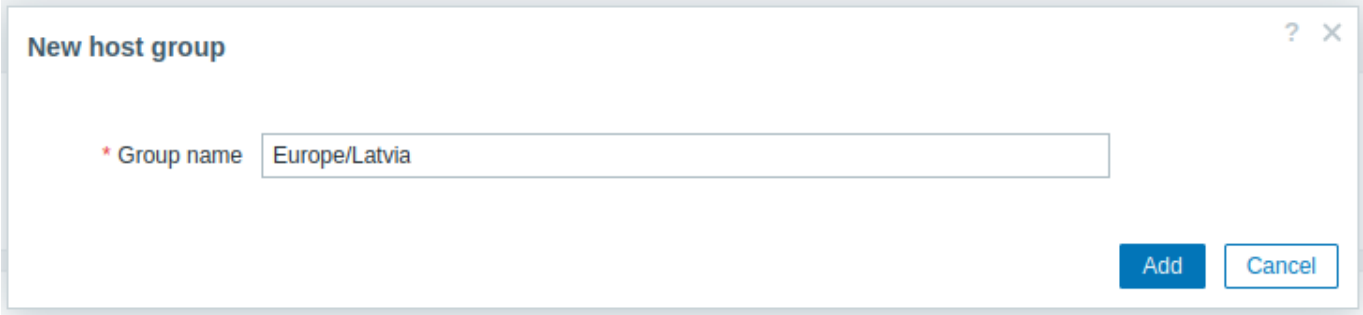
配置

> **Attention:**
> 只有超级管理员用户可以创建主机组。

在 Zabbix 前端中有两种创建主机组的方式。

方式一：

1. 转到 数据采集 > 主机组。
2. 单击屏幕右上角的 创建主机组。
3. 在表单中输入组名。



方式二：在配置主机时，在 主机组输入字段中输入一个不存在的组名。

创建主机组后，您可以在 数据采集 > 主机组下的列表中单击组名，以编辑组名、克隆该组或删除该组。

删除主机组只会删除逻辑组，不会删除组中的主机。对于任何现有主机而言，如果某个主机组是其唯一所属组，则无法删除该主机组。

创建主机子组

主机子组（或嵌套主机组）是包含它的父主机组的子组。

子组通过在组名输入字段中使用正斜杠 / 来表示其与父组的关系。例如：

- 输入 Europe/Latvia 会创建父组 Europe 下的 Europe/Latvia 子组。
- 输入 Europe/Latvia/Riga/Zabbix servers 会在嵌套的父组 Europe、Europe/Latvia、Europe/Latvia/Riga 中创建相应的子组。

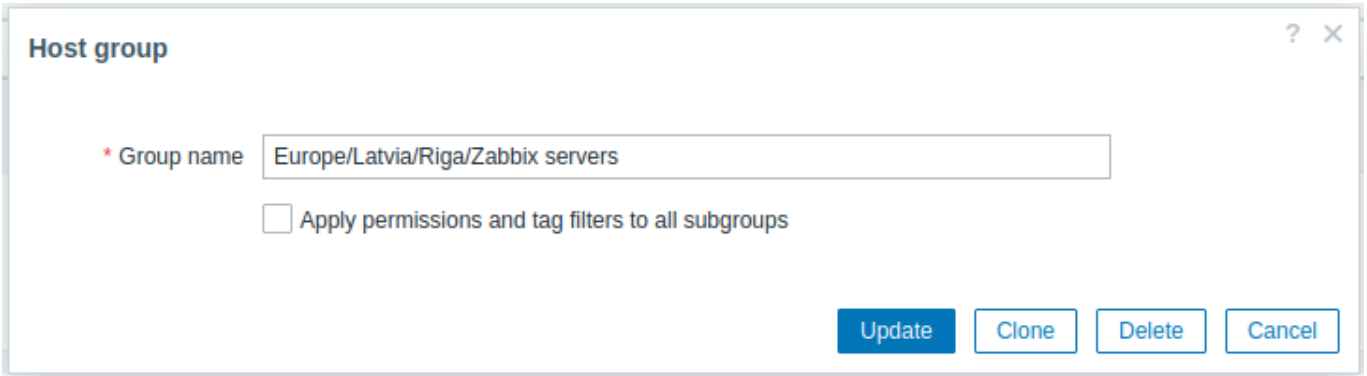
创建子组时，不允许使用前导或尾随斜杠，也不允许连续使用多个斜杠。不支持对 / 进行转义。

创建子组前，不要求先创建任何父主机组。您可以选择先创建子组（例如 Europe/Latvia），也可以先创建任意父主机组（在本例中为 Europe）。如果您先创建子组，父主机组不会自动创建。

主机组的权限

- 当为现有的父主机组创建子组时（例如，在 Europe 已存在的情况下创建 Europe/Latvia），用户组 对该子组的权限会从父组继承。
- 当为现有的子组创建父主机组时（例如，在 Europe/Latvia 已存在的情况下创建 Europe），不会为父组设置任何权限。

在编辑任何主机组时，您还可以设置一个附加选项，将权限和标签过滤器应用于所有子组。



勾选此复选框并点击 *Update* 后，将把相同级别的权限和标签过滤器应用于正在编辑的主机组的所有当前和未来子组。

因此，如果已为正在编辑的主机组的子组分配了不同的权限，勾选该复选框将使所有当前和未来子组获得与正在编辑的组相同的用户权限和基于标签的权限。

请注意，此选项不会保存在数据库中，并且会覆盖现有权限。通过此选项所做的任何更改只能手动撤销。

库存

概述

你可以将联网设备的资产信息保存在 Zabbix 里。

Zabbix 的前端管理页面中有一个特殊的 *Inventory*（资产记录）菜单。但你一开始不会看到任何数据，这里你也不能输入任何资产相关的信息。资产信息是在配置主机时人工录入建立的资产数据，或者是通过使用某些自动填充选项完成的录入。

构建资产库

手动模式

在配置主机时，您可以在资产清单选项卡中输入设备类型、序列号、位置、负责人、URL 等详细信息，这些数据将用于填充资产清单信息。

如果主机资产清单信息中包含 URL，且该 URL 以 “http” 或 “https” 开头，则会在资产清单部分显示为可点击的链接。

自动模式

主机资产信息也可以自动填充。要实现此功能，在配置主机时，必须将 资产清单选项卡中的资产清单模式设置为 自动。

然后，您可以配置主机监控项，使用其值填充任意主机资产清单字段，并在监控项配置中通过相应属性（称为 监控项将填充主机资产清单字段）指定目标字段。

以下监控项对于自动收集资产数据尤其有用：

- system.hw.chassis[full|type|vendor|model|serial] - 默认为 [full]，需要 root 权限
- system.hw.cpu[all|cpunum,full|maxfreq|vendor|model|curfreq] - 默认为 [all,full]
- system.hw.devices[pci|usb] - 默认为 [pci]
- system.hw.macaddr[interface,short|full] - 默认为 [all,full]，interface 是正则表达式
- system.sw.arch
- system.sw.os[name|short|full] - 默认为 [name]
- system.sw.packages[regexp,manager,short|full] - 默认为 [all,all,full]。

清单模式选择

可以在主机配置表单中选择清单模式。

新主机默认使用的清单模式取决于管理 > 常规 > 其他中的默认主机清单模式设置。

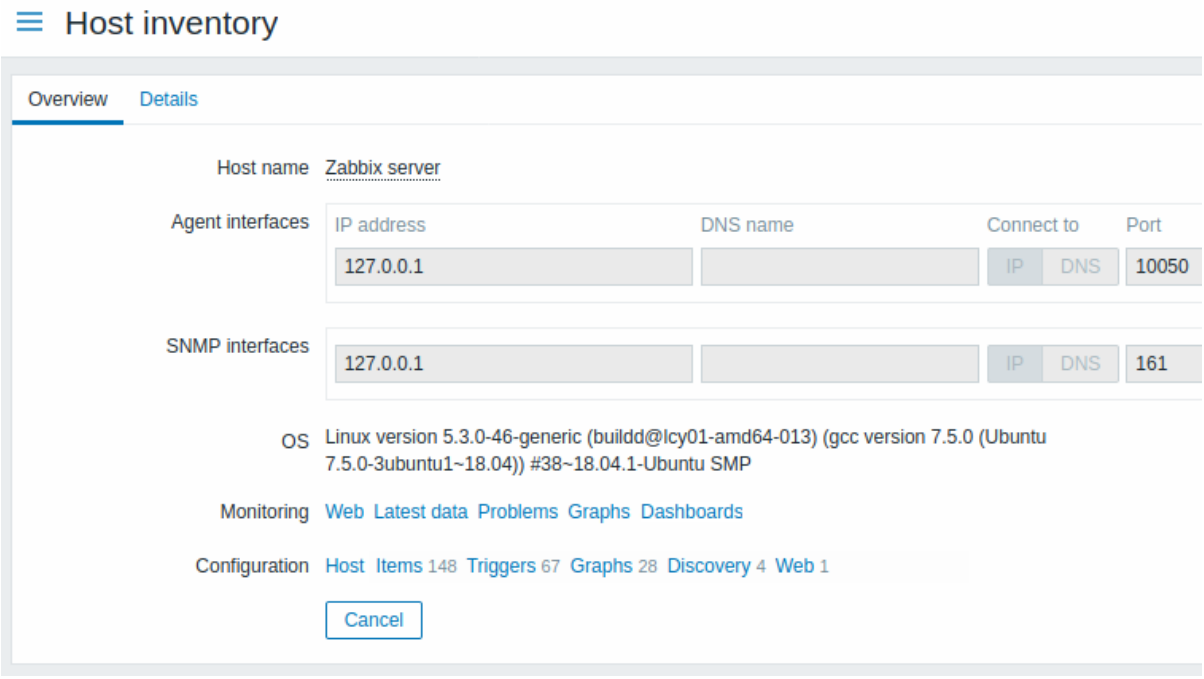
对于通过网络发现或自动注册操作添加的主机，可以定义设置主机清单模式操作，并选择手动或自动模式。此操作会覆盖默认主机清单模式设置。

库存概览

所有现有库存数据的详细信息都可在 *Inventory* 菜单中查看。

在 *Inventory* > *Overview* 中，您可以按库存的各个字段查看主机数量。

在 *Inventory* > *Hosts* 中，您可以看到所有包含库存信息的主机。单击主机名称后，会在表单中显示库存详细信息。



Overview 选项卡显示：

Parameter	Description
Host name	主机名称。 单击名称会打开一个菜单，其中包含为该主机定义的脚本。 如果主机处于维护中，主机名称会显示为橙色图标。
Visible name	主机的可见名称（如果已定义）。
Host (Agent, SNMP, JMX, IPMI) interfaces	此块提供为主机配置的接口详细信息。
OS	主机的操作系统库存字段（如果已定义）。
Hardware	主机硬件库存字段（如果已定义）。
Software	主机软件库存字段（如果已定义）。
Description	主机描述。
Monitoring	指向该主机数据的监控部分链接：*Web*、*Latest data*、*Problems*、*Graphs*、*Dashboards*。
Configuration	指向该主机配置部分的链接：*Host*、*Items*、*Triggers*、*Graphs*、*Discovery*、*Web*。 每个链接后都会列出已配置实体的数量。

Details 选项卡显示所有已填充（非空）的库存字段。

清单宏

可在通知中使用主机清单宏 `{INVENTORY.*}`，例如：

”位于 {INVENTORY.LOCATION1} 的服务器出现问题，负责人是 {INVENTORY.CONTACT1}，电话号码是 {INVENTORY.POC.PRIMARY.PHONE.A1}。”

有关更多详细信息，请参见支持的宏 页面。

批量更新

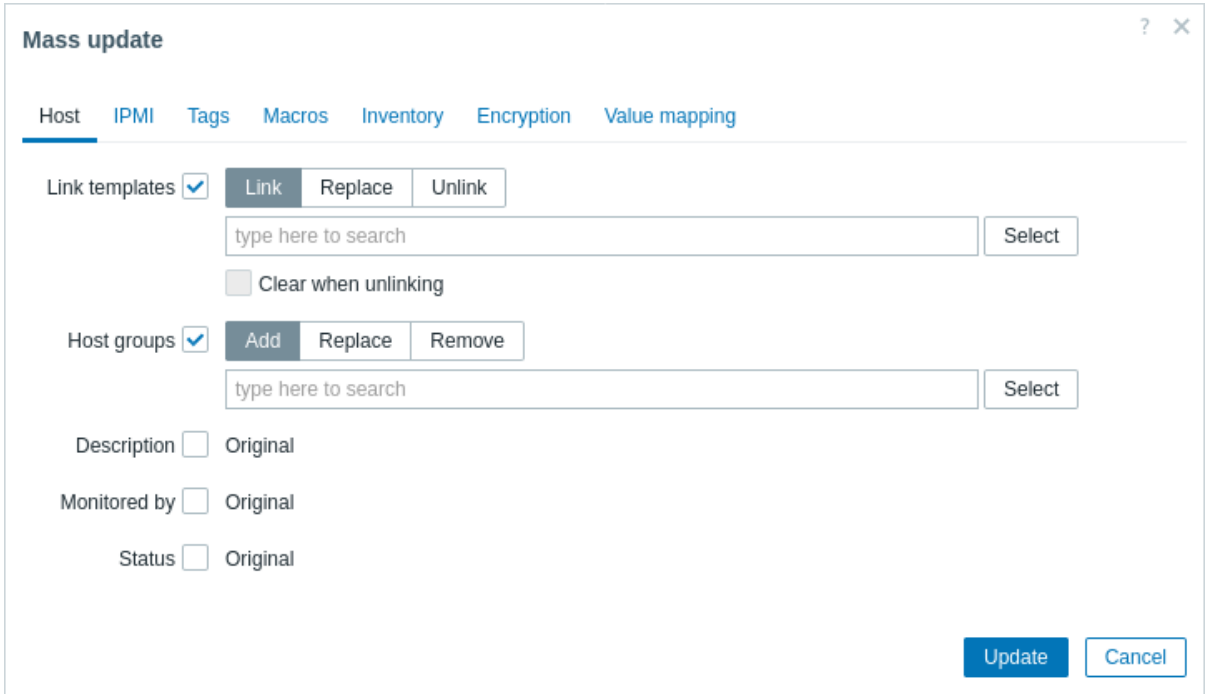
概述

有的时候你可能想要一次更改多个主机的某些属性。那么你可以使用批量更新功能来实现，而不是打开每个主机页面进行编辑。

使用批量更新

要批量更新某些主机，请执行以下操作：

- 在主机列表中，勾选要更新的主机前面的复选框
- 单击列表下方的批量更新
- 导航到包含所需属性的选项卡（主机、*IPMI*、标签、宏、资产清单、加密或值映射）
- 勾选要更新的属性对应的复选框，并为其输入新值



选择相应按钮更新模板链接时，可以使用以下选项：

- 链接 - 指定要链接的其他模板
- 替换 - 指定要链接的模板，同时取消链接之前已链接到主机的所有模板
- 取消链接 - 指定要取消链接的模板

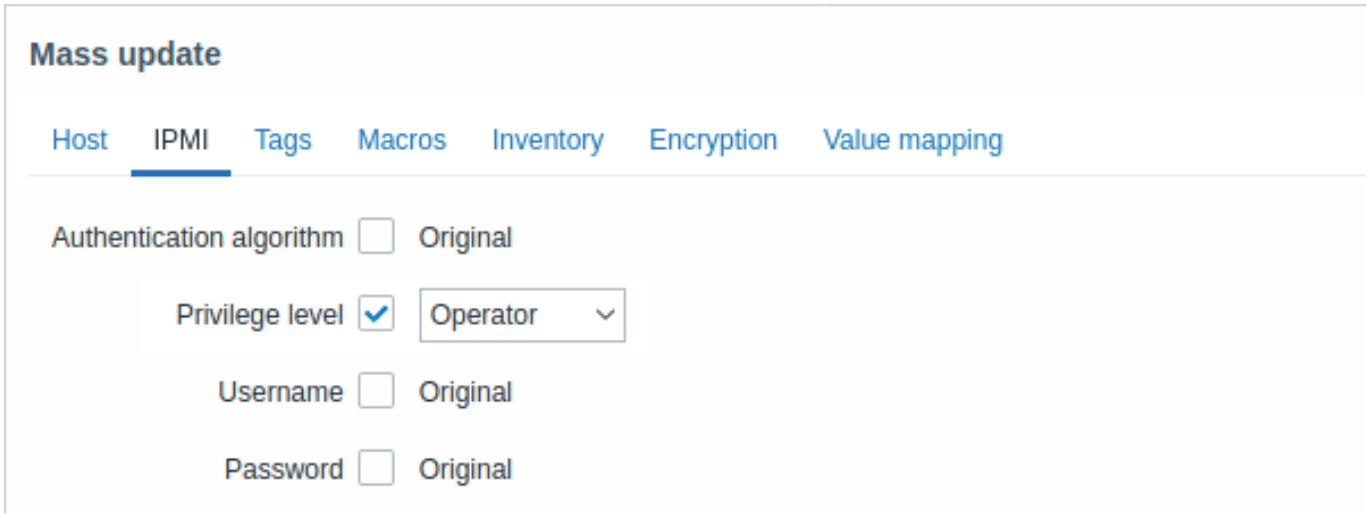
要指定要链接或取消链接的模板，请在自动完成字段中开始输入模板名称，直到出现下拉列表，其中会显示匹配的模板。向下滚动以选择所需的模板。

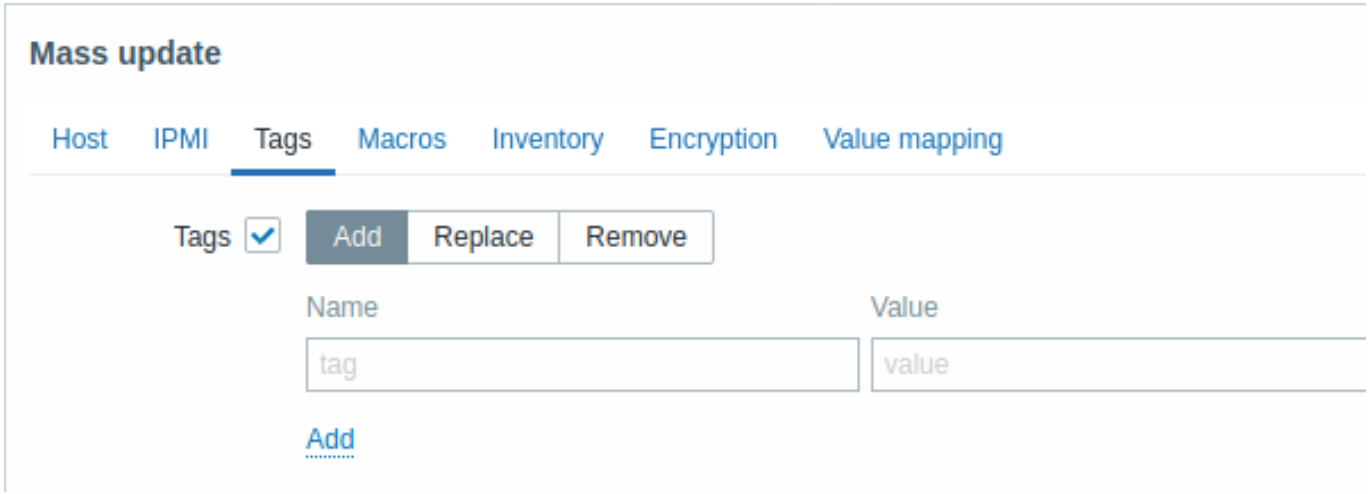
取消链接时清除选项不仅可以取消链接之前已链接的模板，还可以删除从这些模板继承的所有实体（监控项、触发器、图形等）。

选择相应按钮更新主机组时，可以使用以下选项：

- 添加 - 从现有主机组中指定其他主机组，或为主机输入全新的主机组；
- 替换 - 从所有现有主机组中移除主机，并将其替换为此字段中指定的主机组（现有或新的主机组）；
- 移除 - 从主机中移除指定的主机组。

这些字段支持自动完成：开始输入内容后，会显示包含匹配主机组的下拉列表。如果主机组是新的，也会显示在下拉列表中，并在字符串后以 (新建) 标识。向下滚动以选择。

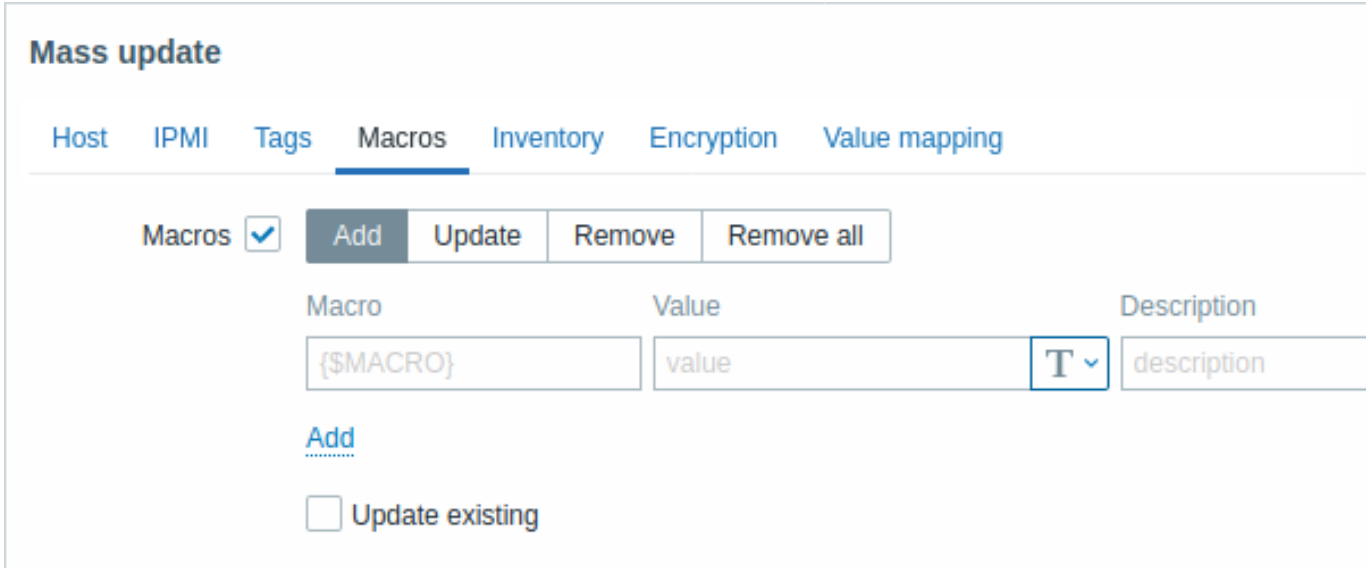




选择相应按钮更新标签时，可以使用以下选项：

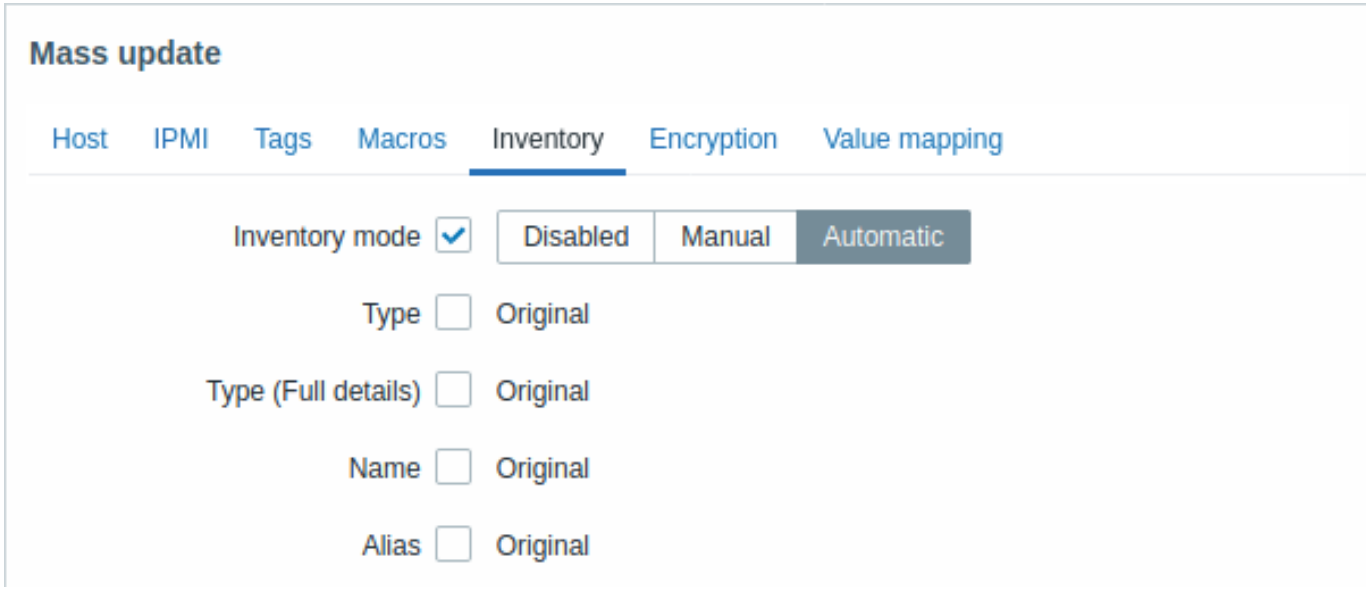
- 添加 - 为主机添加新标签；
- 替换 - 从主机中移除所有现有标签，并将其替换为下方指定的标签；
- 移除 - 从主机中移除指定的标签。

标签支持用户宏、{INVENTORY.*} 宏、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 宏。请注意，名称相同但值不同的标签不被视为“重复标签”，可以添加到同一主机。

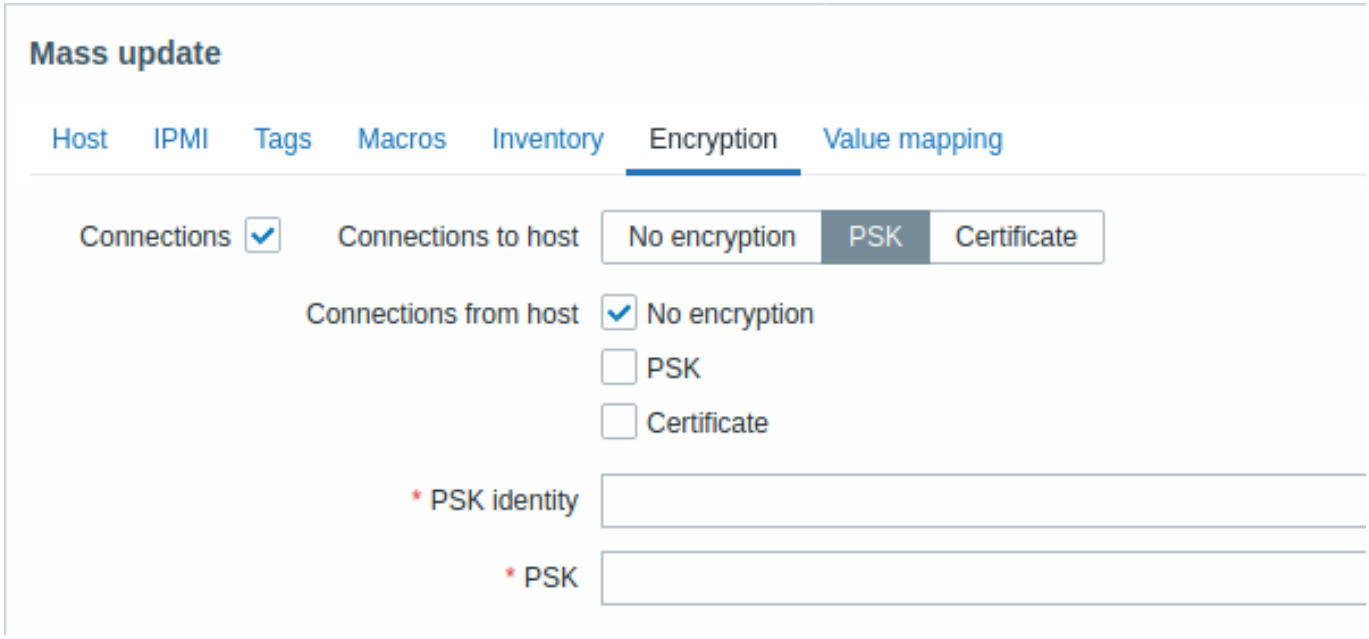


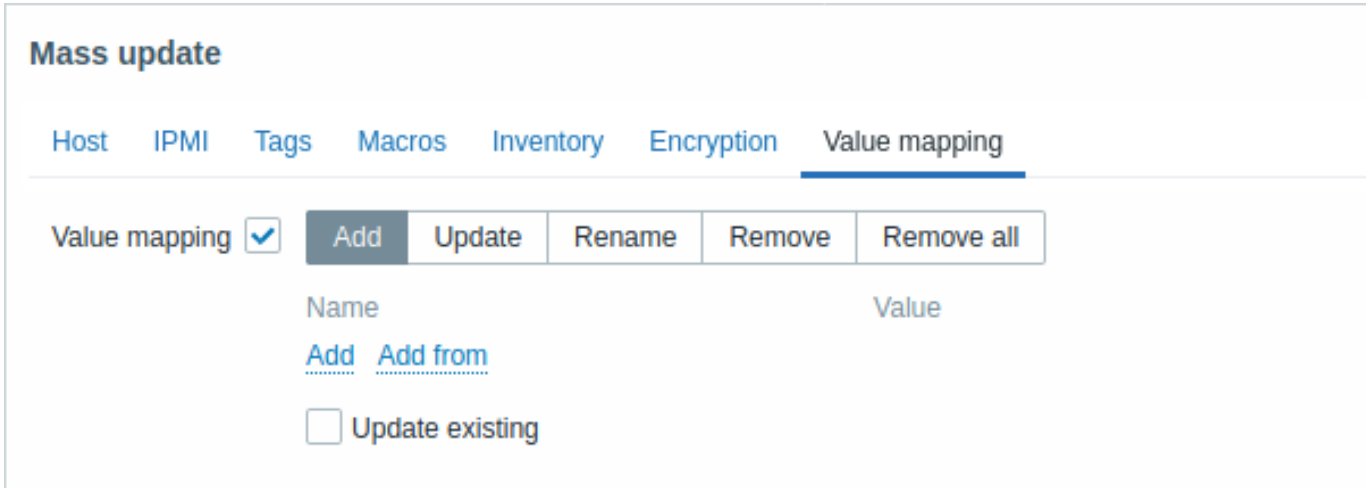
选择相应按钮更新宏时，可以使用以下选项：

- 添加 - 为主机指定其他用户宏。如果选中更新现有项复选框，则会更新指定宏名称的值、类型和描述。如果未选中，则当主机上已存在同名宏时，不会更新该宏。
- 更新 - 替换此列表中指定宏的值、类型和描述。如果选中添加缺失项复选框，则会将主机上之前不存在的宏作为新宏添加。如果未选中，则只会更新主机上已存在的宏。
- 移除 - 从主机中移除指定的宏。如果选中排除所选项复选框，则会移除列表中未指定的所有宏。如果未选中，则只会移除列表中指定的宏。
- 全部移除 - 从主机中移除所有用户宏。如果未选中我确认移除所有宏复选框，则会打开新的弹出窗口，要求确认移除所有宏。



要批量更新资产清单字段，资产清单模式应设置为 “手动” 或 “自动”。





更新值映射时，可以使用包含以下选项的按钮：

- 添加 - 向主机添加值映射。如果勾选更新现有项，则会更新具有此名称的值映射的所有属性。否则，如果已存在同名值映射，则不会更新该值映射。
- 更新 - 更新现有值映射。如果勾选添加缺失项，则会将主机上之前不存在的值映射作为新的值映射添加。否则，只会更新主机上已存在的值映射。
- 重命名 - 为现有值映射指定新名称

- 移除 - 从主机中移除指定的值映射。如果勾选排除所选项，则会移除所有值映射，但指定的值映射除外。
- 全部移除 - 从主机中移除所有值映射。如果未勾选我确认移除所有值映射复选框，则会打开新的弹出窗口，要求确认移除。

完成所有所需更改后，单击更新。所有选定主机的属性将相应更新。

监控项

概述

监控项是单独的指标。

监控项用于收集数据。配置主机后，必须添加监控项以获取实际的监控数据。快速添加多个监控项的一种方法是将一个预定义模板附加到主机。但是，为了优化系统性能，您可能需要微调模板以拥有尽可能多的监控项和尽可能频繁的监控。

要指定从主机收集哪种数据，请使用 [监控项键]（/manual/config/items/item/key）。例如，键名为 **system.cpu.load** 的监控项将收集处理器负载数据，而键名为 **net.if.in** 的监控项将收集传入流量信息。

可以在键名后的方括号中指定其他参数。例如，system.cpu.load**[avg5]** 将返回过去 5 分钟的处理器平均负载，而 net.if.in**[eth0]** 将显示接口 “eth0” 中的传入流量。

> **Note:**
> 请参阅监控项类型的各个部分，了解所有受支持的监控项类型和监控项键。

继续创建和配置监控项。

创建监控项

概述

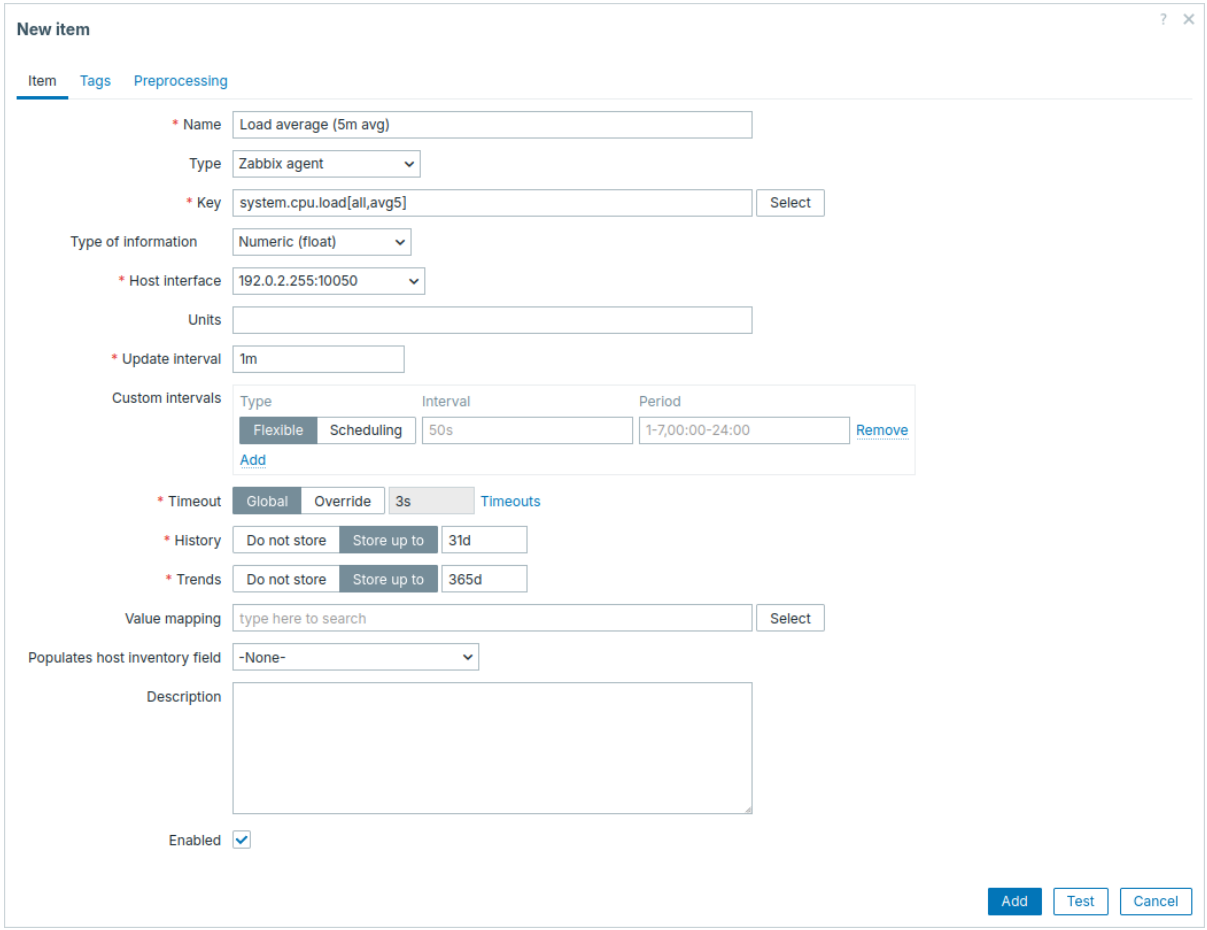
要在 Zabbix 前端中创建监控项，请执行以下操作：

1. 转到 数据采集 > 主机。
2. 在主机所在行中单击 监控项。
3. 单击屏幕右上角的 创建监控项。
4. 在表单中输入监控项的参数。

或者，您也可以打开现有监控项，单击 克隆按钮，然后使用其他名称保存。

配置

监控项选项卡包含监控项的一般属性。



所有必填字段均以红色星号标记。

参数	描述
名称	监控项名称。 支持用户宏。
类型	监控项类型。请参阅各个监控项类型部分。
键值	监控项键值（最多 2048 个字符）。 支持的监控项键值可在各个监控项类型部分中找到。 在单个主机内，键值必须唯一。 如果键值类型为 “Zabbix agent”、“Zabbix agent (active)” 或 “Simple check”，则键值必须受 Zabbix agent 或 Zabbix 服务器支持。 另请参阅正确的键值格式。
信息类型	用于验证监控项值并将其存储在数据库中的数据类型，包括进行任何转换之后的数据： 数值（无符号） - 64 位无符号整数； 数值（浮点） - 64 位浮点数； 字符 - 短文本； 日志 - 带有或不带日志属性（时间戳、来源、严重性、日志事件 ID）的长文本； 文本 - 长文本； 二进制 - 二进制数（仅依赖监控项支持）； **JSON** - 结构化 JSON 数据，以原生格式存储在数据库中（计算监控项不支持）。 返回二进制或 JSON 数据类型值的监控项不支持用于计算监控项公式或触发器表达式。 有关每种数据类型及数据库后端的存储限制，请参阅监控项数据限制。 对于以单一特定格式返回数据的监控项，系统会自动选择匹配的数据类型——例如，`system.cpu.load` 和 `system.cpu.util` 都返回 64 位浮点数，因此会自动选择数值（浮点）。
主机接口	选择主机接口。在主机级别编辑监控项时，此字段可用。

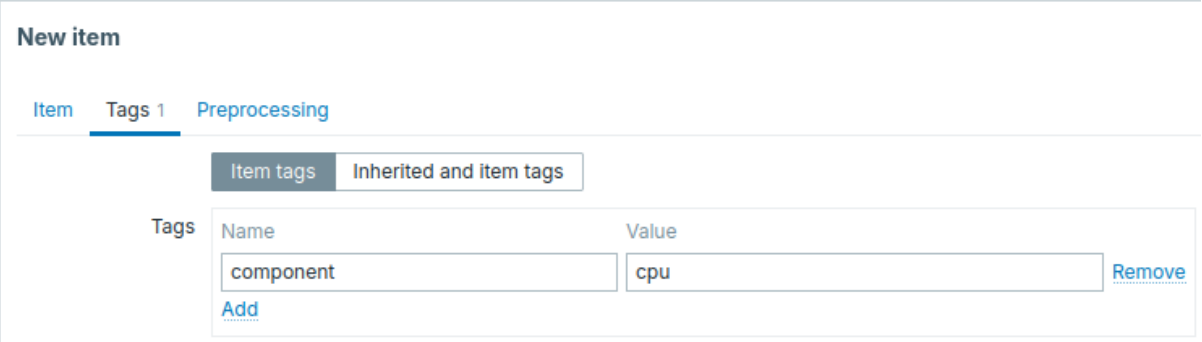
参数	描述
单位	如果设置了单位符号，Zabbix 会对接收到的监控项值进行后处理，并使用指定的单位后缀显示该值。 支持特殊格式的单位符号（以及接收到的监控项值 → 显示值示例）： **B** - 字节（1024 → 1 KB） **Bps** - 每秒字节数（1024 → 1 KBps） **s** - 秒，使用最多三个最大的非零时间单位显示（881764 → 10d 4h 56m） **uptime** - 以 hh:mm:ss 或 N days, hh:mm:ss 格式显示的运行时间（881764 → 10 days, 04:56:04） **unixtime** - Unix 时间戳，格式为 yyyy.mm.dd hh:mm:ss（881764 → 1970-01-11 04:56:04 AM）； 要正确格式化，接收到的监控项值必须为数值（无符号）。 对于其他单位（如 Hz、W 等），如果接收到的值超过 1000，则会除以 1000，并使用相应的前缀显示（5000 → 5 KHz，881764 → 881.76 KW）。 如果单位符号以 ！ 开头，则会禁用单位转换和前缀处理（1024 !B → 1024 B，61 !s → 61 s）。 有关单位符号和单位转换的更多示例及详细信息，请参阅监控项值后缀。
更新间隔	每 N 秒获取一次此监控项的新值。允许的最大更新间隔为 86400 秒（1 天）。 支持时间后缀，例如 30s、1m、2h、1d。 支持用户宏。 单个宏必须填充整个字段。不支持在一个字段中使用多个宏，也不支持宏与文本混合使用。 注意：仅当存在值不为零的自定义间隔时，更新间隔才能设置为 “0”。如果设置为 “0”，并且存在值不为零的自定义间隔（灵活或计划），则会在自定义间隔持续时间内轮询监控项。 注意，监控项变为活动状态后，或更新间隔发生更改后，第一次轮询可能早于配置的值发生。 新监控项将在创建后的 60 秒内进行检查，除非该监控项使用计划或灵活更新间隔，且更新间隔设置为 0。 可以通过点击立即执行按钮，立即轮询现有被动监控项的值。
自定义间隔	可以为监控项创建自定义检查规则： 灵活 - 为更新间隔创建例外（使用不同频率的间隔）。 计划 - 创建自定义轮询计划。 有关详细信息，请参阅自定义间隔。 间隔字段支持时间后缀，例如 30s、1m、2h、1d。 支持用户宏。 单个宏必须填充整个字段。不支持在一个字段中使用多个宏，也不支持宏与文本混合使用。
超时	设置监控项检查超时时间（适用于受支持的监控项类型）。选择超时选项： 全局 - 使用 proxy/全局超时（显示在灰显的超时字段中）。 覆盖 - 使用自定义超时（在超时字段中设置；允许范围：1 - 600s）。支持时间后缀（例如 30s、1m）和用户宏。 点击超时链接，可以配置proxy超时或全局超时（未使用 proxy 时）。请注意，只有具有超级管理员类型且拥有管理 > 常规或管理 > *Proxy*前端部分权限的用户，才能看到超时链接。
历史数据	选择以下选项之一： 不存储 - 不存储监控项历史数据。适用于主监控项仅需由依赖监控项保留历史数据的情况。 此设置不能被全局 housekeeper 设置覆盖。 存储时长 - 指定在数据库中保留详细历史数据的时长（1 小时至 25 年）。更早的数据将由 housekeeper 删除。以秒为单位存储。 支持时间后缀，例如 2h、1d。支持用户宏。 存储时长值可以在管理 > *Housekeeping*中进行全局覆盖。 如果存在全局覆盖设置，则会显示橙色 信息图标。将鼠标放在该图标上时，会显示警告消息，例如：已被全局 *housekeeper* 设置覆盖（*1d*）。 建议将记录值保留尽可能短的时间，以减小数据库中值历史数据的大小。与其存储较长的值历史，不如存储更长时间的趋势数据。 另请参阅历史数据和趋势。
趋势	选择以下选项之一： 不存储 - 不存储趋势数据。 此设置不能被全局 housekeeper 设置覆盖。 存储时长 - 指定在数据库中保留聚合历史数据（每小时最小值、最大值、平均值、计数）的时长（1 天至 25 年）。更早的数据将由 housekeeper 删除。以秒为单位存储。 支持时间后缀，例如 24h、1d。支持用户宏。 存储时长值可以在管理 > *Housekeeping*中进行全局覆盖。 如果存在全局覆盖设置，则会显示橙色 信息图标。将鼠标放在该图标上时，会显示警告消息，例如：已被全局 *housekeeper* 设置覆盖（*7d*）。 注意：非数值数据（字符、日志和文本）不支持保留趋势数据。 另请参阅历史数据和趋势。

参数	描述
值映射	将值映射应用于此监控项。值映射不会更改接收到的值，仅用于显示数据。 适用于数值（无符号）、数值（浮点）和字符监控项。 例如：“Windows 服务状态”。
日志时间格式	仅适用于日志类型的监控项。支持以下占位符： **y**：年份（*1970-2038*）。 **M**：月份（*01-12*）。 **d**：日期（*01-31*）。 **h**：小时（*00-23*）。 **m**：分钟（*00-59*）。 **s**：秒（*00-59*）。 如果留空，时间戳将设置为 Unix 时间 0，即 1970 年 1 月 1 日。 例如，考虑以下来自 Zabbix agent 日志文件的行： “ 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).” 该行以 6 个字符位置的 PID 开头，后面依次为日期、时间和其余消息。 此行的日志时间格式为 “ppppp:yyyyMMdd:hhmmss”。 请注意，“p” 和 “:” 字符是占位符，可以是除 “yMdhms” 之外的任意字符。
填充主机资产清单字段	可以选择一个主机资产清单字段，由监控项的值填充。当主机启用了自动资产清单填充，并且监控项类型为数值（无符号）、数值（浮点）、字符或文本时，此功能有效。
描述	输入监控项描述。支持用户宏。
启用	选中复选框以启用监控项，使其得到处理。 禁用监控项时，它会立即从历史缓存中移除（日志除外，日志的最后一个值会保留）。
最新数据	点击链接查看监控项的最新数据。 仅在编辑已存在的监控项时提供此链接。

Note:
监控项类型特定字段在相应页面中进行了说明。

Note:
在主机级别编辑现有的模板级别监控项时，部分字段为只读。可以使用表单标题中的链接转到模板级别并在那里进行编辑，但请注意，在模板级别进行的更改将修改与该模板关联的所有主机上的监控项。

标签选项卡允许您定义监控项级别的标签。



监控项值预处理

预处理选项卡允许您为接收到的值定义转换规则。

监控项测试

Attention:
要执行监控项测试，请确保服务器和 proxy 上的系统时间已同步。如果服务器时间落后，监控项测试可能会返回错误消息”The task has been expired.” 不过，即使服务器和 proxy 设置了不同的时区，也不会影响测试结果。

可以测试监控项，并在配置正确时返回一个真实值。即使在监控项保存之前，也可以进行测试。

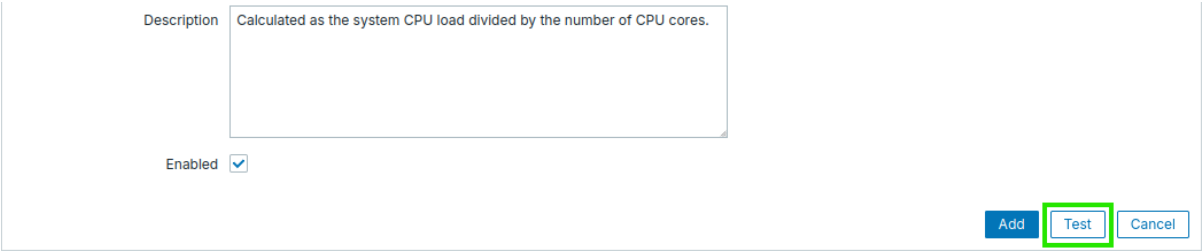
可测试的对象包括主机和模板监控项、监控项原型以及低级别发现规则。主动式监控项不支持测试。

以下被动式监控项类型支持测试：

- Zabbix agent
- SNMP agent (v1, v2, v3)

- IPMI agent
- SSH checks
- Telnet checks
- JMX agent
- Simple checks (except `icmpping*`, `vmware.*` items)
- Zabbix internal
- Calculated items
- External checks
- Database monitor
- HTTP agent
- Script
- Browser

要测试监控项，请点击监控项配置表单底部的 *Test* 按钮。请注意，对于无法测试的监控项（例如主动检查、被排除的简单检查），*Test* 按钮将被禁用。

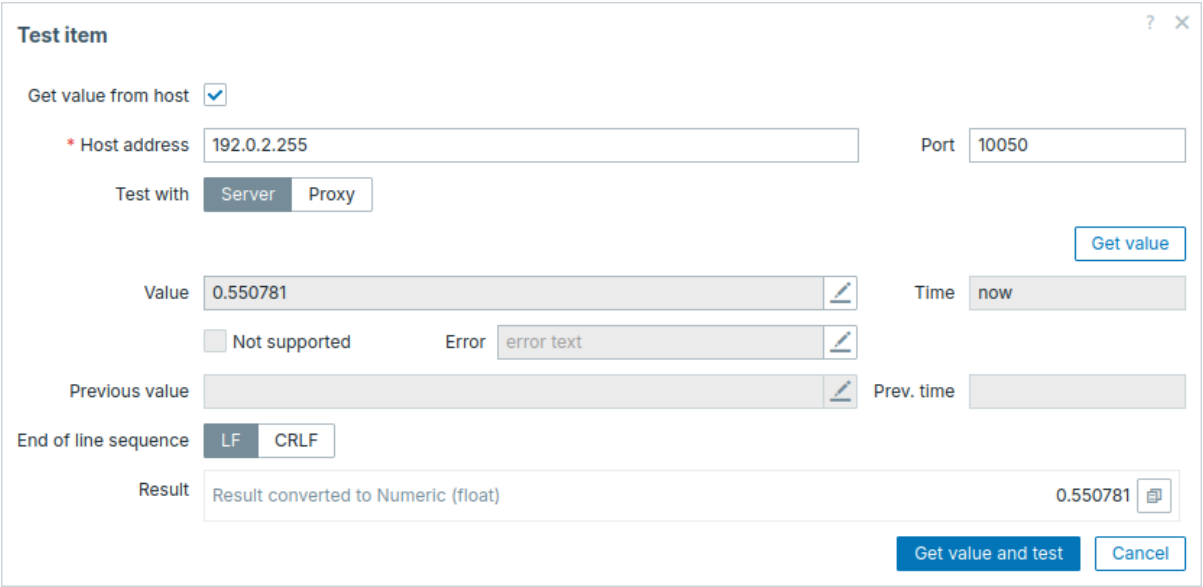


监控项测试表单包含所需主机参数的字段（主机地址、端口、使用服务器/proxy 测试（proxy 名称））以及监控项特定的详细信息（例如 SNMPv2 community 或 SNMPv3 安全凭据）。这些字段具有上下文感知能力：

- 在可能的情况下会预填充值，例如对于需要 agent 的监控项，会从主机所选的 agent 接口中获取信息。
- 模板监控项的值必须手动填写。
- 明文宏值会被解析。
- 值（或值的一部分）为密钥或 Vault 宏的字段会为空，必须手动输入。如果任何监控项参数包含密钥宏值，将显示以下警告消息：”Item contains user-defined macros with secret values. Values of these macros should be entered manually.”
- 在当前监控项类型不需要时，这些字段会被禁用（例如，对于计算型监控项，主机地址字段和 proxy 字段会被禁用）

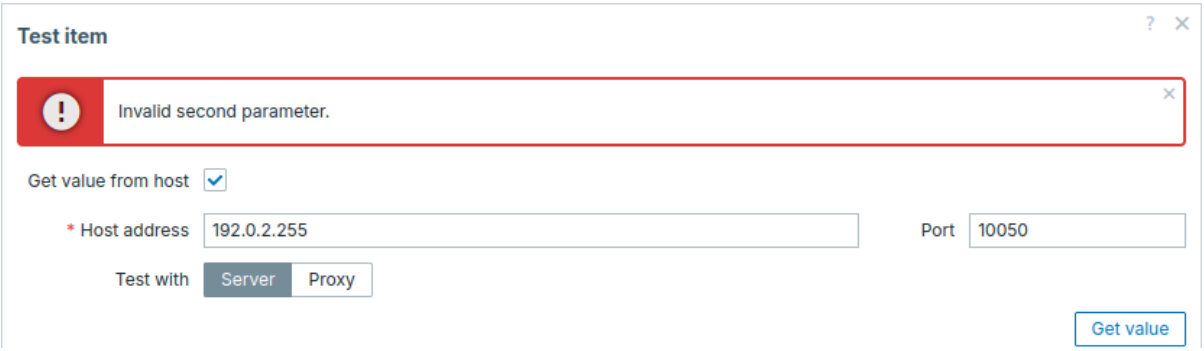
要测试监控项，请点击 *Get value*。如果成功获取到值，它会填充 *Value* 字段，同时将当前值（如果有）移到 *Previous value* 字段，并计算 *Prev. time* 字段，即两个值（点击）之间的时间差，同时尝试检测 EOL 序列，并在检测到返回值中的”\n\r” 时切换为 CRLF。

点击 *Get value and test* 可测试预处理。



从主机获取的值和测试结果在发送到前端时会被截断，最大大小为 512KB。如果测试结果被截断，会显示一个警告图标；将鼠标悬停其上可查看详情。如果值被截断，在后续比较大值（例如 JSON）的测试中，可能会导致数据类型验证失败；在这种情况下，会显示错误消息。请注意，超过 512KB 的数据仍会由 Zabbix 服务器完整处理。

如果配置不正确，会显示描述可能原因的错误消息。



从主机成功获取的值也可用于测试预处理步骤。

表单按钮

表单底部的按钮可用于执行多项操作。

Add	添加监控项。此按钮仅适用于新建监控项。
Update	更新监控项的属性。
Clone	根据当前监控项的属性创建另一个监控项。
Execute now	立即对新的监控项值执行检查。仅支持被动检查（请参阅更多详细信息）。 注意，立即检查值时不会更新配置缓存，因此该值不会反映监控项配置最近发生的更改。
Test	通过获取值测试监控项配置是否正确。
Clear history and trends	删除监控项的历史记录和趋势数据。
Delete	删除监控项。
Cancel	取消编辑监控项属性。

监控项数据限制

监控项数据限制取决于监控项数据类型和数据库后端。

数值（无符号）值按原样存储。如果接收到浮点值，则只存储整数部分（例如，1.23 → 1）。

数值（浮点）值按原样存储，并支持大约 15 到 17 位精度，范围约为 -1.79E+308 到 1.79E+308。也支持科学计数法表示的值（例如，1.23E+7、1e308、1.1E-4）。

文本值在存储前会被截断，以匹配数据库值类型限制：

Database	Type of information		
	Character	**Log**	**Text**
MySQL	255 characters	65536 bytes	65536 bytes
PostgreSQL	255 characters	65536 characters	65536 characters
SQLite (only Zabbix proxy)	255 characters	65536 characters	65536 characters

二进制值限制为 16MiB（16777216 字节）。超过此限制的值将被丢弃，并且会在前端显示相应错误。

JSON 数据类型的值限制为 128MiB（134217728 字节）。超过此限制的值将被丢弃，并且会在前端显示相应错误。对于 MySQL/MariaDB，建议为 `history_json` 表（用于存储 JSON 值）设置每日分区，因为它可能会迅速增长，并长时间锁定housekeeper。如果你的使用场景涉及较大的 JSON 值（1MiB 或更大），请参阅如何为large JSON value support 配置系统。

当 Zabbix 将任何监控项数据写入数据库时，会使用 INSERT 查询并记录这些查询（如果将`DebugLevel` 设置为 4 或 5）。较大的日志条目会被截断为 64KB。

自定义脚本限制

可用的自定义脚本长度取决于所使用的数据库：

数据库	字符数限制	字节数限制
MySQL	65535	65535
PostgreSQL	65535	不受限制
SQLite（仅 *Zabbix proxy*）	65535	不受限制

监控项超时

监控项超时指定 Zabbix 在中止检查并将其判定为失败之前应等待多长时间。

当达到超时时间时，即使数据检索尚未完成，检查也会被中止。如果只接收到部分数据，该监控项将变为不受支持，并记录一条错误消息（例如，在 SNMP 检查中，如果多个 OID 里只有一个成功收集到数据）。

对于许多监控项类型，您可以设置灵活的监控项超时：

- 按单个监控项设置
- 按监控项类型设置（在proxy 级别）
- 按监控项类型设置（在全局 级别）

如果您希望为某个特定监控项指定更长的超时时间，同时保持其他监控项的超时时间较低，那么自定义的单个监控项超时会很有用。

超时优先级

1. 单个监控项超时会覆盖任何其他超时设置。
2. proxy 级别超时会覆盖全局超时。

对于已设置灵活超时的检查，Zabbix 服务器或proxy 配置中的超时不起作用。

灵活超时支持

以下监控项类型支持灵活的监控项超时：

- Zabbix agent（被动和主动检查均支持）
- SNMP agent（不包括传统 SNMP 检查[1]）
- 简单检查（不包括 `icmpping*` 和 VMware 监控项[2]）
- SSH agent
- Telnet agent
- 外部检查
- 数据库监控
- 脚本
- HTTP agent
- 浏览器

[1] 对于传统 SNMP 检查（单个 OID 轮询），适用来自服务器 或proxy 的超时设置。
 [2] 对于 `icmpping*` 监控项，超时值直接在监控项键值中指定；同时还适用硬编码的最大可能超时 600 秒。VMware 监控项使用其自身的 VMwareTimeout 参数，该参数来自服务器或proxy 配置。

不受支持的监控项

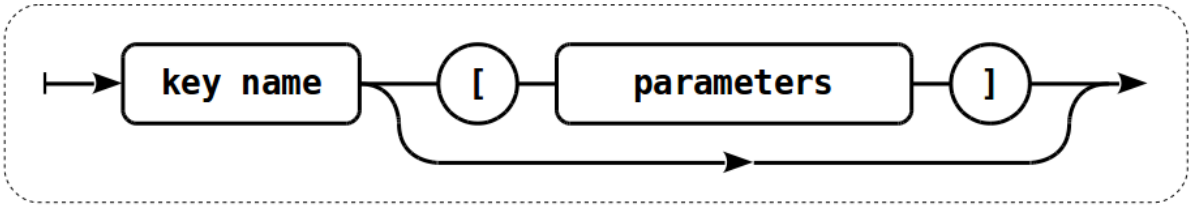
不受支持的监控项会显示为 不支持状态，并且仍会按照其标准的 更新间隔 重新检查。

如果由于某种原因无法获取监控项的值，则该监控项会变为不受支持状态（例如，连接错误，或未配置用于处理该监控项的 poller）。此外，未接收到历史数据的监控项也会保持在 不支持状态。这是预期行为——只有当监控项接收到新的有效历史数据时，才会转换为 正常状态。

与问题类似，不受支持的监控项也仅在接收到新数据时才会重新评估——即使该监控项已不再有任何可用的历史数据。换句话说，监控项和触发器仅会在接收到新数据时改变其状态。但是，如果触发器表达式包含日期和时间 和/或nodata() 函数，则触发器将每 30 秒重新计算一次（详见触发器计算时间）。

监控项键值格式

监控项键值格式，包括键值参数，必须遵循语法规则。下图展示了支持的语法。每个位置允许的元素和字符可以通过沿箭头方向查看来确定 - 如果某个块可以通过连线到达，则表示允许；如果不能到达 - 则表示不允许。



要构造一个有效的监控项键值，首先要指定键值名称，然后可以选择是否带参数 - 如图中可沿着两条线中的任意一条继续。

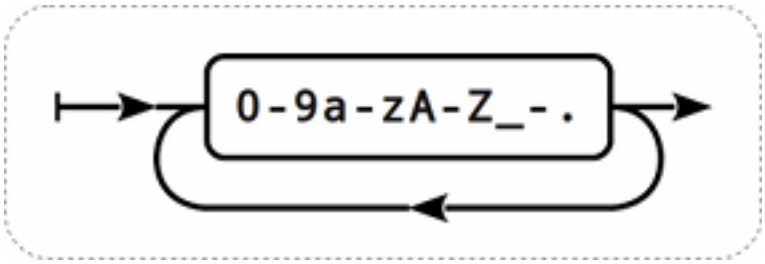
键值的名称

键名本身具有有限的允许字符范围，允许的字符是：

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

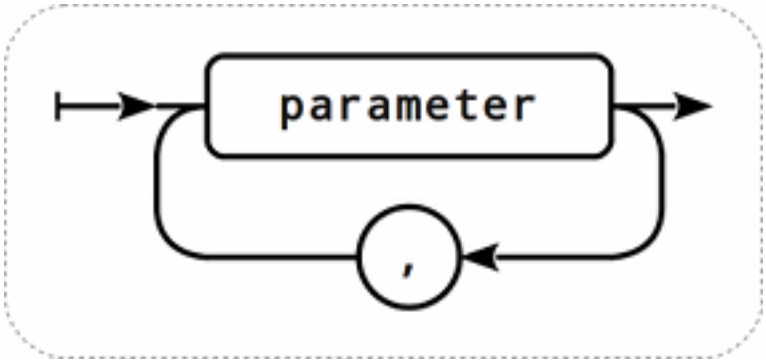
即:

- 所有的数字;
- 所有的小写字母;
- 所有大写字母;
- 下划线;
- 减号;
- 点。

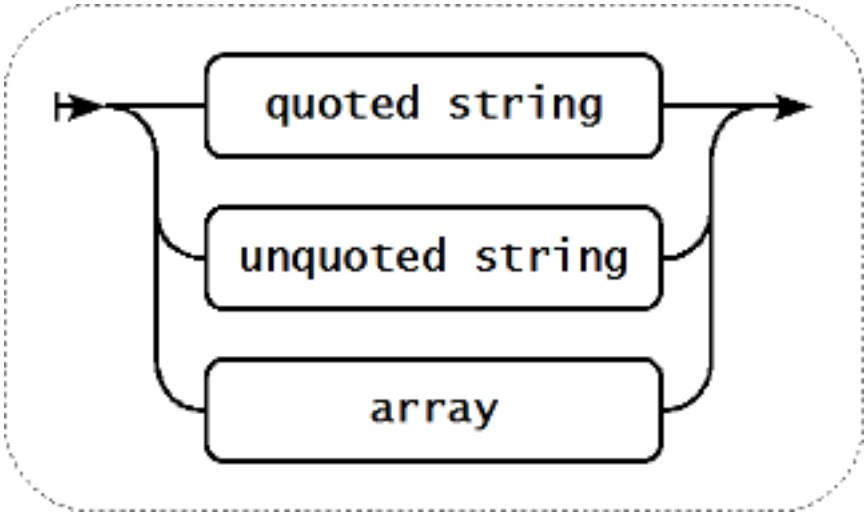


键值的参数

监控项的键值可以有多个被逗号所分隔的参数。



每个键值参数可以是带引号、不带引号的字符串或数组。

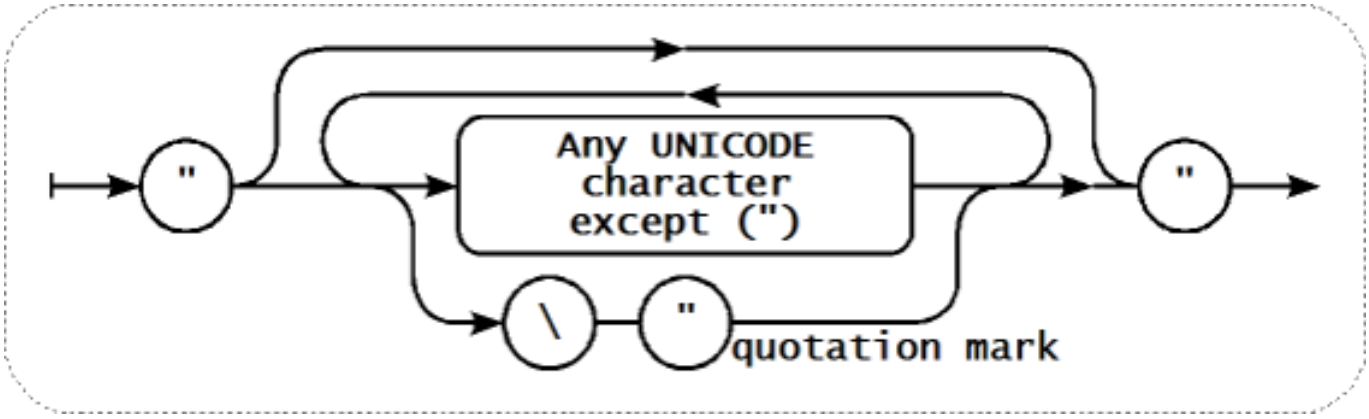


参数也可以为空，此时使用默认值。在这种情况下，如果指定了任何其它参数，则必须添加适当数量的逗号。例如，键值 **icmp-ping[„200„500]** 将指定单个 ping 之间的时间间隔为 200 毫秒，超时时间为 500 毫秒，其它所有参数为默认值。

参数中可以包含宏。这些宏可以是用户宏 或某些内置宏。要查看监控项键值参数中支持哪些特定的内置宏，请在页面支持的宏 中搜索 “监控项键值参数”。

参数 - 带引号的字符串

如果键参数是带引号的字符串，则允许使用任何 Unicode 字符。如果键参数字符串包含引号，则该参数必须加引号，并且作为参数字符串一部分的每个引号都必须使用反斜杠（\）字符进行转义。如果键参数字符串包含逗号，则该参数必须加引号。

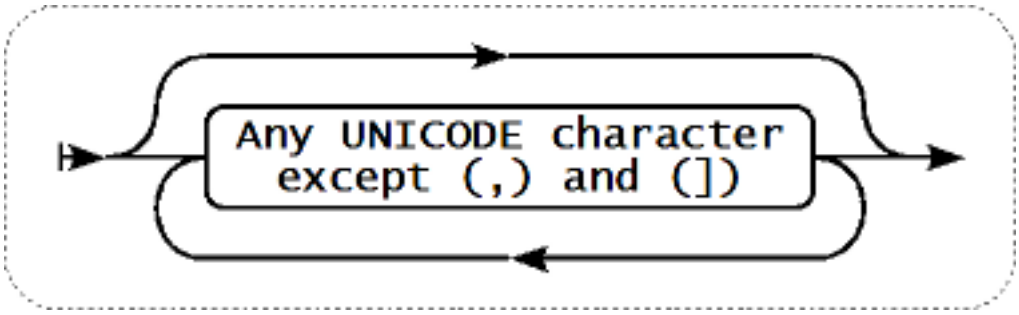


Warning:
为监控项键参数加引号时，仅可使用双引号。不支持单引号。

Attention:
不允许使用多级参数数组，例如 `[a,[b,[c,d]],e]`。

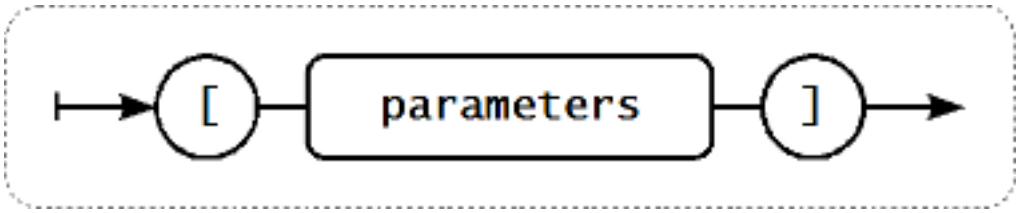
参数 - 不带引号的字符串

如果键值的参数是一个不带引号的字符串，可使用除逗号和右方括号（]）之外的任何 Unicode 字符。不带引号的参数不能以左方括号（[）作为开头。



参数 - array

如果键参数是一个数组，则它同样需要用方括号括起来，其中各个参数按照指定多个参数的规则和语法依次排列。



自定义间隔

概述

可以创建有关监控项检查时间的自定义规则。有两种方法可以实现这一点：灵活间隔允许重新定义默认更新间隔；调度则可以在特定时间或按特定时间序列执行监控项检查。

灵活间隔

灵活间隔允许您为特定时间段重新定义默认更新间隔。灵活间隔由 *Interval* 和 *Period* 定义，其中：

- *Interval* - 指定时间段的更新间隔。支持时间后缀，例如 30s、1m、2h、1d。
- *Period* - 灵活间隔生效的时间段（有关 *Period* 格式的详细说明，请参阅时间段）

如果多个灵活间隔存在重叠，则对重叠时间段使用最小的 *Interval* 值。请注意，如果重叠灵活间隔的最小值为 “0”，则不会执行轮询。在灵活间隔之外，将使用默认更新间隔。

请注意，如果灵活间隔等于时间段的长度，则监控项将恰好检查一次。如果灵活间隔大于时间段，则监控项可能检查一次，也可能完全不检查（因此不建议采用此类配置）。如果灵活间隔小于时间段，则监控项至少会检查一次。

如果将灵活间隔设置为“0”，则在灵活间隔时间段内不会对监控项执行轮询；时间段结束后，将根据默认的 *Update interval* 恢复轮询。
示例：

Interval	Period	Description
10	1-5,09:00-18:00	工作时间内，监控项每 10 秒检查一次。
0	1-7,00:00-7:00	夜间不会检查监控项。
0	7-7,00:00-24:00	星期日不会检查监控项。
60	1-7,12:00-12:01	每天 12:00 检查一次监控项。请注意，此配置曾用于变通实现计划检查，建议对此类检查使用调度间隔。

调度间隔

调度间隔用于在特定时间检查监控项。灵活间隔用于重新定义监控项的默认更新间隔，而调度间隔则用于指定独立的检查计划，并行执行。

调度间隔定义为：`md<filter>wd<filter>h<filter>m<filter>s<filter>`，其中：

- **md** - 每月日期
- **wd** - 星期几
- **h** - 小时
- **m** - 分钟
- **s** - 秒

`<filter>` 用于为其前缀指定值（日期、小时、分钟、秒），定义为：`[<from>[-<to>]][/<step>][,<filter>]`，其中：

- `<from>` 和 `<to>` 定义匹配值的范围（包含边界值）。如果省略 `<to>`，则过滤器匹配 `<from> - <from>` 范围。如果 `<from>` 也省略，则过滤器匹配所有可能的值。
- `<step>` 定义在该范围内数值跳过的步长。默认情况下 `<step>` 的值为 1，这意味着将匹配所定义范围内的所有值。

虽然过滤器定义是可选的，但必须至少使用一个过滤器。过滤器必须定义范围或 *<step>* 值之一。

空过滤器在未定义更低级别过滤器时匹配“0”，否则匹配所有可能的值。例如，如果省略小时过滤器，则仅匹配“0”时，前提是分钟和秒过滤器也都被省略；否则，空的小时过滤器将匹配所有小时值。

各过滤器前缀对应的有效 `<from>` 和 `<to>` 值如下：

Prefix	Description	*<from>*	*<to>*
md	每月日期	1-31	1-31
wd	星期几	1-7	1-7
h	小时	0-23	0-23
m	分钟	0-59	0-59
s	秒	0-59	0-59

`<from>` 值必须小于或等于 `<to>` 值。`<step>` 值必须大于或等于 1，且小于或等于 `<to> - <from>`。

个位数的每月日期、小时、分钟和秒值可以带前导 0。例如，`md01-31` 和 `h/02` 是有效间隔，但 `md01-031` 和 `wd01-07` 无效。

在 Zabbix 前端中，多个调度间隔需分别输入在单独的行中。在 Zabbix API 中，它们会被连接成一个字符串，并以分号 `;` 作为分隔符。

如果某个时间同时匹配多个间隔，则只执行一次。例如，`wd1h9;h9` 在星期一上午 9 点只会执行一次。

示例：

Interval	Will be executed
m0-59	每分钟
h9-17/2	从 9:00 开始每 2 小时一次（9:00、11:00 ...）
m0,30 or m/30	每小时的 hh:00 和 hh:30
m0,5,10,15,20,25,30,35,40,45,50,55 or m/5	每五分钟
wd1-5h9	每周一至周五的 9:00
wd1-5h9-18	每周一至周五的 9:00、10:00、...、18:00
h9,10,11 or h9-11	每天的 9:00、10:00 和 11:00
md1h9m30	每月 1 日的 9:30
md1wd1h9m30	如果每月 1 日是星期一，则在 9:30 执行
h9m/30	每天的 9:00、9:30
h9m0-59/30	每天的 9:00、9:30
h9,10m/30	每天的 9:00、9:30、10:00、10:30
h9-10m30	每天的 9:30、10:30

Interval	Will be executed
h9m10-40/30	每天的 9:10、9:40
h9,10m10-40/30	每天的 9:10、9:40、10:10、10:40
h9-10m10-40/30	每天的 9:10、9:40、10:10、10:40
h9m10-40	每天的 9:10、9:11、9:12、... 9:40
h9m10-40/1	每天的 9:10、9:11、9:12、... 9:40
h9-12,15	每天的 9:00、10:00、11:00、12:00、15:00
h9-12,15m0	每天的 9:00、10:00、11:00、12:00、15:00
h9-12,15m0s30	每天的 9:00:30、10:00:30、11:00:30、12:00:30、15:00:30
h9-12s30	每天的 9:00:30、9:01:30、9:02:30 ... 12:58:30、12:59:30
h9m/30;h10 (*API-specific syntax*)	每天的 9:00、9:30、10:00
h9m/30 h10 (在前端中将其添加为另一行)	每天的 9:00、9:30、10:00

为 proxy 和 agent 对齐时区

请注意，Zabbix proxy 和 agent 在处理调度间隔时会使用其本地时区。

因此，当调度间隔应用于由 Zabbix proxy 监控的监控项或 agent active 监控项时，建议将相应 proxy 或 agent 的时区设置为与 Zabbix 服务器相同，否则，队列 可能会错误地报告监控项延迟。

Zabbix proxy 或 agent 的时区可以通过在 `systemd` 单元文件中使用环境变量 `TZ` 来设置：

```
[Service]
...
Environment="TZ=Europe/Amsterdam"
```

监控项值预处理

概述

预处理允许您在将接收到的监控项值保存到数据库之前，对其应用转换。这些转换/预处理步骤由 Zabbix 服务器或 proxy 执行（如果监控项由 proxy 监控）。

此功能支持多种使用场景，例如：

- 将字节转换为位（例如，将网络流量值乘以 “8”）；
- 计算持续递增数值的每秒统计信息；
- 应用正则表达式以提取或修改值；
- 对值执行自定义脚本；
- 丢弃未变化的值以优化数据库存储。

可以为一个监控项配置一个或多个预处理步骤。这些步骤将按照其配置顺序执行。

Attention:
如果某个预处理步骤失败，监控项将变为不支持状态。这可以通过 *Custom on fail* 错误处理来避免（大多数转换都支持此功能），它允许您丢弃值或设置自定义值。

 对于日志监控项，日志元数据（不含值）始终会重置监控项的不支持状态，使其重新变为受支持状态。即使初始错误是在从 agent 接收到日志值之后发生的，也会如此。

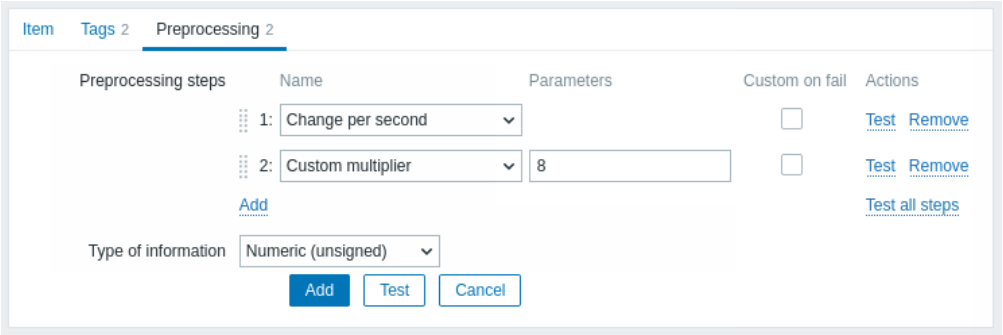
传递给预处理的所有值在初始阶段都被视为字符串。在预处理流水线结束时，系统会将其转换为所需的值类型（如监控项配置中所定义）。但是，如果需要，某些特定的预处理步骤可能会更早触发转换。有关详细技术信息，请参见预处理详情。

为确保您的预处理配置按预期工作，您可以进行测试。

另请参见：预处理示例

配置

预处理步骤在监控项配置表单的预处理选项卡中定义。



单击添加以选择受支持的转换。

定义至少一个预处理步骤后，选项卡底部会显示信息类型字段。如有需要，可以直接在预处理选项卡中更改信息类型。有关参数的详细说明，请参阅创建监控项。

支持的转换

下面列出了所有支持的转换。
单击转换名称可查看其完整详细信息。

名称	描述	类型
正则表达式	将值与正则表达式进行匹配，并替换为所需的输出。	文本
替换	查找搜索字符串，并将其替换为其他字符串（或不替换为任何内容）。	
修剪	从值的开头和结尾删除指定字符。	
右侧修剪	从值的结尾删除指定字符。	
左侧修剪	从值的开头删除指定字符。	
XML XPath	使用 XPath 功能从 XML 数据中提取值或片段。	结构化数据
JSON Path	使用JSONPath 功能从 JSON 数据中提取值或片段。	
CSV 转 JSON	将 CSV 文件数据转换为 JSON 格式。	
XML 转 JSON	将 XML 格式的数据转换为 JSON。	
SNMP walk 值	根据指定的 OID/MIB 名称提取值，并应用格式化选项。	SNMP
SNMP walk 转 JSON	将 SNMP 值转换为 JSON。	
SNMP get 值	将格式化选项应用于 SNMP get 值。	
自定义乘数	将值乘以指定的整数或浮点值。	算术
简单变化	计算当前值与前一个值之间的差异。	变化
每秒变化	计算值变化（当前值与前一个值之间的差异）的每秒速率。	
布尔值转十进制	将值从布尔格式转换为十进制。	数字系统
八进制转十进制	将值从八进制格式转换为十进制。	
十六进制转十进制	将值从十六进制格式转换为十进制。	
JavaScript	输入 JavaScript 代码。	自定义脚本
范围内	定义值应处于的范围。	验证
匹配正则表达式	指定值必须匹配的正则表达式。	
不匹配正则表达式	指定值不得匹配的正则表达式。	
检查 JSON 中的错误	检查位于 JSONPath 的应用程序级错误消息。	
检查 XML 中的错误	检查位于 XPath 的应用程序级错误消息。	
使用正则表达式检查错误	使用正则表达式检查应用程序级错误消息。	
检查不支持的值	检查是否无法获取监控项值。	
丢弃未更改的值	如果值未发生变化，则丢弃该值。	限流
带心跳的丢弃未更改值	如果值在定义的时间段内未发生变化，则丢弃该值。	
Prometheus 模式	使用以下查询从 Prometheus 指标中提取所需数据。	Prometheus
Prometheus 转 JSON	将所需的 Prometheus 指标转换为 JSON。	

如果为所选转换启用了 失败时自定义处理复选框，则可以选中该选项并指定自定义错误处理方式：丢弃值（丢弃该值）、将值设置为（设置指定的值）或 将错误设置为（设置指定的错误消息）。
如果预处理步骤失败，并且选择了 丢弃值或 将值设置为选项，则监控项不会变为不支持状态。

请注意，对于 变化和 限流预处理步骤，Zabbix 必须记住上一个值，以便按要求计算或比较新值。
这些先前的值由预处理管理器处理。
如果 Zabbix 服务器或 proxy 重新启动，或者对预处理步骤进行了任何更改，则相应监控项的上一个值会被重置，从而导致：

- 对于 简单变化、每秒变化步骤，下一个值将被忽略，因为没有可用于计算变化的上一个值；
- 对于 丢弃未更改的值、带心跳的丢弃未更改值步骤，即使根据丢弃规则本应丢弃下一个值，也永远不会将其丢弃。

正则表达式

将值与正则表达式进行匹配，并替换为所需的输出。

参数：

- **pattern** - 正则表达式；
- **output** - 输出格式模板。\N（其中 N=1...9）转义序列将替换为第 N 个匹配组。\0 转义序列将替换为匹配的文本。

注释：

- 如果无法匹配输入值，监控项将变为不受支持；
- 正则表达式支持使用 \N 序列提取最多 10 个捕获组；
- 有关现有示例，请参阅正则表达式部分。

替换

查找搜索字符串，并将其替换为另一个字符串（或不替换为任何内容）。

参数：

- **search string** - 要查找和替换的字符串，区分大小写（必填）；
- **replacement** - 用于替换搜索字符串的字符串。替换字符串也可以为空，这样在找到搜索字符串时即可将其删除。

注释：

- 搜索字符串的所有出现位置都会被替换；
- 可以使用转义序列来搜索或替换换行符、回车符、制表符和空格"\n \r \t \s"；反斜杠可以转义为"\\"，转义序列可以转义为"\\n"；
- 在低级别发现期间，换行符、回车符和制表符会自动进行转义。

截取

从值的开头和结尾移除指定的字符。

右截取

从值的结尾移除指定的字符。

左截取

从值的开头移除指定的字符。

XML XPath

使用 XPath 功能从 XML 数据中提取值或片段。

注释：

- 要使此选项正常工作，Zabbix 服务器（或 Zabbix proxy）必须在编译时启用 libxml 支持；
- 不支持命名空间。

示例：

```
number(/document/item/value) # 将从 <document><item><value>10</value></item></document> 中提取“10”
number(/document/item/@attribute) # 将从 <document><item attribute="10"></item></document> 中提取“10”
/document/item # 将从 <document><item><value>10</value></item></document> 中提取“<item><value>10</value></it
```

JSON Path

使用JSONPath 功能 从 JSON 数据中提取值或片段。

CSV 转 JSON

将 CSV 文件数据转换为 JSON 格式。

更多信息，请参见：CSV 转 JSON 预处理。

XML 转 JSON

将 XML 格式的数据转换为 JSON。

有关更多信息，请参阅：序列化规则。

SNMP walk 值

通过指定的 OID/MIB 名称提取值，并应用格式化选项：

- 不变 - 将十六进制字符串作为未转义的十六进制字符串返回（注意，显示提示仍会应用）；
- 从十六进制字符串转换为 **UTF-8** - 将十六进制字符串转换为 UTF-8 字符串；

- 从十六进制字符串转换为 **MAC** - 将十六进制字符串验证为 MAC 地址，并返回规范的 MAC 地址字符串（其中的 ' ' 将替换为 ':'）；
- 从 **BITS** 转换为整数 - 将以十六进制字符序列表示的位字符串的前 8 个字节（例如，“1A 2B 3C 4D”）转换为 64 位无符号整数。对于超过 8 个字节的位字符串，后续字节将被忽略。

SNMP walk 转 JSON

将 SNMP 值转换为 JSON。

在 JSON 中指定字段名称及其对应的 SNMP OID 路径。字段值将由指定 SNMP OID 路径中的值填充。

注释：

- 提供与 *SNMP walk value* 步骤中类似的值格式化选项；
- 您可以将此预处理步骤用于SNMP OID 发现。

SNMP 获取值

对 SNMP 获取值应用格式选项：

- 从 **Hex-STRING** 转换为 **UTF-8** - 将十六进制字符串转换为 UTF-8 字符串；
- 从 **Hex-STRING** 转换为 **MAC** - 将十六进制字符串验证为 MAC 地址，并返回正确的 MAC 地址字符串（其中的 ' ' 将替换为 ':'）；
- 从 **BITS** 转换为整数 - 将以十六进制字符序列表示的位字符串的前 8 个字节（例如”1A 2B 3C 4D”）转换为 64 位无符号整数。对于长度超过 8 个字节的位字符串，后续字节将被忽略。

自定义乘数

将值乘以指定的整数或浮点数。

注释：

- 使用此选项可将以 KB、MBps 等为单位接收的值转换为 B、Bps。否则，Zabbix 无法正确设置前缀（K、M、G 等）。
- 请注意，如果监控项的信息类型为 *Numeric (unsigned)*，则在应用自定义乘数之前，带小数部分的传入值将被截断（即，’0.9’ 将变为’0’）；
- 如果对信息类型设置为 *Numeric (unsigned)* 的监控项使用自定义乘数，或将值存储为 *Change per second*，且最终计算出的值实际为浮点数，则仍会将该计算值作为正确值接受：截去小数部分，并将其作为整数存储；
- 支持科学计数法，例如 1e+70；用户宏和 LLD 宏；包含宏的字符串，例如 `{#MACRO}e+10`、`{$MACRO1}e+{$MACRO2}`。宏必须解析为整数或浮点数。

简单变化

计算当前值与之前值之间的差异。

注释：

- 此步骤可用于测量持续增长的值；
- 计算方式为 **value-prev_value**，其中 *value* - 当前值；*prev_value* - 之前接收到的值；
- 每个监控项只允许执行一种变化操作（“简单变化” 或 “每秒变化”）；
- 如果当前值小于之前的值，Zabbix 将丢弃该差异（不存储任何内容），并等待下一个值。

每秒变化量

计算每秒的值变化速度（当前值与之前值之间的差值）。

注释：

- 此步骤适用于计算持续增长值的每秒增长速度；
- 由于此计算可能产生浮点数，因此建议将 “信息类型” 设置为 *Numeric (float)*，即使传入的原始值为整数。对于较小的数值，小数部分尤为重要。如果浮点值较大，可能超出 “float” 字段长度，从而导致整个值丢失，则建议使用 *Numeric (unsigned)*，这样只会截去小数部分。
- 计算方式为 (**value-prev_value**)/(**time-prev_time**)，其中 *value* - 当前值；*prev_value* - 之前接收的值；*time* - 当前时间戳；*prev_time* - 之前值的时间戳；
- 每个监控项只允许执行一个变化操作（“Simple change” 或 “Change per second”）；
- 如果当前值小于之前的值，Zabbix 将丢弃该差值（不存储任何内容），并等待下一个值。这有助于正确处理例如 32 位 SNMP 计数器回绕（溢出）的情况。

布尔值转十进制

将值从布尔格式转换为十进制。

注释：

- 文本表示形式将转换为 0 或 1。因此，'TRUE' 存储为 1，'FALSE' 存储为 0。所有值均以不区分大小写的方式进行匹配。当前可识别的值包括：*TRUE* - true、t、yes、y、on、up、running、enabled、available、ok、master；*FALSE* - false、f、no、n、off、down、unused、disabled、unavailable、err、slave。此外，任何非零数值都视为 TRUE，零视为 FALSE。

八进制转十进制

将值从八进制格式转换为十进制。

十六进制转十进制

将值从十六进制格式转换为十进制。

JavaScript

在参数字段中单击或单击其旁边的铅笔图标时，在打开的模态编辑器中输入 JavaScript 代码。

> **Attention:**
> 请勿在预处理 JavaScript 中使用未声明的赋值。请使用 `var` 声明局部变量。

注释：

- 可用的 JavaScript 长度取决于所使用的数据库；
- 更多信息，请参见：JavaScript 预处理。

在范围内

通过指定最小值/最大值（包括边界值），定义数值应处于的范围。

注释：

- 接受数值（包括任意位数、可选的小数部分和可选的指数部分，以及负值）；
- 最小值应小于最大值；
- 至少必须存在一个值；
- 可以使用用户宏和低级别发现宏。

匹配正则表达式

指定值必须匹配的正则表达式。

不匹配正则表达式

指定值不得匹配的正则表达式。

检查 JSON 中的错误

检查位于 JSONPath 的应用程序级错误消息。

如果检查成功且消息不为空，则停止处理；否则，继续使用此预处理步骤之前的值进行处理。

注释：

- 这些外部服务错误将原样报告给用户，不会添加预处理步骤信息；
- 如果无法解析无效的 JSON，则不会报告错误。

检查 XML 中的错误

检查位于 XPath 的应用程序级错误消息。如果检查成功且消息不为空，则停止处理；否则，继续使用此预处理步骤之前的值进行处理。

注释：

- 这些外部服务错误将原样报告给用户，不会添加预处理步骤信息；
- 如果无法解析无效的 XML，则不会报告错误。

使用正则表达式检查错误

使用正则表达式检查应用程序级错误消息。
如果检查成功且消息不为空，则停止处理；否则，继续使用此预处理步骤之前的值进行处理。

参数：

- **pattern** - 正则表达式；
- **output** - 输出格式模板。
  \N（其中 N=1...9）转义序列将替换为第 N 个匹配组。
  \0 转义序列将替换为匹配的文本。

注释：

- 这些外部服务错误将原样报告给用户，不会添加预处理步骤信息。

检查不支持的值

检查是否无法获取监控项值。通过检查返回的错误消息，指定应如何处理该故障。

参数：

- **scope** - 选择错误处理范围：
 任意错误 - 任意错误；
 错误匹配 - 仅匹配 *pattern* 中指定正则表达式的错误；
 错误不匹配 - 仅不匹配 *pattern* 中指定正则表达式的错误
- **pattern** - 用于匹配错误的正则表达式。如果在范围参数中选择了 任意错误，则不显示此字段。如果显示此字段，则为必填字段。

注释：

- 通常，无法获取值或获取值失败会导致监控项变为不支持。此预处理步骤允许您修改此行为。
- 此预处理步骤仅检查是否无法获取监控项值。例如，它不会检查获取的值类型（如字符串）是否与监控项的信息类型（如数值）匹配；有关详细信息，请参阅预处理示例。如果类型不匹配，在执行完所有预处理步骤后，监控项仍可能变为不支持。要检查类型是否匹配，可以使用例如 自定义倍数预处理步骤；请参阅预处理示例。
- 支持在 设置值为或 设置错误为字段中使用正则表达式捕获组。使用 \N（其中 N=1...9）获取第 N 个匹配组；使用 \0 获取匹配的文本；
- 这些步骤始终作为第一个预处理步骤执行，并在保存对监控项的更改后置于所有其他步骤之上；
- 支持多个 检查不支持的值步骤，并按指定顺序执行。任意错误步骤将自动作为此组中的最后一个步骤。

丢弃未变化的值

如果某个值未发生变化，则将其丢弃。

说明：

- 如果某个值被丢弃，则不会将其保存到数据库中，Zabbix 服务器也不会知道已接收到该值。
  不会对任何触发器表达式进行求值，因此也不会为相关触发器创建/恢复任何问题。
  函数将仅基于实际保存到数据库中的数据运行。
  由于趋势数据是基于数据库中的数据构建的，如果某一小时没有保存任何值，则该小时也不会有趋势数据。
- 每个监控项只能指定一个限流选项。

在心跳周期内丢弃未更改的值

如果某个值在定义的时间段（以秒为单位）内未发生更改，则丢弃该值。

备注：

- 支持使用正整数指定秒数（最小值为 1 秒）；
- 可以使用时间后缀（例如：30s、1m、2h、1d）；
- 可以使用用户宏和低级别发现宏；
- 如果某个值被丢弃，则不会将其保存到数据库中，Zabbix 服务器也不会知道已接收到该值。因此，不会评估任何触发器表达式，也不会为相关触发器创建或解决任何问题。函数仅基于实际保存到数据库中的数据运行。由于趋势数据基于数据库中的数据构建，如果某个小时内没有保存任何值，则该小时也不会有趋势数据。
- 每个监控项只能指定一个限流选项。

Prometheus 模式

使用以下查询从 Prometheus 指标中提取所需数据。

有关更多详细信息，请参见Prometheus 检查。

Prometheus 转 JSON

将所需的 Prometheus 指标转换为 JSON。

有关更多详细信息，请参见Prometheus 检查。

宏支持

用户宏 和带上下文的用户宏支持用于：

- 预处理步骤参数，包括 JavaScript 代码；
- 自定义错误处理参数（设置值为和 设置错误为字段）。

> **Note:**
> 当宏被替换为其值时，宏上下文会被忽略。宏值会按原样插入到代码中，无法在将该值放入 JavaScript 代码之前添加额外的转义。请注意，这在某些情况下可能会导致 JavaScript 错误。

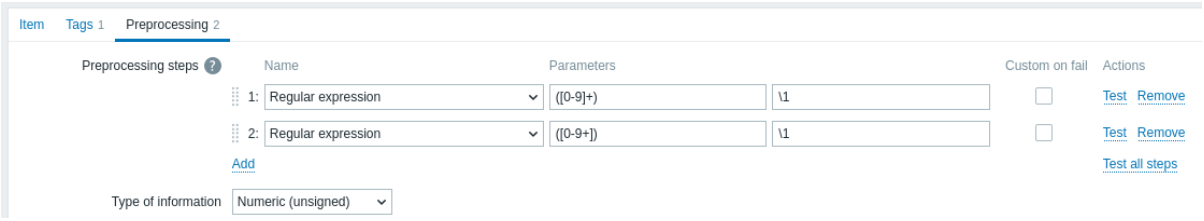
测试

参见预处理测试。

预处理测试

测试

测试预处理步骤有助于确保复杂的预处理流水线能够产生预期结果，而无需等待监控项值被接收并完成预处理。

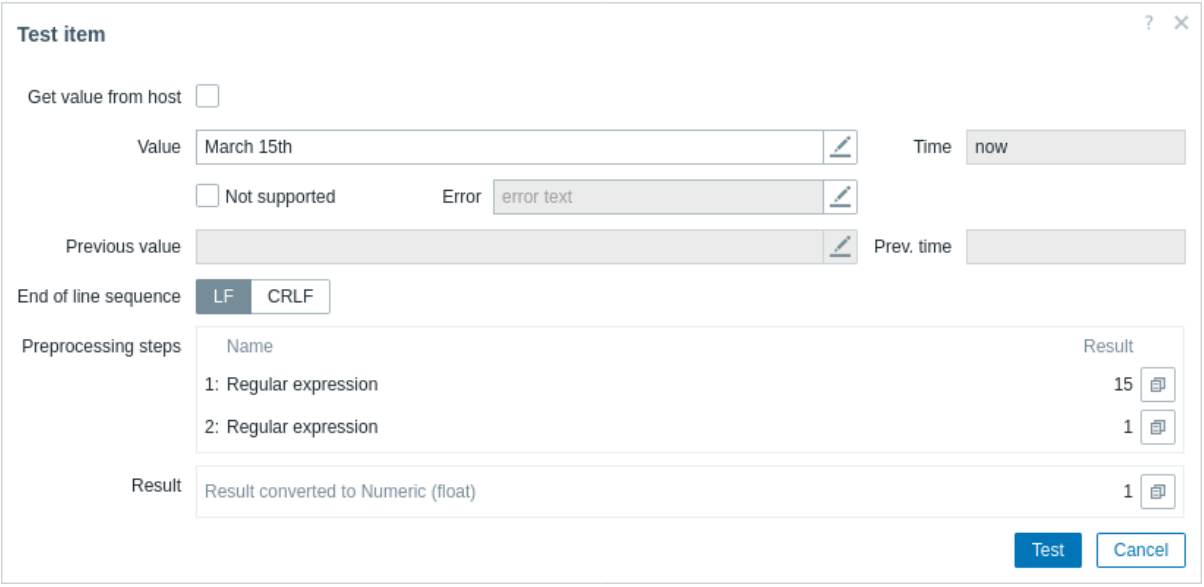


可以测试：

- 针对一个假设值
- 针对来自主机的真实值

每个预处理步骤都可以单独测试，也可以将所有步骤一起测试。当你在 Actions 区块中分别点击 *Test* 或 *Test all steps* 按钮时，会打开一个测试窗口。

测试假设值



Parameter	Description
Get value from host	如果要测试假设值，请不要勾选此复选框。 另请参见：测试真实值。
Value	输入要测试的输入值。 单击参数字段或查看/编辑按钮 将打开一个文本区域窗口，用于输入值或代码块。
Not supported	勾选此复选框以测试不受支持的值。 此选项适用于测试 *Check for not supported value* 预处理步骤。
Error	输入错误文本。 当未勾选 *Get value from host*，但已勾选 *Not supported* 时，此字段可用。 如果勾选了 *Get value from host*，则此字段会填入来自主机的实际错误消息（只读）。
Time	显示输入值的时间：`now`（只读）。
Previous value	输入要比较的上一个输入值。 仅适用于 *Change* 和 *Throttling* 预处理步骤。
Previous time	输入要比较的上一个输入值时间。 仅适用于 *Change* 和 *Throttling* 预处理步骤。 默认值基于监控项的'Update interval' 字段值（如果为'1m'，则此字段填充为 `now-1m`）。如果未指定任何内容，或者用户无权访问主机，则默认值为 `now-30s`。
Macros	如果使用了任何宏，它们会连同其值一起列出。为便于测试，这些值可以编辑，但更改只会保存在测试上下文中。
End of line sequence	为多行输入值选择行尾序列： **LF** - LF（换行）序列 **CRLF** - CRLF（回车换行）序列。

Parameter	Description
Preprocessing steps	会列出预处理步骤；点击 *Test* 按钮后，会显示每个步骤的测试结果。 测试结果在发送到前端时会被截断，最大大小为 512KB。测试结果可以复制（不超过截断后的 512KB）。 如果结果被截断，会显示警告图标。将鼠标悬停在警告上会显示说明。请注意，超过 512KB 的数据仍会由 Zabbix 服务器完整处理。 如果某个步骤在测试中失败，会显示错误图标。将鼠标悬停在错误上会显示说明。 如果为该步骤指定了"Custom on fail" 且执行了该操作，则会在预处理测试步骤行的下一行显示一条新记录，说明执行了什么操作以及产生了什么结果（错误或值）。
Result	当所有步骤一起测试时（即单击 *Test all steps* 按钮），无论何种情况，都会显示预处理步骤测试的最终结果。 还会显示转换为监控项值类型的类型转换结果，例如 `Result converted to Numeric (unsigned)`。 测试结果在发送到前端时会被截断，最大大小为 512KB。最终结果可以复制（不超过截断后的 512KB）。 如果结果被截断，会显示警告图标。将鼠标悬停在警告上会显示说明。请注意，超过 512KB 的数据仍会由 Zabbix 服务器完整处理。

单击 *Test* 可查看每个预处理步骤后的结果。

测试值会在单个步骤或所有步骤的测试会话之间保存，方便用户更改预处理步骤或监控项配置后返回测试窗口，而无需重新输入信息。不过，刷新页面后这些值会丢失。

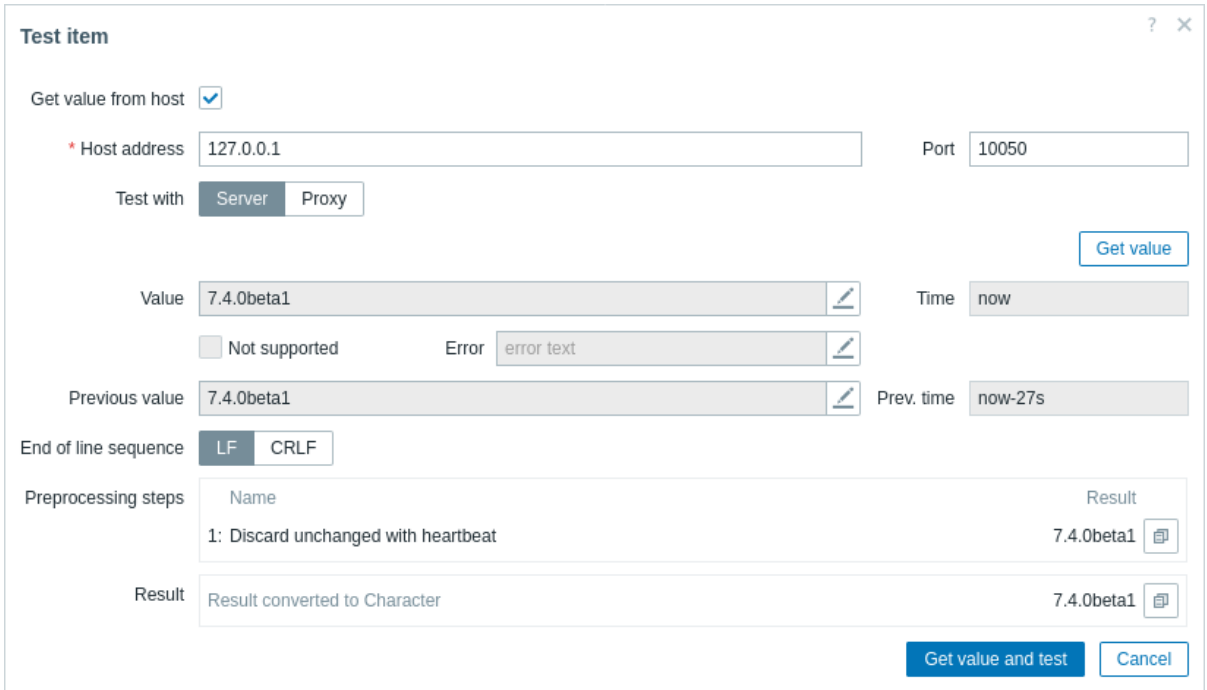
测试由 Zabbix 服务器执行。前端向服务器发送相应请求并等待结果。请求包含输入值和预处理步骤（已展开用户宏）。对于 *Change* 和 *Throttling* 步骤，可以指定可选的上一个值和时间。服务器会返回每个预处理步骤的结果。

所有技术错误或输入验证错误都会显示在测试窗口顶部的错误框中。

测试真实值

要针对真实值测试预处理：

- 勾选 从主机获取值复选框
- 输入或确认主机参数（主机地址、端口、proxy 名称/无 proxy）以及监控项特定的详细信息（例如 SNMPv2 团体名或 SNMPv3 安全凭据）。这些字段具有上下文感知能力：
  - 在可能的情况下，字段会预填充，例如，对于需要 agent 的监控项，会从主机选定的 agent 接口中获取信息
  - 对于模板监控项，必须手动填写这些值
  - 纯文本宏值会被解析
  - 如果字段值（或值的一部分）是机密信息或 Vault 宏，字段将为空，必须手动填写。如果任何监控项参数包含机密宏值，则会显示以下警告消息：" 监控项包含带有机密值的用户定义宏。这些宏的值应手动输入。"
  - 当监控项类型的上下文不需要这些字段时，字段将被禁用（例如，对于计算型监控项，主机地址和 proxy 字段会被禁用）
- 点击 获取值并测试以测试预处理



如果你在监控项配置表单中指定了值映射（显示值字段），监控项测试对话框将在最终结果之后显示另一行，名称为 应用值映射后的结果。

从主机获取真实值时特有的参数：

参数	描述
从主机获取值	勾选此复选框以从主机获取真实值。
主机地址	输入主机地址。 此字段会自动填充监控项主机接口的地址。
端口	输入主机端口。 此字段会自动填充监控项主机接口的端口。
SNMP 接口的其他字段 * *（*SNMP* 版本、*SNMP* 团体名、上下文名称等）	有关配置 SNMP 接口（v1、v2 和 v3）的更多详细信息，请参阅配置 SNMP 监控。 这些字段会自动从监控项主机接口中填充。
Proxy	如果主机由 proxy 监控，请指定 proxy。 此字段会自动填充主机的 proxy（如果有）。
值	从主机获取的值。 点击参数字段或查看/编辑按钮 将打开值或代码块的文本区域窗口。 值会被截断为最大 512KB，且仅在前端中执行。如果结果被截断，则会显示警告图标。将鼠标悬停在警告图标上时会显示警告描述。请注意，超过 512KB 的数据仍会由 Zabbix 服务器完整处理。

有关其余参数，请参阅上面的测试假设值。

预处理详情

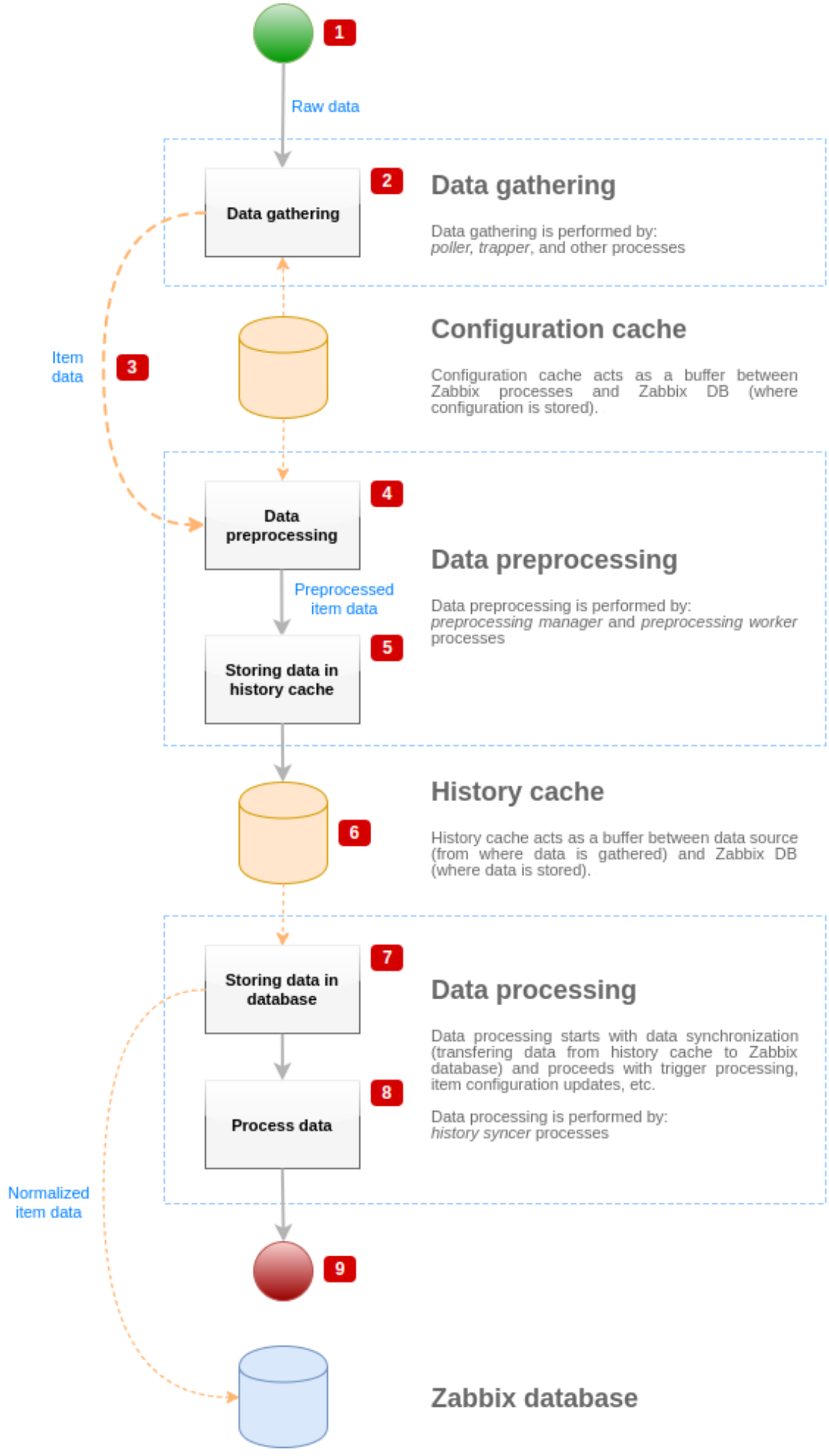
概述

本节介绍监控项值预处理的详细信息。监控项值预处理允许您为接收到的监控项值定义并执行转换规则。

预处理由预处理管理器进程以及执行预处理步骤的预处理工作进程共同管理。来自不同数据采集器的所有需要预处理的值，在添加到历史记录缓存之前，都会经过预处理管理器。数据采集器（轮询器、捕获器等）与预处理进程之间使用基于套接字的 IPC 通信。预处理步骤由 Zabbix 服务器或 Zabbix proxy（对于由 proxy 监控的监控项）执行。

监控项值处理

为了直观展示从数据源到 Zabbix 数据库的数据流，我们可以使用下图这个简化示意图：


1
Raw data
Data gathering
2
Data gathering
Data gathering is performed by:
poller, trapper, and other processes
Item data
3
Configuration cache
Configuration cache acts as a buffer between Zabbix processes and Zabbix DB (where configuration is stored).
Data preprocessing
4
Data preprocessing
Data preprocessing is performed by:
preprocessing manager and preprocessing worker processes
Preprocessed item data
Storing data in history cache
5
History cache
6
History cache acts as a buffer between data source (from where data is gathered) and Zabbix DB (where data is stored).
Storing data in database
7
Data processing
Data processing starts with data synchronization (transfering data from history cache to Zabbix database) and proceeds with trigger processing, item configuration updates, etc.
Process data
8
Data processing is performed by:
history syncer processes
Normalized item data
9
Zabbix database

上图仅以简化形式展示了与监控项值处理相关的进程、对象和操作。图中未显示条件性方向变化、错误处理或循环。预处理管理器的本地数据缓存也未显示，因为它不会直接影响数据流。该图的目的是展示参与监控项值处理的各个进程及其交互方式。

- 数据采集从数据源中的原始数据开始。此时，数据仅包含 ID、时间戳和值（也可以是多个值）。
- 无论使用哪种类型的数据采集器，其基本思路都是相同的，适用于主动检查、被动检查、trapper 监控项等；变化的只是数据格式以及通信的发起方（即数据采集器等待连接和数据，或由数据采集器发起通信并请求数据）。原始数据会先经过验证，然后从配置缓存中获取监控项配置（即为数据补充配置数据）。
- 基于套接字的 IPC 机制用于将数据从数据采集器传递到预处理管理器。此时，数据采集器会继续采集数据，而无需等待预处理管理器的响应。
- 执行数据预处理。这包括执行预处理步骤以及依赖监控项处理。

Note:
如果任一预处理步骤失败，监控项在执行预处理期间可能会变为 NOT SUPPORTED 状态。

- 预处理管理器本地数据缓存中的历史数据会被刷新到历史缓存中。
- 此时，数据流会停止，直到下一次历史缓存同步（即 history syncer 进程执行数据同步时）。
- 同步过程开始时，会先进行数据规范化，然后再将数据存储到 Zabbix 数据库中。数据规范化会将数据转换为所需的监控项类型（即监控项配置中定义的类型），其中包括根据这些类型允许的预定义大小对文本数据进行截断（字符串使用 HISTORY_STR_VALUE_LEN，文本使用 HISTORY_TEXT_VALUE_LEN，日志值使用 HISTORY_LOG_VALUE_LEN）。规范化完成后，数据将被发送到 Zabbix 数据库。

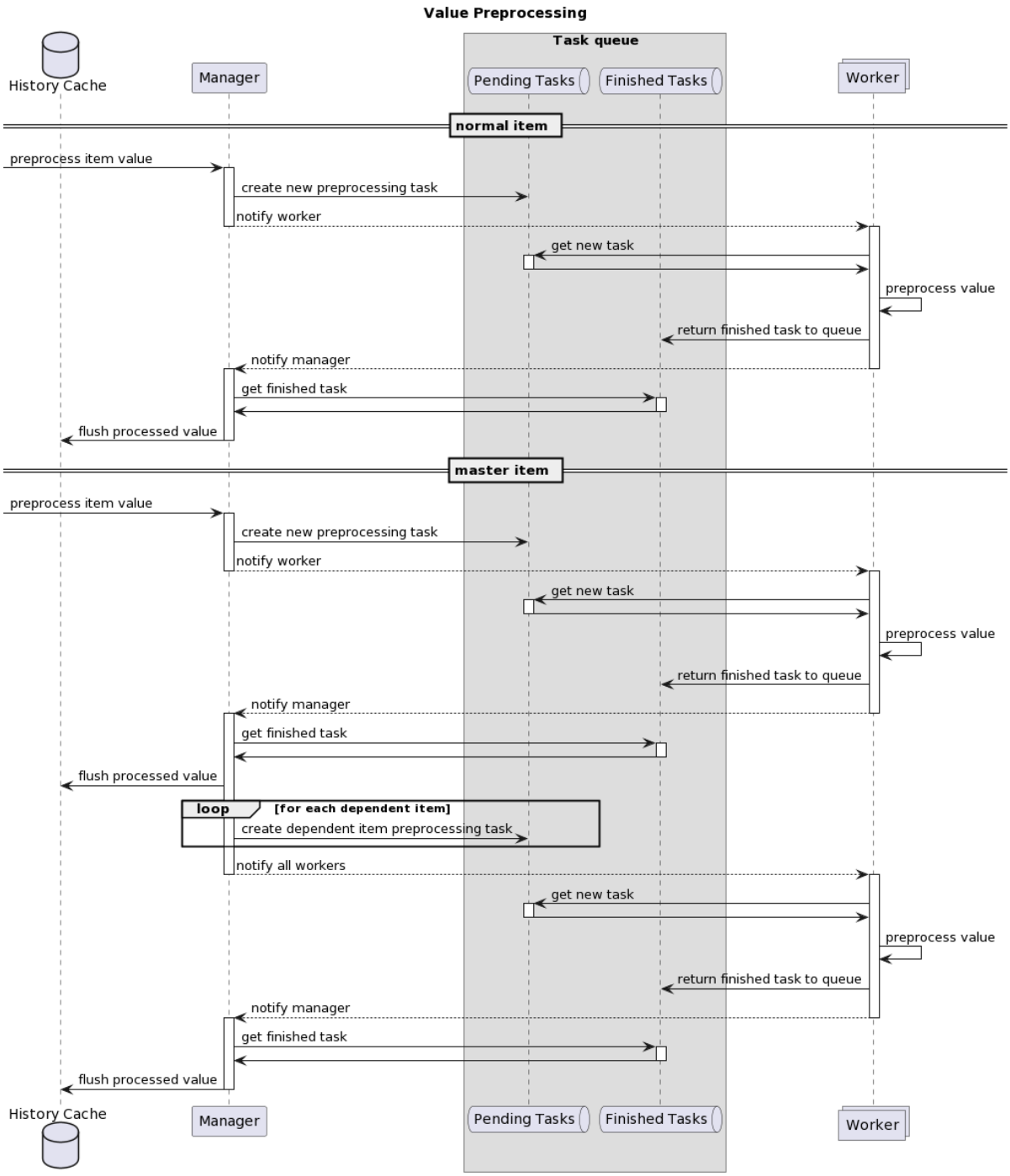
Note:
如果数据规范化失败，监控项可能会变为 NOT SUPPORTED 状态（例如，文本值无法转换为数字时）。

- 采集到的数据会被进一步处理——检查触发器、在监控项变为 NOT SUPPORTED 时更新监控项配置等。
- 从监控项值处理的角度来看，这就是数据流的终点。

监控项值预处理

数据预处理按以下步骤执行：

- 如果监控项既没有预处理也没有依赖监控项，则其值会被添加到历史缓存中，或发送给 LLD 管理器。否则，监控项值会通过基于 UNIX socket 的 IPC 机制传递给预处理管理器。
- 系统会创建一个预处理任务并将其添加到队列中，同时通知预处理 worker 有新任务。
- 此时，数据流会停止，直到至少有一个未被占用的预处理 worker（即未执行任何任务）。
- 当有可用的预处理 worker 时，它会从队列中获取下一个任务。
- 预处理完成后（无论预处理步骤执行失败还是成功），预处理后的值都会被添加到已完成任务队列中，并通知管理器有新的已完成任务。
- 预处理管理器会将结果转换为所需格式（由监控项值类型定义），并将其添加到历史缓存中或发送给 LLD 管理器。
- 如果已处理的监控项存在依赖监控项，则会将依赖监控项连同预处理后的主监控项值一起添加到预处理队列中。依赖监控项会绕过常规的值预处理请求直接入队，但仅适用于已设置值且不处于 NOT SUPPORTED 状态的主监控项。



请注意，图中对主监控项预处理过程做了少许简化，省略了预处理缓存。

预处理队列

预处理队列组织如下：

- 待处理任务列表：
  - 直接从值预处理请求中按接收顺序创建的任务
- 立即任务列表（在待处理任务之前处理）：
  - 测试任务（作为前端的监控项/预处理测试请求的响应而创建）
  - 依赖监控项任务
  - 顺序任务（必须按严格顺序执行的任务）：
    - 具有使用最后一个值的预处理步骤的任务：
      - 更改
      - 节流
      - JavaScript（字节码缓存）
    - 依赖监控项预处理缓存
- 已完成任务列表

预处理缓存

预处理缓存被引入，以改善具有相似预处理步骤的多个依赖监控项（这是常见的 LLD 结果）的预处理性能。

通过对一个依赖监控项进行预处理，并重用一些内部预处理数据，来为其余的依赖监控项提供缓存。预处理缓存仅支持以下类型的第一个预处理步骤：

- Prometheus 模式（按度量衡输入的索引）
- JSONPath（将数据解析为对象树，并索引第一个表达式 `[?(@.path == "value")]`）

预处理 worker

Zabbix 服务器配置文件允许用户设置预处理 worker 线程的数量。
应使用StartPreprocessors 配置参数来设置预启动的预处理 worker 实例数量，该数量至少应与可用 CPU 核心数相匹配。

如果预处理任务不受 CPU 限制，并且涉及频繁的网络请求，建议配置更多 worker。
预处理 worker 的最佳数量可以由多种因素决定，包括 “可预处理” 监控项的数量（即需要执行任何预处理步骤的监控项）、数据采集进程的数量、监控项预处理的平均步骤数等。
worker 不足可能导致内存使用率过高。有关 Zabbix 安装中内存使用率过高的排查，请参见使用 tcmalloc 分析过高的内存使用率。

但假设没有诸如解析大型 XML/JSON 数据块之类的高负载预处理操作，那么预处理 worker 的数量可以与数据采集器的总数相匹配。这样，通常（除非来自采集器的数据是批量到达）至少会有一个空闲的预处理 worker 可用于已采集的数据。

> **Warning:**
> 过多的数据采集进程（pollers、unreachable pollers、ODBC pollers、HTTP pollers、Java pollers、pingers、trappers、proxypollers）以及 IPMI manager、SNMP trapper 和预处理 worker，可能会耗尽预处理 manager 的每进程文件描述符限制。
>

 耗尽每进程文件描述符限制会导致 Zabbix 服务器停止，通常会在启动后不久发生，但有时也可能需要更长时间。
> 为避免此类问题，请查看Zabbix 服务器配置文件，以优化并发检查和进程数量。
> 此外，如有必要，请通过检查和调整系统限制，确保文件描述符限制设置得足够高。

值处理管道

监控项值处理由多个进程分多个步骤（或阶段）执行。这可能导致：

- 依赖监控项可以接收到值，而主监控项却不能。这可以通过以下用例实现：
  - 主监控项的值类型为 `UINT`（可使用 trapper 监控项），依赖监控项的值类型为 `TEXT`。
  - 主监控项和依赖监控项都不需要任何预处理步骤。
  - 应将文本值（例如”abc”）传递给主监控项。
  - 由于没有需要执行的预处理步骤，预处理管理器会检查主监控项是否不处于 NOT SUPPORTED 状态以及值是否已设置（两者都为真），然后将依赖监控项以与主监控项相同的值加入队列（因为没有预处理步骤）。
  - 当主监控项和依赖监控项都进入历史同步阶段时，主监控项会因值转换错误而变为 NOT SUPPORTED 状态（文本数据无法转换为无符号整数）。

因此，依赖监控项会接收到一个值，而主监控项则会将其状态更改为 NOT SUPPORTED。

- 依赖监控项接收到的值在主监控项历史记录中不存在。该用例与前一个非常相似，区别仅在于主监控项类型。例如，如果主监控项使用 `CHAR` 类型，那么主监控项的值会在历史同步阶段被截断，而依赖监控项将从主监控项的初始值（未截断的值）中接收其值。

预处理示例

概述

本节介绍了一些使用预处理步骤完成实际任务的示例。

过滤 VMware 事件日志记录

本示例使用匹配正则表达式预处理步骤，从 VMware 事件日志中过滤不必要的事件。

1. 在一个正常工作的 VMware Hypervisor 主机上，检查vmware.eventlog 监控项是否存在并正常工作。
请注意，如果在创建主机期间已关联VMware 模板，则该事件日志监控项可能已经存在于该 hypervisor 上。

2. 在 VMware Hypervisor 主机上，创建一个 *Log* 类型的依赖监控项，并将事件日志监控项设置为其主监控项。

3. 在该依赖监控项的 预处理选项卡中，单击 添加以创建一个预处理步骤，并从下拉列表中选择 匹配正则表达式。然后，指定以下模式之一：

- 用于过滤所有日志事件：

```
.* logged in .*
```

- 用于过滤在 “User” 之后包含用户名的行：

```
\bUser\s+\K\S+
```

Attention:
如果正则表达式未匹配，则该依赖监控项将变为不支持状态，并显示相应的错误消息。
为避免这种情况，请勾选 失败时自定义复选框，并选择一个选项，例如丢弃该值或设置自定义值。
请注意，被丢弃的值不会存储在数据库中；因此，不会对触发器进行评估，也不会生成趋势数据。

或者，您也可以使用正则表达式预处理步骤来提取匹配组并控制输出：

- 若要提取并输出包含 “logged in” 的整个日志事件，请指定以下参数：

```
Pattern: .*logged in.*
Output: \0
```

- 若要提取并输出 “User” 后面的用户名：

```
Pattern: User (.*?)(?=\ )
Output: \1
```

检查获取到的值类型

本示例使用自定义乘数 预处理步骤来检查获取到的监控项值类型是否为数值型。

在监控项的 预处理选项卡中，选择 自定义乘数预处理步骤，并指定以下参数（将获取到的值乘以 1）：

```
1
```

Attention:
如果预处理失败（例如，输入不是数值），则该监控项将变为不支持状态，并显示相应的错误消息。为避免这种情况，请勾选 失败时的自定义处理复选框，并选择一个选项，例如丢弃该值或设置一个自定义值。请注意，被丢弃 的值不会存储在数据库中；因此，不会评估触发器，也不会生成趋势数据。

检查不支持的值

本示例使用检查不支持的值预处理步骤来检查是否无法获取监控项值。

当 Zabbix 服务器/proxy 轮询进程尝试采集监控项值时，可能会：

- 返回有效结果。
- 返回一个起初看似有效、但之后可能变为不支持的结果（例如，由于预处理后的值类型不匹配）。
- 返回采集该值时的错误，导致监控项变为不支持。常见原因包括：
  - 未知的监控项键值（对于 Zabbix agent、简单检查或 Zabbix 内部监控项）
  - 未知的 OID（SNMP agent）、未知的传感器（IPMI agent）或没有 JMX 指标（JMX agent）
  - 无法读取 trap 文件（SNMP trap）
  - 找不到脚本（外部检查）
  - 没有这样的 URL（HTTP agent、Browser）
  - 登录失败（SSH agent、TELNET agent）
  - 公式语法无效（Calculated）、JavaScript 语法错误（Script）或无效的 SQL（Database monitor）

要检测并处理采集监控项值时的错误，可以使用检查不支持的值预处理步骤。
请注意，此步骤始终最先执行，并且只能检测在预处理开始之前发生的错误。

在监控项的预处理选项卡中，选择检查不支持的值预处理步骤，并指定以下参数之一：

- 对于任意错误：

```
Parameter: any error
```

- 对于包含”cannot connect” 的错误：

```
Parameter: error matches
Pattern: (?i)cannot connect
```

然后，使用失败时自定义选项来丢弃该值（在本例中即错误）、设置自定义值，或返回自定义错误消息。
请注意，被丢弃的值不会存储在数据库中；因此，不会评估触发器，也不会生成趋势数据。

JSONPath 功能

概述

本节概述了监控项值预处理步骤中支持的 JSONPath 功能。

JSONPath 由以点号分隔的段组成。一个段可以是简单单词的形式，表示 JSON 值名称；也可以是通配符字符（*）；或者是包含在方括号内的更复杂结构。方括号段前面的点号是可选的，可以省略。

JSONPath 示例	说明
`$.object.name`	返回 `object.name` 的内容。
`$.object['name']`	返回 `object.name` 的内容。
`$.object.['name']`	返回 `object.name` 的内容。
`$["object"]['name']`	返回 `object.name` 的内容。
`$.['object'].["name"]`	返回 `object.name` 的内容。
`$.object.history.length()`	返回 `object.history` 数组元素的数量。
`$[?(@.name == 'Object')].price.first()`	返回名称为 “Object” 的第一个对象中 `price` 属性的值。
`$[?(@.name == 'Object')].history.first().length()`	返回名称为 “Object” 的第一个对象中 history 数组元素的数量。
`$[?(@.price > 10)].length()`	返回价格大于 10 的对象数量。

另请参见：在 JSONPath 中转义 LLD 宏值中的特殊字符。

支持的段

段	描述
`<name>`	按名称匹配对象属性。
`*`	匹配所有对象属性。
`['<name>']`	按名称匹配对象属性。
`['<name>', '<name>', ...]`	按列出的任意名称匹配对象属性。
`[<index>]`	按索引匹配数组元素。
`[<number>, <number>, ...]`	按列出的任意索引匹配数组元素。
`[*]`	匹配所有对象属性或数组元素。
`[<start>:<end>]`	按定义的范围匹配数组元素： **<start>** - 要匹配的第一个索引（包含）；如果未指定，则匹配从开头开始的所有数组元素；如果为负数，则表示从数组末尾开始的起始偏移量； **<end>** - 要匹配的最后一个索引（不包含）；如果未指定，则匹配直到末尾的所有数组元素；如果为负数，则表示从数组末尾开始的起始偏移量。
`[?(<expression>)]`	通过应用过滤表达式匹配对象/数组元素。

要在忽略其祖先的情况下查找匹配的段（分离段），必须在其前面加上两个点（`..`）。例如，`$..name` 或 `$..['name']` 会返回所有 `name` 属性的值。

可以通过在 JSONPath 后添加波浪号（`~`）后缀来提取匹配元素的名称。它会返回匹配对象的名称，或匹配数组项的字符串格式索引。输出格式遵循与其他 JSONPath 查询相同的规则——确定路径的结果按原样返回，而不确定路径的结果则以数组形式返回。不过，提取匹配确定路径的元素名称意义不大，因为该名称本身已经是已知的。

过滤表达式

过滤表达式是使用中缀表示法的算术表达式。

支持的操作数：

操作数	说明

`"<text>"`
`'<text>'`
	文本常量。
	
	示例：
	`'value: \'1\''`
	`"value: '1'"`

`<number>`	支持科学计数法的数值常量。 示例：123
`<jsonpath starting with $>`	由输入文档根节点中的 JSONPath 引用的值；仅支持确定性路径。 示例：`$.object.name`

操作数	说明
<jsonpath starting with @>	由当前对象/元素中的 JSONPath 引用的值；仅支持确定性路径。 示例：@.name

支持的运算符：| 运算符 | 类型 | 说明 | 结果 | |-|---|---|---| |-|Binary| 减法 |Number| |+|Binary| 加法 |Number| |/|Binary| 除法 |Number| |*|Binary| 乘法 |Number| |==|Binary| 相等 |Boolean (1/0)| |!=|Binary| 不等 |Boolean (1/0)| |<|Binary| 小于 |Boolean (1/0)| |<=|Binary| 小于或等于 |Boolean (1/0)| |>|Binary| 大于 |Boolean (1/0)| |>=|Binary| 大于或等于 |Boolean (1/0)| |=~|Binary| 匹配正则表达式 |Boolean (1/0)| |!|Unary| 布尔非 |Boolean (1/0)| ||||Binary| 布尔或 |Boolean (1/0)| |&&|Binary| 布尔与 |Boolean (1/0)|

函数

函数可用于 JSONPath 的末尾。如果前一个函数返回的值可被后一个函数接受，则可以链式调用多个函数。

支持的函数：

函数	描述	输入	输出
avg	输入数组中数字的平均值	数字数组	数字
min	输入数组中数字的最小值	数字数组	数字
max	输入数组中数字的最大值	数字数组	数字
sum	输入数组中数字的总和	数字数组	数字
length	输入数组中的元素个数	数组	数字
first	数组的第一个元素	数组	取决于输入数组内容的 JSON 结构（对象、数组、值）

JSONPath 聚合函数接受带引号的数值。当需要聚合时，这些值会自动从字符串转换为数值类型。不兼容的输入将导致函数生成错误。

输出值

JSONPath 可以分为确定路径和不确定路径。确定路径只能返回 null 或单个匹配项。不确定路径可以返回多个匹配项：具有分离的、多个名称/索引列表、数组切片或表达式段的 JSONPath。但是，当使用函数时，JSONPath 变为确定，因为函数始终输出单个值。

确定路径返回它引用的对象/数组/值。相反，不确定路径返回匹配对象/数组/值的数组。

> **Attention:**
> 由于内部优化方法的原因，JSONPath 查询结果中的属性顺序可能与原始 JSON 属性顺序不一致。例如，JSONPath `$.books[1]["author", "title"]` 可能返回 `["title", "author"]`。如果保留原始属性顺序很重要，则应考虑使用替代的查询后处理方法。

路径格式化规则

在方括号表示法的段和表达式中可以使用空格（空格、制表符），例如：`$[ 'a' ][ 0 ][ ?( $.b == 'c' ) ][ : -1 ].first()`。

字符串应该用单引号（'）或双引号（"）括起来。在字符串内部，单引号或双引号（取决于用哪个来括起来）和反斜杠（\）需要用反斜杠（\）进行转义。

示例

```
{
  "books": [
    {
      "category": "reference",
      "author": "Nigel Rees",
      "title": " 世纪名言",
      "price": 8.95,
      "id": 1
    },
    {
      "category": "fiction",
      "author": "Evelyn Waugh",
      "title": " 荣誉之剑",
      "price": 12.99,
      "id": 2
    },
    {
```

```
      "category": "fiction",
      "author": "Herman Melville",
      "title": " 白鲸",
      "isbn": "0-553-21311-3",
      "price": 8.99,
      "id": 3
    },
    {
      "category": "fiction",
      "author": "J. R. R. Tolkien",
      "title": " 指环王",
      "isbn": "0-395-19395-8",
      "price": 22.99,
      "id": 4
    }
  ],
  "services": {
    "delivery": {
      "servicegroup": 1000,
      "description": " 本地城镇次日送达",
      "active": true,
      "price": 5
    },
    "bookbinding": {
      "servicegroup": 1001,
      "description": "A5 格式图书印刷与装订",
      "active": true,
      "price": 154.99
    },
    "restoration": {
      "servicegroup": 1002,
      "description": " 各种修复方法",
      "active": false,
      "methods": [
        {
          "description": " 化学清洁",
          "price": 46
        },
        {
          "description": " 压平受潮损坏的页面",
          "price": 24.5
        },
        {
          "description": " 重新装订破损的书籍",
          "price": 99.49
        }
      ]
    }
  },
  "filters": {
    "price": 10,
    "category": "fiction",
    "no filters": "no \"filters\""
  },
  "closed message": " 商店已关闭",
  "tags": [
    "a",
    "b",
    "c",
    "d",
    "e"
  ]
```

```
}
```

JSONPath	Type	Result
`$.filters.price`	definite	10
`$.filters.category`	definite	fiction
`$.filters['no filters']`	definite	no ”filters”

`$.filters`	definite	{
		”price”: 10,
		”category”: ”fiction”,
		”no filters”: ”no \”filters\””
		}

`$.books[1].title`	definite	Sword of Honour
`$.books[-1].author`	definite	J. R. R. Tolkien
`$.books.length()`	definite	4
`$.tags[:]`	indefinite	[”a”, ”b”, ”c”, ”d”, ”e”]
`$.tags[2:]`	indefinite	[”c”, ”d”, ”e”]
`$.tags[:3]`	indefinite	[”a”, ”b”, ”c”]
`$.tags[1:4]`	indefinite	[”b”, ”c”, ”d”]
`$.tags[-2:]`	indefinite	[”d”, ”e”]
`$.tags[:-3]`	indefinite	[”a”, ”b”]
`$.tags[:-3].length()`	definite	2
`$.books[0, 2].title`	indefinite	[”Moby Dick”, ”Sayings of the Century”]
`$.books[1]['author', "title"]`	indefinite	[”Sword of Honour”, ”Evelyn Waugh”]
`$..id`	indefinite	[1, 2, 3, 4]
`$.services..price`	indefinite	[154.99, 5, 46, 24.5, 99.49]
`$.books[?(@.id == 4 - 0.4 * 5)].title`	indefinite	[”Sword of Honour”] 注：此查询显示，查询中可以使用算术运算；它可以简化为 `$.books[?(@.id == 2)].title`
`$.books[?(@.id == 2 || @.id == 4)].title`	indefinite	[”Sword of Honour”, ”The Lord of the Rings”]
`$.books[?(!(@.id == 2))].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]
`$.books[?(@.id != 2)].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]
`$.books[?(@.title =~ " of ")].title`	indefinite	[”Sayings of the Century”, ”Sword of Honour”, ”The Lord of the Rings”]
`$.books[?(@.price > 12.99)].title`	indefinite	[”The Lord of the Rings”]
`$.books[?(@.author > "Herman Melville")].title`	indefinite	[”Sayings of the Century”, ”The Lord of the Rings”]
`$.books[?(@.price > $.filters.price)].title`	indefinite	[”Sword of Honour”, ”The Lord of the Rings”]
`$.books[?(@.category == $.filters.category)].title`	indefinite	[”Sword of Honour”,”Moby Dick”,”The Lord of the Rings”]
`$.books[?(@.category == "fiction" && @.price < 10)].title`	indefinite	[”Moby Dick”]

JSONPath	Type	Result

`$..[?(@.id)]`	indefinite	[
		{
		”price”: 8.95,
		”id”: 1,
		”category”: ”reference”,
		”author”: ”Nigel Rees”,
		”title”: ”Sayings of the Century”
		},
		{
		”price”: 12.99,
		”id”: 2,
		”category”: ”fiction”,
		”author”: ”Evelyn Waugh”,
		”title”: ”Sword of Honour”
		},
		{
		”price”: 8.99,
		”id”: 3,
		”category”: ”fiction”,
		”author”: ”Herman Melville”,
		”title”: ”Moby Dick”,
		”isbn”: ”0-553-21311-3”
		},
		{
		”price”: 22.99,
		”id”: 4,
		”category”: ”fiction”,
		”author”: ”J. R. R. Tolkien”,
		”title”: ”The Lord of the Rings”,
		”isbn”: ”0-395-19395-8”
		}
		]

`$.services..[?(@.price > 50)].description`	indefinite	[”A5 格式图书印刷与装订”, ” 重新装订破损的书籍”]
`$..id.length()`	definite	4
`$.books[?(@.id == 2)].title.first()`	definite	Sword of Honour
`$..tags.first().length()`	definite	5 注：`$..tags` 是一个不确定路径，因此它返回匹配元素的数组，即 `[["a", "b", "c", "d", "e" ]]`；`first()` 返回第一个元素，即 `["a", "b", "c", "d", "e"]`；`length()` 计算该元素的长度，即 5。
`$.books[*].price.min()`	definite	8.95
`$..price.max()`	definite	154.99
`$.books[?(@.category == "fiction")].price.avg()`	definite	14.99
`$.books[?(@.category == $.filters.xyz)].title`	indefinite	注：没有匹配项的查询对确定性和不确定性路径都返回 NULL。
`$.services[?(@.active=="true")].servicegroup`	indefinite	[1001,1000] 注：在布尔值比较中必须使用文本常量。
`$.services[?(@.active=="false")].servicegroup`	indefinite	[1002] 注：在布尔值比较中必须使用文本常量。

`$.services[?(@.servicegroup==1002)]~.first()`	definite	{
		”name”: ”Service 3”,
		”active”: ”false”,
		”servicegroup”: 1002
		}

在 JSONPath 中转义 LLD 宏值中的特殊字符

当在 JSONPath 预处理中过滤低级别发现宏并解析其值时，会应用以下特殊字符转义规则：

- 仅将反斜杠（\）和双引号（”）视为需要转义的字符；
- 如果解析后的宏值包含这些字符，则每个字符都使用反斜杠进行转义；
- 如果这些字符已经被反斜杠转义，则不视为已完成转义，反斜杠和其后的特殊字符都会再次转义一次。

例如：

JSONPath	LLD 宏值	替换后
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]
	c:\temp	$.[?(@.value == ”c:\\temp”)]
	a\\b	$.[?(@.value == ”a\\\\b”)]

在表达式中使用时，可能包含特殊字符的宏应放在双引号中：

JSONPath	LLD 宏值	替换后	结果
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]	OK
$.[?(@.value == {#MACRO})]		$.[?(@.value == special \”value\”)]	错误的 **JSONPath** 表达式

在路径中使用时，可能包含特殊字符的宏应放在方括号 和双引号中：

JSONPath	LLD 宏值	替换后	结果
$.[”{#MACRO}”].value	c:\temp	$.[”c:\\temp”].value	OK
$.{#MACRO}.value		$.c:\\temp.value	错误的 **JSONPath** 表达式

JavaScript 预处理

概述

本节提供使用 JavaScript 进行预处理的详细信息。

> **Attention:**
> 请勿在预处理 JavaScript 中使用未声明的赋值。请使用 `var` 声明局部变量。

JavaScript 预处理

JavaScript 预处理通过调用带有单个参数’value’ 和用户提供的函数体的 JavaScript 函数来完成。
预处理步骤的结果是该函数返回的值，例如，要执行华氏度到摄氏度的转换，请输入：

```
return (value - 32)  * 5 / 9
```

在 JavaScript 预处理参数中，服务器会将其包装为一个 JavaScript 函数：

```
function (value)
{
   return (value - 32) * 5 / 9
}
```

输入参数’value’ 始终以字符串形式传递。
返回值会通过 toString() 方法自动转换为字符串（如果转换失败，则返回错误的字符串值），但有少数例外：

- 返回 undefined 值将导致错误；
- 返回 null 值会导致输入值被丢弃，类似于’Custom on fail’ 操作中的’Discard value’ 预处理。

可以通过抛出值/对象来返回错误（通常是字符串或 Error 对象）。

例如：

```
if (value == 0)
    throw "Zero input value"
return 1/value
```

每个脚本都有 10 秒的执行超时限制（具体取决于脚本，超时触发可能会更晚）；超过该时间将返回错误。
强制执行 512 MB 的堆内存限制。

JavaScript 预处理步骤的字节码会被缓存，并在下次应用该步骤时复用。
对监控项预处理步骤的任何更改都会导致缓存的脚本被重置，并在之后重新编译。

连续运行失败（连续 3 次）会导致引擎重新初始化，以降低某个脚本破坏后续脚本执行环境的可能性（此操作会在 DebugLevel 4 及以上级别记录）。

JavaScript 预处理由 Duktape JavaScript 引擎实现。

> **Note:**
> 内置 JavaScript 对象的方法定义不能在监控项预处理脚本中被重新定义。

另请参见：Additional JavaScript objects and global functions

在脚本中使用宏

可以在 JavaScript 代码中使用用户宏（以及在低级别发现上下文中使用的LLD macros）。如果脚本包含用户宏，这些宏会在执行特定预处理步骤之前由服务器/proxy 解析。请注意，在前端中测试预处理步骤时，不会拉取宏值，需要手动输入。

> **Note:**
> 当宏被替换为其值时，上下文会被忽略。宏值会按原样插入到代码中，无法在将该值放入 JavaScript 代码之前额外添加转义。请注意，这在某些情况下可能会导致 JavaScript 错误。

在下面的示例中，如果接收到的值超过 {$THRESHOLD} 宏值，则将返回阈值（如果存在）而不是接收到的值：

```
var threshold = '{$THRESHOLD}';
return (!isNaN(threshold) && value > threshold) ? threshold : value;
```

示例

以下示例说明了如何使用 JavaScript 预处理。

每个示例包含一个简要描述、JavaScript 预处理参数的函数体以及预处理步骤的结果 - 函数返回的值。

示例 1：将数字（科学计数法转换为整数）

将数字”2.62128e+07” 从科学计数法转换为整数。

```
return (Number(value))
```

函数返回的值：26212800。

示例 2：将数字转换为十进制（二进制转十进制）

将二进制数”11010010” 转换为十进制数。

```
return(parseInt(value,2))
```

函数返回的值：210。

示例 3：四舍五入

将数字”18.2345” 四舍五入为 2 位小数。

```
return(Math.round(value* 100) / 100)
```

函数返回的值：18.23。

示例 4：计算字符串中的字母数

计算字符串”Zabbix” 中的字母数。

```
return (value.length)
```

函数返回的值：6。

示例 5：获取剩余时间

获取距离证书到期日期（Feb 12 12:33:56 2022 GMT）的剩余时间（以秒为单位）。

```
var split = value.split(' '),
    MONTHS_LIST = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    month_index = ('0' + (MONTHS_LIST.indexOf(split[0]) + 1)).slice(-2),
```

```
    ISOdate = split[3] + '-' + month_index + '-' + split[1] + 'T' + split[2],
    now = Date.now();

return parseInt((Date.parse(ISOdate) - now) / 1000);
```

函数返回的值：44380233。

示例 6：移除 JSON 属性

通过删除具有键为"data_size" 或"index_size" 的属性来修改 JSON 数据结构。

```
var obj=JSON.parse(value);

for (i = 0; i < Object.keys(obj).length; i++) {
    delete obj[i]["data_size"];
    delete obj[i]["index_size"];
}

return JSON.stringify(obj)
```

函数接受的值：

```
[
    {
        "table_name":"history",
        "data_size":"326.05",
        "index_size":"174.34"
    },
    {
        "table_name":"history_log",
        "data_size":"6.02",
        "index_size":"3.45"
    }
]
```

函数返回的值：

```
[
    {
        "table_name":"history"
    },
    {
        "table_name":"history_log"
    }
]
```

示例 7：将 Apache 状态转换为 JSON

将从web.page.get Zabbix agent 监控项接收到的值（例如，web.page.get[http://127.0.0.1:80/server-status?auto]）转换为 JSON 对象。

```
// 将 Apache 状态转换为 JSON

// 将值拆分为子字符串，并将这些子字符串放入数组中
var lines = value.split('\n');

// 创建一个空对象“output”
var output = {};

// 创建一个具有预定义属性的对象“workers”
var workers = {
    '_': 0, 'S': 0, 'R': 0, 'W': 0,
    'K': 0, 'D': 0, 'C': 0, 'L': 0,
    'G': 0, 'I': 0, '.': 0
};

// 将“lines”数组中的子字符串作为属性（键值对）添加到“output”对象中
```

```
for (var i = 0; i < lines.length; i++) {
    var line = lines[i].match(/([A-z0-9 ]+): (.*)/);

    if (line !== null) {
        output[line[1]] = isNaN(line[2]) ? line[2] : Number(line[2]);
    }
}

// 多版本指标
output.ServerUptimeSeconds = output.ServerUptimeSeconds || output.Uptime;
output.ServerVersion = output.ServerVersion || output.Server;

// 解析“Scoreboard”属性以获取工作进程计数
if (typeof output.Scoreboard === 'string') {
    for (var i = 0; i < output.Scoreboard.length; i++) {
        var char = output.Scoreboard[i];

        workers[char]++;
    }
}

// 将工作进程数据添加到“output”对象中
output.Workers = {
    waiting: workers['_'], starting: workers['S'], reading: workers['R'],
    sending: workers['W'], keepalive: workers['K'], dnslookup: workers['D'],
    closing: workers['C'], logging: workers['L'], finishing: workers['G'],
    cleanup: workers['I'], slot: workers['.']
};

// 返回 JSON 字符串
return JSON.stringify(output);
```

函数接受的值：

```
HTTP/1.1 200 OK
Date: Mon, 27 Mar 2023 11:08:39 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 405
Content-Type: text/plain; charset=ISO-8859-1

127.0.0.1
ServerVersion: Apache/2.4.52 (Ubuntu)
ServerMPM: prefork
Server Built: 2023-03-08T17:32:01
CurrentTime: Monday, 27-Mar-2023 14:08:39 EEST
RestartTime: Monday, 27-Mar-2023 12:19:59 EEST
ParentServerConfigGeneration: 1
ParentServerMPMGeneration: 0
ServerUptimeSeconds: 6520
ServerUptime: 1 hour 48 minutes 40 seconds
Load1: 0.56
Load5: 0.33
Load15: 0.28
Total Accesses: 2476
Total kBytes: 8370
Total Duration: 52718
CPUUser: 8.16
CPUSystem: 3.44
CPUChildrenUser: 0
CPUChildrenSystem: 0
CPULoad: .177914
```

```
Uptime: 6520
ReqPerSec: .379755
BytesPerSec: 3461.58
BytesPerReq: 3461.58
DurationPerReq: 21.2916
BusyWorkers: 2
IdleWorkers: 6
Scoreboard: ____KW__.....................................................................................
```

函数返回的值：


```
{
    "Date": "Mon, 27 Mar 2023 11:08:39 GMT",
    "Server": "Apache/2.4.52 (Ubuntu)",
    "Vary": "Accept-Encoding",
    "Encoding": "gzip",
    "Length": 405,
    "Type": "text/plain; charset=ISO-8859-1",
    "ServerVersion": "Apache/2.4.52 (Ubuntu)",
    "ServerMPM": "prefork",
    "Server Built": "2023-03-08T17:32:01",
    "CurrentTime": "Monday, 27-Mar-2023 14:08:39 EEST",
    "RestartTime": "Monday, 27-Mar-2023 12:19:59 EEST",
    "ParentServerConfigGeneration": 1,
    "ParentServerMPMGeneration": 0,
    "ServerUptimeSeconds": 6520,
    "ServerUptime": "1 hour 48 minutes 40 seconds",
    "Load1": 0.56,
    "Load5": 0.33,
    "Load15": 0.28,
    "Total Accesses": 2476,
    "Total kBytes": 8370,
    "Total Duration": 52718,
    "CPUUser": 8.16,
    "CPUSystem": 3.44,
    "CPUChildrenUser": 0,
    "CPUChildrenSystem": 0,
    "CPULoad": 0.177914,
    "Uptime": 6520,
    "ReqPerSec": 0.379755,
    "BytesPerSec": 1314.55,
    "BytesPerReq": 3461.58,
    "DurationPerReq": 21.2916,
    "BusyWorkers": 2,
    "IdleWorkers": 6,
    "Scoreboard": "____KW__.....................................................................................
    "Workers": {
        "waiting": 6,
        "starting": 0,
        "reading": 0,
        "sending": 1,
        "keepalive": 1,
        "dnslookup": 0,
        "closing": 0,
        "logging": 0,
        "finishing": 0,
        "cleanup": 0,
        "slot": 142
    }
}
```


其他 JavaScript 对象

概述

本节介绍了 Zabbix 对使用 Duktape 实现的 JavaScript 语言所做的扩展，以及受支持的全局 JavaScript 函数。

Attention:
请勿在预处理 JavaScript 中使用未声明的赋值。请使用 `var` 声明局部变量。

内置对象

Zabbix

Zabbix 对象提供与 Zabbix 内部功能的交互。

Method	Description
`log(loglevel, message)`	使用 <loglevel> 日志级别将 <message> 写入 Zabbix 日志（请参见配置文件中的 DebugLevel 参数）。

示例：

```
Zabbix.log(3, "this is a log entry written with 'Warning' log level")
```

您可以使用以下别名：

Alias	Alias to
console.log(object)	Zabbix.log(4, JSON.stringify(object))
console.warn(object)	Zabbix.log(3, JSON.stringify(object))
console.error(object)	Zabbix.log(2, JSON.stringify(object))

Attention:
所有已记录消息的总大小在每次脚本执行时限制为 8 MB。

Method	Description
`sleep(delay)`	将 JavaScript 执行延迟 `delay` 毫秒。

示例（将执行延迟 15 秒）：

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

HttpRequest

此对象封装了 cURL 句柄，可用于发起简单的 HTTP 请求。发生错误时会抛出异常。

Attention:
每次脚本执行初始化的 `HttpRequest` 对象数量最多为 10 个。

Method	Description
`addHeader(value)`	添加 HTTP 标头字段。该字段将用于后续所有请求，直到使用 `clearHeader()` 方法将其清除。 单个 `HttpRequest` 对象可添加的标头字段总长度限制为 128 Kbytes（包括特殊字符和标头名称）。
`clearHeader()`	清除 HTTP 标头。如果未设置标头字段，则当要发布的数据采用 JSON 格式时，`HttpRequest` 会将 Content-Type 设置为 application/json；否则设置为 text/plain。
`connect(url)`	向 URL 发送 HTTP CONNECT 请求并返回响应。
`customRequest(method, url, data)`	允许在第一个参数中指定任意 HTTP 方法。向 URL 发送该方法请求，并可选择性地携带 *data* 负载，然后返回响应。
`delete(url, data)`	向 URL 发送 HTTP DELETE 请求，并可选择性地携带 *data* 负载，然后返回响应。

Method	Description
getHeaders(<asArray>)	返回接收到的 HTTP 标头字段对象。 asArray 参数可以设置为"true"（例如 getHeaders(true)）、"false"，也可以不定义。设置为"true" 时，接收到的 HTTP 标头字段值将以数组形式返回；在获取多个同名标头的字段值时应使用此设置。 未设置或设置为"false" 时，接收到的 HTTP 标头字段值将以字符串形式返回。
get(url, data)	向 URL 发送 HTTP GET 请求，并可选择性地携带 *data* 负载，然后返回响应。
head(url)	向 URL 发送 HTTP HEAD 请求并返回响应。
options(url)	向 URL 发送 HTTP OPTIONS 请求并返回响应。
patch(url, data)	向 URL 发送 HTTP PATCH 请求，并可选择性地携带 *data* 负载，然后返回响应。
put(url, data)	向 URL 发送 HTTP PUT 请求，并可选择性地携带 *data* 负载，然后返回响应。
post(url, data)	向 URL 发送 HTTP POST 请求，并可选择性地携带 *data* 负载，然后返回响应。
getStatus()	返回上次 HTTP 请求的状态码。
setProxy(proxy)	将 HTTP proxy 设置为"proxy" 值。如果此参数为空，则不使用 proxy。
setHttpAuth(bitmask, username, password)	在'bitmask' 参数中设置启用的 HTTP 身份验证方法（HTTPAUTH_BASIC、HTTPAUTH_DIGEST、HTTPAUTH_NEGOTIATE、HTTPAUTH_NTLM、HTTPAUTH_NONE）。 HTTPAUTH_NONE 标志允许禁用 HTTP 身份验证。 示例： request.setHttpAuth(HTTPAUTH_NTLM | HTTPAUTH_BASIC, username, password) request.setHttpAuth(HTTPAUTH_NONE)
trace(url, data)	向 URL 发送 HTTP TRACE 请求，并可选择性地携带 *data* 负载，然后返回响应。

示例：

```
try {
    Zabbix.log(4, 'jira webhook script value='+value);

    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new HttpRequest(),
    fields = {},
    resp;

    req.addHeader('Content-Type: application/json');
    req.addHeader('Authorization: Basic '+params.authentication);

    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {"key": params.project_key};
    fields.issuetype = {"id": params.issue_id};
    resp = req.post('https://jira.example.com/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.getStatus() != 201) {
        throw 'Response code: '+req.getStatus();
    }

    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;
} catch (error) {
    Zabbix.log(4, 'jira issue creation failed json : '+JSON.stringify({"fields": fields}));
    Zabbix.log(4, 'jira issue creation failed : '+error);

    result = {};
}
```

```
return JSON.stringify(result);
```

XML

XML 对象允许在监控项和低级发现预处理中以及 webhook 中处理 XML 数据。

> **Attention:**
> 要使用 XML 对象，服务器/proxy 必须在编译时启用 libxml2 支持。

Method	Description
XML.query(data, expression)	使用 XPath 检索节点内容。如果未找到节点，则返回 null。 **expression** - XPath 表达式； **data** - 字符串形式的 XML 数据。
XML.toJson(data)	将 XML 格式的数据转换为 JSON。
XML.fromJson(object)	将 JSON 格式的数据转换为 XML。

示例：

输入：

```
<menu>
    <food type = "breakfast">
        <name>Chocolate</name>
        <price>$5.95</price>
        <description></description>
        <calories>650</calories>
    </food>
</menu>
```

输出：

```
{
    "menu": {
        "food": {
            "@type": "breakfast",
            "name": "Chocolate",
            "price": "$5.95",
            "description": null,
            "calories": "650"
        }
    }
}
```

序列化规则

XML 到 JSON 的转换将根据以下规则进行处理（对于 JSON 到 XML 的转换，则应用相反的规则）：

1. XML 属性将被转换为键，且其名称前会加上'@' 前缀。

示例：

输入：

```
<xml foo="FOO">
  <bar>
    <baz>BAZ</baz>
  </bar>
</xml>
```

输出：

```
{
  "xml": {
    "@foo": "FOO",
    "bar": {
      "baz": "BAZ"
    }
```

```
  }
}
```

2. 自闭合元素（<foo/>）将被转换为值为'null' 的元素。

示例：

输入：

```
<xml>
  <foo/>
</xml>
```

输出：

```
{
  "xml": {
    "foo": null
  }
}
```

3. 空属性（值为""）将被转换为值为空字符串（''）的属性。

示例：

输入：

```
<xml>
  <foo bar="" />
</xml>
```

输出：

```
{
  "xml": {
    "foo": {
      "@bar": ""
    }
  }
}
```

4. 具有相同元素名称的多个子节点将被转换为单个键，其值为一个值数组。

示例：

输入：

```
<xml>
  <foo>BAR</foo>
  <foo>BAZ</foo>
  <foo>QUX</foo>
</xml>
```

输出：

```
{
  "xml": {
    "foo": ["BAR", "BAZ", "QUX"]
  }
}
```

5. 如果文本元素没有属性也没有子元素，它将被转换为字符串。

示例：

输入：

```
<xml>
    <foo>BAZ</foo>
</xml>
```

输出：

```
{
  "xml": {
    "foo": "BAZ"
   }
}
```

6. 如果文本元素没有子元素但有属性，则文本内容将被转换为一个键为'#text'、值为该内容的元素；属性将按照序列化规则 1 中所述进行转换。

示例：

输入：

```
<xml>
  <foo bar="BAR">
    BAZ
  </foo>
</xml>
```

输出：

```
{
  "xml": {
    "foo": {
      "@bar": "BAR",
      "#text": "BAZ"
    }
  }
}
```

全局 JavaScript 函数

已使用 Duktape 实现了额外的全局 JavaScript 函数：

- `btoa(data)` - 将数据编码为 base64 字符串。
- `atob(base64_string)` - 将 base64 字符串解码为 Uint8Array 缓冲区。

```
try {
    b64 = btoa("test string");
    buffer = atob(b64);

    // Note that decoding logic depends on the data format of the buffer.
    decoded = String.fromCharCode.apply(this, [].slice.call(buffer));
}
catch (error) {
    return {'error.name' : error.name, 'error.message' : error.message};
}
```

- `md5(data)` - 计算数据的 MD5 哈希值。
- `sha256(data)` - 计算数据的 SHA256 哈希值。
- `hmac('<hash type>',key,data)` - 以十六进制格式字符串形式返回 HMAC 哈希值；`hash type` 支持 `md5` 和 `sha256`；`key` 和 `data` 参数支持二进制数据。

  示例：

  - `hmac('md5',key,data)`
  - `hmac('sha256',key,data)`

- `sign(hash,key,data)` - 以字符串形式返回计算出的签名（使用 SHA-256 的 RSA 签名），其中：
 hash - 仅允许使用 sha256，否则将抛出错误。
 key - 私钥。它应符合 PKCS#1 或 PKCS#8 标准。密钥可以通过不同形式提供：

  - 使用空格代替换行
  - 使用转义或未转义的 \n 代替换行
  - 不包含任何换行，作为单行字符串
  - 作为 JSON 格式字符串

  密钥也可以从用户宏/密文宏/vault 中加载。

  data - 将被签名的数据。它可以是字符串（也支持二进制数据）或缓冲区（Uint8Array/ArrayBuffer）。

示例：

```
- sign('sha256',key,data)
```

使用 OpenSSL 或 GnuTLS 计算签名。如果 Zabbix 在构建时未包含这些加密库中的任何一个，则会抛出错误（'missing OpenSSL or GnuTLS library'）。

浏览器监控项 JavaScript 对象

概述

本节描述了使用 Duktape 实现的 Zabbix 对 JavaScript 语言的扩展，用于在浏览器项目脚本中使用。这些扩展补充了附加 JavaScript 对象页面中描述的 JavaScript 对象。

Browser

`Browser` 对象用于管理 WebDriver 会话，在创建时初始化会话，并在销毁时终止会话。单个脚本最多可支持四个 `Browser` 对象。

要构造 `Browser` 对象，请使用 `new Browser(options)` 语法。`options` (*JSON object*) 参数用于指定浏览器选项，通常为 WebDriver options 方法的返回结果（例如 `Browser.chromeOptions()`）。

`Browser` 对象支持以下方法。

Method	Description
`navigate(url)`	导航到指定的 URL。 参数： `url` - (string) 要导航到的 URL。
`getUrl()`	返回已打开页面 URL 的字符串。
`getPageSource()`	返回已打开页面源代码的字符串。
`findElement(strategy, selector)`	返回一个`Element` 对象，表示已打开页面中的单个元素（如果没有元素匹配 `strategy` 和 `selector`，则返回 `null`）。 参数： `strategy` - (string, css selector/link text/partial link text/tag name/xpath) 定位策略； `selector` - (string) 使用指定定位策略的元素选择器。
`findElements(strategy, target)`	返回一个包含多个`Element` 对象的数组，表示已打开页面中的多个元素（如果没有元素匹配定位策略和 target，则返回空数组）。 参数： `strategy` - (string, css selector/link text/partial link text/tag name/xpath) 定位策略； `target` - (string) 使用指定定位策略的元素选择器。
`getCookies()`	返回一个`Cookie` 对象数组。
`addCookie(cookie)`	设置 Cookie。 参数： `cookie` - (`Cookie` object) 要设置的 Cookie。
`getScreenshot()`	返回一个 base64 编码的字符串，表示浏览器视口的图像。
`setScreenSize(x,y)`	设置浏览器视口大小。 参数： `x` - (string) 以像素为单位的宽度； `y` - (string) 以像素为单位的高度。
`setScriptTimeout(timeout)`	设置脚本加载超时时间。 参数： `timeout` - (integer) 以毫秒为单位的超时值。
`setSessionTimeout(timeout)`	设置会话（页面加载）超时时间。 参数： `timeout` - (integer) 以毫秒为单位的超时值。
`setElementWaitTimeout(timeout)`	设置元素定位策略的（隐式）超时时间。 参数： `timeout` - (integer) 以毫秒为单位的超时值。

Method	Description
collectPerfEntries(mark)	收集性能条目，以便通过 getResult() 方法获取。 参数： mark - (string, optional) 性能快照标记。
getRawPerfEntries()	返回一个性能条目对象数组。
getResult()	返回一个Result 对象，其中包含浏览器会话统计信息（错误信息、性能快照等）。
getError()	返回一个包含浏览器错误的 BrowserError 对象（如果没有浏览器错误，则返回 null）。
setError(message)	设置一条自定义错误消息，以包含在Result 对象中。 参数： message - (string) 错误消息。
discardError()	丢弃将在Result 对象中返回的错误。
getAlert()	返回一个包含浏览器警报的Alert 对象（如果没有浏览器警报，则返回 null）。
chromeOptions()	返回一个包含预定义 Chrome 浏览器选项的 chromeOptions 对象。
firefoxOptions()	返回一个包含预定义 Firefox 浏览器选项的 firefoxOptions 对象。
safariOptions()	返回一个包含预定义 Safari 浏览器选项的 safariOptions 对象。
edgeOptions()	返回一个包含预定义 Edge 浏览器选项的 edgeOptions 对象。
switchFrame(target)	切换到指定的 frame。 参数： target - （浏览器元素或 integer，可选）目标 frame。要按元素选择 frame，请传入该元素。要按索引选择 frame，请传入数字。如果留空，则切换到顶层浏览上下文。

所有 Browser 方法都可能抛出以下错误：

- BrowserError - 派生自 Error 对象；如果 Browser 构造函数失败，则会抛出该错误；包含一个额外的 browser 属性，其值为抛出此 BrowserError 的 Browser 对象。
- WebdriverError - 派生自 BrowserError；包含与 BrowserError 对象相同的属性，用于指示该错误是否是在响应 WebDriver 返回错误时生成的。

元素

Element 对象由Browser 对象的 findElement()/findElements() 方法返回，且不能直接构造。

Element 对象表示网页中的一个元素，并提供与其交互的方法。

Element 对象支持以下方法。

方法	描述
getAttribute(name)	返回元素属性的属性值字符串（如果未找到指定属性，则返回 null）。 参数： name - (string) 属性名。
getProperty(name)	返回元素属性的属性值字符串（如果未找到指定属性，则返回 null）。 参数： name - (string) 属性名。
getText()	返回元素文本的文本值字符串。
click()	单击元素。
clear()	清除可编辑元素的内容。
sendKeys(keys)	发送按键。 参数： keys - (string) 要发送的按键。

Cookie

Cookie 对象由Browser 对象的 getCookies() 方法返回，并传递给 addCookie() 方法。

虽然 Cookie 对象没有任何方法，但它可以包含以下属性：

属性	Type	说明
name	string	Cookie 的名称。
value	string	Cookie 的值。
path	string	Cookie 有效的路径。 添加 Cookie 时如果省略，默认为 "/"。
domain	string	Cookie 可见的域。 添加 Cookie 时如果省略，默认为会话当前浏览上下文的活动文档的 URL 域。
secure	boolean	指示 Cookie 是否为安全 Cookie。 添加 Cookie 时如果省略，默认为 false。
httpOnly	boolean	指示 Cookie 是否仅限 HTTP。 添加 Cookie 时如果省略，默认为 false。
expiry	integer	Cookie 的过期时间（自 Unix 纪元以来的秒数）。添加 Cookie 时如果省略，则不得设置此属性。
sameSite	string	Cookie 的 sameSite 属性，用于控制 Cookie 是否应限制为第一方或同站点上下文。 可设置为 "Lax" 或 "Strict"。 添加 Cookie 时如果省略，默认为 "None"。

Alert

Alert 对象表示网页警报，由Browser对象的 getAlert() 方法返回，不能直接构造。

Alert 对象包含具有警报文本的 text 属性（如果没有警报，则为 null）。

以下方法与 Alert 对象一起使用。

方法	描述
accept()	接受警报。
dismiss()	关闭警报。

Result

Result 对象包含会话统计信息，并由Browser 对象的 getResult() 方法返回。

通常，Result 对象会被转换为字符串并由脚本返回，然后通过预处理解析为依赖项监控项值。

虽然 Result 对象没有任何方法，但它可以包含以下属性。

Property			Type	Description
duration			string	从会话创建到结果检索之间的会话持续时间。
error			object	错误信息。
	http_status		integer	WebDriver 返回的 HTTP 状态（如果没有 WebDriver 错误，则为 0）。
	error_code		string	WebDriver 返回的错误（如果没有 WebDriver 错误，则为空字符串）。
	message		string	WebDriver 错误消息（如果没有 WebDriver 错误，则为空字符串）。
performance_data			object	性能统计信息。
	summary		object	性能摘要。
		navigation	object	导航摘要。
		resource	object	资源摘要。
	details		array of objects	每次可能导致导航的操作后的性能统计信息。
		mark	string	*(可选)* 使用collectPerfEntries() 方法指定的性能快照标记。
		navigation	object	导航统计信息。
		resource	object	此步骤的资源摘要。
		user	array of objects	标记/度量类型统计信息数组。
	marks		array of objects	已标记的性能快照索引。
		name	string	性能快照标记名称。
		index	integer	details 数组中的性能快照索引。

CSV 到 **JSON** 预处理

概述

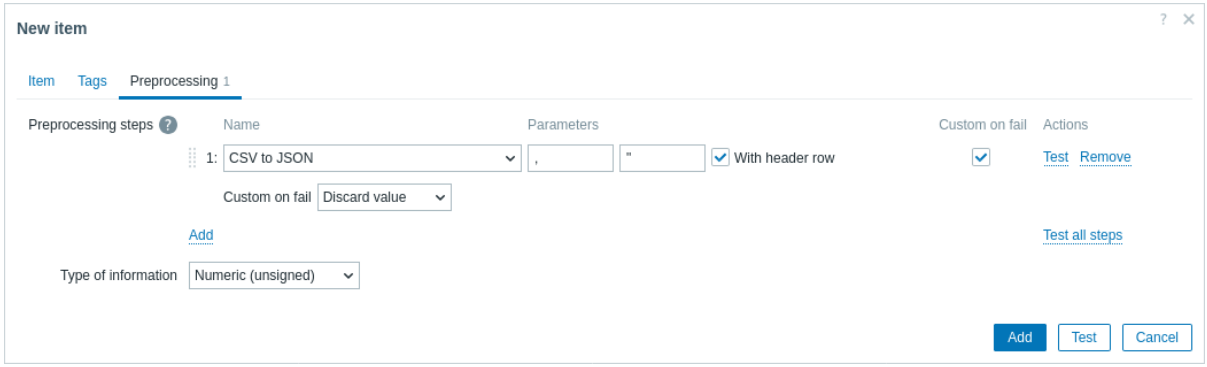
在此预处理步骤中，可以将 CSV 文件数据转换为 JSON 格式。支持以下对象：

- 监控项（监控项原型）
- 低级别发现规则

配置

要配置 CSV 转 JSON 预处理步骤：

1. 转到监控项/发现规则配置中的预处理选项卡。
2. 单击添加。
3. 选择 *CSV 转 JSON* 选项。



第一个参数允许您设置自定义分隔符。请注意，如果 CSV 输入的第一行以"Sep=" 开头，后面紧跟一个 UTF-8 字符，则在未设置第一个参数的情况下，该字符将用作分隔符。如果未设置第一个参数，且未从"Sep=" 行中获取分隔符，则使用逗号作为分隔符。

第二个可选参数允许您设置引号符号。

如果选中包含标题行复选框，标题行中的值将被解释为列名（有关更多信息，请参阅标题处理）。

如果选中失败时自定义处理复选框，则在预处理步骤失败时，监控项不会变为 "不支持"。此外，还可以设置自定义错误处理选项：丢弃值、设置指定值或设置指定的错误消息。

标题预处理

CSV 文件标题行可以用两种不同的方式处理:

- 如果勾选 *With header row* - 标题行值被解释为列名。在这种情况下，列名必须是唯一的，数据行不应该包含比标题行更多的列；
- 如果勾选 *With header row* - 标题行解释为数据。自动生成列名 (1,2,3,4...)

CSV 文件示例:

```
Nr,Item name,Key,Qty
1,active agent item,agent.hostname,33
"2","passive agent item","agent.version","44"
3,"active,passive agent items",agent.ping,55
```

Note:
输入中的引号字段中的引号字符必须在其前面加上另一个引号字符进行转义。

有标题行预处理

有标题行预处理时期望 JSON 的输出:

```
[
   {
      "Nr":"1",
      "Item name":"active agent item",
      "Key":"agent.hostname",
      "Qty":"33"
   },
   {
      "Nr":"2",
      "Item name":"passive agent item",
      "Key":"agent.version",
      "Qty":"44"
   },
   {
      "Nr":"3",
      "Item name":"active,passive agent items",
```

```
        "Key":"agent.ping",
        "Qty":"55"
    }
]
```

无标题行预处理

无标题行预处理时期望 JSON 的输出:

```
[
    {
        "1":"Nr",
        "2":"Item name",
        "3":"Key"
        "4":"Qty"
    },
    {
        "1":"1",
        "2":"active agent item",
        "3":"agent.hostname"
        "4":"33"
    },
    {
        "1":"2",
        "2":"passive agent item",
        "3":"agent.version"
        "4":"44"
    },
    {
        "1":"3",
        "2":"active,passive agent items",
        "3":"agent.ping"
        "4":"55"
    }
]
```

监控项类型

概述

监控项类型涵盖了从系统获取数据的各种方法。每种监控项类型都有自己支持的监控项键值和必需参数。

Zabbix 当前提供以下监控项类型：

- Zabbix agent
  - 日志文件监控
- 简单检查
  - VMware 监控
- SNMP agent
- SNMP trap
- Zabbix 内部
- Zabbix trapper
- 外部检查
- 数据库监控
- HTTP agent
  - Prometheus 检查
- IPMI agent
- SSH agent
- Telnet agent
- JMX agent
- 计算监控项
- 依赖监控项
- 脚本监控项
- 浏览器监控项
- 遥测查询

本节的子页面中包含所有监控项类型的详细信息。尽管监控项类型提供了许多数据采集选项，但还可以通过用户参数 或可加载模块提供更多选项。

某些检查仅由 Zabbix 服务器执行（作为无 agent 监控），而其他检查则需要 Zabbix agent，甚至需要 Zabbix Java 网关（用于 JMX 监控）。

Attention:
如果某种监控项类型需要特定的接口（例如，IPMI 检查需要主机上存在 IPMI 接口），则该接口必须存在于主机定义中。

主机定义中可以设置多个接口：Zabbix agent、SNMP agent、JMX 和 IPMI。如果监控项可以使用多个接口，则会按顺序（Agent→SNMP→JMX→IPMI）搜索可用的主机接口，并关联找到的第一个适用接口。

所有返回文本的监控项（字符、日志和文本信息类型）也可以仅返回空白字符（如果适用），将返回值设置为空字符串（自 2.0 版本起支持）。

Zabbix agent

概述

本节提供使用与 Zabbix agent 通信进行数据采集的监控项键的详细信息。

Zabbix agent 检查分为被动和主动两种。在配置监控项时，您可以选择所需的类型：

- *Zabbix agent* - 用于被动检查
- *Zabbix agent (active)* - 用于主动检查

请注意，Zabbix agent 支持的所有监控项键也同样受新一代 Zabbix agent 2 支持。请参阅仅可与 agent 2 一起使用的附加监控项键。

支持的监控项键

下面列出了可与 Zabbix agent 一起使用的监控项键。

监控项键以不带参数和附加信息的形式列出。单击监控项键可查看完整详细信息。

监控项键	描述	监控项组
kernel.maxfiles	操作系统支持的最大打开文件数。	内核
kernel.maxproc	操作系统支持的最大进程数。	
kernel.openfiles	当前打开的文件描述符数量。	
log	监控日志文件。	日志监控
log.count	受监控日志文件中匹配行的数量。	
logrt	监控已轮转的日志文件。	
logrt.count	已轮转的受监控日志文件中匹配行的数量。	
modbus.get	读取 Modbus 数据。	Modbus
net.dns	检查 DNS 服务的状态。	网络
net.dns.perf	检查 DNS 服务的性能。	
net.dns.record	执行 DNS 查询。	
net.if.collisions	窗口外冲突的数量。	
net.if.discovery	网络接口列表。	
net.if.get	包含详细信息的网络接口列表。	
net.if.in	网络接口的入站流量统计信息。	
net.if.out	网络接口的出站流量统计信息。	
net.if.total	网络接口入站和出站流量统计信息的总和。	
net.tcp.listen	检查此 TCP 端口是否处于 LISTEN 状态。	
net.tcp.port	检查是否可以与指定端口建立 TCP 连接。	
net.tcp.service	检查服务是否正在运行并接受 TCP 连接。	
net.tcp.service.perf	检查 TCP 服务的性能。	
net.tcp.socket.count	返回与参数匹配的 TCP 套接字数量。	
net.udp.listen	检查此 UDP 端口是否处于 LISTEN 状态。	
net.udp.service	检查服务是否正在运行并响应 UDP 请求。	
net.udp.service.perf	检查 UDP 服务的性能。	
net.udp.socket.count	返回与参数匹配的 UDP 套接字数量。	
proc.cpu.util	进程 CPU 使用率百分比。	进程
proc.get	操作系统进程及其参数的列表。	
proc.mem	进程使用的内存大小（以字节为单位）。	
proc.num	进程数量。	
sensor	硬件传感器读数。	传感器
system.boottime	系统启动时间。	系统

监控项键	描述	监控项组
system.cpu.discovery	已检测到的 CPU/CPU 核心列表。	
system.cpu.intr	设备中断。	
system.cpu.load	CPU 负载。	
system.cpu.num	CPU 数量。	
system.cpu.switches	上下文切换次数。	
system.cpu.util	CPU 使用率百分比。	
system.hostname	系统主机名。	
system.hw.chassis	机箱信息。	
system.hw.cpu	CPU 信息。	
system.hw.devices	PCI 或 USB 设备列表。	
system.hw.macaddr	MAC 地址列表。	
system.localtime	系统时间。	
system.run	在主机上运行指定命令。	
system.stat	系统统计信息。	
system.sw.arch	软件架构信息。	
system.sw.os	操作系统信息。	
system.sw.os.get	有关操作系统的详细信息（版本、类型、发行版名称、次版本和主版本等）。	
system.sw.packages	已安装软件包的列表。	
system.sw.packages.get	已安装软件包的详细列表。	
system.swap.in	换入（从设备到内存）统计信息。	
system.swap.out	换出（从内存到设备）统计信息。	
system.swap.size	交换空间大小（以字节为单位或占总量的百分比）。	
system.uname	系统标识信息。	
system.uptime	系统运行时间（以秒为单位）。	
system.users.num	已登录用户数量。	
vfs.dev.discovery	块设备及其类型的列表。	虚拟文件系统
vfs.dev.get	包含详细信息的块设备列表。	
vfs.dev.read	磁盘读取统计信息。	
vfs.dev.write	磁盘写入统计信息。	
vfs.dir.count	目录项数量。	
vfs.dir.get	目录项列表。	
vfs.dir.size	目录大小。	
vfs.file.cksum	使用 UNIX cksum 算法计算的文件校验和。	
vfs.file.contents	获取文件内容。	
vfs.file.exists	检查文件是否存在。	
vfs.file.get	返回有关文件的信息。	
vfs.file.md5sum	文件的 MD5 校验和。	
vfs.file.owner	获取文件所有者。	
vfs.file.permissions	返回包含 UNIX 权限八进制数的 4 位字符串。	
vfs.file.regexp	获取文件中的字符串。	
vfs.file.regmatch	在文件中查找字符串。	
vfs.file.size	文件大小。	
vfs.file.time	文件时间信息。	
vfs.fs.discovery	已挂载文件系统的列表，包括其类型和挂载选项。	
vfs.fs.get	已挂载文件系统的列表，包括其类型和挂载选项；在 *full* 模式下，还包括可用磁盘空间和 inode 统计信息。	
vfs.fs.inode	inode 数量或百分比。	
vfs.fs.size	磁盘空间（以字节为单位或占总量的百分比）。	
vm.memory.size	内存大小（以字节为单位或占总量的百分比）。	虚拟内存
web.page.get	获取网页内容。	Web 监控
web.page.perf	完整网页的加载时间。	
web.page.regexp	在网页中查找字符串。	
agent.hostmetadata	agent 主机元数据。	Zabbix
agent.hostname	agent 主机名。	
agent.ping	agent 可用性检查。	
agent.variant	Zabbix agent 的变体（Zabbix agent 或 Zabbix agent 2）。	
agent.version	Zabbix agent 的版本。	
zabbix.stats	返回 Zabbix 服务器或 proxy 的一组内部指标。如果指定了 `ip` 和 `port`，则从远程实例获取指标；否则从本地实例获取。	

监控项键	描述	监控项组
zabbix.stats (queue)	返回 Zabbix 服务器或 proxy 队列中延迟的受监控监控项数量。如果指定了 ip 和 port，则从远程实例获取指标；否则从本地实例获取。	

支持的平台

除非在监控项详情中另有说明，否则 agent 监控项（以及所有参数）支持以下平台：

- **Linux**
- **FreeBSD**
- **Solaris**
- **HP-UX**
- **AIX**
- **MacOS X**
- **OpenBSD**
- **NetBSD**

许多 agent 监控项也支持 **Windows**。
详情请参见Windows agent 监控项 页面。

监控项键值详情

不带尖括号的参数为必填参数。
带有尖括号 **< >** 标记的参数为可选参数。

kernel.maxfiles

 操作系统支持的最大打开文件数。
 返回值：整数。
 支持的平台：Linux, FreeBSD, MacOS X, OpenBSD, NetBSD.

kernel.maxproc

 操作系统支持的最大进程数。
 返回值：整数。
 支持的平台：Linux 2.6 及更高版本，FreeBSD，Solaris，MacOS X，OpenBSD，NetBSD。

kernel.openfiles

 当前打开文件描述符的数量。
 返回值：整数。
 支持的平台：Linux（该项可能在其他类 UNIX 平台上也可以工作）。

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent dir>]

 监控日志文件。
 返回值：*Log*。
 请参阅支持的平台。

参数：

- **file** - 日志文件的完整路径和名称；
- **regexp** - 描述所需模式的正则表达式；
- **encoding** - 代码页标识符；
- **maxlines** - agent 每秒发送到 Zabbix 服务器或 proxy 的最大新行数。此参数会覆盖`zabbix_agentd.conf` 中 `MaxLinesPerSecond` 的值；
- **mode** - 可选值：
  - *all*（默认）；
  - *skip* - 跳过对较旧数据的处理（仅影响新创建的监控项）；
- **output** - 可选的输出格式化模板。**\0** 转义序列会被匹配到的文本部分替换（从匹配开始的第一个字符到匹配结束的字符），而 **\N**（其中 N=1...9）转义序列会被第 N 个匹配组替换（如果 N 超过捕获组数量，则替换为空字符串）；
- **maxdelay** - 最大延迟，单位为秒。类型：float。取值：
  - 0 -（默认）从不忽略日志文件行；
  - > 0.0 - 忽略较旧的行，以便在”maxdelay” 秒内分析到最新的行。在使用之前请先阅读maxdelay 说明！
- **options** - 其他选项：
mtime-noreread - 非唯一记录，仅在文件大小变化时重新读取（忽略修改时间变化）。（自 5.0.2 起此参数已弃用，因为现在会忽略 mtime。）
- **persistent dir**（仅适用于 Unix 系统上的 zabbix_agentd；Zabbix agent 2 不支持）- 用于存储持久化文件的目录绝对路径。另请参阅关于持久化文件的附加说明。

注释：

- 该监控项必须配置为主动检查。
- 如果文件缺失或权限不允许访问，该监控项将变为不受支持。
- 如果 `output` 为空，则返回包含匹配文本的整行。请注意，除 *Matches regular expression* 之外的所有全局正则表达式类型始终返回整条匹配行，并且会忽略 `output` 参数。
- 使用 `output` 参数进行内容提取是在 agent 上完成的。

示例：

```
log[/var/log/syslog]
log[/var/log/syslog,error]
log[/home/zabbix/logs/logfile,,,100]
```

使用 output 参数从日志记录中提取数字的示例：

```
log[/app1/app.log,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,,\1] # 此监控项将匹配日
```

使用 output 参数在发送到服务器之前重写日志记录的示例：

```
log[/app1/app.log,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",,,,"\1
```

log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent dir>]

 监控日志文件中匹配行的数量。
 返回值：*Integer*。
 参见supported platforms。

参数：

- **file** - 日志文件的完整路径和名称；
- **regexp** - 描述所需模式的正则expression；
- **encoding** - 代码页identifier；
- **maxproclines** - agent 每秒将分析的新行最大数量(不能超过 10000)。默认值为zabbix_agentd.conf 中 MaxLinesPerSecond 的 10 倍。
- **mode** - 可能的值：
  - *all*（默认）；
  - *skip* - 跳过处理较旧的数据（仅影响新创建的监控项）。
- **maxdelay** - 最大延迟（秒）。类型：float。取值：
  - 0 -（默认）从不忽略日志文件行；
  - > 0.0 - 为了让在”maxdelay” 秒内分析到最新的行，忽略较旧的行。使用前请先阅读maxdelay 说明！
- **options** - 其他选项：
mtime-noreread - 非唯一记录，仅在文件大小发生变化时重新读取（忽略修改时间变化）。(由于现在会忽略 mtime，因此该参数自 5.0.2 起已弃用。）
- **persistent dir**（仅适用于 Unix 系统上的 zabbix_agentd；Zabbix agent 2 不支持）- 用于存储持久化文件的目录绝对路径。另请参见关于persistent files 的附加说明。

注释：

- 该监控项必须配置为active check。
- 匹配行会统计 agent 自上次日志检查以来新增的行，因此取决于监控项更新间隔。
- 如果文件不存在或权限不允许访问，该监控项将变为不受支持。

logrt[file regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent dir>]

 监控已轮转的日志文件。
 返回值：*Log*。
 请参见支持的平台。

参数：

- **file regexp** - 文件的绝对路径，文件名使用正则表达式指定。请注意，正则表达式仅适用于文件名，不需要匹配整个名称（例如，/path/to/agent 将匹配 zabbix_agentd.log)。
- **regexp** - 描述所需内容模式的正则表达式。
- **encoding** - 代码页标识符。
- **maxlines** - agent 每秒发送到 Zabbix 服务器或 proxy 的最大新行数。此参数会覆盖zabbix_agentd.conf 中 MaxLinesPerSecond 的值。
- **mode** - 可选值：
  - *all*（默认）；
  - *skip* - 跳过较旧数据的处理（仅影响新创建的监控项）。
- **output** - 可选的输出格式化模板。**\0** 转义序列会被匹配到的文本部分替换（从匹配开始的第一个字符到匹配结束的字符），而 **\N**（其中 N=1...9）转义序列会被第 N 个匹配组替换（如果 N 超过捕获组数量，则替换为空字符串）。
- **maxdelay** - 最大延迟时间，单位为秒。类型：float。取值：
  - 0 -（默认）从不忽略日志文件行；
  - > 0.0 - 为了在”maxdelay” 秒内分析到最新的行，忽略较旧的行。在使用之前请阅读maxdelay 说明！
- **options** - 日志文件轮转类型及其他选项。可选值：
  - *rotate*（默认）。
  - *copytruncate* - 请注意，*copytruncate* 不能与 *maxdelay* 一起使用。在这种情况下，*maxdelay* 必须为 0 或未指定；请参见copytruncate 说明。
  - *mtime-reread* - 非唯一记录，如果修改时间或大小发生变化则重新读取（如果未显式设置任何 mtime-* 选项，则默认使用）。
  - *mtime-noreread* - 非唯一记录，仅在大小发生变化时重新读取（忽略修改时间变化）。
- **persistent dir**（仅适用于 Unix 系统上的 zabbix_agentd；Zabbix agent 2 不支持）- 用于存储持久化文件的目录绝对路径。另请参见关于持久化文件的补充说明。

说明：

- 该监控项必须配置为主动检查。
- 日志轮转基于文件的最后修改时间。
- 请注意，logrt 设计用于处理一个当前处于活动状态的日志文件，以及若干匹配到的已轮转非活动文件。例如，如果某个目录中有许多活动日志文件，则应为每个文件分别创建一个 logrt 监控项。否则，如果一个 logrt 监控项拾取了过多文件，可能会导致内存耗尽并使监控崩溃。
- 如果 output 为空，则返回包含匹配文本的整行内容。请注意，除 *Matches regular expression* 之外的所有全局正则表达式类型始终返回整条匹配行，并且会忽略 output 参数。
- 使用 output 参数进行内容提取是在 agent 上完成的。
- 在 file regexp 参数中，日志目录路径和日志文件正则表达式必须使用正确的目录分隔符分隔：
  - 在 Windows 上，分隔符必须是反斜杠 (\)；其他位置可以容忍使用正斜杠，但分隔日志目录路径和日志文件正则表达式的那个位置除外（请参见下面的示例）。
  - 在其他系统上，必须使用正斜杠 (/)。

Windows 示例：

```
logrt["c:/dir1/dir2/dir3\filename.*\.log","pattern_to_match"] #this item will collect data from log files
logrt["//example.com/share/dir1/dir2/dir3\filename.*\.log","pattern_to_match"] #this item will collect dat
```

其他系统示例：

```
logrt["/home/zabbix/logs/^logfile[0-9]{1,3}$",,,100] #this item will match a file like "logfile1" (will no
logrt["/home/user/^logfile_.*_[0-9]{1,3}$","pattern_to_match","UTF-8",100] #this item will collect data fr
```

使用 output 参数从日志记录中提取数字的示例：

```
logrt[/app1/^test.*log$,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,,\1] #this ite
```

使用 output 参数在发送到服务器之前重写日志记录的示例：

```
logrt[/app1/^test.*log$,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",
```

logrt.count[file regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent dir>]

 监控的轮转日志文件中匹配行的数量。
 返回值：*Integer*。
 请参阅支持的平台。

参数：

- **file regexp** - 文件的绝对路径，文件名使用正则表达式指定。请注意，正则表达式仅适用于文件名，不需要匹配整个名称（例如，/path/to/agent 将匹配 zabbix_agentd.log）。
- **regexp** - 描述所需模式的正则表达式。
- **encoding** - 代码页标识符。
- **maxproclines** - agent 每秒将分析的新行最大数量(不能超过 10000)。默认值为zabbix_agentd.conf 中 MaxLinesPerSecond 的 10 倍。
- **mode** - 可选值：
  - *all*（默认）；
  - *skip* - 跳过处理较旧的数据（仅影响新创建的监控项）。
- **maxdelay** - 最大延迟（秒）。类型：float。取值：
  - 0 -（默认）从不忽略日志文件行；
  - > 0.0 - 忽略较旧的行，以便在”maxdelay” 秒内分析最新的行。在使用前请先阅读maxdelay 说明！
- **options** - 日志文件轮转类型及其他选项。可选值：
  - *rotate*（默认）。
  - *copytruncate* - 请注意，copytruncate 不能与 maxdelay 一起使用。在这种情况下，maxdelay 必须为 0 或未指定；请参阅copytruncate 说明。
  - *mtime-reread* - 非唯一记录，如果修改时间或大小发生变化则重新读取（默认）。
  - *mtime-noreread* - 非唯一记录，仅在大小发生变化时重新读取（忽略修改时间变化）。
- **persistent dir**（仅适用于 Unix 系统上的 zabbix_agentd；Zabbix agent 2 不支持）- 用于存储持久化文件的目录绝对路径。另请参阅关于持久化文件的附加说明。

备注：

- 该监控项必须配置为主动检查。
- 匹配行的计数基于 agent 上次日志检查之后的新行，因此取决于监控项更新间隔。
- 日志轮转基于文件的最后修改时间。
- 在 file regexp 参数中，日志目录路径和日志文件正则表达式必须使用正确的目录分隔符分隔：
  - 在 Windows 上，分隔符必须是反斜杠 (\)；其他位置可以容忍正斜杠，但分隔日志目录路径和日志文件正则表达式的那个位置除外（见下方示例）。
  - 在其他系统上，必须使用正斜杠 (/)。

Windows 示例：

```
logrt.count["c:/dir1/dir2/dir3\filename.*\.log","pattern_to_match"] # 此监控项将统计位于 "c:/dir1/dir2/dir3"
logrt.count["//example.com/share/dir1/dir2/dir3\filename.*\.log","pattern_to_match"] # 此监控项将统计网络共享
```

其他系统示例：

```
logrt.count["/home/zabbix/logs/.*","^logfile[0-9]{1,3}$",,100] # 此监控项将统计 "/home/zabbix/logs" 目录中日志
logrt.count["/home/user/^logfile_.*_[0-9]{1,3}$","pattern_to_match","UTF-8",100] # 此监控项将统计位于 "/home/
```

modbus.get[endpoint,<slave id>,<function>,<address>,<count>,<type>,<endianness>,<offset>]

 读取 Modbus 数据。
 返回值：*JSON* 字符串。
 支持的平台：Linux、Windows。

参数：

- **endpoint** - 以 `protocol://connection_string` 定义的端点；
- **slave id** - 从站 ID；
- **function** - Modbus 功能码；
- **address** - 第一个寄存器、线圈或输入的地址；
- **count** - 要读取的记录数；
- **type** - 数据类型；
- **endianness** - 字节序配置；
- **offset** - 从 `address` 开始的寄存器数量，其结果将被丢弃。

在 Windows 上，此监控项会遵循 `endpoint` 参数中的 UNC 路径，这可能带来安全风险。

有关参数的详细说明。

net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]

 检查 DNS 服务的状态。
 返回值：- 0 - DNS 解析失败（DNS 服务器未响应或返回错误）；- 1 - DNS 解析成功。
 参见supported platforms。

参数：

- **ip**（在 Windows 上忽略，除非使用 Zabbix agent 2）- DNS 服务器的 IP 地址（留空则使用默认 DNS 服务器）；
- **name** - 要查询的 DNS 名称；
- **type** - 要查询的记录类型（默认是 SOA）；
- **timeout**（在 Windows 上忽略，除非使用 Zabbix agent 2）- 请求超时时间，单位为秒（默认是 1 秒）；
- **count**（在 Windows 上忽略，除非使用 Zabbix agent 2）- 请求的重试次数（默认是 2）；
- **protocol** - 用于执行 DNS 查询的协议：*udp*（默认）或 *tcp*。

注释：

- `type` 的可选值为：*ANY*、*A*、*NS*、*CNAME*、*MB*、*MG*、*MR*、*PTR*、*MD*、*MF*、*MX*、*SOA*、*NULL*、*WKS*（Zabbix agent on Windows 不支持，所有操作系统上的 Zabbix agent 2 也不支持）、*HINFO*、*MINFO*、*TXT*、*SRV*。
- 对于反向 DNS 查询（当 `type` 设置为 *PTR* 时），您可以同时以反转和非反转格式提供 DNS 名称（请参见下面的示例）。请注意，当请求 PTR 记录时，DNS 名称实际上是一个 IP 地址。
- 不支持国际化域名，请改用经过 IDNA 编码的名称。

示例：

```
net.dns[198.51.100.1,example.com,MX,2,1]

net.dns[,198.51.100.1,PTR]
net.dns[,1.100.51.198.in-addr.arpa,PTR]

net.dns[,2a00:1450:400f:800::200e,PTR]
net.dns[,e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.f.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa,PTR]
```

net.dns.perf[<ip>,name,<type>,<timeout>,<count>,<protocol>]

 检查 DNS 服务的性能。
 返回值：*Float*（0 - 服务已停止；秒 - 等待服务响应所花费的秒数）。
 参见支持的平台。

参数：

- **ip**（在 Windows 上忽略，除非使用 Zabbix agent 2）- DNS 服务器的 IP 地址（留空则使用默认 DNS 服务器）；
- **name** - 要查询的 DNS 名称；
- **type** - 要查询的记录类型（默认是 SOA）；
- **timeout**（在 Windows 上忽略，除非使用 Zabbix agent 2）- 请求超时时间，单位为秒（默认是 1 秒）；
- **count**（在 Windows 上忽略，除非使用 Zabbix agent 2）- 请求的重试次数（默认是 2）；
- **protocol** - 用于执行 DNS 查询的协议：*udp*（默认）或 *tcp*。

注释：

- type 的可选值为：
ANY、*A*、*NS*、*CNAME*、*MB*、*MG*、*MR*、*PTR*、*MD*、*MF*、*MX*、*SOA*、*NULL*、*WKS*（Zabbix agent on Windows、所有操作系统上的 Zabbix agent 2 不支持）、*HINFO*、*MINFO*、*TXT*、*SRV*。
- 对于反向 DNS 查询（当 type 设置为 *PTR* 时），你可以同时以反转和非反转格式提供 DNS 名称（请参见下面的示例）。请注意，当请求 PTR 记录时，DNS 名称实际上是一个 IP 地址。
- 不支持国际化域名，请改用经过 IDNA 编码的名称。
- 当 DNS 服务器返回错误代码时，监控项返回响应时间而不是 0（例如，NXDOMAIN 或 SERVFAIL）。

示例：

```
net.dns.perf[198.51.100.1,example.com,MX,2,1]

net.dns.perf[,198.51.100.1,PTR]
net.dns.perf[,1.100.51.198.in-addr.arpa,PTR]

net.dns.perf[,2a00:1450:400f:800::200e,PTR]
net.dns.perf[,e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.f.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa,PTR]
```

net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]

 执行 DNS 查询。

Zabbix agent 2 还提供net.dns.get，它提供了更多功能，例如支持更多记录类型以及对 DNS 监控更精细的控制。返回值：包含所需类型信息的字符串。
 请参见支持的平台。

参数：

- **ip**（在 Windows 上会被忽略，除非使用 Zabbix agent 2）- DNS 服务器的 IP 地址（留空则使用默认 DNS 服务器）；
- **name** - 要查询的 DNS 名称；
- **type** - 要查询的记录类型（默认值为 SOA）；
- **timeout**（在 Windows 上会被忽略，除非使用 Zabbix agent 2）- 请求超时时间，单位为秒（默认值为 1 秒）；
- **count**（在 Windows 上会被忽略，除非使用 Zabbix agent 2）- 请求重试次数（默认值为 2）；
- **protocol** - 执行 DNS 查询所使用的协议：*udp*（默认）或 *tcp*。

注释：

- type 的可选值为：
ANY、*A*、*NS*、*CNAME*、*MB*、*MG*、*MR*、*PTR*、*MD*、*MF*、*MX*、*SOA*、*NULL*、*WKS*（Zabbix agent on Windows 不支持，所有操作系统上的 Zabbix agent 2 也不支持）、*HINFO*、*MINFO*、*TXT*、*SRV*。
- 对于反向 DNS 查询（当 type 设置为 *PTR* 时），可以使用反向或非反向格式提供 DNS 名称（请参见下面的示例）。请注意，当请求 PTR 记录时，DNS 名称实际上是一个 IP 地址。
- 不支持国际化域名，请改用 IDNA 编码后的名称。

示例：

```
net.dns.record[198.51.100.1,example.com,MX,2,1]

net.dns.record[,198.51.100.1,PTR]
net.dns.record[,1.100.51.198.in-addr.arpa,PTR]

net.dns.record[,2a00:1450:400f:800::200e,PTR]
net.dns.record[,e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.f.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa,PTR]
```

net.if.collisions[if]

 窗口外冲突的数量。
 返回值：整数。
 支持的平台：Linux、FreeBSD、Solaris、AIX、MacOS X、OpenBSD、NetBSD。在 NetBSD 上需要 root 权限。

参数：

- **if** - 网络接口名称

net.if.discovery

 网络接口列表。用于低级别发现。
 返回值：*JSON string*。
 支持的平台：Linux、FreeBSD、Solaris、HP-UX、AIX、OpenBSD、NetBSD、Windows。

net.if.get[<regexp>]

 包含详细信息的网络接口列表。
 用于低级别发现。
 此监控项是net.if.discovery Zabbix agent 监控项的扩展版本。
 返回值：*JSON object*。
 支持的平台：Linux。

参数：

- **regexp** - 用于匹配网络接口名称的正则表达式。

注释：

- 返回一个 JSON object，其中包含名称与给定正则表达式匹配的网络接口（更多详情请参见more details）。

net.if.in[if,<mode>]

 网络接口上的入站流量统计。
 返回值：*Integer*。
 支持的平台：Linux、FreeBSD、Solaris[5]、HP-UX、AIX、MacOS X、OpenBSD、NetBSD、Windows。在 NetBSD 上需要 root 权限。

参数：

- **if** - 网络接口名称（Unix）；网络接口完整描述或 IPv4 地址；或者，如果用大括号括起，则为网络接口 GUID（Windows）；
- **mode** - 可能的值：
  - *bytes* - 字节数（默认）；
  - *packets* - 数据包数；
  - *errors* - 错误数；
  - *dropped* - 丢弃的数据包数；
  - *overruns (fifo)* - FIFO 缓冲区错误数；
  - *frame* - 数据包成帧错误数；
  - *compressed* - 设备驱动程序接收到的压缩数据包数；
  - *multicast* - 设备驱动程序接收到的多播帧数。

注释：

- 可将此键与 *Change per second* 预处理步骤一起使用，以获取每秒字节数统计。
- *dropped* 模式仅支持 Linux、FreeBSD、HP-UX、MacOS X、OpenBSD、NetBSD。
- *overruns*、*frame*、*compressed*、*multicast* 模式仅支持 Linux。
- 在 HP-UX 上，此监控项不提供回环接口（例如 lo0）的详细信息。

示例：

```
net.if.in[eth0]
net.if.in[eth0,errors]
```

net.if.out[if,<mode>]

 网络接口上的出站流量统计。
 返回值：*Integer*。
 支持的平台：Linux、FreeBSD、Solaris[5]、HP-UX、AIX、MacOS X、OpenBSD、NetBSD、Windows。在 NetBSD 上需要 root 权限。

参数：

- **if** - 网络接口名称（Unix）；网络接口完整描述或 IPv4 地址；或者，如果用大括号括起，则为网络接口 GUID（Windows）；
- **mode** - 可选值：
  - *bytes* - 字节数（默认）；
  - *packets* - 数据包数；
  - *errors* - 错误数；
  - *dropped* - 丢弃的数据包数；
  - *overruns (fifo)* - FIFO 缓冲区错误数；
  - *collisions (colls)* - 检测到的接口冲突数；
  - *carrier* - 设备驱动程序检测到的载波丢失数；
  - *compressed* - 设备驱动程序发送的压缩数据包数。

注释：

- 可将此键与 *Change per second* 预处理步骤一起使用，以获取每秒字节数统计。
- *dropped* 模式仅支持 Linux、HP-UX。
- *overruns*、*collision*、*carrier*、*compressed* 模式仅支持 Linux。
- 在 HP-UX 上，此监控项不提供回环接口（例如 lo0）的详细信息。

示例：

```
net.if.out[eth0]
net.if.out[eth0,errors]
```

net.if.total[if,<mode>]

 网络接口上入站和出站流量统计的总和。
 返回值：整数。
 支持的平台：Linux、FreeBSD、Solaris[5]、HP-UX、AIX、MacOS X、OpenBSD、NetBSD、Windows。在 NetBSD 上需要 root 权限。

参数：

- **if** - 网络接口名称（Unix）；网络接口完整描述或 IPv4 地址；或者，如果用大括号括起，则为网络接口 GUID（Windows）；

- **mode** - 可能的值：
  - *bytes* - 字节数（默认）；
  - *packets* - 数据包数；
  - *errors* - 错误数；
  - *dropped* - 丢弃的数据包数；
  - *overruns (fifo)* - FIFO 缓冲区错误数；
  - *collisions (colls)* - 检测到的接口冲突数；
  - *compressed* - 设备驱动程序发送或接收的压缩数据包数。

注释：

- 你可以将此键与 *Change per second* 预处理步骤一起使用，以获取每秒字节数统计。
- *dropped* 模式仅在 Linux、HP-UX 上受支持。只有当你平台上的 `net.if.in` 和 `net.if.out` 都支持 dropped 数据包时，才支持 dropped 数据包。
- *overruns*、*collision*、*compressed* 模式仅在 Linux 上受支持。
- 在 HP-UX 上，此监控项不提供回环接口（例如 lo0）的详细信息。

示例：

```
net.if.total[eth0]
net.if.total[eth0,errors]
```

net.tcp.listen[port]

 检查此 TCP 端口是否处于 LISTEN 状态。
 返回值：0 - 不处于 LISTEN 状态；1 - 处于 LISTEN 状态。
 支持的平台：Linux、FreeBSD、Solaris、MacOS X、Windows。

参数：

- **port** - TCP 端口号

在 Linux 内核 2.6.14 及以上版本中，如果可能，监听中的 TCP 套接字信息将从内核的 NETLINK 接口获取。否则，将从 `/proc/net/tcp` 和 `/roc/net/tcp6` 文件中检索信息。

示例：

```
net.tcp.listen[80]
```

net.tcp.port[<ip>,port]

 检查是否可以与指定端口建立 TCP 连接。
 返回值：0 - 无法连接；1 - 可以连接。
 请参见支持的平台。

参数：

- **ip** - IP 地址或 DNS 名称（默认值为 `127.0.0.1`）；
- **port** - 端口号。

注释：

- 如需进行简单的 TCP 性能测试，请使用 `net.tcp.service.perf[tcp,<ip>,<port>]`。
- 这些检查可能会在系统守护进程日志文件中产生额外消息（通常会记录 SMTP 和 SSH 会话）。

示例：

```
net.tcp.port[,80] # 此监控项可用于测试运行在 80 端口上的 web 服务器可用性
```

net.tcp.service[service,<ip>,<port>]

 检查某项服务是否正在运行并接受 TCP 连接。
 返回值：0 - 服务已停止；1 - 服务正在运行。
 请参见支持的平台。

参数：

- **service** - *ssh*、*ldap*、*smtp*、*ftp*、*http*、*pop*、*nntp*、*imap*、*tcp*、*https* 或 *telnet*（请参见详细信息）；
- **ip** - IP 地址或 DNS 名称（默认值为 `127.0.0.1`）；
- **port** - 端口号（默认使用该服务的标准端口号）。

注释：

- 这些检查可能会在系统守护进程日志文件中产生额外消息（通常会记录 SMTP 和 SSH 会话）。
- 目前不支持对加密协议进行检查（例如端口 993 上的 IMAP 或端口 995 上的 POP）。作为变通方法，请对这类检查使用 `net.tcp.port[]`。
- 在 Windows 上，对 LDAP 和 HTTPS 的检查仅由 Zabbix agent 2 支持。
- telnet 检查会查找登录提示符（末尾带有 `:`）。

示例：

```
net.tcp.service[ftp,,45] # 此监控项可用于测试 TCP 端口 45 上 FTP 服务器的可用性
```

net.tcp.service.perf[service,<ip>,<port>]

 检查 TCP 服务的性能。
 返回值：*Float*（0 - 服务已停止；秒 - 等待服务响应所花费的秒数）。
 请参见受支持的平台。

参数：

- **service** - *ssh*、*ldap*、*smtp*、*ftp*、*http*、*pop*、*nntp*、*imap*、*tcp*、*https* 或 *telnet*（请参见详细信息）；
- **ip** - IP 地址或 DNS 名称（默认值为 127.0.0.1）；
- **port** - 端口号（默认使用该服务的标准端口号）。

注释：

- 目前不支持检查加密协议（例如端口 993 上的 IMAP 或端口 995 上的 POP）。作为替代方案，请对这类检查使用 net.tcp.service.perf[tcp,<ip>,<port>]。
- telnet 检查会查找登录提示符（末尾带有 :）。

示例：

```
net.tcp.service.perf[ssh] # 此监控项可用于测试来自 SSH 服务器的初始响应速度
```

net.tcp.socket.count[<laddr>,<lport>,<raddr>,<rport>,<state>]

 返回与参数匹配的 TCP 套接字数量。
 返回值：整数。
 支持的平台：Linux、Windows。

参数：

- **laddr** - 本地 IPv4/6 地址或 CIDR 子网；
- **lport** - 本地端口号或服务名称；
- **raddr** - 远程 IPv4/6 地址或 CIDR 子网；
- **rport** - 远程端口号或服务名称；
- **state** - 连接状态（*established*、*syn_sent*、*syn_recv*、*fin_wait1*、*fin_wait2*、*time_wait*、*close*、*close_wait*、*last_ack*、*listen*、*closing*）。

示例：

```
net.tcp.socket.count[,80,,,established] # 本地 TCP 端口 80 上处于 established 状态的连接数
```

net.udp.listen[port]

 检查此 UDP 端口是否处于 LISTEN 状态。
 返回值：0 - 不处于 LISTEN 状态；1 - 处于 LISTEN 状态。
 支持的平台：Linux、FreeBSD、Solaris、MacOS X。

参数：

- **port** - UDP 端口号

示例：

```
net.udp.listen[68]
```

net.udp.service[service,<ip>,<port>]

 检查某个服务是否正在运行并响应 UDP 请求。
 返回值：0 - 服务已停止；1 - 服务正在运行。
 请参见支持的平台。

参数：

- **service** - *ntp*（请参见详细信息）；
- **ip** - IP 地址或 DNS 名称（默认值为 127.0.0.1）；
- **port** - 端口号（默认使用该服务的标准端口号）。

示例：

```
net.udp.service[ntp,,45] #this item can be used to test the availability of NTP service on UDP port 45
```

net.udp.service.perf[service,<ip>,<port>]

 检查 UDP 服务的性能。
 返回值：*Float*（0 - 服务已停止；秒 - 等待服务响应所花费的秒数）。
 请参见支持的平台。

参数：

- **service** - *ntp*（参见详细信息）；
- **ip** - IP 地址或 DNS 名称（默认值为 127.0.0.1）；
- **port** - 端口号（默认使用标准服务端口号）。

示例：

```
net.udp.service.perf[ntp] #this item can be used to test response time from NTP service
```

net.udp.socket.count[<laddr>,<lport>,<raddr>,<rport>,<state>]

 返回与参数匹配的 UDP 套接字数量。
 返回值：整数。
 支持的平台：Linux、Windows。

参数：

- **laddr** - 本地 IPv4/6 地址或 CIDR 子网；
- **lport** - 本地端口号或服务名称；
- **raddr** - 远程 IPv4/6 地址或 CIDR 子网；
- **rport** - 远程端口号或服务名称；
- **state** - 连接状态（*established*、*unconn*）。

示例：

```
net.udp.socket.count[,,,,established] # 返回处于已连接状态的 UDP 套接字数量
```

proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]

 进程 CPU 利用率百分比。
 返回值：*Float*。
 支持的平台：Linux、Solaris[6]。

参数：

- **name** - 进程名称（默认值为 *all processes*）；
- **user** - 用户名（默认值为 *all users*）；
- **type** - CPU 利用率类型：*total*（默认）、*user* 或 *system*；
- **cmdline** - 按命令行过滤（这是一个正则表达式）；
- **mode** - 数据采集模式：*avg1*（默认）、*avg5* 或 *avg15*；
- **zone** - 目标区域：*current*（默认）或 *all*。此参数仅在 Solaris 上受支持。

注释：

- 返回值基于单个 CPU 核心的利用率百分比。例如，某个进程如果完全占用两个核心，其 CPU 利用率为 200%。
- 进程 CPU 利用率数据由一个采集器收集，该采集器最多支持 1024 个唯一查询（按名称、用户和命令行区分）。在过去 24 小时内未被访问的查询会从采集器中移除。
- 当将 zone 参数设置为 *current*（或使用默认值）时，如果 agent 是在不支持 zone 的 Solaris 上编译的，但运行在一个支持 zone 的较新 Solaris 版本上，那么 agent 将返回 NOTSUPPORTED（agent 无法将结果限制为仅当前 zone）。不过，在这种情况下，*all* 是受支持的。

示例：

```
proc.cpu.util[,root] # 所有以 "root" 用户运行的进程的 CPU 利用率
proc.cpu.util[zabbix_server,zabbix] # 以 zabbix 用户运行的所有 zabbix_server 进程的 CPU 利用率
```

proc.get[<name>,<user>,<cmdline>,<mode>]

 操作系统进程及其参数列表。可用于低级别发现。
 返回值：*JSON* 字符串。
 支持的平台：Linux、FreeBSD、Windows、OpenBSD、NetBSD。

参数：

- **name** - 进程名称（默认 *all processes*）。
- **user** - 用户名（默认 *all users*）。
- **cmdline** - 按命令行过滤（这是一个正则表达式）。此参数不支持 Windows；在其他平台上，如果 mode 设置为'summary'，则不支持。
- **mode** - 可能的值：
  - *process*（默认）；
  - *thread*（不支持 NetBSD）；
  - *summary*。请参见每种模式和操作系统返回的进程参数列表。

注释：

- 如果无法获取某个值，例如由于错误（进程已退出、权限不足、系统调用失败），将返回 -1。
- 有关使用 name 和 cmdline 参数选择进程的说明，请参见注释（Linux 特定）。

示例：

```
proc.get[zabbix_server,zabbix,,process] # 列出以 zabbix 用户运行的所有 zabbix_server 进程，每个 PID 返回一条记录
proc.get[java,,,thread] # 列出所有 Java 进程，每个线程返回一条记录
proc.get[,zabbix,,summary] # 汇总以 zabbix 用户运行的各类进程数据，每个进程名称返回一条记录
```

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

 进程使用的内存，单位为字节。
 返回值：- *Integer* - 当 mode 为 *max*、*min*、*sum* 时；- *Float* - 当 mode 为 *avg* 时
 Supported platforms：Linux、FreeBSD、Solaris、AIX、OpenBSD、NetBSD。

参数：

- **name** - 进程名称（默认是 *all processes*）；
- **user** - 用户名（默认是 *all users*）；
- **mode** - 可选值：*avg*、*max*、*min* 或 *sum*（默认）；
- **cmdline** - 按命令行过滤（这是一个正则expression）；
- **memtype** - 进程使用的内存类型

注释：

- memtype 参数仅在 Linux、FreeBSD、Solaris[6]、AIX 上受支持。
- 当多个进程使用共享内存时，进程使用内存的总和可能会得到较大且不切实际的值。

 有关使用 name 和 cmdline 参数选择进程的说明（仅适用于 Linux），请参见注释。

 当此监控项从命令行调用并包含命令行参数时（例如使用 agent 测试模式：zabbix_agentd -t proc.mem[,,,apache2]），会额外统计一个进程，因为 agent 会统计自身。

示例：

```
proc.mem[,root] # 所有以 "root" 用户运行的进程所使用的内存
proc.mem[zabbix_server,zabbix] # 所有以 zabbix 用户运行的 zabbix_server 进程所使用的内存
proc.mem[,oracle,max,oracleZABBIX] # 在 Oracle 下运行且命令行中包含 oracleZABBIX 的、最占内存的进程所使用的内存
```

proc.num[<name>,<user>,<state>,<cmdline>,<zone>]

 进程数量。
 返回值：*Integer*。
 支持的平台：Linux、FreeBSD、Solaris[6]、HP-UX、AIX、OpenBSD、NetBSD、Windows。

参数：

- **name** - 进程名称（默认值为 *all processes*）。
- **user** - 用户名（默认值为 *all users*）。
- **state** - 可能的值：
  - *all*（默认）；
  - *disk* - 不可中断睡眠；
  - *run* - 运行中；
  - *sleep* - 可中断睡眠；
  - *trace* - 已停止；
  - *zomb* - 僵尸进程。
- **cmdline** - 按命令行过滤（这是一个正则表达式）。
- **zone** - 目标区域：*current*（默认）或 *all*。此参数仅在 Solaris 上受支持。

注释：

- *disk* 和 *trace* 状态参数仅在 Linux、FreeBSD、OpenBSD、NetBSD 上受支持。
- 在 Windows 上，仅支持 name 和 user 参数。
- 当此监控项从命令行调用并包含命令行参数时（例如使用 agent 测试模式：zabbix_agentd -t proc.num[,,,apache2]），会额外统计一个进程，因为 agent 会将自身也计入。
- 当将 zone 参数设置为 current（或默认值）时，如果 agent 是在不支持 zone 的 Solaris 上编译的，但运行在较新且支持 zone 的 Solaris 上，则 agent 将返回 NOTSUPPORTED（agent 无法将结果限制为仅当前 zone）。不过，在这种情况下，*all* 是受支持的。
- 有关使用 name 和 cmdline 参数选择进程的说明（仅适用于 Linux），请参见注释。

示例：

```
proc.num[,mysql] #mysql 用户下运行的进程数量
proc.num[apache2,www-data] #www-data 用户下运行的 apache2 进程数量
proc.num[,oracle,sleep,oracleZABBIX] # 处于 sleep 状态、由 Oracle 运行且命令行中包含 oracleZABBIX 的进程数量
```

sensor[device,sensor,<mode>]

 硬件传感器读数。
 返回值：*Float*。
 支持的平台：Linux、OpenBSD。

参数：

- **device** - 设备名称；
- **sensor** - 传感器名称；
- **mode** - 可能的值：*avg*、*max* 或 *min*。

注释：

- 在 Linux 2.4 上读取 /proc/sys/dev/sensors。

- 在 Linux 2.6+ 上读取 /sys/class/hwmon。有关 Linux 上sensor 监控项的更详细说明，请参见。
- 在 OpenBSD 上读取 hw.sensors MIB。
- 在 Linux 2.4 上，如果指定了 mode，则 device 和 sensor 会被视为正则表达式。
- 在 Linux 2.6+ 和 OpenBSD 上，或者如果省略 mode，则 device 和 sensor 会被按原样处理。

示例：

```
sensor[w83781d-i2c-0-2d,temp1]
sensor[cpu0,temp0] # 一颗 CPU 的温度
sensor["cpu[0-2]$",temp,avg] # 前三颗 CPU 的平均温度
```

system.boottime

系统启动时间。
 返回值：整数（*Unix* 时间戳）。
 支持的平台：Linux、FreeBSD、Solaris、MacOS X、OpenBSD、NetBSD。

system.cpu.discovery

 检测到的 CPU/CPU 核心列表。
用于低级别发现。
 返回值：*JSON string*。
 请参见支持的平台。

system.cpu.intr

设备中断数。
 返回值：整数。
 支持的平台：Linux、FreeBSD、Solaris、AIX、OpenBSD、NetBSD。

system.cpu.load[<cpu>,<mode>]

 CPU 负载。
 返回值：*Float*。
 请参见支持的平台。

参数：

- **cpu** - 可能的值：*all*（默认）或 *percpu*（总负载除以在线 CPU 数量）；
- **mode** - 可能的值：*avg1*（1 分钟平均值，默认）、*avg5* 或 *avg15*。

示例：

```
system.cpu.load[,avg5]
```

system.cpu.num[<type>]

 CPU 的数量。
 返回值：整数。
 支持的平台：Linux、FreeBSD、Solaris、HP-UX、AIX、MacOS X、OpenBSD、NetBSD、Windows。

参数：

- **type** - 可能的值：*online*（默认）或 *max*

max 类型参数仅在 Linux、FreeBSD、Solaris、MacOS X 上受支持。

示例：

```
system.cpu.num
```

system.cpu.switches

上下文切换次数。
 返回值：整数。
 支持的平台：Linux、FreeBSD、Solaris、AIX、OpenBSD、NetBSD。

system.cpu.util[<cpu>,<type>,<mode>,<logical or physical>]

 CPU 利用率百分比。
 返回值：*Float*。
 支持的平台：Linux、FreeBSD、Solaris、HP-UX、AIX、OpenBSD、NetBSD、Windows。

参数：

- **cpu** - *<CPU 编号 >* 或 *all*（默认）；
- **type** - 可选值：*user*（默认）、*idle*、*nice*、*system*、*iowait*、*interrupt*、*softirq*、*steal*、*spin*（在 OpenBSD 上）、*guest*（在 Linux 内核 2.6.24 及以上版本上），或 *guest_nice*（在 Linux 内核 2.6.33 及以上版本上）；
- **mode** - 可选值：*avg1*（1 分钟平均值，默认）、*avg5* 或 *avg15*；
- **logical or physical** - 可选值：*logical*（默认）或 *physical*。此参数仅在 AIX 上受支持。

说明：

- *nice* 类型参数仅在 Linux、FreeBSD、HP-UX、OpenBSD、NetBSD 上受支持。
- *iowait* 类型参数仅在 Linux 2.6 及更高版本、Solaris、AIX 上受支持。
- *interrupt* 类型参数仅在 Linux 2.6 及更高版本、FreeBSD、OpenBSD 上受支持。
- *softirq*、*steal*、*guest*、*guest_nice* 类型参数仅在 Linux 2.6 及更高版本上受支持。
- *avg5* 和 *avg15* 模式参数在 Linux、FreeBSD、Solaris、HP-UX、AIX、OpenBSD、NetBSD 上受支持。

示例：

```
system.cpu.util[0,user,avg5]
```

system.hostname[<type>,<transform>]

 系统主机名。
 返回值：*String*。
 请参见支持的平台。

参数：

- **type** - 可能的值：
  - *netbios*（Windows 上的默认值）；
  - *host*（Linux 上的默认值）；
  - *shorthost*（返回主机名中第一个点之前的部分；如果名称中不包含点，则返回完整字符串）；
  - *fqdn*（返回完全限定域名）。
- **transform** - 可能的值：
  - *none*（默认值）；
  - *lower*（转换为小写）。

该值通过从 uname() 系统 API 输出中获取 nodename 得到。

返回值示例：

```
system.hostname → linux-w7x1
system.hostname → example.com
system.hostname[shorthost] → example
system.hostname → WIN-SERV2008-I6
system.hostname[host] → Win-Serv2008-I6LonG
system.hostname[host,lower] → win-serv2008-i6long
system.hostname[fqdn,lower] → blog.zabbix.com
```

system.hw.chassis[<info>]

 机箱信息。
 返回值：*String*。
 支持的平台：Linux。

参数：

- **info** - 可能的值：*full*（默认）、*model*、*serial*、*type* 或 *vendor*

注释：

- 此监控项键依赖于 SMBIOS 表的可用性。
- 它会尝试从 sysfs 读取 DMI 表；如果访问 sysfs 失败，则会尝试直接从内存中读取。
- 由于该值是通过读取 sysfs 或内存获取的，因此需要 **root** 权限。

示例：

```
system.hw.chassis[full] → Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop
```

system.hw.cpu[<cpu>,<info>]

 CPU 信息。
 返回值：*String* 或 *Integer*。
 支持的平台：Linux。

参数：

- **cpu** - <*CPU* 编号 > 或 *all*（默认）；
- **info** - 可选值：*full*（默认）、*curfreq*、*maxfreq*、*model* 或 *vendor*。

注释：

- 从 /proc/cpuinfo 和 /sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_freq 收集信息。
- 如果指定了 CPU 编号以及 *curfreq* 或 *maxfreq*，则返回数值（Hz）。

示例：

```
system.hw.cpu[0,vendor] → AuthenticAMD
```

system.hw.devices[<type>]

 PCI 或 USB 设备列表。
 返回值：*Text*。
 支持的平台：Linux。

参数：

- **type** - *pci*（默认）或 *usb*

返回 lspci 或 lsusb 工具的输出（执行时不带任何参数）。

示例：

```
system.hw.devices → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge
```

system.hw.macaddr[<interface>,<format>]

 MAC 地址列表。
 返回值：*String*。
 支持的平台：Linux。

参数：

- **interface** - *all*（默认）或正则expression；
- **format** - *full*（默认）或 *short*。

注释：

- 列出名称与给定 `interface` 正则expression 匹配的接口的 MAC 地址（*all* 表示列出所有接口）。
- 如果 `format` 指定为 *short*，则不会列出接口名称和相同的 MAC 地址。

示例：

```
system.hw.macaddr["eth0$",full] → [eth0] 00:11:22:33:44:55
```

system.localtime[<type>]

 系统时间。
 返回值：*Integer* - 当 `type` 为 *utc* 时；*String* - 当 `type` 为 *local* 时。
 请参见受支持的平台。

参数：

- **type** - 可能的值：
  - *utc* -（默认）自 Unix 纪元（1970 年 1 月 1 日 00:00:00 UTC）以来的时间，以秒为单位；
  - *local* - 采用'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm' 格式的时间。

必须作为 Zabbix agent 的被动检查使用；在 Zabbix agent 2 上，也可以配置为主动检查。

示例：

```
system.localtime[local] # 使用此键创建一个监控项，然后用它在 *Clock* 仪表板小部件中显示主机时间。
```

system.run[command,<mode>]

 在主机上运行指定命令。
 返回值：命令的 *Text* 结果；如果 `mode` 为 *nowait*，则返回 1（无论命令结果如何）。
 请参阅支持的平台。

参数：

- **command** - 要执行的命令；
- **mode** - 可能的值：
  - *wait* - 等待执行结束（默认）；
  - *nowait* - 不等待。

注释：

- 此监控项默认处于禁用状态。了解如何启用它们。
- 该监控项的返回值是命令产生的标准输出和标准错误。不执行退出码检查。
- 为了正确处理，命令的返回值必须是 `text` 数据类型。也允许返回空结果。
- 返回值限制为 16MB（包括会被截断的尾随空白）；同时也适用数据库限制。
- 另请参阅：命令执行。

示例：

```
system.run[ls -l /] # 返回根目录的详细文件列表
```

system.stat[resource,<type>]

 系统统计信息。
 返回值：*Integer* 或 *float*。
 支持的平台：AIX。

参数：

- **ent** - 此分区有权接收的处理器单元数量（float）。
- **kthr,<type>** - 内核线程状态信息：
  - *r* - 可运行内核线程的平均数量（float）；
  - *b* - 放入虚拟内存管理器等待队列的内核线程平均数量（float）。
- **memory,<type>** - 虚拟内存和物理内存使用情况信息：
  - *avm* - 活跃虚拟页（integer）；

    - *fre* - 空闲列表大小（integer）。
- **page,<type>** - 页面错误和分页活动信息：
    - *fi* - 每秒文件页调入数（float）；
    - *fo* - 每秒文件页调出数（float）；
    - *pi* - 从分页空间调入的页数（float）；
    - *po* - 调出到分页空间的页数（float）；
    - *fr* - 释放的页数（页面替换）（float）；
    - *sr* - 页面替换算法扫描的页数（float）。
- **faults,<type>** - 陷阱和中断速率：
    - *in* - 设备中断（float）；
    - *sy* - 系统调用（float）；
    - *cs* - 内核线程上下文切换（float）。
- **cpu,<type>** - 处理器时间使用百分比明细：
    - *us* - 用户时间（float）；
    - *sy* - 系统时间（float）；
    - *id* - 空闲时间（float）；
    - *wa* - 系统存在未完成磁盘/NFS I/O 请求时的空闲时间（float）；
    - *pc* - 消耗的物理处理器数量（float）；
    - *ec* - 消耗的配额容量百分比（float）；
    - *lbusy* - 表示在用户级和系统级执行时发生的逻辑处理器利用率百分比（float）；
    - *app* - 表示共享池中可用的物理处理器（float）。
- **disk,<type>** - 磁盘统计信息：
    - *bps* - 表示传输到驱动器或从驱动器传输的数据量，单位为每秒字节数（integer）；
    - *tps* - 表示发送到物理磁盘/磁带的每秒传输次数（float）。

注释：

- 请注意这些监控项的以下限制：
    - `system.stat[cpu,app]` - 仅支持 AIX 类型为”Shared” 的 LPAR；
    - `system.stat[cpu,ec]` - 支持 AIX 类型为”Shared” 和”Dedicated” 的 LPAR（”Dedicated” 始终返回 100（percent））；
    - `system.stat[cpu,lbusy]` - 仅支持 AIX 类型为”Shared” 的 LPAR；
    - `system.stat[cpu,pc]` - 支持 AIX 类型为”Shared” 和”Dedicated” 的 LPAR；
    - `system.stat[ent]` - 支持 AIX 类型为”Shared” 和”Dedicated” 的 LPAR。

system.sw.arch

 软件架构信息。
 返回值：*String*。
 请参见支持的平台。

该信息通过 `uname()` 函数获取。

示例：

```
system.sw.arch → i686
```

system.sw.os[<info>]

 操作系统信息。
 返回值：*String*。
 支持的平台：Linux、Windows。

参数：

- **info** - 可能的值：*full*（默认）、*short* 或 *name*

info 的获取来源如下（请注意，并非所有发行版都包含所有文件和选项）：

- Linux 上的 `/proc/version`（*full*）；
- Linux 上的 `/proc/version_signature`（*short*）；
- 支持该参数的 Linux 系统中的 `/etc/os-release` 里的 PRETTY_NAME 参数，或 `/etc/issue.net`（*name*）；
- Windows 上的注册表项 `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion`。

示例：

```
system.sw.os[short] → Ubuntu 2.6.35-28.50-generic 2.6.35.11
system.sw.os[full] → [s|Windows 10 Enterprise 22621.1.asd64fre.ni_release.220506-1250 Build 22621.963]
```

system.sw.os.get

 有关操作系统的详细信息（版本、类型、发行版名称、次版本和主版本等）。
 返回值：*JSON string*。
 支持的平台：Linux、Windows。

system.sw.packages[<regexp>,<manager>,<format>]

 已安装软件包的列表。
 返回值：*Text*。
 支持的平台：Linux。

参数：

- **regexp** - *all*（默认）或正则表达式；
- **manager** - *all*（默认）或软件包管理器；
- **format** - *full*（默认）或 *short*。

注释：

- 列出名称与给定正则表达式匹配的已安装软件包（按字母顺序）（*all* 会列出全部）。
- 支持的软件包管理器（执行的命令）：
  - pkg (`dpkg --get-selections`)；
  - pkgtool (`ls /var/log/packages`)；
  - rpm (`rpm -qa`)；
  - pacman (`pacman -Q`)；
  - portage。
- 如果 `format` 指定为 *full*，软件包会按软件包管理器分组（每个管理器单独占一行，并以方括号中的名称开头）。
- 如果 `format` 指定为 *short*，软件包不会分组，而是显示在同一行上。

示例：

```
system.sw.packages[mini,dpkg,short] → python-minimal, python2.6-minimal, ubuntu-minimal
```

system.sw.packages.get[<regexp>,<manager>]

 已安装软件包的详细列表。
 返回值：*JSON string*。
 Supported platforms：Linux。

参数：

- **regexp** - *all*（默认）或正则表达式；
- **manager** - *all*（默认）或软件包管理器（可选值：*rpm*、*dpkg*、*pkgtool*、*pacman* 或 *portage*）。

注释：

- 返回未格式化的 JSON，其中包含名称与给定正则表达式匹配的已安装软件包。
- 输出为对象数组，每个对象包含以下键：name、manager、version、size、architecture、buildtime 和 installtime（请参见更多详情）。

system.swap.in[<device>,<type>]

 swap-in（从设备到内存）的统计信息。
 返回值：*Integer*。
 支持的平台：Linux、FreeBSD、OpenBSD。

参数：

- **device** - 指定用于交换的设备（仅限 Linux）或 *all*（默认）；
- **type** - 可选值：
  - *count*（swapin 次数，非 Linux 平台默认值）；
  - *sectors*（交换入的扇区数）；
  - *pages*（交换入的页数，Linux 上的默认值）。

注释：

- 此信息的来源为：
  - `/proc/swaps`、`/proc/partitions`、`/proc/stat`（Linux 2.4）；
  - `/proc/swaps`、`/proc/diskstats`、`/proc/vmstat`（Linux 2.6）。
- 请注意，只有在未指定 device 时，*pages* 才会生效。
- *sectors* 类型参数仅在 Linux 上受支持。

示例：

```
system.swap.in[,pages]
```

system.swap.out[<device>,<type>]

 交换输出（从内存到设备）的统计信息。
 返回值：*Integer*。
 支持的平台：Linux、FreeBSD、OpenBSD。

参数：

- **device** - 指定用于交换的设备（仅 Linux）或 *all*（默认）；
- **type** - 可选值：
  - *count*（交换输出次数，非 Linux 平台默认值）；
  - *sectors*（交换输出的扇区数）；
  - *pages*（交换输出的页数，Linux 上的默认值）。

注释：

- 此信息的来源为：
  - /proc/swaps、/proc/partitions、/proc/stat（Linux 2.4）；
  - /proc/swaps、/proc/diskstats、/proc/vmstat（Linux 2.6）。
- 请注意，只有在未指定 device 时，*pages* 才会生效。
- *sectors* 类型参数仅在 Linux 上受支持。

示例：

```
system.swap.out[,pages]
```

system.swap.size[<device>,<type>]

 交换空间大小，单位为字节，或占总量的百分比。
 返回值：*Integer* - 表示字节；*Float* - 表示百分比。
 Supported platforms：Linux、FreeBSD、Solaris、AIX、OpenBSD、Windows。

参数：

- **device** - 指定用于交换的设备（仅 FreeBSD）或 *all*（默认）；
- **type** - 可能的值：
  - *free*（可用交换空间，默认）；
  - *pfree*（可用交换空间，百分比）；
  - *pused*（已用交换空间，百分比）；
  - *total*（交换空间总量）；
  - *used*（已用交换空间）。

注释：

- 请注意，如果 swap size 为 0，Windows 不支持 *pfree*、*pused*。
- 如果未指定 device，Zabbix agent 只会考虑交换设备（文件），物理内存将被忽略。例如，在 Solaris 系统上，`swap -s` 命令会包含一部分物理内存和交换设备（不同于 `swap -l`）。

示例：

```
system.swap.size[,pfree] → 可用交换空间百分比
```

system.uname

 系统的标识信息。
 返回值：字符串。
 请参见支持的平台。

注释：

- 在 UNIX 上，此监控项的值通过 uname() 系统调用获取。
- 在 Windows 上，此监控项返回操作系统架构，而在 UNIX 上则返回 CPU 架构。

示例：

```
system.uname → FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386
system.uname → Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86
```

system.uptime

 系统运行时间（以秒为单位）。
 返回值：*Integer*。
 支持的平台：Linux、FreeBSD、Solaris、AIX、MacOS X、OpenBSD、NetBSD、Windows。

在监控项配置中，使用 **s** 或 **uptime** 单位可获得易读的值。

system.users.num

已登录用户数。
 返回值：*Integer*。
 参见支持的平台。

agent 使用 **who** 命令来获取该值。

vfs.dev.discovery

 块设备及其类型的列表。
用于低级发现。
 返回值：*JSON string*。
 支持的平台：Linux。

vfs.dev.get[<mode>,<regexp>]

 带有详细信息的块设备列表。用于低级别发现。
 返回值：*JSON* 字符串。
 支持的平台：Linux、AIX。

参数：

- **mode** - 可能的值：
  - *disks*（默认）- 返回磁盘列表；
  - *disk_stats* - 返回每个磁盘的统计信息；

  - *devices*（仅限 Linux）- 返回块设备列表（磁盘和分区）；
  - *device_stats*（仅限 Linux）- 返回每个设备（磁盘和分区）的统计信息。
- **regexp** - 用于按名称过滤块设备的正则表达式。

vfs.dev.read[<device>,<type>,<mode>]

 磁盘读取统计信息。
 返回值：- *Integer* - 当 type 为 *sectors*、*operations*、*bytes* 时；- *Float* - 当 type 为 *sps*、*ops*、*bps* 时。
 支持的平台：Linux、FreeBSD、Solaris、AIX、OpenBSD。

参数：

- **device** - 磁盘设备（默认值为 *all* [3]）；
- **type** - 可选值：*sectors*、*operations*、*bytes*、*sps*、*ops* 或 *bps*（*sps*、*ops*、*bps* 分别表示：每秒扇区数、每秒操作数、每秒字节数）；
- **mode** - 可选值：*avg1*（1 分钟平均值，默认）、*avg5* 或 *avg15*。此参数仅在 type 为 sps、ops、bps 时受支持。

注释：

- 如果使用 3 小时或更长的更新间隔[2]，此监控项将始终返回 0。
- *sectors* 和 *sps* 类型参数仅在 Linux 上受支持。
- *ops* 类型参数仅在 Linux 和 FreeBSD 上受支持。
- *bps* 类型参数仅在 FreeBSD 上受支持。
- *bytes* 类型参数仅在 FreeBSD、Solaris、AIX、OpenBSD 上受支持。
- mode 参数仅在 Linux、FreeBSD 上受支持。
- 您可以使用相对设备名（例如 sda），也可以使用可选的 /dev/ 前缀（例如 /dev/sda）。
- 支持 LVM 逻辑卷。
- 不同操作系统中 type 参数的默认值：
  - *AIX* - operations；
  - *FreeBSD* - bps；
  - *Linux* - sps；
  - *OpenBSD* - operations；
  - *Solaris* - bytes。
- 支持的平台上，*sps*、*ops* 和 *bps* 的设备数量上限为 1024 个（1023 个单独设备和 1 个 *all*）。

示例：

```
vfs.dev.read[,operations]
```

vfs.dev.write[<device>,<type>,<mode>]

 磁盘写入统计信息。
 返回值：*Integer* - 当 type 为 *sectors*、*operations*、*bytes* 时；*Float* - 当 type 为 *sps*、*ops*、*bps* 时。
 支持的平台：Linux、FreeBSD、Solaris、AIX、OpenBSD。

参数：

- **device** - 磁盘设备（默认值为 *all* [3]）；
- **type** - 可能的值：*sectors*、*operations*、*bytes*、*sps*、*ops* 或 *bps*（*sps*、*ops*、*bps* 分别表示：每秒扇区数、每秒操作数、每秒字节数）；
- **mode** - 可能的值：*avg1*（1 分钟平均值，默认）、*avg5* 或 *avg15*。此参数仅在 type 为以下值时受支持：sps、ops、bps。

备注：

- 如果使用 3 小时或更长的更新间隔[2]，此监控项将始终返回 0。
- *sectors* 和 *sps* 类型参数仅在 Linux 上受支持。
- *ops* 类型参数仅在 Linux 和 FreeBSD 上受支持。
- *bps* 类型参数仅在 FreeBSD 上受支持。
- *bytes* 类型参数仅在 FreeBSD、Solaris、AIX、OpenBSD 上受支持。
- mode 参数仅在 Linux、FreeBSD 上受支持。
- 您可以使用相对设备名（例如 sda），也可以使用可选的 /dev/ 前缀（例如 /dev/sda）。
- 支持 LVM 逻辑卷。
- 不同操作系统中 type 参数的默认值：
  - *AIX* - operations
  - *FreeBSD* - bps
  - *Linux* - sps
  - *OpenBSD* - operations
  - *Solaris* - bytes
- 支持平台上的 *sps*、*ops* 和 *bps* 最多限制为 1024 个设备（1023 个单独设备和 1 个 *all*）。

示例：

```
vfs.dev.write[,operations]
```

vfs.dir.count[dir,<regex incl>,<regex excl>,<types incl>,<types excl>,<max depth>,<min size>,<max size>,<min age>,<max age>,<regex excl dir>]

 目录项数量。
 返回值：*Integer*。
 请参阅支持的平台。

参数：

- **dir** - 目录的绝对路径；
- **regex incl** - 用于描述要包含的实体（文件、目录、符号链接）名称模式的正则表达式；为空时包含全部实体（默认值）；
- **regex excl** - 用于描述要排除的实体（文件、目录、符号链接）名称模式的正则表达式；为空时不排除任何实体（默认值）；
- **types incl** - 要计数的目录项类型，可选值：
  - *file* - 普通文件；
  - *dir* - 子目录；
  - *sym* - 符号链接；
  - *sock* - 套接字；
  - *bdev* - 块设备；
  - *cdev* - 字符设备；
  - *fifo* - FIFO；
  - *dev* - 与"bdev,cdev" 同义；
  - *all* - 所有类型（默认），即"file,dir,sym,sock,bdev,cdev,fifo"。多个类型必须使用逗号分隔，并加引号。
- **types excl** - 不计数的目录项类型（请参阅 `types incl`）。如果某个目录项类型同时存在于 `types incl` 和 `types excl` 中，则不计数该类型的目录项。
- **max depth** - 要遍历的子目录最大深度：
  - **-1**（默认）- 不限制；
  - **0** - 不进入子目录。
- **min size** - 要计数文件的最小大小（以字节为单位）。不会计数小于此大小的文件。可以使用内存后缀。
- **max size** - 要计数文件的最大大小（以字节为单位）。不会计数大于此大小的文件。可以使用内存后缀。
- **min age** - 要计数目录项的最小存在时间（以秒为单位）。不会计数较新的目录项。可以使用时间后缀。
- **max age** - 要计数目录项的最大存在时间（以秒为单位）。不会计数存在时间达到或超过此值的目录项（修改时间）。可以使用时间后缀。
- **regex excl dir** - 用于描述要排除的目录名称模式的正则表达式。该目录的所有内容都将被排除（不同于 `regex_excl`）。

注释：

- 在 Windows 上，此监控项支持 UNC 路径，这可能会带来安全风险。
- 不支持环境变量，例如 `%APP_HOME%`、`$HOME` 和 `%TEMP%`。
- 从不计数伪目录"." 和"..."。
- 遍历目录时从不跟随符号链接。
- 在计算目录项数量时，会对文件和目录应用 `regex incl` 和 `regex excl`，但在选择要遍历的子目录时会忽略这两个参数（如果 `regex incl` 为 `(?i)^.+\.zip$` 且未设置 `max depth`，则会遍历所有子目录，但只计数 zip 类型的文件）。
- 执行时间受agent 配置中的超时值限制（3 秒）。由于遍历大型目录可能需要更长时间，因此不会返回数据，监控项将变为不受支持。不会返回部分计数结果。
- 按大小筛选时，只有普通文件的大小具有实际意义。在 Linux 和 BSD 下，目录也具有非零大小（通常为几 KB）。设备的大小为零，例如，`/dev/sda1` 的大小并不反映相应分区的大小。因此，使用 `<min_size>` 和 `<max_size>` 时，建议将 `<types_incl>` 指定为*"file"*，以避免出现意外结果。

示例：

```
vfs.dir.count[/dev] # 监控 /dev 中的设备数量（Linux）
vfs.dir.count["C:\Users\ADMINI~1\AppData\Local\Temp"] # 监控临时目录中的文件数量
```

vfs.dir.get[dir,<regex incl>,<regex excl>,<types incl>,<types excl>,<max depth>,<min size>,<max size>,<min age>,<max age>,<regex excl dir>]

 目录条目列表。
 返回值：*JSON* 字符串。
 请参阅支持的平台。

参数：

- **dir** - 目录的绝对路径；
- **regex incl** - 用于描述要包含的实体（文件、目录、符号链接）名称模式的正则表达式；为空时包含全部实体（默认值）；
- **regex excl** - 用于描述要排除的实体（文件、目录、符号链接）名称模式的正则表达式；为空时不排除任何实体（默认值）；
- **types incl** - 要列出的目录条目类型，可选值：
  - *file* - 普通文件；
  - *dir* - 子目录；
  - *sym* - 符号链接；
  - *sock* - 套接字；

    - *bdev* - 块设备；
    - *cdev* - 字符设备；
    - *fifo* - FIFO；
    - *dev* - 与"bdev,cdev" 同义；
    - *all* - 所有类型（默认），即"file,dir,sym,sock,bdev,cdev,fifo"。多个类型必须以逗号分隔，并使用引号括起。
- **types excl** - 不要列出的目录条目类型（请参阅 `types incl`）。如果某个条目类型同时存在于 `types incl` 和 `types excl` 中，则不会列出该类型的目录条目。
- **max depth** - 要遍历的子目录最大深度：
    - **-1**（默认）- 不限制；
    - **0** - 不进入子目录。
- **min size** - 要列出的文件的最小大小（以字节为单位）。不会列出更小的文件。可以使用内存后缀。
- **max size** - 要列出的文件的最大大小（以字节为单位）。不会列出更大的文件。可以使用内存后缀。
- **min age** - 要列出的目录条目的最小存在时间（以秒为单位）。不会列出较新的条目。可以使用时间后缀。
- **max age** - 要列出的目录条目的最大存在时间（以秒为单位）。不会列出存在时间达到或超过该值的条目（修改时间）。可以使用时间后缀。
- **regex excl dir** - 用于描述要排除的目录名称模式的正则表达式。该目录中的所有内容都将被排除（不同于 `regex excl`）。

注释：

- 在 Windows 上，此监控项遵循 UNC 路径，这可能会带来安全风险。
- 不支持环境变量，例如 `%APP_HOME%`、`$HOME` 和 `%TEMP%`。
- 从不列出伪目录"." 和"..."。
- 遍历目录时从不跟随符号链接。
- 生成条目列表时，会对文件和目录应用 `regex incl` 和 `regex excl`，但在选择要遍历的子目录时会忽略这两个参数（如果 `regex incl` 为 `(?i)^.+\.zip$` 且未设置 `max depth`，则会遍历所有子目录，但只统计 zip 类型的文件）。
- 执行时间受agent 配置中的超时值限制。由于遍历大型目录可能需要更长时间，因此不会返回数据，并且该监控项将变为不受支持。不会返回部分列表。
- 按大小筛选时，只有普通文件的大小具有实际意义。在 Linux 和 BSD 下，目录也具有非零大小（通常为几 KB）。设备的大小为零，例如，`/dev/sda1` 的大小并不反映相应分区的大小。因此，使用 `min size` 和 `max size` 时，建议将 `types incl` 指定为"*file*"，以避免出现意外结果。

示例：

```
vfs.dir.get[/dev] # 获取 /dev 中的设备列表（Linux）
vfs.dir.get["C:\Users\ADMINI~1\AppData\Local\Temp"] # 获取临时目录中的文件列表
```

vfs.dir.size[dir,<regex incl>,<regex excl>,<mode>,<max depth>,<regex excl dir>]

 目录大小（以字节为单位）。
 返回值：*Integer*。
 支持的平台：Linux、Windows。该监控项也可能在其他类 UNIX 平台上工作。

参数：

- **dir** - 目录的绝对路径；
- **regex incl** - 描述要包含的实体（文件、目录、符号链接）名称模式的正则表达式；如果为空，则包含全部（默认值）；
- **regex excl** - 描述要排除的实体（文件、目录、符号链接）名称模式的正则表达式；如果为空，则不排除任何内容（默认值）；
- **mode** - 可能的值：
    - *apparent*（默认）- 获取表观文件大小，而不是磁盘占用（相当于 `du -sb dir`）；
    - *disk* - 获取磁盘占用（相当于 `du -s -B1 dir`）。与 du 命令不同，vfs.dir.size 监控项在计算目录大小时会考虑隐藏文件（在 dir 内部相当于 `du -sb .[^.]* *`）。
- **max depth** - 要遍历的子目录最大深度：
    - **-1**（默认）- 不限制；
    - **0** - 不进入子目录。
- **regex excl dir** - 描述要排除的目录名称模式的正则表达式。目录中的所有内容都将被排除（与 `regex excl` 不同）

注释：

- 在 Windows 上，该监控项会跟随 UNC 路径，这可能构成安全风险。
- 仅计算至少对 *zabbix* 用户具有读取权限的目录。对于仅具有读取权限的目录，仅计算目录本身的大小。具有读取和执行权限的目录会连同其内容一起计算。
- 对于大型目录或较慢的磁盘驱动器，由于agent 和server/proxy 配置文件中的 Timeout 设置，该监控项可能会超时。请根据需要增加超时值。
- 文件大小限制取决于大文件支持。

示例：

```
vfs.dir.size[/tmp,log] # 计算 /tmp 中所有名称包含 'log' 的文件大小
vfs.dir.size[/tmp,log,^.+\.old$] # 计算 /tmp 中所有名称包含 'log' 的文件大小，但排除名称以 '.old' 结尾的文件
```

vfs.file.cksum[file,<mode>]

 文件校验和，使用 UNIX cksum 算法计算。
 返回值：*Integer* - 当 mode 为 *crc32* 时，*String* - 当 mode 为 *md5*、*sha256* 时。
 参见supported platforms。

参数：

- **file** - 文件的完整路径；
- **mode** - *crc32*（默认）、*md5* 或 *sha256*。

注释：

- 在 Windows 上，此监控项遵循 UNC 路径，这可能带来安全风险。
- 文件大小限制取决于large file support。

示例：

```
vfs.file.cksum[/etc/passwd]
```

返回值示例（分别为 crc32/md5/sha256）：

```
675436101
9845acf68b73991eb7fd7ee0ded23c44
ae67546e4aac995e5c921042d0cf0f1f7147703aa42bfbfb65404b30f238f2dc
```

vfs.file.contents[file,<encoding>]

 检索文件的内容[7]。
 返回值：*Text*。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **encoding** - 代码页标识符。

注释：

- 在 Windows 上，此监控项会遵循 UNC 路径，这可能构成安全风险。
- 返回值限制为 16MB（包括会被截断的尾随空白）；数据库限制也同样适用。
- 如果文件为空，或仅包含 LF/CR 字符，则返回空字符串。
- 字节顺序标记（BOM）不会包含在输出中。

示例：

```
vfs.file.contents[/etc/passwd]
```

vfs.file.exists[file,<types incl>,<types excl>]

 检查文件是否存在。
 返回值：0 - 未找到；1 - 指定类型的文件存在。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **types incl** - 要包含的文件类型列表，可用值：
  - *file*（普通文件，默认值（如果未设置 types_excl））；
  - *dir*（目录）；
  - *sym*（符号链接）；
  - *sock*（套接字）；
  - *bdev*（块设备）；
  - *cdev*（字符设备）；
  - *fifo*（FIFO）；
  - *dev*（与"bdev,cdev" 同义）；
  - *all*（上述所有类型，如果设置了 types_excl，则为默认值）。
- **types excl** - 要排除的文件类型列表，可用值请参见 types_incl（默认不排除任何类型）

注释：

- 在 Windows 上，此监控项会跟随 UNC 路径，这可能带来安全风险。
- 多个类型必须用逗号分隔，并且整个集合要用引号 "" 括起来。
- 如果同一种类型同时出现在 <types_incl> 和 <types_excl> 中，则该类型的文件会被排除。
- 文件大小限制取决于大文件支持。

示例：

```
vfs.file.exists[/tmp/application.pid]
vfs.file.exists[/tmp/application.pid,"file,dir,sym"]
vfs.file.exists[/tmp/application_dir,dir]
```

vfs.file.get[file]

 返回有关文件的信息。
 返回值：*JSON* 字符串。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径

注释：

- 在 Windows 上，此监控项遵循 UNC 路径，这可能带来安全风险。
- UNIX-like 系统支持的文件类型：普通文件、目录、符号链接、套接字、块设备、字符设备、FIFO。
- 文件大小限制取决于大文件支持。

示例：

```
vfs.file.get[/etc/passwd] #return a JSON with information about the /etc/passwd file (type, user, permissi
```

vfs.file.md5sum[file]

 文件的 MD5 校验和。
 返回值：字符串（文件的 MD5 哈希值）。
 请参见supported platforms。

参数：

- **file** - 文件的完整路径

注释：

- 在 Windows 上，此监控项遵循 UNC 路径，这可能带来安全风险。
- 文件大小限制取决于large file support。

示例：

```
vfs.file.md5sum[/usr/local/etc/zabbix_agentd.conf]
```

返回值示例：

```
b5052decb577e0fffd622d6ddc017e82
```

vfs.file.owner[file,<ownertype>,<resulttype>]

 获取文件的所有者。
 返回值：*String*。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **ownertype** - *user*（默认）或 *group*（仅 Unix）；
- **resulttype** - *name*（默认）或 *id*；对于 id - 在 Unix 上返回 uid/gid，在 Windows 上返回 SID。

注释：

- 在 Windows 上，此监控项会跟随 UNC 路径，这可能带来安全风险。
- 文件大小限制取决于大文件支持。

示例：

```
vfs.file.owner[/tmp/zabbix_server.log] # 返回 /tmp/zabbix_server.log 的文件所有者
vfs.file.owner[/tmp/zabbix_server.log,,id] # 返回 /tmp/zabbix_server.log 的文件所有者 ID
```

vfs.file.permissions[file]

 返回一个包含 UNIX 权限八进制数字的 4 位字符串。
 返回值：*String*。
 支持的平台：Linux。该监控项也可能在其他类 UNIX 平台上工作。

参数：

- **file** - 文件的完整路径

文件大小限制取决于大文件支持。

示例：

```
vfs.file.permissions[/etc/passwd] # 返回 /etc/passwd 的权限，例如 '0644'
```

vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]

 获取文件中的字符串[7]。
 返回值：包含匹配字符串的行，或按可选 output 参数指定的内容返回。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **regexp** - 描述所需模式的正则表达式；
- **encoding** - 代码页标识符；
- **start line** - 要搜索的起始行号（默认是文件的第一行）；
- **end line** - 要搜索的结束行号（默认是文件的最后一行）；
- **output** - 可选的输出格式化模板。**\0** 转义序列会被匹配到的文本部分替换（从匹配开始的第一个字符到匹配结束的字符），而 **\N**（其中 N=1...9）转义序列会被第 N 个匹配组替换（如果 N 超过捕获组数量，则替换为空字符串）。

注释：

- 在 Windows 上，此监控项会遵循 UNC 路径，这可能带来安全风险。
- 文件大小限制取决于大文件支持。
- 仅返回第一条匹配的行。
- 如果没有任何行匹配该表达式，则返回空字符串。
- 字节顺序标记（BOM）会从输出中排除。
- 使用 output 参数进行内容提取是在 agent 上完成的。

示例：

```
vfs.file.regexp[/etc/passwd,zabbix]
vfs.file.regexp[/path/to/some/file,"([0-9]+)$",,3,5,\1]
vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)",,,,\1] → 获取用户 *zabbix* 的 ID
```

vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]

 在文件中查找字符串[7]。
 返回值：0 - 未找到匹配；1 - 找到。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **regexp** - 描述所需模式的正则表达式；
- **encoding** - 代码页标识符；
- **start line** - 要搜索的起始行号（默认是文件的第一行）；
- **end line** - 要搜索的结束行号（默认是文件的最后一行）。

注释：

- 在 Windows 上，此监控项会遵循 UNC 路径，这可能带来安全风险。
- 文件大小限制取决于大文件支持。
- 字节顺序标记（BOM）会被忽略。

示例：

```
vfs.file.regmatch[/var/log/app.log,error]
```

vfs.file.size[file,<mode>]

 文件大小（以字节为单位）。
 返回值：*Integer*。
 参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **mode** - 可选值：
  - *bytes*（默认）；
  - *lines*（空行也会计数）。

注释：

- 在 Windows 上，此监控项会跟随 UNC 路径，这可能带来安全风险。
- 该文件必须对用户 *zabbix* 具有读取权限。
- 文件大小限制取决于大文件支持。

示例：

```
vfs.file.size[/var/log/syslog]
```

vfs.file.time[file,<mode>]

 文件时间信息。
 返回值：*Integer*（Unix 时间戳）。
 请参见支持的平台。

参数：

- **file** - 文件的完整路径；
- **mode** - 可能的值：
  - *modify*（默认）- 文件内容最后修改时间（默认）；
  - *access* - 文件最后访问时间；
  - *change* - 文件属性最后更改时间。

注释：

- 在 Windows 上，此监控项会跟随 UNC 路径，这可能带来安全风险。
- 文件大小限制取决于大文件支持。

示例：

```
vfs.file.time[/etc/passwd,modify]
```

vfs.fs.discovery

 已挂载文件系统及其类型和挂载选项的列表。用于低级别发现。
 返回值：*JSON string*。
 支持的平台：Linux、FreeBSD、Solaris、HP-UX、AIX、MacOS X、OpenBSD、NetBSD、Windows。

vfs.fs.get[<mode>,<mountpoint>]

 已挂载文件系统的列表，包括其类型和挂载选项；在 *full* 模式下，还包括可用磁盘空间和 inode 统计信息。可用于低级别发现。
 返回值：*JSON* 字符串。
 支持的平台：Linux、FreeBSD、Solaris、HP-UX、AIX、MacOS X、OpenBSD、NetBSD、Windows。

参数：

- **mode** - 可能的值：
  - *full*（默认）- 文件系统属性，以及磁盘空间和 inode 统计信息；
  - *short* - 仅文件系统属性，不包括磁盘空间和 inode 统计信息；
- **mountpoint** - 要返回其数据的挂载点（例如 `/var` 或 `C:`）。如果未指定，则返回所有已挂载的文件系统。

注释：

- 在 *short* 模式下，仅返回文件系统名称、类型和挂载选项（在 Windows 上：名称、类型、标签和驱动器类型）。不会查询文件系统的统计信息；在类 Unix 系统上，仅读取已挂载文件系统的列表（例如 Linux 上的 `/proc/mounts`），因此该检查不会触发自动挂载，也不受无响应网络文件系统的影响。因此，*short* 模式适用于低级别发现。
- **mountpoint** 值必须与 `fsname` 属性完全匹配。在 Windows 上，比较不区分大小写；请指定不带末尾反斜杠的挂载点（例如 `C:`）。
- 如果指定了 **mountpoint**，则仅返回该挂载点的数据。如果没有已挂载的文件系统与其匹配，则返回空数组（`[]`）。
- 如果同一目录上挂载了多个文件系统，则输出可能包含多个具有相同 `fsname` 的条目。
- 在 *full* 模式下，仅当 Zabbix agent 用户对挂载点路径中的每个目录都具有遍历权限时，才会返回可用磁盘空间和 inode 统计信息；否则，该文件系统将从输出中省略。这不适用于 *short* 模式，因此 *short* 模式可能会列出在 *full* 模式输出中缺失的文件系统。
- `fslabel` 和 `fsdrivetype` 属性仅在 Windows 上返回；在 Windows 上不返回 `inodes` 和 `options` 属性。
- inode 数量等于零的文件系统（动态 inode 的文件系统可能出现这种情况，例如 btrfs）也会被报告。
- 另请参阅：已挂载文件系统的发现。

示例：

```
vfs.fs.get # 包含统计信息的所有已挂载文件系统；与 vfs.fs.get[full] 相同
vfs.fs.get[short] # 不包含统计信息的所有已挂载文件系统
vfs.fs.get[full,/var] # 仅 /var 挂载点的统计信息
vfs.fs.get[short,C:] # 仅 C: 卷的属性（Windows）
```

vfs.fs.inode[fs,<mode>]

 inode 的数量或百分比。
 返回值：*Integer* - 表示数量；*Float* - 表示百分比。
 请参见支持的平台。

参数：

- **fs** - 文件系统；
- **mode** - 可选值：*total*（默认）、*free*、*used*、*pfree*（空闲，百分比）或 *pused*（已使用，百分比）。

如果 inode 计数等于零，则在具有动态 inode 的文件系统中可能会出现这种情况（例如 btrfs），此时 pfree/pused 值将分别报告为 “100” 和 “0”。

示例：

```
vfs.fs.inode[/,pfree]
```

vfs.fs.size[fs,<mode>]

 磁盘空间大小，以字节或占总量的百分比表示。
 返回值：*Integer* - 用于字节；*Float* - 用于百分比。
 请参见支持的平台。

参数：

- **fs** - 文件系统；
- **mode** - 可选值：*total*（默认）、*free*、*used*、*pfree*（空闲，百分比）或 *pused*（已使用，百分比）。

注释：

- 如果文件系统未挂载，则返回挂载点所在本地文件系统的大小。
- 文件系统的保留空间会被计入，并且在使用 *free* 模式时不包含在内。

示例：

```
vfs.fs.size[/tmp,free]
```

vm.memory.size[<mode>]

 内存大小，以字节或占总内存的百分比表示。
 返回值：*Integer* - 字节；*Float* - 百分比。
 请参见支持的平台。

参数：

- **mode** - 可选值：*total*（默认）、*active*、*anon*、*buffers*、*cached*、*exec*、*file*、*free*、*inactive*、*pinned*、*shared*、*slab*、*wired*、*used*、*pused*（已使用，百分比）、*available* 或 *pavailable*（可用，百分比）。

说明：

- 此监控项接受三类参数：
  - *total* - 内存总量；
  - 平台特定的内存类型：*active*、*anon*、*buffers*、*cached*、*exec*、*file*、*free*、*inactive*、*pinned*、*shared*、*slab*、*wired*；
  - 用户级估算的已使用和可用内存量：*used*、*pused*、*available*、*pavailable*。
- *active* 模式参数仅在 FreeBSD、HP-UX、MacOS X、OpenBSD、NetBSD 上受支持。
- *anon*、*exec*、*file* 模式参数仅在 NetBSD 上受支持。
- *buffers* 模式参数仅在 Linux、FreeBSD、OpenBSD、NetBSD 上受支持。
- *cached* 模式参数仅在 Linux、FreeBSD、AIX、OpenBSD、NetBSD 上受支持。
- *inactive*、*wired* 模式参数仅在 FreeBSD、MacOS X、OpenBSD、NetBSD 上受支持。
- *pinned* 模式参数仅在 AIX 上受支持。
- *shared* 模式参数仅在 Linux 2.4、FreeBSD、OpenBSD、NetBSD 上受支持。
- 另请参见此监控项的附加详细信息。

示例：

```
vm.memory.size[pavailable]
```

web.page.get[host,<path>,<port>,<redirect_limit>]

 获取网页内容。
 返回值：网页源代码文本（包括标头）。
 请参阅支持的平台。

参数：

- **host** - 主机名或 URL（格式为 `scheme://host:port/path`，其中只有 *host* 是必需的）。允许的 URL 方案：*http*、*https*[4]。如果缺少方案，则按 *http* 处理。如果指定了 URL，则 `path` 和 `port` 必须为空。连接到需要身份验证的服务器时，可以指定用户名/密码，例如：`http://user:password@www.example.com`，但这仅在支持 cURL 时可用[4]。主机名支持 Punycode。
- **path** - HTML 文档的路径（默认为 `/`）；
- **port** - 端口号（默认为 HTTP 的 80）
- **redirect_limit** - 要跟随的最大重定向次数[4]，范围为 0 到 1000（默认为 0 - 不跟随重定向）。

注释：

- 如果 `host` 中指定的资源不存在或不可用，或者 `redirect_limit` 大于 0 且重定向次数超过该值，则此监控项将变为不受支持。
- `host` 可以是主机名、域名、IPv4 或 IPv6 地址。但是，对于 IPv6 地址，Zabbix agent 必须在启用 IPv6 支持的情况下进行编译。
- 如果跟随重定向，则返回值包含每个重定向响应的标头，后面是最终响应的标头和正文。

示例：

```
web.page.get[www.example.com,index.php,80]
web.page.get[https://www.example.com]
web.page.get[https://blog.example.com/?s=zabbix]
web.page.get[localhost:80]
web.page.get["[::1]/server-status"]
web.page.get[http://www.example.com,,,5]
```

web.page.perf[host,<path>,<port>,<redirect_limit>]

 完整网页的加载时间（以秒为单位）。
 返回值：*Float*。
 请参阅支持的平台。

参数：

- **host** - 主机名或 URL（格式为 `scheme://host:port/path`，其中只有 *host* 是必需的）。允许的 URL 方案：*http*、*https*[4]。如果缺少方案，则将其视为 *http*。如果指定了 URL，则 `path` 和 `port` 必须为空。连接到需要身份验证的服务器时指定用户名/密码，例如：`http://user:password@www.example.com`，仅在支持 cURL 时可用 [4]。主机名支持 Punycode。
- **path** - HTML 文档的路径（默认为 `/`）；
- **port** - 端口号（HTTP 默认为 80）
- **redirect_limit** - 要跟随的最大重定向次数 [4]，范围为 0 到 1000（默认为 0 - 不跟随重定向）。

注释：

- 如果 `host` 中指定的资源不存在或不可用，或者 `redirect_limit` 大于 0 且重定向次数超过该值，则此监控项将变为不受支持。
- `host` 可以是主机名、域名、IPv4 或 IPv6 地址。但是，对于 IPv6 地址，Zabbix agent 必须在启用 IPv6 支持的情况下进行编译。
- 如果跟随重定向，加载时间包括所有重定向所花费的时间。

示例：

```
web.page.perf[www.example.com,index.php,80]
web.page.perf[https://www.example.com]
web.page.perf[http://www.example.com,,,5]
```

web.page.regexp[host,<path>,<port>,regexp,<length>,<output>,<redirect_limit>]

 在网页中查找字符串。
 返回值：匹配的字符串，或按照可选的 `output` 参数指定的格式返回。
 请参阅支持的平台。

参数：

- **host** - 主机名或 URL（格式为 `scheme://host:port/path`，其中只有 *host* 是必需的）。允许的 URL 方案：*http*、*https*[4]。如果缺少方案，则将其视为 *http*。如果指定了 URL，则 `path` 和 `port` 必须为空。连接到需要身份验证的服务器时指定用户名/密码（例如：`http://user:password@www.example.com`）仅在支持 cURL 时可用 [4]。主机名支持 Punycode。
- **path** - HTML 文档的路径（默认为 `/`）。
- **port** - 端口号（HTTP 的默认值为 80）。
- **regexp** - 描述所需模式的正规表达式。
- **length** - 要返回的最大字符数；
- **output** - 可选的输出格式模板。**\0** 转义序列将替换为匹配的文本部分（从匹配开始的第一个字符到匹配结束的字符），而 **\N**（其中 N=1...9）转义序列将替换为第 N 个匹配组（如果 N 超出捕获组数量，则替换为空字符串）。
- **redirect_limit** - 要跟随的最大重定向次数 [4]，范围为 0 到 1000（默认为 0 - 不跟随重定向）。

注释：

- 如果 `host` 中指定的资源不存在或不可用，或者 `redirect_limit` 大于 0 且重定向次数超过该值，则此监控项将变为不受支持。
- `host` 可以是主机名、域名、IPv4 或 IPv6 地址。但是，对于 IPv6 地址，Zabbix agent 必须在启用 IPv6 支持的情况下进行编译。
- 使用 `output` 参数提取内容的操作在 agent 上执行。
- 如果跟随重定向，也会搜索每个重定向响应的标头。

示例：

```
web.page.regexp[www.example.com,index.php,80,OK,2]
web.page.regexp[https://www.example.com,,,OK,2]
web.page.regexp[http://www.example.com,,,OK,2,,5]
```

agent.hostmetadata

 agent 主机元数据。
 返回值：*String*。
 参见supported platforms。

返回HostMetadata 或HostMetadataItem 参数的值；如果未定义任何参数，则返回空字符串。

agent.hostname

 agent 主机名。
 返回值：*String*。
 参见supported platforms。

返回：

- 作为被动检查 - agent 配置文件中Hostname 参数里列出的第一个主机的名称；
- 作为主动检查 - 当前主机名。

agent.ping

 agent 可用性检查。
 返回值：无 - 不可用；1 - 可用。
 请参见supported platforms。

使用 `nodata()` 触发器函数检查主机是否不可用。

agent.variant

Zabbix agent 的变体（Zabbix agent 或 Zabbix agent 2）。
 返回值：1 - Zabbix agent；2 - Zabbix agent 2。
 请参阅支持的平台。

agent.version

 Zabbix agent 的版本。
 返回值：*String*。
 请参见支持的平台。

返回值示例：

```
6.0.3
```

zabbix.stats[<ip>,<port>]

 返回一组 Zabbix 服务器或 proxy 的内部指标。如果指定了 `ip` 和 `port`，则从远程实例获取指标；否则，从本地实例获取。
 返回值：*JSON* 字符串。
 请参见支持的平台。

参数：

- **ip** - 要远程查询的服务器/proxy 的 IP/DNS/网络掩码列表（默认值为 `127.0.0.1`）；
- **port** - 要远程查询的服务器/proxy 端口（默认值为 10051）

注释：

- 此监控项返回一组选定的内部指标。有关详细信息，请参见Zabbix stats 的远程监控。
- 请注意，stats 请求只会接受目标实例上 `StatsAllowedIP` 服务器/proxy 参数中列出的地址。

zabbix.stats[<ip>,<port>,queue,<from>,<to>]

 返回队列中延迟于 Zabbix 服务器或 proxy 的受监控监控项数量。如果指定了 `ip` 和 `port`，则从远程实例获取指标；否则，从本地实例获取。
 返回值：*JSON string*。
 请参见支持的平台。

参数：

- **ip** - 要远程查询的服务器/proxy 的 IP/DNS/网络掩码列表（默认值为 `127.0.0.1`）；
- **port** - 要远程查询的服务器/proxy 的端口（默认值为 10051）
- **queue** - 常量（按原样使用）
- **from** - 至少延迟（默认值为 6 秒）
- **to** - 最多延迟（默认值为无穷大）

请注意，stats 请求仅会接受来自目标实例上 `StatsAllowedIP` server/proxy 参数中列出的地址。

脚注

[1]Linux 特有的说明。
Zabbix agent 必须对文件系统 `/proc` 具有只读访问权限。
来自 www.grsecurity.org 的内核补丁会限制非特权用户的访问权限。

[2] `vfs.dev.read[]`、`vfs.dev.write[]`：如果监控项值超过 3 小时未被访问，Zabbix agent 将终止 “过期” 的设备连接。
如果系统中的设备路径会动态变化，或者设备被手动移除，则可能发生这种情况。
另请注意，如果这些监控项使用 3 小时或更长的更新间隔，则始终会返回 “0”。

[3] `vfs.dev.read[]`、`vfs.dev.write[]`：如果第一个参数使用默认值 `all`，则键将返回汇总统计信息，其中包括所有块设备（如 sda、sdb）及其分区（sda1、sda2、sdb3...），以及基于这些块设备/分区创建的多设备（MD raid）和逻辑卷（LVM）。
在这种情况下，返回值只能视为相对值（随时间动态变化），而不能视为绝对值。

[4] 仅当 Zabbix agent 在编译时启用了 cURL 支持时，才支持 HTTPS 和后续重定向（`redirect_limit` 大于 0）；否则，该监控项将变为不受支持。Zabbix agent 2 在这两种情况下都不需要 cURL。

[5] 在 Solaris 系统（包括 Solaris 10 6/06）中，环回接口不支持 `bytes` 和 `errors` 值，因为内核不会存储和/或报告字节、错误和利用率统计信息。
但是，如果通过 net-snmp 监控 Solaris 系统，则可能会返回这些值，因为 net-snmp 携带了源自 cmu-snmp 的旧代码，该代码最早可追溯到 1997 年：当无法从接口统计信息中读取字节值时，它会返回数据包计数器（环回接口确实存在该计数器）乘以 308 这一任意值。
这相当于假设数据包的平均长度为 308 个八位字节；这是一个非常粗略的估计，因为 Solaris 系统环回接口的 MTU 限制为 8892 字节。
不应认为这些值正确，甚至不应认为它们接近准确。
这些只是推测值。
Zabbix agent 不会进行任何猜测，但 net-snmp 会为这些字段返回一个值。

[6]Solaris 上从 `/proc/pid/psinfo` 获取的命令行长度限制为 80 字节，并且包含进程启动时的命令行内容。

[7] 可以使用 `vfs.file.contents[]`、`vfs.file.regexp[]`、`vfs.file.regmatch[]` 监控项来获取文件内容。
如果希望限制对包含敏感信息的特定文件的访问，请以无权查看这些文件的用户身份运行 Zabbix agent。

与命令行工具一起使用

请注意，当使用命令行通过 zabbix_agentd 或 zabbix_get 测试或使用监控项键值时，还应考虑 shell 语法。

例如，如果键值的某个参数必须用双引号括起来，则必须显式转义双引号；否则，shell 会将其作为特殊字符去掉，这些双引号将不会传递给 Zabbix 工具。

示例：

```
zabbix_agentd -t 'vfs.dir.count[/var/log,,,"file,dir",,0]'

zabbix_agentd -t vfs.dir.count[/var/log,,,\"file,dir\",,0]
```

编码设置

为确保获取的数据不会损坏，您可以在 encoding 参数中指定用于处理检查的正确编码（例如 vfs.file.contents）。支持的编码列表（代码页标识符）可在 libiconv（GNU Project）的文档中找到，也可参阅 Microsoft Windows SDK 文档中的 "Code Page Identifiers"。请注意，Microsoft 有时会将某些代码页标记为 "仅适用于托管应用程序"，这意味着它们只能通过.NET 运行时公开，而无法通过原生 Win32 API 使用。Zabbix agent 实现了自己的编码转换逻辑，因此即使原生 Windows 函数不提供这些代码页，Zabbix agent 仍支持它们。

如果未在 encoding 参数中指定编码，则会应用以下解析策略：

- 如果未指定编码（或编码为空字符串），则假定编码为 UTF-8，数据将 "按原样" 处理；
- BOM 分析 - 适用于监控项 vfs.file.contents、vfs.file.regexp、vfs.file.regmatch。系统会尝试使用文件开头的字节顺序标记（BOM）确定正确的编码。如果不存在 BOM，则改为应用标准解析策略（见上文）。

agent 监控项故障排查

对于被动检查，为防止服务器向 agent 发送的请求先超时，导致监控项无法获取任何值，请注意以下事项：

- 如果 agent 版本低于服务器版本，则监控项配置中的 Timeout 值（或全局超时）可能需要高于agent 配置文件中的 Timeout 值。
- 如果 agent 版本高于服务器版本，则服务器配置文件中的 Timeout 值可能需要高于agent 配置文件中的 Timeout 值。

Zabbix agent 2

Zabbix agent 2 支持 Zabbix agent 在Unix 和Windows 上支持的所有监控项键值。本页提供仅可与 Zabbix agent 2 一起使用的附加监控项键值的详细信息，并按其所属插件分组。

监控项键值列表未包含参数和附加信息。单击监控项键值可查看完整详细信息。

监控项键值	描述	插件
ceph.df.details	集群的数据使用情况以及在各个池之间的分布。	Ceph (可加载)
ceph.osd.stats	汇总的以及按 OSD 分别统计的信息。	
ceph.osd.discovery	已发现的 OSD 列表。	
ceph.osd.dump	OSD 的使用阈值和状态。	
ceph.ping	测试是否可以建立到 Ceph 的连接。	
ceph.pool.discovery	已发现的池列表。	
ceph.status	整个集群的状态。	
docker.container_info	有关容器的底层信息。	Docker
docker.container_stats	容器资源使用统计信息。	
docker.containers	返回容器列表。	
docker.containers.discovery	返回容器列表。用于低级别发现。	
docker.data.usage	有关当前数据使用情况的信息。	
docker.images	返回镜像列表。	
docker.images.discovery	返回镜像列表。用于低级别发现。	
docker.info	系统信息。	
docker.ping	测试 Docker 守护进程是否存活。	
ember.get	返回所需设备的结果。	Ember+ (可加载)
memcached.ping	测试连接是否存活。	Memcached
memcached.stats	获取 STATS 命令的输出。	
mongodb.cfg.discovery	返回已发现的配置服务器列表。	MongoDB (可加载)
mongodb.collection.stats	返回指定集合的各种存储统计信息。	
mongodb.collections.discovery	返回已发现的集合列表。	
mongodb.collections.usage	返回集合的使用统计信息。	
mongodb.connpool.stats	返回当前数据库实例到分片集群或副本集其他成员的已打开出站连接信息。	
mongodb.db.stats	返回反映指定数据库系统状态的统计信息。	
mongodb.db.discovery	返回已发现的数据库列表。	

监控项键值	描述	插件
mongodb.jumbo_chunks.count	返回 jumbo chunk 的数量。	
mongodb.oplog.stats	使用从 oplog 轮询的数据返回副本集状态。	
mongodb.ping	测试连接是否存活。	
mongodb.rs.config	返回副本集的当前配置。	
mongodb.rs.status	从运行该方法的成员视角返回副本集状态。	
mongodb.server.status	返回数据库状态。	
mongodb.sh.discovery	返回集群中存在的已发现分片列表。	
mongodb.version	返回数据库服务器版本。	
mqtt.get	订阅所提供 broker 的一个或多个特定主题（支持通配符），并等待发布消息。	MQTT
mssql.availability.group.get	返回可用性组。	MSSQL (可加载)
mssql.custom.query	返回自定义查询的结果。	
mssql.db.get	返回所有可用的 MSSQL 数据库。	
mssql.job.status.get	返回作业状态。	
mssql.last.backup.get	返回所有数据库的上次备份时间。	
mssql.local.db.get	返回参与 Always On 可用性组和副本（主副本或辅助副本），且位于建立连接的服务器上的数据库。	
mssql.mirroring.get	返回镜像信息。	
mssql.nonlocal.db.get	返回参与 Always On 可用性组和副本（主副本或辅助副本），且位于其他服务器上的数据库（该数据库不在建立连接的 SQL Server 实例本地）。	
mssql.perfcounter.get	返回性能计数器。	
mssql.ping	测试连接是否存活。	
mssql.quorum.get	返回仲裁信息。	
mssql.quorum.member.get	返回仲裁成员。	
mssql.replica.get	返回副本。	
mssql.version	返回 MSSQL 版本。	
mysql.custom.query	返回自定义查询的结果。	MySQL
mysql.db.discovery	返回 MySQL 数据库列表。	
mysql.db.size	数据库大小，单位为字节。	
mysql.get_status_variables	全局状态变量的值。	
mysql.ping	测试连接是否存活。	
mysql.replication.discovery	返回 MySQL 复制列表。	
mysql.replication.get_slave_status	复制状态。	
mysql.version	MySQL 版本。	
net.dns.get	执行 DNS 查询并返回详细的 DNS 记录信息。	Network
nvml.device.count	GPU 设备数量。	NVIDIA GPU (可加载)
nvml.device.decoder.utilization	GPU 设备解码器利用率。	
nvml.device.ecc.mode	GPU 设备 ECC 模式。	
nvml.device.encoder.stats.get	GPU 设备编码器统计信息。	
nvml.device.encoder.utilization	GPU 设备编码器利用率。	
nvml.device.energy.consumption	GPU 设备能耗。	
nvml.device.errors.memory	GPU 设备 ECC 内存错误信息。	
nvml.device.errors.register	GPU 设备 ECC 寄存器错误信息。	
nvml.device.fan.speed.avg	GPU 设备风扇平均转速。	
nvml.device.get	发现的 GPU，包含 UUID 和名称。	
nvml.device.graphics.frequency	GPU 设备图形时钟频率。	
nvml.device.memory.bar1.get	GPU 设备 BAR1 内存信息。	
nvml.device.memory.fb.get	GPU 设备帧缓冲内存信息。	
nvml.device.memory.frequency	GPU 设备内存时钟频率。	
nvml.device.pci.utilization	GPU 设备 PCI 利用率。	
nvml.device.performance.state	GPU 设备性能状态。	
nvml.device.power.limit	GPU 设备功率限制。	
nvml.device.power.usage	GPU 设备功耗。	
nvml.device.serial	GPU 设备序列号。	
nvml.device.sm.frequency	GPU 设备流式多处理器时钟频率。	
nvml.device.temperature	GPU 设备温度。	
nvml.device.utilization	GPU 设备利用率统计信息。	
nvml.device.video.frequency	GPU 设备视频时钟频率。	
nvml.system.driver.version	NVIDIA 驱动程序版本。	

监控项键值	描述	插件
nvml.version	NVML 库版本。	
oracle.diskgroups.stats	返回 Automatic Storage Management (ASM) 磁盘组统计信息。	Oracle
oracle.diskgroups.discovery	返回 ASM 磁盘组列表。	
oracle.archive.info	归档日志统计信息。	
oracle.cdb.info	Container Databases (CDBs) 信息。	
oracle.custom.query	自定义查询的结果。	
oracle.datafiles.stats	返回数据文件统计信息。	
oracle.db.discovery	返回数据库列表。	
oracle.fra.stats	返回 Fast Recovery Area (FRA) 统计信息。	
oracle.instance.info	实例统计信息。	
oracle.pdb.info	Pluggable Databases (PDBs) 信息。	
oracle.pdb.discovery	返回 PDB 列表。	
oracle.pga.stats	返回 Program Global Area (PGA) 统计信息。	
oracle.ping	测试是否可以建立到 Oracle 的连接。	
oracle.proc.stats	返回进程统计信息。	
oracle.redolog.info	来自控制文件的日志文件信息。	
oracle.sga.stats	返回 System Global Area (SGA) 统计信息。	
oracle.sessions.stats	返回会话统计信息。	
oracle.sys.metrics	返回一组系统指标值。	
oracle.sys.params	返回一组系统参数值。	
oracle.ts.stats	返回表空间统计信息。	
oracle.ts.discovery	返回表空间列表。	
oracle.user.info	返回 Oracle 用户信息。	
oracle.version	返回数据库服务器版本。	
pgsql.autovacuum.count	autovacuum worker 数量。	PostgreSQL (可加载)
pgsql.archive	有关归档文件的信息。	
pgsql.bgwriter	数据库集群的检查点总数，按检查点类型细分。	
pgsql.cache.hit	PostgreSQL 缓冲区缓存命中率。	
pgsql.connections	按类型返回连接。	
pgsql.custom.query	返回自定义查询的结果。	
pgsql.db.age	数据库中最老 FrozenXID 的年龄。	
pgsql.db.bloating_tables	每个数据库中的膨胀表数量。	
pgsql.db.discovery	PostgreSQL 数据库列表。	
pgsql.db.size	数据库大小，单位为字节。	
pgsql.dbstat	收集每个数据库的统计信息。	
pgsql.dbstat.sum	集群中所有数据库的汇总数据。	
pgsql.locks	每个数据库已授予锁的信息。	
pgsql.oldest.xid	最老 XID 的年龄。	
pgsql.ping	测试连接是否存活。	
pgsql.queries	按执行时间统计的查询指标。	
pgsql.replication.count	备用服务器数量。	
pgsql.replication.process	每个发送进程的 flush lag、write lag 和 replay lag。	
pgsql.replication.process.discovery	复制进程名称发现。	
pgsql.replication.recovery_role	恢复状态。	
pgsql.replication.status	复制状态。	
pgsql.replication_lag.b	以字节为单位的复制延迟。	
pgsql.replication_lag.sec	以秒为单位的复制延迟。	
pgsql.uptime	PostgreSQL 运行时间，单位为毫秒。	
pgsql.version	返回 PostgreSQL 版本。	
pgsql.wal.stat	WAL 统计信息。	
redis.config	获取与模式匹配的 Redis 实例配置参数。	Redis
redis.info	获取 INFO 命令的输出。	
redis.ping	测试连接是否存活。	
redis.slowlog.count	自 Redis 启动以来的慢日志条目数。	
smart.attribute.discovery	返回 S.M.A.R.T. 设备属性列表。	S.M.A.R.T.
smart.disk.discovery	返回 S.M.A.R.T. 设备列表。	
smart.disk.get	返回 S.M.A.R.T. 设备的所有可用属性。	
systemd.unit.get	返回 systemd 单元的所有属性。	Systemd
systemd.unit.info	systemd 单元信息。	
systemd.unit.discovery	systemd 单元及其详细信息列表。	

监控项键值	描述	插件
web.certificate.get	验证证书并返回证书详细信息。	Web certificates

监控项键值详情

不带尖括号的参数为必填参数。
带有尖括号 **< >** 标记的参数为可选参数。

ceph.df.details[connString,<user>,<apikey>]

 集群的数据使用情况以及在各个池之间的分布。
 返回值：*JSON object*。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接到 Ceph 的用户名和 API 密钥。

ceph.osd.stats[connString,<user>,<apikey>]

 聚合统计信息和每个 OSD 的统计信息。
 返回值：*JSON object*。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接 Ceph 的用户名和 API 密钥。

ceph.osd.discovery[connString,<user>,<apikey>]

 已发现的 OSD 列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接 Ceph 的用户名和 API 密钥。

ceph.osd.dump[connString,<user>,<apikey>]

 OSD 的使用率阈值和状态。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接 Ceph 的用户名和 API 密钥。

ceph.ping[connString,<user>,<apikey>]

 测试是否可以建立到 Ceph 的连接。
 返回值：*0* - 连接已中断（如果出现任何错误，包括 AUTH 和配置问题）；*1* - 连接成功。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接到 Ceph 的用户名和 API 密钥。

ceph.pool.discovery[connString,<user>,<apikey>]

 已发现的池列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接 Ceph 的用户名和 API 密钥。

ceph.status[connString,<user>,<apikey>]

 集群的整体状态。
 返回值：*JSON object*。

参数：

- **connString** - URI 或会话名称；
- **user, apikey** - 用于连接 Ceph 的用户名和 API 密钥。

docker.container_info[<ID>,<info>]

 容器的低级别信息。
 返回值：ContainerInspect API 调用的输出，序列化为 JSON。

参数：

- **ID** - 容器的 ID 或名称；
- **info** - 返回的信息量。支持的值：*short*（默认）或 *full*。

必须将 Agent 2 用户（'zabbix'）添加到'docker' group 中，以获得足够的权限。否则，该检查将失败。

docker.container_stats[<ID>]

 容器资源使用统计信息。
 返回值：以 JSON 序列化的 ContainerStats API 调用输出以及 CPU 使用率百分比。

参数：

- **ID** - 容器的 ID 或名称。

必须将 Agent 2 用户（'zabbix'）添加到'docker' group 中，以获得足够的权限。否则检查将失败。

docker.containers

 容器列表。
 返回值：ContainerList API 调用的输出，序列化为 JSON。

必须将 Agent 2 用户（'zabbix'）添加到'docker' group 中，以获得足够的权限。否则，该检查将失败。

docker.containers.discovery[<options>]

 返回容器列表。用于低级别发现。
 返回值：*JSON object*。

参数：

- **options** - 指定应发现所有容器还是仅发现正在运行的容器。支持的值：*true* - 返回所有容器；*false* - 仅返回正在运行的容器（默认）。

Agent 2 用户（'zabbix'）必须添加到'docker' group 中，以获得足够的权限。否则，该检查将失败。

docker.data.usage

 有关当前数据使用情况的信息。
 返回值：以 JSON 序列化的 SystemDataUsage API 调用输出。

必须将 Agent 2 用户（'zabbix'）添加到'docker' group 中，以获得足够的权限。否则，该检查将失败。

docker.images

 返回镜像列表。
 返回值：ImageList API 调用序列化后的 JSON 输出。

必须将 Agent 2 用户（'zabbix'）添加到'docker' group 中，以获得足够的权限。否则检查将失败。

docker.images.discovery

 返回镜像列表。用于低级别发现。
 返回值：*JSON* 对象。

必须将 Agent 2 用户（'zabbix'）添加到'docker' 组中，以获得足够的权限。否则检查将失败。

docker.info

 系统信息。
 返回值：SystemInfo API 调用的输出，序列化为 JSON。

必须将 Agent 2 用户（'zabbix'）添加到'docker' group 中，以获得足够的权限。否则，该检查将失败。

docker.ping

 测试 Docker 守护进程是否存活。
 返回值：*1* - 连接存活；*0* - 连接已断开。

必须将 Agent 2 用户（'zabbix'）添加到'docker' 组 中，以获得足够的权限。否则，该检查将失败。

ember.get[<uri>,<path>]

 返回所需设备的结果。
 返回值：*JSON* 对象。

参数：

- **uri** - Ember+ 设备 URI。
  默认值：127.0.0.1:9998;
- **path** - 设备的 OID 路径。
  默认为空，返回根集合数据。

memcached.ping[connString,<user>,<password>]

测试连接是否存活。

返回值：*1* - 连接存活；*0* - 连接断开（如果出现任何错误，包括 AUTH 和配置问题）。

参数：

- **connString** - URI 或会话名称；

- **user, password** - Memcached 登录凭据。

memcached.stats[connString,<user>,<password>,<type>]

获取 STATS 命令的输出。

返回值：*JSON* - 输出被序列化为 JSON。

参数：

- **connString** - URI 或会话名称；
- **user, password** - Memcached 登录凭据；
- **type** - 要返回的统计类型：*items*、*sizes*、*slabs* 或 *settings*（默认为空，返回一般统计信息）。

mongodb.collection.stats[connString,<user>,<password>,<database>,collection]

返回给定集合的各种存储统计信息。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据；
- **database** - 数据库名称（默认为 admin）；
- **collection** - 集合名称。

mongodb.cfg.discovery[connString,<user>,<password>]

 返回已发现的配置服务器列表。
 返回值：*JSON object*。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.collections.discovery[connString,<user>,<password>]

 返回已发现集合的列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.collections.usage[connString,<user>,<password>]

返回集合的使用统计信息。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.connpool.stats[connString,<user>,<password>]

 返回有关当前数据库实例到分片集群或副本集其他成员的已打开出站连接的信息。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.db.stats[connString,<user>,<password>,<database>]

返回反映给定数据库系统状态的统计信息。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据；
- **database** - 数据库名称（默认为 admin）。

mongodb.db.discovery[connString,<user>,<password>]

 返回已发现数据库的列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.jumbo_chunks.count[connString,<user>,<password>]

返回超大块的计数。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.oplog.stats[connString,<user>,<password>]

使用从 oplog 中获取的数据返回副本集的状态。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.ping[connString,<user>,<password>]

测试连接是否存活。

返回值：*1* - 连接存活；*0* - 连接断开（如果出现任何错误，包括 AUTH 和配置问题）。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.rs.config[connString,<user>,<password>]

返回副本集的当前配置。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.rs.status[connString,<user>,<password>]

从运行方法的成员的角度返回副本集的状态。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.server.status[connString,<user>,<password>]

返回数据库状态。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.sh.discovery[connString,<user>,<password>]

返回集群中存在的已发现 shard 的列表。

返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mongodb.version[connString,<user>,<password>]

返回数据库服务器版本。

返回值：字符串。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MongoDB 登录凭据。

mqtt.get[<broker url>,topic,<user>,<password>]

 订阅所提供的 broker 的特定 topic 或多个 topic（支持通配符），并等待发布消息。
 返回值：取决于 topic 内容。如果使用了通配符，则以 JSON 形式返回 topic 内容。

参数：

- **broker url** - MQTT broker URL，格式为 `protocol://host:port`，不带查询参数（支持的协议：`tcp`、`ssl`、`ws`）。如果未指定值，agent 将使用 `tcp://localhost:1883`。如果省略了协议或端口，则分别使用默认协议（`tcp`）或默认端口（1883）；
- **topic** - MQTT topic（必填）。支持通配符（+、#）；
- **user, password** - 认证凭据（如需要）。

注释：

- 该监控项必须配置为active check（“Zabbix agent (active)” 监控项类型）；
- 可以使用 TLS 加密证书，只需将其保存到默认位置（例如 Ubuntu 的 `/etc/ssl/certs/` 目录）。对于 TLS，请使用 `tls://` 方案。

mssql.availability.group.get[URI,<user>,<password>]

 返回可用性组。
 返回值：*JSON object*。

参数：

- **URI** - MSSQL 服务器 URI（唯一支持的 schema 是 `sqlserver://`）。嵌入的凭据将被忽略。可以将实例名称指定为 URI 的一部分，例如：`sqlserver://localhost/InstanceName`（无端口）。如果指定了端口，则实例名称将被忽略；
- **user, password** - 用于发送到受保护 MSSQL 服务器的用户名、密码。

更多信息请参见 MSSQL plugin 的 readme。

mssql.custom.query[URI,<user>,<password>,queryName,<args...>]

 返回自定义查询的结果。
 返回值：*JSON object*。

参数：

- **URI** - MSSQL 服务器 URI（唯一支持的 schema 是 `sqlserver://`）。嵌入的凭据将被忽略。可以将实例名称指定为 URI 的一部分，例如：`sqlserver://localhost/InstanceName`（无端口）。如果指定了端口，则实例名称将被忽略；
- **user, password** - 要发送到受保护 MSSQL 服务器的用户名、密码；
- **queryName** - 在 `Plugins.MSSQL.CustomQueriesDir` 中配置的自定义查询名称，不包含.sql 扩展名；
- **args** - 传递给查询的一个或多个以逗号分隔的参数。

更多信息请参见 MSSQL plugin readme。

mssql.db.get

返回所有可用的 MSSQL 数据库。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.job.status.get

返回作业的状态。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.last.backup.get

返回所有数据库的最后备份时间。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.local.db.get

返回参与 Always On 可用性组和副本（主要或次要）并位于建立连接的服务器上的数据库。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.mirroring.get

返回镜像信息。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.nonlocal.db.get

返回参与 Always On 可用性组和副本（主要或次要）并位于其他服务器上（数据库不是连接建立到的 SQL Server 实例的本地数据库）的数据库。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.perfcounter.get

返回性能计数器。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.ping

 Ping 数据库。测试连接是否已正确配置。
 返回值：*1* - 存活，*0* - 不存活。

更多信息请参见 MSSQL plugin 的 readme。

mssql.quorum.get

返回仲裁信息。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.quorum.member.get

返回仲裁成员。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.replica.get

返回副本。

返回值：*JSON* 对象。

有关更多信息，请参阅MSSQL 插件的自述文件。

mssql.version

返回 MSSQL 版本。

返回值：字符串。

有关更多信息，请参阅MSSQL 插件的自述文件。

mysql.custom.query[connString,<user>,<password>,queryName,<args...>]

 返回自定义查询的结果。
 返回值：*JSON* 对象。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据；

- **queryName** - 自定义查询的名称，必须与不带扩展名的 SQL 文件名匹配；
- **args** - 传递给查询的一个或多个以逗号分隔的参数。

更多信息请参见 MySQL plugin 的 readme 中的 *Custom queries* 部分。

mysql.db.discovery[connString,<user>,<password>]

 返回 MySQL 数据库列表。
用于低级别发现。
 返回值：”show databases” SQL 查询的结果，采用 LLD JSON 格式。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据。

mysql.db.size[connString,<user>,<password>,<database name>]

数据库大小（字节）。

返回值：针对特定数据库的”select coalesce(sum(data_length + index_length),0) as size from information_schema.tables where table_schema=?” SQL 查询的结果，以字节为单位。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据；
- **database name** - 数据库名称。

mysql.get_status_variables[connString,<user>,<password>]

全局状态变量的值。

返回值：以 JSON 格式呈现的”show global status” SQL 查询的结果。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据。

mysql.ping[connString,<user>,<password>]

测试连接是否存活。

返回值：*1* - 连接存活；*0* - 连接中断（如果出现任何错误，包括 AUTH 和配置问题）。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据。

mysql.replication.discovery[connString,<user>,<password>]

 返回 MySQL 复制列表。用于低级别发现。
 返回值：`SHOW SLAVE STATUS` 或 `SHOW REPLICA STATUS` SQL 查询的结果，采用 LLD JSON 格式。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据。

mysql.replication.get_slave_status[connString,<user>,<password>,<master host>]

 复制状态。
 返回值：`SHOW SLAVE STATUS` 或 `SHOW REPLICA STATUS` SQL 查询的结果，采用 JSON 格式。

参数：

- **connString** - URI 或会话名称；
- **user, password** - MySQL 登录凭据；
- **master host** - 复制主机的主机名。如果未找到，则返回错误。如果未指定此参数，则返回所有主机。

mysql.version[connString,<user>,<password>]

MySQL 版本。

返回值：字符串（包含 MySQL 实例的版本号）。

参数：

- **connString** - URI 或会话名称；

- **user, password** - MySQL 登录凭据。

net.dns.get[<ip>,name,<type>,<timeout>,<count>,<protocol>,"<flags>"]

执行 DNS 查询并返回详细的 DNS 记录信息。
 此监控项是net.dns.record Zabbix agent 监控项的扩展版本，支持更多记录类型和可自定义的标志。
 返回值：*JSON object*

参数：

- **ip** - DNS 服务器的 IP 地址（留空则使用默认 DNS 服务器）；
- **name** - 要查询的 DNS 名称；
- **type** - 要查询的记录类型（默认是 *SOA*）；
- **timeout** - 请求超时时间，单位为秒（默认是 1 秒）；
- **count** - 请求重试次数（默认是 2）；
- **protocol** - 用于执行 DNS 查询的协议：*udp*（默认）或 *tcp*；
- **flags** - 一个或多个以逗号分隔的参数，用于传递给查询。

说明：

- type 的可选值为：*A*, *NS*, *MD*, *MF*, *CNAME*, *SOA*, *MB*, *MG*, *MR*, *NULL*, *PTR*, *HINFO*, *MINFO*, *MX*, *TXT*, *RP*, *AFSDB*, *X25*, *ISDN*, *RT*, *NSAPPTR*, *SIG*, *KEY*, *PX*, *GPOS*, *AAAA*, *LOC*, *NXT*, *EID*, *NIMLOC*, *SRV*, *ATMA*, *NAPTR*, *KX*, *CERT*, *DNAME*, *OPT*, *APL*, *DS*, *SSHFP*, *IPSECKEY*, *RRSIG*, *NSEC*, *DNSKEY*, *DHCID*, *NSEC3*, *NSEC3PARAM*, *TLSA*, *SMIMEA*, *HIP*, *NINFO*, *RKEY*, *TALINK*, *CDS*, *CDNSKEY*, *OPENPGPKEY*, *CSYNC*, *ZONEMD*, *SVCB*, *HTTPS*, *SPF*, *UINFO*, *UID*, *GID*, *UNSPEC*, *NID*, *L32*, *L64*, *LP*, *EUI48*, *EUI64*, *URI*, *CAA*, *AVC*, *AMTRELAY*。请注意，值必须全部使用大写；不支持小写或大小写混合的值。
- 对于反向 DNS 查询（当 type 设置为 *PTR* 时），你可以同时以反向和非反向格式提供 DNS 名称（请参见下面的示例）。请注意，当请求 PTR 记录时，DNS 名称实际上是一个 IP 地址。
- flags 的可选值为：
  - *cdflag* 或 *nocdflag*（默认）；
  - *rdflag*（默认）或 *nordflag*；
  - *dnssec* 或 *nodnssec*（默认）；
  - *nsid* 或 *nonsid*（默认）；
  - *edns0*（默认）或 *noedns0*；
  - *aaflag* 或 *noaaflag*（默认）；
  - *adflag* 或 *noadflag*（默认）。flags 中的 *dnssec* 和 *nsid* 不能与 *noedns0* 同时使用，因为它们都需要 *edns0*。请注意，值必须全部使用小写；不支持大写或大小写混合的值。
- 不支持国际化域名，请改用 IDNA 编码后的名称。
- 输出为一个对象，其中包含基于所提供参数的 DNS 记录信息（请参见更多详情）。

示例：

```
net.dns.get[192.0.2.1,example.com,DNSKEY,3,3,tcp,"cdflag,rdflag,nsid"]

net.dns.get[,198.51.100.1,PTR,,,,"cdflag,rdflag,nsid"]
net.dns.get[,1.100.51.198.in-addr.arpa,PTR,,,,"cdflag,rdflag,nsid"]

net.dns.get[,2a00:1450:400f:800::200e,PTR,,,,"cdflag,rdflag,nsid"]
net.dns.get[,e.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.f.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa,PTR,,,,"cdflag,rdfla
```

nvml.device.count

 GPU 设备的数量。
 返回值：整数。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.decoder.utilization[<deviceUUID>]

 GPU 设备解码器利用率，以百分比表示。
 返回值：*Integer*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.ecc.mode[<deviceUUID>]

 GPU 设备 ECC 模式信息（当前、待定）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.encoder.stats.get[<deviceUUID>]

 GPU 设备编码器统计信息。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.encoder.utilization[<deviceUUID>]

 GPU 设备编码器利用率，以百分比表示。
 返回值：*Integer*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.energy.consumption[<deviceUUID>]

 自驱动程序上次重新加载以来，GPU 设备的总能耗，单位为毫焦耳（mJ）。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.errors.memory[<deviceUUID>]

 GPU 设备 ECC 内存错误信息（已纠正、未纠正）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.errors.register[<deviceUUID>]

 GPU 设备 ECC 寄存器错误信息（已纠正、未纠正）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.fan.speed.avg[<deviceUUID>]

 GPU 设备平均风扇转速，以最大转速的百分比表示。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.get

 发现的 GPU 及其 UUID 和名称。
 返回值：*JSON*。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.graphics.frequency[<deviceUUID>]

 GPU 设备图形时钟频率，单位为 MHz。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.memory.fb.get[<deviceUUID>]

 GPU 设备帧缓冲区内存统计信息（总量、保留、空闲、已使用）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 自述文件。

nvml.device.memory.bar1.get[<deviceUUID>]

 GPU 设备 BAR1 内存统计信息（总量、空闲、已使用）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.memory.frequency[<deviceUUID>]

 GPU 设备显存时钟频率，单位为 MHz。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.pci.utilization[<deviceUUID>]

 GPU 设备 PCI 利用率（发送/接收吞吐量，单位为 KBps）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.performance.state[<deviceUUID>]

 GPU 设备性能状态。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.power.limit[<deviceUUID>]

 GPU 设备功率限制，单位为毫瓦。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.power.usage[<deviceUUID>]

 GPU 设备当前功耗，单位为毫瓦。
 返回值：*Integer*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.serial[<deviceUUID>]

 GPU 设备序列号。
 返回值：*String*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.sm.frequency[<deviceUUID>]

 GPU 设备流式多处理器时钟频率，单位为 MHz。
 返回值：*Integer*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.temperature[<deviceUUID>]

 GPU 设备温度，单位为摄氏度。
 返回值：*Integer*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.utilization[<deviceUUID>]

 GPU 设备利用率统计信息（GPU/内存利用率，以百分比表示）。
 返回值：*JSON*。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.device.video.frequency[<deviceUUID>]

 GPU 设备视频时钟频率，单位为 MHz。
 返回值：整数。

参数：

- **deviceUUID** - GPU 设备 UUID。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.system.driver.version

 NVIDIA 驱动程序版本。
 返回值：*String*。

更多信息请参见 NVIDIA GPU plugin 的 readme。

nvml.version

 NVML 库版本。
 返回值：*String*。

更多信息请参见 NVIDIA GPU plugin 的 readme。

oracle.diskgroups.stats[connString,<user>,<password>,<service>,<diskgroup>]

 返回 Automatic Storage Management (ASM) 磁盘组统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果它包含 schema（例如”tcp://”）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号”(” 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式追加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称；
- **diskgroup** - 要查询的 ASM 磁盘组名称。

oracle.diskgroups.discovery[connString,<user>,<password>,<service>]

 返回 ASM 磁盘组列表。用于低级别发现。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果它包含 schema（例如”tcp://”）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号”(” 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；

- **service** - Oracle 服务名称。

oracle.archive.info[connString,<user>,<password>,<service>,<destination>]

 归档日志统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下任一登录选项：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称；
- **destination** - 要查询的目标名称。

oracle.cdb.info[connString,<user>,<password>,<service>,<database>]

 容器数据库（CDB）信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称；
- **database** - 要查询的数据库名称。

oracle.custom.query[connString,<user>,<password>,<service>,queryName,<args...>]

 自定义查询的结果。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了此名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式追加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称；
- **queryName** - 自定义查询的名称，必须与不带扩展名的 SQL 文件名匹配；
- **args** - 传递给查询的一个或多个以逗号分隔的参数。

注释：- 返回的数据会自动转换为 JSON。- 避免让查询直接返回 JSON，因为当插件再次尝试将其转换为 JSON 时，它会被破坏。

oracle.datafiles.stats[connString,<user>,<password>,<service>]

 返回数据文件统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。

- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，且末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.db.discovery[connString,<user>,<password>,<service>]

 返回数据库列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.fra.stats[connString,<user>,<password>,<service>]

 返回快速恢复区（FRA）统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且仍将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下任一登录选项：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.instance.info[connString,<user>,<password>,<service>]

 实例统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式追加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.pdb.info[connString,<user>,<password>,<service>,<database>]

 可插拔数据库（PDB）信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式追加以下任一登录选项：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；

- **service** - Oracle 服务名；
- **database** - 要查询的数据库名称。

oracle.pdb.discovery[connString,<user>,<password>,<service>]

 返回 PDB 列表。用于低级别发现。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式追加以下任一登录选项：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.pga.stats[connString,<user>,<password>,<service>]

 返回 Program Global Area (PGA) 统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式追加以下任一登录选项：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.ping[connString,<user>,<password>,<service>]

 测试是否可以建立到 Oracle 的连接。
 返回值：*1* - 连接成功；*0* - 连接中断（如果出现任何错误，包括 AUTH 和配置问题）。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式追加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

示例：

```
oracle.ping[tcp://127.0.0.1:1521,ZABBIX_MON,zabbix,xe]
oracle.ping[localhost,ZABBIX_MON,zabbix,xe]
oracle.ping[zbx_tns_example,ZABBIX_MON,zabbix,xe]
oracle.ping["(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=x
```

oracle.proc.stats[connString,<user>,<password>,<service>]

 返回进程统计信息。
 返回值：*JSON* 对象。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果其包含 schema（例如 "tcp://"）、端口（例如 1521）或两者，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；

  - TNS 值 - 如果它以左括号 "(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式追加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.redolog.info[connString,<user>,<password>,<service>]

 控制文件中的日志文件信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了此名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，且末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.sga.stats[connString,<user>,<password>,<service>]

 返回系统全局区（SGA）统计信息。
 返回值：*JSON* 对象。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.sessions.stats[connString,<user>,<password>,<service>,<lockMaxTime>]

 返回会话统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式追加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名；
- **lockMaxTime** - 将会话计为长时间锁定时允许的最大会话锁定持续时间，单位为秒。默认值：600 秒。

oracle.sys.metrics[connString,<user>,<password>,<service>,<duration>]

 返回一组系统指标值。
 返回值：*JSON* 对象。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 `Plugins.Oracle.ResolveTNS` 设置为 false。如果其中包含架构（例如"tcp://"）、端口（例如 1521）或两者，则不会考虑 ResolveTNS 选项，而是始终将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到此名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且以上条件均不适用；
  - TNS 值 - 如果其以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。

- **user** - Oracle 用户名，支持以 as sysdba、as sysoper、as sysasm、as sysbackup、as sysdg、as syskm 或 as sysrac 之一追加登录选项，格式为 user as sysdba（登录选项不区分大小写，且末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称；
- **duration** - 系统指标值的采集间隔（以秒为单位）。可选值：*60* - 长间隔（默认值），*15* - 短间隔。

oracle.sys.params[connString,<user>,<password>,<service>]

 返回一组系统参数值。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 user as sysdba 格式追加以下登录选项之一：as sysdba、as sysoper、as sysasm、as sysbackup、as sysdg、as syskm 或 as sysrac（登录选项不区分大小写，且末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.ts.stats[connString,<user>,<password>,<service>,<tablespace>,<type>,<conname>]

 返回表空间统计信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 user as sysdba 格式追加以下登录选项之一：as sysdba、as sysoper、as sysasm、as sysbackup、as sysdg、as syskm 或 as sysrac（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名；
- **tablespace** - 要查询的表空间名称。默认值（如果留空且设置了 type）：
  - "TEMP"（如果 type 设置为"TEMPORARY"）；
  - "USERS"（如果 type 设置为"PERMANENT"）。
- **type** - 要查询的表空间类型。默认值（如果设置了 tablespace）："PERMANENT"。
- **conname** - 需要其信息的容器名称。

如果省略 tablespace、type 或 conname，则该监控项将返回所有匹配容器（包括 PDB 和 CDB）的表空间统计信息。

oracle.ts.discovery[connString,<user>,<password>,<service>]

 返回表空间列表。用于低级别发现。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：
  - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
  - 会话名称 - 如果在插件配置文件中找到了该名称；
  - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
  - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 user as sysdba 格式附加以下登录选项之一：as sysdba、as sysoper、as sysasm、as sysbackup、as sysdg、as syskm 或 as sysrac（登录选项不区分大小写，且末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

oracle.user.info[connString,<user>,<password>,<service>,<username>]

 返回 Oracle 用户信息。
 返回值：*JSON object*。

参数：

- **connString** - 可以是：

    - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
    - 会话名称 - 如果在插件配置文件中找到了该名称；
    - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
    - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不会考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持以 `user as sysdba` 格式追加以下任一登录选项：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名；
- **username** - 需要获取其信息的用户名。不支持小写用户名。默认值：当前用户。

oracle.version[connString,<user>,<password>,<service>]

 返回数据库服务器版本。
 返回值：*String*。

参数：

- **connString** - 可以是：
    - URI - 如果未找到具有给定名称的会话，且 Plugins.Oracle.ResolveTNS 设置为 false。如果它包含 schema（例如"tcp://"）、端口（例如 1521）或两者都包含，则不会考虑 ResolveTNS 选项，并且无论如何都会将其视为 URI；
    - 会话名称 - 如果在插件配置文件中找到了此名称；
    - TNS 键 - 如果 ResolveTNS 选项设置为 true，且上述条件均不适用；
    - TNS 值 - 如果它以左括号"(" 开头（忽略前导空格）。在这种情况下，不考虑 ResolveTNS 选项。
- **user** - Oracle 用户名，支持按 `user as sysdba` 格式附加以下登录选项之一：`as sysdba`、`as sysoper`、`as sysasm`、`as sysbackup`、`as sysdg`、`as syskm` 或 `as sysrac`（登录选项不区分大小写，且末尾不得包含空格）。
- **password** - Oracle 密码；
- **service** - Oracle 服务名称。

pgsql.autovacuum.count[uri,<username>,<password>,<database name>]

 autovacuum worker 的数量。
 返回值：*Integer*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.archive[uri,<username>,<password>,<database name>]

 归档文件的信息。
 返回值：*JSON object*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.bgwriter[uri,<username>,<password>,<database name>]

 数据库集群的检查点总数，按检查点类型细分。
 返回值：*JSON* 对象。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.cache.hit[uri,<username>,<password>,<database name>]

 PostgreSQL 缓冲区缓存命中率。
 返回值：*Float*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.connections[uri,<username>,<password>,<database name>]

 按类型返回连接。
 返回值：*JSON* 对象。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.custom.query[uri,<username>,<password>,<database name>,queryName,<args...>]

 返回自定义查询的结果。
 返回值：*JSON object*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）；
- **queryName** - 自定义查询的名称，必须与不带扩展名的 SQL 文件名匹配；
- **args** - 一个或多个以逗号分隔的参数，用于传递给查询。

pgsql.db.age[uri,<username>,<password>,<database name>]

 数据库中最老的 FrozenXID 的年龄（事务数）。
 返回值：*Integer*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.db.bloating_tables[uri,<username>,<password>,<database name>]

 每个数据库中的膨胀表数量。
 返回值：*Integer*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.db.discovery[uri,<username>,<password>,<database name>]

 PostgreSQL 数据库列表。用于低级别发现。
 返回值：*JSON object*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.db.size[uri,<username>,<password>,<database name>]

 数据库大小，单位为字节。
 返回值：*Integer*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.dbstat[uri,<username>,<password>,<database name>]

 按数据库收集统计信息。用于低级别发现。
 返回值：*JSON object*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；

- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.dbstat.sum[uri,<username>,<password>,<database name>]

 集群中所有数据库的汇总数据。
 返回值：*JSON* 对象。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.locks[uri,<username>,<password>,<database name>]

 每个数据库中已授予锁的信息。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.oldest.xid[uri,<username>,<password>,<database name>]

 最老 XID 的年龄（事务数量）。
 返回值：*Integer*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.ping[uri,<username>,<password>,<database name>]

 测试连接是否存活。
 返回值：*1* - 连接存活；*0* - 连接已断开（如果出现任何错误，包括 AUTH 和配置问题）。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）。

pgsql.queries[uri,<username>,<password>,<database name>,time period]

 按执行时间查询指标。
 返回值：*JSON object*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 postgresql.conf 中Plugins.PostgreSQL.Default.Database 的值）；
- **time period** - 统计慢查询数量的执行时间限制（必须为正整数）。

pgsql.replication.count[uri,<username>,<password>]

 备库服务器的数量。
 返回值：整数。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.replication.process[uri,<username>,<password>]

 每个发送进程的刷新延迟、写入延迟和重放延迟。
 返回值：*JSON* 对象。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.replication.process.discovery[uri,<username>,<password>]

 复制进程名称发现。
 返回值：*JSON* 对象。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.replication.recovery_role[uri,<username>,<password>]

 恢复状态。
 返回值：*0* - 主节点模式；*1* - 正在进行恢复（备库模式）。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.replication.status[uri,<username>,<password>]

 复制状态。
 返回值：*0* - 流复制已停止；*1* - 流复制正常工作；*2* - 主节点模式。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.replication_lag.b[uri,<username>,<password>]

 复制延迟（以字节为单位）。
 返回值：整数。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.replication_lag.sec[uri,<username>,<password>]

 复制延迟（以秒为单位）。
 返回值：整数。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据。

pgsql.uptime[uri,<username>,<password>,<database name>]

 PostgreSQL 的运行时间，单位为毫秒。
 返回值：*Float*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.version[uri,<username>,<password>,<database name>]

 返回 PostgreSQL 版本。
 返回值：*String*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

pgsql.wal.stat[uri,<username>,<password>,<database name>]

 WAL 统计信息。
 返回值：*JSON object*。

参数：

- **uri** - URI 或会话名称；
- **username, password** - PostgreSQL 凭据；
- **database name** - 数据库名称（如果省略，则使用 `postgresql.conf` 中`Plugins.PostgreSQL.Default.Database` 的值）。

redis.config[connString,<password>,<pattern>]

 获取与模式匹配的 Redis 实例配置参数。
 返回值：*JSON* - 如果使用了 glob 风格模式；单个值 - 如果模式中不包含任何通配符字符。

参数：

- **connString** - URI 或会话名称；
- **password** - Redis 密码；
- **pattern** - glob 风格模式（默认为 *）。

注释：

- 安全警告：`redis.config` 会运行 Redis 的 `CONFIG GET` 命令，并且可能返回敏感配置参数（例如 `requirepass`、`masterauth`、与 TLS 相关的设置以及其他密钥信息）。使用默认模式（`*`）或宽泛模式，可能会将这些密钥信息暴露给任何有权在 Zabbix 中查看监控项值的用户。

redis.info[connString,<password>,<section>]

 获取 INFO 命令的输出。
 返回值：*JSON* - 输出序列化为 JSON 格式。

参数：

- **connString** - URI 或会话名称；
- **password** - Redis 密码；
- **section** - 信息的部分（默认为 *default*）。详细参考 INFO 命令。

redis.ping[connString,<password>]

 测试连接是否存活。
 返回值：*1* - 连接存活；*0* - 连接中断（如果出现任何错误，包括认证和配置问题）。

参数：

- **connString** - URI 或会话名称；
- **password** - Redis 密码。

redis.slowlog.count[connString,<password>]

 自 Redis 启动以来的慢日志条目数量。
 返回值：整数。

参数：

- **connString** - URI 或会话名称；
- **password** - Redis 密码。

smart.attribute.discovery

 返回 S.M.A.R.T. 设备属性列表。
 返回值：*JSON object*。

注释：

- 返回以下宏及其值：`{#NAME}`、`{#DISKTYPE}`、`{#ID}`、`{#ATTRNAME}`、`{#THRESH}`；
- 支持 HDD、SSD 和 NVME 驱动器类型。驱动器可以单独使用，也可以组合成 RAID。在 RAID 情况下，`{#NAME}` 会附加额外信息，例如：`{"{#NAME}": "/dev/sda cciss,2"}`。

smart.disk.discovery[<type>]

 返回 S.M.A.R.T. 设备列表。
 返回值：*JSON object*。

参数：

- **type** - 指定要扫描磁盘的值。可选值：*id*、*name*（默认）。在 Windows 上不支持 *id*，如果选择该值将返回错误

注释：

- 返回以下宏及其值：`{#NAME}`、`{#DISKTYPE}`、`{#MODEL}`、`{#SN}`、`{#PATH}`、`{#ATTRIBUTES}`、`{#RAIDTYPE}`；
- 支持 HDD、SSD 和 NVME 驱动器类型。如果驱动器不属于 RAID，则 `{#RAIDTYPE}` 为空。在 RAID 情况下，`{#NAME}` 会带有附加信息，例如：`{"{#NAME}": "/dev/sda cciss,2"}`。

smart.disk.get[<path>,<raid type>]

 返回 S.M.A.R.T. 设备的所有可用属性。
 返回值：*JSON object*。

参数：

- **path** - 磁盘路径，可使用 `{#PATH}` 宏作为值；
- **raid_type** - RAID 类型，可使用 `{#RAID}` 宏作为值

注释：

- 支持 HDD、SSD 和 NVME 驱动器类型。驱动器可以单独存在，也可以组合在 RAID 中；
- 数据包括 smartctl 版本和调用参数，以及附加字段：
disk_name - 保存用于 RAID 发现所需附加信息的名称，例如：`{"disk_name": "/dev/sda cciss,2"}`
disk_type - 保存磁盘类型 HDD、SSD 或 NVME，例如：`{"disk_type": "ssd"}`；
- 如果未指定参数，监控项将返回所有磁盘的信息。

systemd.unit.get[unit name,<interface>]

 返回 systemd 单元的所有属性。
 返回值：*JSON object*。

参数：

- **unit name** - 单元名称（您可能希望在监控项原型中使用 `{#UNIT.NAME}` 宏来发现名称）；
- **interface** - 单元接口类型，可选值：*Unit*（默认）、*Service*、*Socket*、*Device*、*Mount*、*Automount*、*Swap*、*Target*、*Path*。

注释：

- 此监控项仅支持 Linux 平台；
- 对于 Unit 接口，LoadState、ActiveState 和 UnitFileState 以文本和整数形式返回：`"ActiveState":{"state":1,"text":"active"}`。

systemd.unit.info[unit name,<property>,<interface>]

 systemd 单元信息。
 返回值：*String*。

参数：

- **unit name** - 单元名称（您可能需要在监控项原型中使用 `{#UNIT.NAME}` 宏来发现名称）；
- **property** - 单元属性（例如，ActiveState（默认）、LoadState、Description）；
- **interface** - 单元接口类型（例如，Unit（默认）、Socket、Service）。

注释：

- 此监控项仅支持 Linux 平台；
- 此监控项允许您从 dbus API 中所述的特定类型接口检索特定属性。

示例：

```
systemd.unit.info["{#UNIT.NAME}"] # 收集已发现 systemd 单元的活动状态（active、reloading、inactive、failed、acti
systemd.unit.info["{#UNIT.NAME}",LoadState] # 收集已发现 systemd 单元的加载状态信息
systemd.unit.info[mysqld.service,Id] # 检索服务技术名称（mysqld.service）
systemd.unit.info[mysqld.service,Description] # 检索服务描述（MySQL Server）
systemd.unit.info[mysqld.service,ActiveEnterTimestamp] # 检索服务进入活动状态的最近时间（1562565036283903）
systemd.unit.info[dbus.socket,NConnections,Socket] # 收集此套接字单元的连接数
```

systemd.unit.discovery[<type>]

 systemd 单元及其详细信息列表。用于低级别发现。
 返回值：*JSON object*。

参数：

- **type** - 可能的值：*all*、*automount*、*device*、*mount*、*path*、*service*（默认）、*socket*、*swap*、*target*。

此监控项仅在 Linux 平台上受支持。

web.certificate.get[hostname,<port>,<address>]

 验证证书并返回证书详细信息。
 返回值：*JSON object*。

参数：

- **hostname** - 可以是 IP 或 DNS。
 可以包含 URL scheme（仅 *https*）、路径（将被忽略）和端口。
 如果第一个参数和第二个参数都提供了端口，则它们的值必须一致。
 如果指定了 address（第 3 个参数），则 hostname 仅用于 SNI 和主机名验证；
- **port** - 端口号（HTTPS 的默认值为 443）；
- **address** - 可以是 IP 或 DNS。如果指定，它将用于连接，而 hostname（第 1 个参数）将用于 SNI 和主机验证。如果第 1 个参数是 IP 而第 3 个参数是 DNS，则第 1 个参数将用于连接，而第 3 个参数将用于 SNI 和主机验证。

注释：

- 如果主机配置中指定的目标不存在、不可用，或者 TLS 握手因除无效证书之外的任何错误而失败，则此监控项将变为不受支持；
- 目前不支持 AIA（Authority Information Access）X.509 扩展、CRL 和 OCSP（包括 OCSP stapling）以及 Certificate Transparency；
- JSON 响应字段：
  - *x509*：包含 X.509 证书的详细信息。
    - *version*：X.509 版本（例如 3）。

        * *serial_number*：证书序列号。
        * *signature_algorithm*：用于签署证书的算法（例如 SHA256-RSA）。
        * *issuer*：证书颁发者。
        * *not_before*：证书有效期开始日期。
        * *not_after*：证书过期日期。
        * *subject*：证书主体。
        * *public_key_algorithm*：公钥使用的算法（例如 RSA）。
        * *alternative_names*：主题备用名称（如果存在），否则为 null。
      - result：包含验证结果。
        * *value*：验证状态（请参见下方可能的值）。
        * *message*：详细的验证消息（例如 "certificate verified successfully"）。
      - *sha1_fingerprint*：证书的 SHA-1 指纹。
      - *sha256_fingerprint*：证书的 SHA-256 指纹。
  - $.result.value 字段表示证书验证结果。可能的值包括：
      - *valid* - 证书有效且受信任。
      - *valid-but-self-signed* - 证书有效但为自签名，即其主体与颁发者相同。
      - *invalid* - 证书无效，原因可能包括过期、主机名不正确或签名机构未知。

示例：

```
web.certificate.get[example.com,443]
```

JSON Response:


```
{
  "x509": {
    "version": 3,
    "serial_number": "0ad893bafa68b0b7fb7a404f06ecaf9a",
    "signature_algorithm": "ECDSA-SHA384",
    "issuer": "CN=DigiCert Global G3 TLS ECC SHA384 2020 CA1,O=DigiCert Inc,C=US",
    "not_before": {
      "value": "Jan 15 00:00:00 2025 GMT",
      "timestamp": 1736899200
    },
    "not_after": {
      "value": "Jan 15 23:59:59 2026 GMT",
      "timestamp": 1768521599
    },
    "subject": "CN=*.example.com,O=Internet Corporation for Assigned Names and Numbers,L=Los Angeles,ST=Ca
    "public_key_algorithm": "ECDSA",
    "alternative_names": [
      "*.example.com",
      "example.com"
    ]
  },
  "result": {
    "value": "valid",
    "message": "certificate verified successfully"
  },
  "sha1_fingerprint": "310db7af4b2bc9040c8344701aca08d0c69381e3",
  "sha256_fingerprint": "455943cf819425761d1f950263ebf54755d8d684c25535943976f488bc79d23b"
}
```


Windows Zabbix agent

概述

Windows Zabbix agent 的监控项分为两个列表：

- 共享监控项 - 与 UNIX Zabbix agent 共享的监控项键值；
- Windows 特有监控项 - 仅在 Windows 上受支持的监控项键值。

请注意，Windows 上的 Zabbix agent 支持的所有监控项键值，同样也受新一代 Zabbix agent 2 支持。另请参见附加监控项键值，这些键值仅可用于 agent 2。

另请参见：Windows 监控项的最低权限要求

共享监控项

下表列出了在 Windows 上受支持、并且与 UNIX Zabbix agent 共享的 Zabbix agent 监控项：

- 监控项键链接到 UNIX Zabbix agent 监控项的完整详细信息
- 包含与 Windows 相关的监控项注释

Item key	Description	Item group
log	日志文件监控。此监控项不支持 Windows 事件日志。 `persistent_dir` 参数在 Windows 上不受支持。	日志监控
log.count	监控日志文件中匹配行的数量。此监控项不支持 Windows 事件日志。 `persistent_dir` 参数在 Windows 上不受支持。	
logrt	轮转日志文件监控。此监控项不支持 Windows 事件日志。 `persistent_dir` 参数在 Windows 上不受支持。	
logrt.count	监控轮转日志文件中匹配行的数量。此监控项不支持 Windows 事件日志。 `persistent_dir` 参数在 Windows 上不受支持。	
modbus.get	读取 Modbus 数据。	Modbus
net.dns	检查 DNS 服务是否运行。 在 Windows 上，除非使用 Zabbix agent 2，否则 `ip`、`timeout` 和 `count` 参数会被忽略。	Network
net.dns.perf	检查 DNS 服务的性能。 在 Windows 上，除非使用 Zabbix agent 2，否则 `ip`、`timeout` 和 `count` 参数会被忽略。	
net.dns.record	执行 DNS 查询。 在 Windows 上，除非使用 Zabbix agent 2，否则 `ip`、`timeout` 和 `count` 参数会被忽略。	
net.if.discovery	网络接口列表。 某些 Windows 版本（例如 Server 2008）可能需要安装最新更新，才能支持接口名称中的非 ASCII 字符。	
net.if.in	网络接口上的入站流量统计。 在 Windows 上，如果可用，监控项会从 64 位计数器获取值。64 位接口统计计数器是在 Windows Vista 和 Windows Server 2008 中引入的。如果没有可用的 64 位计数器，agent 将使用 32 位计数器。 Windows 支持多字节接口名称。 您可以使用 net.if.discovery 或 net.if.list 监控项获取 Windows 上的网络接口描述。	
net.if.out	网络接口上的出站流量统计。 在 Windows 上，如果可用，监控项会从 64 位计数器获取值。64 位接口统计计数器是在 Windows Vista 和 Windows Server 2008 中引入的。如果没有可用的 64 位计数器，agent 将使用 32 位计数器。 Windows 支持多字节接口名称。 您可以使用 net.if.discovery 或 net.if.list 监控项获取 Windows 上的网络接口描述。	
net.if.total	网络接口上的入站和出站流量统计总和。 在 Windows 上，如果可用，监控项会从 64 位计数器获取值。64 位接口统计计数器是在 Windows Vista 和 Windows Server 2008 中引入的。如果没有可用的 64 位计数器，agent 将使用 32 位计数器。 Windows 支持多字节接口名称。 您可以使用 net.if.discovery 或 net.if.list 监控项获取 Windows 上的网络接口描述。	
net.tcp.listen	检查此 TCP 端口是否处于 LISTEN 状态。	
net.tcp.port	检查是否可以与指定端口建立 TCP 连接。	
net.tcp.service	检查服务是否正在运行并接受 TCP 连接。 在 Windows 上，仅 Zabbix agent 2 支持检查 LDAP 和 HTTPS。	
net.tcp.service.perf	检查 TCP 服务的性能。 在 Windows 上，仅 Zabbix agent 2 支持检查 LDAP 和 HTTPS。	
net.tcp.socket.count	返回与参数匹配的 TCP 套接字数量。 此监控项在 Linux 上由 Zabbix agent 支持，但在 Windows 上仅由 64 位 Windows 上的Zabbix agent 2 支持。	
net.udp.service	检查服务是否正在运行并响应 UDP 请求。	
net.udp.service.perf	检查 UDP 服务的性能。	
net.udp.socket.count	返回与参数匹配的 UDP 套接字数量。 此监控项在 Linux 上由 Zabbix agent 支持，但在 Windows 上仅由 64 位 Windows 上的Zabbix agent 2 支持。	

Item key	Description	Item group
proc.get	操作系统进程及其参数列表。 cmdline 参数在 Windows 上不受支持。	Processes
proc.num	进程数量。 在 Windows 上，仅支持 name 和 user 参数。	
system.cpu.discovery	检测到的 CPU/CPU 核心列表。	System
system.cpu.load	CPU 负载。 当在 Zabbix agent 上启动收集器进程时，会初始化以下性能计数器，并在之后用于此监控项：\System\Processor Queue Length	
system.cpu.num	CPU 数量。	
system.cpu.util	CPU 利用率百分比。 该值通过 *Processor Time* 性能计数器获取。请注意，自 Windows 8 起，其任务管理器基于 *Processor Utility* 性能计数器显示 CPU 利用率，而在更早版本中使用的是 *Processor Time* 计数器（参见更多详情）。 在 Windows 上，仅支持 system 作为 type 参数。	
system.hostname	系统主机名。 在 Windows 上，该值通过 GetComputerName()（用于 **netbios**）、GetComputerNameExA()（用于 **fqdn**）或 gethostname()（用于 **host**）函数获取。另请参见更详细的说明。	
system.localtime	系统时间。	
system.run	在主机上运行指定命令。	
system.sw.arch	软件架构信息。	
system.sw.os	操作系统信息。	
system.sw.os.get	操作系统的详细信息（版本、类型、发行版名称、次版本号和主版本号等）。	
system.swap.size	交换空间大小，单位为字节，或占总量的百分比。 pused 类型参数在 Linux 上由 Zabbix agent 支持，但在 Windows 上仅由Zabbix agent 2 支持。 请注意，在虚拟化（VMware ESXi、VirtualBox）Windows 平台上，此键可能报告不正确的交换空间大小/百分比。在这种情况下，您可以使用 perf_counter[\700(_Total)\702] 键获取正确的交换空间百分比。	
system.uname	系统标识。 在 Windows 上，此监控项的值通过 Win32_OperatingSystem 和 Win32_Processor WMI 类获取。操作系统名称（包括版本）可能会翻译为用户的显示语言。在某些 Windows 版本中，它包含商标符号和额外空格。	
system.uptime	系统运行时间，单位为秒。	
vfs.dir.count	目录条目数量。 在 Windows 上，会跳过目录符号链接，硬链接只计数一次。	Virtual file systems
vfs.dir.get	目录条目列表。 在 Windows 上，会跳过目录符号链接，硬链接只计数一次。	
vfs.dir.size	目录大小。 在 Windows 上会跳过任何符号链接，硬链接只计入一次。	
vfs.file.cksum	文件校验和，由 UNIX cksum 算法计算。	
vfs.file.contents	获取文件内容。	
vfs.file.exists	检查文件是否存在。 在 Windows 上，使用命令行工具调用 zabbix_get.exe 或 agent2 时，双引号必须用反斜杠 \\ 转义，并且整个监控项键必须用双引号括起来。 请注意，如果在不存在的目录中搜索目录，例如 vfs.file.exists[C:\no\dir,dir]（其中'no' 不存在），该监控项在 Windows 上可能会变为不受支持。	
vfs.file.get	返回有关文件的信息。 Windows 上支持的文件类型：普通文件、目录、符号链接	
vfs.file.md5sum	文件的 MD5 校验和。	
vfs.file.owner	获取文件所有者。	
vfs.file.regexp	从文件中提取字符串。	
vfs.file.regmatch	在文件中查找字符串。	
vfs.file.size	文件大小。	
vfs.file.time	文件时间信息。 在 Windows XP 中，vfs.file.time[file,change] 可能等于 vfs.file.time[file,access]。	
vfs.fs.discovery	已挂载文件系统及其类型和挂载选项列表。 Windows 上支持 {#FSLABEL} 宏。	
vfs.fs.get	已挂载文件系统及其类型、可用磁盘空间、inode 统计信息和挂载选项列表。 Windows 上支持 {#FSLABEL} 宏。	

Item key	Description	Item group
vfs.fs.size	磁盘空间，单位为字节，或占总量的百分比。	
vm.memory.size	内存大小，单位为字节，或占总量的百分比。	Virtual memory
web.page.get	获取网页内容。	Web monitoring
web.page.perf	完整网页的加载时间。	
web.page.regexp	在网页中查找字符串。	
agent.hostmetadata	agent 主机元数据。	Zabbix
agent.hostname	agent 主机名。	
agent.ping	agent 可用性检查。	
agent.variant	Zabbix agent 的变体（Zabbix agent 或 Zabbix agent 2）。	
agent.version	Zabbix agent 版本。	
zabbix.stats	远程返回一组 Zabbix 服务器或 proxy 内部指标。	
zabbix.stats (queue)	远程返回 Zabbix 服务器或 proxy 上延迟的监控项队列数量。	

仅适用于 Windows 的监控项

该表提供了仅适用于 Windows Zabbix agent 的监控项键的详细信息。

仅适用于 Windows 的监控项有时是类似代理监控项的近似对应物，例如，在 Windows 上支持的 `proc_info` 大致对应于在 Windows 上不支持的 `proc.mem` 监控项。

监控项键是到完整监控项键详细信息的链接。

键值	描述	监控项组
eventlog	Windows 事件日志监视。	日志监视
eventlog.count	Windows 事件日志中行数的计数。	
net.if.list	网络接口列表（包括接口类型、状态、IPv4 地址、描述）。	网络
perf_counter	任何 Windows 性能计数器的值。	性能计数器
perf_counter_en	任何 Windows 性能计数器的值（英文）。	
perf_instance.discovery	Windows 性能计数器的对象实例列表。	
perf_instance_en.discovery	使用英文对象名称发现的 Windows 性能计数器的对象实例列表。	
proc_info	有关特定进程的各种信息。	进程
registry.data	返回 Windows 注册表键中指定值名称的数据。	注册表
registry.get	位于给定键处的 Windows 注册表值或键的列表。	
service.discovery	Windows 服务列表。	服务
service.info	有关服务的信息。	
services	服务列表。	
vm.vmemory.size	虚拟内存大小（以字节或占总内存的百分比）。	虚拟内存
wmi.get	执行 WMI 查询并返回第一个选定的对象。	WMI
wmi.getall	执行 WMI 查询并返回整个响应。	

监控项键值详情

不带尖括号的参数为必填参数。
带有尖括号 **< >** 标记的参数为可选参数。

eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]

 事件日志监控。
 返回值：*Log*。

参数：

- **name** - 事件日志通道的名称（Event Viewer GUI 中的 *Log Name*）；
- **regexp** - 描述所需模式的正则表达式（区分大小写）；
- **severity** - 描述严重级别的正则表达式（不区分大小写）。此参数接受基于以下值的正则表达式：”Information”、”Warning”、”Error”、”Critical”、”Verbose”（运行于 Windows Vista 或更新版本）。
- **source** - 描述来源标识符的正则表达式（不区分大小写）；
- **eventid** - 描述事件标识符的正则表达式（区分大小写）；
- **maxlines** - agent 每秒发送到 Zabbix 服务器或 proxy 的最大新增行数。此参数会覆盖`zabbix_agentd.conf` 中 `MaxLinesPerSecond` 的值。
- **mode** - 可选值：
  - *all*（默认）；

- *skip* - 跳过较旧数据的处理（仅影响新创建的监控项）。

注释：

- 该监控项必须配置为主动检查；
- agent 无法发送来自"Forwarded events" 日志中的事件；
- 支持 Windows Eventing 6.0；
- 对此监控项选择非 Log 信息类型 将导致本地时间戳以及日志严重级别和来源信息丢失；
- 另请参阅关于日志监控的补充信息。

示例：

```
eventlog[Application]
eventlog[Microsoft-Windows-Application-Experience/Program-Compatibility-Assistant]
eventlog[Security,,"Failure Audit",,^(529|680)$]
eventlog[System,,"Warning|Error"]
eventlog[System,,,,^1$]
eventlog[Windows PowerShell,,,,,,skip]
eventlog[System,,,,@TWOSHORT] #here a custom regular expression named `TWOSHORT` is referenced (defined as
```

eventlog.count[name,<regexp>,<severity>,<source>,<eventid>,<maxproclines>,<mode>]

 Windows 事件日志中的行数。
 返回值：*Integer*。

参数：

- **name** - 事件日志通道的名称（Event Viewer GUI 中的 *Log Name*）；
- **regexp** - 描述所需模式的正则表达式（区分大小写）；
- **severity** - 描述严重级别的正则表达式（不区分大小写）。此参数接受基于以下值的正则表达式："Information"、"Warning"、"Error"、"Critical"、"Verbose"（运行于 Windows Vista 或更新版本）。
- **source** - 描述来源标识符的正则表达式（不区分大小写）；
- **eventid** - 描述事件标识符的正则表达式（区分大小写）；
- **maxproclines** - agent 每秒将分析的最大新行数(不能超过 10000)。默认值为zabbix_agentd.conf 中 MaxLinesPerSecond 的 10 倍。
- **mode** - 可能的值：
  - *all*（默认）；
  - *skip* - 跳过较旧数据的处理（仅影响新创建的监控项）。

注释：

- 该监控项必须配置为主动检查；
- agent 无法发送来自"Forwarded events" 日志的事件；
- 支持 Windows Eventing 6.0；
- 对此监控项选择非 Log 的信息类型会导致本地时间戳以及日志严重级别和来源信息丢失；
- 另请参阅关于日志监控的补充信息。

示例：

```
eventlog.count[System,,"Warning|Error"]
eventlog.count[Windows PowerShell,,,,,,skip]
```

net.if.list

网络接口列表（包括接口类型、状态、IPv4 地址、描述）。
 返回值：文本。

注释：

- 支持多字节接口名称；
- 禁用的接口不会列出；
- 启用/禁用某些组件可能会更改它们在 Windows 接口名称中的顺序；
- 某些 Windows 版本（例如，Server 2008）可能需要安装最新更新才能支持接口名称中的非 ASCII 字符。

perf_counter[counter,<interval>]

 任何 Windows 性能计数器的值。
 返回值：*Integer*、*float*、*string* 或 *text*（取决于请求）。

参数：

- **counter** - 计数器的路径；
- **interval** - 用于保存平均值的最近 N 秒。`interval` 必须介于 1 到 900 秒之间（含），默认值为 1。

注释：

- 此监控项会跟随 UNC 路径，这可能带来安全风险；
- `interval` 用于需要多个样本的计数器（例如 CPU 利用率），因此每次检查都会返回最近"interval" 秒的平均值；
- 可使用 Performance Monitor 获取可用计数器列表。
- 另请参见：Windows performance counters。

perf_counter_en[counter,<interval>]

 任何 Windows 性能计数器的英文值。
 返回值：*Integer*、*float*、*string* 或 *text*（取决于请求）。

参数：

- **counter** - 英文计数器的路径；
- **interval** - 用于保存平均值的最近 N 秒。`interval` 必须介于 1 到 900 秒之间（含），默认值为 1。

注释：

- 此监控项会遵循 UNC 路径，这可能带来安全风险；
- `interval` 用于需要多个样本的计数器（例如 CPU 利用率），因此每次检查都会返回最近"interval" 秒的平均值；
- 此监控项仅支持 **Windows Server 2008/Vista** 及以上版本；
- 你可以通过查看以下注册表项找到英文字符串列表：`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\`

perf_instance.discovery[object]

 Windows 性能计数器的对象实例列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **object** - 对象名称（本地化）。

perf_instance_en.discovery[object]

 使用英文对象名称发现的 Windows 性能计数器对象实例列表。用于低级别发现。
 返回值：*JSON* 对象。

参数：

- **object** - 对象名称（英文）。

proc_info[process,<attribute>,<type>]

 关于特定进程的各种信息。
 返回值：*Float*。

参数：

- **process** - 进程名称；
- **attribute** - 请求的进程属性；
- **type** - 表示类型（当存在多个同名进程时有意义）

注释：

- 支持以下 `attributes`：
  - *vmsize*（默认）- 进程虚拟内存大小，单位为 Kbytes；
  - *wkset* - 进程工作集大小（进程使用的物理内存量），单位为 Kbytes；
  - *pf* - 页面错误次数；
  - *ktime* - 进程内核时间，单位为毫秒；
  - *utime* - 进程用户时间，单位为毫秒；
  - *io_read_b* - 进程在 I/O 操作期间读取的字节数；
  - *io_read_op* - 进程执行的读取操作次数；
  - *io_write_b* - 进程在 I/O 操作期间写入的字节数；
  - *io_write_op* - 进程执行的写入操作次数；
  - *io_other_b* - 进程在除读写操作之外的其他操作期间传输的字节数；
  - *io_other_op* - 进程执行的 I/O 操作次数，不包括读写操作；
  - *gdiobj* - 进程使用的 GDI 对象数量；
  - *userobj* - 进程使用的 USER 对象数量。
- 有效的 `types` 为：
  - *avg*（默认）- 名称为 <process> 的所有进程的平均值；
  - *min* - 名称为 <process> 的所有进程中的最小值；
  - *max* - 名称为 <process> 的所有进程中的最大值；
  - *sum* - 名称为 <process> 的所有进程值之和。
- 在 64 位系统上，此监控项要正常工作需要 64 位 Zabbix agent。

示例：

```
proc_info[iexplore.exe,wkset,sum] # 检索所有 Internet Explorer 进程占用的物理内存量
proc_info[iexplore.exe,pf,avg] # 检索 Internet Explorer 进程的平均页面错误次数
```

registry.data[key,<value name>]

 返回 Windows 注册表键中指定值名称的数据。
 返回值：*Integer*、*string* 或 *text*（取决于值类型）

参数：

- **key** - 包含根键的注册表键；允许使用根键缩写（例如 HKLM）；
- **value name** - 键中的注册表值名称（默认为空字符串""）。如果未提供值名称，则返回默认值。

注释：

- 支持的根键缩写：
  - HKCR - HKEY_CLASSES_ROOT
  - HKCC - HKEY_CURRENT_CONFIG
  - HKCU - HKEY_CURRENT_USER
  - HKCULS - HKEY_CURRENT_USER_LOCAL_SETTINGS
  - HKLM - HKEY_LOCAL_MACHINE
  - HKPD - HKEY_PERFORMANCE_DATA
  - HKPN - HKEY_PERFORMANCE_NLSTEXT
  - HKPT - HKEY_PERFORMANCE_TEXT
  - HKU - HKEY_USERS
- 包含空格的键必须用双引号括起来。

示例：

```
registry.data["HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting"] # 返回此键默认值的数据
registry.data["HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting","EnableZip"] # 返回此键中名为 "Enable
```

registry.get[key,<mode>,<name regexp>]

 位于给定键下的 Windows 注册表值或键列表。
 返回值：*JSON* 对象。

参数：

- **key** - 包括根键在内的注册表键；允许使用根键缩写（例如 HKLM）（有关缩写的完整列表，请参见 `registry.data[]` 的注释）；
- **mode** - 可能的值：
values（默认）或 *keys*；
- **name regexp** - 仅发现名称与正则表达式匹配的值（默认 - 发现所有值）。仅在 *mode* 为 *values* 时允许。

包含空格的键必须用双引号括起来。

示例：

```
registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,values,"^DisplayName|DisplayVersion$
JSON 将包含键、最后一个子键、值名称、值类型和值数据的详细信息。
registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,values] # 返回此键中所有值的数据。
JSON 将包含键、最后一个子键、值名称、值类型和值数据的详细信息。
registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,keys] # 返回此键的所有子键。
JSON 将包含键和最后一个子键的详细信息。
```

service.discovery

 Windows 服务列表。
用于低级发现。
 返回值：*JSON object*。

service.info[service,<param>]

 有关服务的信息。
 返回值：- *Integer* - 当 `param` 为 *state*、*startup* 时；

 具体对于 *state*：- 0 - 运行中 - 1 - 已暂停 - 2 - 正在启动 - 3 - 正在暂停 - 4 - 正在继续 - 5 - 正在停止 - 6 - 已停止 - 7 - 未知 - 255 - 不存在此服务

 具体对于 *startup*：- 0 - 自动 - 1 - 自动（延迟启动）- 2 - 手动 - 3 - 已禁用 - 4 - 未知 - 5 - 自动触发启动 - 6 - 自动延迟触发启动 - 7 - 手动触发启动 - *String* - 当 `param` 为 *displayname*、*path*、*user* 时；- *Text* - 当 `param` 为 *description* 时。

参数：

- **service** - MMC Services 管理单元中显示的实际服务名称或其显示名称；
- **param** - *state*（默认）、*displayname*、*path*、*user*、*startup* 或 *description*。

注释：

- 像 `service.info[service,state]` 和 `service.info[service]` 这样的项将返回相同的信息；
- 只有当 `param` 为 *state* 时，此监控项才会为不存在的服务返回值（255）。

示例：

```
service.info[SNMPTRAP] - SNMPTRAP 服务的状态；
service.info[SNMP Trap] - 同一服务的状态，但指定了显示名称；
service.info[EventLog,startup] - EventLog 服务的启动类型
```

services[<type>,<state>,<exclude>]

 服务列表。
 返回值：*0* - 如果为空；*Text* - 以换行符分隔的服务列表。

参数：

- **type** - *all*（默认）、*automatic*、*manual* 或 *disabled*；
- **state** - *all*（默认）、*stopped*、*started*、*start_pending*、*stop_pending*、*running*、*continue_pending*、*pause_pending* 或 *paused*；
- **exclude** - 要从结果中排除的服务。被排除的服务应使用双引号括起来，以逗号分隔，且不包含空格。

示例：

```
services[,started] # 返回已启动服务的列表；
services[automatic, stopped] # 返回应处于运行状态但当前已停止的服务列表；
services[automatic, stopped, "service1,service2,service3"] # 返回应处于运行状态但当前已停止的服务列表，不包括名为 "s
```

vm.vmemory.size[<type>]

 虚拟内存大小，以字节或占总量的百分比表示。
 返回值：*Integer* - 用于字节；*float* - 用于百分比。

参数：

- **type** - 可选值：*available*（可用虚拟内存）、*pavailable*（可用虚拟内存，百分比）、*pused*（已用虚拟内存，百分比）、*total*（虚拟内存总量，默认值）或 *used*（已用虚拟内存）

注释：

- 虚拟内存统计信息的监控基于：
  - Windows 上的虚拟内存总量（物理内存总量 + 页面文件大小）；
  - Zabbix agent 可提交的最大内存量；
  - 系统或 Zabbix agent 当前的已提交内存限制，以较小者为准。

示例：

```
vm.vmemory.size[pavailable] # 返回可用虚拟内存的百分比
```

wmi.get[<namespace>,<query>]

 执行 WMI 查询并返回第一个选中的对象。
 返回值：*Integer*、*float*、*string* 或 *text*（取决于请求）。

参数：

- **namespace** - WMI 命名空间；
- **query** - 返回单个对象的 WMI 查询。

WMI 查询使用 WQL 执行。

示例：

```
wmi.get[root\cimv2,select status from Win32_DiskDrive where Name like '%PHYSICALDRIVE0%'] # 返回第一个物理磁盘
```

wmi.getall[<namespace>,<query>]

 执行 WMI 查询并返回完整响应。可用于低级别发现。
 返回值：*JSON object*

参数：

- **namespace** - WMI 命名空间；
- **query** - WMI 查询。

注释：

- WMI 查询使用 WQL 执行。
- 可以使用 JSONPath 预处理 来指向返回的 JSON 中更具体的值。

示例：

```
wmi.getall[root\cimv2,select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'] # 返回物理磁盘的状态信息
```

监控 Windows 服务

本教程提供了设置 Windows 服务监控的分步说明。
假定 Zabbix 服务器和 agent 已配置并正常运行。

步骤 1

获取服务名称。

你可以通过打开 MMC 的“服务”管理单元并查看该服务的属性来获取服务名称。在 常规选项卡中，你应该会看到一个名为 服务名称的字段。其后的值就是你在设置监控项时要使用的名称。例如，如果你想监控 “workstation” 服务，那么你的服务名称可能是：**lanmanworkstation**。

第 2 步

为监控该服务，配置一个监控项。

监控项 `service.info[service,<param>]` 用于检索特定服务的信息。根据所需信息，指定 `param` 选项，其可接受以下值：*displayname*、*state*、*path*、*user*、*startup* 或 *description*。如果未指定 `param`（`service.info[service]`），则默认值为 *state*。

返回值类型取决于所选的 `param`：*state* 和 *startup* 返回 integer；*displayname*、*path* 和 *user* 返回字符串；*description* 返回文本。

示例：

- 键值： `service.info[lanmanworkstation]`
- 信息类型： Numeric (unsigned)

监控项 `service.info[lanmanworkstation]` 将以数值形式检索该服务的状态信息。要在前端中将数值映射为文本表示形式（例如将 “0” 显示为 “Running”、“1” 显示为 “Paused” 等），可以在配置该监控项的主机上配置value mapping。为此，可以将模板 *Windows services by Zabbix agent* 或 *Windows services by Zabbix agent active* 链接到主机，或者在主机上新建一个值映射，并基于上述模板中配置的 *Windows service state* 值映射进行设置。

请注意，上述两个模板都配置了发现规则，会自动发现服务。如果不需要此功能，可在将模板链接到主机后，在主机级别禁用发现规则。

发现 Windows 服务

低级别发现 提供了一种为计算机上的不同实体自动创建监控项、触发器和图形的方法。
Zabbix 可以自动开始监控您机器上的 Windows 服务，而无需知道服务的确切名称，也无需为每个服务手动创建监控项。
可以使用过滤器仅为感兴趣的服务生成实际的监控项、触发器和图形。

日志文件监控

概述

Zabbix 可用于对支持或不支持日志轮转的日志文件进行集中监控和分析。

当日志文件包含某些字符串或字符串模式时，可以使用通知来提醒用户。

要监控日志文件，必须具备以下条件：

- 主机上正在运行 Zabbix agent。
- 已设置日志监控监控项。

> **Attention:**
> 被监控日志文件的大小限制取决于大文件支持。

配置

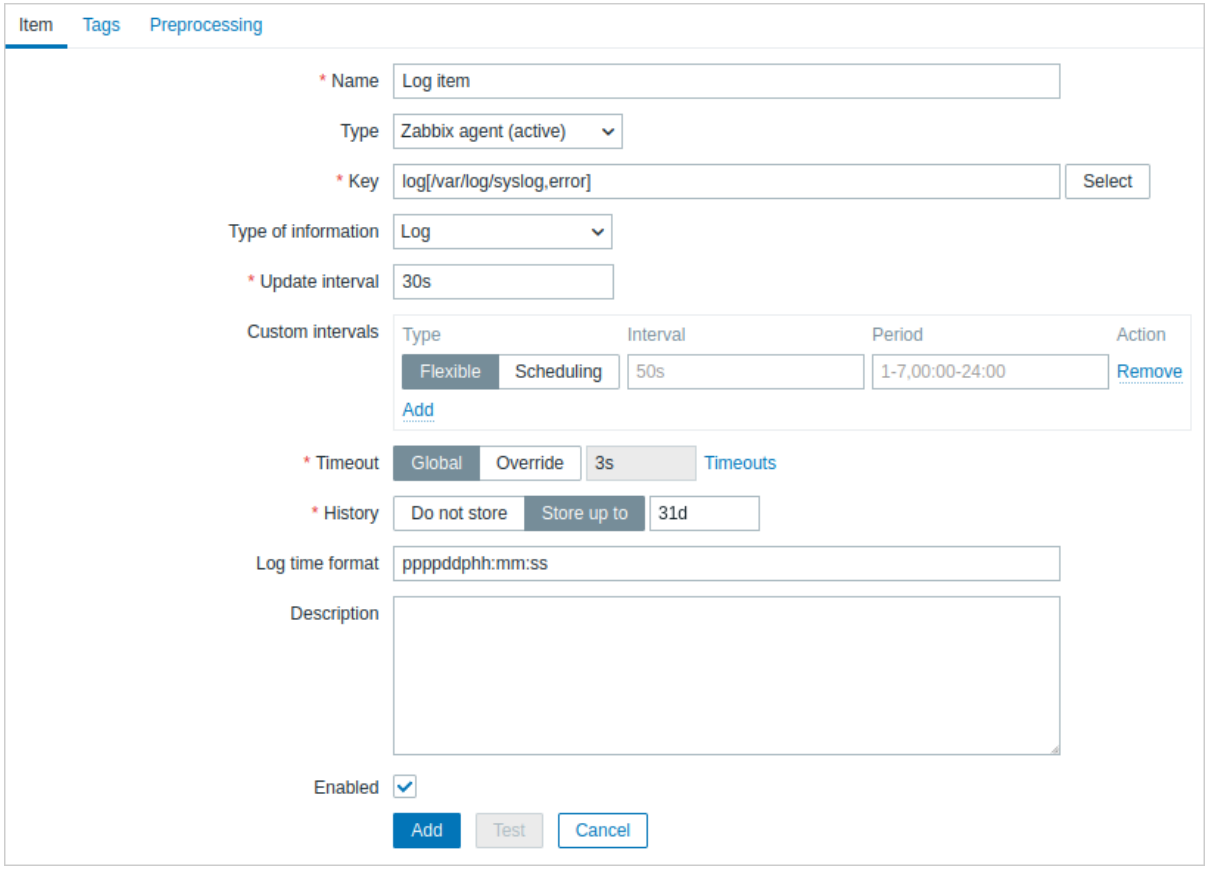
验证 agent 参数

请确保在agent 配置文件 中：

- `Hostname` 参数与前端中的主机名一致。
- `ServerActive` 参数中指定了用于处理主动检查的服务器。

监控项配置

配置日志监控监控项。



所有必填字段均以红色星号标记。

对于日志监控监控项，您需要输入：

类型	在此处选择 **Zabbix agent (active)**。
键值	使用以下监控项键之一： `log[]` 或 `logrt[]`： 这两个监控项键允许您监控日志，并根据内容正则表达式（如果存在）筛选日志条目。 例如：`log[/var/log/syslog,error]`。请确保'zabbix' 用户对该文件具有读取权限；否则，监控项状态将设置为 "不支持"。 **log.count[]** 或 **logrt.count[]**： 这两个监控项键仅返回匹配行的数量。 有关这些监控项键及其参数的使用详情，请参阅受支持的Zabbix agent 监控项键部分。
信息类型	自动预填充： 对于 `log[]` 或 `logrt[]` 监控项 - 日志； 对于 `log.count[]` 或 `logrt.count[]` 监控项 - 数值（无符号）。 如果选择使用 `output` 参数，则可以手动选择除 日志之外的适当信息类型。 请注意，选择非 "日志" 信息类型将导致本地时间戳丢失。
更新间隔（秒）	此参数定义 Zabbix agent 检查日志文件是否发生更改的频率。将其设置为 1 秒可确保尽快获取新记录。
日志时间格式	您可以在此字段中指定用于解析日志行时间戳的模式。支持以下占位符： * **y**：年（*1970-2038*） * **M**：月（*01-12*） * **d**：日（*01-31*） * **h**：小时（*00-23*） * **m**：分钟（*00-59*） * **s**：秒（*00-59*） 如果留空，时间戳将设置为 Unix 时间中的 0，表示 `January 1, 1970`。 例如，考虑以下 Zabbix agent 日志文件中的一行： `23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).` 该行以六个字符位置的 PID 开头，后面依次是日期、时间以及消息的其余部分。 此行的日志时间格式为"ppppp:yyyyMMdd:hhmmss"。 请注意，"p" 和":" 字符是占位符，可以是除"yMdhms" 之外的任意字符。

重要说明

- 服务器和 agent 会通过两个计数器跟踪被监控日志的大小以及最后修改时间（对于 `logrt`）。此外：
  - agent 还会在内部使用 inode 编号（在 UNIX/GNU/Linux 上）、文件索引（在 Microsoft Windows 上）以及前 512 个日志文件字节的 MD5 校验和，以便在日志文件被截断和轮转时改进判断。
  - 在 UNIX/GNU/Linux 系统上，假定存放日志文件的文件系统会报告 inode 编号，这些编号可用于跟踪文件。
  - 在 Microsoft Windows 上，Zabbix agent 会确定日志文件所在的文件系统类型，并使用：
    * 在 NTFS 文件系统上使用 64 位文件索引。
    * 在 ReFS 文件系统上（仅从 Microsoft Windows Server 2012 起）使用 128 位文件 ID。
    * 在文件索引会变化的文件系统上（例如 FAT32、exFAT），会使用回退算法，以便在日志文件轮转导致多个日志文件具有相同最后修改时间时，在不确定条件下采取合理处理方式。
  - inode 编号、文件索引和 MD5 校验和由 Zabbix agent 在内部收集。它们不会传输到 Zabbix 服务器，并且在 Zabbix agent 停止时会丢失。
  - 不要修改日志文件的最后修改时间（例如使用 `touch`），也不要通过将文件复制回其原始名称来替换被监控的日志文件（这会创建新的 inode）。在这两种情况下，Zabbix 可能会将该文件视为不同的文件，并从头重新读取，这可能会产生重复告警。
  - 如果 `logrt[]` 监控项存在多个匹配的日志文件，并且 Zabbix agent 正在跟踪其中最新的一个，而这个最新日志文件被删除，则会记录警告消息 `"there are no files matching "<regexp mask>" in "<directory>"`。Zabbix agent 会忽略修改时间早于当前正在检查的 `logrt[]` 监控项所见到的最新修改时间的日志文件。
- agent 会从上次停止的位置开始读取日志文件。
- 已分析的字节数（大小计数器）和最后修改时间（时间计数器）会存储在 Zabbix 数据库中，并发送给 agent，以确保在 agent 刚启动，或接收到之前被禁用或不受支持的监控项时，agent 会从该位置开始读取日志文件。不过，如果 agent 从服务器接收到一个非零大小计数器，但 `logrt[]` 或 `logrt.count[]` 监控项无法找到匹配文件，则当这些文件稍后出现时，大小计数器会重置为 0，以便从头开始分析。
- 每当日志文件变得比 agent 已知的日志大小计数器更小时，计数器都会重置为零，agent 会在考虑时间计数器的情况下从日志文件开头开始读取。
- 如果目录中存在多个具有相同最后修改时间的匹配文件，那么 agent 会尝试正确分析所有具有相同修改时间的日志文件，并避免跳过数据或重复分析相同数据，尽管在所有情况下都不能保证。agent 不会假定任何特定的日志文件轮转方案，也不会识别轮转方案。当出现多个具有相同最后修改时间的日志文件时，agent 会按字典序降序处理它们。因此，对于某些轮转方案，日志文件会按其原始顺序进行分析和报告。对于其他轮转方案，则不会遵循原始日志文件顺序，这可能导致匹配到的日志文件记录以不同顺序报告（如果日志文件具有不同的最后修改时间，则不会出现此问题）。
- Zabbix agent 每隔 *Update interval* 秒处理一次日志文件的新记录。
- Zabbix agent 每秒发送的日志文件内容不超过 `maxlines`。该限制可防止网络和 CPU 资源过载，并覆盖agent 配置文件 中 `MaxLinesPerSecond` 参数提供的默认值。
- 为了找到所需字符串，Zabbix 会处理比 `MaxLinesPerSecond` 设置多 10 倍的新行。因此，例如，如果 `log[]` 或 `logrt[]` 监控项的 *Update interval* 为 1 秒，默认情况下 agent 最多会分析 200 条日志文件记录，并且在一次检查中最多向 Zabbix 服务器发送 20 条匹配记录。通过在 agent 配置文件中增加 `MaxLinesPerSecond`，或在监控项键中设置 `maxlines` 参数，可将限制提高到一次检查最多分析 10000 条日志文件记录，并向 Zabbix 服务器发送 1000 条匹配记录。如果 *Update interval* 设置为 2 秒，则一次检查的限制会比 *Update interval* 为 1 秒时高 2 倍。
- 此外，即使其中没有非日志值，`log` 和 `log.count` 的值也始终限制为 agent 发送缓冲区大小的 50%。因此，要使 `maxlines` 的值能够在一次连接中发送（而不是分多次连接），agent `BufferSize` 参数至少必须为 `maxlines` x 2。Zabbix agent 可以在收集日志期间上传数据，从而释放缓冲区；而 Zabbix agent 2 会停止日志收集，直到数据上传并释放缓冲区为止，该过程是异步执行的。
- 在没有日志监控项的情况下，所有 agent 缓冲区大小都用于非日志值。当日志值到来时，它们会根据需要替换较旧的非日志值，最多占用指定的 50%。
- 对于长度超过 256kB 的日志记录，只有前 256kB 会与正则表达式进行匹配，其余记录内容会被忽略。不过，如果 Zabbix agent 在处理长记录时停止，agent 内部状态会丢失，且在 agent 重新启动后，该长记录可能会再次被分析，并且结果可能不同。
- 关于 \ 路径分隔符的特别说明：如果 `file_format` 为 `file\.log`，则不应存在 `file` 目录，因为无法明确判断 `.` 是转义字符，还是文件名的第一个字符。
- `logrt` 的正则表达式仅支持文件名，不支持目录正则匹配。
- 在 UNIX 平台上，如果预期存放日志文件的目录不存在，`logrt[]` 监控项会变为 NOTSUPPORTED。
- 在 Microsoft Windows 上，如果目录不存在，该监控项不会变为 NOTSUPPORTED（例如，监控项键中的目录拼写错误时）。
- `logrt[]` 监控项缺少日志文件不会使其变为 NOTSUPPORTED。读取 `logrt[]` 监控项日志文件时发生的错误会作为警告记录到 Zabbix agent 日志文件中，但不会使该监控项变为 NOTSUPPORTED。
- Zabbix agent 日志文件有助于找出 `log[]` 或 `logrt[]` 监控项为何变为 NOTSUPPORTED。Zabbix 可以监控其 agent 日志文件，但 `DebugLevel=4` 或 `DebugLevel=5` 时除外。
- 使用正则表达式搜索问号，例如 `\?`，如果文本文件包含 NUL 符号，可能会产生误报，因为 Zabbix 会将这些符号替换为 ?，以继续处理该行直到换行符。

提取正则表达式的匹配部分

有时，当找到正则表达式匹配项时，我们可能只希望从目标文件中提取感兴趣的值，而不是返回整行内容。

日志监控项能够从匹配的行中提取所需的值。这是通过在 `log` 和 `logrt` 监控项中添加 **output** 参数实现的。

使用 `output` 参数可以指定匹配结果中我们感兴趣的 “捕获组”。

例如：

```
log[/path/to/the/file,"large result buffer allocation.*Entries: ([0-9]+)",,,,\1]
```

该监控项应返回以下内容中找到的条目数：

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result
buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC
ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

由于 \1 引用了唯一的第一个捕获组（[0-9]+），因此只会返回数字。

此外，借助提取并返回数字的功能，该值还可以用于定义触发器。

使用 maxdelay 参数

日志监控项中的 maxdelay 参数允许忽略日志文件中的一些较旧行，以便优先分析在 maxdelay 秒内的最新行。

Warning:
指定 maxdelay > 0 可能会导致忽略重要的日志文件记录并漏报告警。请谨慎使用，仅在必要时自行承担风险。

默认情况下，用于日志监控的监控项会跟踪日志文件中出现的所有新行。不过，在某些情况下，有些应用程序会开始向日志文件写入大量消息。例如，如果数据库或 DNS 服务器不可用，这类应用程序会向日志文件中大量写入几乎相同的错误消息，直到恢复正常运行。默认情况下，所有这些消息都会被认真分析，并且匹配的行会按照 log 和 logrt 监控项的配置发送到服务器。

内置的过载保护由可配置的 maxlines 参数组成（用于保护服务器免受过多传入的匹配日志行影响），以及 10*maxlines 限制（用于保护主机 CPU 和 I/O，避免 agent 在一次检查中过载）。不过，内置保护仍然存在两个问题。首先，大量可能并不那么有信息量的消息会被报告到服务器，并占用数据库空间。其次，由于每秒可分析的行数有限，agent 可能会在数小时内落后于最新的日志记录。很可能，您更希望尽快获知日志文件中的当前情况，而不是花几个小时去翻看旧记录。

解决这两个问题的方法就是使用 maxdelay 参数。如果指定了 maxdelay > 0，则在每次检查期间会测量已处理字节数、剩余字节数和处理时间。根据这些数值，agent 会计算一个估计延迟——也就是分析日志文件中所有剩余记录所需的秒数。

如果延迟不超过 maxdelay，则 agent 会像往常一样继续分析日志文件。

如果延迟大于 maxdelay，则 agent 会通过“跳过”日志文件中的一段内容，将其忽略并跳转到一个新的估计位置，以便在 maxdelay 秒内分析剩余行。

请注意，agent 甚至不会将被忽略的行读入缓冲区，而是会计算一个大致的跳转位置。

跳过日志文件行的事实会记录在 agent 日志文件中，如下所示：

```
14287:20160602:174344.206 item:"logrt["/home/zabbix32/test[0-9].log",ERROR,,1000,,,120.0]"
logfile:"/home/zabbix32/test1.log" skipping 679858 bytes
(from byte 75653115 to byte 76332973) to meet maxdelay
```

其中“to byte”的数值是近似值，因为在“跳转”之后，agent 会将文件中的位置调整到某一日志行的开头，而该位置可能会更靠后，也可能更靠前。

根据日志文件增长速度与分析速度的对比，您可能会看到没有“跳转”、很少或经常“跳转”、大幅或小幅“跳转”，甚至每次检查都会有一次小“跳转”。系统负载和网络延迟的波动也会影响延迟的计算，因此也会影响为保持跟进 maxdelay 参数而进行的“跳转”。

不建议设置 maxdelay < update interval（这可能会导致频繁的小“跳转”）。

关于处理'copytruncate' 日志文件轮转的说明

带有 copytruncate 选项的 logrt 假定不同的日志文件包含不同的记录（至少它们的时间戳不同），因此初始块（最多到前 512 字节）的 MD5 校验和也会不同。如果两个文件的初始块 MD5 校验和相同，则表示其中一个是原始文件，另一个是副本。

带有 copytruncate 选项的 logrt 会尽力正确处理日志文件副本，而不会报告重复项。但是，不建议生成多个具有相同时间戳的日志文件副本、日志文件轮转频率高于 logrt[] 监控项更新间隔，以及频繁重启 agent。agent 会尽量合理处理所有这些情况，但无法保证在所有情况下都能得到良好结果。

关于 log*[] 监控项的持久化文件说明

持久化文件的用途

当 Zabbix agent 启动时，它会从 Zabbix 服务器或 proxy 接收一组主动检查。对于 log*[] 监控项，它会接收已处理的日志大小和修改时间，以便确定从哪里开始监控日志文件。根据文件系统报告的实际日志文件大小和修改时间，agent 会决定是从已处理的日志大小继续监控日志文件，还是从头重新分析日志文件。

运行中的 agent 会维护一组更大的属性，用于在各次检查之间跟踪所有受监控的日志文件。当 agent 停止时，这些内存中的状态会丢失。

新的可选参数 persistent_dir 指定一个目录，用于将 log[]、log.count[]、logrt[] 或 logrt.count[] 监控项的状态存储到文件中。Zabbix agent 重启后，日志监控项的状态会从持久化文件中恢复。

其主要使用场景是监控位于镜像文件系统上的日志文件。在某个时间点之前，日志文件会同时写入两个镜像。随后镜像被拆分。在活动副本上，日志文件仍在增长，并产生新的记录。Zabbix agent 会分析这些记录，并将已处理的日志大小和修改时间发送到服务器。在被动副本上，日志文件保持不变，且明显落后于活动副本。之后，操作系统和 Zabbix agent 会从被动副本重新启动。Zabbix agent 从服务器接收到的已处理日志大小和修改时间，可能不适用于被动副本上的情况。为了从文件系统镜像拆分时 agent 停止的位置继续监控日志文件，agent 会从持久化文件中恢复其状态。

使用持久文件的 agent 操作

在启动时，Zabbix agent 对持久文件一无所知。
只有在从 Zabbix 服务器 (proxy) 收到活动检查列表后，agent 才会发现某些日志监控项应由指定目录下的持久文件提供支持。

在 agent 运行期间，持久文件会以写入方式打开（使用 fopen(filename, "w")），并用最新数据覆盖。
如果覆盖操作与文件系统镜像分裂同时发生，导致持久文件数据丢失的可能性非常小，因此无需特殊处理。
向持久文件写入后，不会强制同步到存储介质（不会调用 fsync()）。

只有在成功将匹配的日志文件记录或元数据（已处理的日志大小和修改时间）报告给 Zabbix 服务器后，才会使用最新数据进行覆盖。
如果日志文件持续变化，这种情况可能会在每次监控项检查时发生。

agent 关闭时不执行任何特殊操作。

在收到活动检查列表后，agent 会将过时的持久文件标记为删除。

当满足以下任一条件时，持久文件会变为过时：

1. 对应的日志监控项不再被监控。
2. 某个日志监控项被重新配置为与之前不同的 persistent_dir 位置。

删除会延迟 24 小时执行，因为处于 NOTSUPPORTED 状态的日志文件不会包含在活动检查列表中，但它们之后可能会变为 SUPPORTED，此时其持久文件将会有用。

如果 agent 在 24 小时到期前停止，那么这些过时文件将不会被删除，因为 Zabbix agent 不再从 Zabbix 服务器获取其位置相关信息。

Warning:
在 agent 停止期间，如果将某个日志监控项的 persistent_dir 重新配置回旧的 persistent_dir 位置，而用户又没有删除旧的持久文件，这将导致从旧持久文件恢复 agent 状态，从而造成消息丢失或误报告警。

持久化文件的命名和位置

Zabbix agent 根据键区分主动检查。例如，logrt[/home/zabbix/test.log] 和 logrt[/home/zabbix/test.log,] 是不同的监控项。在前端中将监控项 logrt[/home/zabbix/test.log,,,10] 修改为 logrt[/home/zabbix/test.log,,,20]，将导致从 agent 的主动检查列表中删除监控项 logrt[/home/zabbix/test.log,,,10]，并创建监控项 logrt[/home/zabbix/test.log,,,20]（在前端/服务器中进行修改时，某些属性会被保留，但在 agent 中不会）。

文件名由监控项键的 MD5 值和附加的监控项键长度组成，以降低发生冲突的可能性。例如，监控项 logrt[/home/zabbix50/test.log,,,,,,,] 的状态将保存在持久化文件 c963ade4008054813bbc0a650bb8e09266 中。

多个日志监控项可以使用相同的 persistent_dir 值。

应根据特定的文件系统布局、挂载点和挂载选项以及存储镜像配置来指定 persistent_dir - 持久化文件应位于与受监控日志文件相同的镜像文件系统上。

如果无法创建 persistent_dir 目录，或该目录不存在，或者 Zabbix agent 没有足够的权限来创建、写入、读取或删除文件，则日志监控项将变为 NOTSUPPORTED。

如果在 agent 运行期间移除了持久化存储文件的访问权限，或发生其他错误（例如磁盘已满），则错误会记录到 agent 日志文件中，但日志监控项不会变为 NOTSUPPORTED。

I/O 负载

监控项的持久化文件会在每次成功将一批数据（包含监控项的数据）发送到服务器后更新。例如，默认 BufferSize 为 100。如果某个日志监控项找到了 70 条匹配记录，那么前 50 条记录将作为一个批次发送，持久化文件会更新，然后剩余的 20 条记录将作为第二个批次发送（在积累更多数据时，可能会有一些延迟），并且持久化文件会再次更新。

agent 与服务器之间通信失败时的处理

来自 log[] 和 logrt[] 监控项的每一条匹配行，以及每个 log.count[] 和 logrt.count[] 监控项检查的结果，都需要在 agent 发送缓冲区中指定的 50% 区域内占用一个空闲槽位。
缓冲区中的元素会定期发送到服务器（或 proxy），随后这些槽位会再次变为空闲。

当 agent 发送缓冲区中指定的日志区域仍有空闲槽位，且 agent 与服务器（或 proxy）之间的通信失败时，日志监控结果会累积在发送缓冲区中。
这有助于缓解短暂的通信故障。

在较长时间的通信故障期间，所有日志槽位都会被占满，并会执行以下操作：

- `log[]` 和 `logrt[]` 监控项检查会停止。
  当通信恢复且缓冲区中有空闲槽位可用时，检查会从上一次的位置继续。
  不会丢失任何匹配行，只是会稍后报告。
- 如果 `maxdelay = 0`（默认值），`log.count[]` 和 `logrt.count[]` 检查会停止。
  其行为与上文所述的 `log[]` 和 `logrt[]` 监控项类似。
  请注意，这可能会影响 `log.count[]` 和 `logrt.count[]` 的结果：例如，某次检查在日志文件中统计到 100 条匹配行，但由于缓冲区中没有空闲槽位，检查被停止。
  当通信恢复后，agent 会再次统计这 100 条匹配行，并且还会统计 70 条新的匹配行。
  此时 agent 发送的 count = 170，就像这些匹配行是在一次检查中发现的一样。
- 对于 `maxdelay > 0` 的 `log.count[]` 和 `logrt.count[]` 检查：如果检查过程中没有发生“跳跃”，则其行为与上文所述类似。
  如果在日志文件行之间发生了“跳跃”，则会保留跳跃后的位置信息，而统计结果会被丢弃。
  因此，即使在通信失败的情况下，agent 也会尽量跟上不断增长的日志文件。

正则表达式编译错误和运行时错误的处理

如果 `log[]`、`logrt[]`、`log.count[]` 或 `logrt.count[]` 监控项中使用的正则表达式无法由 PCRE 或 PCRE2 库编译，则该监控项将进入“不支持”状态，并显示错误消息。要继续监控日志监控项，应修复该正则表达式。

如果正则表达式编译成功，但在运行时失败（针对部分或全部日志记录），则日志监控项仍保持受支持状态，监控继续进行。运行时错误会记录在 Zabbix agent 日志文件中（不包含日志文件记录）。

日志记录速率限制为每次检查最多记录一个运行时错误，以便 Zabbix agent 能够监控自身的日志文件。例如，如果分析了 10 条记录，其中 3 条因正则表达式运行时错误而失败，则 agent 日志中只会生成一条记录。

例外情况：如果 `MaxLinesPerSecond=1` 且 `update interval=1`（每次检查只能分析一条记录），则不会记录正则表达式运行时错误。

发生运行时错误时，zabbix_agentd 会记录监控项键值，zabbix_agent2 会记录监控项 ID，以帮助识别发生运行时错误的日志监控项。如果发生运行时错误，建议重新设计正则表达式。

简单检查

概述

简单检查通常用于对服务进行远程的无 agent 检查。

请注意，简单检查不需要 Zabbix agent。Zabbix 服务器/proxy 负责处理简单检查（建立外部连接等）。

使用简单检查的示例：

```
net.tcp.service[ftp,,155]
net.tcp.service[http]
net.tcp.service.perf[http,,8080]
net.udp.service.perf[ntp]
```

> **Note:**
> 简单检查监控项配置中的 *User name* 和 *Password* 字段（限制为 255 个字符）用于 VMware 监控项；在其他情况下将被忽略。

支持的检查

监控项键以不带可选参数和附加信息的形式列出。单击监控项键可查看完整详情。

另请参见VMware 监控项键。

监控项键	描述
icmpping	主机通过 ICMP ping 的可访问性。
icmppingloss	丢包百分比。
icmppingretry	主机通过带重试的 ICMP ping 的可访问性。
icmppingsec	ICMP ping 响应时间。
net.tcp.service	检查某个服务是否正在运行并接受 TCP 连接。
net.tcp.service.perf	检查 TCP 服务的性能。
net.udp.service	检查某个服务是否正在运行并响应 UDP 请求。

监控项键	描述
net.udp.service.perf	检查 UDP 服务的性能。

监控项键值详情

不带尖括号的参数为必填参数。带有尖括号 **<** **>** 标记的参数为可选参数。

icmpping[<target>,<packets>,<interval>,<size>,<timeout>,<options>]

 通过 ICMP ping 检测主机可达性。
 返回值：*0* - ICMP ping 失败；*1* - ICMP ping 成功。

参数：

- **target** - 主机 IP 或 DNS 名称；
- **packets** - 数据包数量；
- **interval** - 连续数据包之间的时间间隔，单位为毫秒；
- **size** - 数据包大小，单位为字节；
- **timeout** - 超时时间，单位为毫秒；
- **options** - 用于允许重定向：如果为空（默认值），则将重定向响应视为目标主机不可达；如果设置为 *allow_redirect*，则将重定向响应视为目标主机可达。

另请参见默认值 表。

注释：

- 如果返回了由 `packets` 参数定义的至少一个数据包，则监控项返回 1。如果所有数据包都未返回，则监控项返回 0。

示例：

```
icmpping[,4] #如果返回了四个数据包中的至少一个，则监控项返回 1。
```

icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>,<options>]

 丢失数据包的百分比。
 返回值：浮点数。

参数：

- **target** - 主机 IP 或 DNS 名称；
- **packets** - 数据包数量；
- **interval** - 连续数据包之间的时间（以毫秒为单位）；
- **size** - 数据包大小（以字节为单位）；
- **timeout** - 超时（以毫秒为单位）；
- **options** - 用于允许重定向：如果为空（默认值），则重定向响应被视为目标主机关闭；如果设置为 *allow_redirect*，则重定向响应被视为目标主机启动。

另请参阅默认值 表。

icmppingretry[<target>,<retries>,<backoff>,<size>,<timeout>,<options>]

 通过带重试的 ICMP ping 检查主机可达性。如果第一个数据包成功，则停止；如果数据包失败，则重试，直到达到 `retries` 参数定义的最大尝试次数。此监控项有助于减少通过网络发送的数据包数量。
 返回值：*0* - ICMP ping 失败；*1* - ICMP ping 成功。

参数：

- **target** - 主机 IP 或 DNS 名称；
- **retries** - 对目标执行 ping 尝试的次数，不包括第一次尝试（0 或更大；默认值为 1）；
- **backoff** - 每次后续请求中等待时间的乘数（范围为 1.0-5.0；默认值为 1.0）；
- **size** - 数据包大小（字节）；
- **timeout** - 超时时间（毫秒）；
- **options** - 用于允许重定向：如果为空（默认值），则重定向响应会被视为目标主机不可用；如果设置为 *allow_redirect*，则重定向响应会被视为目标主机可用。

另请参见默认值表。

icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>,<options>]

 ICMP ping 响应时间（以秒为单位）。
 返回值：*Float*。

参数：

- **target** - 主机 IP 或 DNS 名称；
- **packets** - 数据包数量；
- **interval** - 连续数据包之间的时间（以毫秒为单位）；

- **size** - 数据包大小（以字节为单位）；
- **timeout** - 超时（以毫秒为单位）；
- **mode** - 可能的值：*min*、*max* 或 *avg*（默认）；
- **options** - 用于允许重定向：如果为空（默认值），则重定向响应将被视为目标主机已关闭；如果设置为 *allow_redirect*，则重定向响应将被视为目标主机已启动。

注释：

- 丢失或超时的数据包不用于计算；
- 如果主机不可用（已达到超时），则该监控项将返回 0；
- 如果返回值小于 0.0001 秒，则该值将设置为 0.0001 秒；
- 另请参阅默认值 表。

net.tcp.service[service,<ip>,<port>]

 检查某个服务是否正在运行并接受 TCP 连接。
 返回值：*0* - 服务不可用；*1* - 服务正在运行。

参数：

- **service** - 可选值：*ssh*、*ldap*、*smtp*、*ftp*、*http*、*pop*、*nntp*、*imap*、*tcp*、*https*、*telnet*（参见详细信息）；
- **ip** - IP 地址或 DNS 名称（默认使用主机 IP/DNS）；
- **port** - 端口号（默认使用标准服务端口号）。

注释：

- 请注意，当 *tcp* 作为服务类型时，必须指定端口；
- 这些检查可能会在系统守护进程日志文件中产生额外消息（通常会记录 SMTP 和 SSH 会话）；
- 当前不支持对加密协议的检查（例如使用 993 端口的 IMAP 或使用 995 端口的 POP）。作为变通方法，请对此类检查使用 `net.tcp.service[tcp,<ip>,port]`。

示例：

`net.tcp.service[ftp,,45]` #此监控项可用于测试 TCP 45 端口上 FTP 服务器的可用性。

Attention:
如果 SELinux 以 enforced 模式运行，自定义 TCP/UDP 简单检查可能会被策略阻止。要验证并允许新的出站连接，请查看审计拒绝记录：使用 `grep denied /var/log/audit/audit.log`

net.tcp.service.perf[service,<ip>,<port>]

 检查 TCP 服务的性能。
 返回值：*Float*：*0.000000* - 服务已关闭；*seconds* - 连接到服务所花费的秒数。

参数：

- **service** - 可能的值：*ssh*、*ldap*、*smtp*、*ftp*、*http*、*pop*、*nntp*、*imap*、*tcp*、*https*、*telnet*（参见详细信息）；
- **ip** - IP 地址或 DNS 名称（默认情况下使用主机 IP/DNS）；
- **port** - 端口号（默认情况下使用标准服务端口号）。

注释：

- 请注意，使用 *tcp* 服务时必须指定端口；
- 目前不支持检查加密协议(如端口 993 上的 IMAP 或端口 995 上的 POP)。作为解决方法，请使用 `net.tcp.service[tcp,<ip>,port]` 进行此类检查。

示例：

`net.tcp.service.perf[ssh]` #此监控项可用于测试 SSH 服务器的初始响应速度。

net.udp.service[service,<ip>,<port>]

 检查服务是否正在运行并响应 UDP 请求。
 返回值：*0* - 服务已关闭；*1* - 服务正在运行。

参数：- **service** - 可能的值：*ntp*（参见details）；- **ip** - IP 地址或 DNS 名称（默认情况下使用主机 IP/DNS）；- **port** - 端口号（默认情况下使用标准服务端口号）。

示例：

`net.udp.service[ntp,,45]` #此监控项可用于测试 UDP 端口 45 上的 NTP 服务的可用性。

net.udp.service.perf[service,<ip>,<port>]

 检查 UDP 服务的性能。
 返回值：*Float*：*0.000000* - 服务已关闭；*seconds* - 等待服务响应所花费的秒数。

Parameters:

- **service** - 可能的值：*ntp*（参见details）；

- **ip** - IP 地址或 DNS 名称（默认情况下使用主机 IP/DNS）；
- **port** - 端口号（默认情况下使用标准服务端口号）。

示例：

`net.udp.service.perf[ntp] #此监控项可用于测试 NTP 服务的响应时间。`

Attention:
对于 LDAP 简单检查中的 SourceIP 支持（例如 `net.tcp.service[ldap]`），需要 OpenLDAP 2.6.1 或更高版本。

超时处理

灵活的监控项超时虽然支持简单检查，但不适用于 `icmpping*` 和 VMware 监控项。请参见灵活超时支持。

ICMP ping

Zabbix 使用外部工具 **fping** 来处理 ICMP ping（**icmpping**、**icmppingloss**、**icmppingretry**、**icmppingsec**）。

安装

fping 不包含在 Zabbix 中，需要单独安装：

- 各种基于 Unix 的平台在其默认仓库中提供 fping 软件包，但默认未预装。在这种情况下，您可以使用包管理器安装 fping。
- Zabbix 为 RHEL 及其衍生版提供了 fping 软件包。请注意，这些软件包不提供官方支持。
- fping 也可以从源码编译安装。

配置

在 Zabbix 服务器/proxy 配置文件的 *FpingLocation* 参数中指定 fping 的位置（如果使用 IPv6 地址，则使用 *Fping6Location* 参数）。

fping 应可由 Zabbix 服务器/proxy 运行所使用的用户执行，并且该用户应具有足够的权限。

另请参见：已知问题，了解使用 3.10 以下版本的 fping 处理简单检查时的相关问题。

默认值

ICMP 检查参数的默认值、限制和说明：

参数	单位	说明	Fping 标志	默认值设置方		Zabbix 允许的限制	
				fping	**Zabbix**	最小值	最大值
packets	数量	发送到目标的请求数据包数量	-C		3	1	10000
interval	毫秒	向单个目标连续发送数据包之间的等待时间	-p	1000		20	无限制
size	字节	数据包大小（以字节为单位） x86 上为 56 字节，x86_64 上为 68 字节	-b	56 或 68		24	65507
timeout	毫秒	**fping v3.x** - 发送最后一个数据包后等待的超时时间，受 -C 标志影响 **fping v4.x** - 每个数据包的单独超时时间	-t	**fping v3.x** - 500 **fping v4.x** 及更高版本 - 继承自 -p 标志，但不超过 2000		50	无限制
retries	数量	尝试对目标执行 ping 的次数，不包括第一次尝试	-r	3	1	0	无限制
backoff factor	数量	每次后续请求中等待时间的乘数	-B	1.5	1.0	1.0	5.0

默认值可能会因平台和版本而略有不同。

此外，Zabbix 使用 fping 选项 `-i interval ms`（不要将其与上表中的监控项参数 `interval` 混淆，后者对应 fping 选项 `-p`）以及 `-S source IP address`（在较旧的 fping 版本中为 `-I`）。这些选项通过使用不同的选项组合运行检查来自动检测。Zabbix 会尝试使用三个值 0、1 和 10，检测 fping 对 `-i` 选项允许的最小毫秒值。第一个成功的值将用于后续的 ICMP 检查。此过程由每个ICMP pinger 进程单独执行。

自动检测到的 fping 选项每小时失效一次，并在下一次执行 ICMP 检查时重新检测。将DebugLevel 设置为 >=4，以便在服务器或 proxy 日志文件中查看此过程的详细信息。

Zabbix 会将任何 icmpping* 键需要检查的 IP 地址写入临时文件，然后将该文件传递给 fping。如果监控项具有不同的键参数，则只有键参数相同的监控项会写入同一个文件。写入同一个文件的所有 IP 地址都将由 fping 并行检查，因此无论文件中的 IP 地址数量是多少，Zabbix ICMP pinger 进程所花费的时间都是固定的。

VMware 监控项键值

VMware 监控item keys列表已移动到VMware monitoring区域。

SNMP agent

概述

您可能希望在打印机、网络交换机、路由器或 UPS 等设备上使用 SNMP 监控，这些设备通常都支持 SNMP，而且在这些设备上尝试安装完整操作系统和 Zabbix agent 并不现实。

为了能够检索这些设备上的 SNMP agent 提供的数据，Zabbix 服务器必须先通过指定 --with-net-snmp 标志进行初始配置，以启用 SNMP 支持。还建议安装 MIB 文件，以确保监控项值以正确的格式显示。如果没有 MIB 文件，可能会出现格式问题，例如将值以 HEX 而不是 UTF-8 显示，或反之。

SNMP 检查仅通过 UDP 协议执行。

如果 Zabbix 服务器和 proxy 守护进程收到不正确的 SNMP 响应，它们会记录类似以下内容的日志行：

```
SNMP response from host "gateway" does not contain all of the requested variable bindings
```

虽然这些日志不能覆盖所有有问题的情况，但它们有助于识别应禁用组合请求的单个 SNMP 设备。

对于 SNMP walk 和 get 监控项，Zabbix 服务器/proxy 最多会重试 5 次。重试机制不适用于 DNS 解析失败。

对于传统 SNMP 检查（单个 OID 编号或字符串），Zabbix 服务器/proxy 在查询失败后至少会重试一次：要么通过 SNMP 库的重试机制，要么通过内部的组合处理 机制。

Warning:
如果监控 SNMPv3 设备，请确保 msgAuthoritativeEngineID（也称为 snmpEngineID 或"Engine ID"）绝不会被两台设备共用。根据 RFC 2571（第 3.1.1.1 节），它对每台设备都必须是唯一的。

Warning:
RFC3414 要求 SNMPv3 设备持久保存其 engineBoots。某些设备不会这样做，这会导致它们在重启后，其 SNMP 消息因被视为过期而被丢弃。在这种情况下，需要在服务器/proxy 上手动清除 SNMP 缓存（使用-R snmp_cache_reload），或者重启服务器/proxy。

Note:
Zabbix 会缓存 SNMPv3 EngineID → IP 映射，并在后续检查中复用缓存的 EngineID，而不是每次都发送探测，从而减少网络流量。如果某个 EngineID 不能复用，则会通过探测重试以发现新的 EngineID。

配置 SNMP 监控

要通过 SNMP 开始监控设备，需要执行以下步骤：

第 1 步

找出你要监控的监控项的 SNMP 字符串（或 OID）。

要获取 SNMP 字符串列表，请使用 snmpwalk 命令（它是 net-snmp 软件的一部分，你应已在安装 Zabbix 时一并安装）或等效工具：

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

由于这里的 2c 表示 SNMP 版本，你也可以将其替换为 1，以表示设备上的 SNMP 版本 1。

这应该会返回一个 SNMP 字符串及其最后一个值的列表。如果没有返回结果，则可能是 SNMP 的 "community" 与标准的 public 不同，在这种情况下，你需要先找出它的实际值。

然后你可以逐项查看列表，直到找到你想监控的字符串，例如：如果你想监控交换机端口 3 的入站字节数，可以使用这一行中的 IF-MIB::ifHCInOctets.3 字符串：

```
IF-MIB::ifHCInOctets.3 = Counter64: 3409739121
```

现在你可以使用 snmpget 命令来查找 IF-MIB::ifHCInOctets.3 的数字 OID：

```
snmpget -v 2c -c public -On <host IP> IF-MIB::ifHCInOctets.3
```

请注意，字符串中的最后一个数字是你要监控的端口号。另请参见：动态索引。

这应该会返回类似如下内容：

```
.1.3.6.1.2.1.31.1.1.1.6.3 = Counter64: 3472126941
```

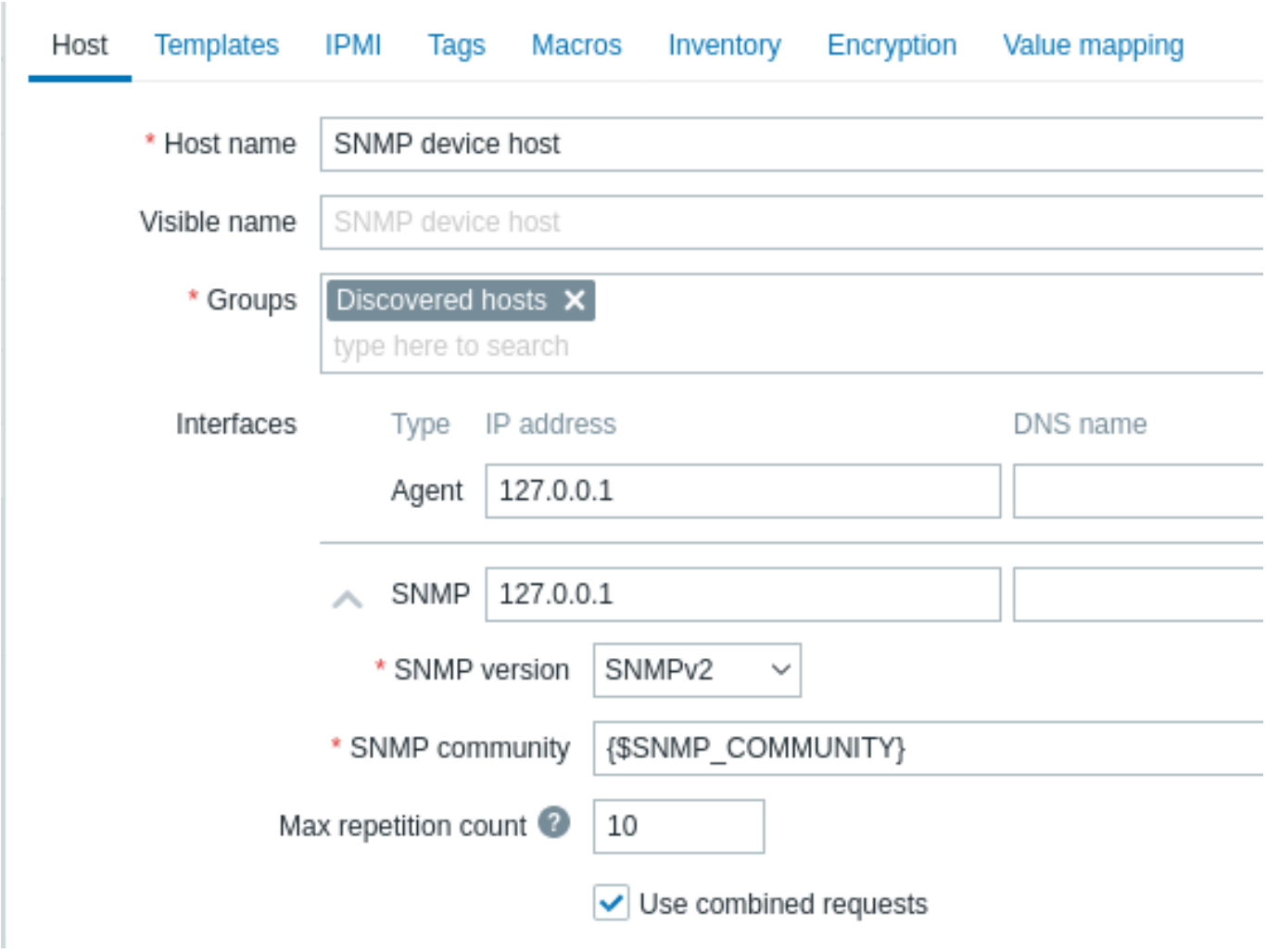
同样，OID 中最后一个数字就是端口号。

Note:
Zabbix 会将一些最常用的 SNMP OID 自动转换为数字表示。

在上一个示例中，值类型是"Counter64"，在内部对应于 ASN_COUNTER64 类型。支持的完整类型列表为 ASN_COUNTER、ASN_COUNTER64、ASN_UINTEGER、ASN_UNSIGNED64、ASN_INTEGER、ASN_INTEGER64、ASN_FLOAT、ASN_DOUBLE、ASN_TIMETICKS、ASN_GAUGE、ASN_IPADDRESS、ASN_OCTET_STR 和 ASN_OBJECT_ID。这些类型大致对应于 **snmpget** 输出中的"Counter32"、"Counter64"、"UInteger32"、"INTEGER"、"Float"、"Double"、"Timeticks"、"Gauge32"、"IpAddress"、"OCTET STRING"、"OBJECT IDENTIFIER"，但也可能显示为"STRING"、"Hex-STRING"、"OID" 等，具体取决于是否存在显示提示。

第 2 步

创建主机，使其对应于某个设备。



为主机添加一个 SNMP 接口：

1. 输入 IP 地址/DNS 名称和端口号。
2. 从下拉列表中选择 *SNMP* 版本。
3. 根据所选的 SNMP 版本添加接口凭据：
   - SNMPv1、v2 只需要 community（通常为'public'）。
   - SNMPv3 需要更具体的选项；请参见SNMPv3。

4. 为原生 SNMP 批量请求（GetBulkRequest-PDUs）指定最大重复值（默认值：10）；仅适用于 SNMPv2 和 v3 中的 `discovery[]` 和 `walk[]` 监控项。请注意，将此值设置得过高可能会导致 SNMP agent 检查超时。
5. 勾选 *Use combined requests* 复选框，以允许对 SNMP 请求进行组合处理（与原生 SNMP 批量请求"walk" 和"get" 无关）。

您可以使用提供的 SNMP 模板之一，它会自动添加一组监控项。使用模板之前，请确认它与该主机兼容。

点击 *Add* 保存主机。

SNMPv3

SNMPv3 需要以下参数：

- *Context name* - 输入上下文名称，以便在 SNMP 子网上标识监控项。
 此字段会解析用户宏。
- *Security name* - 输入安全名称。
 此字段会解析用户宏。
- *Security level* - 选择安全级别：
  - *noAuthNoPriv* - 不使用认证和隐私协议；
  - *AuthNoPriv* - 使用认证协议，不使用隐私协议；
  - *AuthPriv* - 同时使用认证和隐私协议。
- *Authentication protocol* - 选择认证协议：*MD5*、*SHA1*；在 net-snmp 5.8 及更新版本中，还可选择 *SHA224*、*SHA256*、*SHA384* 或 *SHA512*。
- *Authentication passphrase* - 输入认证口令。
 此字段会解析用户宏。
- *Privacy protocol* - 选择隐私协议：*DES*、*AES128*、*AES192*、*AES256*、*AES192C*（Cisco）或 *AES256C*（Cisco）。
 请参阅隐私协议支持说明。
- *Privacy passphrase* - 输入隐私口令。
 此字段会解析用户宏。

如果 SNMPv3 凭据（安全名称、认证协议/口令、隐私协议）错误：

- Zabbix 会从 net-snmp 收到 ERROR，唯独 *Privacy passphrase* 错误时，Zabbix 会从 net-snmp 收到 TIMEOUT 错误。
- SNMP 接口可用性将变为红色（不可用）。

Note:
在不更改 *Security name* 的情况下，对 *Authentication protocol*、*Authentication passphrase*、*Privacy protocol* 或 *Privacy passphrase* 所做的更改，通常会在 Zabbix 中更新相应的 SNMPv3 接口时自动应用。如果 *Security name* 也发生更改，则所有参数都会立即更新。

隐私协议支持

根据您的操作系统和 net-snmp 配置，某些隐私协议可能不可用：

- 在某些较新的操作系统上（例如 RHEL9），`net-snmp` 包已不再支持 DES。
- 在早于 RHEL 8、CentOS 8、Oracle Linux 8、Debian 12、Ubuntu LTS 22.04、openSUSE Leap 15.5 的操作系统上，AES192 及更强的加密协议默认不受支持。

要检查 net-snmp 库是否支持 AES192+，请使用以下任一方法：

1. `net-snmp-config`：

```
net-snmp-config --configure-options
```

如果输出中包含 `--enable-blumenthal-aes`，则表示支持 AES192+。

请注意，`net-snmp-config` 是 SNMP 开发包的一部分（Debian/Ubuntu 上为 `libsnmp-dev`，CentOS/RHEL/OL/SUSE 上为 `net-snmp-devel`），默认情况下可能未安装。

2. `snmpget`：

```
snmpget -v 3 -x AES-256
```

如果输出中包含 `Invalid privacy protocol specified after -3x flag: AES-256`，则表示不支持 AES192+。如果输出中包含 `No hostname specified.`，则表示不支持 AES192+。

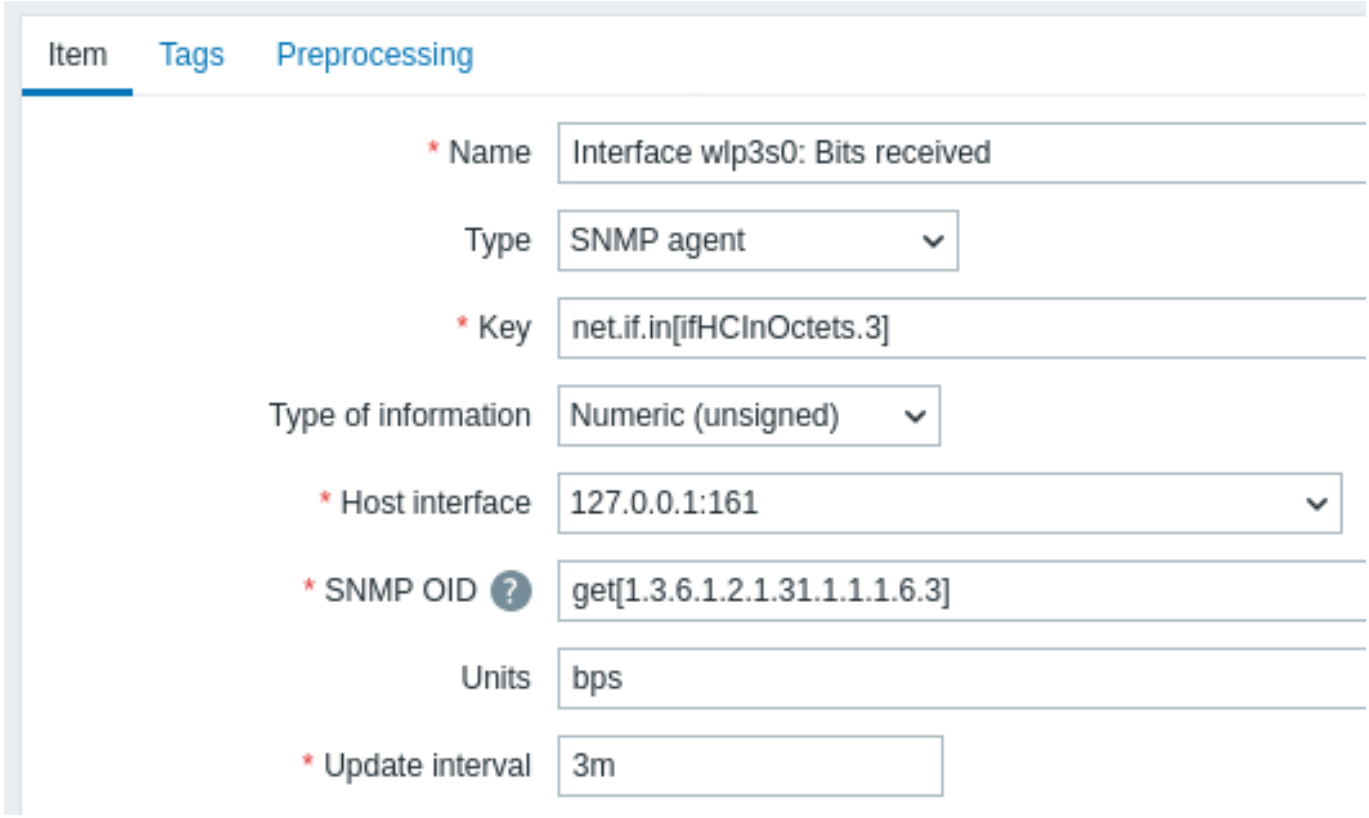
如果您的 net-snmp 库不支持 AES192 及更高版本的协议，请使用 `--enable-blumenthal-aes` 选项重新编译 net-snmp，然后在重新编译 Zabbix 服务器时指定选项 `--with-net-snmp=/home/user/yourcustomnetsnmp/bin/net-snmp-config`。

步骤 3

创建监控项进行监控。

现在返回 Zabbix，点击之前创建的 SNMP 主机的监控项。创建主机时是否使用了模板，决定了此处显示的是与主机关联的 SNMP 监控项列表，还是空列表。我们假设您将使用刚才通过 snmpwalk 和 snmpget 收集的信息自行创建监控项，因此请点击创建监控项。

在新的监控项表单中填写必填参数：



参数	描述
名称	输入监控项名称。
类型	在此选择 **SNMP agent**。
键值	输入具有明确含义的键值。
主机接口	确保选择 SNMP 接口，例如交换机或路由器的 SNMP 接口。

参数	描述
SNMP OID	使用以下支持的格式输入 OID 值： **walk[OID1,OID2,...]** - 获取值的子树。 例如：`walk[1.3.6.1.2.1.2.2.1.2,1.3.6.1.2.1.2.2.1.3]`。 此选项异步使用原生 SNMP 批量请求（GetBulkRequest-PDU）。 可以在监控项配置表单中设置此监控项的超时设置。如果设备不可访问，建议设置较低的超时值，以避免长时间延迟，因为在之前的请求超时或失败时，最多会重试 5 次（例如，3 秒超时可能导致等待 15 秒）。 您可以将此监控项用作主监控项，并使用依赖监控项通过预处理从主监控项中提取数据。 单个 snmp walk 中可以指定多个 OID，例如 `walk[OID1,OID2,...]`，以便一次异步处理一个 OID。 如果批量请求未返回结果，则会尝试在不使用批量请求的情况下获取单条记录。 支持将 MIB 名称作为参数；因此，`walk[1.3.6.1.2.1.2.2.1.2]` 和 `walk[ifDescr]` 将返回相同的输出。 如果指定了多个 OID/MIB，即 `walk[ifDescr,ifType,ifPhysAddress]`，则输出为拼接后的列表。 SNMPv2 和 v3 接口使用 GetBulk 请求，SNMPv1 接口使用 GetNext 请求；批量请求的最大重复次数在接口级别进行配置。 最大重复次数参数通过确定单个批量响应中返回的 OID 最大数量来影响批量请求。 较高的值会产生更大的批量响应，从而减少所需的传输次数。但是，并非所有设备都支持非常高的值，这可能会导致问题。 此监控项返回带有 -Oe -Ot -On 参数的 snmpwalk 工具输出。 您可以在SNMP 发现中将此监控项用作主监控项。 **get[OID]** - 异步获取单个值。 例如：`get[1.3.6.1.2.1.31.1.1.1.6.3]` 可以在监控项配置表单中设置此监控项的超时设置。如果设备不可访问，建议设置较低的超时值，以避免长时间延迟，因为在之前的请求超时或失败时，最多会重试 5 次（例如，3 秒超时可能导致等待 15 秒）。 **OID** - （旧版）输入单个文本或数字 OID，以同步获取单个值，也可以选择与其他值组合。 例如：`1.3.6.1.2.1.31.1.1.1.6.3`。 对于此选项，监控项检查超时将等于服务器配置文件中设置的值。 建议使用 `walk[OID]` 和 `get[OID]` 监控项以获得更好的性能。所有 `walk[OID]` 和 `get[OID]` 监控项均异步执行，无需在启动其他检查前等待某个请求的响应。DNS 解析同样是异步的。 异步检查的最大并发数为 1000（由MaxConcurrentChecksPerPoller 定义）。异步 SNMP poller 的数量由StartSNMPPollers 参数定义。 请注意，对于任一方法返回的网络流量统计数据，必须在预处理选项卡中添加每秒变化步骤；否则，您将从 SNMP 设备获取累计值，而不是最新变化。

所有必填输入字段均以红色星号标记。

现在保存监控项，然后转到监测 > 最新数据查看 SNMP 数据。

示例 1

通用示例：

Parameter	Description
OID	1.2.3.45.6.7.8.0（或.1.2.3.45.6.7.8.0）
Key	< 用作触发器引用的唯一字符串 > 例如，”my_param”。

请注意，OID 可以使用数字形式或字符串形式。但是，在某些情况下，字符串 OID 必须转换为数字表示形式。可使用 snmpget 工具实现此目的：

```
snmpget -On localhost public enterprises.ucdavis.memory.memTotalSwap.0
```

示例 2

监控运行时间：

Parameter	Description
OID	MIB::sysUpTime.0
Key	router.uptime
Value type	浮点数
Units	运行时间
Preprocessing step: Custom multiplier	0.01

原生 SNMP 批量请求

walk[OID1,OID2,...] 监控项允许使用原生 SNMP 功能执行批量请求（GetBulkRequest-PDU），该功能适用于 SNMP 版本 2/3。

SNMP 中的 GetBulk 请求会执行多个 GetNext 请求，并在单个响应中返回结果。这既可用于常规 SNMP 监控项，也可用于 SNMP 发现，以尽量减少网络往返次数。

SNMP **walk[OID1,OID2,...]** 监控项可用作主监控项，在一个请求中收集数据，并通过依赖监控项使用预处理按需解析响应。

请注意，使用原生 SNMP 批量请求与合并 SNMP 请求的选项无关。后者是 Zabbix 自有的合并多个 SNMP 请求的方式（请参见下一节）。

SNMP 批量监控项最多会重试五次，以避免其中一个数据包丢失导致请求失败。对于使用 `get` 和 `walk` 的 SNMP 监控项，其超时时间（在监控项配置表单中设置）适用于整个会话。无论是否完整检索到数据，都会应用超时时间；如果仅收到部分数据（例如，多个 OID 中只有一个 OID 的数据成功收集），则监控项将变为不支持，并显示消息 “Only partial data received”。如果达到超时时间，将进行重试，超时时间会重置，并重新发送上一次请求，使会话能够从上一次请求继续执行，即使单个数据包丢失或到达过晚。建议设置较短的超时时间，以避免设备不可访问时出现长时间延迟，因为如果前几次请求超时或失败，最多会进行 5 次重试（例如，3 秒的超时时间可能导致等待 15 秒）。

组合处理的内部机制

Zabbix 服务器和 proxy 可以在单个请求中向 SNMP 设备查询多个值。这会影响以下几种 SNMP 监控项：

- 常规 SNMP 监控项
- 具有动态索引的 SNMP 监控项
- SNMP 低级别发现规则

单个接口上具有相同参数的所有 SNMP 监控项都会被安排在同一时间进行查询。前两种监控项由轮询器以每批最多 128 个监控项的方式获取，而低级别发现规则仍像以前一样单独处理。

在较低层级，查询值时会执行两种操作：获取多个指定对象，以及遍历 OID 树。

对于 “获取” 操作，会使用 GetRequest-PDU，其中最多包含 128 个变量绑定。对于 “遍历” 操作，SNMPv1 使用 GetNextRequest-PDU，而 SNMPv2 和 SNMPv3 使用 GetBulkRequest，其 “max-repetitions” 字段最大为 128。

因此，组合处理对每种 SNMP 监控项类型带来的优势如下：

- 常规 SNMP 监控项受益于 “获取” 操作的改进；
- 具有动态索引的 SNMP 监控项同时受益于 “获取” 和 “遍历” 操作的改进：“获取” 用于验证索引，“遍历” 用于构建缓存；
- SNMP 低级别发现规则受益于 “遍历” 操作的改进。

但是，存在一个技术问题：并非所有设备都能够在单个请求中返回 128 个值。有些设备始终会返回正确的响应，但另一些设备在潜在响应超过某个限制后，会返回 “tooBig(1)” 错误，或者完全不响应。

为了确定针对给定设备的最佳对象查询数量，Zabbix 使用以下策略。它会谨慎地从每个请求查询 1 个值开始。如果成功，则在每个请求中查询 2 个值。如果再次成功，则在每个请求中查询 3 个值，并继续以类似方式将查询对象数量乘以 1.5，得到以下请求大小序列：1、2、3、4、6、9、13、19、28、42、63、94、128。

但是，一旦设备拒绝返回正确响应（例如，针对 42 个变量时），Zabbix 会执行两项操作。

首先，对于当前监控项批次，它会将单个请求中的对象数量减半，并查询 21 个变量。如果设备处于活动状态，那么在绝大多数情况下查询应该能够成功，因为已知 28 个变量可以正常工作，而 21 个变量明显少于该数量。但是，如果仍然失败，Zabbix 会退回到逐个查询值。如果此时仍然失败，则可以确定设备没有响应，问题与请求大小无关。

对于后续监控项批次，Zabbix 执行的第二项操作是从最后一个成功的变量数量开始（在本例中为 28 个），然后将请求大小逐个递增，直到达到限制。例如，假设最大的响应大小为 32 个变量，则后续请求的大小将依次为 29、30、31、32 和 33。最后一个请求将失败，Zabbix 之后将不再发出大小为 33 的请求。从此以后，Zabbix 对该设备的查询最多包含 32 个变量。

如果包含此数量变量的大型查询失败，可能有两种原因。设备用于限制响应大小的确切标准无法得知，但我们会尝试使用变量数量来近似判断。第一种可能性是，该变量数量大致处于设备实际响应大小限制附近：有时响应小于限制，有时则大于限制。第二种可能性是，任一方向上的 UDP 数据包恰好丢失。因此，如果 Zabbix 获取到失败的查询结果，它会减少要尝试获取的最大变量数量，以进一步进入设备能够稳定处理的范围，但最多只会执行两次。

在上面的示例中，如果包含 32 个变量的查询恰好失败，Zabbix 会将数量减少到 31 个。如果该请求也失败，Zabbix 会将数量减少到 30 个。但是，Zabbix 不会将数量减少到 30 个以下，因为它会认为后续失败是由于 UDP 数据包丢失，而不是设备的限制。

但是，如果设备由于其他原因无法正确处理组合请求，并且上述启发式方法不起作用，则每个接口都有一个 “Use combined requests” 设置，可以使用该设置为此设备禁用组合请求。

如果组合请求导致部分响应或格式错误的响应，从而造成每秒（增量）计算不正确（例如，接口计数器出现异常尖峰），请为受影响的接口禁用 *Use combined requests*，以强制执行按监控项分别查询；这通常可以防止错误尖峰。

或者，可以考虑使用异步 `get[]` 或 `walk[]` 监控项。这些监控项会异步执行，不受每个接口的 *Use combined requests* 批处理限制约束，因此可以替代传统的同步 OID 检查，以避免与组合请求相关的问题。请查找类似于概述部分中所示内容的服务器/proxy 日志条目，以帮助识别受影响的设备。

此外，如果接口经常变为不可用，可能需要增加Zabbix 服务器或Zabbix proxy配置文件中的 `UnavailableDelay` 参数，以降低请求频率。如果在发现或 OID 遍历期间接收到部分数据，监控项可能会变为不受支持。

动态索引

概述

虽然您可能会在 SNMP OID 中找到所需的索引号（例如网络接口的索引号），但有时不能完全依赖该索引号始终保持不变。

索引号可能是动态的——它们可能会随时间变化，因此您的监控项可能会因此停止工作。

为避免这种情况，可以定义一个 OID，以考虑索引号发生变化的可能性。

例如，如果您需要获取要附加到 **ifInOctets** 的索引值，并且该值对应于 Cisco 设备上的 **GigabitEthernet0/1** 接口，请使用以下 OID：

```
ifInOctets["index","ifDescr","GigabitEthernet0/1"]
```

语法

OID 使用一种特殊语法：

< 数据的 OID>[”index”,”< 索引的基础 OID>”,”< 要搜索的字符串 >”]

参数	说明
数据的 OID	用于在监控项中检索数据的主 OID。
index	处理方法。目前仅支持一种方法： **index** - 搜索索引并将其附加到数据 OID
索引的基础 OID	将查找此 OID，以获取与该字符串对应的索引值。
要搜索的字符串	查找时用于与某个值进行精确匹配的字符串。区分大小写。

示例

获取 *apache* 进程的内存使用情况。

如果使用此 OID 语法：

```
HOST-RESOURCES-MIB::hrSWRunPerfMem["index","HOST-RESOURCES-MIB::hrSWRunPath", "/usr/sbin/apache2"]
```

则会在这里查找索引号：

```
...
HOST-RESOURCES-MIB::hrSWRunPath.5376 = STRING: "/sbin/getty"
HOST-RESOURCES-MIB::hrSWRunPath.5377 = STRING: "/sbin/getty"
HOST-RESOURCES-MIB::hrSWRunPath.5388 = STRING: "/usr/sbin/apache2"
HOST-RESOURCES-MIB::hrSWRunPath.5389 = STRING: "/sbin/sshd"
...
```

现在我们得到了索引 5388。
该索引将附加到数据 OID 后，以便获取我们感兴趣的值：

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.5388 = INTEGER: 31468 KBytes
```

索引查找缓存

当请求动态索引监控项时，Zabbix 会在索引的基础 OID 下检索并缓存整个 SNMP 表，即使更早就能找到匹配项也是如此。
这样做是为了防止之后其他监控项引用相同的基础 OID——Zabbix 将在缓存中查找索引，而不是再次查询被监控主机。
请注意，每个 poller 进程都使用独立的缓存。

在之后所有的值获取操作中，只会验证已找到的索引。
如果索引未发生变化，则请求该值。
如果索引已发生变化，则会重建缓存——每个遇到索引变化的 poller 都会再次遍历索引 SNMP 表。

特殊 OID

Zabbix 会自动将一些最常用的 SNMP OID 转换为数值表示。例如，**ifIndex** 会被转换为 **1.3.6.1.2.1.2.2.1.1**，**ifIndex.0** 会被转换为 **1.3.6.1.2.1.2.2.1.1.0**。

下表列出了特殊 OID。

Special OID	Identifier	Description
ifIndex	1.3.6.1.2.1.2.2.1.1	每个接口的唯一值。
ifDescr	1.3.6.1.2.1.2.2.1.2	包含接口信息的文本字符串。该字符串应包括制造商名称、产品名称以及硬件接口版本。
ifType	1.3.6.1.2.1.2.2.1.3	接口类型，根据协议栈中紧邻网络层“下方”的物理/链路协议进行区分。
ifMtu	1.3.6.1.2.1.2.2.1.4	接口上可发送/接收的最大数据报大小，以八位组为单位。
ifSpeed	1.3.6.1.2.1.2.2.1.5	接口当前带宽的估计值，单位为比特/秒。
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	协议栈中紧邻网络层“下方”的协议层上的接口地址。
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	接口当前的管理状态。
ifOperStatus	1.3.6.1.2.1.2.2.1.8	接口当前的运行状态。
ifInOctets	1.3.6.1.2.1.2.2.1.10	接口接收的八位组总数，包括帧定界字符。
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	传递给更高层协议的子网单播数据包数量。
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12	传递给更高层协议的非单播（即子网广播或子网多播）数据包数量。
ifInDiscards	1.3.6.1.2.1.2.2.1.13	入站数据包中被选择丢弃的数量，即使未检测到错误，也会为了防止其被传递给更高层协议而丢弃。丢弃此类数据包的一个可能原因是释放缓冲区空间。
ifInErrors	1.3.6.1.2.1.2.2.1.14	入站数据包中包含错误、导致其无法被传递给更高层协议的数量。
ifInUnknownProtos	1.3.6.1.2.1.2.2.1.15	通过接口接收但因协议未知或不受支持而被丢弃的数据包数量。
ifOutOctets	1.3.6.1.2.1.2.2.1.16	从接口发送出去的八位组总数，包括帧定界字符。
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	更高层协议请求发送的、在该子层未被寻址到多播或广播地址的总数据包数，包括被丢弃或未发送的数据包。
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18	更高层协议请求发送的、在该子层被寻址到多播或广播地址的总数据包数，包括被丢弃或未发送的数据包。
ifOutDiscards	1.3.6.1.2.1.2.2.1.19	出站数据包中被选择丢弃的数量，即使未检测到错误，也会为了防止其被发送而丢弃。丢弃此类数据包的一个可能原因是释放缓冲区空间。
ifOutErrors	1.3.6.1.2.1.2.2.1.20	由于错误而无法发送的出站数据包数量。
ifOutQLen	1.3.6.1.2.1.2.2.1.21	输出数据包队列的长度（以数据包计）。

MIB 文件

简介

MIB 是 Management Information Base（管理信息库）的缩写。MIB 文件支持使用 OID（对象标识符）的文本表示形式。使用 Zabbix 监控 SNMP 设备时，可以使用原始 OID；但如果您更习惯使用文本表示形式，则需要安装 MIB 文件。

例如，

```
ifHCOutOctets
```

是以下 OID 的文本表示形式：

```
1.3.6.1.2.1.31.1.1.1.10
```

安装 MIB 文件

在基于 Debian 的系统上：

```
apt install snmp-mibs-downloader
download-mibs
```

在基于 RedHat 的系统上：

```
dnf install net-snmp-libs
```

启用 MIB 文件

在基于 RedHat 的系统上，默认情况下应该启用 mib 文件。在基于 Debian 的系统，您必须编辑文件 `/etc/snmp/snmp.conf` 和注释掉 `mibs :` 行

```
# 由于许可原因，snmp 软件包没有 MIB 文件，默认情况下禁用 MIB 加载。如果您添加了 MIB，您可以重新启用
# 通过注释掉以下行来加载它们。
mibs：
```

测试 MIB 文件

可以使用 snmpwalk 工具测试 SNMP MIB。如果尚未安装，请使用以下说明。

在基于 Debian 的系统上：

```
apt install snmp
```

在基于 RedHat 的系统上：

```
dnf install net-snmp-utils
```

之后，当您查询网络设备时，以下命令不应报错：

```
$ snmpwalk -v 2c -c public <NETWORK DEVICE IP> ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 176137634
IF-MIB::ifInOctets.2 = Counter32: 0
IF-MIB::ifInOctets.3 = Counter32: 240375057
IF-MIB::ifInOctets.4 = Counter32: 220893420
[...]
```

在 Zabbix 中使用 MIB

最需要牢记的一点是，Zabbix 进程不会获知对 MIB 文件所做的更改。因此，每次更改后，您都必须重启 Zabbix 服务器或 proxy，例如：

```
systemctl restart zabbix-server
```

之后，对 MIB 文件所做的更改才会生效。

使用自定义 MIB 文件

每个 GNU/Linux 发行版都有标准的 MIB 文件。但是一些设备供应商提供他们自己的。

假设您想使用CISCO-SMI MIB 文件。这以下说明将下载并安装它：

```
# wget ftp://ftp.cisco.com/pub/mibs/v2/CISCO-SMI.my -P /tmp
# mkdir -p /usr/local/share/snmp/mibs
# grep -q '^mibdirs +/usr/local/share/snmp/mibs' /etc/snmp/snmp.conf 2>/dev/null || echo "mibdirs +/usr/lo
# cp /tmp/CISCO-SMI.my /usr/local/share/snmp/mibs
```

现在你应该可以使用它了。试着翻译一下名字对象 *ciscoProducts* 从 MIB 文件到 OID：

```
# snmptranslate -IR -On CISCO-SMI::ciscoProducts
.1.3.6.1.4.1.9.1
```

如果您收到错误而不是 OID，请确保之前的所有命令没有返回任何错误。

对象名称翻译成功，您可以使用自定义 MIB 文件。请注意查询中使用的 MIB 名称前缀 (*CISCO-SMI::*)。你在使用命令行工具以及 Zabbix 时将需要这个。

在 Zabbix 中使用此 MIB 文件之前，请不要忘记重新启动 Zabbix 服务器/代理。

> **Attention:**
> 请记住，MIB 文件可以具有依赖关系。也就是说，一个 MIB 可能需要另一个 MIB。为了满足这些您必须安装所有受影响的 MIB 的依赖项文件。

SNMP trap

概述

接收 SNMP trap 与查询启用 SNMP 的设备正好相反。

在这种情况下，信息从启用 SNMP 的设备发送到 snmptrapd，并由 Zabbix 服务器或 Zabbix proxy 从文件中收集或 “捕获”。

通常，trap 会在某些条件发生变化时发送，agent 通过 162 端口连接到服务器（这与 agent 侧用于查询的 161 端口相对）。使用 trap 可以检测在查询间隔期间发生的一些短暂问题，而这些问题可能会被查询数据遗漏。

Zabbix 中接收 SNMP trap 的设计是与 snmptrapd 以及将 trap 传递给 Zabbix 的一种机制配合工作的 - 该机制可以是 Bash 或 Perl 脚本，也可以是 SNMPTT。

Note:
在配置好 Zabbix 后，设置 trap 监控的最简单方法是使用 Bash 脚本方案，因为现代发行版中通常缺少 Perl 和 SNMPTT，而且它们需要更复杂的配置。不过，这种方案使用的是配置为 `traphandle` 的脚本。为了在生产系统上获得更好的性能，请使用内嵌 Perl 方案（使用 `do perl` 选项的脚本，或 SNMPTT）。

接收 trap 的工作流程：

1. `snmptrapd` 接收一个 trap。
2. `snmptrapd` 将 trap 传递给接收脚本（Bash、Perl）或 SNMPTT。
3. 接收端解析、格式化并将 trap 写入文件。
4. Zabbix SNMP trapper 读取并解析 trap 文件。
5. 对于每个 trap，Zabbix 会查找所有 SNMP trapper 监控项，并匹配其主机接口与接收到的 trap 地址。请注意，匹配时只使用主机接口中选定的 *IP* 或 *DNS*。
6. 对于找到的每个监控项，将 trap 与 `snmptrap[regexp]` 中的正则表达式进行比较。trap 会被设置为所有匹配到的监控项的值。如果没有找到匹配的监控项，并且存在 `snmptrap.fallback` 监控项，则会将 trap 设置为该监控项的值。
7. 如果 trap 未被设置为任何监控项的值，Zabbix 默认会记录未匹配的 trap。（可通过 *Administration* > *General* > *Other* 中的 *Log unmatched SNMP traps* 进行配置。）

######HA 故障转移注意事项

在高可用性（HA）节点切换期间，Zabbix 将在最后一个 ISO 8601 时间戳内的最后一条记录之后继续处理；如果未找到相同的记录，则仅使用时间戳来识别最后位置。

配置 SNMP trap

此监控项类型需要以下前端配置：

步骤 1：为主机创建 SNMP 接口

1. 在 数据采集 > 主机中，创建主机 或编辑现有主机。
2. 在 接口字段中，选择 *SNMP* 接口类型。
3. 输入 IP 地址/DNS 名称和端口号。
4. 从下拉列表中选择 *SNMP* 版本：(*SNMPv1*、*SNMPv2*，或*SNMPv3*)，并根据所选 SNMP 版本添加接口凭据。

每个接收到的 trap 中的地址都会与所有 SNMP 接口的 IP 和 DNS 地址进行比较，以找到对应的主机。

步骤 2：配置监控项

1. 对于该主机，创建监控项 或编辑现有监控项。
2. 在 *Key* 字段中，使用以下 SNMP trap key 之一：
    - `snmptrap[regexp]` - 捕获所有与 `regexp` 参数中指定的正则表达式匹配的 SNMP trap；如果未指定 `regexp`，则捕获任意 trap。
 该参数支持用户宏和全局正则表达式。
 返回值：SNMP trap。
 此监控项只能为 SNMP 接口设置。
    - `snmptrap.fallback` - 捕获该接口上未被任何 `snmptrap[]` 监控项捕获的所有 SNMP trap。
 返回值：SNMP trap。
 此监控项只能为 SNMP 接口设置。

Note:
目前不支持多行正则表达式匹配。

3. 将 信息类型设置为 日志，以便解析时间戳。其他格式（例如 数值）也可以接受，但可能需要自定义 trap 处理程序。

设置 SNMP trap 监控

配置 Zabbix 服务器/proxy

要读取 trap，必须配置 Zabbix 服务器或 proxy，以启动 SNMP trapper 进程，并指向由 SNMPTT 或 Bash/Perl trap 接收器写入的 trap 文件。为此，请编辑配置文件（`zabbix_server.conf` 或`zabbix_proxy.conf`）：

```
StartSNMPTrapper=1
SNMPTrapperFile=[TRAP FILE]
```

Warning:
如果使用了 systemd 参数 `PrivateTmp`，则此文件很可能无法在 `/tmp` 中正常工作。

配置 Bash trap 接收器

要求：仅需 snmptrapd。

可以使用 Bash trap 接收器 脚本，通过 trapper 文件将 trap 从 snmptrapd 传递到 Zabbix 服务器。要进行配置，请在 snmptrapd 配置文件（snmptrapd.conf）中添加 traphandle 选项，参见示例。

> **Note:**
> 可能需要重启 snmptrapd 才能使其加载配置更改。

配置 Perl trap 接收器

要求：Perl、使用 --enable-embedded-perl 编译的 Net-SNMP（自 Net-SNMP 5.4 起默认启用）

可以使用 Perl trap 接收器（请查看 misc/snmptrap/zabbix_trap_receiver.pl），将 trap 直接从 snmptrapd 传递到 Zabbix 服务器。配置方法如下：

- 将 Perl 脚本添加到 snmptrapd 配置文件（snmptrapd.conf）中，例如：

```
perl do "[FULL PATH TO PERL RECEIVER SCRIPT]";
```

- 配置接收器，例如：

```
$SNMPTrapperFile = '[TRAP FILE]';
$DateTimeFormat = '[DATE TIME FORMAT]';
```

> **Note:**
> 可能需要重启 snmptrapd 才能使其加载配置更改。

> **Note:**
> 如果脚本名称没有加引号，snmptrapd 将拒绝启动，并显示类似以下的消息：

>
> ```
> Regexp modifiers "/l" and "/a" are mutually exclusive at (eval 2) line 1, at end of line
> Regexp modifier "/l" may not appear twice at (eval 2) line 1, at end of line
> ```

配置 SNMPTT

首先，应配置 snmptrapd 以使用 SNMPTT。

> **Note:**
> 为了获得最佳性能，应将 SNMPTT 配置为守护进程，并使用 snmptthandler-embedded 将 trap 传递给它。请参阅 配置 SNMPTT 的说明。

当 SNMPTT 配置为接收 trap 后，请配置 snmptt.ini：

1. 启用来自 NET-SNMP 包的 Perl 模块：

```
net_snmp_perl_enable = 1
```

2. 将 trap 记录到 Zabbix 将读取的 trap 文件中：

```
log_enable = 1
log_file = [TRAP FILE]
```

3. 设置日期时间格式：

```
date_time_format = %Y-%m-%dT%H:%M:%S%z
```

> **Warning:**
> net-snmp-perl 包已在 RHEL 8.0-8.2 中移除；在 RHEL 8.3 中重新添加。有关更多信息，请参阅已知问题。

现在请将 trap 格式化为 Zabbix 可识别的形式（编辑 snmptt.conf）：

1. 每个 FORMAT 语句都应以 ZBXTRAP [address] 开头，其中 [address] 将与 Zabbix 上 SNMP 接口的 IP 和 DNS 地址进行比较。例如：

```
EVENT coldStart .1.3.6.1.6.3.1.1.5.1 "Status Events" Normal
FORMAT ZBXTRAP $aA Device reinitialized (coldStart)
```

2. 有关 SNMP trap 格式的更多信息，请参见下文。

Attention:
不要使用未知 trap - Zabbix 将无法识别它们。可以通过在 snmptt.conf 中定义一个通用事件来处理未知 trap：


```
EVENT general .* "General event" Normal
```

SNMP trap 格式

所有自定义的 Perl trap 接收器和 SNMPTT trap 配置都必须按以下方式格式化 trap：

```
[timestamp] [the trap, part 1] ZBXTRAP [address] [the trap, part 2]
```

其中

- [timestamp] - 采用 %Y-%m-%dT%H:%M:%S%z 格式的时间戳。
- ZBXTRAP - 表示此行中开始了一个新 trap 的头部。
- [address] - 用于查找此 trap 所属主机的 IP 地址。

请注意，在处理过程中，ZBXTRAP 和 [address] 会从消息中被截取掉。如果 trap 的格式不同，Zabbix 可能会以意外的方式解析这些 trap。

示例 trap：

```
2024-01-11T15:28:47+0200 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" localhost - ZBXTRAP 192.168.1.1 Link
```

这将为 IP=192.168.1.1 的 SNMP 接口生成如下 trap：

```
2024-01-11T15:28:47+0200 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events"
localhost - Link down on interface 2. Admin state: 1. Operational state: 2
```

系统要求

Note:
建议安装 MIB 文件，以确保监控项值以正确的格式显示。如果没有 MIB 文件，可能会出现格式问题，例如以 HEX 而不是 UTF-8 显示值，或反之亦然。

大文件支持

Zabbix 支持 SNMP trapper 文件的大文件。Zabbix 可读取的最大文件大小为 2^63（8 EiB）。请注意，文件系统可能会对文件大小施加更低的限制。

日志轮转

Zabbix 不提供任何日志轮转系统 - 这应由用户自行处理。日志轮转应先重命名旧文件，之后再删除，以确保不会丢失任何 trap：

1. Zabbix 在上次已知位置打开 trap 文件，并转到步骤 3。
2. Zabbix 通过将 inode 编号与已定义 trap 文件的 inode 编号进行比较，检查当前打开的文件是否已轮转。如果没有打开的文件，Zabbix 会重置上次位置并转到步骤 1。
3. Zabbix 读取当前打开文件中的数据，并设置新位置。
4. 解析新数据。如果这是已轮转的文件，则关闭该文件并返回步骤 2。
5. 如果没有新数据，Zabbix 休眠 1 秒，然后返回步骤 2。

文件系统

由于 trap 文件的实现方式，Zabbix 需要文件系统支持 inode，以便区分文件（这些信息通过 stat() 调用获取）。

使用不同 SNMP 协议版本的设置示例

此示例使用 snmptrapd 和 Bash 接收脚本，将 trap 传递给 Zabbix 服务器。

设置：

1. 配置 Zabbix 启动 SNMP trapper 并设置 trap 文件。在 zabbix_server.conf 中添加：

```
StartSNMPTrapper=1
SNMPTrapperFile=/var/lib/zabbix/snmptraps/snmptraps.log
```

2. 将 Bash 脚本下载到 /usr/sbin/zabbix_trap_handler.sh：

```
curl -o /usr/sbin/zabbix_trap_handler.sh https://raw.githubusercontent.com/zabbix/zabbix-docker/trunk/temp
```

如有必要，请调整脚本中的 ZABBIX_TRAPS_FILE 变量。若要使用默认值，请先创建父目录：

```
mkdir -p /var/lib/zabbix/snmptraps
```

3. 将以下内容添加到 snmtrapd.conf 中（请参考可正常工作的示例）：

```
traphandle default /bin/bash /usr/sbin/zabbix_trap_handler.sh
```

> **Note:**
> 可能需要重启 snmptrapd 才能加载其配置更改。

4. 创建一个 SNMP 监控项 TEST（请注意初始配置要求）：

   类型：*SNMP trap*
 信息类型：日志
 主机接口：*SNMP 127.0.0.1*
 键值：snmptrap["linkUp"]
 日志时间格式：yyyyMMdd.hhmmss

请注意，此处使用 ISO 8601 日期和时间格式。

5. 接下来，我们将根据所选的 SNMP 协议版本配置 snmptrapd，并使用 snmptrap 工具发送测试 trap。

SNMPv1、SNMPv2

SNMPv1 和 SNMPv2 协议依赖 “community string” 身份验证。在下面的示例中，我们将使用 “secret” 作为 community string。SNMP trap 发送方必须设置相同的值。

请注意，尽管 SNMPv2 仍广泛用于生产环境，但它不提供任何加密和真正的发送方身份验证。数据以明文形式发送，因此这些协议版本只能用于安全环境，例如专用网络，绝不能通过任何公共网络或第三方网络使用。

如今基本不再使用 SNMP 版本 1，因为它不支持 64 位计数器，并且被视为旧版协议。

要启用接收 SNMPv1 或 SNMPv2 trap，请将以下行添加到 snmptrapd.conf 中。将 secret 替换为 SNMP trap 发送方上配置的 SNMP community string：

```
authCommunity log,execute,net secret
```

接下来，我们可以使用 snmptrap 发送测试 trap。在此示例中，我们将使用常见的 “link up”OID：

```
snmptrap -v 2c -c secret localhost '' linkUp.0
```

SNMPv3

SNMPv3 解决了 SNMPv1/v2 的安全问题，并提供身份验证和加密功能。您可以使用 MD5 或多种 SHA 身份验证方法，以及 DES/多种 AES 作为密码。

通过将以下行添加到 snmptrapd.conf，配置 snmptrapd 接收 SNMPv3 trap：

```
##### 创建具有身份验证和隐私保护功能的 SNMPv3 USM 用户
createUser -e <engineID> <securityName> SHA <authKey> AES <privKey>
##### 允许该用户记录 trap 并执行 trap 处理程序脚本
authUser log,execute <securityName>
```

例如：

```
createUser -e 0x8000000001020304 traptest SHA myAuthPassword AES myPrivPassword
authUser log,execute traptest
```

> **Attention:**
> 请注意其中的 “execute” 关键字，它允许为此用户安全模型执行脚本。

要测试 SNMPv3 trap 配置，请使用以下命令发送测试 trap：

```
snmptrap -v 3 -n "" -a SHA -A myAuthPassword -x AES -X myPrivPassword -l authPriv -u traptest -e 0x8000000
```

> **Warning:**
> 如果希望使用 AES192 或 AES256 等强加密方法，请使用 5.8 或更高版本的 net-snmp。可能还需要使用 configure 选项重新编译：
> --enable-blumenthal-aes。
> 较旧版本的 net-snmp 不支持 AES192/AES256。另请参阅：Strong Authentication or Encryption。

验证

在这两个示例中，您将在 /var/lib/zabbix/snmptraps/snmptraps.log 中看到类似以下内容的行：

```
2024-01-30T10:04:23+0200 ZBXTRAP 127.0.0.1
UDP: [127.0.0.1]:56585->[127.0.0.1]:162
```

```
DISMAN-EVENT-MIB::sysUpTimeInstance = 2538834
SNMPv2-MIB::snmpTrapOID.0 = IF-MIB::linkUp.0
```

Zabbix 中监控项的值将为：

```
2024-01-30 10:04:23 2024-01-30 10:04:21

2024-01-30T10:04:21+0200 UDP: [127.0.0.1]:56585->[127.0.0.1]:162
DISMAN-EVENT-MIB::sysUpTimeInstance = 2538834
SNMPv2-MIB::snmpTrapOID.0 = IF-MIB::linkUp.0
```

Perl 示例：

```
2024-01-30T11:42:54+0200 ZBXTRAP 127.0.0.1
PDU INFO:
  receivedfrom                   UDP: [127.0.0.1]:58649->[127.0.0.1]:162
  notificationtype               TRAP
  version                        1
  community                      public
  errorstatus                    0
  transactionid                  1
  requestid                      2101882550
  messageid                      0
  errorindex                     0
VARBINDS:
  DISMAN-EVENT-MIB::sysUpTimeInstance type=67 value=Timeticks: (457671) 1:16:16.71
  SNMPv2-MIB::snmpTrapOID.0       type=6  value=OID: IF-MIB::linkUp.0
```

请参阅

- 关于 SNMP traps 的 Zabbix 博客文章
- 配置 snmptrapd（net-snmp 官方文档）
- 配置 snmptrapd 接收 SNMPv3 通知（net-snmp 官方文档）

Zabbix 内部

概述

内部检查可用于监控 Zabbix 的内部进程。换句话说，您可以监控 Zabbix 服务器或 Zabbix proxy 的运行情况。

内部检查的计算位置如下：

- 在 Zabbix 服务器上 - 如果主机由服务器监控
- 在 Zabbix proxy 上 - 如果主机由 proxy 监控

无论主机是否处于维护状态，内部检查都会由服务器或 proxy 进行处理。

要使用此监控项，请选择 **Zabbix internal** 监控项类型。

> **Note:**
> 内部检查由 Zabbix 轮询器处理。

性能

使用某些内部监控项可能会对性能产生负面影响。这些监控项包括：

- zabbix[host,,items]
- zabbix[host,,items_unsupported]
- zabbix[hosts]
- zabbix[items]
- zabbix[items_unsupported]
- zabbix[queue,,]
- zabbix[requiredperformance]
- zabbix[stats,,,queue,,]
- zabbix[triggers]

系统信息 和队列
前端部分也会受到影响。

支持的检查

监控项键以不含可自定义参数和其他信息的形式列出。
点击监控项键可查看完整详细信息。

监控项键	描述
zabbix[apm,]	APM 收集器接收的遥测数据统计信息。
zabbix[boottime]	Zabbix 服务器或 Zabbix proxy 进程的启动时间，以秒为单位。
zabbix[cluster,discovery,nodes]	发现高可用集群节点。
zabbix[connector_queue]	连接器队列中排队的值数量。
zabbix[discovery_queue]	发现队列中排队的网络检查数量。
zabbix[host,,items]	主机上已启用监控项（受支持和不受支持）的数量。
zabbix[host,,items_unsupported]	主机上已启用的不受支持监控项的数量。
zabbix[host,maintenance]	主机当前的维护状态。
zabbix[host,active_agent,available]	主机上 active agent 检查的可用性。
zabbix[host,discovery,interfaces]	Zabbix 前端中主机所有已配置接口的详细信息。
zabbix[host,,available]	主机上特定类型检查的主接口可用性。
zabbix[hosts]	受监控主机的数量。
zabbix[items]	已启用监控项（受支持和不受支持）的数量。
zabbix[items_unsupported]	不受支持监控项的数量。
zabbix[java,,]	有关 Zabbix Java 网关的信息。
zabbix[lld_queue]	低级别发现处理队列中排队的值数量。
zabbix[preprocessing]	预处理管理器接收的值的统计信息。
zabbix[preprocessing_queue]	预处理队列中排队的值数量。
zabbix[process,,,]	特定 Zabbix 进程或进程组（由 <type> 和 <mode> 标识）在 <state> 中所占用时间的百分比。
zabbix[proxy,,]	有关 Zabbix proxy 的信息。
zabbix[proxy,discovery]	Zabbix proxy 列表。
zabbix[proxy group,,available]	proxy 组中在线 proxy 的数量。
zabbix[proxy group,,pavailable]	proxy 组中在线 proxy 的百分比。
zabbix[proxy group,,proxies]	proxy 组中的 Zabbix proxy 列表。
zabbix[proxy group,,state]	proxy 组的状态。
zabbix[proxy group,discovery]	返回包含配置数据和实时数据的 proxy 组列表。
zabbix[proxy_buffer,buffer,]	返回 proxy 内存缓冲区的使用统计信息。
zabbix[proxy_buffer,state,changes]	返回自启动以来磁盘/内存缓冲区模式之间的状态变更次数。
zabbix[proxy_buffer,state,current]	返回当前工作状态，即新数据存储的位置。
zabbix[proxy_history]	proxy 历史记录表中等待发送到服务器的值数量。
zabbix[queue,,]	队列中延迟至少 <from> 秒但少于 <to> 秒的受监控监控项数量。
zabbix[rcache,,]	Zabbix 配置缓存的可用性统计信息。
zabbix[requiredperformance]	Zabbix 服务器或 Zabbix proxy 所需的性能，以预期每秒新增值数量表示。
zabbix[stats,,]	返回 Zabbix 服务器或 proxy 的内部指标。如果指定了 <ip> 和 <port>，则从远程实例获取指标；否则从本地实例获取。
zabbix[stats,,,queue,,]	返回 Zabbix 服务器或 proxy 的内部队列指标。如果指定了 <ip> 和 <port>，则从远程实例获取指标；否则从本地实例获取。
zabbix[tcache,,]	Zabbix 趋势函数缓存的有效性统计信息。
zabbix[triggers]	Zabbix 数据库中已启用触发器的数量，其中启用的主机上的所有监控项均已启用。
zabbix[uptime]	Zabbix 服务器或 proxy 进程的运行时间，以秒为单位。
zabbix[vcache,buffer,]	Zabbix 值缓存的可用性统计信息。
zabbix[vcache,cache,]	Zabbix 值缓存的有效性统计信息。
zabbix[version]	Zabbix 服务器或 proxy 的版本。
zabbix[vmware,buffer,]	Zabbix VMware 缓存的可用性统计信息。
zabbix[vps,written]	写入数据库的历史值总数。
zabbix[wcache,,]	Zabbix 写入缓存的统计信息和可用性。

监控项键值详情

- 不带尖括号的参数为必填项，必须按原样使用（例如，`zabbix[host,<type>,available]` 中的 “host” 和 “available”）。
- 带有尖括号 **< >** 的参数必须替换为有效值。如果参数具有默认值，则可以省略。

- 标记为 “not supported on proxy” 的监控项和监控项参数，其值只有在主机由服务器监控时才能获取。相反，标记为 “not supported on server” 的值，只有在主机由 proxy 监控时才能获取。

zabbix[apm,<type>]

 APM collector 接收的遥测数据统计信息：

- *requests* - APM collector 接收的请求数（计数器）：
  - *accepted* - 接受处理的请求数；
  - *dropped* - 被丢弃的请求数；
- *rows* - 按数据类型写入数据源的遥测数据行数（计数器）。

返回值：*JSON* 对象。

参数：

- **type** - *traces*、*metrics* 或 *logs*。如果未指定，则返回所有类型的统计信息。

注释：

- Zabbix 服务器不支持此监控项。

zabbix[apm] 的返回值示例：

```
{
  "requests": {
    "accepted": 15230,
    "dropped": 12
  },
  "rows": {
    "traces": 48710,
    "metrics_gauge": 31250,
    "metrics_sum": 18400,
    "metrics_histogram": 5120,
    "metrics_exponential_histogram": 640,
    "metrics_summary": 210,
    "logs": 22870
  }
}
```

zabbix[boottime]

 Zabbix 服务器或 Zabbix 代理进程的启动时间（以秒为单位）。
 返回值：整数。

zabbix[cluster,discovery,nodes]

 发现高可用集群节点。
 返回值：*JSON* 对象。

注释：

- 此监控项可用于低级别发现。
- 返回的字段大多与高可用节点对象的属性相匹配。
- 其他字段：
  - db_timestamp - 响应中包含的当前服务器时间（Unix 时间戳）
  - lastaccess_age - 自上次节点心跳以来经过的秒数（db_timestamp - lastaccess）
  - status - 节点状态：
    - 0 - standby
    - 1 - 手动停止
    - 2 - 不可用
    - 3 - active

返回值示例：

```
[
  {
    "id": "ckvupihk70001z8mkpw5cg0u3",
    "name": "zabbix-prod-01",
    "status": 3,
    "address": "10.0.4.12:10051",
    "port": 10051,
    "lastaccess": 1756115995,
    "db_timestamp": 1756116000,
```

```
    "lastaccess_age": 5
  },
  {
    "id": "ckvx2a9k70004b1nq2hz9d7f",
    "name": "zabbix-standby-02",
    "status": 0,
    "address": "10.0.4.13:10051",
    "port": 10051,
    "lastaccess": 1756115550,
    "db_timestamp": 1756116000,
    "lastaccess_age": 450
  },
  {
    "id": "ckw0bq3l70007y4r1a0m5kz8",
    "name": "zabbix-backup-eu1",
    "status": 1,
    "address": "backup.example.com:10051",
    "port": 10051,
    "lastaccess": 1756105080,
    "db_timestamp": 1756116000,
    "lastaccess_age": 10920
  }
]
```


zabbix[connector_queue]

 连接器队列中排队的值的数量。
 返回值：整数。

zabbix[discovery_queue]

 发现队列中排队的网络检查数量。
 返回值：整数。

zabbix[host,,items]

 主机上启用的监控项数（支持和不支持）。
 返回值：整数。

zabbix[host,,items_unsupported]

 主机上启用的不支持监控项的数量。
 返回值：整数。

zabbix[host,,maintenance]

 主机当前的维护状态。
 返回值：*0* - 正常状态；*1* - 采集数据的维护状态；*2* - 不采集数据的维护状态。

注释：

- 无论主机位于何处（在服务器上或在 proxy 上），该监控项始终由 Zabbix 服务器处理。
  proxy 不会在配置数据中收到该监控项。
- 第二个参数必须为空，并保留供将来使用。

zabbix[host,active_agent,available]

 检查主机上活动代理的可用性。
 返回值：*0* - 未知；*1* - 可用；*2* - 不可用。

zabbix[host,discovery,interfaces]

 Zabbix 前端中主机的所有已配置接口的详细信息。

返回值：*JSON* 对象。

注释：

- 此项可用于低级别自动发现；
- Zabbix 代理不支持此项。

zabbix[host,<type>,available]

 主机上特定检查类型的主接口可用性。
 返回值：*0* - 不可用；*1* - 可用；*2* - 未知。

参数：

- **type** - *agent*、*snmp*、*ipmi* 或 *jmx*。

注释：

- 监控项值根据与主机不可达/不可用相关的配置参数计算。

zabbix[hosts]

 受监控主机的数量。
 返回值：整数。

zabbix[items]

 启用的监控项数（支持和不支持）。
 返回值：整数。

zabbix[items_unsupported]

 不支持的监控项数。
 返回值：整数。

zabbix[java,,<param>]

 有关 Zabbix Java 网关的信息。
 返回值：*1* - 如果 <param> 为 *ping*；*Java gateway version* - 如果 <param> 为 *version*（例如："8.0.0"）。

参数：

- **param** - *ping* 或 *version*。

注释：

- 此监控项可用于使用 nodata() 触发器函数检查 Java 网关可用性。
- 第二个参数必须为空，并保留供将来使用。

zabbix[lld_queue]

 低级发现处理队列中已入队的值的数量。
 返回值：整数。

备注：

- 此监控项可用于监控低级发现处理队列的长度。

zabbix[preprocessing]

 预处理管理器接收到的值的统计信息：

- *queued* - 需要预处理的排队值的数量和大小（计数器）
- *direct* - 不需要预处理的排队值的数量和大小（计数器）
- *queue* - 已进入预处理队列的值的数量（与 zabbix[preprocessing_queue] 相同）

返回值：*JSON*。

返回值示例：

```
{"data":
    {
    "queued": {
        "count": 106,
        "size": 58620
    },
    "direct": {
        "count": 395,
        "size": 33843
    },
    "queue": 0
    }
}
```

zabbix[preprocessing_queue]

 预处理队列中已入队的值的数量。
 返回值：*Integer*。

注释：

- 此监控项可用于监控预处理队列长度。

zabbix[process,<type>,<mode>,<state>]

 特定 Zabbix 进程或进程组（由 <type> 和 <mode> 标识）在 <state> 状态下所占用的时间百分比。仅针对最近一分钟进行计算。
 返回值：*Float*。

参数：

- **type**：

  - 对于服务器进程：*agent poller*、*alert manager*、*alert syncer*、*alerter*、*availability manager*、*browser poller*、*configuration syncer*、*configuration syncer worker*、*connector manager*、*connector worker*、*discovery manager*、*discovery worker*、*escalator*、*ha manager*、*history poller*、*history syncer*、*housekeeper*、*http agent poller*、*http poller*、*icmp pinger*、*internal poller*、*ipmi manager*、*ipmi poller*、*java poller*、*lld manager*、*lld worker*、*odbc poller*、*poller*、*preprocessing manager*、*preprocessing worker*、*proxy group manager*、*proxy poller*、*self-monitoring*、*service manager*、*snmp poller*、*snmp trapper*、*task manager*、*timer*、*trapper*、*trigger housekeeper*、*unreachable poller*、*vmware collector*；
  - 对于proxy 进程：*agent poller*、*apm manager*、*apm worker*、*availability manager*、*browser poller*、*configuration syncer*、*data sender*、*discovery manager*、*discovery worker*、*history syncer*、*housekeeper*、*http agent poller*、*http poller*、*icmp pinger*、*internal poller*、*ipmi manager*、*ipmi poller*、*java poller*、*odbc poller*、*poller*、*preprocessing manager*、*preprocessing worker*、*self-monitoring*、*snmp poller*、*snmp trapper*、*task manager*、*trapper*、*unreachable poller*、*vmware collector*；
- **mode**：
  - *avg* - 给定类型的所有进程的平均值（默认）；
  - *count* - 返回给定进程类型的派生进程数量，不应指定 <state>；
  - *max* - 最大值；
  - *min* - 最小值；
  - *<process number>* - 进程编号（介于 1 和预派生实例数量之间；例如，如果正在运行 4 个 trapper，则该值介于 1 和 4 之间）；
- **state**：
  - *busy* - 进程处于忙碌状态，例如正在处理请求（默认）；
  - *idle* - 进程处于空闲状态，未执行任何操作。

注释：

- 如果 <mode> 是一个未运行的 Zabbix 进程编号（例如，运行 5 个 poller 时将 <mode> 指定为 6），则此监控项将变为不受支持。
- 最小值和最大值指单个进程的使用百分比。因此，如果一个包含 3 个 poller 的组中，各进程的使用百分比分别为 2、18 和 66，则 min 将返回 2，max 将返回 66。
- 进程会在共享内存中报告其正在执行的操作，而 self-monitoring 进程每秒汇总一次这些数据。状态变化（busy/idle）会在发生变化时进行记录 - 因此，变为忙碌状态的进程会记录为忙碌，并且在变为空闲之前不会更改或更新其状态。这样可以确保即使完全挂起的进程也会被正确记录为 100% 忙碌。
- 目前，“busy” 表示 “不处于休眠状态”，但未来可能会引入其他状态 - 等待锁、执行数据库查询等。请注意，如果异步 poller 已达到 MaxConcurrentChecksPerPoller 服务器/proxy 配置参数设置的限制，则会被视为忙碌。
- 在 Linux 和大多数其他系统上，分辨率为 1/100 秒。

示例：

```
zabbix[process,poller,avg,busy] # 最近一分钟内 poller 进程执行操作所用时间的平均值
zabbix[process,"icmp pinger",max,busy] # 最近一分钟内任一 ICMP pinger 进程执行操作所用时间的最大值
zabbix[process,"history syncer",2,busy] # 最近一分钟内第 2 个 history syncer 执行操作所用的时间
zabbix[process,trapper,count] # 当前正在运行的 trapper 进程数量
```

zabbix[proxy,<name>,<param>]

 Zabbix proxy 的相关信息。
 返回值：*Integer*。

参数：

- **name** - proxy 名称；
- **param**：
  - *lastaccess* - 从 proxy 接收到的最后一条心跳消息的时间戳；
  - *delay* - 已收集但尚未发送的值持续了多长时间；计算方式为 “proxy delay” +（“当前服务器时间” - “proxy lastaccess”），其中 “proxy delay” 是当前 proxy 时间与 proxy 上最早的未发送值的时间戳之间的差值。

注释：

- 监控由 proxy 或 proxy 组执行的主机不支持此监控项。
- 可以使用fuzzytime() 函数检查 proxy 的可用性。

示例：

```
zabbix[proxy,"Germany",lastaccess] # 从 "Germany" proxy 接收到的最后一条心跳消息的时间戳
```

zabbix[proxy,discovery]

 Zabbix proxy 列表，包括名称、模式、加密、压缩、版本、最后发现时间、主机数量、监控项数量、每秒所需值（vps）、版本状态（当前/过时/不受支持）、按监控项类型划分的超时时间、proxy 组名称（如果 proxy 属于某个组）以及状态（未知/离线/在线）。
 返回值：*JSON* 对象。

注释：

- proxy 或 proxy 组监控的主机不支持此监控项。

返回值示例：


```
[
  {
    "name":"Zabbix proxy",
    "passive":false,
    "unencrypted":true,
    "psk":false,
    "cert":false,
    "items":22,
    "compression":true,
    "version":"8.0.0beta2",
    "compatibility":1,
    "last_seen":1,
    "hosts":1,
    "requiredperformance":0.333333,
    "timeouts": {
        "zabbix_agent":3,
        "simple_check":3,
        "snmp_check":3,
        "external_check":3,
        "db_monitor":3,
        "http_agent":3,
        "ssh_agent":3,
        "telnet_agent":3,
        "script":3,
        "browser":60
    },
    "state":"online"
  }
]
```


zabbix[proxy group,<name>,available]

 proxy 组中在线 proxy 的数量。
 返回值：*Integer*。

参数：

- **name** - proxy 组名称。

注释：

- proxy 或 proxy 组监控的主机不支持此监控项。

zabbix[proxy group,<name>,pavailable]

 proxy 组中在线 proxy 的百分比。
 返回值：*Float*。

参数：

- **name** - proxy 组名称。

注释：

- proxy 或 proxy 组监控的主机不支持此监控项。

zabbix[proxy group,<name>,proxies]

 具有名称、模式、加密、压缩、版本、最后发现时间、主机数量、监控项数量、每秒所需值数 (vps)、版本状态（当前/过时/不受支持）、超时、proxy 组名称和状态（未知/离线/在线）的 proxy 组中的 Zabbix proxy 列表。
 返回值：*JSON*。

参数：

- **name** - proxy 组名称。

注释：

- proxy 或 proxy 组监控的主机不支持此监控项。

zabbix[proxy group,<name>,state]

 proxy 组的状态。
 返回值：*0* - 未知；*1* - 离线；*2* - 正在恢复；*3* - 在线；*4* - 性能下降。

参数：

- **name** - proxy 组名称。

注释：

- proxy 或 proxy 组监控的主机不支持此监控项。

zabbix[proxy group,discovery]

 返回包含配置数据和实时数据的 proxy 组列表。配置数据包括 proxy 组名称、故障转移延迟时间以及所需的最少在线 proxy 数量。实时数据包括 proxy 组状态（详情请参见注释）、在线 proxy 数量以及在线 proxy 百分比。
 返回值：*JSON*。

注释：

- proxy 或 proxy 组监控的主机不支持此监控项。
- 此监控项不会返回不属于任何组的 proxy。
- 如果 `failover_delay` 或 `min_online` 的值无效，则会报告特殊值 `-1` 以表示该情况。如果使用宏进行配置，且宏无法展开为有效值，则可能出现无效值。
- proxy 组状态以整数表示：*0* - 未知；*1* - 离线；*2* - 正在恢复；*3* - 在线；*4* - 性能下降。

返回值示例：

```
{
    "groups": [
       { "name": "Riga", "failover_delay": 60, "min_online": 1 },
       { "name": "Tokyo", "failover_delay": 60, "min_online": 2 },
       { "name": "Porto Alegre", "failover_delay": 60, "min_online": 3 }
    ],
    "details": {
        "Riga": { "state": 3, "available": 10, "pavailable": 20 },
        "Tokyo": { "state": 3, "available": 10, "pavailable": 20 },
        "Porto Alegre": { "state": 1, "available": 0, "pavailable": 0 }
    }
}
```

zabbix[proxy_buffer,buffer,<mode>]

 proxy 内存缓冲区使用情况统计。
 返回值：*Integer*（大小）；*Float*（百分比）。

参数：

- **mode**：
  - *total* - 缓冲区的总大小（可用于检查是否启用了内存缓冲区）；
  - *free* - 空闲缓冲区的大小；
  - *pfree* - 空闲缓冲区的百分比；
  - *used* - 已用缓冲区的大小；
  - *pused* - 已用缓冲区的百分比。

注释：

- 内存缓冲区被禁用时，返回 “Proxy memory buffer is disabled” 错误；
- 此监控项在 Zabbix 服务器上不受支持。

zabbix[proxy_buffer,state,changes]

 返回自启动以来磁盘/内存缓冲区模式之间的状态变化次数。
 返回值：整数；*0* - 内存缓冲区已禁用。

注释：

- 频繁的状态变化表明必须增加内存缓冲区的大小或使用期限；
- 如果不频繁监控内存缓冲区状态（例如，每分钟一次），则缓冲区可能会在未注册的情况下翻转其状态。

zabbix[proxy_buffer,state,current]

 返回当前工作状态，新数据将存储在该状态中。
 返回值：*0* - 磁盘；*1* - 内存。

注释：

- 禁用内存缓冲区时也会返回 0。

zabbix[proxy_history]

 proxy 历史记录表中等待发送到服务器的值的数量。
 返回值：*Integer*。

注释：

- Zabbix 服务器不支持此监控项。

zabbix[queue,<from>,<to>]

 队列中监控项的数量，这些监控项的延迟至少为 <from> 秒，但少于 <to> 秒。
 返回值：*Integer*。

参数：

- **from** - 延迟至少为（默认值为 6 秒）；
- **to** - 延迟至多为（默认值为无穷大）。

注释：

- 参数中支持时间后缀（s,m,h,d,w）。

Attention:
Zabbix proxy 不会感知维护期；有关详细信息，请参见维护期间队列的计算。

zabbix[rcache,<cache>,<mode>]

 Zabbix 配置缓存的可用性统计信息。
 返回值：*Integer*（表示大小）；*Float*（表示百分比）。

参数：

- **cache** - *buffer*；
- **mode**：
  - *total* - buffer 的总大小；
  - *free* - 空闲 buffer 的大小；
  - *pfree* - 空闲 buffer 的百分比；
  - *used* - 已使用 buffer 的大小；
  - *pused* - 已使用 buffer 的百分比。

zabbix[requiredperformance]

 Zabbix 服务器或 Zabbix proxy 所需的性能，即预期每秒新增值的数量。
 返回值：*Float*。

注释：

- 大致对应于 *Reports* > *System information* 中的 “Required server performance, new values per second”。

zabbix[stats,<ip>,<port>]

 返回 Zabbix 服务器或 proxy 的内部指标。如果指定了 <ip> 和 <port>，则从远程实例获取指标；否则，从本地实例获取。
 返回值：*JSON* 对象。

参数：

- **ip** - 要远程查询的服务器/proxy 的 IP/DNS/网络掩码列表（默认值为 127.0.0.1）；
- **port** - 要远程查询的服务器/proxy 的端口（默认值为 10051）。

注释：

- stats 请求仅会从目标实例上 StatsAllowedIP 服务器/proxy 参数中列出的地址接受。
- 此监控项返回一组选定的内部指标。有关详细信息，请参见Zabbix stats 的远程监控。

zabbix[stats,<ip>,<port>,queue,<from>,<to>]

 返回 Zabbix 服务器或 proxy 的内部队列指标（参见 zabbix[queue,<from>,<to>]）。如果指定了 <ip> 和 <port>，则从远程实例获取指标；否则，从本地实例获取。
 返回值：*JSON* 对象。

参数：

- **ip** - 要远程查询的服务器/proxy 的 IP/DNS/网络掩码列表（默认值为 127.0.0.1）；
- **port** - 要远程查询的服务器/proxy 端口（默认值为 10051）；
- **from** - 至少延迟（默认值为 6 秒）；
- **to** - 最多延迟（默认值为无限）。

注释：

- 仅当请求来自目标实例上 StatsAllowedIP 服务器/proxy 参数中列出的地址时，才会接受 stats 请求。
- 此监控项返回一组选定的内部指标。有关详细信息，请参见Zabbix stats 的远程监控。

Attention:
Zabbix proxy 不感知维护期；有关详细信息，请参见维护期间队列的计算。

zabbix[tcache,cache,<parameter>]

 Zabbix 趋势函数缓存的有效性统计信息。
 返回值：*Integer*（表示大小）；*Float*（表示百分比）。

参数：

- **parameter**：
  - *all* - 总缓存请求数（默认）；
  - *hits* - 缓存命中数；
  - *phits* - 缓存命中百分比；
  - *misses* - 缓存未命中数；
  - *pmisses* - 缓存未命中百分比；
  - *items* - 缓存项数量；
  - *requests* - 缓存请求数量；
  - *pitems* - 缓存项占缓存项 + 请求的百分比。较低的百分比很可能表示可以减小缓存大小。
  - *paccessed* - 最近 24 小时内被访问条目的百分比。

注释：

- Zabbix proxy 不支持此监控项。

zabbix[triggers]

 Zabbix 数据库中已启用触发器的数量，且这些触发器所在的已启用主机上的所有监控项也均已启用。
 返回值：*Integer*。

说明：

- 此监控项在 Zabbix proxy 上不受支持。

zabbix[uptime]

 Zabbix 服务器或代理进程的正常运行时间（以秒为单位）。
 返回值：整数。

zabbix[vcache,buffer,<mode>]

 Zabbix 值缓存的可用性统计信息。
 返回值：*Integer*（表示大小）；*Float*（表示百分比）。

参数：

- **mode**：
  - *total* - 缓冲区总大小；
  - *free* - 空闲缓冲区大小；
  - *pfree* - 空闲缓冲区所占百分比；
  - *used* - 已使用缓冲区大小；
  - *pused* - 已使用缓冲区所占百分比。

注释：

- 此监控项不支持 Zabbix proxy。

zabbix[vcache,cache,<parameter>]

 Zabbix 值缓存的有效性统计信息。
 返回值：*Integer*。如果 <parameter> 设置为 *mode*，则返回值为：*0* - 正常模式；*1* - 低内存模式。

参数：

- **parameter**：
  - *requests* - 请求总数；
  - *hits* - 缓存命中次数（从缓存中获取的历史值）；
  - *misses* - 缓存未命中次数（从数据库中获取的历史值）；
  - *mode* - 值缓存的运行模式。

注释：

- 一旦切换到低内存模式，值缓存将保持该状态 24 小时，即使触发此模式的问题更早得到解决也是如此。
- 你可以将此键与 *Change per second* 预处理步骤一起使用，以获取每秒值统计信息。
- Zabbix proxy 不支持此监控项。

zabbix[version]

 Zabbix 服务器或 proxy 的版本。
 返回值：*String*。例如：`8.0.0`。

zabbix[vmware,buffer,<mode>]

 Zabbix vmware 缓存的可用性统计信息。
 返回值：*Integer*（表示大小）；*Float*（表示百分比）。

参数：

- **mode**：
  - *total* - 缓冲区总大小；
  - *free* - 空闲缓冲区大小；
  - *pfree* - 空闲缓冲区百分比；
  - *used* - 已使用缓冲区大小；
  - *pused* - 已使用缓冲区百分比。

zabbix[vps,written]

 写入数据库的历史值总数。
 返回值：整数。

zabbix[wcache,<cache>,<mode>]

 Zabbix 写缓存的统计信息和可用性。
 返回值：*Integer*（用于数量/大小）；*Float*（用于百分比）。

参数：

- **cache** - *values*、*history*、*index* 或 *trend*；
- **mode**：
  - 配合 *values* 缓存参数：
    - *all* - Zabbix 服务器/proxy 处理的值总数，不包括不受支持的监控项（计数器）（默认）；
    - *float* - 处理的浮点值数量（计数器）；
    - *uint* - 处理的无符号整数值数量（计数器）；
    - *str* - 处理的字符/字符串值数量（计数器）；
    - *log* - 处理的日志值数量（计数器）；
    - *text* - 处理的文本值数量（计数器）；
    - *bin* - 处理的二进制值数量（计数器）；
    - *json* - 处理的 JSON 值数量（计数器）；
    - *not supported* - 监控项处理后变为不受支持，或保持该状态的次数（计数器）；
  - 配合 *history*、*index*、*trend* 缓存参数：
    - *pfree* - 空闲缓冲区百分比（默认）；
    - *total* - 缓冲区总大小；
    - *free* - 空闲缓冲区大小；
    - *used* - 已用缓冲区大小；
    - *pused* - 已用缓冲区百分比。

注释：

- Zabbix proxy 不支持 `trend` 缓存参数。
- history 缓存用于存储监控项值。较低的数值表示数据库端存在性能问题。
- history index 缓存用于对存储在 history 缓存中的值建立索引。
- 在 history 缓存被填满并清空后，history index 缓存仍会保留部分数据。这是预期行为，通过避免持续调整内存大小所需的额外处理，有助于系统更高效地运行。
- trend 缓存存储所有接收数据的监控项在当前小时的聚合数据。
- 你可以将 `zabbix[wcache,values]` 键与 *Change per second* 预处理步骤一起使用，以获取每秒值统计信息。

Zabbix trapper

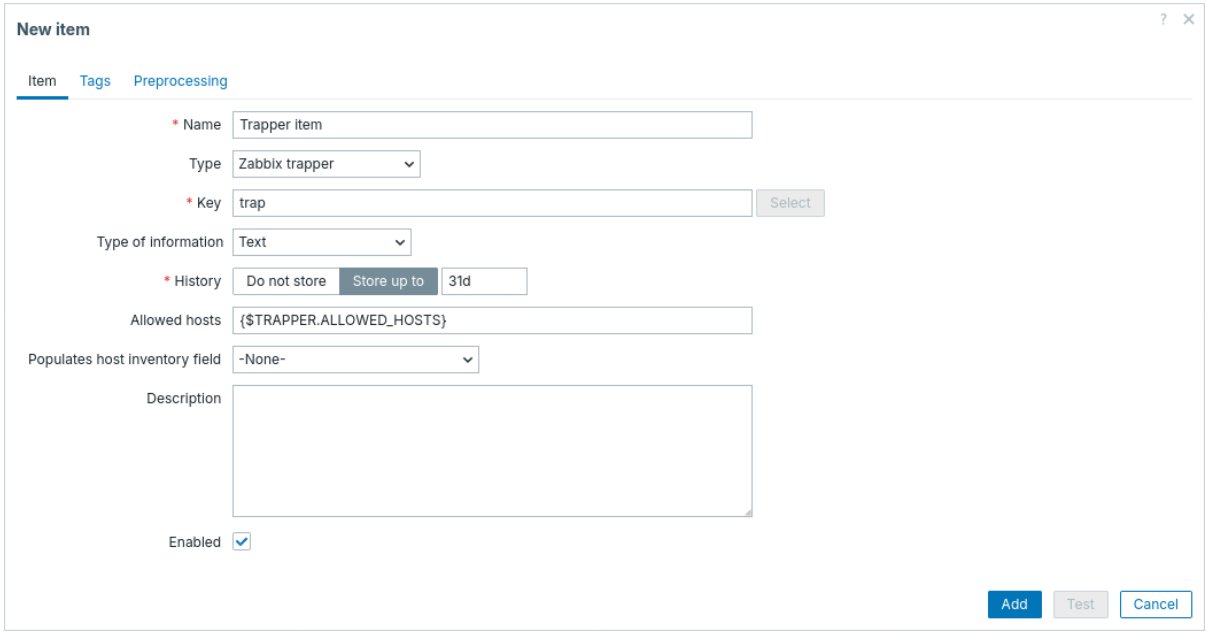
概述

Trapper 监控项接收传入的数据，而不是主动查询数据。这对于任何你想发送到 Zabbix 的数据都很有用。

配置

要配置 trapper 监控项：

1. 转到 数据采集 > 主机。
2. 点击该主机所在行中的 监控项。
3. 点击 创建监控项。
4. 在配置表单中输入监控项参数。



所有必填输入字段都用红色星号标记。

trapper 监控项需要填写的特定字段如下：

类型	选择”Zabbix trapper”。
键值	输入一个键值，用于在向 Zabbix 服务器发送数据时识别该监控项。
信息类型	选择与将要发送的数据格式相对应的信息类型。
允许的主机	以逗号分隔的 IP 地址列表（可选使用 CIDR 表示法）或 DNS 名称。 默认值为user macro {$TRAPPER.ALLOWED_HOSTS}。默认情况下，{$TRAPPER.ALLOWED_HOSTS} 为 127.0.0.1,::1。您可以直接在字段中输入地址，也可以修改宏值。如果留空，将拒绝来自所有主机的传入连接。如果指定了该字段，则只接受来自此处列出的主机的传入连接。 如果启用了 IPv6 支持，则’127.0.0.1’、’::127.0.0.1’、’::ffff:127.0.0.1’ 会被视为相同，且’::/0’ 将允许任何 IPv4 或 IPv6 地址。’0.0.0.0/0’ 可用于允许任何 IPv4 地址。 请注意，“IPv4-compatible IPv6 addresses”（0000::/96 前缀）虽受支持，但已被 RFC4291 弃用。 示例：127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, mysqlserver1, zabbix.example.com, {HOST.HOST} 支持空格、用户宏以及主机宏 {HOST.HOST}、{HOST.NAME}、{HOST.IP}、{HOST.DNS}、{HOST.CONN}。

Note:
在发送值之前，您可能需要在保存监控项后等待最多 60 秒，直到 Zabbix 服务器从配置缓存更新中获取到更改。

发送数据

可以使用Zabbix sender 实用程序或 Zabbix sender 协议 将数据发送到 Zabbix 服务器或 proxy。也可以使用history.push API 方法将数据发送到 Zabbix 服务器。

Zabbix sender

要使用 Zabbix sender 实用程序向 Zabbix 服务器或 proxy 发送数据，可以运行以下命令发送 “test value”：

```
zabbix_sender -z <server IP address> -p 10051 -s "New host" -k trap -o "test value"
```

要发送 “test value”，使用以下命令选项：

- -z 用于指定 Zabbix 服务器 IP 地址
- -p 用于指定 Zabbix 服务器端口号（默认是 10051）
- -s 用于指定主机（请确保使用技术名称，而不是可见的host name）
- -k 用于指定在 trapper 监控项中配置 的监控项键值

- -o 用于指定要发送的值

Attention:
Zabbix trapper 进程不会展开监控项键值中使用的宏，因此不会为目标主机检查相应的监控项键值是否存在。

有关 Zabbix sender 与 Zabbix 服务器或 proxy 之间通信的更多信息，请参见*Zabbix sender protocol*。

history.push

使用 API 方法 `history.push` 向 Zabbix server 发送数据时，您可以发起以下包含一些测试值的 HTTP POST 请求：:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer 0424bd59b807674191e7d77572075f33' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"history.push","params":[{"itemid":10600,"value":"test value 1"},{"ite
```

如果请求正确，API 返回的响应可能如下所示：:

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "data": [
            {
                "itemid": "10600"
            },
            {
                "itemid": "10601",
                "error": "Item is disabled."
            },
            {
                "error": "No permissions to referred object or it does not exist."
            }
        ]
    },
    "id": 1
}
```

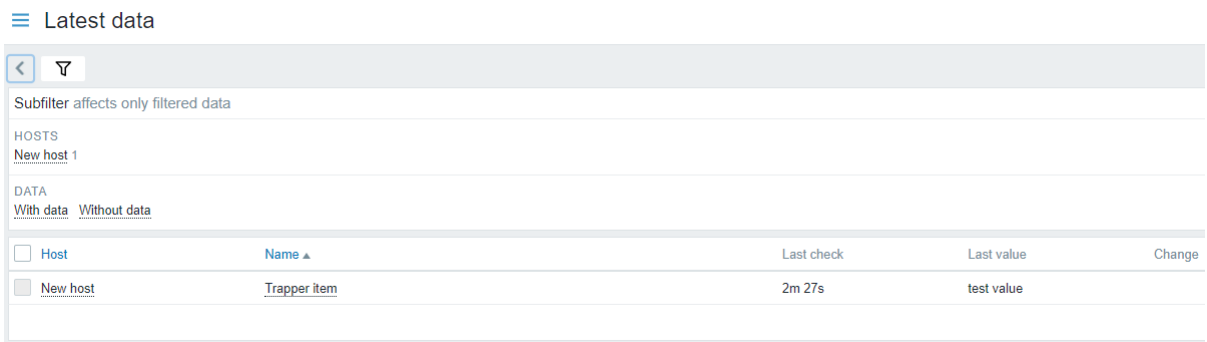
响应数据中的错误表明，特定监控项的数据发送未能通过 Zabbix server 的验证。这种情况可能由以下原因导致：

- 发送数据的用户没有对该监控项主机的 “读取” 权限;
- 主机被禁用或处于维护期且没有数据采集;
- 监控项不存在或尚未包含在服务器配置缓存中;
- 监控项被禁用或其类型不是 Zabbix trapper 或HTTP agent (启用 trapping);
- 用户的 IP 或 DNS 不在监控项的允许主机列表中;
- 另一个监控项有一个在纳秒级别上具有重复时间戳的值.

没有错误表明发送的值已被接受处理，这包括预处理（若配置预处理）、触发器处理以及保存到数据库中。请注意，即使值已被接受处理，处理过程也可能会失败（例，在预处理期间），导致该值被丢弃。关于获取更多如何使用 Zabbix API 信息，请参见API.

显示数据

数据发送后，您可以导航到 *Monitoring* > *Latest data* 查看结果：



Note:
如果发送的是单个数值，数据图形将在该值的时间点左右显示一条水平线。

外部检查

概述

外部检查是由 Zabbix 服务器通过运行 shell 脚本或二进制文件执行的检查。不过，当主机由 Zabbix proxy 监控时，外部检查由 proxy 执行。

外部检查不需要在被监控的主机上运行任何 agent。

监控项键值的语法为：

```
script[<parameter1>,<parameter2>,...]
```

其中：

ARGUMENT	DEFINITION
script	shell 脚本或二进制文件的名称。
parameter(s)	可选的命令行参数。

如果你不想向脚本传递任何参数，可以使用：

```
script[] or
script
```

Zabbix 服务器或 proxy 将搜索为外部脚本指定的目录并执行该命令（请参见 Zabbix 服务器/proxy 配置文件中的 `ExternalScripts` 参数）。该命令将以与 Zabbix 服务器/proxy 相同的用户身份执行，因此如有必要，任何访问权限或环境变量都应在包装脚本中处理。该命令的权限也应允许该用户执行它。只有指定目录中的命令才可执行。

Warning:
不要过度使用外部检查，因为每个脚本都需要由 Zabbix 服务器/proxy 启动一个 fork 进程，运行过多脚本会显著降低 Zabbix 性能。

使用示例

使用第一个参数 `'-h'` 执行脚本 `check_oracle.sh`。第二个参数将根据主机属性中的选择，替换为 IP 地址或 DNS 名称。

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

假设主机配置为使用 IP 地址，Zabbix 服务器/proxy 将执行：

```
check_oracle.sh '-h' '192.168.1.4'
```

外部检查结果

外部检查的返回值是检查产生的标准输出以及标准错误。

Attention:
返回文本（character、log 或 text 类型信息）的监控项在出现标准错误输出时不会变为不支持。

返回值限制为 16MB（包括会被截断的尾随空白字符）；数据库限制 同样适用。

如果找不到请求的脚本，或者 Zabbix 服务器/proxy 没有执行该脚本的权限，则该监控项将变为不支持，并显示相应的错误消息。

如果发生超时，该监控项将变为不支持，并显示相应的错误消息，同时为该脚本派生的进程将被终止。

数据库监控

概述

Zabbix 前端中的 *Database monitor* 监控项类型用于 ODBC 监控（ODBC 检查）。

ODBC 是一种用于访问数据库管理系统（DBMS）的 C 编程语言中间件 API。ODBC 概念由 Microsoft 开发，后来移植到其他平台。

Zabbix 可以查询任何受 ODBC 支持的数据库。为此，Zabbix 不会直接连接到数据库，而是使用 ODBC 接口以及在 ODBC 中配置的驱动程序。这使得可以更高效地对不同数据库进行多种用途的监控（例如，检查特定数据库队列、使用统计信息等）。

Zabbix 支持 unixODBC，它是最常用的开源 ODBC API 实现之一。

Attention:
另请参见：ODBC 检查的已知问题。

安装 unixODBC

推荐使用 Linux 操作系统默认的软件包仓库来安装 unixODBC。在最流行的 Linux 发行版中，unixODBC 默认已包含在软件包仓库中。如果没有可用的软件包，可以从 unixODBC 主页获取源文件：http://www.unixodbc.org/download.html。

要安装 unixODBC，请使用适用于你所选系统的软件包管理器：

```
##### 对于 Ubuntu/Debian 系统：
apt install unixodbc unixodbc-dev

##### 对于基于 RedHat/Fedora 的系统：
dnf install unixODBC unixODBC-devel

##### 对于基于 SUSE 的系统：
zypper in unixODBC-devel
```

Attention:
`unixodbc-dev` 或 `unixODBC-devel` 软件包是使用 unixODBC 支持编译 Zabbix 所必需的。要启用 ODBC 支持，Zabbix 应使用以下配置选项 进行编译：

`--with-unixodbc[=ARG] # 使用 unixODBC 软件包的 ODBC 驱动。`

安装 unixODBC 驱动程序

unixODBC 数据库驱动程序应该为将要监控的数据库安装。要查看支持的数据库和驱动程序列表，请访问 unixODBC 主页：: http://www.unixodbc.org/drivers.html.

Note:
在某些 Linux 发行版中，数据库驱动程序包含在软件包库中。

MySQL

要安装 MySQL 的 unixODBC 数据库驱动程序，请使用您所选择的系统的软件包管理工具：

```
##### 对于 Ubuntu/Debian 系统:
apt install odbc-mariadb

##### 对于 RedHat/Fedora-based 系统:
dnf install mariadb-connector-odbc

##### 对于 SUSE-based 系统:
zypper install mariadb-connector-odbc
```

如果不使用包管理器安装数据库驱动程序，请参考 `mysql-connector-odbc`MySQL 文档，或 `mariadb-connector-odbc`MariaDB 文档 。

PostgreSQL

要安装 PostgreSQL 的 unixODBC 数据库驱动程序，请使用您所选择的系统的软件包管理工具:

```
##### 对于 Ubuntu/Debian 系统:
apt install odbc-postgresql

##### 对于 RedHat/Fedora-based 系统:
dnf install postgresql-odbc

##### 对于 SUSE-based 系统:
zypper install psqlODBC
```

如果不使用包管理器安装数据库驱动程序，请参考PostgreSQL 文档.

Oracle

要安装 unixODBC 数据库驱动程序，请参考 Oracle documentation.

MSSQL

要安装 MSSQL unixODBC 数据库驱动，请使用所选系统的软件包管理器：

```
##### 对于 Ubuntu/Debian 系统：
apt install tdsodbc

##### 对于基于 RedHat/Fedora 的系统（EPEL 软件包：https://docs.fedoraproject.org/en-US/epel/）：
dnf install epel-release
dnf install freetds

##### 对于基于 SUSE 的系统：
zypper install libtdsodbc0
```

如果不使用软件包管理器安装数据库驱动，请参阅 FreeTDS 用户指南。

配置 unixODBC

要对 unixODBC 进行配置，需编辑 `odbcinst.ini` 和 `odbc.ini` 文件。可通过执行如下命令来确认这些文件的位置：

```
odbcinst -j
```

命令的输出应该包含类似于以下内容信息：

```
unixODBC 2.3.9
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
```

odbcinst.ini

`odbcinst.ini` 文件列出已安装的 ODBC 数据库驱动程序。如果缺少 `odbcinst.ini` 文件，则需手动创建它。

```
[TEST_MYSQL]
Description=ODBC for MySQL
Driver=/usr/lib/libmyodbc5.so
FileUsage=1
```

参数	描述
TEST_MYSQL	数据库驱动名称.
Description	数据库驱动描述.
Driver	数据库驱动库位置.
FileUsage	确定数据库驱动是否支持在没有访问本地文件支持的情况下连接到数据库服务器（0）；支持从文件读取数据（1）；支持将数据写入文件（2）。
Threading	线程串行化级别。PostgreSQL 支持此功能。 自 1.6 版本起，如果驱动管理器构建时包含了线程支持，您可能可以添加另一个驱动程序条目。

odbc.ini

`odbc.ini` 文件用于配置数据源。请注意，支持的参数列表取决于数据库驱动程序（例如，Oracle 数据库可能使用 ServerName 而不是 Server，等等）。

```
[TEST_MYSQL]
Description=MySQL Test Database
Driver=mysql
Server=127.0.0.1
User=root
Password=
Port=3306
Socket=
Database=zabbix
```

Parameter	Description
TEST_MYSQL	数据源名称（DSN）。
Description	数据源描述。
Driver	数据库驱动程序名称（如 `odbcinst.ini` 中所指定）。
Server	数据库服务器 IP/DNS。

Parameter	Description
User	用于连接的数据库用户。
Password	数据库用户密码。
Port	数据库连接端口。
Socket	数据库连接套接字。
Database	数据库名称。

有关其他可能的配置参数选项，请参见 MySQL documentation。

PostgreSQL 的 odbc.ini 文件可能包含其他参数：

```
[TEST_PSQL]
Description=PostgreSQL Test Database
Driver=postgresql
Username=zbx_test
Password=zabbix
Servername=127.0.0.1
Database=zabbix
Port=5432
ReadOnly=No
Protocol=8.0+
ShowOidColumn=No
FakeOidIndex=No
RowVersioning=No
ShowSystemTables=No
Fetch=Yes
BoolsAsChar=Yes
SSLmode=Require
ConnSettings=
```

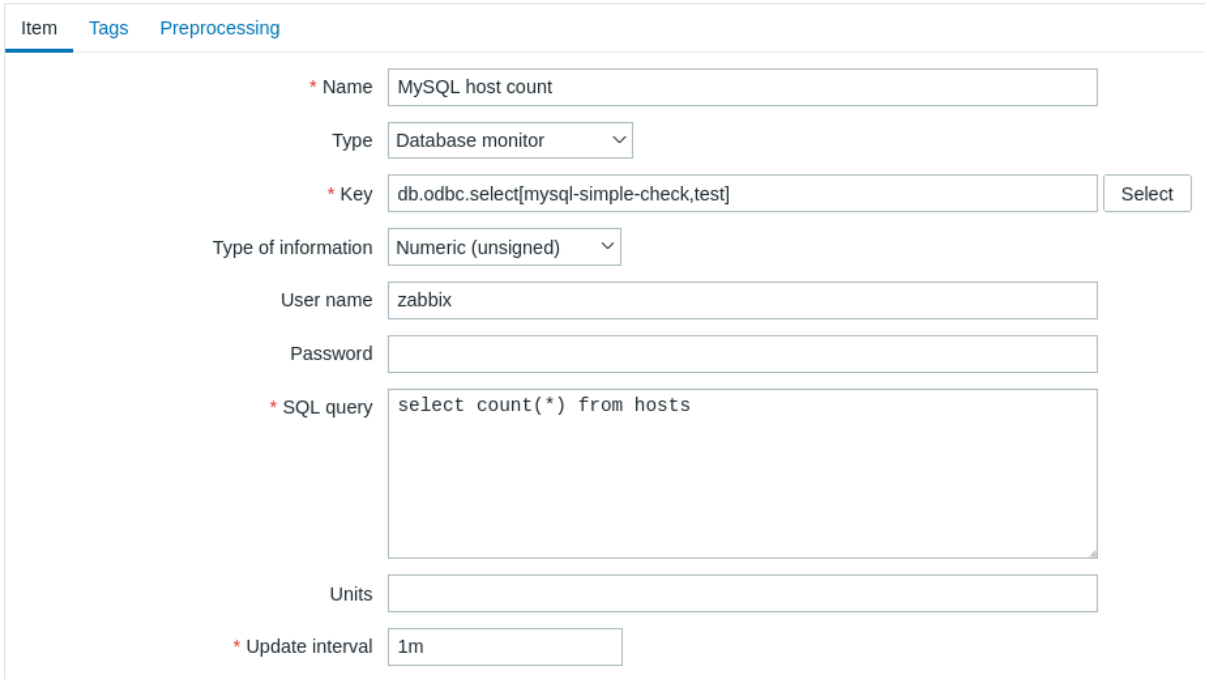
参数	描述
ReadOnly	指定数据库连接是否仅允许读取操作（SELECT 查询）并限制修改操作（INSERT、UPDATE 和 DELETE 语句）；适用于数据应保持不变的场景。
Protocol	PostgreSQL 后端协议版本（使用 SSL 连接时将被忽略）。
ShowOidColumn	指定是否在 SQLColumns 中包含对象 ID（OID）。
FakeOidIndex	指定是否在 OID 上创建一个伪唯一索引。
RowVersioning	指定是否启用应用程序在尝试更新某一行时检测数据是否已被其他用户修改。请注意，此参数可以加快更新过程，因为更新一行时，不需要在 WHERE 子句中指定每一个列。
ShowSystemTables	指定数据库驱动程序是否应在 SQLTables 中将系统表视为常规表；这有助于提高可访问性，使系统表可见。
Fetch	指定驱动程序是否应自动使用 declare cursor/fetch 来处理 SELECT 语句并维护一个包含 100 行的缓存。
BoolsAsChar	控制布尔类型的映射。 如果设置为 “Yes”，布尔值将映射为 SQL_CHAR；否则将映射为 SQL_BIT。
SSLmode	指定连接的 SSL 模式。
ConnSettings	连接时发送到后端的其他设置。

要检验 ODBC 连接是否成功建立，您可以使用 isql 工具（包含在 unixODBC 软件包内）：

```
isql test
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
```

Zabbix 前端中的监控项配置

配置一个 数据库监控监控项。



所有必填输入字段都以红色星号标记。

对于数据库监控监控项，您必须指定：

类型	在此选择 “Database monitor”。
键值	输入受支持的监控项键值之一： **db.odbc.select[]** - 此监控项返回一个值（SQL 查询结果第一行的第一列）； **db.odbc.get[]** - 此监控项以 JSON 格式返回多行/多列； **db.odbc.discovery[]** - 此监控项返回低级别发现数据。
用户名	输入数据库用户名（最多 255 个字符）。 如果数据库用户名已在 `odbc.ini` 文件中指定，则此参数为可选。 如果使用连接字符串，并且 用户名字段不为空，则会将其追加到连接字符串中，格式为 `UID=<user>`。
密码	输入数据库用户密码（最多 255 个字符）。 如果密码已在 `odbc.ini` 文件中指定，则此参数为可选。 如果使用连接字符串，并且 密码字段不为空，则会将其追加到连接字符串中，格式为 `PWD=<password>`。 此字段支持特殊字符。对于除 Oracle 之外的所有数据库，包含 % 的密码必须用花括号 {} 括起来。 密码将追加到连接字符串中用户名之后，例如：`UID=<username>;PWD=P?;)*word`。 要测试生成的字符串，可以运行以下命令： `isql -v -k 'Driver=libmaodbc.so;Database=zabbix;UID=zabbix;PWD=P?;)*word'`
SQL 查询	输入 SQL 查询。 请注意，对于 `db.odbc.select[]`，查询必须且只能返回一个值。
信息类型	在此选择查询将返回的信息类型。 如果信息类型选择错误，该监控项将变为不受支持。

重要说明

- 如果在服务器或 proxy 配置中未启动任何 *odbc poller* 进程，数据库监控项将变为不受支持。要启用 ODBC poller，请在 Zabbix 服务器 配置文件中设置 `StartODBCPollers` 参数；对于由 proxy 执行的检查，请在 Zabbix proxy 配置文件中设置该参数。
- 监控项配置 表单中的 *Timeout* 参数值将同时用作 ODBC 登录超时和查询执行超时。请注意，如果已安装的 ODBC 驱动不支持这些超时设置，则它们可能会被忽略。
- SQL 命令必须返回结果集，就像使用 `select` 语句的任何查询一样。查询语法将取决于处理它们的 RDBMS。对存储过程的请求语法必须以 `call` 关键字开头。

监控项键值详细信息

不带尖角括号的参数必填项。用尖角括号 **< >** 标记的参数是可选的。

db.odbc.select[< 唯一简短描述 >,< 配置 dsn>,< 连接字符串 >]

 返回一个值，即 SQL 查询结果中第一行的第一列。
 返回值：根据 SQL 查询而定。

参数:

- **unique short description** -一个唯一简短的描述来标识监控项（用于触发器等）；
- **dsn** - 数据源名称（如在 odbc.ini 中指定的）；
- **connection string** - 连接字符串（可能包含特定于驱动程序的参数）。

注释：

- dsn will be ignored. 尽管 dsn 和 connection string 是可选参数，但至少需要其中一个；如果两个都被定义了，dsn 将被忽略。
- 如果查询返回多个列，只有第一列会被读取。如果查询返回多行，只有第一行会被读取。

db.odbc.get[<unique short description>,<dsn>,<connection string>]

 将 SQL 查询结果转换为 JSON 数组。
 返回值：*JSON object*。

参数：

- **unique short description** - 用于标识监控项的唯一简短描述（供触发器等使用）；
- **dsn** - 数据源名称（按 odbc.ini 中的定义）；
- **connection string** - 连接字符串（可包含特定驱动程序的参数）。

注释：

- 尽管 dsn 和 connection string 是可选参数，但至少需要提供其中一个；如果两者都定义了，则会忽略 dsn。
- 可以返回 JSON 格式的多行/多列数据。此监控项可用作主监控项，在一次系统调用中收集所有数据，而在依赖监控项中可使用 JSONPath 预处理提取单个值。有关返回格式的更多信息，请参见低级别发现中使用的返回格式的示例。

示例：

```
# MySQL ODBC 驱动程序 5 的连接：
db.odbc.get[MySQL example,,"Driver=/usr/local/lib/libmyodbc5a.so;Database=master;Server=127.0.0.1;Port=330
```

db.odbc.discovery[< 唯一简短描述 >,< 配置 dsn>,< 连接字符串 >]

 将 SQL 查询结果转换为 JSON 数组，用于低级别自动发现. 查询结果中的列名被转换为与发现字段值配对的低级发现宏名称。这些宏可以在创建监控项、触发器等原型时使用。
 返回值：*JSON object*。

参数:

- **unique short description** - 一个唯一简短的描述来标识监控项（用于触发器等）。
- **dsn** - 数据源名称（如在 odbc.ini 文件中指定的）。
- **connection string** - 连接字符串（可能包含特定于驱动程序的参数）。

注释:

- 尽管 dsn 和 connection string 是可选参数，但至少需要其中一个；如果两个都被定义了，dsn 将被忽略。

错误消息

ODBC 错误消息按字段组织，以提供详细信息。例如，一条错误消息可能如下所示：

```
无法执行 ODBC 查询：[SQL_ERROR]:[42601][7][ERROR: syntax error error at or near ";"; Error while executing
```

- ”无法执行 ODBC 查询” - Zabbix 消息
- ”[SQL_ERROR]” - ODBC 返回代码
- ”[42601]” - SQLState
- ”[7]” - 本地错误代码
- ”[ERROR: syntax error at or near ";"; Error while executing the query]” - 本地错误消息

请注意，错误消息长度限制为 2048 字节，因此消息可能会被截断。如果存在多条 ODBC 诊断记录，Zabbix 会尝试将它们连接起来（使用 | 分隔），直到达到长度限制为止。

HTTP agent

概述

此监控项类型允许使用 HTTP/HTTPS 协议进行数据轮询。也可以使用Zabbix sender 工具或 Zabbix sender 协议 进行捕获（用于向 Zabbix 服务器或 proxy 发送数据），或者使用history.push API 方法（用于向 Zabbix 服务器发送数据）。

HTTP 监控项检查由 Zabbix 服务器执行。但是，当主机由 Zabbix proxy 监控时，HTTP 监控项检查由 proxy 执行。

HTTP 监控项检查不需要被监控主机上运行任何 agent。

HTTP agent 同时支持 HTTP 和 HTTPS。Zabbix 会按需跟随重定向（请参见下方的 跟随重定向选项）。重定向的最大数量硬编码为 10（使用 cURL 选项 CURLOPT_MAXREDIRS）。

Attention:
Zabbix 服务器/proxy 必须先配置好 cURL（libcurl）支持。

HTTP 检查以异步方式执行 - 在启动其他检查之前，不需要先接收某个请求的响应。DNS 解析同样是异步的。

异步检查的最大并发数为 1000（由MaxConcurrentChecksPerPoller 定义）。

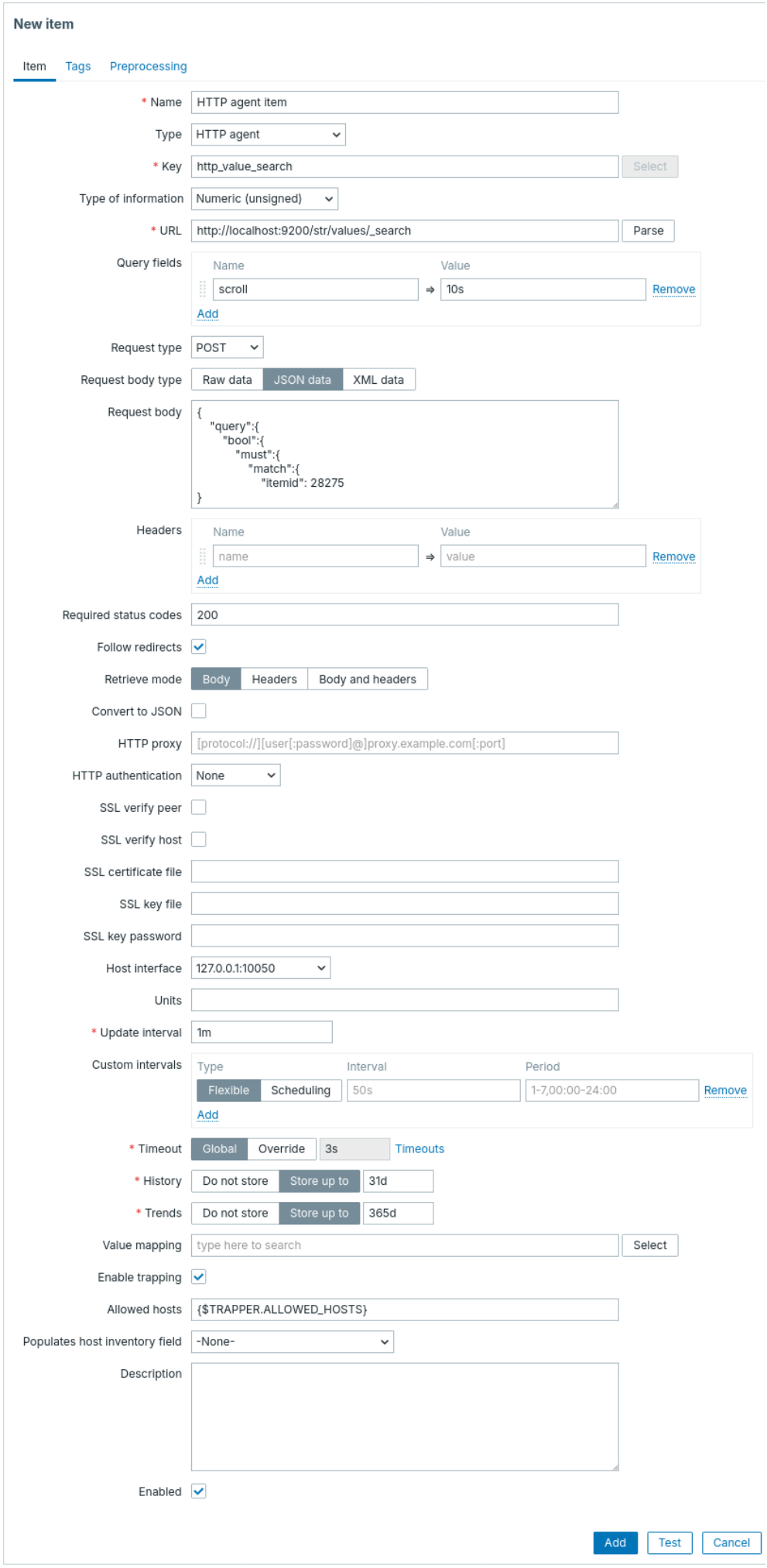
异步 HTTP agent poller 的数量由StartHTTPAgentPollers 参数定义。

自 Zabbix 7.0 起，HTTP agent 检查已加入 cURL 的持久连接功能。

配置

要配置 HTTP 监控项：

1. 转到 数据采集 > 主机。
2. 单击主机所在行中的 监控项。
3. 单击 创建监控项。
4. 在表单中输入监控项参数。


New item
Item
Tags
Preprocessing
* Name
HTTP agent item
Type
HTTP agent
* Key
http_value_search
Select
Type of information
Numeric (unsigned)
* URL
http://localhost:9200/str/values/_search
Parse
Query fields
Name
Value
scroll
10s
Remove
Add
Request type
POST
Request body type
Raw data
JSON data
XML data
Request body
{
"query":{
"bool":{
"must":{
"match":{
"itemid": 28275
}
Headers
Name
Value
name
value
Remove
Add
Required status codes
200
Follow redirects
Retrieve mode
Body
Headers
Body and headers
Convert to JSON
HTTP proxy
[protocol://][user[:password]@]proxy.example.com[:port]
HTTP authentication
None
SSL verify peer
SSL verify host
SSL certificate file
SSL key file
SSL key password
Host interface
127.0.0.1:10050
Units
* Update interval
1m
Custom intervals
Type
Interval
Period
Flexible
Scheduling
50s
1-7,00:00-24:00
Remove
Add
* Timeout
Global
Override
3s
Timeouts
* History
Do not store
Store up to
31d
* Trends
Do not store
Store up to
365d
Value mapping
type here to search
Select
Enable trapping
Allowed hosts
{$TRAPPER.ALLOWED_HOSTS}
Populates host inventory field
-None-
Description
Enabled
Add
Test
Cancel

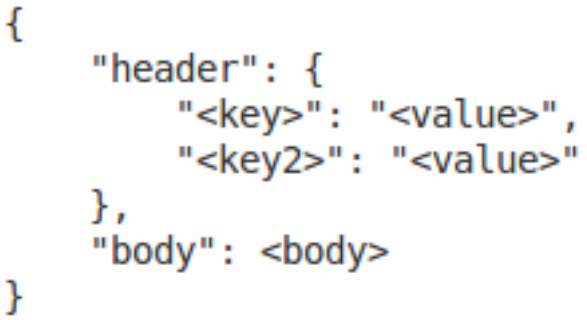
所有必填字段均以红色星号标记。

HTTP 监控项需要特定信息的字段如下：

参数	描述
类型	在此选择 **HTTP agent**。
键值	输入唯一的监控项键值。

URL	要连接并从中检索数据的 URL。例如：
	`https://www.example.com`
	`http://www.example.com/download`
	域名可以使用 Unicode 字符指定。在执行 HTTP 检查时，它们会自动转换为 ASCII 格式的 punycode。
	可以使用 解析按钮将可选查询字段（例如 `?key=value&limit=10`）与 URL 分离，并将属性和值移至 查询字段，以便自动进行 URL 编码。
	出于安全原因，请勿在 URL 中包含实际的用户名或密码。请改用用户宏，以避免凭据泄露。
	限制为 2048 个字符。
	支持的宏：`{HOST.IP}`、`{HOST.CONN}`、`{HOST.DNS}`、`{HOST.HOST}`、`{HOST.NAME}`、`{HOST.PORT}`、`{ITEM.ID}`、`{ITEM.KEY}`、`{ITEM.KEY.ORIG}`、用户宏、低级别发现宏。
	此项设置 CURLOPT_URL cURL 选项。

查询字段	URL 的变量（见上文）。 指定为属性和值对。 值会自动进行 URL 编码。宏的值会先解析，然后自动进行 URL 编码。 支持的宏：`{HOST.IP}`、`{HOST.CONN}`、`{HOST.DNS}`、`{HOST.HOST}`、`{HOST.NAME}`、`{HOST.PORT}`、`{ITEM.ID}`、`{ITEM.KEY}`、`{ITEM.KEY.ORIG}`、用户宏、低级别发现宏。 此项设置 CURLOPT_URL cURL 选项。
请求类型	选择请求方法类型：*GET*、*POST*、*PUT* 或 *HEAD*。
请求正文类型	选择请求正文类型： 原始数据 - 自定义 HTTP 请求正文，会替换宏，但不会执行编码。 **JSON** 数据 - JSON 格式的 HTTP 请求正文。宏可以用作字符串、数字、true 和 false；用作字符串的宏必须用双引号括起来。宏的值会先解析，然后自动进行转义。如果标头中未指定”Content-Type”，则默认为”Content-Type: application/json”。 **XML** 数据 - XML 格式的 HTTP 请求正文。宏可以用作文本节点、属性或 CDATA 节。文本节点和属性中的宏值会先解析，然后自动进行转义。如果标头中未指定”Content-Type”，则默认为”Content-Type: application/xml”。
请求正文	输入请求正文。 支持的宏：`{HOST.IP}`、`{HOST.CONN}`、`{HOST.DNS}`、`{HOST.HOST}`、`{HOST.NAME}`、`{HOST.PORT}`、`{ITEM.ID}`、`{ITEM.KEY}`、`{ITEM.KEY.ORIG}`、用户宏、低级别发现宏。
标头	执行请求时将发送的自定义 HTTP 标头。 指定为属性和值对。 支持的宏：`{HOST.IP}`、`{HOST.CONN}`、`{HOST.DNS}`、`{HOST.HOST}`、`{HOST.NAME}`、`{HOST.PORT}`、`{ITEM.ID}`、`{ITEM.KEY}`、`{ITEM.KEY.ORIG}`、用户宏、低级别发现宏。 此项设置 CURLOPT_HTTPHEADER cURL 选项。
所需状态码	预期 HTTP 状态码列表。如果 Zabbix 获取的状态码不在列表中，监控项将变为不支持。如果为空，则不执行检查。 例如：200、201、210-299。 列表中支持的宏：用户宏、低级别发现宏。 此项使用 CURLINFO_RESPONSE_CODE cURL 选项。
跟随重定向	选中复选框以跟随 HTTP 重定向。 此项设置 CURLOPT_FOLLOWLOCATION cURL 选项。
检索模式	选择必须检索的响应部分： 正文 - 仅正文 标头 - 仅标头 正文和标头 - 正文和标头

参数	描述
转换为 *JSON*	如果 检索模式未设置为 正文，则标头会作为属性和值对保存在"header" 键下。 如果遇到'Content-Type: application/json'，则正文会保存为对象；否则会保存为字符串，例如： 
HTTP proxy	可以使用格式 [protocol://][username[:password]@]proxy.example.com[:port] 指定要使用的 HTTP proxy。 可选的 protocol:// 前缀可用于指定其他 proxy 协议（例如 https、socks4、socks5；请参阅文档；cURL 7.21.7 添加了对协议前缀的支持）。如果未指定协议，proxy 将被视为 HTTP proxy。如果指定了错误的协议，连接将失败，监控项将变为不支持。 默认使用端口 1080。 如果指定了 proxy，它将覆盖 http_proxy、HTTPS_PROXY 等与 proxy 相关的环境变量。如果未指定 proxy，则不会覆盖与 proxy 相关的环境变量。输入的值将按 "原样" 传递，不会执行有效性检查。 请注意，HTTP proxy 仅支持简单身份验证。 支持的宏：{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG}、用户宏、低级别发现宏。 此项设置 CURLOPT_PROXY cURL 选项。
HTTP 身份验证	选择身份验证选项： 无 - 不使用身份验证； **Basic** - 使用基本身份验证； **NTLM** - 使用 NTLM（Windows NT LAN Manager）身份验证； **Kerberos** - 使用 Kerberos 身份验证（另请参阅：在 Zabbix 中配置 Kerberos）； **Digest** - 使用 Digest 身份验证。 此项设置 CURLOPT_HTTPAUTH cURL 选项。
用户名	输入用户名（最多 255 个字符）。 当 *HTTP* 身份验证设置为 *Basic*、*NTLM*、*Kerberos* 或 *Digest* 时，此字段可用。支持用户宏和低级别发现宏。
密码	输入用户密码（最多 255 个字符）。 当 *HTTP* 身份验证设置为 *Basic*、*NTLM*、*Kerberos* 或 *Digest* 时，此字段可用。支持用户宏和低级别发现宏。
SSL 验证对等方	选中复选框以验证 web 服务器的 SSL 证书。服务器证书会根据系统范围 CA 位置中的受信任证书颁发机构（CA）证书进行验证。可以使用 Zabbix 服务器或 proxy 配置参数 SSLCALocation 覆盖 CA 文件的位置。 此项设置 CURLOPT_SSL_VERIFYPEER cURL 选项。
SSL 验证主机	选中复选框以验证 web 服务器证书的 *Common Name* 字段或 *Subject Alternate Name* 字段是否匹配。 此项设置 CURLOPT_SSL_VERIFYHOST cURL 选项。
SSL 证书文件	用于客户端身份验证的 SSL 证书文件名称。证书文件必须为 PEM[1] 格式。如果证书文件还包含私钥，请将 *SSL* 密钥文件字段留空。如果密钥已加密，请在 *SSL* 密钥密码字段中指定密码。包含此文件的目录由 Zabbix 服务器或 proxy 配置参数 SSLCertLocation 指定。 支持的宏：{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG}、用户宏、低级别发现宏。 此项设置 CURLOPT_SSLCERT cURL 选项。
SSL 密钥文件	用于客户端身份验证的 SSL 私钥文件名称。私钥文件必须为 PEM[1] 格式。包含此文件的目录由 Zabbix 服务器或 proxy 配置参数 SSLKeyLocation 指定。 支持的宏：{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG}、用户宏、低级别发现宏。 此项设置 CURLOPT_SSLKEY cURL 选项。
SSL 密钥密码	SSL 私钥文件密码。 支持的宏：用户宏、低级别发现宏。 此项设置 CURLOPT_KEYPASSWD cURL 选项。
超时	Zabbix 处理 URL 所用的时间不会超过设定值（1-600 秒）。实际上，此参数定义了与 URL 建立连接以及执行 HTTP 请求的最长时间。因此，Zabbix 对一次检查所用的时间不会超过 2 x 超时秒。 此项设置 CURLOPT_TIMEOUT cURL 选项。 有关 超时参数的更多信息，请参阅监控项常规属性。

参数	描述
启用捕获	选中此复选框后，监控项还将作为 trapper 监控项运行，并接受通过Zabbix sender 工具或 Zabbix sender 协议 发送到 Zabbix 服务器或 proxy 的数据，或者接受通过history.push API 方法发送到 Zabbix 服务器的数据。有关发送数据的更多信息，请参阅：Trapper 监控项。
允许的主机	仅当选中 启用捕获复选框时可见。 以逗号分隔的 IP 地址列表（可选用 CIDR 表示法）或 DNS 名称。 默认值为用户宏 {$TRAPPER.ALLOWED_HOSTS}。默认情况下，{$TRAPPER.ALLOWED_HOSTS} 为 127.0.0.1,::1。可以直接在字段中输入地址，也可以更改宏值。如果留空，将拒绝来自所有主机的传入连接。如果指定了值，则仅接受来自此处列出的主机的传入连接。 如果启用了 IPv6 支持，则 127.0.0.1、::127.0.0.1、::ffff:127.0.0.1 将被视为相同，并且 ::/0 将允许任何 IPv4 或 IPv6 地址。 可以使用 0.0.0.0/0 允许任何 IPv4 地址。 请注意，“IPv4 兼容的 IPv6 地址”（0000::/96 前缀）受支持，但已被 RFC4291 弃用。 示例：127.0.0.1、192.168.1.0/24、192.168.3.1-255、192.168.1-10.1-255、::1、2001:db8::/32、mysqlserver1、zabbix.example.com、{HOST.HOST} 此字段允许使用空格和用户宏。 此字段允许使用主机宏：{HOST.HOST}、{HOST.NAME}、{HOST.IP}、{HOST.DNS}、{HOST.CONN}。

Note:
如果 *HTTP proxy* 字段留空，还可以通过设置与 proxy 相关的环境变量来使用 HTTP proxy。
对于 HTTP - 为 Zabbix 服务器用户设置 http_proxy 环境变量。例如：
 http_proxy=http://proxy_ip:proxy_port。
对于 HTTPS - 设置 HTTPS_PROXY 环境变量。例如：
 HTTPS_PROXY=http://proxy_ip:proxy_port。运行 shell 命令可获取更多详细信息：\# *man curl*。

Attention:
[1] Zabbix 仅支持 PEM 格式的证书和私钥文件。如果证书和私钥数据位于 PKCS #12 格式的文件中（通常扩展名为 *.p12 或 *.pfx），可以使用以下命令从中生成 PEM 文件：

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

示例

示例 1

发送简单的 GET 请求以从 Elasticsearch 等服务检索数据：

- 创建一个 GET 监控项，URL 为：localhost:9200/?pretty
- 注意响应：

```
{
  "name" : "YQ2VAY-",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "kH4CYqh5QfqgeTsjh2F9zg",
  "version" : {
    "number" : "6.1.3",
    "build_hash" : "af51318",
    "build_date" : "2018-01-26T18:22:55.523Z",
    "build_snapshot" : false,
    "lucene_version" : "7.1.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You know, for search"
}
```

- 现在使用 JSONPath 预处理步骤提取版本号：$.version.number

示例 2

发送简单的 POST 请求，从 Elasticsearch 等服务中检索数据：

- 创建一个 POST 监控项，URL 为：`http://localhost:9200/str/values/_search?scroll=10s`
- 配置以下 POST 请求体以获取处理器负载（每个核心 1 分钟平均值）

```
{
    "query": {
        "bool": {
            "must": [{
                "match": {
                    "itemid": 28275
                }
            }],
            "filter": [{
                "range": {
                    "clock": {
                        "gt": 1517565836,
                        "lte": 1517566137
                    }
                }
            }]
        }
    }
}
```

- 返回结果：

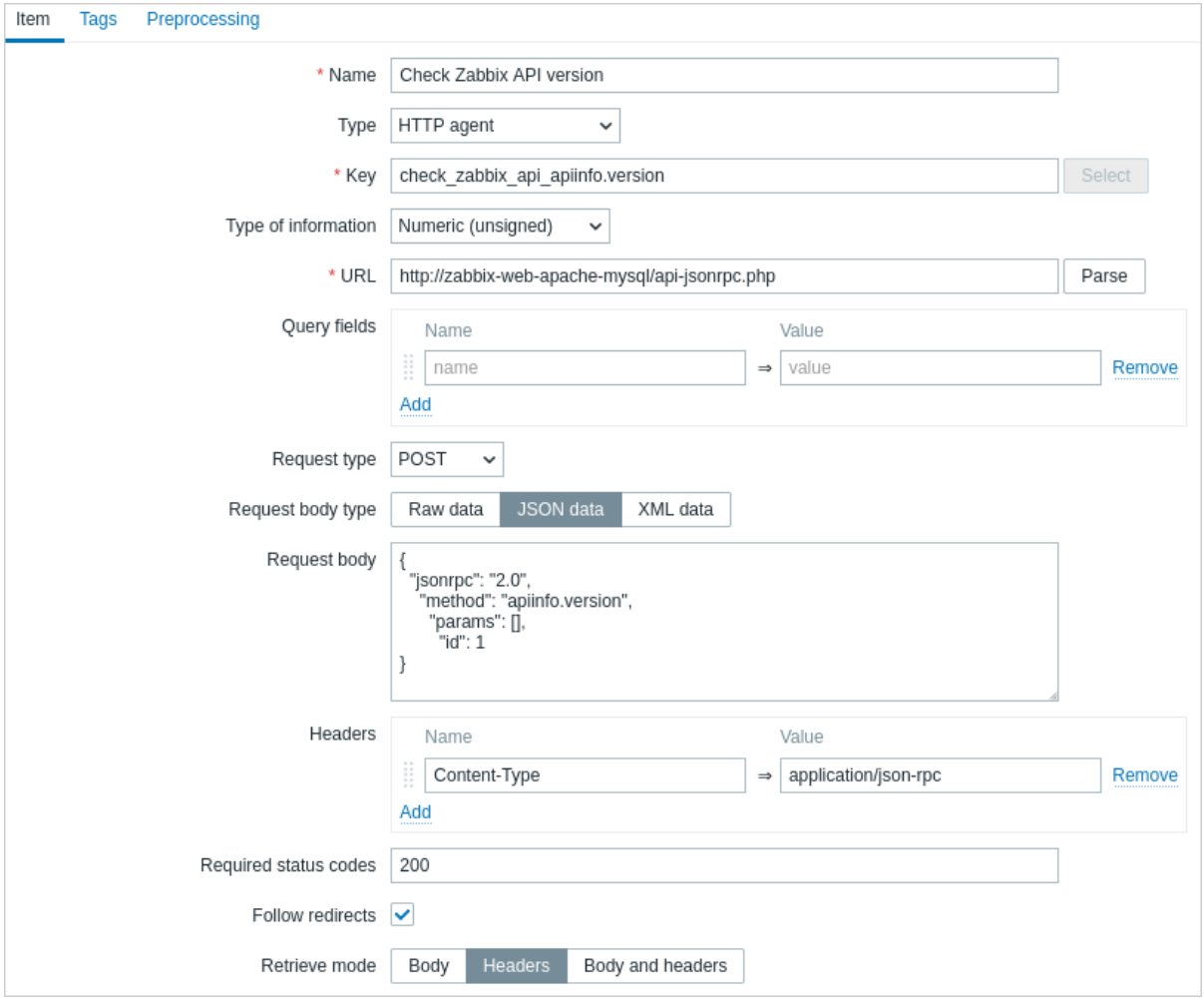
```
{
    "_scroll_id": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAJRZZUTJWQVk
    "took": 18,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 1.0,
        "hits": [{
            "_index": "dbl",
            "_type": "values",
            "_id": "dqX9VWEBV6sEKSMyk6sw",
            "_score": 1.0,
            "_source": {
                "itemid": 28275,
                "value": "0.138750",
                "clock": 1517566136,
                "ns": 25388713,
                "ttl": 604800
            }
        }]
    }
}
```

- 现在使用 JSONPath 预处理步骤获取监控项值：`$.hits.hits[0]._source.value`

示例 3

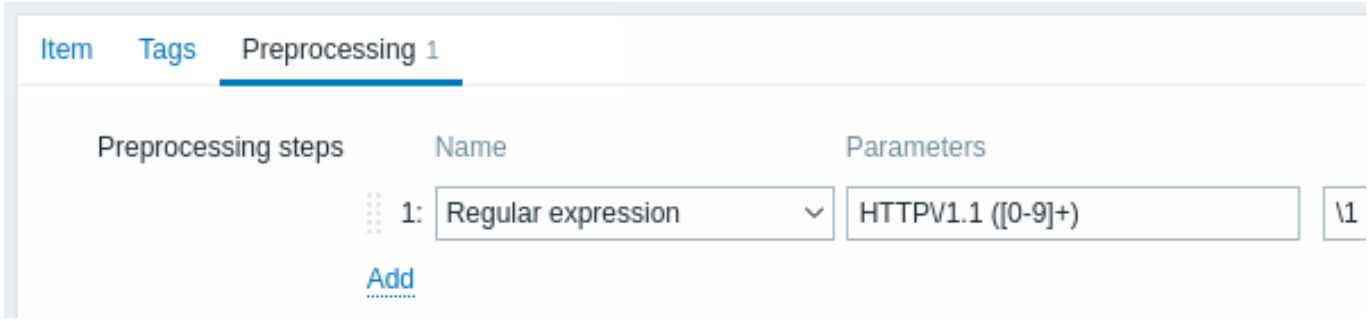
使用apiinfo.version 检查 Zabbix API 是否存活。

- 监控项配置：

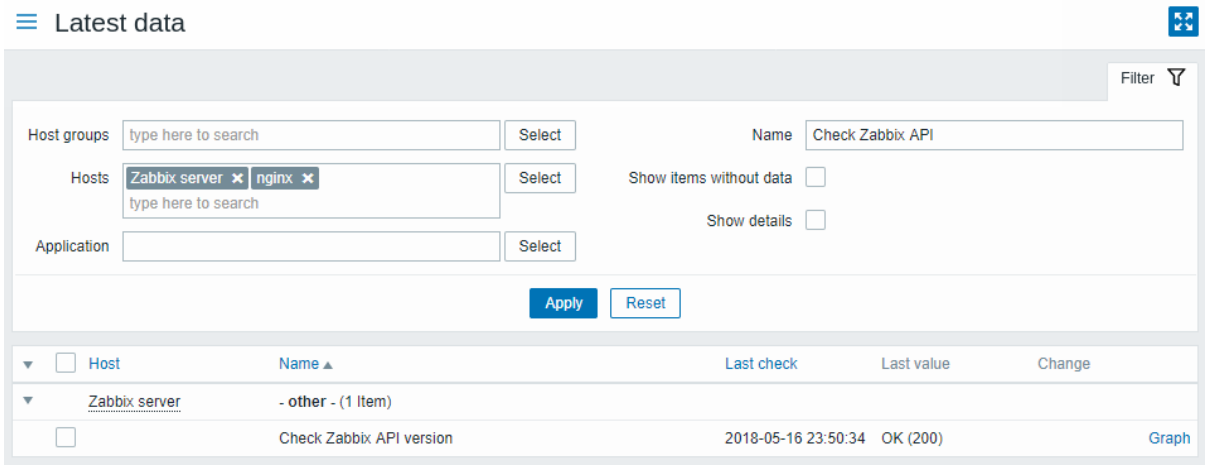


注意这里使用了带有 JSON 数据的 POST 方法，设置请求头，并要求仅返回响应头：

- 使用正则表达式对监控项值进行预处理，以获取 HTTP 状态码：



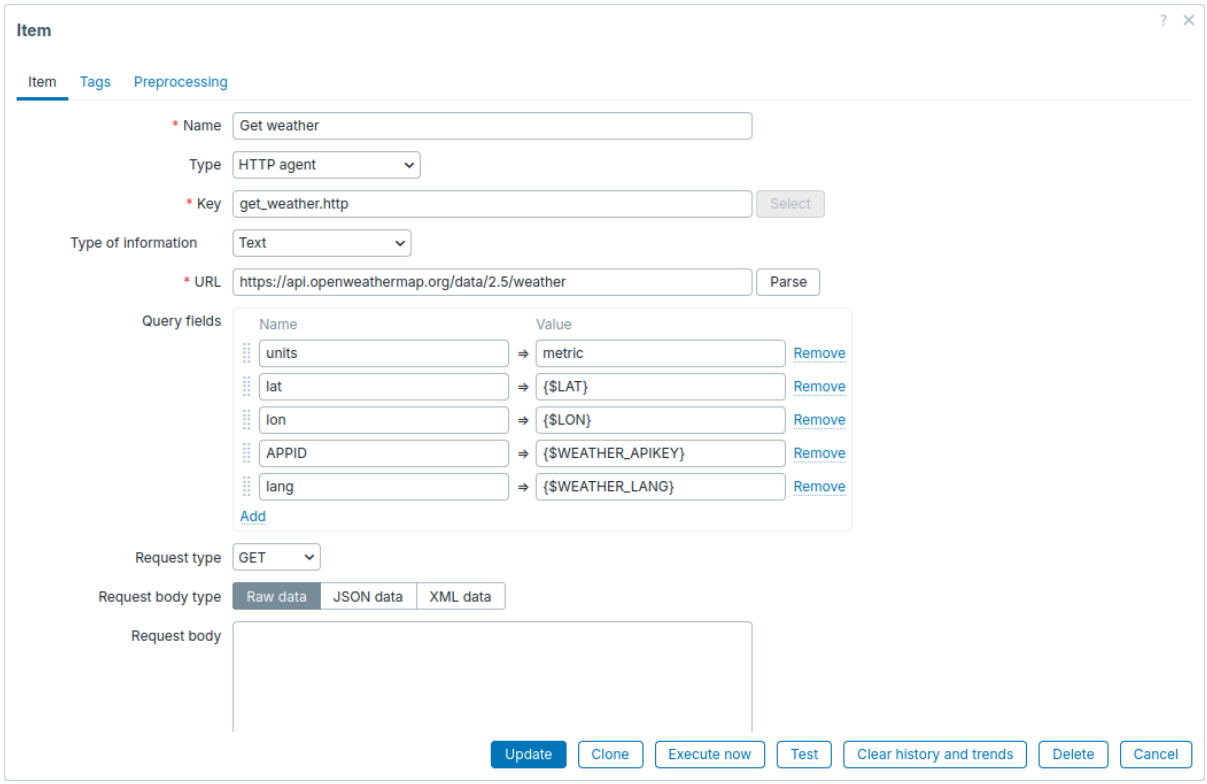
- 在 *Latest data* 中检查结果：



示例 4

通过连接 Openweathermap 公共服务获取天气信息。

- 配置一个主监控项，用于在单个 JSON 中批量收集数据：



请注意在查询字段中使用宏。有关如何填写，请参阅 Openweathermap API。

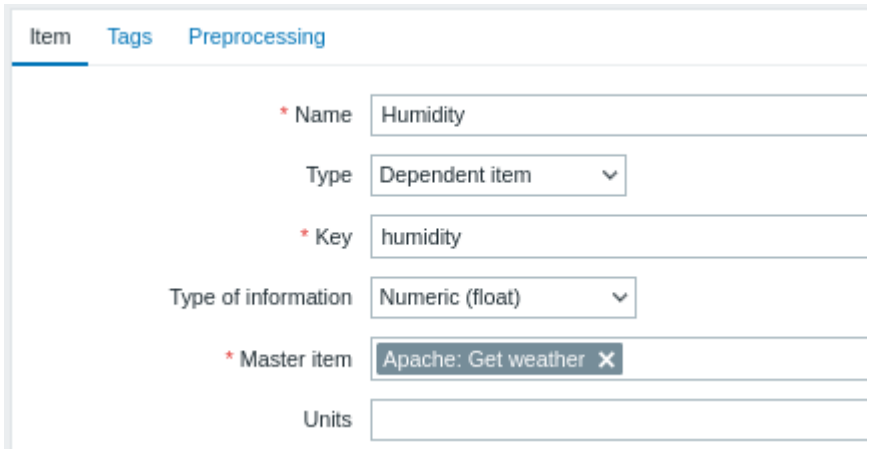
HTTP agent 返回的示例 JSON：


```
{
    "body": {
        "coord": {
            "lon": 40.01,
            "lat": 56.11
        },
        "weather": [{
            "id": 801,
            "main": "Clouds",
            "description": "few clouds",
            "icon": "02n"
        }],
        "base": "stations",
        "main": {
            "temp": 15.14,
            "pressure": 1012.6,
            "humidity": 66,
            "temp_min": 15.14,
            "temp_max": 15.14,
            "sea_level": 1030.91,
            "grnd_level": 1012.6
        },
        "wind": {
            "speed": 1.86,
            "deg": 246.001
        },
        "clouds": {
            "all": 20
        },
        "dt": 1526509427,
        "sys": {
            "message": 0.0035,
```

```
            "country": "RU",
            "sunrise": 1526432608,
            "sunset": 1526491828
        },
        "id": 487837,
        "name": "Stavrovo",
        "cod": 200
    }
}
```

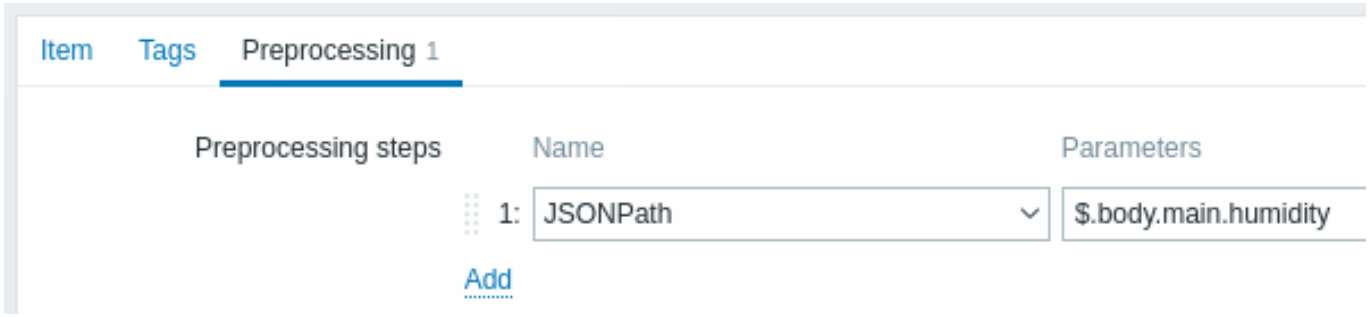

下一步是配置依赖监控项，从 JSON 中提取数据。

- 配置一个用于湿度的示例依赖监控项：

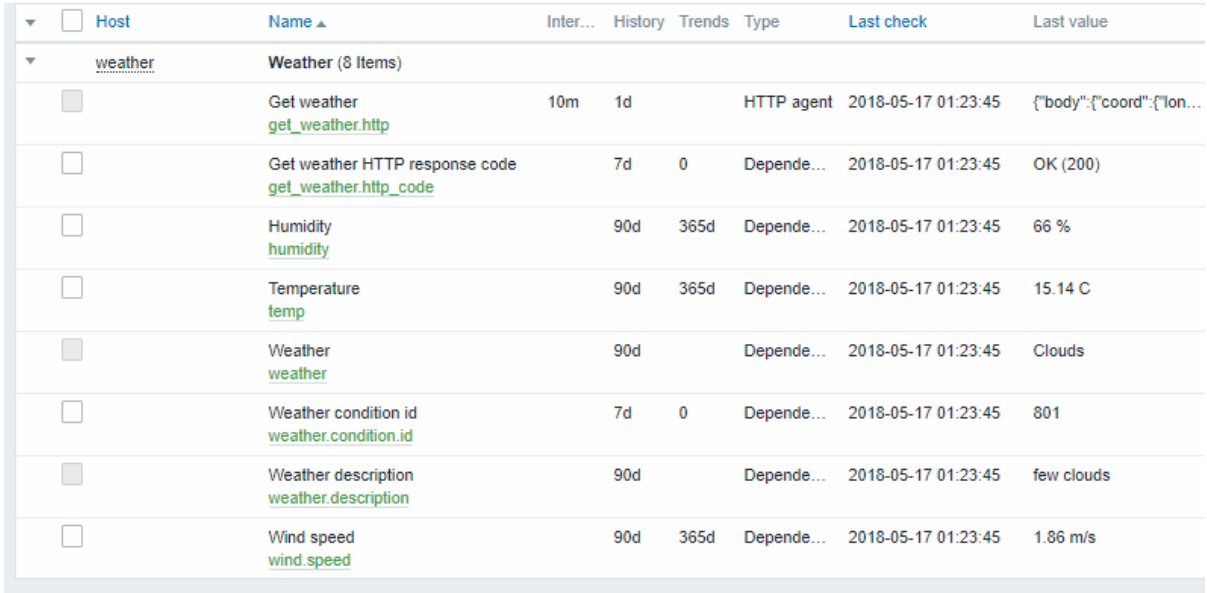


其他天气指标，例如 “Temperature”，也以相同方式添加。

- 使用 JSONPath 的示例依赖监控项值预处理：



- 在 *Latest data* 中查看天气数据结果：

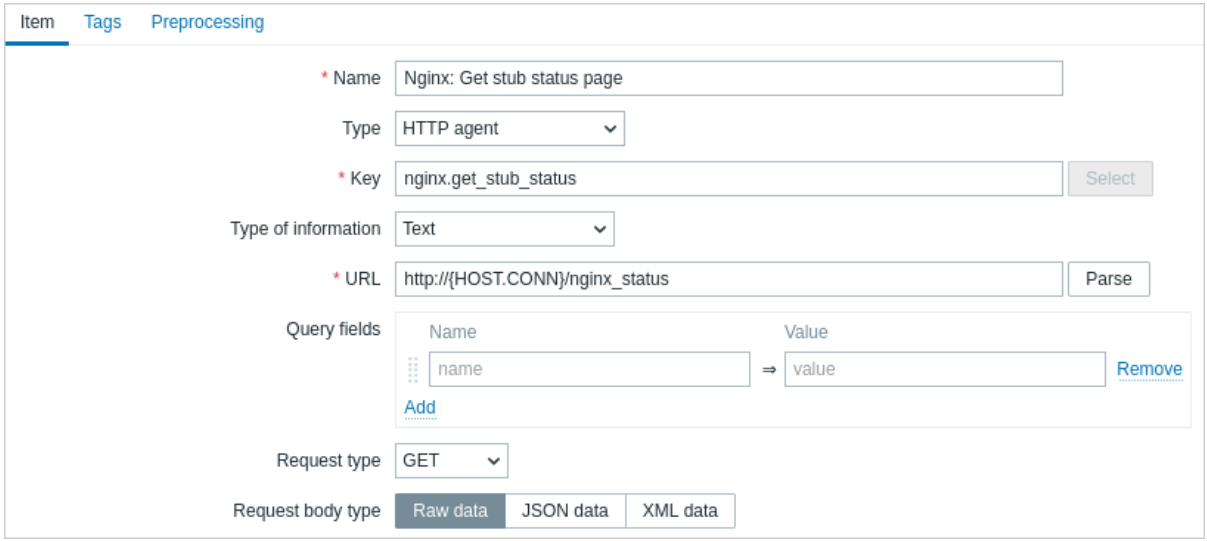


Host	Name ▲	Inter...	History	Trends	Type	Last check	Last value
weather	**Weather** (8 Items)						
	Get weather get_weather.http	10m	1d		HTTP agent	2018-05-17 01:23:45	{"body":{"coord":{"lon...
	Get weather HTTP response code get_weather.http_code		7d	0	Depende...	2018-05-17 01:23:45	OK (200)
	Humidity humidity		90d	365d	Depende...	2018-05-17 01:23:45	66 %
	Temperature temp		90d	365d	Depende...	2018-05-17 01:23:45	15.14 C
	Weather weather		90d		Depende...	2018-05-17 01:23:45	Clouds
	Weather condition id weather.condition.id		7d	0	Depende...	2018-05-17 01:23:45	801
	Weather description weather.description		90d		Depende...	2018-05-17 01:23:45	few clouds
	Wind speed wind.speed		90d	365d	Depende...	2018-05-17 01:23:45	1.86 m/s

示例 5

连接到 NGINX 状态页面并批量获取其指标。

- 按照官方指南配置 NGINX。
- 配置用于批量数据采集的主监控项：

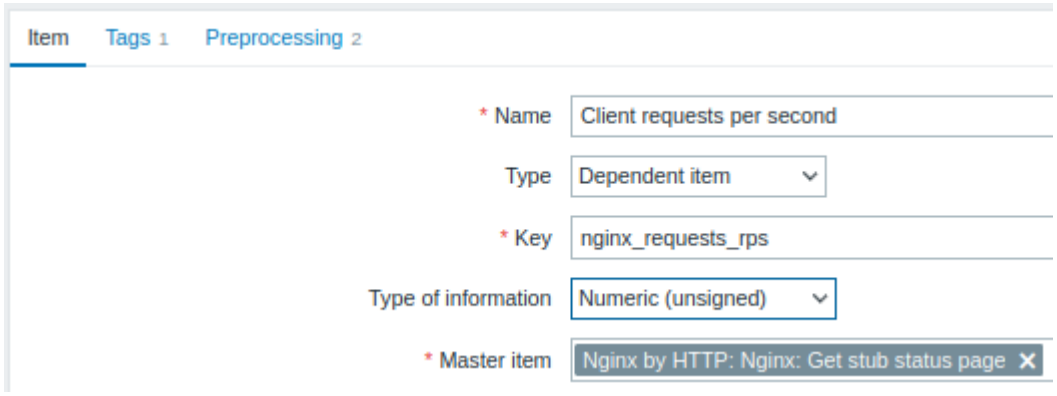


NGINX stub status 输出示例：

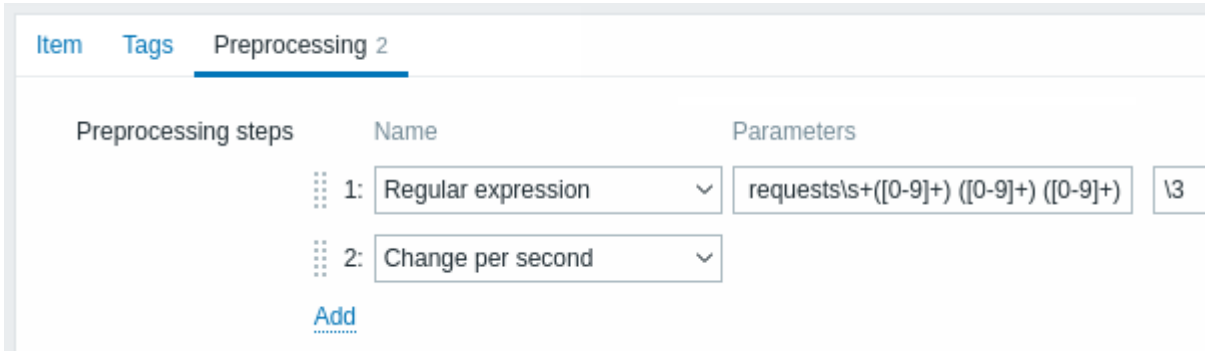
```
Active connections: 1 Active connections:
server accepts handled requests
 52 52 52
Reading: 0 Writing: 1 Waiting: 0
```

下一步是配置用于提取数据的依赖监控项。

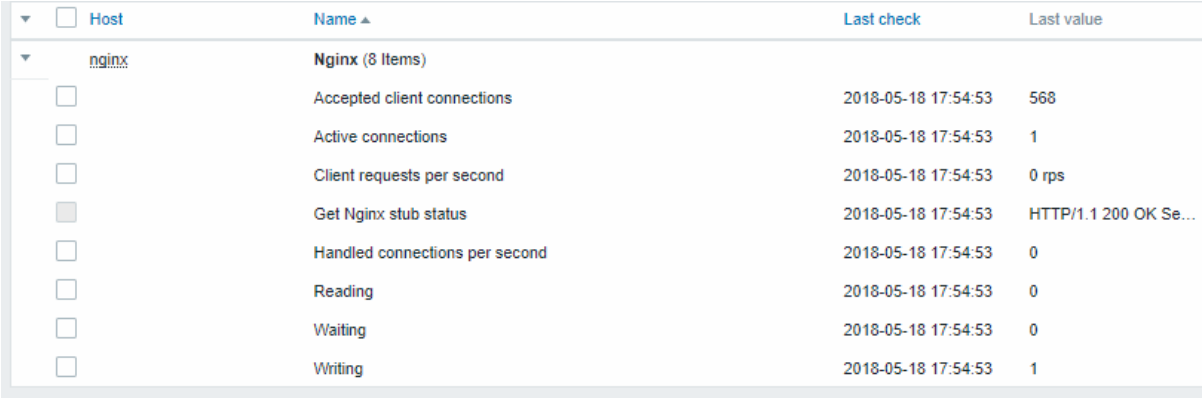
- 配置用于获取每秒请求数的依赖监控项示例：



- 使用正则表达式 `server accepts handled requests\s+([0-9]+) ([0-9]+) ([0-9]+)` 对依赖监控项值进行预处理的示例：



- 在最新数据中检查 stub 模块返回的完整结果：



Host	Name ▲	Last check	Last value
nginx	**Nginx** (8 Items)		
	Accepted client connections	2018-05-18 17:54:53	568
	Active connections	2018-05-18 17:54:53	1
	Client requests per second	2018-05-18 17:54:53	0 rps
	Get Nginx stub status	2018-05-18 17:54:53	HTTP/1.1 200 OK Se...
	Handled connections per second	2018-05-18 17:54:53	0
	Reading	2018-05-18 17:54:53	0
	Waiting	2018-05-18 17:54:53	0
	Writing	2018-05-18 17:54:53	1

Prometheus 检查

概述

Zabbix 可以查询以 Prometheus 行格式暴露的监控指标。

开始收集 Prometheus 数据需要两个步骤：

- 一个指向相应数据端点的HTTP 主监控项，例如 `https://<prometheus host>/metrics`
- 使用 Prometheus 预处理选项的依赖监控项，用于从主监控项收集的监控指标中查询所需数据

有两种 Prometheus 数据预处理选项：

- *Prometheus pattern* - 用于普通监控项，以查询 Prometheus 数据
- *Prometheus to JSON* - 用于普通监控项和低级别发现。在这种情况下，查询到的 Prometheus 数据会以 JSON 格式返回。

批量处理

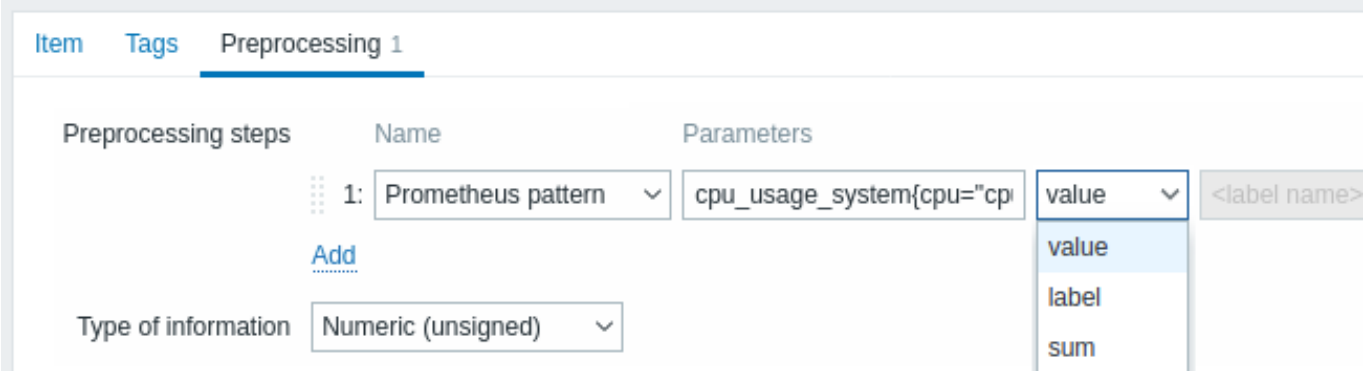
依赖监控项支持批量处理。为了启用缓存和索引，*Prometheus pattern* 预处理必须是第一个预处理步骤。
当 *Prometheus pattern* 是第一个预处理步骤时，已解析的 Prometheus 数据会根据 *Prometheus pattern* 预处理步骤中的第一个 `<label>==<value>` 条件进行缓存和索引。处理此批次中的其他依赖监控项时，会复用该缓存。为了获得最佳性能，第一个标签应选择取值差异最多的那个。

如果在第一个步骤之前还需要执行其他预处理，则应将其移到主监控项，或者移到一个新的依赖监控项中，并将该监控项用作这些依赖监控项的主监控项。

配置

在配置好 HTTP 主监控项的前提下，您需要创建一个使用 Prometheus 预处理步骤的依赖监控项：

1. 在配置表单中输入常规依赖监控项参数。
2. 转到预处理选项卡。
3. 选择 Prometheus 预处理选项（*Prometheus* 模式或 *Prometheus* 转 *JSON*）。



以下参数专用于 *Prometheus* 模式预处理选项：

参数	描述	示例
模式	要定义所需的数据模式，可以使用类似于 Prometheus 查询语言的查询语言（请参阅比较表），例如： <metric name> - 按指标名称选择 {__name__="<metric name>"} - 按指标名称选择 {__name__=~"<regex>"} - 选择与正则表达式匹配的指标名称 {<label name>="<label value>",...} - 按标签名称选择 {<label name>=~"<regex>",...} - 选择与正则表达式匹配的标签名称 {__name__=~".*"}==<value> - 按指标值选择 或者组合使用上述条件： <metric name>{<label1 name>="<label1 value>",<label2 name>=~"<regex>",...}==<value> 标签值可以是任意 UTF-8 字符序列，但反斜杠、双引号和换行符必须分别转义为 \\、\" 和 \n；其他字符不应转义。	*wmi_os_physical_memory_free_bytes* *cpu_usage_system{cpu="cpu-total"}* *cpu_usage_system{cpu=".*"}* *cpu_usage_system{cpu="cpu-total",host="* *wmi_service_state{name="dhcp"}==1* *wmi_os_timezone{timezone=~".*"}==1*
结果处理	指定返回值、标签，或应用适当的函数（如果模式匹配多行且需要聚合结果）： **value** - 返回指标值（匹配多行时出错） **label** - 返回标签字段中指定标签的值（匹配多个指标时出错） **sum** - 返回值的总和 **min** - 返回最小值 **max** - 返回最大值 **avg** - 返回平均值 **count** - 返回值的数量 此字段仅适用于 *Prometheus* 模式选项。	另请参阅下面的参数使用示例。
输出	定义标签名称（可选）。在这种情况下，将返回与该标签名称对应的值。 仅当在结果处理字段中选择 "Label" 时，此字段才适用于 *Prometheus* 模式选项。	

参数使用示例

1. 最常见的用例是返回值。要从以下数据中返回 /var/db 的值：

   node_disk_usage_bytes{path="/var/cache"} 2.1766144e+09
 node_disk_usage_bytes{path="/var/db"} 20480
 node_disk_usage_bytes{path="/var/dpkg"} 8192
 node_disk_usage_bytes{path="/var/empty"} 4096

请使用以下参数：

- 模式 - node_disk_usage_bytes{path="/var/db"}
- 结果处理 - 选择 "value"

2. 您可能还需要所有 node_disk_usage_bytes 参数的平均值：

- 模式 - node_disk_usage_bytes
- 结果处理 - 选择 "avg"

3. 虽然 Prometheus 仅支持数值数据，但通常可以通过变通方式同时返回相关的文本描述。这可以通过使用过滤器并指定标签来实现。因此，要从以下数据中返回 "color" 标签的值：

   elasticsearch_cluster_health_status{cluster="elasticsearch",color="green"} 1
 elasticsearch_cluster 0
 elasticsearch_cluster_health_status{cluster="elasticsearch",color="yellow"} 0

请使用以下参数：

- 模式 - elasticsearch_cluster_health_status {cluster="elasticsearch"} == 1
- 结果处理 - 选择 "label"
- 标签 - 指定 "color"

基于数值 “1” 的过滤器将匹配相应的行，而标签将返回健康状态描述（当前为 “green”，但也可能是 “red” 或 “yellow”）。

Prometheus 转 JSON

Prometheus 中的数据可用于低级别发现。在这种情况下，需要 JSON 格式的数据，而 *Prometheus to JSON* 预处理选项将正好返回这种格式。

有关更多详细信息，请参见使用 Prometheus 数据进行发现。

查询语言比较

下表列出了 PromQL 和 Zabbix Prometheus 预处理查询语言之间的差异和相似之处。

PromQL instant vector selector		Zabbix Prometheus preprocessing
差异		
查询目标	Prometheus 服务器	Prometheus exposition 格式的纯文本
返回值	瞬时向量	监控项或标签值（Prometheus 模式） JSON 中单个值的监控项数组（Prometheus to JSON）
标签匹配运算符	**=**, **!=**, **=~**, **!~**	**=**, **!=**, **=~**, **!~**
标签或监控项名称匹配中使用的正则表达式	RE2	PCRE
比较运算符	参见 列表	仅支持 **==**（等于）用于值过滤
相似之处		
按等于字符串的监控项名称选择	<metric name> 或 {__name__=”<metric name>”}	<metric name> 或 {__name__=”<metric name>”}
按匹配正则表达式的监控项名称选择	{__name__=~”<regex>”}	{__name__=~”<regex>”}
按等于字符串的 <label name> 值选择	{<label name>=”<label value>”,...}	{<label name>=”<label value>”,...}
按匹配正则表达式的 <label name> 值选择	{<label name>=~”<regex>”,...}	{<label name>=~”<regex>”,...}
按等于字符串的值选择	{__name__=~”.*”} == <value>	{__name__=~”.*”} == <value>

IPMI agent

概述

您可以在 Zabbix 中监控智能平台管理接口（IPMI）设备的运行状况和可用性。要执行 IPMI 检查，必须先为 Zabbix 服务器配置 IPMI 支持。

IPMI 是用于远程 “带外” 管理计算机系统的标准化接口。它允许您直接从所谓的 “带外” 管理卡监控硬件状态，而不受操作系统影响，也不受计算机是否已开机的影响。

Zabbix IPMI 监控仅适用于支持 IPMI 的设备（HP iLO、DELL DRAC、IBM RSA、Sun SSP 等）。

IPMI 管理器进程通过 IPMI 轮询器调度 IPMI 检查。一个主机在同一时间始终只由一个 IPMI 轮询器进行轮询，从而减少与 BMC 控制器建立的开放连接数。因此，您可以放心增加 IPMI 轮询器的数量，而不必担心 BMC 控制器过载。当至少启动一个 IPMI 轮询器时，IPMI 管理器进程会自动启动。

另请参阅 IPMI 检查的已知问题。

配置

主机配置

主机必须配置为处理 IPMI 检查。必须添加 IPMI 接口，必须定义相应的 IP 和端口号，并且必须定义 IPMI 认证参数。

更多细节请查看主机配置 。

服务器配置

默认情况下，Zabbix 服务器未配置为启动任何 IPMI poller，因此添加的任何 IPMI 监控项都无法工作。要更改此设置，请以 root 身份打开 Zabbix 服务器配置文件 (`zabbix_server.conf`)，并查找以下行：

```
# StartIPMIPollers=0
```

取消注释并将 poller 数量设置为例如 3，使其变为：

```
StartIPMIPollers=3
```

保存文件，然后重启 zabbix_server。

监控项配置

配置主机级别的监控项:

- 选择'IPMI agent' 作为 类型
- 在主机中输入唯一的监控项键值 (比如, ipmi.fan.rpm)
- 对于 主机接口选择 IPMI 接口 (IP 和端口). 注意 IPMI 接口在主机上必须存在.
- 指定 *IPMI* 传感器 (比如在 Dell Poweredge 型号服务器上的'FAN MOD 1A RPM') 用于检索对应的指标。默认情况下，应指定传感器 ID。也可以在值之前使用前缀:
  - `id:` - 指定传感器 ID;
  - `name:` - 指定传感器全名。这在传感器只能通过指定全名来区分的情况下非常有用.
- 选择相应的信息类型 (' 浮点数' 在这个例子中；对于离散传感器使用' 数字 (无正负)'), units (类似'rpm') 和任何其它必需监控项属性

支持的检查项

IPMI agent 支持内置监控项 **ipmi.get**，该监控项返回与 IPMI 传感器相关的信息，并可用于发现 IPMI 传感器。
 返回值：*JSON* 对象

超时和会话终止

IPMI 消息超时和重试次数由 OpenIPMI 库定义。由于 OpenIPMI 当前的设计，无法在 Zabbix 中将这些值设置为可配置，无论是在接口级别还是监控项级别。

LAN 的 IPMI 会话非活动超时时间为 60 +/-3 秒。目前无法使用 OpenIPMI 实现定期发送 Activate Session 命令。如果 Zabbix 在超过 BMC 中配置的会话超时时间内，没有对某个特定 BMC 执行任何 IPMI 监控项检查，那么在超时到期后的下一次 IPMI 检查将因单个消息超时、重试或接收错误而超时。之后将打开一个新会话，并启动对 BMC 的完整重新扫描。还可能会打开一个新的 UDP 端口来管理新会话。非活动是指既没有传出请求，也没有传入响应。如果您想避免对 BMC 进行不必要的重新扫描，建议将 IPMI 监控项轮询间隔设置为低于 BMC 中配置的 IPMI 会话非活动超时时间。

关于 IPMI 离散传感器的说明

要查找主机上的传感器，请在启用 **DebugLevel=4** 的情况下启动 Zabbix 服务器。等待几分钟后，在 Zabbix 服务器日志文件中查找传感器发现记录：

```
$ grep 'Added sensor' zabbix_server.log
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:7 id:'CATERR' reading_type:
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'CPU Therm Trip' read
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'System Event Log' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'PhysicalSecurity' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'IPMI Watchdog' readi
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'Power Unit Stat' rea
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Ctrl %' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Margin' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 2' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 3' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'P1 Mem Margin' readi
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'Front Panel Temp' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'Baseboard Temp' read
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +5.0V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +3.3V STBY' readi
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +3.3V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.5V P1 DDR3' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.1V P1 Vccp' re
8358:20130318:111122.174 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +1.05V PCH' readi
```

要解码 IPMI 传感器类型和状态，可参考一份 IPMI 2.0 规范 副本（请注意，IPMI 规范 不再计划进行后续更新）。

首先要查看的参数是"reading_type"。使用规范中的 "Table 42-1, Event/Reading Type Code Ranges" 来解码"reading_type" 代码。在我们的示例中，大多数传感器都具有"reading_type:0x1"，这表示 "阈值" 传感器。"Table 42-3, Sensor Type Codes" 显示，"type:0x1" 表示温度传感器，"type:0x2" 表示电压传感器，"type:0x4" 表示风扇等。阈值传感器有时也被称为 "模拟" 传感器，因为它们测量的是温度、电压、每分钟转数等连续参数。

再看一个例子：某个传感器的"reading_type:0x3"。"Table 42-1, Event/Reading Type Code Ranges" 说明，读取类型代码 02h-0Ch 表示 "通用离散" 传感器。离散传感器最多可有 15 种可能状态（换句话说，最多有 15 个有意义的位）。例如，对于传感器'CATERR'，其"type:0x7"，而 "Table 42-3, Sensor Type Codes" 显示该类型表示 "处理器"，各个位的含义分别为：00h（最低有效位）- IERR，01h - Thermal Trip 等。

在我们的示例中，还有几个传感器的"reading_type:0x6f"。对于这些传感器，"Table 42-1, Event/Reading Type Code Ranges" 建议使用 "Table 42-3, Sensor Type Codes" 来解码各个位的含义。例如，传感器'Power Unit Stat' 的类型为"type:0x9"，表示 "电源单元"。偏移量 00h 表示 "PowerOff/Power Down"。换句话说，如果最低有效位为 1，则表示服务器已关机。要测试这一位，可以使用带有掩码'1' 的`bitand` 函数。触发器表达式可以是：

```
bitand(last(/www.example.com/Power Unit Stat,#1),1)=1
```

用于对服务器关机发出警告。

关于 OpenIPMI-2.0.16,2.0.17,2.0.18 和 2.0.19 中离散传感器名称的注释

OpenIPMI-2.0.16,2.0.17 和 2.0.18 中的离散传感器的名称通常在附近附加一个额外的"0"（或其它数字或字母）。例如，当 ipmitool 和 OpenIPMI-2.0.19 将传感器名称显示为"PhysicalSecurity" 或"CATERR" 时，在 OpenIPMI-2.0.16,2.0.17 和 2.0.18 中，名称分别为"PhysicalSecurity0" 或"CATERR0"。

当使用 OpenIPMI-2.0.16, 2.0.17 和 2.0.18 配置 IPMI 监控项时，请在 IPMI 代理监控项的 *IPMI* 传感器字段中使用以 "0" 结尾的名称。当你的 Zabbix server 升级到使用 OpenIPMI-2.0.19（或更高版本）的新 Linux 发行版时，具有这些 IPMI 离散传感器的监控项将变为 "不支持"。你必须更改其 *IPMI* 传感器名称（最后删除 "0"），并等待一段时间才能再次转为"Enabled"。

关于阈值和离散传感器同时可用的注意事项

某些 IPMI 代理以相同的名称提供阈值传感器和离散传感器。始终优先考虑阈值传感器。

连接终止注意事项

如果不执行 IPMI 检查（由于任何原因：所有主机 IPMI 监控项禁用/不支持、主机已禁用/已删除、主机维护等），IPMI 连接将从 Zabbix server 或 proxy 终止 3 到 4 小时，具体时间取决于 Zabbix server/proxy 何时启动。

SSH agent

概述

SSH 检查作为无 agent 监控执行。SSH 检查不需要 Zabbix agent。

要执行 SSH 检查，Zabbix 服务器必须首先启用 SSH2 支持（libssh 或 libssh2）进行配置。另请参见：要求。

Attention:
从 RHEL 8 开始，仅支持 libssh。对于其他发行版，建议使用 libssh 而不是 libssh2。

配置

密码验证

SSH 检查提供两种验证方法 - 用户/密码对和基于密钥文件。

如果您不打算使用密钥，则无需进行其他配置，除了将 libssh 或 libssh2 链接到 Zabbix（如果您从源代码构建）之外。

密钥文件认证

要为 SSH 监控项使用基于密钥的认证，需要对服务器配置进行一些更改。

以 root 身份打开 Zabbix 服务器配置文件 (zabbix_server.conf)，并查找以下行：

```
##### SSHKeyLocation=
```

取消注释，并将其设置为公钥和私钥所在文件夹的完整路径：

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

保存文件，然后重启 Zabbix 服务器。

这里的路径 /home/zabbix 是 *zabbix* 用户账户的主目录，.ssh 是一个目录，默认情况下，公钥和私钥会通过 ssh-keygen 命令在主目录中生成。

通常，不同操作系统发行版提供的 Zabbix 服务器安装包会将 *zabbix* 用户账户创建在其他位置的主目录下，例如 /var/lib/zabbix (系统账户通常如此)。

在生成密钥之前，你可以先将主目录重新分配到 /home/zabbix，使其与上面提到的 Zabbix 服务器 SSHKeyLocation 配置参数相对应。

Note:
如果 *zabbix* 账户是按照安装部分手动添加的，则可以跳过以下步骤。在这种情况下，*zabbix* 账户的主目录很可能已经是 /home/zabbix。

要更改 *zabbix* 用户账户的主目录，必须先停止所有正在使用它的工作进程：

```
systemctl stop zabbix-agent
systemctl stop zabbix-server
```

要更改主目录位置并尝试移动它（如果它存在），应执行以下命令：

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

也有可能旧位置中并不存在主目录，因此应在新位置创建它。一个安全的做法是：

```
test -d /home/zabbix || mkdir /home/zabbix
```

为确保一切安全，还可以执行额外命令来设置主目录权限：

```
chown zabbix:zabbix /home/zabbix
chmod 700 /home/zabbix
```

现在可以重新启动之前停止的进程：

```
systemctl start zabbix-agent
systemctl start zabbix-server
```

接下来，可以使用以下命令生成公钥和私钥（为提高可读性，命令提示符已注释掉）：

```
sudo -u zabbix ssh-keygen -t rsa
##### Generating public/private rsa key pair.
##### Enter file in which to save the key (/home/zabbix/.ssh/id_rsa):
/home/zabbix/.ssh/id_rsa
##### Enter passphrase (empty for no passphrase):
<Leave empty>
##### Enter same passphrase again:
<Leave empty>
##### Your identification has been saved in /home/zabbix/.ssh/id_rsa.
##### Your public key has been saved in /home/zabbix/.ssh/id_rsa.pub.
##### The key fingerprint is:
##### 90:af:e4:c7:e3:f0:2e:5a:8d:ab:48:a2:0c:92:30:b9 zabbix@it0
##### The key's randomart image is:
##### +--[ RSA 2048]----+
##### |                 |
##### |       .         |
##### |      o          |
##### | .     o         |
##### |+     . S        |
##### |.+   o =         |
##### |E .   * =        |
##### |=o . ..* .       |
##### |... oo.o+        |
##### +-----------------+
```

Note:
公钥和私钥（id_rsa.pub 和 id_rsa）默认会生成在 /home/zabbix/.ssh 目录中，这与 Zabbix 服务器的 SSHKeyLocation 配置参数相对应。

Attention:
除"rsa" 之外的其他密钥类型可能被 ssh-keygen 工具和 SSH 服务器支持，但它们可能不被 Zabbix 使用的 libssh2 支持。

Shell 配置表单

对于每个将通过 SSH 检查进行监控的主机，此步骤只需执行一次。

使用以下命令，可以将 公钥文件安装到远程主机 *10.10.10.10* 上，这样就可以使用 *root* 账户执行 SSH 检查（为便于阅读，命令提示符已注释掉）：

```
sudo -u zabbix ssh-copy-id root@10.10.10.10
##### The authenticity of host '10.10.10.10 (10.10.10.10)' can't be established.
##### RSA key fingerprint is 38:ba:f2:a4:b5:d9:8f:52:00:09:f7:1f:75:cc:0b:46.
##### Are you sure you want to continue connecting (yes/no)?
yes
##### Warning: Permanently added '10.10.10.10' (RSA) to the list of known hosts.
##### root@10.10.10.10's password:
<Enter root password>
##### Now try logging into the machine, with "ssh 'root@10.10.10.10'",
```

```
##### and check to make sure that only the key(s) you wanted were added.
```

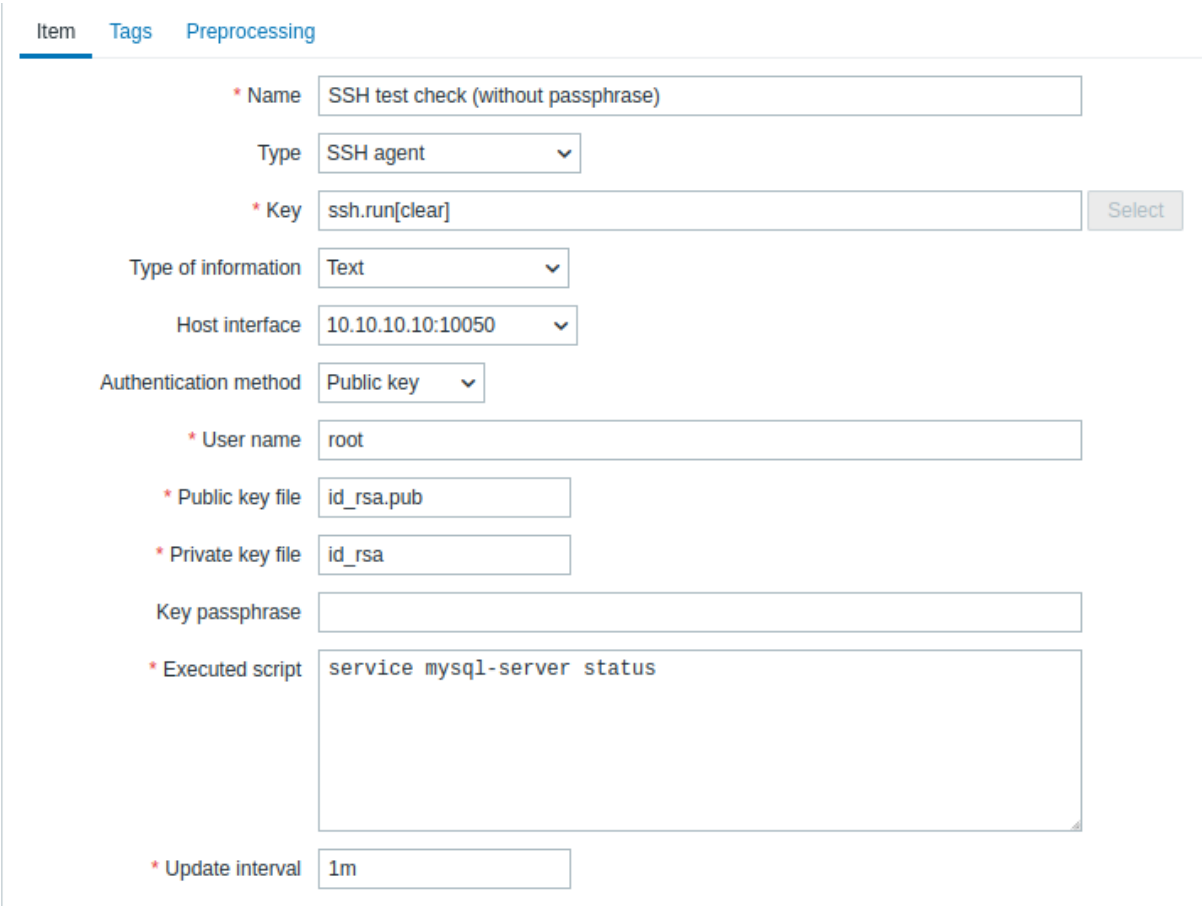
现在，可以使用 *zabbix* 用户账户的默认私钥 (`/home/zabbix/.ssh/id_rsa`) 来检查 SSH 登录：

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

如果登录成功，则 shell 中的配置部分已完成，可以关闭远程 SSH 会话。

监控项配置

实际要执行的命令必须放置在监控项配置的 *Executed script* 字段中。通过将多个命令放置在新行中，可以依次执行这些命令。在这种情况下，返回值也将以多行格式显示。



所有必填输入字段均以红色星号标记。

SSH 监控项需要特定信息的字段如下：

参数	描述	备注
Type	在此处选择 **SSH agent**。	

参数	描述	备注

Key	每个主机唯一的监控项键，格式为
	`ssh.run[unique short description,<ip>,<port>,<encoding>,<ssh options>,<subsystem>]`
		unique short description 为必填项，并且对于每个主机上的每个 SSH 监控项都应保持唯一。
		
		默认端口为 22，而不是分配给此监控项的接口中指定的端口。
		
		ssh options 允许以 `key1=value1;key2=value2,value3` 格式传递其他 SSH 选项。一个键可以传递多个值，多个值之间使用逗号分隔（在这种情况下，参数必须使用引号）；多个选项键可以使用分号分隔。
		
		支持以下选项键：`KexAlgorithms`、`HostkeyAlgorithms`、`Ciphers`、`MACs`、`PubkeyAcceptedKeyTypes`。选项键和值的支持情况取决于 SSH 库（例如，`PubkeyAcceptedKeyTypes` 仅受 libssh 支持）；如果不支持某个选项，将返回错误，并且该监控项将变为不受支持。
		
		请注意，不支持用于追加密码设置的 “+” 符号以及用于禁用特定密码设置的 “!” 符号（如 GnuTLS 和 OpenSSL 中的用法）。
		
		示例：
		`=>`
		`ssh.run[KexAlgorithms,127.0.0.1,,,Ciphers=aes128-`
		`=>`
		`ssh.run[KexAlgorithms,,,,"KexAlgorithms=diffie-he`
		`=>`
		`ssh.run[PubkeyAcceptedKeyTypes,127.0.0.1,,,Pubkey`
		
		subsystem 允许传递 SSH 子系统，将 SSH 连接限制为该子系统允许的特定操作（例如，使用 SFTP 传输文件或使用 NETCONF 管理网络设备）。请注意，使用子系统可能还需要在 *Executed script* 参数中使用特定的脚本语法。
		
		示例：
		`=>`
		`ssh.run[SFTPBackup,192.0.2.18,,,,sftp]`
		`=>`
		`ssh.run[Cisco1234,192.0.2.18,,,,netconf]`

Authentication method	选择 “Password” 或 “Public key” 之一。	
User name	用于在远程主机上进行身份验证的用户名（最多 255 个字符）。必填。	
Public key file	当 *Authentication method* 为 “Public key” 时，公钥的文件名。必填。	示例：*id_rsa.pub* - 由 ssh-keygen 命令生成的默认公钥文件名。
Private key file	当 *Authentication method* 为 “Public key” 时，私钥的文件名。必填。	示例：*id_rsa* - 默认私钥文件名。
Password 或 *Key passphrase*	用于身份验证的密码（最多 255 个字符），或如果私钥使用了密码短语，则填写密码短语。	如果未使用密码短语，请将 *Key passphrase* 字段留空。 另请参阅有关密码短语使用的已知问题。

参数	描述	备注

Executed script	使用 SSH 远程会话执行的 shell 命令。	执行的 shell 命令返回值限制为 16MB（包括会被截断的尾随空白）；数据库限制同样适用。
		
		请注意，libssh2 库可能会将可执行脚本截断至约 32kB。
		
		示例：
		`date +%s`
		`systemctl status mysql-server`
		`ps auxww | grep httpd | wc -l`
		
		NETCONF 子系统示例：
		`<rpc>`
		`<get-software-information/>`
		`</rpc>`
		`]]>]]>`
		`<rpc>`
		`<close-session/>`
		`</rpc>`
		`]]>]]>`

Telnet agent

概述

Telnet 检查不需要安装 Zabbix agent，可对未安装代理的主机进行监控。

可配置字段

实际要执行的命令必须放在监控项配置中的 **Executed script** 字段中。
 可通过将多条命令分别放在新的一行中来依次执行。在这种情况下，返回值也将被格式化为多行。

shell 提示符支持以下结尾字符：

- $
- #
- >
- %

Note:
如果 telnet 提示行以这些字符之一结尾，则该提示行会从返回值中移除，但这仅适用于命令列表中的第一条命令，也就是说，仅在 telnet 会话开始时生效。

键	描述
telnet.run[< 唯一简短描述 >,<ip>,<port>,<encoding>]	使用 telnet 连接在远程设备上运行命令

Attention:
如果 telnet 检查返回的值包含非 ASCII 字符，且编码不是 UTF-8，则应正确指定键中的 *<encoding>* 参数。更多详情请参见返回值编码页面。

JMX agent

概述

JMX 监控端可用于监控 Java 应用程序的 JMX 计数器。

JMX 监控在 Zabbix 中原生支持，形式为一个名为"Zabbix Java 网关"（Zabbix Java Gateway）的 Zabbix 守护进程。

为了检索主机上特定 JMX 计数器的值，Zabbix server 查询 Zabbix **Java gateway**，网关使用JMX 管理 API远程查询指定的应用程序，将结果返回给 zabbix server。

关于更多细节和设置，请参见章节Zabbix Java gateway。

Warning:
Java gateway 和 JMX 应用程序之间的通信应该在防火墙上放行。

为 Java 应用程序启用远程 JMX 监控

Java 应用程序不需要安装任何额外软件，但需要使用下面指定的命令行选项启动，才能支持远程 JMX 监控。

作为最基本的配置，如果您只是希望开始监控本地主机上的一个简单 Java 应用程序，且不启用任何安全机制，请使用以下选项启动：

```
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-Dcom.sun.management.jmxremote.registry.ssl=false \
-jar /path/to/your/application.jar
```

这样会使 Java 在端口 12345 上侦听传入的 JMX 连接，并且仅允许来自本地主机的连接，同时不要求身份验证或 SSL。

如果您希望允许通过其他网络接口进行连接，请将 -Djava.rmi.server.hostname 参数设置为该接口的 IP 地址。

如果您希望采用更严格的安全设置，Java 还提供了许多其他可用选项。例如，下面的示例使用了一组更灵活的选项启动应用程序，并将其开放到更广泛的网络，而不仅限于本地主机。

```
java \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-openjdk/management/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.ssl=true \
-Dcom.sun.management.jmxremote.registry.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /path/to/your/application.jar
```

这些设置中的大多数（如果不是全部）都可以在 $JRE/lib/management/management.properties 中指定（或者在您系统中该文件所在的位置）。

请注意，如果您希望使用 SSL，则必须修改 startup.sh 脚本，为 Java 网关添加 -Djavax.net.ssl.* 选项，以便它知道到哪里查找密钥库和信任库。

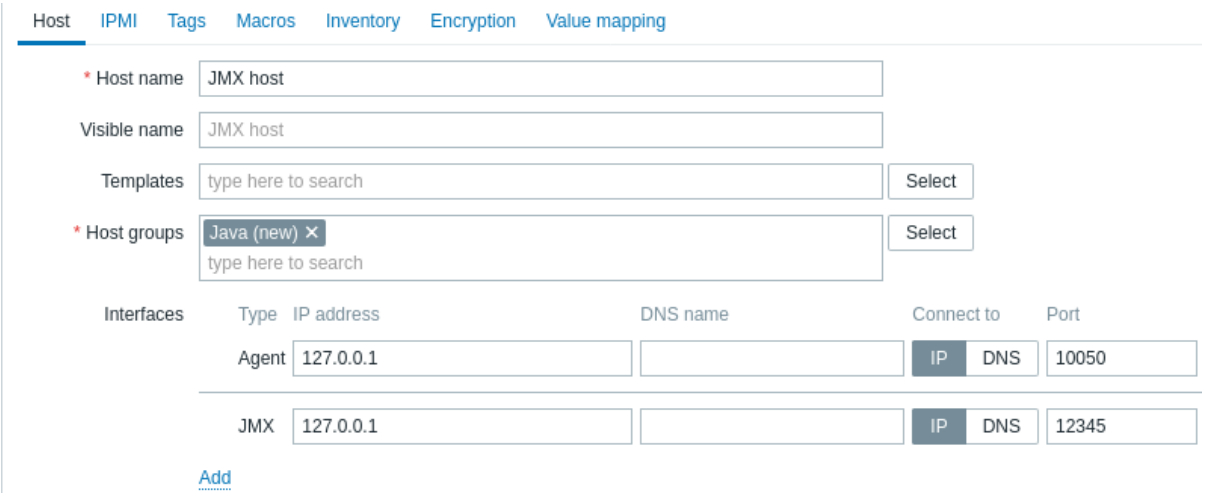
有关详细说明，请参见 使用 JMX 进行监控和管理。

在 Zabbix Web 前端页面配置 JMX 接口和监控项

Java 网关在运行时，服务器会主动连接它，Java 应用程序已启用并远程 JMX 监控，现在可以在 Zabbix GUI 中配置接口和监控项了。

配置 JMX 接口

首先，在目标主机上创建一个 JMX 类型的接口。

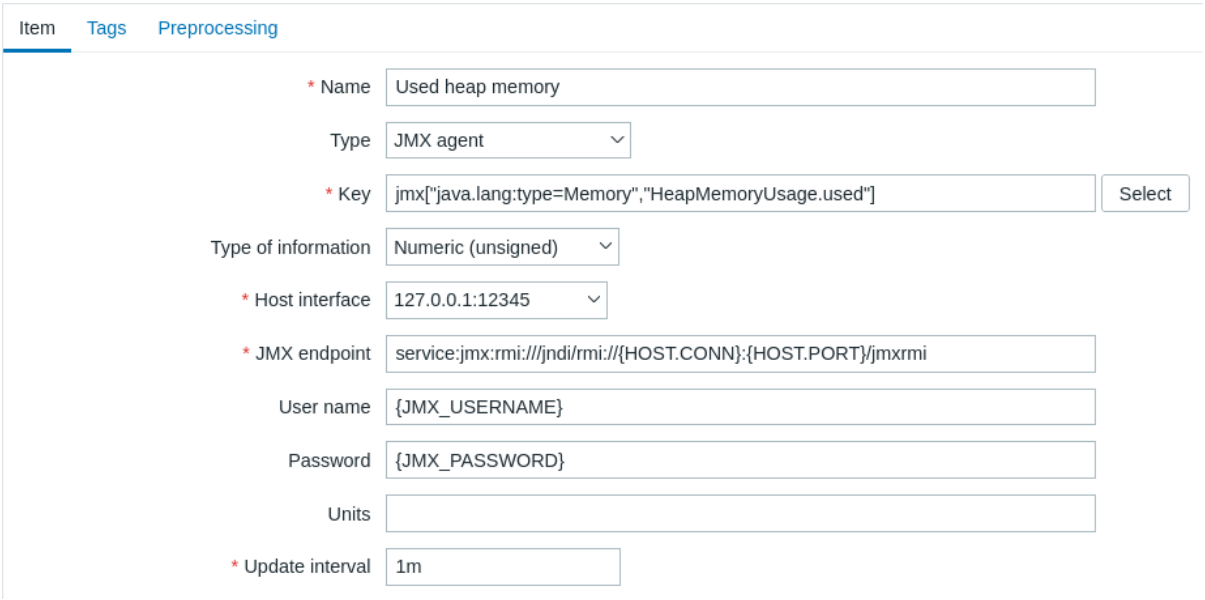


所有必填输入字段都以红色星号标记。

添加 JMX agent 监控项

对于您感兴趣的每个 JMX 计数器，都需要添加一个附加到该接口的 **JMX agent** 监控项。

下面截图中的键值为 `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`。



所有必填输入字段都用红色星号标记。

JMX 监控项需要填写的特定字段如下：

类型	在此处设置为 **JMX agent**。
键值	`jmx[]` 监控项键值包含三个参数： 对象名 - MBean 的对象名 属性名 - MBean 属性名，可选地包含以点号分隔的复合数据字段名 唯一简短描述 - 一个唯一描述，允许在主机上创建多个具有相同对象名和属性名的 JMX 监控项（可选） 有关 JMX 监控项键值的更多详细信息，请参见下文。 您可以使用 `jmx.discovery[]` 低级别发现 监控项来发现 MBean 和 MBean 属性。
JMX 端点	您可以指定自定义 JMX 端点。请确保 JMX 端点连接参数与 JMX 接口匹配。可通过使用 `{HOST.*}` 宏来实现这一点，就像默认 JMX 端点中所做的那样。 支持 `{HOST.*}` 宏 和用户宏。
用户名	如果您已在 Java 应用程序上配置身份验证，请指定用户名（最多 255 个字符）。 支持用户宏。
密码	如果您已在 Java 应用程序上配置身份验证，请指定密码（最多 255 个字符）。 支持用户宏。

如果您希望监控布尔计数器，其值为”true” 或”false”，则应将信息类型指定为”Numeric (unsigned)”，并在 Preprocessing 选项卡中选择”Boolean to decimal” 预处理步骤。服务器将分别把布尔值存储为 1 或 0。

JMX 监控项键值详解

简单属性

MBean 对象名只不过是您在 Java 应用程序中定义的字符串。另一方面，属性名可能更复杂。如果一个属性返回原始数据类型 (例如整数，字符串等)，请不要担心，监控项键值参考以下例子：

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

在这个例子中，一个对象的名称是"com.example:Type=Hello"，属性名是"weight"，可能返回值类型应该是"Numeric (float)"。

属性返回复合数据

当您的属性返回复合数据时，它变得更加复杂。例如：您的属性名称是"apple"，它返回一个表示其参数的哈希，如 "重量"、"颜色" 等。您的密钥可能看起来像这样：

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

这就是使用"." 符号将属性名称和哈希键是如何分隔开。同样，如果一个属性返回嵌套的复合数据，那部分由"." 分隔：

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

返回表格数据的属性

表格数据属性由一个或多个复合属性组成。如果在属性名参数中指定了这样一个属性，那么这个监控项值将以 JSON 格式返回该属性的完整结构。表格数据属性中的单个元素值可以使用预处理来检索。

表格数据属性示例:

```
 jmx[com.example:type=Hello,foodinfo]
```

监控项值示例:

```
[
  {
    "a": "apple",
    "b": "banana",
    "c": "cherry"
  },
  {
    "a": "potato",
    "b": "lettuce",
    "c": "onion"
  }
]
```

关于点的问题

到目前为止一切都好。当属性名或哈希键包含点，下面就是个例子:

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

这是一个问题。如何让 Zabbix 知道属性名是"all.fruits" 而不只是"all"？如何区分作为名称一部分的点号与分隔属性名和哈希键的点号？

这是可行的，你所要做的，就是使用反斜杠 () 来转义那些作为名称一部分的点号。

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

同理，如果哈希键中包含点号，你需要对它进行转义。

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

其他问题

属性名称中的反斜杠字符应进行转义：

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

如需处理 JMX 监控项键中的任何其他特殊字符，请参见监控项键格式章节。

实际上，内容就是这些。祝您 JMX 监控顺利！

非基本数据类型

可以操作自定义的 MBean，这些 MBean 返回非基本数据类型，并重写了 **toString()** 方法。

使用 JBoss EAP 6.4 的自定义端点

自定义端点允许使用除默认 RMI 之外的其他传输协议。

为了说明这一点，我们以配置 JBoss EAP 6.4 监控为例。首先，先做一些假设：

- 你已经安装了 Zabbix Java 网关。如果没有，请按照文档进行安装。
- Zabbix 服务器和 Java 网关安装在前缀目录 /usr/local/
- JBoss 已安装在 /opt/jboss-eap-6.4/，并以独立模式运行
- 我们假设所有这些组件都运行在同一台主机上
- 防火墙和 SELinux 已禁用（或已正确配置）

让我们在 zabbix_server.conf 中做一些简单设置：

```
JavaGateway=127.0.0.1
StartJavaPollers=5
```

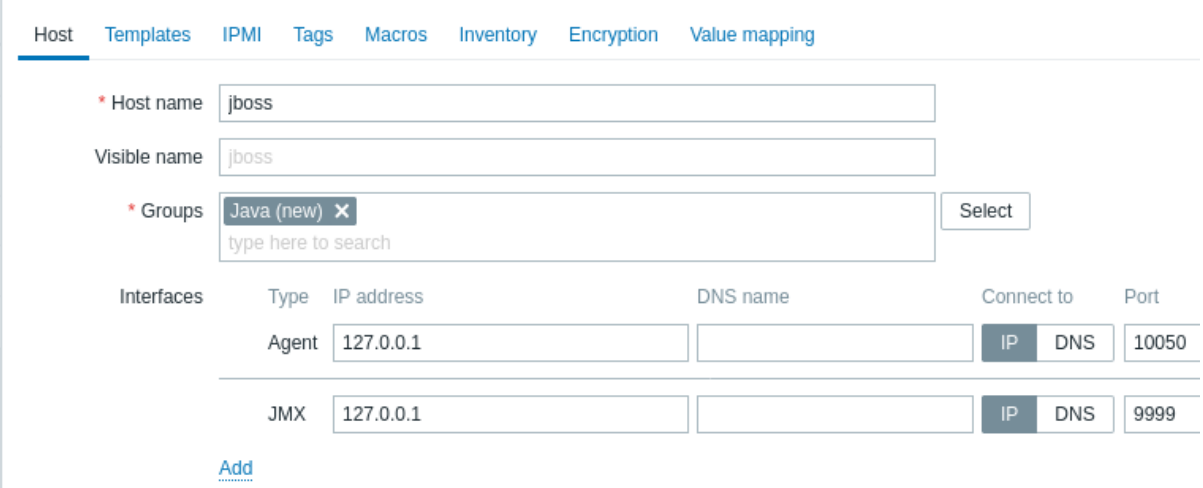
以及在 zabbix_java/settings.sh 配置文件（或 zabbix_java_gateway.conf）中：

```
START_POLLERS=5
```

检查 JBoss 是否监听其标准管理端口：

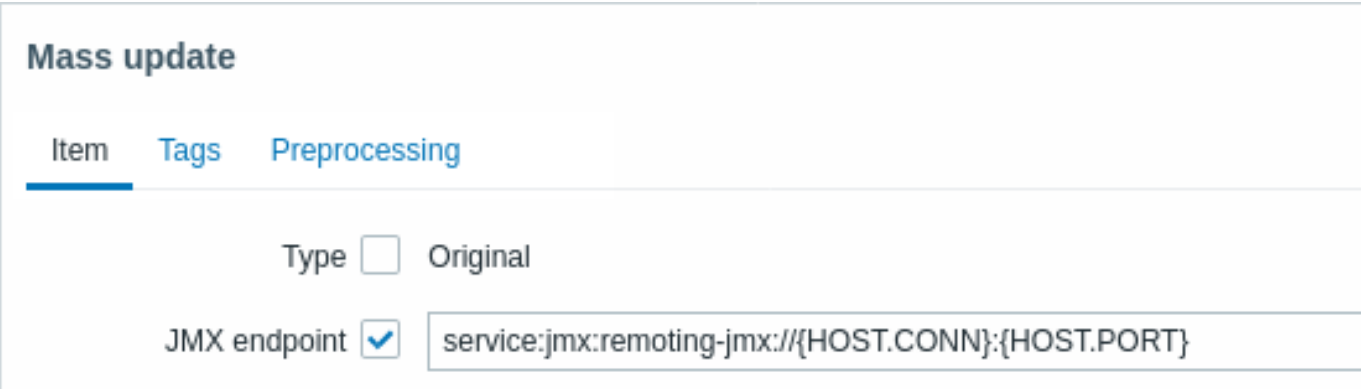
```
$ netstat -natp | grep 9999
tcp        0      0 127.0.0.1:9999          0.0.0.0:*               LISTEN      10148/java
```

现在在 Zabbix 中创建一个 JMX 接口为 127.0.0.1:9999 的主机。



由于我们知道这个版本的 JBoss 使用的是 JBoss Remoting 协议而不是 RMI，因此我们可以相应地批量更新 JMX 模板中监控项的 JMX 端点参数：

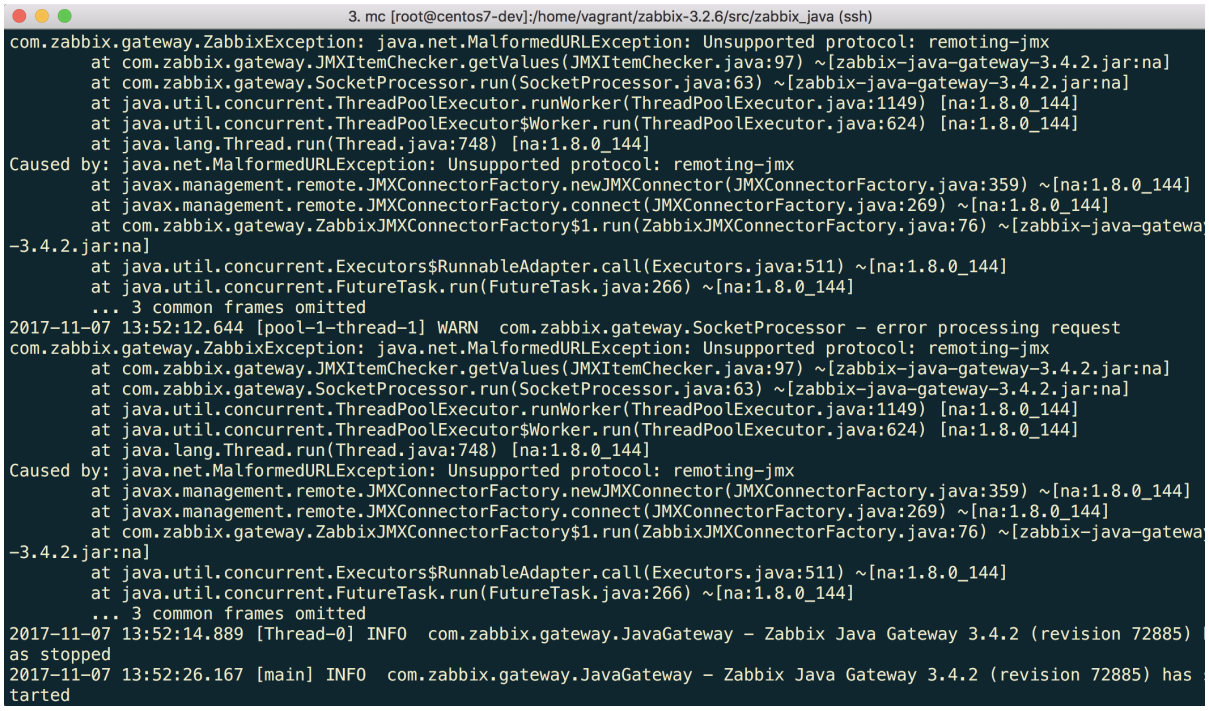
```
service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}
```



让我们更新配置缓存：

```
/usr/local/sbin/zabbix_server -R config_cache_reload
```

请注意，你可能会先遇到一个错误。



“Unsupported protocol: remoting-jmx” 表示 Java 网关不知道如何使用指定的协议。可以通过创建一个 `~/needed_modules.txt` 文件并写入以下内容来解决：

```
jboss-as-remoting
jboss-logging
jboss-logmanager
jboss-marshalling
jboss-remoting
jboss-sasl
jcl-over-slf4j
jul-to-slf4j-stub
log4j-jboss-logmanager
remoting-jmx
slf4j-api
xnio-api
xnio-nio
```

然后执行以下命令：

```
for i in $(cat ~/needed_modules.txt); do find /opt/jboss-eap-6.4 -iname "${i}*.jar" -exec cp '{}' /usr/loc
```

这样，Java 网关就会拥有与 jmx-remoting 协同工作所需的全部模块。接下来只需重启 Java 网关，稍等片刻，如果一切配置正确，就会看到 JMX 监控数据开始进入 Zabbix（另请参见：最新数据）。

计算监控项

概述

此监控项类型允许您对已存在的指标执行计算。计算型监控项本身不会收集任何数据。

例如，您可能希望：

- 计算两个值的总和。
- 计算多个监控项值的平均值。
- 计算一组监控项的总值。

计算结果会像其他监控项一样存储在 Zabbix 数据库中；历史值和趋势值都会被存储，并且可以生成图形。

计算基于公式进行。公式的语法与触发器表达式相同。

实际上，您可以将复杂的数据处理逻辑放在计算型监控项中，从而简化触发器，然后在触发器中使用监控项键引用该计算型监控项。

指定计算数据源有两种方式：

1. 在公式中指定监控项。
2. 根据筛选出监控项的过滤器执行数据聚合。

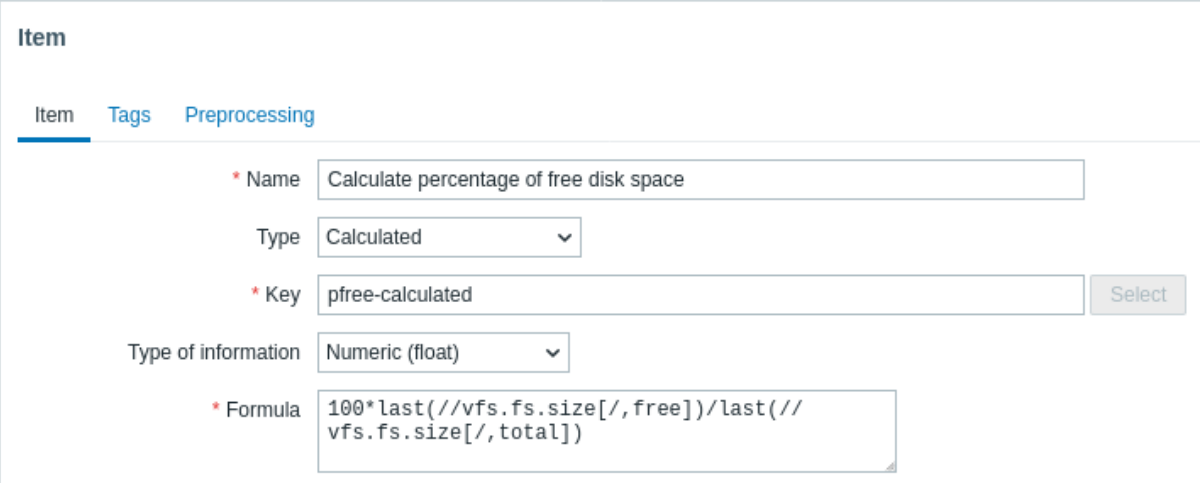
第二种方式允许您计算组平均值、总值等。

所有计算均由 Zabbix 服务器执行，并由 `history poller` 进程处理；proxy 或 agent 完全不参与。请注意，大量使用时间范围较长的计算型监控项可能会影响服务器性能。

请注意，计算型监控项可以访问其他监控项的历史数据，包括创建该计算型监控项的用户无权访问的监控项。如果不希望出现此行为，可以将`StartHistoryPollers` 配置参数设置为 0，以禁用计算型监控项。

可配置字段

要使用计算监控项，请选择监控项类型 **Calculated**。



key 是唯一的监控项标识符（每个主机内唯一）。您可以使用受支持的符号创建任意 key 名称。通过 key，您可以在触发器中引用计算监控项。

必须在 **Formula** 字段中输入计算定义。

公式与 key 之间没有关联。key 参数不会以任何方式用于公式中。

与触发器表达式不同，Zabbix 按照监控项更新间隔处理计算监控项，而不是在接收到新值时处理。

请注意，如果您选择 *Numeric (unsigned)* 作为信息类型，而计算结果为浮点值，则该值会被截断为整数。

简单计算公式

简单公式的语法为：

`function(/主机/key,<parameter1>,<parameter2>,...)`

其中：

Element	Description
`function`	支持的函数之一：last、min、max、avg、count 等
`host`	用于计算的监控项所属的主机。 可省略当前主机（即如 `function(//key,parameter,...)` 所示）。
`key`	用于计算的监控项的键。 不支持返回二进制或 JSON 数据类型值的监控项。 如果更改了被引用监控项的键，则必须在公式中手动更新该键。 历史函数引用的所有监控项都必须存在并且正在采集数据。
`parameter(s)`	函数的参数（如需要）。 支持时间后缀和内存大小后缀。

更复杂的公式可以组合使用函数、运算符和括号。

请参见计算型监控项公式示例：

- (1) 接口总流量
- (2) 可用磁盘空间百分比
- (3) 入站流量百分比
- (4) 已处理值的平均数量

某些变量可通过用户宏传递给公式。
请注意，用户宏：

- 将被解析——如果用于引用函数参数、监控项过滤器参数或常量。
- 不会被解析——如果用于引用函数、主机名、监控项键、监控项键参数或运算符。

允许与字符串进行比较。

如果缓存中没有最近的数据，并且函数中未定义查询时间段，Zabbix 默认会向过去最多回溯一周，以查询数据库中的历史值。

计算型监控项在以下几种情况下可能变为不支持：

- 被引用的监控项
  - 未找到
  - 已禁用，或属于已禁用的主机
  - 不受支持（nodata() 函数以及带有未知值的运算符除外）
- 没有可用于计算函数的数据
- 除以零
- 使用了错误的语法

数据聚合

数据聚合基于一个用于选择多个监控项的过滤器。数据聚合也是一种计算型监控项，不过在公式中使用的语法略有不同。

要获取聚合值，请使用支持的聚合函数之一：avg、max、min、sum 等。然后添加 **foreach** 函数作为唯一参数，以及其监控项过滤器来选择所需的监控项：

```
aggregate_function(function_foreach(/host/key?[group="host group"],timeperiod))
```

foreach 函数（例如 *avg_foreach*、*count_foreach* 等）会为每个选定的监控项返回一个聚合值。监控项通过监控项过滤器（/host/key?[group="host group"]）从监控项历史数据中选出。更多详细信息，请参见foreach 函数。

如果某些监控项在请求的时间段内没有数据，则在计算时会忽略它们。如果没有任何监控项有数据，则该函数将返回错误。

或者，您也可以将多个监控项列为聚合的参数：

```
aggregate_function(function(/host/key,parameter),function(/host2/key2,parameter),...)
```

请注意，这里的 function 必须是历史/趋势函数。

请参见聚合计算示例：

- (5) 主机的总入站流量
- (6) 组的总磁盘空间
- (7) 组的平均处理器负载
- (8) 组和标签的平均处理器负载
- (9) 已处理值的平均数量
- (10) 组的平均查询数量
- (11) 成功 DNS 检查的总数
- (12) 组中不受支持监控项的总数
- (13) 组的接口统计

以下位置支持用户宏和低级发现宏：

- 监控项键参数
- 函数参数
- 过滤条件（主机组名称和标签名称）
- 表达式常量

在以下情况下，聚合计算可能变为不受支持：

- 找不到任何被引用的监控项（这可能发生在监控项键不正确、监控项均不存在，或所有包含的组都不正确时）
- 没有可用于计算函数的数据

计算监控项示例

(1) 接口上的总流量

```
last(//net.if.in[eth0,bytes])+last(//net.if.out[eth0,bytes])
```

计算 eth0 上的总带宽。

(2) 可用磁盘空间百分比

```
100*last(//vfs.fs.size[/,free])/last(//vfs.fs.size[/,total])
```

Zabbix 将获取 / 上可用磁盘空间和总磁盘空间的最新值，并根据给定公式计算百分比。

(3) 入站流量百分比

```
100*last(//net.if.in[eth0,bytes])/(last(//net.if.in[eth0,bytes])+last(//net.if.out[eth0,bytes]))
```

计算总流量中入站流量所占的百分比。

(4) 已处理值的平均数量

```
avg(/Zabbix server/zabbix[wcache,values],10m)
```

计算 Zabbix 在 10 分钟内处理的值数量的平均值。

(5) 主机的总入站流量

```
sum(last_foreach(/host/net.if.in[*]))
```

计算主机上所有匹配 net.if.in[*] 的监控项总和。

(6) 组的总磁盘空间

```
sum(last_foreach(/*/vfs.fs.size[/,total]?[group="MySQL Servers"]))
```

计算所有 MySQL Servers 的总磁盘空间。

(7) 组的平均处理器负载

```
avg(last_foreach(/*/system.cpu.load[,avg1]?[group="MySQL Servers"]))
```

计算所有 MySQL Servers 的平均处理器负载。

(8) 组和标签的平均处理器负载

```
avg(last_foreach(/*/system.cpu.load?[(group="Servers A" or group="Servers B" or group="Servers C") and (ta
```

计算多个主机组中具有特定标签的所有主机上的平均 CPU 负载。

(9) 组的平均查询数

```
avg(avg_foreach(/*/mysql.qps?[group="MySQL Servers"],5m))
```

计算 MySQL Servers 每秒查询数的 5 分钟平均值。

(10) 成功 DNS 检查总数

```
sum(last_foreach(/*/net.dns[*,*,*]))
```

计算所有主机上的成功 DNS 检查总数。所示监控项形式为 net.dns[192.0.2.1,example.com,A]，作为一种可能的键值示例。

请注意，通配符必须与键中的参数数量匹配（此处net.dns 有三个参数：ip、name、type）。

(11) 组中不受支持监控项的总数

```
sum(last_foreach(/*/zabbix[host,,items_unsupported]?[group="Zabbix servers"]))
```

计算 Zabbix servers 的不受支持监控项总数。

(12) 组的接口统计信息

```
sum(last_foreach(/*/net.if.out[eth0,bytes]?[group="video"])) / sum(last_foreach(/*/nginx_stat.sh[active]?[
```

计算 video 组的接口统计信息。

正确/错误语法示例

表达式（包括函数调用）不能用作历史数据、趋势数据或 foreach 函数的参数。但是，这些函数本身可以用作其他（非历史数据）函数的参数。

表达式	示例

有效	`avg(last(/host/key1),last(/host/key2)*10,last(/host/key1)*100)`
	
	`max(avg(avg_foreach(/*/system.cpu.load?[group="Servers A"],5m)),avg(avg_foreach(/*/system.cpu.load?[group="Servers B"],5m)),avg(avg_foreach(/*/system.cpu.load?[group="Servers C"],5m)))`

表达式	示例

无效	`sum(/host/key,10+2)`
	
	`sum(/host/key, avg(10,2))`
	
	`sum(/host/key,last(/host/key2))`

请注意，在如下表达式中：

```
sum(sum_foreach(//resptime[*],5m))/sum(count_foreach(//resptime[*],5m))
```

(其中，`resptime[*]` 是一个监控项键模式示例，可匹配 `resptime[1]`、`resptime[2]` 等键，用于获取多个活动集群节点的响应时间)，无法保证等式两部分始终具有相同的值集合。在计算表达式的一部分时，请求时间段内可能会收到新值，随后表达式的另一部分将具有不同的值集合。

依赖监控项

概述

在某些情况下，一个监控项会同时采集多个指标，或者同时收集相关指标会更有意义，例如：

- 各个 CPU 核心的利用率
- 入站/出站/总网络流量

为了支持批量采集指标并在多个相关监控项中同时使用，Zabbix 支持依赖监控项。依赖监控项依赖于主监控项，后者通过一次查询同时采集它们的数据。主监控项的新值会自动填充到依赖监控项的值中。依赖监控项不能设置与主监控项不同的更新间隔。

可以使用 Zabbix 预处理选项，从主监控项数据中提取依赖监控项所需的部分。

预处理由 `preprocessing manager` 进程管理，并配合执行预处理步骤的工作线程一起运行。所有带有预处理的值在由不同的数据采集器接收后，都会先经过预处理管理器，然后才会被添加到历史缓存中。数据采集器（poller、trapper 等）与预处理进程之间使用基于套接字的 IPC 通信。

Zabbix 服务器或 Zabbix proxy（如果主机由 proxy 监控）负责执行预处理步骤并处理依赖监控项。

任何类型的监控项，甚至依赖监控项本身，都可以设置为主监控项。还可以使用更多层级的依赖监控项，从现有依赖监控项的值中提取更小的部分。

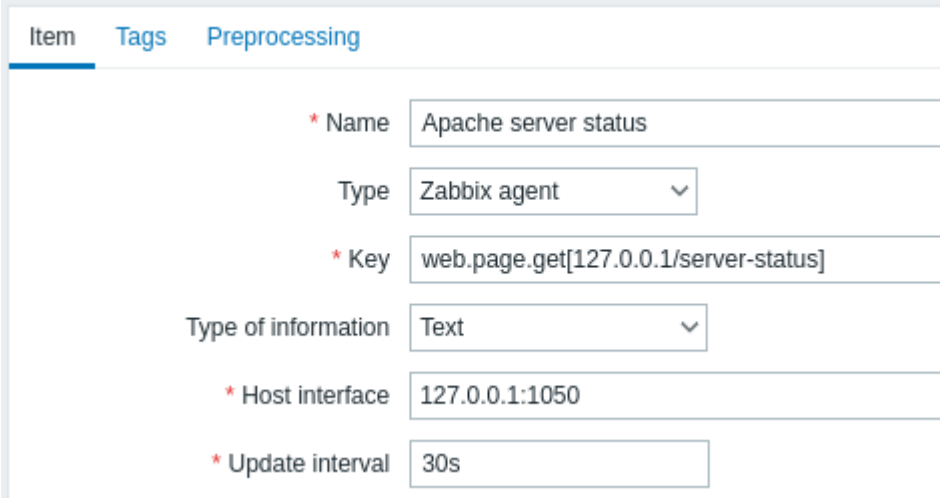
限制

- 仅允许相同主机（模板）上的依赖关系
- 监控项原型可以依赖于同一主机上的另一个监控项原型或常规监控项
- 如果主机上的从属监控项的主监控项来自模板，则该从属监控项不会被导出到 XML

监控项配置

依赖监控项的数据来自其主监控项。因此，必须先配置（或存在）主监控项：

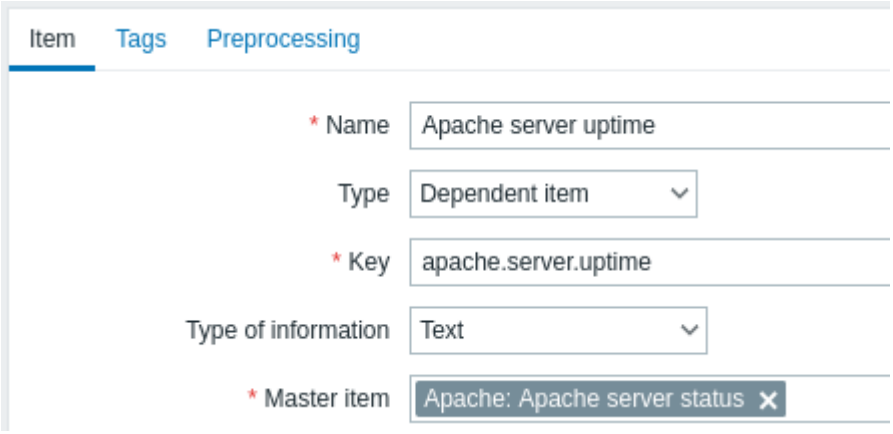
1. 转到 数据采集 > 主机。
2. 在主机所在行中点击 监控项。
3. 点击 创建监控项。
4. 在表单中输入监控项参数。



所有必填输入字段都用红色星号标记。

点击 添加保存主监控项。

然后，您就可以配置依赖监控项了。

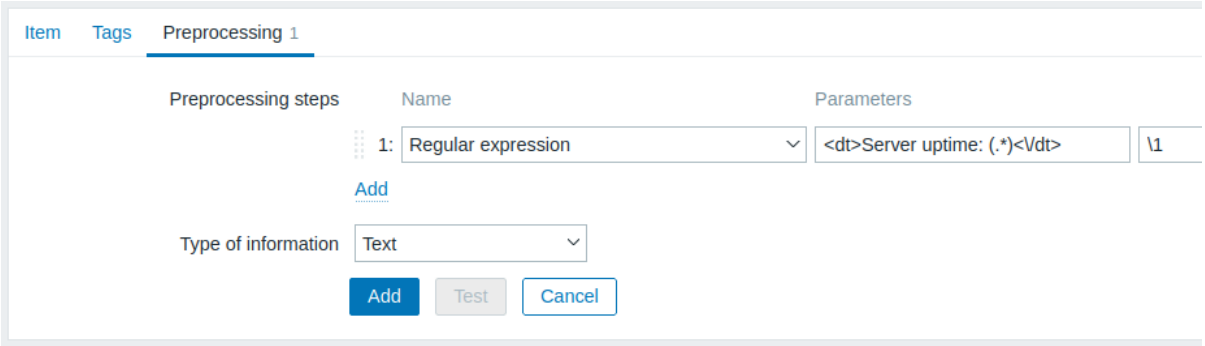


所有必填输入字段都用红色星号标记。

依赖监控项需要填写的特定字段如下：

类型	在此选择 依赖监控项。
键值	输入一个用于识别该监控项的键值。
主监控项	选择主监控项。主监控项的值将用于填充依赖监控项的值。
信息类型	选择与将要存储的数据格式相对应的信息类型。

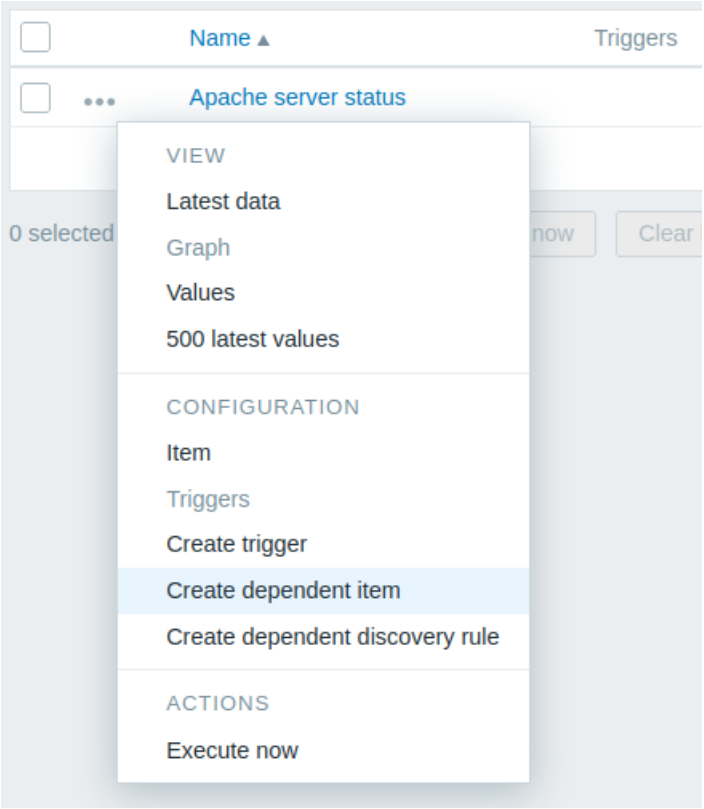
您可以使用监控项值预处理 来提取主监控项值中所需的部分。



如果不进行预处理，依赖监控项的值将与主监控项的值完全相同。

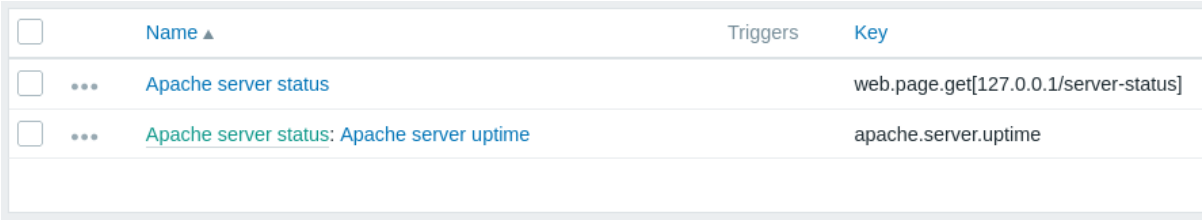
点击 添加保存依赖监控项。

您还可以通过在监控项列表中点击 ••• 按钮并选择 创建依赖监控项，更快捷地创建依赖监控项。



展示

在监控项列表中，依赖监控项会以它们的主监控项名称作为前缀显示。



如果主监控项被删除，所有依赖于它的依赖监控项也会被删除。

脚本监控项

概览

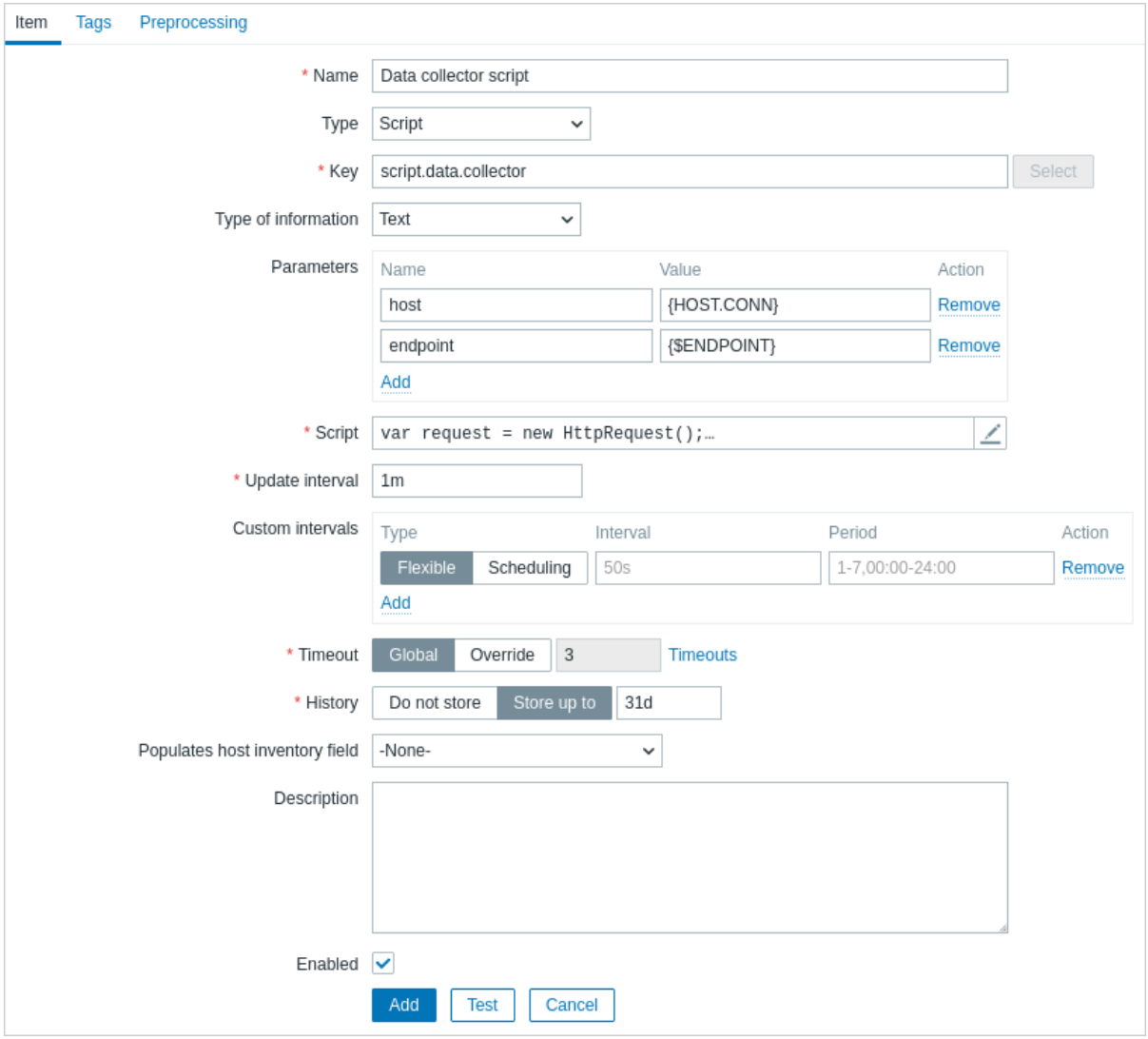
脚本监控项可以通过执行用户自定义的 JavaScript 代码，检索 HTTP/HTTPS 的方式来收集数据。除了脚本外，可以指定一个可选的参数列表（一些键值对）并配置超时限制。

此监控项类型在有收集数据过程中需要多个步骤或复杂逻辑的场景非常有用。举个例子，一个脚本监控项可以被配置为执行一个 HTTP 调用，然后经过某些方式处理从第一步调用得到的数据，并将转换后的数值传递给第二个 HTTP 调用。

脚本监控项由 Zabbix server 或 proxy 的轮询器处理。

配置

在监控项配置表单的 *Type* 字段中选择 Script，然后填写必填字段。



所有必填输入字段都用红色星号标记。

Script 监控项需要填写特定信息的字段如下：

Field	Description
Key	输入一个唯一的键值，用于标识该监控项。
Parameters	指定要传递给脚本的变量，采用属性和值对的形式。 支持用户宏。要查看支持哪些内置宏，请在支持的宏表中搜索 “Script-type item”。
Script	在参数字段中单击，或单击其旁边的铅笔图标时打开的模态编辑器中输入 JavaScript 代码。该代码必须提供返回指标值的逻辑。 该代码可以访问所有参数，以及 Zabbix 添加的所有附加 JavaScript 对象。 另请参见：JavaScript Guide。
Timeout	JavaScript 执行超时（1-600 秒；超过该时间将返回错误）。 请注意，根据脚本的不同，触发超时可能需要更长时间。 有关 *Timeout* 参数的更多信息，请参见通用监控项属性。

示例

简单的数据采集

从 *https://www.example.com/release_notes* 页面收集内容:

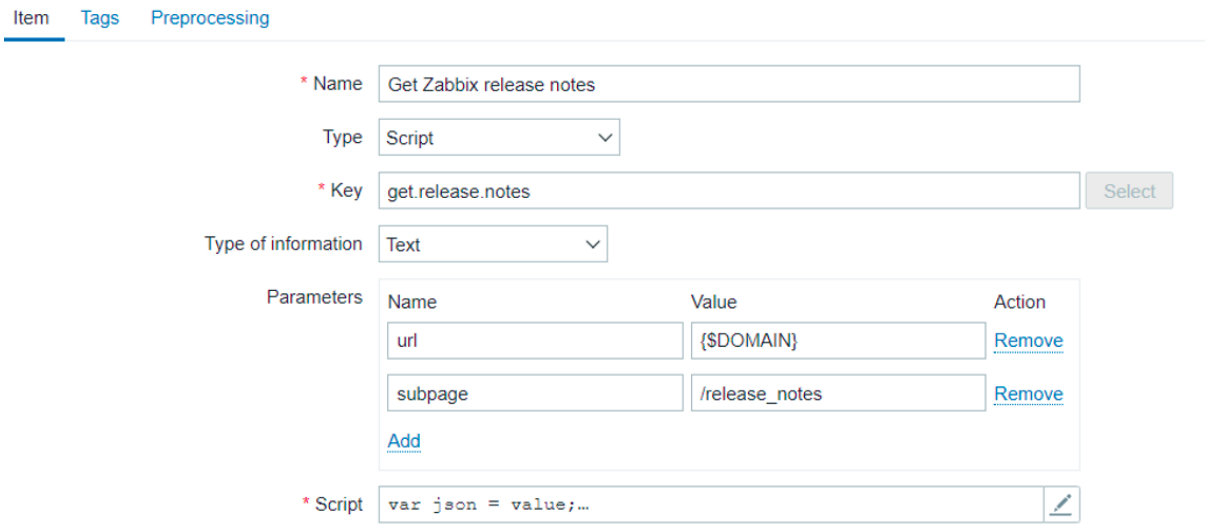
- 创建一个监控项，类型选择” 脚本 “。
- 在脚本字段填写下面的代码：

```
var request = new HttpRequest();
return request.get("https://www.example.com/release_notes");
```

带参数收集数据

收集特定页面的内容并使用参数：

- 创建一个" 脚本" 类型的项目，并设置两个参数：
  - **url：{$DOMAIN}**（应定义用户宏 {$DOMAIN}，最好是在主机级定义）
  - 子页面：/release_notes**



- 在 *Script* 字段中输入

```
var obj = JSON.parse(value);
var url = obj.url;
var subpage = obj.subpage;
var request = new HttpRequest();
return request.get(url + subpage);
```

多个 HTTP 请求

同时收集 *https://www.example.com* 和 *https://www.example.com/release_notes*：

- 创建一个类型为" 脚本" 的监控项。
- 在 *Script* 字段中输入

```
var request = new HttpRequest();
return request.get("https://www.example.com") + request.get("https://www.example.com/release_notes");
```

日志

在 Zabbix 服务器日志中添加" 日志测试" 项并接收监控项值"1"：

- 创建一个类型为" 脚本" 的监控项。
- 在 *Script* 字段中输入

```
Zabbix.log(3, 'Log test');
return 1;
```

浏览器监控项

概述

Browser 监控项允许使用浏览器监控复杂的网站和 Web 应用程序。

> **Attention:**
> Browser 监控项的支持目前处于实验阶段。

Browser 监控项通过执行用户定义的 JavaScript 代码并通过 HTTP/HTTPS 获取数据来收集数据。此监控项可以模拟与浏览器相关的操作，例如点击、输入文本、浏览网页，以及其他与网站或 Web 应用程序的用户交互。

除了脚本之外，还可以指定一个可选的参数列表（名称和值对）以及超时时间。

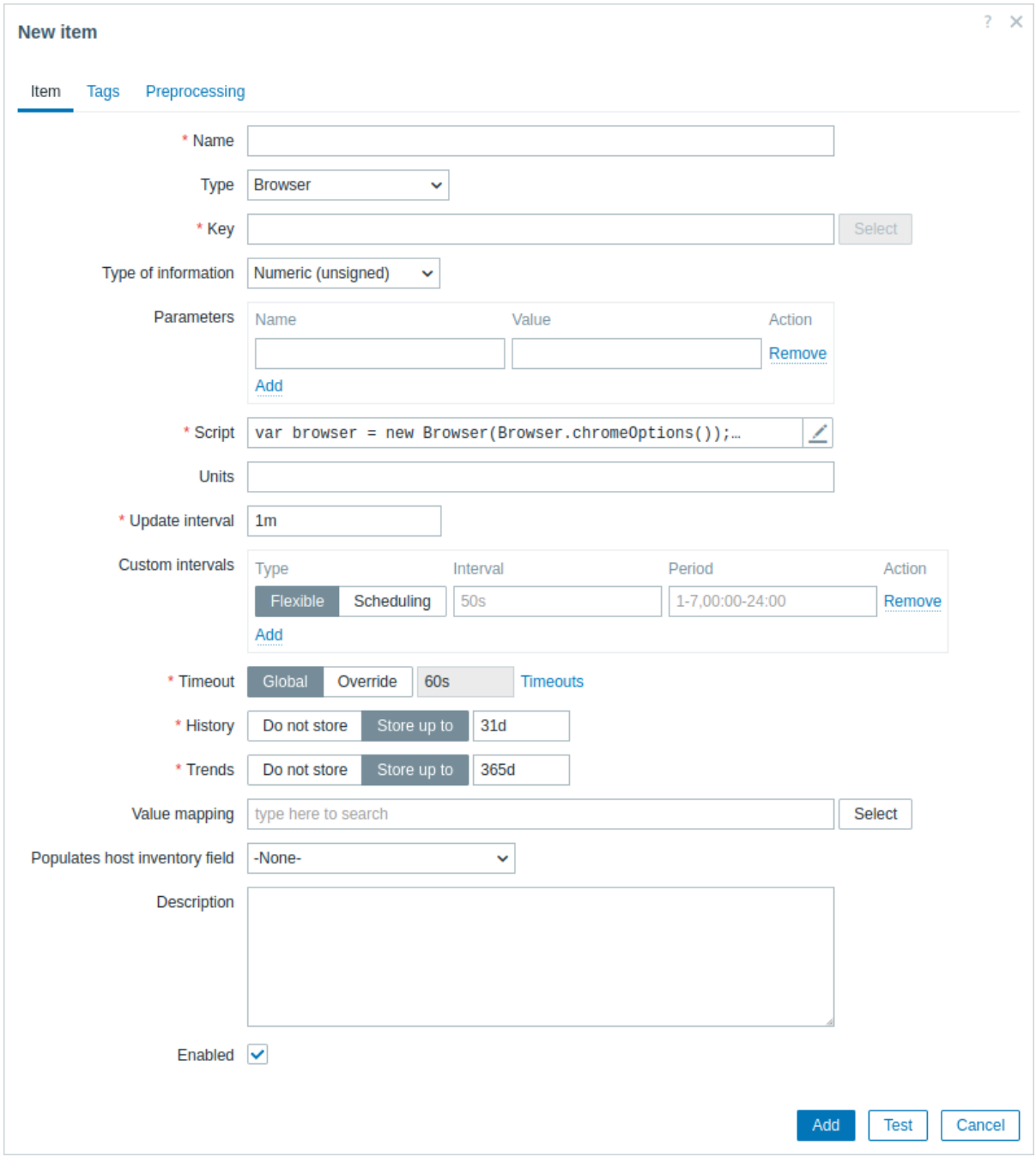
Attention:
该监控项通过 Selenium Server 或普通 WebDriver（例如 ChromeDriver）作为 Web 测试端点，部分实现了 W3C WebDriver 标准。要使该监控项正常工作，请在 Zabbix 服务器/proxy 配置文件中设置 `WebDriverURL` 参数（如果使用 ChromeDriver，请参见 安全注意事项）。为了获得更好的性能，建议为 Web 测试环境使用专用服务器。

Browser 监控项检查由 Zabbix 服务器或 proxy 的 browser poller 执行和处理。如有必要，您可以在 Zabbix 服务器/proxy 配置文件中调整 `StartBrowserPollers` 参数，以修改 browser poller 的预先 fork 实例数量。

对于监控复杂的网站和 Web 应用程序，可使用 Website by Browser 模板，作为开箱即用模板。

配置

在监控项配置表单的 *Type* 字段中，选择 Browser，然后填写必填字段。



所有必填输入字段都以红色星号标记。

Browser 监控项需要填写特定信息的字段如下：

字段	描述
Key	输入一个唯一的键值，用于标识该监控项。
Parameters	指定要作为属性和值对传递给脚本的变量。 支持用户宏。要查看支持哪些内置宏，请在支持的宏表中搜索 “Browser-type item”。

字段	描述
Script	在参数字段中单击或单击其旁边的铅笔图标时，会打开模态编辑器，在其中输入 JavaScript 代码。该代码必须提供返回指标值的逻辑。 该代码可以访问所有参数、Zabbix 添加的所有附加 JavaScript 对象以及Browser 监控项 JavaScript 对象。 另请参见：JavaScript 指南。
Timeout	JavaScript 执行超时（1-600 秒；超过该时间将返回错误）。 请注意，根据脚本的不同，触发超时可能需要更长时间。 有关 *Timeout* 参数的更多信息，请参见监控项通用属性。

示例

有关如何使用 *Website by Browser* 模板设置 Zabbix 以监控网站的示例，请参见使用 Browser 监控项监控网站。

默认脚本

以下脚本：

1. 初始化一个浏览器会话。
2. 导航到指定的 URL。
3. 收集性能条目和会话统计数据，并以 JSON 字符串的形式返回它们。

在脚本字段中，输入以下内容：

```
var browser = new Browser(Browser.chromeOptions());

try {
    browser.navigate("http://example.com");
    browser.collectPerfEntries();
}
finally {
    return JSON.stringify(browser.getResult());
}
```

使用自定义 capabilities 初始化浏览器

以下脚本：

1. 根据脚本中指定的顺序，基于第一个匹配的可用浏览器初始化浏览器会话。
2. 定义浏览器 capabilities，包括页面加载策略以及各浏览器特有的选项，例如 Chrome、Firefox 和 Microsoft Edge 浏览器的无头模式。

在 *Script* 字段中，输入：

```
var browser = new Browser({
    "capabilities":{
        "firstMatch":[
            {
                "browserName":"chrome",
                "pageLoadStrategy":"normal",
                "goog:chromeOptions":{
                    "args":[
                        "--headless=new"
                    ]
                }
            },
            {
                "browserName":"firefox",
                "pageLoadStrategy":"normal",
                "moz:firefoxOptions":{
                    "args":[
                        "--headless"
                    ]
                }
            },
            {
                "browserName":"MicrosoftEdge",
```

```
                "pageLoadStrategy":"normal",
                "ms:edgeOptions":{
                    "args":[
                        "--headless=new"
                    ]
                }
            },
            {
                "browserName":"safari",
                "pageLoadStrategy":"normal"
            }
        ]
    }
});
```

使用 GUI 初始化浏览器

默认情况下，浏览器会话（Safari 除外）以无头模式初始化，这意味着不会显示浏览器的图形用户界面（GUI）。

以下脚本会初始化一个启用了 GUI 的浏览器会话。

请注意，如果 WebDriver 无法定位浏览器二进制文件，您可以手动指定其路径。

```
var opts = Browser.chromeOptions();
opts.capabilities.alwaysMatch['goog:chromeOptions'].args = [];

// To initialize a Firefox session with GUI, uncomment the following lines:
// var opts = Browser.firefoxOptions();
// opts.capabilities.alwaysMatch['moz:firefoxOptions'].binary = 'usr/bin/firefox';
// opts.capabilities.alwaysMatch['moz:firefoxOptions'].args = [];

// To initialize a Microsoft Edge session with GUI, uncomment the following lines:
// var opts = Browser.edgeOptions();
// opts.capabilities.alwaysMatch['ms:edgeOptions'].binary = 'usr/bin/microsoft-edge';
// opts.capabilities.alwaysMatch['ms:edgeOptions'].args = [];

var browser = new Browser(opts);
```

> **Note:**
> 如果您的测试在远程服务器上运行或在容器中运行，您可以使用虚拟网络计算（VNC）客户端连接到该机器的 VNC 服务器。这样您就可以远程查看并与浏览器的 GUI 进行交互。

截取屏幕截图

以下脚本：

1. 初始化浏览器会话。
2. 设置浏览器视口大小以确定屏幕截图大小（作为参数指定，见下文）。
3. 导航到某个 URL（作为参数指定，见下文）。
4. 收集会话统计信息，截取屏幕截图，并将其添加到已收集的统计信息中。
5. 通过捕获错误消息和屏幕截图来处理错误。
6. 以 JSON 字符串形式返回收集到的结果。

该脚本还使用来自监控项配置表单的参数：

- webURL - http://example.com
- width - 1920
- height - 1080

在 *Script* 字段中，输入：

```
var browser, result;

var browser = new Browser(Browser.chromeOptions());

try {
    var params = JSON.parse(value); // 解析包含从 Zabbix 传递参数的 JSON 字符串。
```

```
    browser.setScreenSize(Number(params.width), Number(params.height))

    browser.navigate(params.webURL);

    result = browser.getResult();
    result.screenshot = browser.getScreenshot();
}
catch (err) {
    if (!(err instanceof BrowserError)) {
        browser.setError(err.message);
    }
    result = browser.getResult();
    result.error.screenshot = browser.getScreenshot();
}
finally {
    return JSON.stringify(result);
}
```

检查 Zabbix 登录

以下脚本：

1. 初始化浏览器会话。
2. 导航到某个页面（作为参数指定，见下文）。
3. 输入用户名和密码（作为参数指定，见下文）。
4. 查找并点击登录按钮。
5. 查找并点击注销按钮。
6. 在登录前后以及注销后收集性能数据。
7. 通过捕获错误消息和屏幕截图来处理错误。
8. 将收集到的结果作为 JSON 字符串返回。

该脚本还使用来自监控项配置表单的参数：

- webURL - http://{HOST.CONN}/index.php
- username - {$USERNAME}
- password - {$PASSWORD}

在 *Script* 字段中，输入：

```
var browser, result;

browser = new Browser(Browser.chromeOptions());

try {
    var params = JSON.parse(value); // Parse the JSON string containing parameters passed from Zabbix.

    browser.navigate(params.webURL);
    browser.collectPerfEntries("open page");

    var el = browser.findElement("xpath", "//input[@id='name']");
    if (el === null) {
        throw Error("cannot find name input field");
    }
    el.sendKeys(params.username);

    el = browser.findElement("xpath", "//input[@id='password']");
    if (el === null) {
        throw Error("cannot find password input field");
    }
    el.sendKeys(params.password);

    el = browser.findElement("xpath", "//button[@id='enter']");
    if (el === null) {
        throw Error("cannot find login button");
    }
```

```
    el.click();

    browser.collectPerfEntries("login");

    el = browser.findElement("link text", "Sign out");
    if (el === null) {
        throw Error("cannot find logout button");
    }
    el.click();

    browser.collectPerfEntries("logout");

    result = browser.getResult();
}
catch (err) {
    if (!(err instanceof BrowserError)) {
        browser.setError(err.message);
    }
    result = browser.getResult();
    result.error.screenshot = browser.getScreenshot();
}
finally {
    return JSON.stringify(result);
}
```

查找链接

以下脚本：

1. 初始化浏览器会话。
2. 定义一个用于从数组中移除重复元素的函数（见步骤 5）。
3. 导航到一个页面（作为参数指定，见下文）。
4. 查找页面中的链接。
5. 移除重复链接以确保其唯一性。
6. 仅提取以 "http" 开头的链接。
7. 将提取的链接格式化为特定结构。
8. 通过捕获错误消息和截图来处理错误。
9. 以 JSON 字符串形式返回收集到的结果。

该脚本还使用来自监控项配置表单的参数：

- scheme - {$WEBSITE.SCHEME}
- domain - {$WEBSITE.DOMAIN}
- path - {$WEBSITE.PATH}

在 *Script* 字段中，输入：

```
var browser, result;

browser = new Browser(Browser.chromeOptions());

try {
    var params = JSON.parse(value);  // Parse the JSON string containing parameters passed from Zabbix.

    function uniq(a) {
        return a.sort().filter(function (item, pos, ary) {
            return !pos || item != ary[pos - 1];
        });
    }

    browser.navigate(params.scheme + '://' + params.domain + params.path);

    var el = browser.findElements("link text", "");
    var links = [];
    for (var n = 0; n < el.length; n++) {
        links.push(el[n].getAttribute('href'));
```

```
    }

    links = uniq(links);

    result = [];
    for (i = 0; i < links.length; i++) {
        if (links[i].match(/^http.*/)) {
            var row = {};
            row["{#URL}"] = links[i];
            result.push(row);
        }
    }
}
catch (err) {
    if (!(err instanceof BrowserError)) {
        browser.setError(err.message);
    }
    result = browser.getResult();
    result.error.screenshot = browser.getScreenshot();
}
finally {
    return JSON.stringify(result);
}
```

遥测查询

此监控项类型用于检索或聚合遥测数据。遥测数据用于应用性能监控（APM），可帮助您了解请求如何流经应用程序和服务、衡量其性能，并找出操作缓慢或失败的原因。

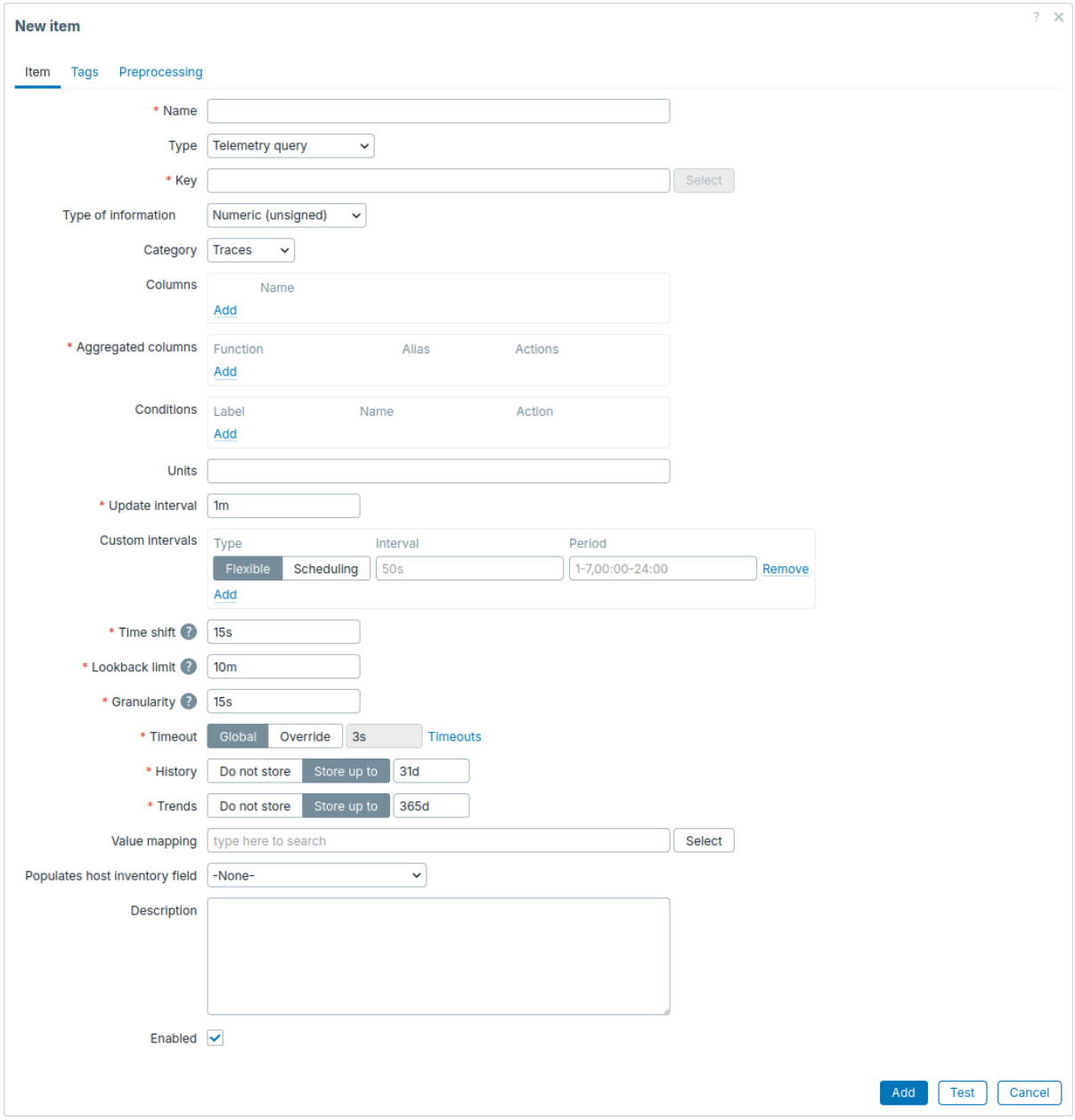
遥测数据分为三种类型：追踪（请求通过应用程序和服务所经过的路径）、指标（例如响应时间等测量值）以及日志（事件的文本记录）。通过使用遥测查询监控项，您可以查找服务返回错误的所有追踪、获取指标中的平均响应时间，或在日志中搜索特定的错误消息。

配置遥测查询监控项时，您需要定义一组指令（查询），用于告知 Zabbix 要请求哪些遥测数据、如何筛选这些数据以及如何聚合这些数据。执行监控项检查时，Zabbix 会将此查询发送到遥测数据源，并根据查询检索结果。

要使用此监控项，请指定遥测数据源，并在Zabbix 服务器 或Zabbix proxy 配置中设置 `StartTelemetryQueryPollers` 参数，具体取决于监控该监控项所属主机的是哪一个。

配置

创建监控项时，将其类型设置为遥测查询：



参数	描述
名称	请参阅所有监控项通用的参数。
类型	选择遥测查询。
键值	请参阅所有监控项通用的参数。
信息类型	
类别	选择要查询的信号类型：追踪、指标或日志。
指标点	选择要查询的指标点类型：总和、仪表、直方图或指数直方图。 当类别设置为指标时，此参数可用。
列	添加要查询的遥测数据列。对于存储键值对的列，还需输入键名。这些列中的记录将按原样包含在监控项值中。
聚合列	添加要查询的遥测数据列，以及从其记录中计算值的方式： 函数 - 选择要应用于列记录的函数：*min*、*max*、*avg*、*count*、*sum* 或 *percentile*。 百分位数 - 输入要从列记录中计算的百分位数（当函数设置为 *percentile* 时）。 列 - 选择要进行计算的列（当函数未设置为 *count* 时）。 别名 - 输入计算值的唯一名称，用于在监控项值中标识该值。

参数	描述
计算类型	选择如何评估以下条件参数中设置的条件： **And/Or** - 必须满足所有条件；具有相同类型和属性键的条件将通过 *Or* 组合。 **And** - 必须满足所有条件。 **Or** - 至少满足一个条件。 自定义表达式 - 必须按照公式中指定的方式满足条件，例如 `A and (B or C)`。 设置了两个或更多条件时，此参数可用。
条件	添加用于筛选遥测数据列记录的条件： 类型 - 选择条件所应用的遥测数据列。 属性键 - 输入条件所应用的键（当类型设置为包含键值对的列时）。 运算符 - 选择记录值与属性值之间要使用的比较方式（见下文）。 属性值 - 输入用于与记录值进行比较的值（当运算符未设置为 *Exists* 时）。 当类型设置为单值列时，条件将应用于该列的记录；如果该列包含值数组，则当数组中至少有一个值满足条件时，记录将匹配。 当类型设置为包含键值对的列时，条件将应用于属性键中指定的键的值；如果该列包含键值对数组，则当至少有一个包含指定属性键的键值对满足条件时，记录将匹配。 对于单值列，可以使用以下运算符： 等于 - 匹配等于属性值的记录（区分大小写）。 不等于 - 匹配不等于属性值的记录（区分大小写）。 包含 - 匹配包含属性值的记录（区分大小写）。 不包含 - 匹配不包含属性值的记录（区分大小写）。 对于包含键值对的列，可以使用以下运算符： 等于 - 匹配存在属性键且其值等于属性值的记录（区分大小写）。 不等于 - 匹配存在属性键且其值不等于属性值的记录（区分大小写）。 存在 - 匹配存在属性键的记录（区分大小写）。
单位	请参阅所有监控项通用的参数。
更新间隔	
自定义间隔	
时间偏移	输入监控项查询的时间范围应在当前时间之前结束多长时间。支持秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d）或用户宏。 每次查询都会请求一个时间范围内的数据。该范围通常从上一次查询的时间范围结束处开始，并在时间偏移设置的时间点结束。 如果上一次查询的时间范围结束时间晚于当前时间减去粒度，则当前查询的时间范围将从当前时间减去粒度处开始，因此会与上一次查询的时间范围重叠。 如果没有可继续的上一次查询，则该范围从回溯限制设置的时间点开始。 如果此监控项的主机由其他实体监控（例如来自proxy 组的其他 proxy），则该监控项可能无法再确定上一次查询的结束时间。这可能导致值重复。
回溯限制	当没有可继续的上一次查询时（例如监控项首次运行时），或上一次查询早于更新间隔时（例如 Zabbix 服务器宕机时），输入监控项查询数据时应回溯到当前时间之前多长时间。支持秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d）或用户宏。 如果遥测数据源包含大量数据（例如每小时超过一百万条记录），当监控项使用较大的回溯限制时，性能可能会显著降低。
粒度	输入将查询的完整时间范围拆分成的时间片长度。支持秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d）或用户宏。
超时	请参阅所有监控项通用的参数。
历史数据	
趋势数据	
值映射	
填充主机资产清单字段	
描述	
启用	

支持的遥测数据列

本节列出了以下监控项参数中支持的列：*Columns*、*Aggregated columns* 和 *Conditions*。这些列由 ClickHouse 用于存储遥测数据，其架构遵循 OpenTelemetry ClickHouse exporter。

选择类别：

- Traces
- Metrics: Sum
- Metrics: Gauge
- Metrics: Histogram
- Metrics: Exponential histogram
- Logs

追踪

列	列	聚合列	条件
持续时间	✓	✓	–
Events.Attributes	–	–	✓
Events.Name	–	–	✓
ParentSpanId	✓	–	✓
ResourceAttributes	✓	–	✓
ScopeName	✓	–	✓
ScopeVersion	✓	–	✓
ServiceName	✓	–	✓
SpanAttributes	✓	–	✓
SpanId	✓	–	✓
SpanKind	✓	–	✓
SpanName	✓	–	✓
StatusCode	✓	–	✓
StatusMessage	✓	–	✓
时间戳	✓	✓	–
TraceId	✓	–	✓
TraceState	✓	–	✓

指标：求和

列	列	聚合列	条件
AggregationTemporality	✓	–	–
Attributes	✓	–	✓
Exemplars.FilteredAttributes	–	–	✓
Flags	✓	–	–
IsMonotonic	✓	–	–
MetricDescription	✓	–	✓
MetricName	✓	–	✓
MetricUnit	✓	–	✓
ResourceAttributes	✓	–	✓
ResourceSchemaUrl	✓	–	✓
ScopeAttributes	✓	–	✓
ScopeDroppedAttrCount	✓	✓	–
ScopeName	✓	–	✓
ScopeSchemaUrl	✓	–	✓
ScopeVersion	✓	–	✓
ServiceName	✓	–	✓
StartTimeUnix	✓	✓	–
TimeUnix	✓	✓	–
Value	✓	✓	–

指标：Gauge

列	列	聚合列	条件
Attributes	✓	–	✓
Exemplars.FilteredAttributes	–	–	✓

列	列	聚合列	条件
Flags	✓	–	–
MetricDescription	✓	–	✓
MetricName	✓	–	✓
MetricUnit	✓	–	✓
ResourceAttributes	✓	–	✓
ResourceSchemaUrl	✓	–	✓
ScopeAttributes	✓	–	✓
ScopeDroppedAttrCount	✓	✓	–
ScopeName	✓	–	✓
ScopeSchemaUrl	✓	–	✓
ScopeVersion	✓	–	✓
ServiceName	✓	–	✓
StartTimeUnix	✓	✓	–
TimeUnix	✓	✓	–
Value	✓	✓	–

指标：直方图

列	列	聚合列	条件
AggregationTemporality	✓	–	–
Attributes	✓	–	✓
Count	✓	✓	–
Exemplars.FilteredAttributes	–	–	✓
Flags	✓	–	–
Max	✓	✓	–
MetricDescription	✓	–	✓
MetricName	✓	–	✓
MetricUnit	✓	–	✓
Min	✓	✓	–
ResourceAttributes	✓	–	✓
ResourceSchemaUrl	✓	–	✓
ScopeAttributes	✓	–	✓
ScopeDroppedAttrCount	✓	✓	–
ScopeName	✓	–	✓
ScopeSchemaUrl	✓	–	✓
ScopeVersion	✓	–	✓
ServiceName	✓	–	✓
StartTimeUnix	✓	✓	–
Sum	✓	✓	–
TimeUnix	✓	✓	–

指标：指数直方图

列	列	聚合列	条件
AggregationTemporality	✓	–	–
Attributes	✓	–	✓
Count	✓	✓	–
Exemplars.FilteredAttributes	–	–	✓
Flags	✓	–	–
Max	✓	✓	–
MetricDescription	✓	–	✓
MetricName	✓	–	✓
MetricUnit	✓	–	✓
Min	✓	✓	–
NegativeOffset	✓	✓	–
PositiveOffset	✓	✓	–
ResourceAttributes	✓	–	✓
ResourceSchemaUrl	✓	–	✓

列	列	聚合列	条件
Scale	✓	✓	–
ScopeAttributes	✓	–	✓
ScopeDroppedAttrCount	✓	✓	–
ScopeName	✓	–	✓
ScopeSchemaUrl	✓	–	✓
ScopeVersion	✓	–	✓
ServiceName	✓	–	✓
StartTimeUnix	✓	✓	–
Sum	✓	✓	–
TimeUnix	✓	✓	–
ZeroCount	✓	✓	–

日志

列	列	聚合列	条件
正文	✓	–	✓
EventName	✓	–	✓
LogAttributes	✓	–	✓
ResourceAttributes	✓	–	✓
ResourceSchemaUrl	✓	–	✓
ScopeAttributes	✓	–	✓
ScopeName	✓	–	✓
ScopeSchemaUrl	✓	–	✓
ScopeVersion	✓	–	✓
SpanId	✓	–	✓
ServiceName	✓	–	✓
SeverityNumber	✓	✓	–
SeverityText	✓	–	✓
Timestamp	✓	✓	–
TraceFlags	✓	–	–
TraceId	✓	–	✓

历史和趋势

概述

历史数据和趋势数据是在 Zabbix 中存储已采集数据的两种方式。

历史数据会保存每一个采集到的值，而趋势数据则按小时保存平均信息，因此占用的资源更少。

保留历史数据

您可以设置历史数据保留的天数：

- 在监控项属性表单中
- 在批量更新监控项时
- 在设置 housekeeper 任务时

任何更早的数据都将由 housekeeper 删除。

通常强烈建议将历史数据保留天数设置为尽可能少，以避免大量历史值给数据库带来过重负载。

与其保留较长时间的历史数据，不如将趋势数据保留更长时间。
例如，您可以将历史数据保留 14 天，而将趋势数据保留 5 年。

您可以参考数据库容量规划页面，从而大致了解历史数据与趋势数据分别需要多少存储空间。

即使历史数据保留时间较短，您仍然可以在图形中查看较早的数据，因为图形在显示较早数据时会使用趋势值。

Attention:
如果历史数据设置为“0”，监控项将只更新依赖监控项和清单。不会评估任何触发器函数，因为触发器评估仅基于历史数据。

Note:
作为保留历史数据的另一种方式，可以考虑使用可加载模块的历史数据导出功能。

保留趋势数据

趋势数据是一种内置的历史数据缩减机制，它会针对数值型数据，每小时存储最小值、最大值、平均值以及值的总数量。

您可以设置趋势数据保留多少天：

- 在监控项属性表单中
- 在批量更新监控项时
- 在设置 Housekeeper 任务时

趋势数据通常可以比历史数据保留更长时间。
任何更早的数据都会被 housekeeper 删除。

随着数据不断流入，Zabbix 服务器会在运行时于趋势缓存中累积趋势数据。
服务器会在以下情况下将每个监控项的前一小时趋势数据刷新到数据库中（前端可从中读取）：

- 服务器接收到该监控项当前小时的第一个值
- 当前小时剩余 5 分钟或更少，且该监控项仍然没有当前小时的值
- 服务器停止

要在图形中查看趋势数据，您至少需要等到下一个小时开始时（如果监控项更新频繁），最多需要等到下一个小时结束时（如果监控项更新较少），也就是说最长需要等待 2 小时。

当服务器刷新趋势缓存时，如果数据库中已经存在该小时的趋势数据（例如服务器在该小时中途重启过），服务器就需要使用 update 语句而不是简单的 insert。
因此，在较大的安装环境中，如果需要重启，建议在某个小时结束时停止服务器，并在下一个小时开始时启动，以避免趋势数据重叠。

历史表不会以任何方式参与趋势数据的生成。

Attention:
如果趋势数据设置为 “0”，Zabbix 服务器将完全不会计算或存储趋势数据。

Note:
趋势数据会使用与原始值相同的数据类型进行计算和存储。
因此，无符号数据类型值的平均值计算结果会被取整，而且值区间越小，结果就越不精确。
例如，如果监控项的值为 0 和 1，则平均值将为 0，而不是 0.5。
此外，重启服务器还可能导致当前小时内无符号数据类型平均值计算的精度损失。

用户参数

概述

有时你可能希望运行一个 Zabbix 中未预定义的 agent 检查。这时，用户参数就派上用场了。

你可以编写一个用于获取所需数据的命令，并将其包含在agent 配置文件 中的用户参数里（`UserParameter` 配置参数）。

用户参数具有以下语法：

```
UserParameter=<key>,<command>
```

如你所见，用户参数也包含一个 key。在配置监控项时，这个 key 是必需的。请输入一个你选择的、便于引用的 key（它在同一主机内必须唯一）。

重启 agent，或使用 agent 的运行时控制 选项来加载新参数，例如：

```
zabbix_agentd -R userparameter_reload
```

然后，在配置监控项时，输入该 key，以引用你希望执行的用户参数中的命令。

用户参数是由 Zabbix agent 执行的命令。请注意，在监控项值预处理 步骤之前，最多可以返回 16MB 的数据。

在 UNIX 操作系统下，**/bin/sh** 用作命令行解释器。用户参数遵循 agent 检查超时；如果达到超时时间，已 fork 的用户参数进程将被终止。

另请参见：

- 关于如何使用用户参数的分步教程
- 命令执行

用户自定义参数用例

一个简单的命令：

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

agent 将始终为使用 “ping” 为 key 的监控项返回 “1”。

一个复杂的例子：

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

如果 Mysql 服务器是活动状态，agent 将返回 “1”，否则会返回 “0”。

灵活用户参数

灵活用户参数接受带有键的参数。这样，灵活用户参数就可以作为创建多个监控项的基础。

灵活用户参数具有以下语法：

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

Parameter	Description
Key	唯一的监控项键。\[*\] 定义了此键接受方括号内的参数。 在配置监控项时提供这些参数。
Command	用于计算该键值的要执行命令。 仅适用于灵活用户参数： 您可以在命令中使用位置引用 $1…$9 来引用监控项键中的相应参数。 Zabbix 会解析监控项键中 [ ] 包含的参数，并相应地在命令中替换 $1,...,$9。 $0 将被替换为要运行的原始命令（在展开 $0,...,$9 之前）。 无论位置引用是否用双引号 (") 或单引号 (') 括起来，都会被解释。 若要原样使用位置引用，请指定双美元符号，例如 awk '{print $$2}'。在这种情况下，$$2 在执行命令时实际上会变成 $2。

Attention:
带有 $ 符号的位置引用仅由 Zabbix agent 在灵活用户参数中进行搜索和替换。对于简单用户参数，会跳过此类引用处理，因此不需要对任何 $ 符号进行转义。

Attention:
默认情况下，用户参数中不允许某些字符。完整字符列表请参见UnsafeUserParameters。

示例 1

先来一个简单的：

```
UserParameter=ping[*],echo $1
```

我们可以定义无数个监控项来监控所有形如 ping[something] 格式的东西。

- ping[0] - 将总是返回 ‘ 0 ’
- ping[aaa] - 将总是返回 ‘aaa’

示例 2

让我们更进一步！

```
UserParameter=mysql.ping[*],mysqladmin -u$1 -p$2 ping | grep -c alive
```

这个用户自定义参数可以用来监控 MySQL 数据库的可用性状态。我们可以传入用户名和密码：

```
mysql.ping[zabbix,our_password]
```

示例 3

一个文件中有多少行匹配正则表达式？

```
UserParameter=wc[*],grep -c "$2" $1
```

这个用户自定义参数能用来计算一个文件中有多少行匹配相应的表达式。就像下面一样：

```
wc[/etc/passwd,root]
wc[/etc/services,zabbix]
```

命令结果

命令的返回值为命令产生的标准输出以及标准错误。

> **Attention:**
> 返回文本（字符、日志或文本类型信息）的监控项在出现标准错误输出时，不会变为不支持。

返回值限制为 16MB（包括会被截断的尾随空白字符）；数据库限制 同样适用。

返回文本（字符、日志或文本类型信息）的用户参数也可以返回空白字符。如果结果无效，监控项将变为不支持。

扩展 **Zabbix agent**

本教程提供了逐步说明，介绍如何使用user parameter 来扩展 Zabbix agent 的功能。

第 1 步

编写脚本或命令行以检索所需参数。

例如，我们可以编写以下命令，以获取 MySQL 服务器执行的查询总数：

```
mysqladmin -uroot status | cut -f4 -d":" | cut -f1 -d"S"
```

执行后，该命令会返回 SQL 查询总数。

步骤 2

将该命令添加到 Zabbix agent 配置文件（zabbix_agentd.conf）中：

```
UserParameter=mysql.questions,mysqladmin -uroot status | cut -f4 -d":" | cut -f1 -d"S"
```

mysql.questions 是唯一标识符。它可以是任何有效的键标识符，例如 *queries*。

使用带有 -t 标志的 Zabbix agent 测试此参数（但是，如果在 root 用户下运行，请注意 agent 作为守护进程启动时可能具有不同的权限）：

```
zabbix_agentd -t mysql.questions
```

第 3 步

通过运行以下命令从配置文件重新加载用户参数：

```
zabbix_agentd -R userparameter_reload
```

你也可以重启 agent，而不是使用运行时控制命令。

使用zabbix_get 工具测试该参数。

第 4 步

向被监控的主机添加新的监控项，Key=mysql.questions。该监控项的类型必须是 *Zabbix Agent* 或 *Zabbix Agent (active)*。

请注意，返回值的类型必须在 Zabbix 服务器上正确设置。否则 Zabbix 将不会接受它们。

Windows 性能计数器

概述

您可以使用 perf_counter[] 键有效地监控 Windows 性能计数器。

例如：

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

或

```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

有关如何使用此键或其仅支持英文的等效键 perf_counter_en 的更多信息，请参阅Windows 专用监控项键。

要获取可用于监控的性能计数器完整列表，可以运行：

```
typeperf -qx
```

您还可以使用低级别发现来发现 Windows 性能计数器的多个对象实例，并自动为多个实例对象创建 perf_counter 监控项。

数字表示

Windows 会为对象名称和性能计数器名称维护数字表示（索引）。

Zabbix 支持将这些数字表示作为 perf_counter、perf_counter_en 监控项键的参数，以及 PerfCounter、PerfCounterEn 配置参数的参数。

但是，除非您能够确保数字索引在特定主机上对应正确的字符串，否则不建议使用这些数字表示。如果您需要创建可移植的监控项，使其能够在不同主机以及各种本地化 Windows 版本上正常工作，可以使用 perf_counter_en 键或 PerfCounterEn 配置参数。它们允许您无论系统区域设置如何，始终使用英文名称。

要查找数字对应值，请运行 regedit，然后在 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\ 中找到 *Counter*。

注册表项包含如下信息：

```
1
1847
2
System
4
Memory
6
% Processor Time
10
File Read Operations/sec
12
File Write Operations/sec
14
File Control Operations/sec
16
File Read Bytes/sec
18
File Write Bytes/sec
....
```

在这里，您可以找到性能计数器每个字符串部分对应的数字，例如 \System\% Processor Time：

```
System → 2
% Processor Time → 6
```

然后，您可以使用这些数字以数字形式表示路径：

```
\2\6
```

性能计数器参数

您可以部署一些 PerfCounter 参数，用于监控 Windows 性能计数器。

例如，您可以将以下参数添加到 Zabbix agent 配置文件中：

```
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30
或
PerfCounter=UserPerfCounter2,"\4\24",30
```

配置这些参数后，您只需使用 UserPerfCounter1 或 UserPerfCounter2 作为键，即可创建相应的监控项。

请注意，修改配置文件后，需要重启 Zabbix agent。

批量更新

概览

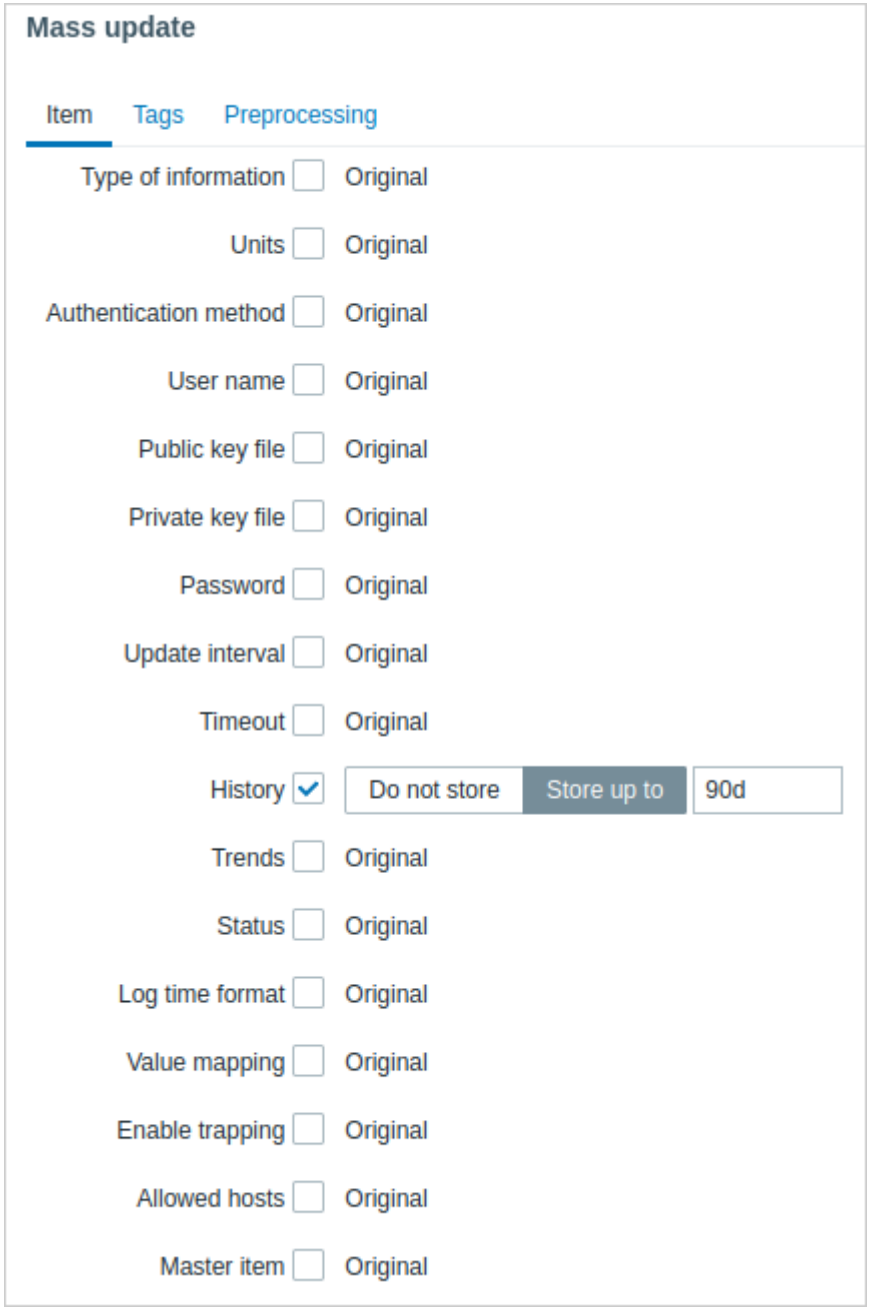
有时你可能想要一次更改多个监控项的某些属性。你可以使用批量更新功能，而不是打开每个独立的监控项进行编辑。

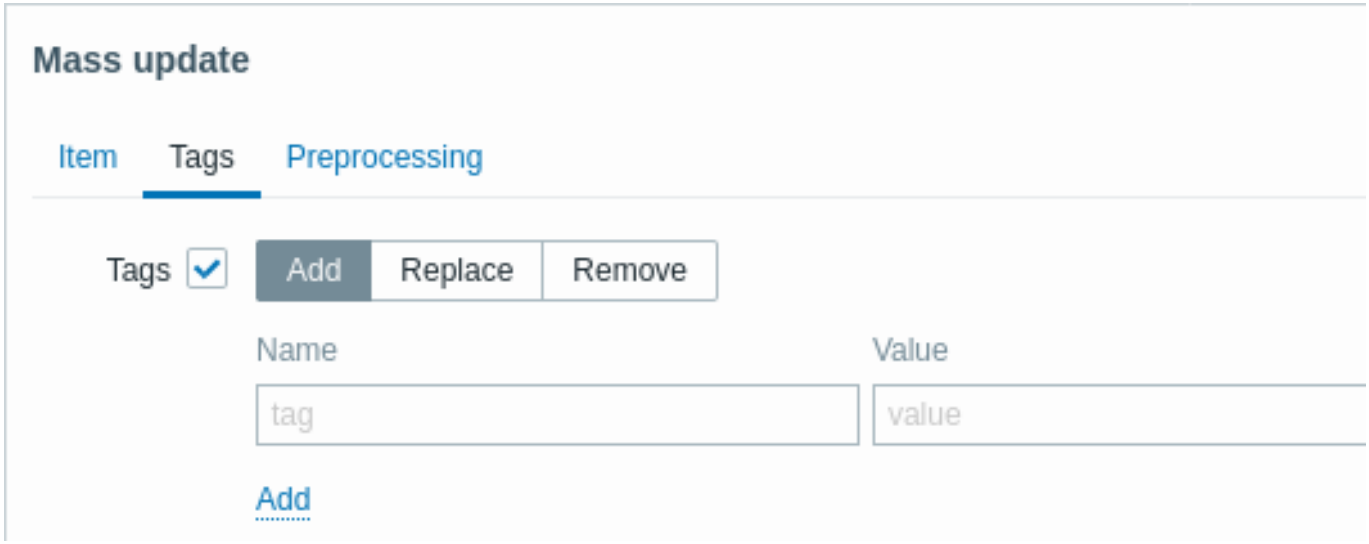
使用批量更新

要批量更新某些监控项，请执行以下操作：

- 在列表中勾选要更新的监控项
- 点击列表下方的批量更新

- 导航到包含所需属性的选项卡（监控项、标签或 预处理）
- 勾选要更新的属性
- 输入属性的新值

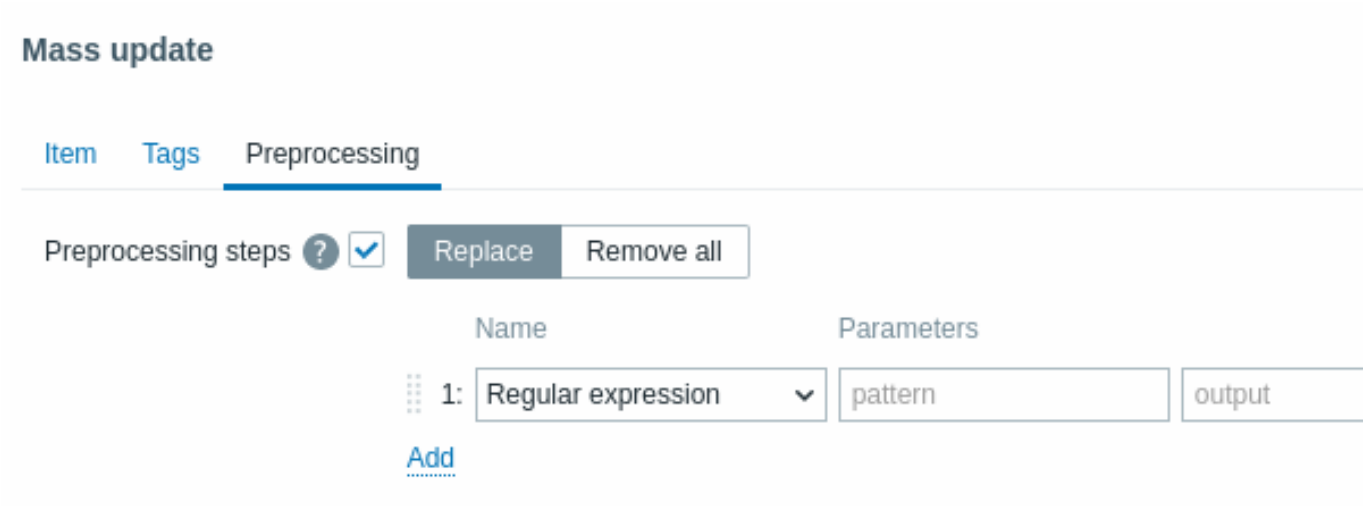




标签选项允许您：

- 添加 - 向监控项添加指定的标签（名称相同但值不同的标签不视为“重复”标签，可以添加到同一监控项）。
- 替换 - 删除指定的标签，并添加具有新值的标签
- 移除 - 从监控项中删除指定的标签

标签支持用户宏、{INVENTORY.*} 宏、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 宏。



勾选预处理步骤旁边的复选框，以启用对预处理步骤的批量更新。

预处理批量更新允许您：

- 替换 - 使用下方指定的预处理步骤替换监控项上现有的预处理步骤
- 全部移除 - 从监控项中移除所有预处理步骤

完成后，点击更新。

值映射

概览

为了接收到的值能更人性化的表示，你可以使用包含表示数值/字符值和字符串之间映射的值映射。

例如，当监控项的值为”0” 或”1” 时，可以使用值映射以更用户友好的方式表示这些值：

- 0 → 不可用
- 1 → 可用

数据备份类型的值映射可配置如下：

- F → 完全备份
- D → 差异备份
- I → 增量备份

电压的值范围可以配置如下：

- <=209 → 低
- 210-230 → 正常
- >=231 → 高

值映射在 Zabbix 前端界面和通过媒体类型发送的通知中使用。

Attention:
在 Zabbix 前端和 server 中，都执行接收值与配置表示的替换；然而，服务器仅在以下情况下处理替换：
- 在填入主机清单;
- 在使用拓展支持的宏 {ITEM.VALUE}、{ITEM.LASTVALUE}、{EVENT.OPDATA} 和 {EVENT.CAUSE.OPDATA}。

值映射在模板或主机上设置。一旦配置，它们将可用于相应模板或主机内的所有监控项。在配置监控项时，在值映射参数中指定之前配置好的值映射的名称。

Note:
没有值映射继承功能 - 主机和模板不会从链接的模板继承值映射。主机上的模板监控项将继续使用在模板上配置的值映射。

Note:
值映射可用于具有无符号数字（*Numeric (unsigned)*）、浮点数字（*Numeric (float)* 和字符（*Character*）类型的信息的监控项。

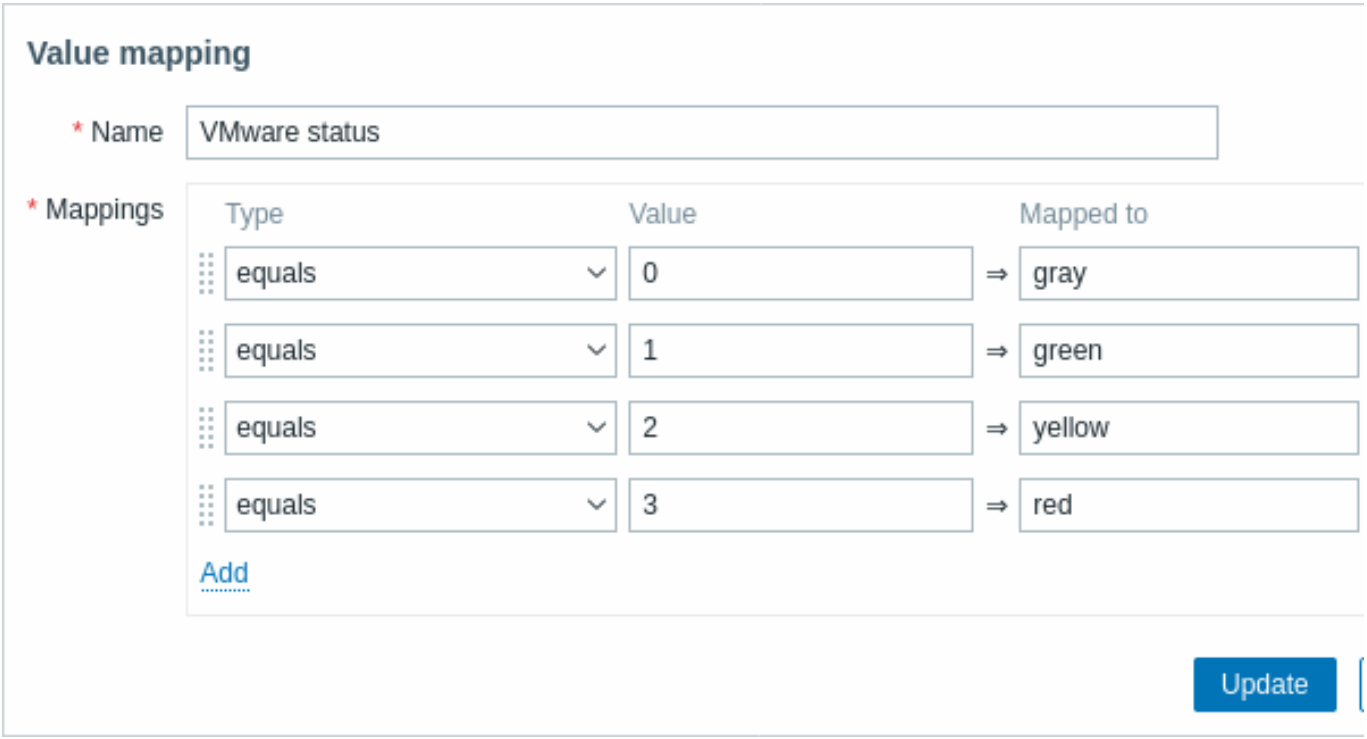
值映射会随着相应的模板或主机一起被导出/导入。也可使用主机 和模板的批量更新表单进行批量更新。

配置

要配置值映射，请执行以下步骤：

1. 打开主机或模板配置表单。

2. 在值映射选项卡中，单击添加以添加新的值映射，或单击现有映射的名称进行编辑。



值映射的参数：

参数	描述
名称	值映射集合的唯一名称。
映射	将数值/字符串值映射为字符串表示形式的单独规则。 映射按照规则的顺序应用；可以通过拖动来调整规则顺序。
类型	映射类型： 等于 - 将相等的值进行映射； 大于或等于 - 将相等或更大的值进行映射； 小于或等于 - 将相等或更小的值进行映射； 范围内 - 将范围内的值进行映射；范围表示为 <number1>-<number2> 或 <number>；支持多个范围（例如：1-10,101-110,201）； 正则表达式 - 将符合正则表达式的值进行映射（不支持全局正则表达式）； 默认 - 将所有未匹配的值进行映射，但具有特定映射的值除外。 对于范围映射，仅支持数值类型（大于或等于、小于或等于、范围内）。
值	传入的值（根据映射类型，可能包含范围或正则表达式）。
映射为	传入值的字符串表示形式（最多 64 个字符）。

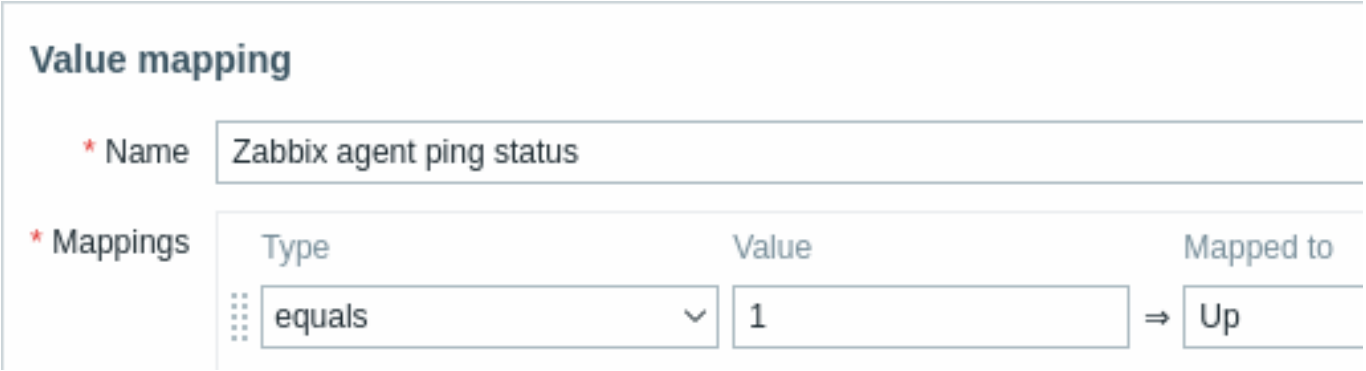
所有必填输入字段均以红色星号标记。

在列表中查看值映射时，仅显示前 3 个映射，三个点表示存在更多映射。



值映射示例

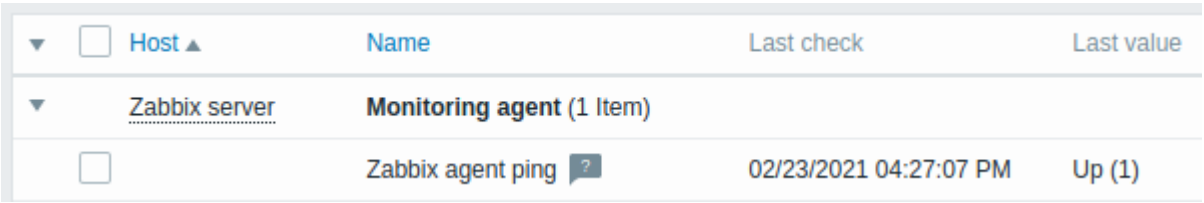
预定义的 agent 监控项 *Zabbix agent ping* 之一使用模板级别的值映射 “Zabbix agent ping status” 来显示其值。



在监控项配置表单中，您可以在值映射字段中找到对此值映射的引用：

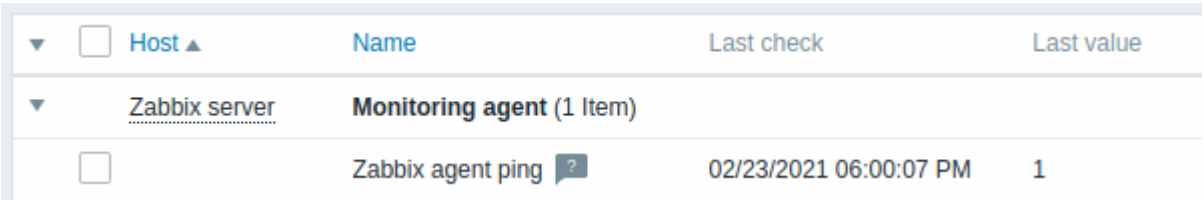


此映射用于监测 > 最新数据部分，以显示 “Up”（原始值显示在括号中）。



Note:
在最新数据部分中，显示的值会缩短为 20 个字符。如果使用了值映射，则不会缩短映射后的值，而只会缩短原始值（显示在括号中）。

如果没有预定义的值映射，您只能看到 “1”，这可能难以理解。



队列

概述

队列显示的是正在等待刷新的监控项。该队列只是数据的一种逻辑表示。在 Zabbix 中，并不存在 IPC 队列或任何其他队列机制。

由 proxy 监控的监控项也包含在队列中——在 proxy 历史数据更新周期内，它们会被计为排队状态。

队列中仅显示具有计划刷新时间的监控项。这意味着以下监控项类型不包含在队列中：

- log、logrt 和事件日志主动式 Zabbix agent 监控项

- SNMP trap 监控项
- trapper 监控项
- Web 监控项
- 依赖监控项

队列显示的统计信息是衡量 Zabbix 服务器性能的一个良好指标。

队列通过 JSON 协议直接从 Zabbix 服务器获取。只有在 Zabbix 服务器运行时，此信息才可用。

> **Attention:**
> 如果监控项接口因连接问题或 agent 工作异常而变得不可用，则该监控项不会被计入队列。

读取队列

要读取队列，请转到 管理 > 队列 > 队列概览。

Queue overview

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

此处的状态通常为“正常”，因此可以认为服务器运行正常。

队列显示有一些监控项等待时间最长达 30 秒。了解这些监控项具体是什么会很有帮助。

要查看这些监控项，请在标题下拉菜单中选择 队列详细信息。现在，您可以看到这些延迟监控项的列表。

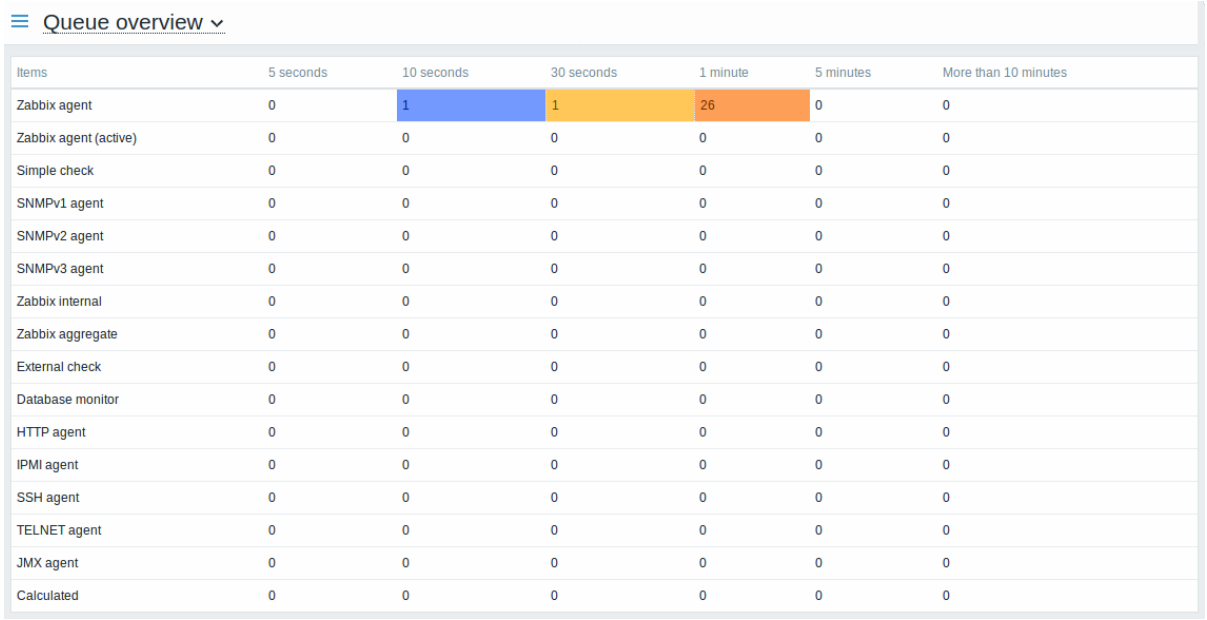
Queue details

Scheduled check	Delayed by	Host	Name	Proxy
2019-09-02 11:46:40	58s	My host	CPU idle time	Remote proxy
2019-09-02 11:46:41	57s	My host	CPU interrupt time	Remote proxy
2019-09-02 11:46:42	56s	My host	CPU iowait time	Remote proxy
2019-09-02 11:46:43	55s	My host	CPU nice time	Remote proxy
2019-09-02 11:46:44	54s	My host	CPU softirq time	Remote proxy
2019-09-02 11:46:45	53s	My host	CPU steal time	Remote proxy
2019-09-02 11:46:46	52s	My host	CPU system time	Remote proxy

有了这些详细信息，您可能就能找出这些监控项延迟的原因。

如果只有一两个监控项出现延迟，可能无需担心。它们可能会在一秒钟后更新。但是，如果您看到大量监控项长时间延迟，则可能存在更严重的问题。

另请参阅：使用调度间隔时对齐时区。



Queue overview

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	1	1	26	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0
IPMI agent	0	0	0	0	0	0
SSH agent	0	0	0	0	0	0
TELNET agent	0	0	0	0	0	0
JMX agent	0	0	0	0	0	0
Calculated	0	0	0	0	0	0

队列监控项

可以使用特殊的内部监控项 `zabbix[queue,<from>,<to>]` 来监控 Zabbix 中队列的运行状况。它将返回延迟时间达到设定时长的监控项数量。有关更多信息，请参阅内部监控项。

值缓存

概览

为了更快地计算触发器表达式、计算或聚合类型监控项和一些宏。Zabbix Server 支持值缓存选项。

这个内存缓存，可以用于访问历史数据，而不需要对数据库直接执行 SQL 调用。如果请求的历史值不在缓存中，则会从数据库请求缺失的数据，并相应地更新缓存。

监控项的值会保留在值缓存中，直到以下情况发生：

- 监控项被删除（缓存值在下次配置同步后被删除）；
- 监控项的值超出了触发器/计算项表达式指定的时间或计数范围（当接收到新值时，缓存值被移除）；
- 触发器/计算项表达式指定的时间或计数范围被更改，以减少计算所需的数据量（不必要的缓存值在 24 小时后被移除）。

Note:
可以使用服务器运行时控制 选项 `diaginfo` (or `diaginfo=valuecache`) 来观察值缓存的状态，并检查值缓存诊断信息的部分。这可以帮助确定配置不当的触发器或计算项。

要启用值缓存功能，可以在 Zabbix 服务器配置 中指定可选的 `ValueCacheSize` 参数。

支持使用两个内部监控项监控值缓存：`zabbix[vcache,buffer,<mode>]` 和 `zabbix[vcache,cache,<parameter>]`。有关详细信息，请参阅内部监控项。

立即执行

概述

立即执行选项允许立即执行被动检查。

Zabbix 中的监控项值采集遵循已配置的更新间隔。某些监控项（例如低级别发现规则）具有较长的间隔，您可能需要立即获取新值——例如，为了检测可发现资源中的变化。

立即执行选项支持以下监控项类型：

- Zabbix agent（被动）
- 简单检查
- SNMP agent（v1/v2/v3）
- Zabbix 内部
- 外部检查
- 数据库监控

- HTTP agent
- IPMI agent
- SSH agent
- TELNET agent
- JMX agent
- 计算型
- 依赖监控项
- 脚本
- 浏览器

如果监控项的类型为“依赖监控项”，则其主监控项也必须是上述类型之一。

配置

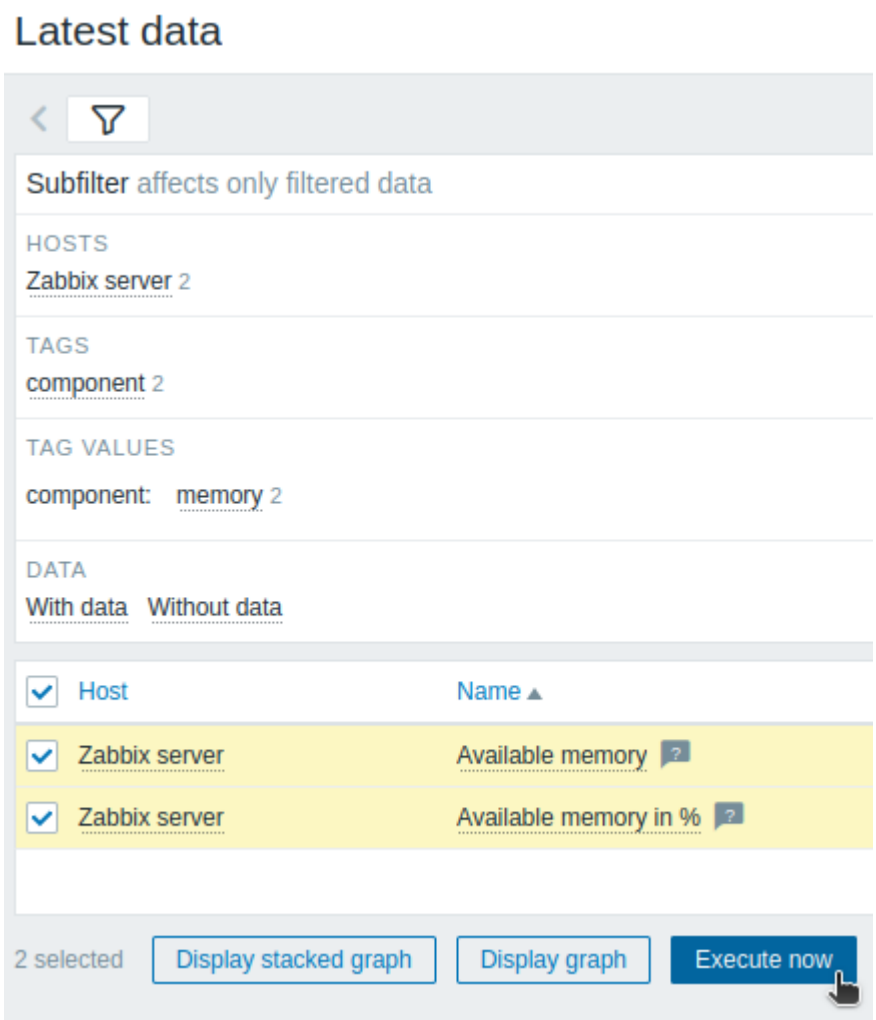
你可以从以下位置执行被动检查：

- 监控 > 最新数据
- 数据采集 > 主机 > 监控项 或发现
- 监控项菜单
- 在编辑监控项或低级别发现规则时（在其配置表单中）

Attention:
要执行检查，必须先将其存在于配置缓存中；另请参见CacheUpdateFrequency。在检查执行之前，不会更新缓存，因此对监控项或低级别发现规则配置所做的最近更改不会被采纳。要测试正在创建或新创建的监控项或 LLD 规则，可以改用测试 选项。

例如，要在 监控 > 最新数据 中执行被动检查：

1. 选择监控项；你可以一次对多个监控项执行检查。
2. 单击 立即执行。



Note:
在 最新数据中，用户可以对其具有 读写权限 的主机上的监控项执行 立即执行，或者在其角色中启用了 读权限以及 在只读主机上调用 “立即执行” 操作。

限制 **agent** 检查

概述

您可以控制 Zabbix agent 或 agent 2 在执行监控项检查、远程命令或脚本时允许或拒绝使用哪些监控项键值。

为此，请使用这些agent 配置 参数来定义允许/拒绝规则：

- AllowKey=<pattern>
- DenyKey=<pattern>
- AllowKeyRegexp=<pattern>
- DenyKeyRegexp=<pattern>

<pattern> 必须包含单个监控项键值。
 对于 AllowKey 和 DenyKey，<pattern> 支持通配符（*）。通配符会匹配其所在位置上的任意数量任意字符，可用于匹配监控项键值或参数（例如，vfs.file.*[*]）。
 对于 AllowKeyRegexp 和 DenyKeyRegexp，<pattern> 支持正则表达式。正则表达式模式可用于匹配监控项键值及其参数字符串。另请参见正确转义示例。
 请注意，Zabbix agent 和 Zabbix agent 2 可能支持不同的正则表达式语法。

Attention:
为提高安全性，建议对 AllowKey 和 DenyKey 使用精确的监控项键值，而不是通配符。同时也建议对 AllowKeyRegexp 和 DenyKeyRegexp 使用尽可能具体的正则表达式模式。有关详细信息，请参见保护允许/拒绝规则。

与其他 agent 配置参数不同，您可以指定无限数量的 AllowKey、DenyKey、AllowKeyRegexp 或 DenyKeyRegexp 参数。

重要说明

- 默认情况下，所有system.run 监控项都会被禁用(即使 DenyKey 和 DenyKeyRegexp 为空)，就像设置了 DenyKey=system.run[*] 或 DenyKeyRegexp=^system\.run\[.*\]$ 作为最后一条规则 一样。因此，您可以允许特定的 system.run 监控项，而无需显式拒绝其他 system.run 监控项。
- 如果可能，请使用 AllowKey/AllowKeyRegexp 仅允许所需的监控项，并拒绝其他所有内容。某些键可能会被滥用，通过路径遍历读取非预期文件（例如，vfs.file.contents["../../../../etc/passwd"]），而新的 Zabbix agent 版本可能会引入您的 DenyKey/DenyKeyRegexp 规则未覆盖的键。
- AllowKey、DenyKey、AllowKeyRegexp 和 DenyKeyRegexp 配置不会影响 HostnameItem、HostMetadataItem 或 HostInterfaceItem agent 参数。
- 被拒绝的监控项会变为不受支持状态，且不会提供任何提示或错误消息；例如：
  - Zabbix agent --print （-p）命令行参数不会显示被拒绝的监控项键。
  - Zabbix agent --test （-t）命令行参数会对被拒绝的监控项键返回 “Unsupported item key.”。
  - 当启用日志记录（LogRemoteCommands=1）时，Zabbix agent 日志文件不会记录被拒绝的远程命令。

Allow/deny 规则顺序

您可以指定无限数量的 AllowKey、DenyKey、AllowKeyRegexp 或 DenyKeyRegexp 规则，但它们的顺序很重要。

- 规则会从上到下逐一评估。
- 当某个监控项键匹配某条规则时，该监控项要么被允许，要么被拒绝，并且规则评估会停止。

例如，在评估 vfs.file.contents[/etc/passwd] 时，规则处理如下：

```
AllowKey=vfs.file.contents[/tmp/app.log]    # 监控项键模式不匹配，agent 继续处理下一条规则。
AllowKey=vfs.file.contents[/etc/passwd]     # 监控项键模式匹配；agent 允许该监控项检查并停止规则评估。
DenyKey=vfs.file.*[*]                       # agent 忽略该规则，因为评估已停止。
```

以下规则顺序将拒绝该监控项检查：

```
DenyKey=vfs.file.*[*]                       # 监控项键模式匹配；agent 拒绝该监控项检查并停止规则评估。
AllowKey=vfs.file.contents[/etc/passwd]     # agent 忽略该规则，因为评估已停止。
AllowKey=vfs.file.contents[/tmp/app.log]    # agent 忽略该规则，因为评估已停止。
```

示例

以下示例展示了 AllowKey、DenyKey、AllowKeyRegexp 和 DenyKeyRegexp 的常见配置模式。

允许特定检查和命令

仅允许两个vfs.file 监控项检查和两个system.run 命令：

使用 AllowKey 和 DenyKey：

```
AllowKey=vfs.file.contents[/tmp/app.log]
AllowKey=vfs.file.size[/tmp/app.log]
AllowKey=system.run[/usr/bin/uptime]
AllowKey=system.run[/usr/bin/df -h /]
DenyKey=vfs.file.*[*]
```

使用 AllowKeyRegexp 和 DenyKeyRegexp：

```
AllowKeyRegexp=^vfs\.file\.(contents|size)\[/tmp/app\.log\]$
AllowKeyRegexp=^system\.run\[/usr/bin/(uptime|df -h /)\]$
DenyKeyRegexp=^vfs\.file\..*\[.*\]$
```

Note:
无需设置 DenyKey=system.run[*] 或 DenyKeyRegexp=^system\.run\[.*\]$，因为默认情况下所有其他 system.run 命令都会被拒绝。

允许脚本

允许 Zabbix agent 通过所有可用方法在主机上执行脚本：

- 全局脚本，可在前端或通过 API 执行（此方法始终使用 system.run[myscript.sh] 键）
- 来自action 操作 的远程命令（此方法始终使用 system.run[myscript.sh,nowait] 键）
- 带有脚本的system.run Zabbix agent 监控项，例如：
  - system.run[myscript.sh]
  - system.run[myscript.sh,wait]
  - system.run[myscript.sh,nowait]

```
AllowKey=system.run[myscript.sh,*]
```

或者

```
AllowKeyRegexp=^system\.run\[myscript\.sh,.*\]$
```

要控制 wait/nowait 参数，必须设置不同的规则。例如，您可以只允许 system.run[myscript.sh,wait] 监控项，从而排除其他方法：

```
AllowKey=system.run[myscript.sh,wait]
```

或者

```
AllowKeyRegexp=^system\.run\[myscript\.sh,wait\]$
```

保护 allow/deny 规则

这些示例展示了如何保护过于宽松的 AllowKey/AllowKeyRegexp 或 DenyKey/DenyKeyRegexp 规则。

请考虑以下规则：

使用 AllowKey 和 DenyKey：

```
AllowKey=system.run["C:\Program^ Files\Zabbix^ Agent^ 2\scripts\test.bat*"]
DenyKey=vfs.file.*
DenyKey=system.cpu.load[*]
```

使用 AllowKeyRegexp 和 DenyKeyRegexp：

```
AllowKeyRegexp=^system\.run\["C:\\Program\sFiles\\Zabbix\sAgent\s2\\scripts\\test\.bat.*"\]$
DenyKeyRegexp=^vfs\.file\..*$
DenyKeyRegexp=^system\.cpu\.load\[.*\]$
```

Note:
在 Windows 上，必须在路径中的空格前使用脱字符 (^) 为 AllowKey 或 DenyKey 转义，或者使用 \s 为 AllowKeyRegexp 或 DenyKeyRegexp 转义。

这些规则包含通配符 (*)（或基于正则表达式的模式中的 .*），可能会被滥用：

- test.bat 脚本可以使用任意参数执行，包括非预期参数。
- vfs.file.* (^vfs\.file\..*$) 模式会匹配带参数和不带参数的监控项键；但是，所有vfs.file 监控项都需要参数。
- system.cpu.load[*] (^system\.cpu\.load\[.*\]$) 模式只会匹配带参数的监控项键；但是system.cpu.load 监控项不需要参数。

要保护这些规则，应明确只允许使用特定参数执行 test.bat，并拒绝正确的监控项键模式；例如：

使用 AllowKey 和 DenyKey：

```
AllowKey=system.run["C:\Program^ Files\Zabbix^ Agent^ 2\scripts\test.bat status"]
AllowKey=system.run["C:\Program^ Files\Zabbix^ Agent^ 2\scripts\test.bat version"]
DenyKey=vfs.file.*[*]
DenyKey=system.cpu.load
DenyKey=system.cpu.load[*]
```

使用 AllowKeyRegexp 和 DenyKeyRegexp：

```
AllowKeyRegexp=^system\.run\["C:\\Program\sFiles\\Zabbix\sAgent\s2\\scripts\\test\.bat (status|version)"\]
DenyKeyRegexp=^vfs\.file\..+\[.*\]$
DenyKeyRegexp=^system\.cpu\.load(\[.*\])?$
```

您可以通过运行以下命令来测试这些规则，这些命令将返回 ZBX_NOTSUPPORTED。

```
cd "C:\Program Files\Zabbix Agent 2"
zabbix_agent2.exe -t system.run["C:\Program^ Files\Zabbix^ Agent^ 2\scripts\test.bat debug"]
zabbix_agent2.exe -t vfs.file.size["C:\ProgramData\MyApp\config.ini"]
zabbix_agent2.exe -t vfs.file.contents["C:\Windows\System32\drivers\etc\hosts"]
zabbix_agent2.exe -t system.cpu.load
zabbix_agent2.exe -t system.cpu.load[all,avg1]
```

模式示例

下表展示了监控项键模式的匹配方式：

- 只有当键满足 *Matches* 列中的所有条件时，才视为匹配该模式。
- 参数必须完整地包含在方括号中（例如，vfs.file.contents[* 和 vfs.file.contents*utf8] 都是无效模式）。
- 这些示例仅用于说明模式匹配；实际监控项需要参数。

对于 AllowKey 和 DenyKey：

Pattern	Matches	Examples
*	任何带参数或不带参数的键	
vfs.file.*	键以 vfs.file. 开头 带参数或不带参数	Matches: vfs.file.size vfs.file.contents vfs.file.contents[] vfs.file.size[/var/log/app.log]
vfs.*.contents	键以 vfs. 开头 键以 .contents 结尾 无参数	Matches: vfs..contents vfs.mount.point.file.contents Does not match: vfs.contents vfs.file.contents[]
vfs.file.*[*]	键以 vfs.file. 开头 任意参数或空参数	Matches vfs.file.get.custom[] vfs.file.size[/var/log/app.log, utf8] Does not match: vfs.file.get.custom
vfs.file.contents	键为 vfs.file.contents 无参数	Matches: vfs.file.contents Does not match: vfs.file.contents[/etc/passwd]

Pattern	Matches	Examples
vfs.file.contents[]	键为 vfs.file.contents[] 空参数	Matches: vfs.file.contents[] Does not match: vfs.file.contents
vfs.file.contents[*]	键为 vfs.file.contents 任意参数或空参数	Matches: vfs.file.contents[/path/to/file] Does not match: vfs.file.contents
vfs.file.contents[/etc/passwd,*]	键为 vfs.file.contents 第一个参数为 /etc/passwd 第二个参数任意或为空	Matches: vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf8] Does not match: vfs.file.contents[] vfs.file.contents[/etc/passwd]
vfs.file.contents[*passwd*]	键为 vfs.file.contents 任意参数，且至少有一个包含 passwd	Matches: vfs.file.contents[/etc/passwd] vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf8]
vfs.file.contents[*passwd*,*]	键为 vfs.file.contents 第一个参数包含 passwd 第二个参数任意或为空	Matches: vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf8] Does not match: vfs.file.contents[/etc/passwd] vfs.file.contents[/tmp/test]
vfs.file.contents[/etc/passwd,utf8]	键为 vfs.file.contents 第一个参数为 /etc/passwd 第二个参数为 utf8	Matches: vfs.file.contents[/etc/passwd,utf8] Does not match: vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf16]

对于 AllowKeyRegexp 和 DenyKeyRegexp：

Pattern	Matches	Examples
^.*$	任何带参数或不带参数的键	
^vfs\.file\..*$	键以 vfs.file. 开头 带参数或不带参数	Matches: vfs.file.size vfs.file.contents vfs.file.contents[] vfs.file.size[/var/log/app.log]
^vfs\..*\.contents$	键以 vfs. 开头 键以 .contents 结尾 无参数	Matches: vfs..contents vfs.mount.point.file.contents Does not match: vfs.contents vfs.file.contents[]
^vfs\.file\..*\[.*\]$	键以 vfs.file. 开头 任意参数或空参数	Matches vfs.file.get.custom[] vfs.file.size[/var/log/app.log, utf8] Does not match: vfs.file.get.custom

Pattern	Matches	Examples

`^vfs\.file\.contents$`	键为 `vfs.file.contents`
	无参数
		Matches:
		`vfs.file.contents`
		
		Does not match:
		`vfs.file.contents[/etc/passwd]`

`^vfs\.file\.contents\[\]$`	键为 `vfs.file.contents[]`
	空参数
		Matches:
		`vfs.file.contents[]`
		
		Does not match:
		`vfs.file.contents`

`^vfs\.file\.contents\[.*\]$`	键为 `vfs.file.contents`
	任意参数或空参数
		Matches:
		`vfs.file.contents[/path/to/file]`
		
		Does not match:
		`vfs.file.contents`

`^vfs\.file\.contents\[/etc/passwd,.*\]$`	键为 `vfs.file.contents`
	第一个参数为 `/etc/passwd`
	第二个参数任意或为空
		Matches:
		`vfs.file.contents[/etc/passwd,]`
		`vfs.file.contents[/etc/passwd,utf8]`
		
		Does not match:
		`vfs.file.contents[]`
		`vfs.file.contents[/etc/passwd]`

`^vfs\.file\.contents\[.*passwd.*,.*\]$`	键为 `vfs.file.contents`
	任意参数，且至少有一个包含 `passwd`
		Matches:
		`vfs.file.contents[/etc/passwd]`
		`vfs.file.contents[/etc/passwd,]`
		`vfs.file.contents[/etc/passwd,utf8]`

`^vfs\.file\.contents\[.*passwd.*,.*\]$`	键为 `vfs.file.contents`
	第一个参数包含 `passwd`
	第二个参数任意或为空
		Matches:
		`vfs.file.contents[/etc/passwd,]`
		`vfs.file.contents[/etc/passwd,utf8]`
		
		Does not match:
		`vfs.file.contents[/etc/passwd]`
		`vfs.file.contents[/tmp/test]`

`^vfs\.file\.contents\[/etc/passwd,(utf8|windows-1252)\]$`	键为 `vfs.file.contents`
	第一个参数为 `/etc/passwd`
	第二个参数为 `utf8` 或 `windows-1252`
		Matches:
		`vfs.file.contents[/etc/passwd,utf8]`
		`vfs.file.contents[/etc/passwd,windows-1252]`
		
		Does not match:
		`vfs.file.contents[/etc/passwd,]`
		`vfs.file.contents[/etc/passwd,utf16]`

使用触发器进行问题检测

Zabbix 中的问题检测基于 触发器。

触发器描述问题条件，使 Zabbix 能够在问题发生时做出响应。

问题检测的完整流程如下：

1. 确保你有一个用于收集数据的监控项。
2. 为该监控项定义触发器。没有触发器时，Zabbix 只会收集数据，但不会对其作出响应。
3. 查看已检测到的问题：
   - 在 *Monitoring* > *Problems*
   - 在 *Problems* widget
4. 接收通知，执行远程命令（如果已定义）。

请注意，官方 Zabbix 模板包含预定义的触发器。
有关将模板应用于monitor Linux 的指南，请参见。

什么是“触发器”？

触发器通过trigger expression 来描述问题条件。

当问题条件在现实中发生时，触发器就会“触发”（即在 Zabbix 中创建一个问题）。

问题条件可以基于最新值、平均值、发现的字符串以及许多其他functions。对于返回值的数据类型为二进制或 JSON 的监控项，无法创建触发器。

一个简单的触发器表达式会对括号中指定的数据（通常是数据监控项和时间窗口）计算某个函数（例如表示最大值的 **max()**），然后将计算结果与阈值进行比较。

例如：

```
max(/host/vfs.fs.size[/,free],5m)<10G
```

如果在 5 分钟内测得的可用disk space 始终低于 10 GB，则该触发器会触发。

触发器可以具有以下状态：

触发器状态	描述	在 Zabbix 中
OK	触发器表达式计算结果为“0”（或 FALSE）	未创建问题 问题已解决（如果之前存在）
Problem	触发器表达式计算结果为“1”（或 TRUE）	已创建问题

有时，由于某些原因，触发器表达式无法进行计算。请参见Unknown expression state。

其他资源：

- Trigger examples
- Configuring a trigger
- Best practices

计算频率

每当 Zabbix 服务器接收到属于表达式一部分的新值时，触发器都会重新计算。
收到新值后，表达式中包含的每个函数都会重新计算（而不仅仅是接收到新值的那个函数）。

此外，如果表达式包含以下任一内容，触发器还会每 30 秒（由历史同步器）重新计算一次：

- 日期和时间函数
- nodata() 函数

仅包含趋势函数的触发器，会按照表达式中最小的时间周期进行一次求值。虽然许多触发器函数使用监控项值的历史数据，但趋势函数使用的是趋势数据。

触发器示例

本页汇集了触发器示例。

示例按使用场景排序：

- 不可用的主机
- 不可用的 proxy
- 不可用的 SMTP 集群
- 无法访问的主机
- 意外重启
- 重要文件的变更
- DNS 查询的变更
- 不同主机上的软件不匹配
- 时钟不同步
- 过时的 agent
- 接口抖动
- 高入站流量
- 磁盘空间不足
- 磁盘空间不足（动态阈值）
- CPU 负载高
- CPU 负载比较
- 长期 CPU 负载比较

- 包含特殊字符的字符串

检测不可用主机

```
max(/host/zabbix[host,agent,available],5m)=0
```

如果主机上的 Zabbix agent 已有 5 分钟不可用，则此触发器会触发。
 函数：max
 数据监控项：zabbix[host,agent,available]

替代方案：

```
nodata(/host/agent.ping,5m)=1
```

如果 5 分钟内未从 Zabbix agent 收到任何数据，则此触发器会触发。
 函数：nodata
 数据监控项：agent.ping

检测不可用的 proxy

```
fuzzytime(/host/zabbix[proxy,{$PROXY_NAME},lastaccess],1m)=0
```

如果 Zabbix proxy 数据落后于 Zabbix 服务器时间 1 分钟，此触发器将被触发。
 函数：fuzzytime
 数据监控项：zabbix[proxy,{$PROXY_NAME},lastaccess]

检测不可用的 SMTP 集群

```
last(/smtp1.example.com/net.tcp.service[smtp])=0 and last(/smtp2.example.com/net.tcp.service[smtp])=0
```

如果两个 SMTP 服务器都宕机，此触发器将触发。
 函数：last
 数据监控项：net.tcp.service

检测不可达主机

```
count(/host/icmpping,30m,,"0")>5
```

如果主机在最近 30 分钟内有超过 5 次 ping 不可达，此触发器将被触发。
 函数：count
 数据监控项：icmpping

检测意外重启

```
change(/host/system.uptime)<0
```

如果检测到系统运行时间值出现负变化（表示发生了重启），则此触发器会触发。
 函数：change
 数据监控项：system.uptime

检测重要文件中的更改

```
last(/host/vfs.file.cksum[/etc/passwd],#1)<>last(/host/vfs.file.cksum[/etc/passwd],#2)
```

如果 /etc/passwd 已被更改，此触发器将触发。当上一个 /etc/passwd 校验和与最新的校验和不同时，该表达式为真。类似的表达式可用于监控重要文件中的更改，例如 /etc/passwd、/etc/inetd.conf、/kernel 等。
 函数：last
 数据监控项：vfs.file.cksum

检测 DNS 查询中的变化

```
last(/Zabbix server/net.dns.record[192.0.2.1,{$WEBSITE_NAME},{$DNS_RESOURCE_RECORD_TYPE},2,1])<>"{$WEBSITE
```

请注意第二个操作数周围的引号。

如果查询结果不等于其通常返回的内容，则此触发器会触发：

```
example.com             MX       0 mail.example.com
```

函数：last
 监控项：net.dns.record[192.0.2.1,{$WEBSITE_NAME},{$DNS_RESOURCE_RECORD_TYPE},2,1]，其宏定义如下：

```
{$WEBSITE_NAME} = example.com
{$DNS_RESOURCE_RECORD_TYPE} = MX
```

检测不同主机上不匹配的软件

```
last(/host/vfs.file.contents[/etc/os-release])<>last(/host2/vfs.file.contents[/etc/os-release])
```

如果不同主机上的 Ubuntu 版本不同，此触发器将被触发。请注意，这里的操作数是返回字符串的函数。
 函数：last
 数据监控项：vfs.file.contents

检测不同步的时钟

```
fuzzytime(/host/system.localtime,10s)=0
```

如果客户端本地时间与 Zabbix 服务器时间相差超过 10 秒，则触发器会触发。
 函数：fuzzytime
 监控项：system.localtime

请注意，对于 Zabbix agent，system.localtime 必须配置为被动检查；在 Zabbix agent 2 上，它也可以配置为主动检查。

检测过时的 agent

```
find(/host/agent.version,,"like","beta")=1
```

如果 Zabbix agent 为 beta 版本，此触发器将被触发。需要升级 Zabbix agent。
 函数：find
 数据监控项：agent.version

检测接口抖动

```
changecount(/host/vfs.file.contents["/sys/class/net/eth0/operstate"],1h)>5
```

如果 eth0 的运行状态（up/down/unknown）在一小时内变化超过 5 次，则此触发器会触发。
 函数：changecount
 数据监控项：vfs.file.contents

检测高入站流量

```
min(/host/net.if.in[eth0,bytes],5m)>100K
```

如果在最近五分钟内，eth0 上接收的字节数始终超过 100 KB，则此触发器会触发。很可能有人正在下载大文件。
 函数：min
 数据监控项：net.if.in[eth0,bytes]

检测磁盘空间不足

```
max(/host/vfs.fs.size[/,free],5m)<10G
```

如果可用磁盘空间持续（5 分钟）低于 10 GB，则触发器会触发。

您还可以定义一个恢复表达式：

```
min(/host/vfs.fs.size[/,free],10m)>40G
```

只有当可用磁盘空间持续（10 分钟）高于 40 GB 时，问题才会恢复。
 函数：min
 数据监控项：vfs.fs.size

检测磁盘空间不足（动态阈值）

```
last(/template/hrStorageFree[{#SNMPVALUE}])<last(/template/hrStorageSize[{#SNMPVALUE}])*0.1
```

如果可用存储空间（以分配单位计）降至 10% 以下，则触发器会触发。请注意，这里使用了另一个监控项的值来获取自适应的触发器阈值，适用于通过自动发现发现的各种大小的存储。函数：last

检测 CPU 高负载

```
last(/host/system.cpu.load[all,avg1])>5
```

当平均处理器负载在一分钟内一直高于 5 时，此触发器会触发。

变体：

```
min(/host/system.cpu.load[all,avg1],5m)>2 and time()<060000
```

```
min(/host/system.cpu.load[all,avg1],5m)>2 and not (dayofweek()=7 and time()>230000) and not (dayofweek()=1
```

此类触发器会分析 5 分钟的数据，并且仅在 CPU 负载从未低于 2 时触发。此外，这些触发器会在以下情况下触发：

- 仅在夜间（00:00-06:00）
- 除每周切换时的 2 小时外的任意时间（周日 23:00 - 周一 01:00）

```
(last(/host/system.cpu.load[all,avg1])>5) + (last(/host2/system.cpu.load[all,avg1])>5) + (last(/host3/syst
```

如果三台主机中至少有两台的处理器负载过高，则此触发器会触发。
 函数：last、min、dayofweek、time
 数据监控项：system.cpu.load

比较 CPU 负载

```
avg(/Zabbix server/system.cpu.load,1h)/avg(/Zabbix server/system.cpu.load,1h:now-1d)>2
```

如果今天的平均负载超过昨天同一小时的平均负载（使用时间偏移 now-1d）两倍以上，则触发器会触发。
 函数：avg
 数据监控项：system.cpu.load

比较 CPU 长期负载

```
trendavg(/host/system.cpu.load,1M:now/M)>1.1*trendavg(/host/system.cpu.load,1M:now/M-1M)
```

如果主机上的 CPU 负载在上个月增长超过 10%，则此触发器会触发。
 函数：trendavg
 数据监控项：system.cpu.load

您还可以在触发器配置中使用事件名称字段来构建有意义的告警消息，例如接收类似以下内容的消息：

"Exchange 服务器的负载在 7 月相较于 6 月增长了 24%（0.69 对比 0.56）"

事件名称必须定义为：

```
Load of {HOST.HOST} server increased by {{?100*trendavg(//system.cpu.load,1M:now/M)/trendavg(//system.cpu.
```

对于这类问题，在触发器配置中允许手动关闭也很有用。

检测包含特殊字符的字符串

```
last(/host/vfs.file.contents[/tmp/hello])={$HELLO_MACRO}
```

如果 /tmp/hello 的内容等于 {$HELLO_MACRO} 中定义的字符串，则触发器会触发：

```
{$HELLO_MACRO} = \" //hello ?\"
```

或者，您也可以直接与该字符串进行比较：

```
last(/Zabbix server/vfs.file.contents[/tmp/hello])="\\\" //hello ?\\\""
```

请注意，当直接比较字符串时，特殊字符（\\ 和 "）会被转义。

函数：last
 数据监控项：vfs.file.contents

配置触发器

本页介绍如何在 Zabbix 前端中配置触发器。

请注意，官方 Zabbix 模板包含预定义的触发器。
请参阅将模板应用于监控 Linux 的指南。
也可以对已配置的触发器进行批量更新。

其他资源：

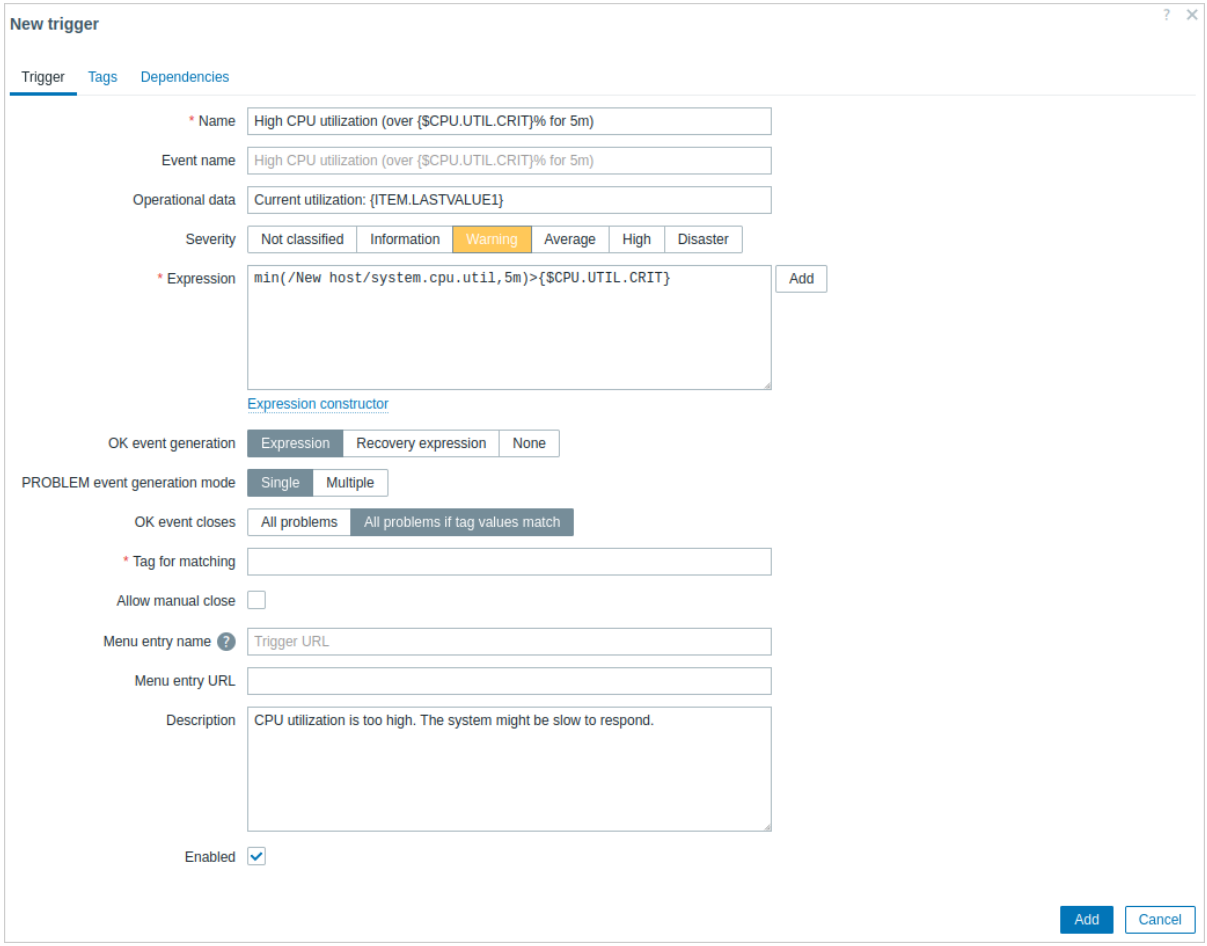
- 使用触发器进行问题检测（概述）
- 触发器示例（按使用场景）
- 触发器表达式和函数（语法详情）
- 最佳实践

配置

要配置触发器，请执行以下操作：

1. 转到 数据采集 > 主机。
2. 单击该主机所在行中的 触发器。
3. 单击右侧的 创建触发器（或单击触发器名称以编辑现有触发器）。
4. 在表单中输入触发器参数。

Trigger 选项卡包含所有必需的触发器属性。

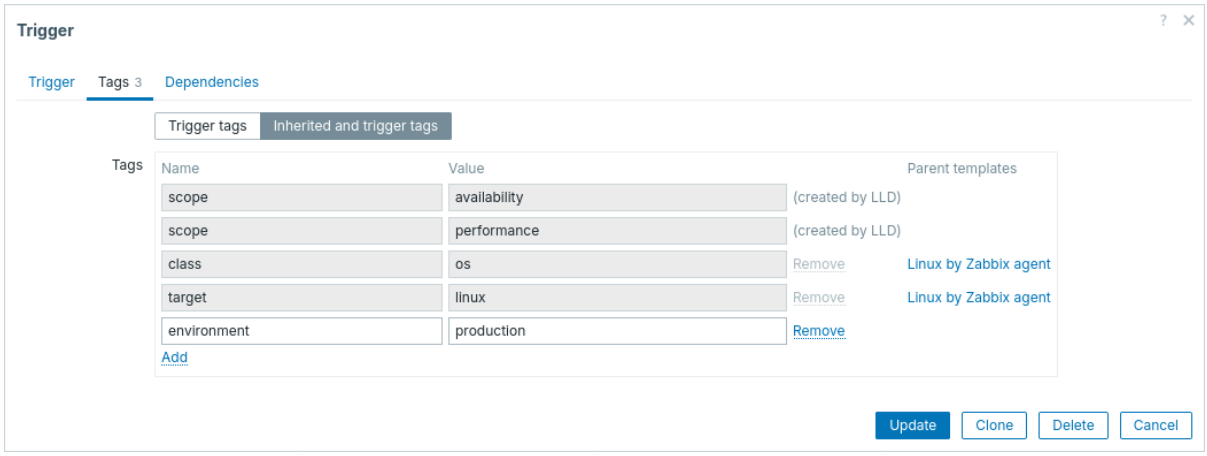


所有必填输入字段都用红色星号标记。

Parameter	Description
Name	触发器名称。 支持的宏 有：{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{ITEM.VALUE}、{ITEM.VALUE.AGE}、{ITEM.VALUE.DATE}、{ITEM.VALUE.TIME}、{ITEM.VALUE.TIMESTAMP}、{ITEM.LASTVALUE}、{ITEM.LASTVALUE.AGE}、{ITEM.LASTVALUE.DATE}、{ITEM.LASTVALUE.TIME}、{ITEM.LASTVALUE.TIMESTAMP}、{ITEM.LOG.*}，以及 {$MACRO} 用户宏。 **$1, $2...$9** 宏可用于引用表达式中的第一、第二.....第九个常量。 注意：如果 $1-$9 宏引用的是相对简单、直接的表达式中的常量，则可以正确解析。例如，如果表达式为 last(/New host/system.cpu.load[percpu,avg1])>5，名称”Processor load above $1 on {HOST.NAME}” 将自动变为”Processor load above 5 on New host”。
Event name	定义后，此名称将用于创建问题事件名称。默认情况下，事件名称与触发器名称相同。 事件名称可用于构建包含问题数据的有意义告警（参见示例）。 支持的宏集合与触发器名称相同，另外还支持 {TIME}、{TIMESTAMP} 和 {?EXPRESSION} 表达式宏。
Operational data	输入包含宏的字符串，以在 监控 > 问题 中显示动态实时数据。或者，也可以将此字段留空，以显示触发器表达式中所有监控项的最新值。 支持的宏集合与触发器名称相同，并且支持动态解析。例如： {ITEM.VALUE<1-9>} 解析为触发器状态变更时刻的监控项值（问题创建、恢复、手动关闭或通过关联关闭）。 {ITEM.LASTVALUE<1-9>} 解析为最新的监控项值。 请注意，手动关闭问题不会产生新值，因此这两个宏仍将显示问题发生时的值。另请注意，如果最新值早于 最大历史显示周期，这两个宏都将解析为 *UNKNOWN*（参见 管理 > 常规）。
Severity	通过单击按钮设置所需的触发器严重性。
Expression	用于定义问题条件的逻辑表达式。 支持时间后缀 和内存大小后缀。 当表达式中包含的所有条件都满足时，即表达式计算结果为 TRUE 时，将创建问题。只要表达式计算结果为 FALSE，问题就会恢复，除非在 恢复表达式中指定了额外的恢复条件。

Parameter	Description
OK event generation	OK 事件生成选项： **Expression** - 基于与问题事件相同的表达式生成 OK 事件； **Recovery expression** - 当问题表达式计算结果为 FALSE 且恢复表达式计算结果为 TRUE 时生成 OK 事件； **None** - 在这种情况下，触发器不会自行恢复为 OK 状态。
Recovery expression	逻辑表达式（可选），用于定义在原始问题表达式已计算为 FALSE 后，问题恢复前必须满足的附加条件。 恢复表达式对触发器滞后 很有用。如果问题表达式仍为 TRUE，则仅靠恢复表达式无法恢复问题。 仅当 *OK event generation* 选择了 “Recovery expression” 时，此字段才可用。
PROBLEM event generation mode	生成问题事件的模式： **Single** - 当触发器首次进入 “Problem” 状态时生成单个事件； **Multiple** - 每次触发器进行 “Problem” 评估时都生成一个事件。
OK event closes	选择 OK 事件是否关闭： **All problems** - 关闭此触发器的所有问题； **All problems if tag values match** - 仅关闭事件标签值匹配的触发器问题。
Tag for matching	输入用于事件关联的事件标签名称。 当 *OK event closes* 属性选择了 “All problems if tag values match” 时会显示此字段，并且在这种情况下为必填。
Allow manual close	勾选后，允许对由此触发器生成的问题事件进行手动关闭。在确认问题事件时可以执行手动关闭。
Menu entry name	如果不为空，这里输入的名称（最多 64 个字符）将作为标签，在多个前端位置中用于 *Menu entry URL* 参数中指定的触发器 URL。若为空，则使用默认名称 *Trigger URL*。 支持的宏集合与触发器名称相同，另外还支持 {EVENT.ID}、{HOST.ID} 和 {TRIGGER.ID}。
Menu entry URL	如果不为空，这里输入的 URL（最多 2048 个字符）将在多个前端位置的事件菜单 中作为链接可用，例如在 监控 > 问题 中单击问题名称时，或在 问题 仪表板小组件中单击时。 支持的宏集合与触发器名称相同，另外还支持 {EVENT.ID}、{HOST.ID} 和 {TRIGGER.ID}。注意：包含密钥值的用户宏不会在 URL 中解析。
Description	用于提供有关此触发器更多信息的文本字段。可包含修复特定问题的说明、负责人员的联系方式等。 支持的宏集合与触发器名称相同。
Enabled	取消勾选此框可在需要时禁用触发器。 已禁用触发器的问题将不再显示在前端中，但不会被删除。

Tags 选项卡允许您定义触发器级别的标签。此触发器的所有问题都会使用此处输入的值进行标记。



此外，*Inherited and trigger tags* 选项允许您查看来自模板级别的标签，前提是该触发器来自该模板。如果有多个模板具有相同的标签，这些标签只会显示一次，模板名称之间用逗号分隔。触发器不会 “继承” 并显示主机级别的标签。

Parameter	Description
Name/Value	设置自定义标签以标记触发器事件。 标签由标签名称和值组成。您可以只使用名称，也可以将其与值配对。一个触发器可以有多个同名但不同值的标签。 事件标签支持用户宏、带上下文的用户宏、低级别发现宏，以及带有 {{ITEM.VALUE}}、{{ITEM.LASTVALUE}} 的宏函数。低级别发现宏可在宏上下文中使用。 {TRIGGER.ID} 宏支持用于触发器标签值。它可用于识别从触发器原型创建的触发器，例如在维护期间抑制这些触发器产生的问题。 如果展开后的值总长度超过 255，则会被截断为 255 个字符。 请参阅事件标签支持的所有宏。 事件标签可用于事件关联、动作条件，并且也会显示在 监控 > 问题或 问题小组件中。

Dependencies 选项卡包含触发器的所有依赖关系。

单击 *Add* 可添加新的依赖关系。

> **Note:**
> 您也可以通过打开现有触发器，单击 *Clone* 按钮，然后以不同名称保存来配置触发器。

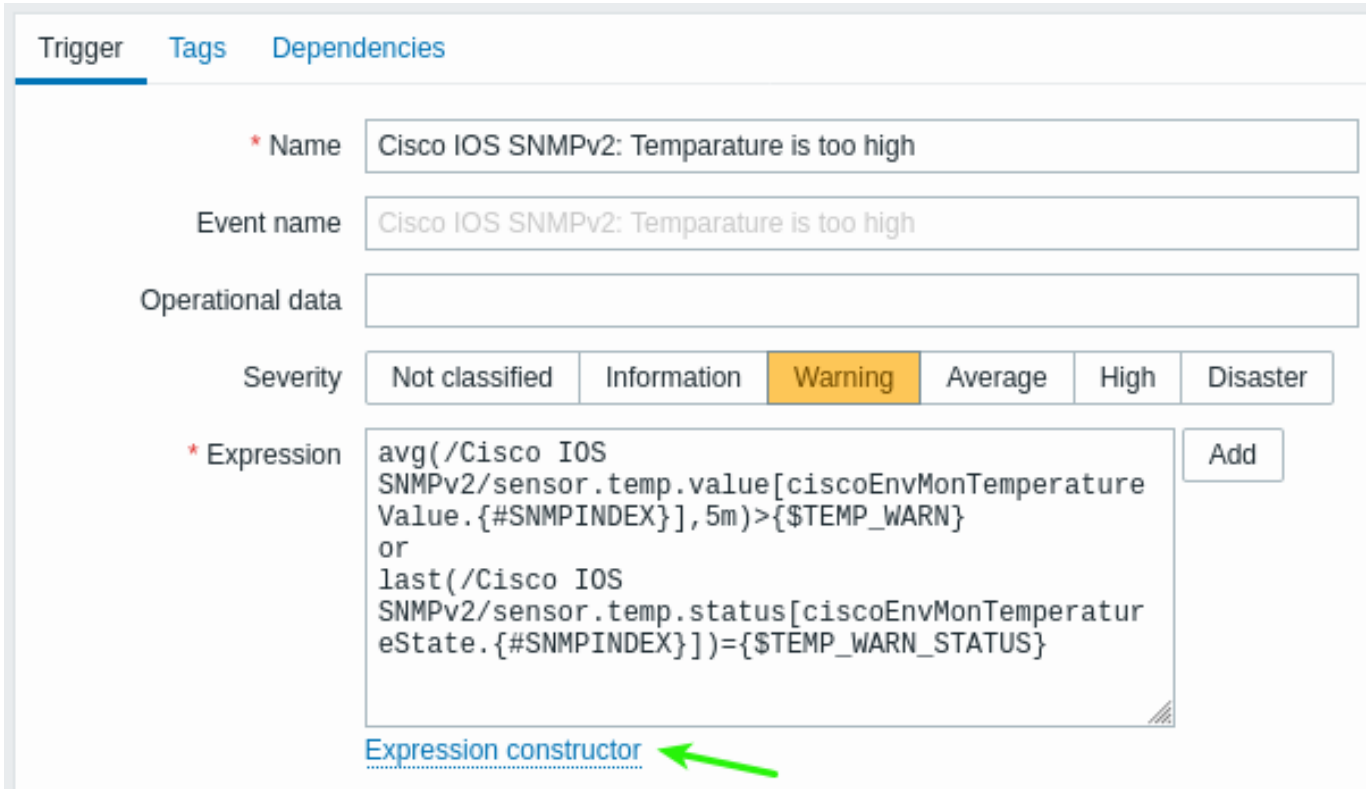
触发器表达式测试

可以根据接收到的值来测试表达式结果会是什么。

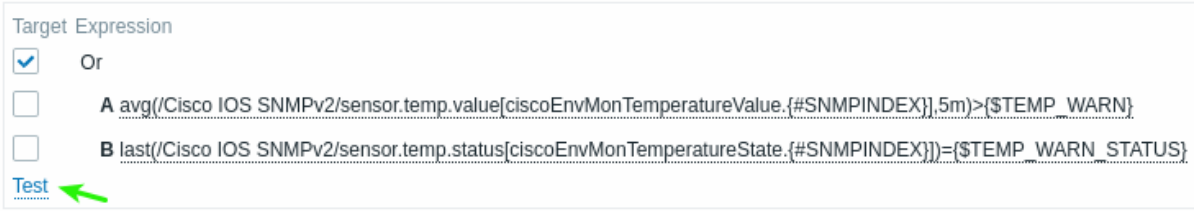
以下来自官方模板的表达式作为示例：

```
avg(/Cisco IOS SNMPv2/sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}],5m)>{$TEMP_WARN}
or
last(/Cisco IOS SNMPv2/sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}])={$TEMP_WARN_STATUS}
```

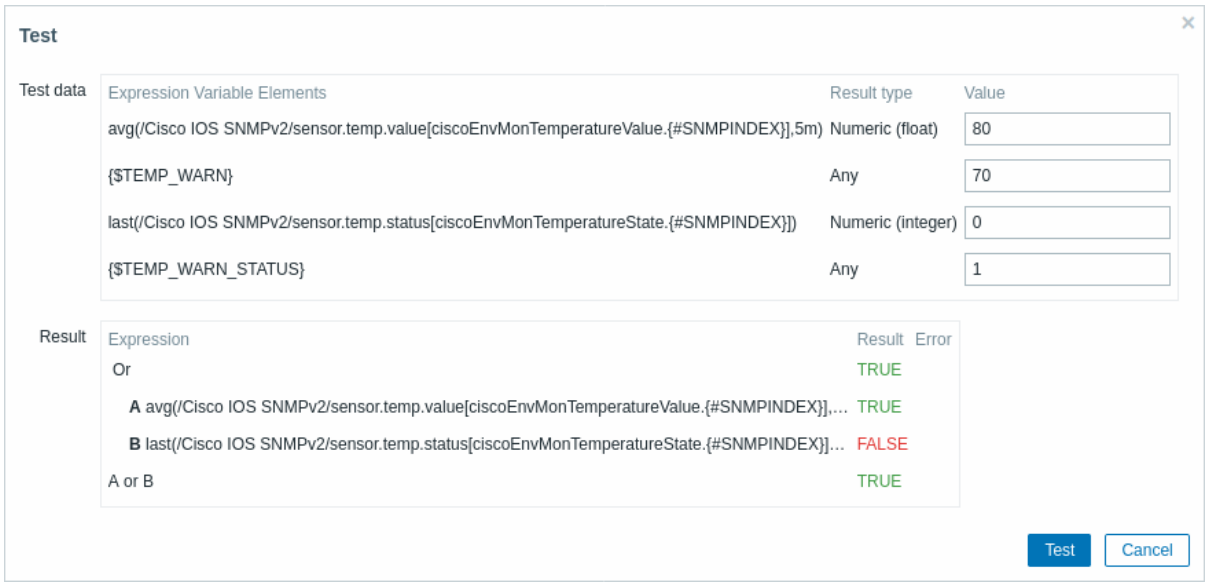
要测试表达式，请点击表达式字段下方的 *Expression constructor*。



在 Expression constructor 中，会列出所有单独的表达式。
要打开测试窗口，请点击表达式列表下方的 *Test*。



在测试窗口中，你可以输入示例值（本例中为'80'、'70'、'0'、'1'），然后点击 *Test* 按钮查看表达式结果。



可以看到各个单独表达式以及整个表达式的结果。

"TRUE" 表示指定的表达式正确。
在这个 A 情况下，"80" 大于 {$TEMP_WARN} 指定的值，本例中为"70"。
如预期一样，会显示"TRUE" 结果。

"FALSE" 表示指定的表达式不正确。
在这个 B 情况下，{$TEMP_WARN_STATUS} 的"1" 需要与指定值相等，本例中为"0"。
如预期一样，会显示"FALSE" 结果。

所选的表达式类型为"OR"。
如果指定条件中至少有一个（此处为 A 或 B）为 TRUE，则整体结果也将为 TRUE。
这意味着当前值超过了警告值，并且已经发生问题。

触发器表达式和函数

概述

本页介绍触发器表达式的语法和求值细节。

简单表达式的语法为：

```
function(/host/item,time_period)<operator><constant>
```

在该表达式中，第一个操作数（位于运算符左侧）是一个带有括号中参数的函数（通常为数据监控项和时间周期）。

该函数用于分析设定时间周期内接收到的数据，并得出一个计算值。

然后使用运算符将该值与第二个操作数进行比较。在此示例中，第二个操作数是一个常量，但也可以是另一个函数。

例如：

```
min(/Zabbix server/net.if.in[eth0,bytes],5m)>100K
```

如果在最近五分钟内，eth0 上接收的字节数始终大于 100 KB，则该触发器会触发。在这种情况下，表达式为真，并会创建一个问题。

触发器表达式非常灵活。在更复杂的表达式中，可以组合多个函数、运算符和常量。

另请参见：

- 触发器示例
- 使用触发器进行问题检测（一般介绍）

函数

函数可用于分析收集的值，例如计算平均值或查找特定字符串。

单击相应的函数组可查看更多详细信息。

函数组	函数
聚合函数	avg, bucket_percentile, count, histogram_quantile, item_count, kurtosis, mad, max, min, skewness, stddevpop, stddevsamp, sum, sumofsquares, varpop, varsamp
Foreach 函数	avg_foreach, bucket_rate_foreach, count_foreach, exists_foreach, last_foreach, max_foreach, min_foreach, sum_foreach
按位函数	bitand, bitlshift, bitnot, bitor, bitrshift, bitxor
日期和时间函数	date, dayofmonth, dayofweek, now, time
历史数据函数	change, changecount, count, countunique, find, first, firstclock, fuzzytime, jsonpath, last, lastclock, logeventid, logseverity, logsource, logtimestamp, monodec, monoinc, nodata, percentile, rate, xmlxpath
趋势函数	baselinedev, baselinewma, trendavg, trendcount, trendmax, trendmin, trendstl, trendsum
数学函数	abs, acos, asin, atan, atan2, avg, cbrt, ceil, cos, cosh, cot, degrees, e, exp, expm1, floor, log, log10, max, min, mod, pi, power, radians, rand, round, signum, sin, sinh, sqrt, sum, tan, truncate
运算符函数	between, in
预测函数	forecast, timeleft
字符串函数	ascii, bitlength, bytelength, char, concat, contains, insert, jsonpath, left, length, ltrim, mid, repeat, replace, right, rtrim, substring, trim, xmlxpath

除非另有说明，这些函数支持用于：

- 触发器表达式
- 计算型监控项
- 表达式宏

Foreach 函数仅支持用于聚合计算。

通常，函数返回数值以供比较。当返回字符串时，可以使用 **=** 和 **<>** 运算符进行比较（请参阅检测不同主机上的不匹配软件示例）。

函数参数

函数参数允许您指定：

- 监控项键（格式为 /主机/键），用于引用主机监控项历史记录的函数
- 时间段（以及其他特定于函数的参数）
- 其他表达式

监控项键值

被引用的监控项必须处于受支持状态（**nodata()** 函数除外，它也会对不受支持的监控项进行计算）。

在第一个参数中省略主机名（即如 function(//key,parameter,...)）仅在特定上下文中受支持：

- 在计算型监控项的公式中
- 在表达式宏中，可用于：
  - 事件名称字段
  - 图形名称
  - “主机”和“触发器”地图元素的标签

在这些上下文中，您还可以使用{HOST.HOST}宏。
在事件名称字段和“触发器”地图元素中，可以使用 {HOST.HOST<1-9>} 来引用触发器表达式中的特定监控项。
当在这些上下文中省略主机名或用 {HOST.HOST} 替换时，该引用将指向触发器表达式中的第一个监控项，或图形中的第一个监控项。
在这些受支持的上下文之外，在触发器表达式中省略主机名将导致错误。
有关在事件名称宏中使用双斜杠的示例，请参见比较长期 CPU 负载。

时间段

特定于函数的参数放置在监控项键之后，并使用逗号与监控项键分隔。

大多数数值函数都接受时间段作为参数。它用于指定我们关注的时间间隔。时间段可以指定为时间长度（30s、10m、1h），也可以指定为值范围（#5 - 表示最近的五个值）。

您可以使用秒数或时间后缀来表示时间段。在参数前加上井号后，该参数具有不同的含义：

表达式	描述
sum(/host/key,**10m**)	最近 10 分钟内的值之和。
sum(/host/key,**#10**)	最近 10 个值之和。

带井号的参数在 **last** 函数中具有不同的含义——它们表示之前第 N 个值。例如，给定值 30、70、20、60、50（从最新到最旧）：

- last(/host/key,#2) 将返回’70’
- last(/host/key,#5) 将返回’50’

时间段的计算截止到“现在”——这里的“现在”是触发器最近一次重新计算的时间（请参阅计算频率）；“现在”并不是服务器的当前时间。

时间段指定以下内容之一：

- “现在 - 时间段”与“现在”之间的所有值（或者，在应用时间偏移后，“现在 - 时间偏移 - 时间段”与“现在 - 时间偏移”之间的所有值）。
- 从过去到“现在”的指定数量的值。

如果指定的评估期间不存在数据，则使用此函数的触发器或计算监控项将变为不支持。

请注意：

- 如果触发器中只使用了一个函数（引用数据历史记录），则“现在”始终是最近接收到值的时间。例如，如果上一个值是在一小时前接收到的，则时间段将被视为截止到一小时前接收到的最新值。
- 新触发器会在接收到第一个值后立即进行计算（历史函数）；对于日期和时间以及nodata()函数，则会在 30 秒内进行计算。因此，即使触发器创建后尚未经过设定的时间段（例如一小时），触发器仍会进行计算。即使时间范围设置为例如最近十个值，触发器也会在接收到第一个值后进行计算。

时间偏移

支持使用时间或值计数作为函数参数的可选时间偏移。此参数允许您引用过去某个时间段内的数据。

时间偏移以 now 开头，用于指定当前时间，后跟 +N< 时间单位 > 或 -N< 时间单位 >，用于增加或减少 N 个时间单位。

例如，avg(/host/key,1h:now-1d) 将返回一天前一小时内的平均值。

> **Attention:**
> 以月（M）和年（y）指定的时间偏移仅支持趋势函数。其他函数支持秒（s）、分钟（m）、小时（h）、天（d）和周（w）。

使用绝对时间段的时间偏移

时间偏移参数支持绝对时间段，例如，一天从午夜到午夜、一周从星期一到星期日、一个月从第一天到最后一天。

绝对时间段的时间偏移以 now 开头，用于指定当前时间，后跟任意数量的时间操作：/< 时间单位 > - 定义时间单位的开始和结束，例如一天从午夜到午夜；+N< 时间单位 > 或 -N< 时间单位 > - 用于增加或减少 N 个时间单位。

请注意，时间偏移的值可以大于或等于 0，而时间段的最小值为 1。

参数	描述
1d:now/d	昨天
1d:now/d+1d	今天
2d:now/d+1d	过去 2 天
1w:now/w	上周
1w:now/w+1w	本周

其他表达式

函数参数可以包含其他表达式，如以下语法所示：

```
min(min(/host/key,1h),min(/host2/key2,1h)*10)
```

请注意，如果函数引用了监控项历史数据，则不能使用其他表达式。例如，以下语法不允许：

~~`min(/host/key,#5*10)`~~

虽然在触发器中，将其他触发器表达式用作函数参数时仅限于非历史函数，但此限制不适用于计算型监控项。

运算符

以下运算符可用于触发器（按执行优先级降序排列）：

优先级	运算符	定义	未知值说明	将操作数强制转换为 float [1]
1	**-**	一元负号	**-**Unknown → Unknown	是
2	**not**	逻辑非	**not** Unknown → Unknown	是
3	*	乘法	0 * Unknown → Unknown (是的，结果是 Unknown，而不是 0——这样可避免在算术运算中丢失 Unknown) 1.2 * Unknown → Unknown	是
	/	除法	Unknown / 0 → error Unknown / 1.2 → Unknown 0.0 / Unknown → Unknown	是
4	+	算术加法	1.2 + Unknown → Unknown	是
	-	算术减法	1.2 - Unknown → Unknown	是
5	<	小于。该运算符定义为： A<B ⇔ (A<B-0.000001)	1.2 < Unknown → Unknown	是
	<=	小于或等于。该运算符定义为： A<=B ⇔ (A≤B+0.000001)	Unknown <= Unknown → Unknown	是
	>	大于。该运算符定义为： A>B ⇔ (A>B+0.000001)		是
	>=	大于或等于。该运算符定义为： A>=B ⇔ (A≥B-0.000001)		是

优先级	运算符	定义	未知值说明	将操作数强制转换为 float [1]
6	**=**	等于。该运算符定义为： A=B ⇔ (A≥B-0.000001) and (A≤B+0.000001)		否 [1]
	<>	不等于。该运算符定义为： A<>B ⇔ (A<B-0.000001) or (A>B+0.000001)		否 [1]
7	**and**	逻辑与	0 **and** Unknown → 0 1 **and** Unknown → Unknown Unknown **and** Unknown → Unknown	是
8	**or**	逻辑或	1 **or** Unknown → 1 0 **or** Unknown → Unknown Unknown **or** Unknown → Unknown	是

[1] 在以下情况下，字符串操作数仍会被转换为数值：

- 另一个操作数是数值
- 对操作数使用了 **=** 或 **<>** 以外的运算符

(如果转换失败，则数值操作数会被转换为字符串操作数，并将两个操作数作为字符串进行比较。)

not、**and** 和 **or** 运算符区分大小写，且必须使用小写。它们还必须由空格或括号包围。

除一元 **-** 和 **not** 外，所有运算符都具有从左到右的结合性。一元 **-** 和 **not** 是非结合的（这意味着应使用 **-(-1)** 和 **not (not 1)**，而不是 **--1** 和 **not not 1**）。

求值结果：

- **<**、**<=**、**>**、**>=**、**=**、**<>** 运算符在指定关系为真时，应在触发器表达式中产生 “1”，为假时产生 “0”。如果至少有一个操作数为 Unknown，则结果为 Unknown；
- 对于已知操作数，**and** 在其两个操作数都不等于 “0” 时产生 “1”；否则产生 “0”；对于未知操作数，**and** 仅当其中一个操作数等于 “0” 时产生 “0”；否则产生 “Unknown”；
- 对于已知操作数，**or** 在其任一操作数不等于 “0” 时产生 “1”；否则产生 “0”；对于未知操作数，**or** 仅当其中一个操作数不等于 “0” 时产生 “1”；否则产生 “Unknown”；
- 对于已知操作数，逻辑非运算符 **not** 的结果为：如果其操作数的值不等于 “0”，则为 “0”；如果其操作数的值等于 “0”，则为 “1”。对于未知操作数，**not** 产生 “Unknown”。

未知表达式状态

当出现以下情况时，触发器表达式中可能会出现未知操作数：

- 使用了不受支持的监控项
- 对受支持的监控项进行函数求值时出错

在这种情况下，触发器表达式通常会计算为 Unknown（因为无法对其进行计算）

可以在未知触发器上接收通知。

例外

尽管存在未知操作数，在某些情况下触发器表达式仍可计算出已知结果（Problem/OK）：

- `nodata()` 函数会被计算，不论所引用的监控项是否受支持。
- 包含 AND/OR 的表达式在两种情况下可以计算出已知结果：
  - 情况 **1**：”`1 or some_function(unsupported_item1) or some_function(unsupported_item2) or ...`” 会计算出已知结果（1 或”Problem”），
  - 情况 **2**：”`0 and some_function(unsupported_item1) and some_function(unsupported_item2) and ...`” 会计算出已知结果（0 或”OK”）。
- 如果对受支持的监控项进行函数求值时出错，则函数值会变为 Unknown，并在后续表达式计算中作为未知操作数参与计算。

未知操作数只有在上述逻辑表达式中才可能“消失”。在算术表达式中，未知操作数总是会导致 Unknown（除以 0 的情况除外）。

Attention:
未知表达式状态不会改变触发器状态（Problem/OK）。因此，如果触发器状态原本是”Problem”（参见情况 1），即使已知部分恢复（1 变为 0），它仍会保持在问题状态，因为此时表达式会被计算为 Unknown，而这不会改变触发器状态。

如果包含多个不受支持监控项的触发器表达式计算结果为 Unknown，则前端中的错误消息会引用最后一个被计算的不受支持监控项。

值缓存

触发器求值所需的值由 Zabbix 服务器缓存。因此，在服务器重启后的一段时间内，触发器求值会导致数据库负载升高。

当监控项历史值被删除时（无论是手动删除还是由 housekeeper 删除），值缓存都不会被清除，因此服务器将继续使用缓存的值，直到这些值早于触发器函数中定义的时间段，或者服务器被重启。

Note:
如果缓存中没有最近的数据，并且函数中未定义查询时间段，Zabbix 默认会向过去最多回溯一周，从数据库中查询历史值。

聚合函数

除非另有说明，这里列出的所有函数都支持以下场景：

- 触发器表达式
- 计算型监控项公式
- 表达式宏

聚合函数可以处理以下任一输入：

- 监控项历史数据，例如 `min(/host/key,1h)`
- foreach 函数 作为唯一参数，例如 `min(last_foreach(/*/key))`（仅适用于计算型监控项公式；如需在触发器中使用，请通过其 key 引用对应的计算型监控项）

以下仅列出函数名称，不包含额外信息。点击函数可查看完整详情。

Function	Description
avg	定义的评估周期内监控项的平均值。
bucket_percentile	根据直方图的桶计算百分位数。
count	foreach 函数返回的数组中值的数量。
histogram_quantile	根据直方图的桶计算 φ 分位数。
item_count	配置中符合筛选条件的现有监控项数量。
kurtosis	定义的评估周期内采集值分布的“尾部厚度”。
mad	定义的评估周期内采集值的中位数绝对偏差。
max	定义的评估周期内监控项的最大值。
min	定义的评估周期内监控项的最小值。
skewness	定义的评估周期内采集值分布的偏斜度。
stddevpop	定义的评估周期内采集值的总体标准差。
stddevsamp	定义的评估周期内采集值的样本标准差。
sum	定义的评估周期内采集值的总和。
sumofsquares	定义的评估周期内采集值的平方和。
varpop	定义的评估周期内采集值的总体方差。
varsamp	定义的评估周期内采集值的样本方差。

通用参数

- `/host/key` 是引用主机监控项历史数据的函数的通用必需第一个参数。
- `(sec|#num)<:time shift>` 是引用主机监控项历史数据的函数的通用第二个参数，其中：
  - **sec** - 以秒为单位的最大评估周期（可以使用时间后缀）。
  - **#num** - 最新收集值中的最大评估范围（前面带有井号时）。
  - **time shift**（可选）允许将评估时间点向过去移动。请参阅有关指定时间偏移的更多详细信息。

函数详情

有关函数参数的一些一般说明：

- 函数参数以逗号分隔

- 可选函数参数（或参数部分）以 < > 表示
- 每个函数的特定参数会在对应函数中说明
- /host/key 和 (sec|#num)<:time shift> 参数绝不能加引号

avg(/host/key,(sec|#num)<:time shift>)

在定义的评估周期内监控项的平均值。
 支持的值类型：*Float*、*Integer*。
 支持的foreach functions：*avg_foreach*、*count_foreach*、*exists_foreach*、*last_foreach*、*max_foreach*、*min_foreach*、*sum_foreach*。

参数：请参见通用参数。

当需要将当前平均值与一段时间之前的平均值进行比较时，时间偏移非常有用。

示例：

```
avg(/host/key,1h) # 计算截至当前的最近一小时平均值
avg(/host/key,#5) # 计算最近五个值的平均值
avg(/host/key,1h:now-1d) # 计算从当前时间往前 25 小时到往前 24 小时这一小时区间内的平均值
avg(last_foreach(/*/system.cpu.load[,avg1]?[group="MySQL Servers"])) # 计算所有 MySQL 服务器的平均处理器负载
avg(/host/proc.num,5m)>300 # 如果最近 5 分钟内的平均进程数超过 300，则触发触发器
```

bucket_percentile(item filter,time period,percentage)

从直方图的桶中计算百分位数。

参数：

- **item filter** - 参见item filter；
- **time period** - 参见time period；
- **percentage** - 百分比（0-100）。

注释：

- 仅在计算型监控项中受支持；
- 此函数是 histogram_quantile(percentage/100, bucket_rate_foreach(item filter, time period, 1)) 的别名。

示例：

```
bucket_percentile(/*/http_request_duration_seconds_bucket[*],5m,95) #calculate the 95th percentile respons
bucket_percentile(/*/apiserver_request_duration_seconds_bucket[*],10m,99) #calculate the 99th percentile A
```

count(func_foreach(item filter,<time period>),<operator>,<pattern>)

由 foreach 函数返回的数组中的值的数量。
 支持的foreach 函数：*avg_foreach*、*count_foreach*、*exists_foreach*、*last_foreach*、*max_foreach*、*min_foreach*、*sum_foreach*。

参数：

- **func_foreach** - 要统计返回值数量的 foreach 函数。详情请参见foreach 函数。请注意，count_foreach 和 bucket_rate_foreach 支持附加参数。
- **item filter** - 参见item filter；
- **time period** - 参见time period；
- **operator**（必须用双引号括起来）。支持的 operators：
eq - 等于
ne - 不等于
gt - 大于
ge - 大于或等于
lt - 小于
le - 小于或等于
like - 当包含 pattern 时匹配（区分大小写）
bitand - 按位 AND
regexp - 与 pattern 中给定的正则表达式进行区分大小写匹配
iregexp - 与 pattern 中给定的正则表达式进行不区分大小写匹配
- **pattern** - 所需的模式（字符串参数必须用双引号括起来）；当第三个参数指定了 *operator* 时支持。

注释：

- 将 **count()** 与基于历史数据的 foreach 函数（max_foreach、avg_foreach 等）一起使用可能会带来性能影响，而仅使用配置数据的 **exists_foreach()** 则不会有这种影响。
- 可选参数 *operator* 或 *pattern* 在逗号后不能留空，只能完全省略。
- 当第三个参数为 *bitand* 时，第四个 pattern 参数可以指定为两个数字，并用'/' 分隔：**number_to_compare_with/mask**。count() 会对值和 *mask* 进行 "按位 AND" 运算，并将结果与 *number_to_compare_with* 比较。如果 "按位 AND" 的结果等于 *number_to_compare_with*，则该值会被计数。
 如果 *number_to_compare_with* 和 *mask* 相等，则只需指定 *mask*（不带'/'）。
- 当第三个参数为 *regexp* 或 *iregexp* 时，第四个 pattern 参数可以是普通正则表达式或全局 正则表达式（以'@' 开头）。对于全局正则表达式，大小写敏感性继承自全局正则表达式设置。为了进行 regexp 匹配，浮点值始终会以小数点后 4 位表示。另请注意，对于较大的数字，十进制（存储在数据库中）和二进制（由 Zabbix 服务器使用）表示之间的差异可能会影响第 4 位小数。

示例：

```
count(max_foreach(/*/net.if.in[*],1h)) # 计算截至当前过去 1 小时内收到数据的 net.if.in 监控项数量
count(last_foreach(/*/vfs.fs.size[*,pused]),"gt",95) # 计算磁盘空间使用率超过 95% 的文件系统数量
```

histogram_quantile(quantile,bucket1,value1,bucket2,value2,...)

从直方图的桶中计算 φ 分位数。
 支持的foreach 函数：*bucket_rate_foreach*。

参数：

- **quantile** - 0 ≤ φ ≤ 1；
- **bucketN, valueN** - 手动输入的参数对（>=2），或bucket_rate_foreach 的返回结果。

说明：

- 仅在计算型监控项中受支持；
- 在功能上对应于 PromQL 的'histogram_quantile'；
- 如果最后一个 "Infinity" 桶（"+*inf*"）的值等于 0，则返回 -1。

示例：

```
histogram_quantile(0.75,1.0,last(/host/rate_bucket[1.0]),"+Inf",last(/host/rate_bucket[Inf]))
histogram_quantile(0.5,bucket_rate_foreach(//item_key,30s))
```

item_count(item filter)

配置中符合筛选条件的现有监控项数量。
 支持的值类型：整数。

参数：

- **item filter** - 用于选择监控项的条件，允许按主机组、主机、监控项键和值标签进行引用。支持通配符。更多详细信息请参见item filter。

注释：

- 仅在计算型监控项中受支持；
- 作为 *count(exists_foreach(item_filter))* 函数的别名工作。

示例：

```
item_count(/*/agent.ping?[group="Host group 1"]) # 计算"Host group 1"中具有 agent.ping 监控项的主机数量
```

kurtosis(/host/key,(sec|#num)<:time shift>)

在定义的评估周期内，已收集值的概率分布的 "尾部厚度"。另请参见：Kurtosis。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

示例：

```
kurtosis(/host/system.cpu.util[,user],5m)>5 # 在 5 分钟内 CPU 利用率出现尖峰时触发
```

mad(/host/key,(sec|#num)<:time shift>)

定义的评估周期内已收集值的中位数绝对偏差。另请参见：中位数绝对偏差。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

示例：

```
last(/host/system.cpu.util[,user])>avg(/host/system.cpu.util[,user],1h)+3*mad(/host/system.cpu.util[,user]
```

max(/host/key,(sec|#num)<:time shift>)

在定义的评估周期内，某监控项的最大值。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*avg_foreach*、*count_foreach*、*exists_foreach*、*last_foreach*、*max_foreach*、*min_foreach*、*sum_foreach*。

参数：请参见通用参数。

示例：

```
max(/host/key,1h) - min(/host/key,1h) # 计算截至当前的最近一小时内最大值与最小值之间的差值（值的增量）
max(last_foreach(/*/vfs.fs.size[*,pused]?[group="Linux servers"])) # 返回所有 Linux 服务器中最高的磁盘使用率
```

min(/host/key,(sec|#num)<:time shift>)

在定义的评估周期内，监控项的最低值。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*avg_foreach*、*count_foreach*、*exists_foreach*、*last_foreach*、*max_foreach*、*min_foreach*、*sum_foreach*。

参数：请参见通用参数。

示例：

```
max(/host/key,1h) - min(/host/key,1h) # 计算截至当前的最近一小时内最大值与最小值之间的差值（值的增量）
min(last_foreach(/*/vfs.file.contents["/sys/class/net/enp0s3/operstate"]?[group="Linux servers"])) # 返回所
```

skewness(/host/key,(sec|#num)<:time shift>)

在定义的评估周期内，已收集值的概率分布的不对称性。另请参见：Skewness。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

示例：

```
skewness(/host/vm.memory.size[used],5m)<-20 #trigger on sudden negative drops indicating crashes/restarts
```

stddevpop(/host/key,(sec|#num)<:time shift>)

已收集值在定义的评估周期内的总体标准差。另请参见：标准差。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

示例：

```
stddevpop(/host/system.cpu.util[,user],10m)>7 # 如果 CPU 使用率波动较大则触发（10 分钟内的总体标准差高于 7）
```

stddevsamp(/host/key,(sec|#num)<:time shift>)

已收集值在定义的评估周期内的样本标准差。另请参见：标准差。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

此函数至少需要两个数据值才能工作。

示例：

```
stddevsamp(/host/system.cpu.util[,user],10m)>7 # 如果 CPU 使用率波动较大则触发（10 分钟内的样本标准差高于 7）
```

sum(/host/key,(sec|#num)<:time shift>)

定义的评估周期内已采集值的总和。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*avg_foreach*、*count_foreach*、*exists_foreach*、*last_foreach*、*max_foreach*、*min_foreach*、*sum_foreach*。

参数：请参见通用参数。

示例：

```
sum(last_foreach(/*/net.if.in[*]?[group="Linux servers"])) # 计算所有 Linux 服务器的网络入站流量总量
sum(last_foreach(/*/vfs.fs.size[/,total]?[group="MySQL Servers"])) # 计算所有 MySQL 服务器的磁盘空间总量
sum(last_foreach(/*/net.dns[*,*,*])) # 计算成功 DNS 检查的总数
```

sumofsquares(/host/key,(sec|#num)<:time shift>)

定义的评估周期内已收集值的平方和。
 支持的值类型：浮点数、整数。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

示例：

```
sumofsquares(/host/calculated.net.if.rate,1m) # 网络流量速率（入站与出站）的平方和
sumofsquares(last_foreach(/*/net.if.in[*]?[group="Linux servers"])) # 所有 Linux 服务器上最新入站网络流量值的平方
```

varpop(/host/key,(sec|#num)<:time shift>)

在定义的评估周期内收集到的值的总体方差。另请参见：方差。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

示例：

```
varpop(/host/system.cpu.util[,user],10m)>50 # 如果 CPU 使用率波动很大则触发（10 分钟内方差高于 50）
```

varsamp(/host/key,(sec|#num)<:time shift>)

定义评估周期内已采集值的样本方差。另请参见：方差。
 支持的值类型：*Float*、*Integer*。
 支持的foreach 函数：*last_foreach*。

参数：请参见通用参数。

此函数至少需要两个数据值才能工作。

示例：

```
varsamp(/host/system.cpu.util[,user],10m)>50 # 如果 CPU 使用率波动很大则触发（10 分钟内样本方差高于 50）
```

请参见所有支持的函数。

Foreach 函数

概述

foreach 函数用于聚合计算，根据所使用的 监控项过滤器为每个被选中的监控项返回一个聚合值。返回的是一个值数组。

例如，`avg_foreach` 函数会返回一个值数组，其中每个值都是在指定时间间隔内所选监控项的 平均历史值。

> **Attention:**
> foreach 函数仅支持在计算型监控项 公式中使用。它们不能在普通监控项键值中调用，也不能直接用于构建触发器表达式。在触发器表达式中，可以通过引用其键值来使用计算型监控项。

item filter 是 foreach 函数语法的一部分。监控项过滤器支持使用通配符，因此可以非常灵活地选择所需的监控项。

支持的函数

Function	Description
avg_foreach	返回每个监控项的平均值。
bucket_rate_foreach	返回成对的值（bucket 上限，速率值），适用于`histogram_quantile()` 函数，其中 “bucket 上限” 是由 <parameter number> 参数 定义的监控项 key 参数的值。
count_foreach	返回每个监控项的值数量。
exists_foreach	为每个已启用的监控项返回'1'。
last_foreach	返回每个监控项的最后一个值。
max_foreach	返回每个监控项的最大值。
min_foreach	返回每个监控项的最小值。
sum_foreach	返回每个监控项的值总和。

函数语法

Foreach 函数支持两个通用参数：监控项过滤器（详见下文）和 时间周期：

```
foreach_function(item filter,time period)
```

例如：

```
avg_foreach(/*/mysql.qps?[group="MySQL Servers"],5m)
```

将返回 MySQL 服务器组中每个 “mysql.qps” 监控项的五分钟平均值。

请注意，某些函数还支持额外的参数。

监控项过滤器语法

监控项过滤器：

```
/host/key[parameters]?[conditions]
```

由四个部分组成，其中：

- host - 主机名称
- key - 监控项键值（不含参数）
- parameters - 监控项键值参数

- conditions - 基于主机组和/或监控项标签的条件（以表达式形式）

仅允许在 conditions 表达式内部使用空格。

通配符用法

- 可使用通配符替代主机名称、监控项键值或单个监控项键值参数。
- host 或 key 中必须有一项是不使用通配符明确指定的。因此，`/host/*` 和 `/*/key` 是有效过滤器，而 `/*/*` 无效。
- 通配符不能用于主机名称、监控项键值或监控项键值参数的一部分。
- 通配符不能匹配多个监控项键值参数。因此，必须分别为每个参数单独指定通配符（即 `key[abc,*,*]`）。

条件表达式

conditions 表达式支持：

- 操作数：
  - group - 主机组
  - tag - 监控项标签
  - "<text>" - 字符串常量，使用转义字符 \ 对 " 和 \ 进行转义
- 区分大小写的字符串比较运算符：=, <>
- 逻辑运算符：and, or, not
- 使用括号分组：( )

字符串常量必须使用引号。仅支持区分大小写的完整字符串比较。

Warning:
在过滤器中指定标签时（即 `tag="tagname:value"`），冒号":" 用作分隔符。
其后的所有内容都会被视为标签值。因此，目前不支持指定名称中包含":" 的标签。

示例

可使用复杂过滤器来引用监控项键值、主机组和标签，如下例所示：

Syntax example	Description
`/host/key[abc,*]`	匹配此主机上的类似监控项。
`/*/key`	匹配任意主机上的相同监控项。
`/*/key?[group="ABC" and tag="tagname:value"]`	匹配 ABC 组中任意主机上带有'tagname:value' 标签的相同监控项。
`/*/key[a,*,c]?[(group="ABC" and tag="Tag1") or (group="DEF" and (tag="Tag2" or tag="Tag3:value"))]`	匹配 ABC 或 DEF 组中任意主机上带有相应标签的类似监控项。

所有被引用的监控项都必须存在并采集数据。
只有已启用主机上的已启用监控项才会被纳入计算。
处于不支持状态的监控项不会被纳入。

Attention:
如果被引用监控项的键值发生更改，则必须手动更新过滤器。

指定父主机组时，将包含该父组及其所有嵌套主机组中的监控项。

时间段

第二个参数用于指定聚合的时间段。时间段只能表示为时间，不支持值的数量（以 # 为前缀）。

此参数中可以使用支持的单位符号以简化表示，例如，使用 5m（五分钟）代替 300s（300 秒），或使用 1d（一天）代替 86400（86400 秒）。

对于 `last_foreach` 函数，时间段是可选参数（Zabbix 7.0 及更高版本支持），可以省略：

```
last_foreach(/*/key?[group="host group"])
```

`exists_foreach` 函数不支持时间段。

附加参数

bucket_rate_foreach

`bucket_rate_foreach` 函数支持第三个可选参数：

```
bucket_rate_foreach(item filter,time period,<parameter number>)
```

其中，<parameter number> 是 item key 中"bucket" 值的位置。例如，如果 myItem[aaa,0.2] 中的"bucket" 值是 0.2，那么它的位置是 2。

<parameter number> 的默认值为 1。

count_foreach

count_foreach 函数支持第三和第四个可选参数：

```
count_foreach(item filter,time period,<operator>,<pattern>)
```

其中：

- **operator** 是 item 值的条件运算符（必须用双引号括起来）。支持的 operators:
eq - 等于
ne - 不等于
gt - 大于
ge - 大于或等于
lt - 小于
le - 小于或等于
like - 如果包含模式则匹配（区分大小写）
bitand - 按位 AND
regexp - 与 pattern 中给出的正则表达式匹配（区分大小写）
iregexp - 与 pattern 中给出的正则表达式匹配（不区分大小写）
- **pattern** 是必需的模式（字符串参数必须用双引号括起来）；当第三个参数中指定了 *operator* 时才支持。

说明：

- 可选参数 *operator* 或 *pattern* 不能在逗号后留空，只能完全省略。
- 当第三个参数为 *bitand* 时，第四个 pattern 参数可以指定为两个数字，并用 / 分隔：**number_to_compare_with/mask**。count_foreach() 会对值和 *mask* 进行 "按位 AND" 运算，并将结果与 *number_to_compare_with* 比较。如果 "按位 AND" 的结果等于 *number_to_compare_with*，则该值会被计数。
 如果 *number_to_compare_with* 和 *mask* 相等，则只需指定 *mask*（不带 /）。
- 当第三个参数为 *regexp* 或 *iregexp* 时，第四个 pattern 参数可以是普通正则表达式，也可以是全局 正则表达式（以 @ 开头）。对于全局正则表达式，大小写敏感性继承自全局正则表达式设置。在进行 regexp 匹配时，浮点值始终会以小数点 . 后 4 位数字表示。另请注意，对于较大的数字，十进制（存储在数据库中）和二进制（由 Zabbix 服务器使用）表示之间的差异可能会影响第 4 位小数。

有关在 foreach 函数中使用的更多细节和示例，请参见聚合计算。

取决于可用性的行为

下表说明了在主机/监控项以及历史数据可用性受限的情况下，各个函数的行为方式。

Function	已禁用的主机	有数据的不可用主机	无数据的不可用主机	已禁用的监控项	不支持的监控项	数据检索错误 (SQL)
avg_foreach	忽略	返回平均值	忽略	忽略	忽略	忽略
bucket_rate_foreach	忽略	返回 bucket rate	忽略	忽略	忽略	忽略
count_foreach	忽略	返回计数	0	忽略	忽略	忽略
exists_foreach	忽略	1	1	忽略	1	n/a
last_foreach	忽略	返回最后值	忽略	忽略	忽略	忽略
max_foreach	忽略	返回最大值	忽略	忽略	忽略	忽略
min_foreach	忽略	返回最小值	忽略	忽略	忽略	忽略
sum_foreach	忽略	返回总和	忽略	忽略	忽略	忽略

如果监控项被忽略，则不会向聚合中添加任何内容。

按位函数

此处列出的所有函数都支持用于：

- 触发器表达式
- 计算型监控项公式
- 表达式宏

这些函数仅列出名称，不包含其他信息。点击函数可查看完整详情。

Function	Description
bitand	监控项值与掩码进行 "按位与" 后的值。
bitlshift	监控项值的按位左移。
bitnot	监控项值的 "按位非" 后的值。
bitor	监控项值与掩码进行 "按位或" 后的值。

Function	Description
bitrshift	监控项值的按位右移。
bitxor	监控项值与掩码进行 “按位异或” 后的值。

函数详情

有关函数参数的一些一般说明：

- 函数参数以逗号分隔
- 表达式可作为参数
- 可选的函数参数（或参数部分）以 < > 表示

bitand(value,mask)

监控项值与掩码进行 “按位与” 运算后的值。
 支持的值类型：整数。

参数：

- **value** - 要检查的值；
- **mask**（必填）- 64 位无符号整数（0 - 18446744073709551615）。

尽管比较是以按位方式进行的，但所有值都必须以十进制提供并以十进制返回。例如，检查第 3 位时，应与 4 比较，而不是 100。

示例：

```
bitand(last(/host/key),12)=8 or bitand(last(/host/key),12)=4 #设置了第 3 位或第 4 位，但不会同时设置
bitand(last(/host/key),20)=16 #第 3 位未设置且第 5 位已设置
```

bitlshift(value,bits to shift)

监控项值的按位左移。
 支持的值类型：整数。

参数：

- **value** - 要检查的值；
- **bits to shift**（必填）- 要移位的位数。

虽然比较是以按位方式进行的，但所有值都必须以十进制提供并以十进制返回。例如，检查第 3 位时应比较 4，而不是 100。

bitnot(value)

监控项值的 “按位取反” 值。
 支持的值类型：整数。

参数：

- **value** - 要检查的值。

尽管比较是以按位方式进行的，但所有值都必须以十进制提供并以十进制返回。例如，检查第 3 位时，应比较 4，而不是 100。

bitor(value,mask)

监控项值与掩码进行 “按位或” 运算后的值。
 支持的值类型：整数。

参数：

- **value** - 要检查的值；
- **mask**（必填）- 64 位无符号整数（0 - 18446744073709551615）。

虽然比较是以按位方式进行的，但所有值都必须以十进制提供并以十进制返回。例如，检查第 3 位时，应比较 4，而不是 100。

bitrshift(value,bits to shift)

监控项值按位右移。
 支持的值类型：整数。

参数：

- **value** - 要检查的值；
- **bits to shift**（必填）- 要移位的位数。

虽然比较是以按位方式进行的，但所有值都必须以十进制提供，并且返回结果也是十进制。例如，检查第 3 位时，应比较 4，而不是 100。

bitxor(value,mask)

监控项值与掩码进行 “按位异或” 运算后的值。
 支持的值类型：整数。

参数：

- **value** - 要检查的值；

- **mask**（必填）- 64 位无符号整数（0 - 18446744073709551615）。

虽然比较是按位方式进行的，但所有值都必须以十进制提供，并且返回值也是十进制。例如，检查第 3 位时，应比较 4，而不是 100。

查看所有支持的函数。

日期和时间函数

此处列出的所有函数都支持用于：

- 触发器表达式
- 计算型监控项公式
- 表达式宏

Attention:
日期和时间函数不能单独用于表达式中；表达式中必须至少包含一个来自另一组的函数，并引用主机监控项（nodata() 函数除外）。有关日期和时间函数在表达式中的详细工作方式，请参见计算时间。

以下仅列出函数名称，不提供额外信息。单击函数可查看完整详情。

Function	Description
date	当前日期，格式为 YYYYMMDD。
dayofmonth	月份中的日期，范围为 1 到 31。
dayofweek	星期中的日期，范围为 1 到 7。
now	自 Unix 纪元（1970 年 1 月 1 日 00:00:00 UTC）以来的秒数。
time	当前时间，格式为 HHMMSS。

功能详情

date

当前日期，格式为 YYYYMMDD。

示例：

```
date()<20220101
```

dayofmonth

一个月中的第几天，范围为 1 到 31。

示例：

```
dayofmonth()=1
```

dayofweek

一周中的第几天，取值范围为 1 到 7（周一 - 1，周日 - 7）。

示例（仅工作日）：

```
dayofweek()<6
```

示例（仅周末）：

```
dayofweek()>5
```

now

自 Epoch（1970 年 1 月 1 日 00:00:00 UTC）以来的秒数。

示例：

```
now()<1640998800
```

time

当前时间，格式为 HHMMSS。

示例（仅夜间，00:00-06:00）：

```
time()<060000
```

请参阅所有受支持的函数。

历史函数

此处列出的所有函数均支持：

- 触发器表达式
- 计算监控项公式
- 表达式宏

函数列表不包含其他信息。单击函数可查看完整详细信息。

函数	描述
change	前一个值与最新值之间的差值。
changecount	在定义的评估周期内，相邻值之间发生变化的次数。
count	在定义的评估周期内的值数量。
countunique	在定义的评估周期内的唯一值数量。
find	在定义的评估周期内查找匹配的值。
first	在定义的评估周期内的第一个（最早的）值。
firstclock	在定义的评估周期内第一个（最早的）值的时间戳。
fuzzytime	检查被动 agent 的时间与 Zabbix 服务器/proxy 时间之间的差异。
jsonpath	返回 JSONPath 查询结果数组。
last	最新的值。
lastclock	在定义的评估周期内第 N 个最新值的时间戳。
logeventid	检查最后一条日志条目的事件 ID 是否与正则表达式匹配。
logseverity	最后一条日志条目的日志严重性。
logsource	检查最后一条日志条目的日志来源是否与正则表达式匹配。
logtimestamp	第 N 个最新日志监控项值的日志消息时间戳。
monodec	检查值是否一直单调递减。
monoinc	检查值是否一直单调递增。
nodata	检查是否未接收到数据。
percentile	某个周期的第 P 个百分位数，其中 P（百分比）由第三个参数指定。
rate	在定义的时间周期内，单调递增计数器增量的每秒平均速率。
xmlxpath	返回 XML XPath 查询结果数组。

常用参数

- `/host/key` 是引用主机监控项历史数据的函数的第一个常用必需参数。
- `(sec|#num)<:time shift>` 是引用主机监控项历史数据的函数的第二个常用参数，其中：
  - **sec** - 以秒为单位的最大评估周期（可以使用时间后缀），或
  - **#num** - 最新采集值中的最大评估范围（如果前面带有井号）。
  - **time shift**（可选）允许将评估时间点向过去移动。有关如何指定时间偏移的更多详细信息。

函数详情

有关函数参数的一些通用说明：

- 函数参数之间使用逗号分隔
- 可选函数参数（或参数部分）用 < > 表示
- 每个函数的特定参数会在对应函数中说明
- `/host/key` 和 `(sec|#num)<:time shift>` 参数绝不能加引号

change(/host/key)

前一个值与最新值之间的差值。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。
 对于字符串，返回值为：0 - 值相同；1 - 值不同。

参数：请参见通用参数。

说明：

- 将计算数值差值，如以下传入值示例所示（“前一个” 值和 “最新” 值 = 差值）：
’1’ 和’5’ = +4
’3’ 和’1’ = −2
’0’ 和’-2.5’ = −2.5。
- 在触发器中，可用于捕获突发峰值（或下降）、计数器重置、数值状态转换。
- 另请参见：abs 以进行比较。

示例：

```
change(/host/system.uptime)<0 #system uptime change has been negative since the last value (indicating a r
change(/host/system.cpu.load[all,avg1])>2 #CPU load (for one minute) has jumped by more than 2 since the l
change(/host/vfs.fs.size[/,free])<-1G #free disk space has dropped by more than 1 GB between checks
```

changecount(/host/key,(sec|#num)<:time shift>,<mode>)

在定义的评估周期内，相邻值之间发生变化的次数。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见通用参数
- **mode**（必须用双引号括起来）- 可选值：*all* - 统计所有变化（默认）；*dec* - 统计减少次数；*inc* - 统计增加次数

注释：

- 对于非数值类型，*mode* 参数将被忽略。
- 在触发器中，可用于捕获频繁的状态变化（表明系统不稳定）。

示例：

```
changecount(/host/icmpping,10m)>5 #10 分钟内 ping 状态变化超过 5 次
changecount(/host/vfs.file.contents["/sys/class/net/eth0/operstate"],1h)>5 #1 小时内 eth0 的运行状态变化超过 5
changecount(/host/proc.num[httpd],15m)>10 #15 分钟内 httpd 进程数量变化超过 10 次
changecount(/host/key,#10,"inc") # 最近 10 个值中，值增加（相对于相邻值）的次数
changecount(/host/key,24h,"dec") # 截至当前，最近 24 小时内值减少（相对于相邻值）的次数
```

count(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

在定义的评估周期内的值数量。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见common parameters
- **operator**（必须用双引号括起来）支持的 operators：
eq - 等于（Integer、Float 的默认值）
ne - 不等于
gt - 大于
ge - 大于或等于
lt - 小于
le - 小于或等于
like（String、Text、Log 的默认值）- 如果包含模式则匹配（区分大小写）
bitand - 按位 AND
regexp - 匹配 pattern 中给出的正则表达式，区分大小写
iregexp - 匹配 pattern 中给出的正则表达式，不区分大小写
- **pattern** - 所需的模式（字符串参数必须用双引号括起来）

注释：

- Float 监控项的匹配精度为 2.22e-16。
- *like* 不支持作为 Integer 值的 operator。
- *like* 和 *bitand* 不支持作为 Float 值的 operator。
- 对于 String、Text 和 Log 值，仅支持 *eq*、*ne*、*like*、*regexp* 和 *iregexp* operators。
- 当 *bitand* 作为 operator 时，第四个 pattern 参数可以指定为两个数字，并用'/' 分隔：**number_to_compare_with/mask**。count() 会对值和 *mask* 进行 "按位 AND" 运算，并将结果与 *number_to_compare_with* 比较。如果 "按位 AND" 的结果等于 *number_to_compare_with*，则该值会被计数。
 如果 *number_to_compare_with* 和 *mask* 相等，则只需指定 *mask*（无需'/'）。
- 当 *regexp* 或 *iregexp* 作为 operator 时，第四个 pattern 参数可以是普通正则表达式，也可以是global 正则表达式（以'@' 开头）。对于全局正则表达式，大小写敏感性继承自全局正则表达式设置。为了进行 regexp 匹配，Float 值始终会以小数点后 4 位表示。另请注意，对于较大的数字，十进制（存储在数据库中）和二进制（由 Zabbix 服务器使用）表示之间的差异可能会影响第 4 位小数。

示例：

```
count(/host/icmpping,30m,,"0")>5 #ping 失败超过 5 次，持续 30 分钟
count(/host/key,10m,"like","error") # 最近 10 分钟内直到现在，包含 'error' 的值数量
count(/host/key,10m,,12) # 最近 10 分钟内直到现在，等于 '12' 的值数量
count(/host/key,10m,"gt",12) # 最近 10 分钟内直到现在，大于 '12' 的值数量
count(/host/key,#10,"gt",12) # 最近 10 个值内直到现在，大于 '12' 的值数量
count(/host/key,10m:now-1d,"gt",12) # 从现在往前 24 小时到 24 小时 10 分钟之前之间，大于 '12' 的值数量
count(/host/key,10m,"bitand","6/7") # 最近 10 分钟内直到现在，最低 3 位中包含 '110'（二进制）的值数量
count(/host/key,10m:now-1d) # 从现在往前 24 小时到 24 小时 10 分钟之前之间的值数量
```

countunique(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

在定义的评估周期内唯一值的数量。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见common parameters

- **operator**（必须用双引号括起来）。支持的 operators：
eq - 相等（整数、浮点数默认）
ne - 不等于
gt - 大于
ge - 大于或等于
lt - 小于
le - 小于或等于
like（字符串、文本、日志默认）- 如果包含模式则匹配（区分大小写）
bitand - 按位 AND
regexp - 匹配 pattern 中给出的正则表达式，区分大小写
iregexp - 匹配 pattern 中给出的正则表达式，不区分大小写
- **pattern** - 所需的模式（字符串参数必须用双引号括起来）

注释：

- 浮点监控项按 2.22e-16 的精度进行匹配。
- *like* 不支持作为整数值的 operator。
- *like* 和 *bitand* 不支持作为浮点值的 operator。
- 对于字符串、文本和日志值，仅支持 *eq*、*ne*、*like*、*regexp* 和 *iregexp* operators。
- 当 *bitand* 作为 operator 时，第四个 pattern 参数可以指定为两个数字，并用'/' 分隔：**number_to_compare_with/mask**。countunique() 会对值和 *mask* 进行 "按位 AND" 运算，并将结果与 *number_to_compare_with* 比较。如果 "按位 AND" 的结果等于 *number_to_compare_with*，则该值会被计数。
 如果 *number_to_compare_with* 和 *mask* 相等，则只需指定 *mask*（无需'/'）。
- 当 *regexp* 或 *iregexp* 作为 operator 时，第四个 pattern 参数可以是普通正则表达式，也可以是global 正则表达式（以'@' 开头）。对于全局正则表达式，大小写敏感性继承自全局正则表达式设置。为了进行 regexp 匹配，浮点值将始终以小数点后 4 位表示。另请注意，对于大数值，十进制（存储在数据库中）和二进制（由 Zabbix 服务器使用）表示之间的差异可能会影响到第 4 位小数。

示例：

```
countunique(/host/key,10m) # 过去 10 分钟内直到现在的唯一值数量
countunique(/host/key,10m,"like","error") # 过去 10 分钟内直到现在包含 'error' 的唯一值数量
countunique(/host/key,10m,,12) # 过去 10 分钟内直到现在等于 '12' 的唯一值数量
countunique(/host/key,10m,"gt",12) # 过去 10 分钟内直到现在大于 '12' 的唯一值数量
countunique(/host/key,#10,"gt",12) # 最近 10 个值中直到现在大于 '12' 的唯一值数量
countunique(/host/key,10m:now-1d,"gt",12) # 从距现在 24 小时到距现在 24 小时 10 分钟之间大于 '12' 的唯一值数量
countunique(/host/key,10m,"bitand","6/7") # 过去 10 分钟内直到现在在最低 3 位中包含 '110'（二进制）的唯一值数量
countunique(/host/key,10m:now-1d) # 从距现在 24 小时到距现在 24 小时 10 分钟之间的唯一值数量
```

find(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

在定义的评估周期内查找匹配的值。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。
 返回：1 - 找到；0 - 其他情况。

参数：

- 参见common parameters
- **sec** 或 **#num**（可选）- 如果未指定，则默认为最新值
- **operator**（必须用双引号括起来）支持的 operators：
eq - 等于（整数、浮点数默认）
ne - 不等于
gt - 大于
ge - 大于或等于
lt - 小于
le - 小于或等于
like（字符串、文本、日志默认）- 当 pattern 中给定的字符串被包含时匹配（区分大小写）
bitand - 按位 AND
regexp - 匹配 pattern 中给定的正则表达式（区分大小写）
iregexp - 匹配 pattern 中给定的正则表达式（不区分大小写）
- **pattern** - 所需的模式（字符串参数必须用双引号括起来）；如果 operator 为 *regexp*、*iregexp*，则为 Perl Compatible Regular Expression（PCRE）正则表达式

说明：

- 如果处理了多个值，只要至少有一个值匹配，就返回'1'。
- *like* 不支持作为整数值的 operator。
- *like* 和 *bitand* 不支持作为浮点值的 operator。
- 对于字符串、文本和日志值，仅支持 *eq*、*ne*、*like*、*regexp* 和 *iregexp* operators。
- 当 operator 为 *regexp* 或 *iregexp* 时，第四个 pattern 参数可以是普通正则表达式，也可以是global 正则表达式（以'@' 开头）。对于全局正则表达式，大小写敏感性继承自全局正则表达式设置。

示例：

```
find(/host/key,10m,"like","error") # 查找最近 10 分钟内直到现在包含 'error' 的值
find(/host/agent.version,,"like","beta")=1 #Zabbix agent 具有 beta 版本，必须升级
find(/host/log[/var/log/nginx/access.log],,"regexp"," 500 ")=1 # 已发现内部 web 服务器错误
```

first(/host/key,sec<:time shift>)

定义的评估周期内的第一个（最旧的）值。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见通用参数

另请参见last()。

示例：

```
first(/host/key,1h) #retrieve the oldest value within the last hour until now
```

firstclock(/host/key,sec<:time shift>)

定义的评估周期内最早值的时间戳。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见通用参数。

如果在给定周期内未收集到任何数据，该函数将因错误而失败。

另请参见lastclock()。

示例：

```
firstclock(/host/key,1h) # 获取最近一小时内最早值的时间戳
firstclock(/host/key,1h:now-24h) # 获取一天前的最近一小时内最早值的时间戳
```

fuzzytime(/host/key,sec)

检查被动 agent 时间与 Zabbix 服务器/proxy 时间相差多少。
 支持的值类型：*Float*、*Integer*。
 返回值：1 - 被动监控项值（作为时间戳）与 Zabbix 服务器/proxy 时间戳（采集值时的时钟）之间的差值小于或等于 *sec* 秒；0 - 否则。

参数：

- 参见common parameters。

注释：

- 通常与'system.localtime' 监控项一起使用，用于检查本地时间是否与 Zabbix 服务器的本地时间同步。注意，对于 Zabbix agent，'system.localtime' 必须配置为passive check；在 Zabbix agent 2 上，它也可以配置为主动检查。
- 也可与 `vfs.file.time[/path/file,modify]` 键一起使用，以检查文件是否长时间未更新。
- 不建议在复杂的触发器表达式中使用此函数（涉及多个监控项），因为它可能导致意外结果（时间差将使用最新的指标进行测量），例如：`fuzzytime(/Host/system.localtime,60s)=0 or last(/Host/trap)<>0`。

示例：

```
fuzzytime(/host/system.localtime,5m)=0 # 客户端本地时间与 Zabbix 服务器/proxy 时间相差超过 5 分钟
fuzzytime(/host/system.localtime,5m)=0 and nodata(/host/system.localtime,10m)=0 # 客户端本地时间与 Zabbix 服务
```

jsonpath(/host/key,(sec|#num)<:time shift>,pattern)

返回 JSONPath 查询结果数组。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- 请参阅通用参数
- **pattern** - 必需的模式（字符串参数必须用双引号括起来）

注释：

- 此函数与聚合函数结合使用。
- 对每个选定的历史值执行 JSONPath 查询；每个非空的成功查询结果都会添加到数组中。如果某个历史值的查询结果包含多个匹配项，则会将每个匹配项分别添加到数组中。
- 在以下情况下，函数会返回错误：
  - `pattern` 格式无效。
  - 任一历史值包含无效 JSON，或该值的 JSONPath 查询失败。
- 在以下情况下，函数会返回空数组：
  - 未选择任何历史值。
  - 所有选定的历史值都包含无效 JSON。

示例：

```
max(jsonpath(/host/http.response,5m,"$.response_time")) > 500
```

last(/host/key,<#num<:time shift»)

最新的值。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见通用参数
- **#num**（可选）- 倒数第 N 个最新值

说明：

- 请注意，带井号的时间周期（#N）在这里与许多其他函数中的用法不同。
  例如：`last(/host/key)` 始终等同于 `last(/host/key,#1)`；`last(/host/key,#3)` 表示倒数第三个最新值（而不是最近三个值）。
- 如果在历史记录中同一秒内存在两个以上的值，Zabbix 不保证这些值的精确顺序。
- 另请参见first()。

示例：

```
last(/host/key) #retrieve the last value
last(/host/key,#2) #retrieve the previous value
last(/host/key,#1)<>last(/host/key,#2) #last two values differ
```

lastclock(/host/key,<#num<:time shift»)

定义的评估周期内第 N 个最近值的时间戳。
 支持的值类型：*Float*、*Integer*、*String*、*Text*、*Log*。

参数：

- 参见通用参数；
- **#num**（可选）- 第 N 个最近值。

如果在给定周期内未收集到任何数据，或者未收集到第 N 个值，则该函数将失败并报错。

另请参见firstclock()。

示例：

```
lastclock(/host/key) # 获取最新值的时间戳
lastclock(/host/key,#2) # 获取倒数第 2 个最新值的时间戳
now()-lastclock(/host/heartbeat.item)>300 # 自心跳监控项更新以来已超过 5 分钟
lastclock(/host/system.cpu.load[all,avg1])-lastclock(/host/system.cpu.load[all,avg1],#2)>300 # 最近两个值之间
((now()-lastclock(/host/system.cpu.load[all,avg1])<120 and last(/host/system.cpu.load[all,avg1])>5)) #CPU
```

logeventid(/host/key,<#num<:time shift»,<pattern>)

检查最后一条日志记录的事件 ID 是否与正则表达式匹配。
 支持的值类型：*Log*。
 返回：0 - 不匹配；1 - 匹配。

参数：

- 参见common parameters
- **#num**（可选）- 第 N 个最近的值
- **pattern**（可选）- 描述所需模式的正则表达式，采用 Perl Compatible Regular Expression（PCRE）风格（字符串参数必须用双引号括起来）

示例：

```
logeventid(/host/eventlog[Security],,"^4625$")=1 # 找到一条 ID 匹配 "4625"（认证失败）的日志记录
logeventid(/host/eventlog[System],,"^(6008|41)$")=1 # 找到一条 ID 匹配 "6008" 或 "41" 的日志记录
```

logseverity(/host/key,<#num<:time shift»)

最后一条日志条目的日志严重性。
 支持的值类型：*Log*。
 返回值：0 - 默认严重性；N - 严重性（整数，对 Windows 事件日志很有用：1 - 信息，2 - 警告，4 - 错误，7 - 失败审核，8 - 成功审核，9 - 严重，10 - 详细）。

参数：

- 参见通用参数
- **#num**（可选）- 倒数第 N 个最新值

Zabbix 从 Windows 事件日志的 **Information** 字段获取日志严重性。

示例：

```
logseverity(/host/log[/var/log/syslog],10m)>3 #a log entry with severity above "3" found
logseverity(/host/eventlog[System],10m)=4 #a log entry with severity equaling "Error" found
```

logsource(/host/key,<#num<:time shift»,<pattern>)

检查最后一条日志记录的日志源是否与正则表达式匹配。
 支持的值类型：*Log*。
 返回：0 - 不匹配；1 - 匹配。

参数：

- 参见common parameters
- **#num**（可选）- 第 N 个最近的值

- **pattern**（可选）- 描述所需模式的正则表达式，采用 Perl Compatible Regular Expression（PCRE）风格（字符串参数必须用双引号括起来）

通常用于 Windows 事件日志。

示例：

```
logsource(/host/eventlog[Application],,"MSSQLSERVER")=1 # 找到源匹配 "MSSQLSERVER" 的日志记录
logsource(/host/eventlog[System],,"Service Control Manager")=1 # 找到源匹配 "Service Control Manager" 的日志记
logsource(/host/eventlog[System],,"Service Control Manager")=1 and logeventid(/host/eventlog[System],,"^70
```

logtimestamp(/host/key,<#num<:time shift»)

第 N 个最近的日志监控项值中的日志消息时间戳。
 支持的值类型：*Log*。

参数：

- 参见通用参数；
- **#num**（可选）- 第 N 个最近的值。

说明：

- 时间偏移的计算基于监控项值的时钟，而不是日志消息时间戳；
- 在以下情况下，该函数会返回错误：
  - 接收到的不是日志类型的监控项；
  - 在给定时间段内未收集到数据；
  - 未收集到第 N 个值；
  - 日志消息不包含时间戳值。

示例：

```
logtimestamp(/host/key) # 获取最新日志消息的时间戳
logtimestamp(/host/key,#2) # 获取倒数第 2 条日志消息的时间戳
logtimestamp(/host/key,#3:now-1d) # 获取一天前倒数第 3 条日志消息的时间戳
```

monodec(/host/key,(sec|#num)<:time shift>,<mode>)

检查值是否持续单调减少。
 支持的值类型：整数。
 返回值：1 - 如果时间段内所有元素都持续减少；0 - 否则。

参数：

- 参见通用参数
- **mode**（必须用双引号括起来）- *weak*（每个值都小于或等于前一个值；默认）或 *strict*（每个值都已减少）

示例：

```
monodec(/host/system.swap.size[all,free],60s) + monodec(/host2/system.swap.size[all,free],60s) + monodec(/
monodec(/host/proc.num[nginx],10m,"strict")=1 #nginx 进程数在 10 分钟内持续单调减少
```

monoinc(/host/key,(sec|#num)<:time shift>,<mode>)

检查值是否存在单调递增。
 支持的值类型：*Integer*。
 返回值：1 - 如果时间段内的所有元素都持续递增；0 - 否则。

参数：

- 参见通用参数
- **mode**（必须用双引号括起来）- *weak*（每个值都大于或等于前一个值；默认）或 *strict*（每个值都已增大）

示例：

```
monoinc(/host/system.localtime,#3,"strict")=0 # 系统本地时间并未持续递增
monoinc(/host/vfs.dir.count[/mnt/data/logs],24h,"weak")=0 # 如果 24 小时内文件数量停滞不变（预期应增长）则触发器
```

nodata(/host/key,sec,<mode>)

检查是否未接收到数据。
 支持的值类型：*Integer*、*Float*、*Character*、*Text*、*Log*。
 返回值：1 - 如果在定义的时间段内未接收到数据；0 - 否则。

参数：

- 请参阅通用参数
- **sec** - 时间段不应少于 30 秒，因为历史记录同步器进程每 30 秒才计算一次此函数；不允许使用 `nodata(/host/key,0)`
- **mode** - 如果设置为 *strict*（使用双引号），则此函数将不受 proxy 可用性的影响（详情请参阅注释）

注释：

- 默认情况下，由 proxy 监控的 “nodata” 触发器会受 proxy 可用性的影响 - 如果 proxy 变为不可用，则连接恢复后 “nodata” 触发器不会立即触发，而是会跳过延迟期间的数据。请注意，对于被动 proxy，如果连接在超过 15 秒且不少于 2 秒后恢复，则会启用抑制。对于主动 proxy，如果连接在超过 15 秒后恢复，则会启用抑制。要关闭对 proxy 可用性的敏感性，请使用第三个参数，例如：`nodata(/host/key,5m,"strict")`；在这种情况下，只要经过了无数据的评估时间段（五分钟），函数就会立即触发。
- 如果在第 1 个参数指定的时间段内：
- 没有数据且 Zabbix 服务器已重启
- 没有数据且维护已完成
- 没有数据且监控项已添加或重新启用
 此函数将显示错误。
- 错误会显示在触发器列表的信息列中。
- 如果 Zabbix 服务器、proxy 和 agent 之间存在时间差异，此函数可能无法正常工作。另请参阅：时间同步要求。

示例：

```
nodata(/host/agent.ping,5m)=1 # 如果 5 分钟内未从 Zabbix agent 接收到数据，则触发
```

percentile(/host/key,(sec|#num)<:time shift>,percentage)

某一时间段的第 P 百分位数，其中 P（percentage）由第三个参数指定。
 支持的值类型：*Float*、*Integer*。

参数：

- 参见通用参数
- **percentage** - 介于 0 和 100 之间（含）的浮点数，小数点后最多 4 位

示例：

```
percentile(/host/net.if.in[eth0,bytes],1h,95)>1000000 #1 小时内网络入站流量（字节/秒）的第 95 百分位数已超过阈值（例
percentile(/host/system.cpu.util,5m,95)>80 #CPU 利用率用户态百分比的第 95 百分位数已超过 80
percentile(/host/icmppingsec[192.168.0.2],15m,95)>0.15 # 大多数延迟测量值低于 150 ms，但较高端的尾部（最高 5%）表明
percentile(/host/net.if.in[eth0,bytes],1h,50) # 计算 1 小时内入站网络流量的第 50 百分位数（中位数）；这与 avg()（平均
(percentile(/host/net.if.in[eth0,bytes],1h,50)+percentile(/host/net.if.in[eth0,bytes],1h,51))/2 # 在 1 小时
```

rate(/host/key,sec<:time shift>)

在定义的时间段内，单调递增计数器的平均每秒增量。
 支持的值类型：*Float*、*Integer*。

参数：

- 参见通用参数

在功能上对应于 PromQL 的’rate’。

示例：

```
rate(/host/key,30s) # 如果 30 秒内的单调增量为 20，则此函数返回 0.67。
rate(/host/net.if.in[eth0,bytes],5m)>500000 # 过去 5 分钟内 eth0 上的入站接口流量速率已超过 500 KB/s
rate(/host/app.requests.count,1m)>100 # 最近 1 分钟内，请求计数器已上升到每秒超过 100 个请求
```

xmlxpath(/host/key,(sec|#num)<:time shift>,pattern)

返回 XML XPath 查询结果数组。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- 请参阅通用参数
- **pattern** - 必需的模式（字符串参数必须用双引号括起来）

注释：

- 此函数与聚合函数结合使用。
- 对每个选定的历史值执行 XML XPath 查询；每个非空的成功查询结果都会添加到数组中。如果单个历史值的查询结果包含多个匹配项，则每个匹配项都会单独添加到数组中。
- 在以下情况下，函数会返回错误：
  - `pattern` 格式无效。
  - Zabbix 在编译时未启用 XML 支持。
  - 任一历史值包含无效 XML，或该值的 XPath 查询失败。
- 在以下情况下，函数会返回空数组：
  - 未选择任何历史值。
  - 所有选定的历史值都包含无效 XML。

示例：

```
max(xmlxpath(/host/http.response,5m,"/response/response_time")) > 500
```

请参阅所有支持的函数。

趋势函数

与历史函数不同，趋势函数使用趋势数据进行计算。

趋势会存储按小时汇总的值。趋势函数使用这些小时平均值，因此适用于长期分析。

趋势函数结果会被缓存，因此对同一函数、相同参数的多次调用，只会从数据库获取一次信息。趋势函数缓存由TrendFunctionCacheSize服务器参数控制。

使用zabbix[tcache,cache,paccessed]来估算缓存的实际使用情况。较低的 paccessed，同时伴随较低的 pitems 和 pmisses，通常意味着可以减小缓存大小。

仅引用趋势函数的触发器会按照表达式中最小的时间周期每次评估一次。例如，像下面这样的触发器：

```
trendavg(/host/key,1d:now/d) > 1 or trendavg(/host/key2,1w:now/w) > 2
```

将按天评估一次。如果触发器同时包含趋势和历史（或日期和时间和/或nodata()）函数，则会按照通常原则进行计算。

此处列出的所有函数都支持用于：

- 触发器表达式
- 计算监控项
- 表达式宏

以下仅列出函数名称，不提供额外信息。点击函数可查看完整详情。

Function	Description
baselinedev	返回最后一个数据周期与前几个季节中相同数据周期之间的偏差数量（通过 stddevpop 算法）。
baselinewma	使用加权移动平均算法，通过对多个相等时间周期（“季节”）中相同时间范围的数据求平均来计算基线。
trendavg	定义时间段内趋势值的平均值。
trendcount	在定义时间段内，用于计算趋势值的已成功检索到的历史值数量。
trendmax	定义时间段内趋势值的最大值。
trendmin	定义时间段内趋势值的最小值。
trendstl	返回检测期间的异常率 - 一个介于 0 和 1 之间的小数值，即 ((异常值数量)/(总值数量))。
trendsum	定义时间段内趋势值的总和。

通用参数

- /host/key 是常见的必填第一参数
- time period:time shift 是常见的第二参数，其中：
  - **time period** - 时间周期（最小为 “1h”），定义为 <N><time unit>，其中 N - 时间单位的数量，time unit - h（小时）、d（天）、w（周）、M（月）或 y（年）。
  - **time shift** - 时间周期偏移（请参见函数示例）

函数详情

有关函数参数的一些通用说明：

- 函数参数以逗号分隔
- 可选函数参数（或参数部分）以 < > 表示
- 每个函数的特定参数会在对应函数中说明
- /host/key 和 time period:time shift 参数绝不能加引号

baselinedev(/host/key,data period:time shift,season unit,num seasons)

返回最后一个数据周期与前几个季节中相同数据周期之间的偏差次数（使用 stddevpop 算法）。

参数：

- 参见通用参数
- **data period** - 一个季节内的数据收集周期，定义为 <N><time unit>，其中：
N - 时间单位的数量
time unit - h（小时）、d（天）、w（周）、M（月）或 y（年），必须等于或小于 season
- **season unit** - 定义一个季节的日历单位（h、d、w、M、y）；不能小于 data period
- **num seasons** - 要评估的季节数量

示例：

```
baselinedev(/host/key,1d:now/d,"M",6) # 计算前一天与前 6 个月中同一天之间的标准差（总体）次数。如果某个月中不存在该日期，
baselinedev(/host/key,1h:now/h,"d",10)  # 计算前一小时与之前 10 天中每天相同时段之间的总体标准差次数
```

baselinewma(/host/key,data period:time shift,season unit,num seasons)

使用加权移动平均算法，通过对多个相等时间周期（“季”）中相同时间范围的数据取平均值来计算基线。

参数：

- 参见通用参数
- **data period** - 一个季内的数据收集周期，定义为 <N><time unit>，其中：
N - 时间单位的数量
time unit - h（小时）、d（天）、w（周）、M（月）或 y（年），必须等于或小于 season
Time shift - 时间周期偏移量，以季为单位定义数据收集时间范围的结束点（见示例）
- **season unit** - 定义一个季的日历单位（h、d、w、M、y）；不能小于 data period
- **num seasons** - 要评估的季数

示例：

```
baselinewma(/host/key,1h:now/h,"d",3) # 计算截至昨天的最近 3 个完整天中相同小时的基线。如果“now”是星期一 13:30，则将分析
baselinewma(/host/key,2h:now/h,"d",3) # 计算截至昨天的最近 3 个完整天中相同两小时时间范围的基线。如果“now”是星期一 13:30
baselinewma(/host/key,1d:now/d,"M",4) # 计算以上一个完整月之前的 4 个月中与“昨天”处于同一月内日期的基线。如果所需日期不存在
```

trendavg(/host/key,time period:time shift)

定义时间段内趋势值的平均值。

参数：

- 参见通用参数

示例：

```
trendavg(/host/key,1h:now/h) # 前一小时的平均值（例如 12:00-13:00）
trendavg(/host/key,1h:now/h-1h) # 两小时前的平均值（11:00-12:00）
trendavg(/host/key,1h:now/h-2h) # 三小时前的平均值（10:00-11:00）
trendavg(/host/key,1M:now/M-1y) # 一年前上个月的平均值
```

trendcount(/host/key,time period:time shift)

在定义的时间段内，用于计算趋势值的成功检索到的历史值数量。

参数：

- 参见通用参数

示例：

```
trendcount(/host/key,1h:now/h) # 前一小时的值数量（例如 12:00-13:00）
trendcount(/host/key,1h:now/h-1h) # 两小时前一小时的值数量（11:00-12:00）
trendcount(/host/key,1h:now/h-2h) # 三小时前一小时的值数量（10:00-11:00）
trendcount(/host/key,1M:now/M-1y) # 一年前上个月的值数量
```

trendmax(/host/key,time period:time shift)

定义时间段内趋势值的最大值。

参数：

- 参见通用参数

示例：

```
trendmax(/host/key,1h:now/h) # 前一小时的最大值（例如 12:00-13:00）
trendmax(/host/key,1h:now/h) - trendmin(/host/key,1h:now/h) → 计算前一小时（12:00-13:00）的最大值与最小值之差（趋
trendmax(/host/key,1h:now/h-1h) # 两小时前（11:00-12:00）的最大值
trendmax(/host/key,1h:now/h-2h) # 三小时前（10:00-11:00）的最大值
trendmax(/host/key,1M:now/M-1y) # 一年前上个月的最大值
```

trendmin(/host/key,time period:time shift)

定义时间段内趋势值中的最小值。

参数：

- 参见通用参数

示例：

```
trendmin(/host/key,1h:now/h) # 前一小时的最小值（例如 12:00-13:00）
trendmax(/host/key,1h:now/h) - trendmin(/host/key,1h:now/h) → 计算前一小时（12:00-13:00）的最大值与最小值之差（趋
trendmin(/host/key,1h:now/h-1h) # 两小时前的最小值（11:00-12:00）
trendmin(/host/key,1h:now/h-2h) # 三小时前的最小值（10:00-11:00）
trendmin(/host/key,1M:now/M-1y) # 一年前上个月的最小值
```

trendstl(/host/key,eval period:time shift,detection period,season,<deviations>,<devalg>,<s window>)

返回检测周期内的异常率——一个介于 0 和 1 之间的小数值，计算方式为（(异常值数量)/(总值数量)）。

参数：

- 参见通用参数
- **eval period** - 必须进行分解的时间周期（最小为 1h），定义为 <N><time unit>，其中
N - 时间单位的数量
time unit - h（小时）、d（天）、w（周）、M（月）或 y（年）
- **detection period** - eval period 结束前用于计算异常的时间周期（最小为 1h，不能长于 eval period），定义为 <N><time unit>，其中
N - 时间单位的数量
time unit - h（小时）、d（天）、w（周）
- **season** - 预期会出现重复模式（“季节性”）的最短时间周期（最小为 2h，不能长于 eval period，且 eval period 中的条目数必须大于结果频率（season/h）的两倍），定义为 <N><time unit>，其中
N - 时间单位的数量
time unit - h（小时）、d（天）、w（周）
- **deviations** - 计为异常的偏差数（由 devalg 计算，可为小数）（必须大于或等于 1，默认值为 3）
- **devalg**（必须用双引号括起来）- 偏差算法，可以是 *stddevpop*、*stddevsamp* 或 *mad*（默认）
- **s window** - 用于提取季节性成分的 loess 窗口跨度（以滞后数计）（默认值为 eval period 中条目数 * 10 + 1）

示例：

```
trendstl(/host/key,100h:now/h,10h,2h) # 分析最近 100 小时的趋势数据，找出该周期最后 10 小时的异常率，预期周期性为 2h，
trendstl(/host/key,100h:now/h-10h,100h,2h,2.1,"mad") # 分析截至 10 小时前的 100 小时趋势数据，找出整个周期的异常率，
trendstl(/host/key,100d:now/d-1d,10d,1d,4,,10) # 分析截至 1 天前的 100 天趋势数据，找出该周期最后 10d 的异常率，预期
trendstl(/host/key,1M:now/M-1y,1d,2h,,"stddevsamp") # 分析一年前的上个月，找出该周期最后一天的异常率，预期周期性为 2h
```

trendsum(/host/key,time period:time shift)

定义时间段内趋势值的总和。

参数：

- 参见通用参数

示例：

```
trendsum(/host/key,1h:now/h) # 前一小时的总和（例如 12:00-13:00）
trendsum(/host/key,1h:now/h-1h) # 两小时前一小时的总和（11:00-12:00）
trendsum(/host/key,1h:now/h-2h) # 三小时前一小时的总和（10:00-11:00）
trendsum(/host/key,1M:now/M-1y) # 一年前上个月的总和
```

参见所有支持的函数。

数学函数

此处列出的所有函数都支持以下场景：

- 触发器表达式
- 计算型监控项公式
- 表达式宏

除非另有说明，数学函数支持 float 和 integer 值类型。

以下函数仅列出名称，不包含附加信息。单击函数可查看完整详情。

Function	Description
abs	值的绝对值。
acos	以弧度表示的值的反余弦。
asin	以弧度表示的值的反正弦。
atan	以弧度表示的值的反正切。
atan2	以弧度表示的由纵坐标（值）和横坐标指定的角度的反正切。
avg	所引用监控项值的平均值。
cbrt	值的立方根。

Function	Description
ceil	将值向上舍入到最接近且不小于它的整数。
cos	值的余弦，其中该值是以弧度表示的角度。
cosh	值的双曲余弦。
cot	值的余切，其中该值是以弧度表示的角度。
degrees	将值从弧度转换为度。
e	欧拉数（2.718281828459045）。
exp	以欧拉数为底的幂。
expm1	以欧拉数为底的幂减 1。
floor	将值向下舍入到最接近且不大于它的整数。
log	自然对数。
log10	十进制对数。
max	所引用监控项值中的最大值。
min	所引用监控项值中的最小值。
mod	除法余数。
pi	圆周率常量（3.14159265358979）。
power	值的幂。
radians	将值从度转换为弧度。
rand	返回随机整数值。
round	将值舍入到小数位。
signum	如果值为负数则返回'-1'，如果值为零则返回'0'，如果值为正数则返回'1'。
sin	值的正弦，其中该值是以弧度表示的角度。
sinh	值的双曲正弦，其中该值是以弧度表示的角度。
sqrt	值的平方根。
sum	所引用监控项值的总和。
tan	值的正切。
truncate	将值截断到小数位。

函数详情

有关函数参数的一些一般说明：

- 函数参数以逗号分隔
- 表达式可作为参数
- 可选的函数参数（或参数部分）以 < > 表示

abs(value)

一个值的绝对值（从 0 开始计算）。

参数：

- **value** - 要检查的值

例如，'3' 或'-3' 的绝对值都将是'3'。

示例：

```
abs(last(/host/key))>10
```

acos(value)

一个值的反余弦，以弧度表示的角度。

参数：

- **value** - 要检查的值

该值必须介于 -1 和 1 之间。例如，值 "0.5" 的反余弦将为 "2.0943951"。

示例：

```
acos(last(/host/key))
```

asin(value)

一个值的反正弦，以弧度表示的角度。

参数：

- **value** - 要检查的值

该值必须介于 -1 和 1 之间。例如，值 “0.5” 的反正弦将为 “-0.523598776”。

示例：

```
asin(last(/host/key))
```

atan(value)

一个值的反正切，以弧度表示的角度。

参数：

- **value** - 要检查的值

例如，值 “1” 的反正切将为 “0.785398163”。

示例：

```
atan(last(/host/key))
```

atan2(value,abscissa)

纵坐标（value）和横坐标坐标指定的反正切角，以弧度表示。

参数：

- **value** - 要检查的值；
- **abscissa** - 横坐标值。

例如，值 “1” 的纵坐标和横坐标坐标的反正切将为 “2.21429744”。

示例：

```
atan2(last(/host/key),2)
```

avg(<value1>,<value2>,...)

被引用监控项值的平均值。

参数：

- **valueX** - 由另一个处理监控项历史记录的函数返回的值。

示例：

```
avg(avg(/host/key,1h),avg(/host2/key2,1h))
```

cbrt(value)

一个值的立方根。

参数：

- **value** - 要检查的值

例如，’64’ 的立方根是’4’，’63’ 的立方根是’3.97905721’。

示例：

```
cbrt(last(/host/key))
```

ceil(value)

将该值向上舍入到最接近且大于或等于它的整数。

参数：

- **value** - 要检查的值

例如，’2.4’ 将向上舍入为’3’。另请参见floor()。

示例：

```
ceil(last(/host/key))
```

cos(value)

一个值的余弦，其中该值是以弧度表示的角度。

参数：

- **value** - 要检查的值

例如，值 “1” 的余弦为 “0.54030230586”。

示例：

```
cos(last(/host/key))
```

cosh(value)

一个值的双曲余弦。返回值为实数，而不是科学计数法表示。

参数：

- **value** - 要检查的值

例如，值 “1” 的双曲余弦将为 “1.54308063482”。

示例：

```
cosh(last(/host/key))
```

cot(value)

一个值的余切，其中该值是以弧度表示的角度。

参数：

- **value** - 要检查的值

例如，值 “1” 的余切将为 “0.54030230586”。

示例：

```
cot(last(/host/key))
```

degrees(value)

将一个值从弧度转换为度。

参数：

- **value** - 要检查的值

例如，值 “1” 转换为度后将为 “57.2957795”。

示例：

```
degrees(last(/host/key))
```

e

欧拉数（2.718281828459045）。

示例：

```
e()
```

exp(value)

欧拉数的 value 次幂。

参数：

- **value** - 要检查的值

例如，value 为 “2” 时，欧拉数的 2 次幂为 “7.38905609893065”。

示例：

```
exp(last(/host/key))
```

expm1(value)

欧拉数的 value 次幂减 1。

参数：

- **value** - 要检查的值

例如，欧拉数的值’2’ 次幂减 1 的结果为’6.38905609893065’。

示例：

```
expm1(last(/host/key))
```

floor(value)

将该值向下舍入为最接近且小于或等于它的整数。

参数：

- **value** - 要检查的值

例如，'2.6' 将向下舍入为'2'。另请参见ceil()。

示例：

```
floor(last(/host/key))
```

log(value)

自然对数。

参数：

- **value** - 要检查的值

例如，值 "2" 的自然对数为 "0.69314718055994529"。

示例：

```
log(last(/host/key))
```

log10(value)

十进制对数。

参数：

- **value** - 要检查的值

例如，值 "5" 的十进制对数为 "0.69897000433"。

示例：

```
log10(last(/host/key))
```

max(<value1>,<value2>,...)

引用的监控项值中的最大值。

参数：

- **valueX** - 由另一个处理监控项历史记录的函数返回的值。

示例：

```
max(avg(/host/key,1h),avg(/host2/key2,1h))
```

min(<value1>,<value2>,...)

引用的监控项值中的最小值。

参数：

- **valueX** - 由另一个处理监控项历史数据的函数返回的值。

示例：

```
min(avg(/host/key,1h),avg(/host2/key2,1h))
```

mod(value,denominator)

除法余数。

参数：

- **value** - 要检查的值；
- **denominator** - 除数。

例如，值 "5" 除以 "2" 的余数为 "1"。

示例：

```
mod(last(/host/key),2)
```

pi

Pi 常量（3.14159265358979）。

示例：

```
pi()
```

power(value,power value)

值的幂。

参数：

- **value** - 要检查的值；
- **power value** - 要使用的 N 次幂。

例如，值 “2” 的 3 次幂将是 “8”。

示例：

```
power(last(/host/key),3)
```

radians(value)

将一个值从度转换为弧度。

参数：

- **value** - 要检查的值

例如，值 “1” 转换为弧度后将为 “0.0174532925”。

示例：

```
radians(last(/host/key))
```

rand

返回一个随机整数值。使用时间作为种子生成的伪随机数（对于数学用途已足够，但不适用于加密）。

示例：

```
rand()
```

round(value,decimal places)

将该值四舍五入到指定的小数位数。

参数：

- **value** - 要检查的值；
- **decimal places** - 指定用于四舍五入的小数位数（也可以为 0）。

例如，值 “2.5482” 四舍五入到 2 位小数后将变为 “2.55”。

示例：

```
round(last(/host/key),2)
```

signum(value)

如果值为负数，则返回’-1’；如果值为零，则返回’0’；如果值为正数，则返回’1’。

参数：

- **value** - 要检查的值。

示例：

```
signum(last(/host/key))
```

sin(value)

一个值的正弦，其中该值是以弧度表示的角度。

参数：

- **value** - 要检查的值

例如，值 “1” 的正弦将为 “0.8414709848”。

示例：

`sin(last(/host/key))`

sinh(value)

值的双曲正弦。

参数：

- **value** - 要检查的值

例如，值 “1” 的双曲正弦为 “1.17520119364”。

示例：

`sinh(last(/host/key))`

sqrt(value)

值的平方根。
 如果值为负数，此函数将失败。

参数：

- **value** - 要检查的值

例如，值 “3.5” 的平方根将为 “1.87082869339”。

示例：

`sqrt(last(/host/key))`

sum(<value1>,<value2>,...)

引用的监控项值之和。

参数：

- **valueX** - 由另一个处理监控项历史数据的函数返回的值。

示例：

`sum(avg(/host/key,1h),avg(/host2/key2,1h))`

tan(value)

一个值的正切。

参数：

- **value** - 要检查的值

例如，值 “1” 的正切将为 “1.55740772465”。

示例：

`tan(last(/host/key))`

truncate(value,decimal places)

将该值截断到指定的小数位数。

参数：

- **value** - 要检查的值；
- **decimal places** - 指定用于截断的小数位数（也可以为 0）。

例如，值 “2.5482” 截断到 2 位小数后将变为 “2.54”。

示例：

`truncate(last(/host/key),2)`

请参见所有支持的函数。

运算符函数

此处列出的所有函数均支持用于：

- 触发器表达式
- 计算监控项公式
- 表达式宏

函数列表不包含其他信息。单击函数名称可查看完整详细信息。

函数	描述
between	检查值是否属于给定范围。
in	检查值是否等于列出的至少一个值。

函数详情

有关函数参数的一些一般说明：

- 函数参数之间使用逗号分隔
- 表达式可作为参数接受

between(value,min,max)

检查该值是否属于给定范围。
 支持的值类型：整数、浮点数。
 返回：1 - 在范围内；0 - 不在范围内。

参数：

- **value** - 要检查的值；
- **min** - 最小值；
- **max** - 最大值。

示例：

`between(last(/host/key),1,10)=1 #如果该值介于 1 和 10 之间，则触发触发器`

in(value,value1,value2,...valueN)

检查该值是否等于所列值中的至少一个。
 支持的值类型：*Integer*、*Float*、*Character*、*Text*、*Log*。
 返回：1 - 如果相等；0 - 否则。

参数：

- **value** - 要检查的值；
- **valueX** - 列出的值（字符串值必须用双引号括起来）。

如果这些值都可以转换为数值，则将该值与列出的值按数字进行比较；否则按字符串进行比较。

示例：

```
in(last(/host/key),5,10)=1 #如果最后一个值等于 5 或 10，则触发触发器
in("text",last(/host/key),last(/host/key,#2))=1 #如果 "text" 等于最后 2 个值中的任意一个，则触发触发器
```

请参见所有支持的函数。

预测函数

此处列出的所有函数都支持以下内容：

- 触发器表达式
- 计算型监控项公式
- 表达式宏

这些函数仅列出名称，不包含其他信息。单击函数可查看完整详情。

Function	Description
forecast	监控项的未来值、最大值、最小值、差值或平均值。
timeleft	监控项达到指定阈值所需的秒数。

常用参数

- `/host/key` 是引用主机监控项历史数据的函数的通用必需第一个参数。
- `(sec|#num)<:time shift>` 是引用主机监控项历史数据的函数的通用第二个参数，其中：
  - **sec** - 以秒为单位的最大评估周期（可以使用时间后缀），或者
  - **#num** - 最新采集值中的最大评估范围（如果前面带有井号）。
  - **time shift**（可选）- 允许将评估时间点向过去移动。有关如何指定时间偏移的更多详细信息。

函数详情

有关函数参数的一些一般说明：

- 函数参数以逗号分隔
- 可选函数参数（或参数部分）以 < > 表示
- 每个函数都会说明其特定参数
- /host/key 和 (sec|#num)<:time shift> 参数绝不能加引号

forecast(/host/key,(sec|#num)<:time shift>,time,<fit>,<mode>)

监控项的未来值、最大值、最小值、差值或平均值。
 支持的值类型：*Float*、*Integer*。

参数：

- 参见通用参数；
- **time** - 预测时间范围（以秒为单位）（例如 1、60）（可以使用时间后缀（例如 30s、1m）；支持负值；
- **fit** (可选；必须使用双引号) - 用于拟合历史数据的函数。支持的拟合函数：
linear - 线性函数 (默认)
polynomialN - N 次多项式 (1 <= N <= 6)
exponential - 指数函数
logarithmic - 对数函数
power - 幂函数
 注意，*polynomial1* 等同于 *linear*；
- **mode**（可选；必须使用双引号）- 所需的输出结果。支持的模式：
value - 值（默认）
max - 最大值
min - 最小值
delta - *max-min*
avg - 平均值
 注意，*value* 估算时刻 now + time 的监控项值；*max*、*min*、*delta* 和 *avg* 则计算从 now 到 now + time 时间区间内的监控项值估算结果。

注释：

- 如果要返回的值大于 1.7976931348623158E+308 或小于 -1.7976931348623158E+308，则返回值分别截取为 1.7976931348623158E+3
  或 -1.7976931348623158E+308；
- 仅当在表达式中使用不当时才会变为不支持（监控项类型错误、参数无效）；否则，发生错误时返回 -1；
- 另请参见预测触发器函数的其他信息。

示例：

```
forecast(/host/key,#10,1h) #根据最近 10 个值预测一小时后的监控项值
forecast(/host/key,1h,30m) #根据最近一小时的数据预测 30 分钟后的监控项值
forecast(/host/key,1h:now-1d,12h) #根据一天前一小时内的数据预测 12 小时后的监控项值
forecast(/host/key,1h,10m,"exponential") #根据最近一小时的数据和指数函数预测 10 分钟后的监控项值
forecast(/host/key,1h,2h,"polynomial3","max") #根据最近一小时的数据和三次（第三阶）多项式，预测监控项在未
forecast(/host/key,#2,-20m) #根据最近两个值估算 20 分钟前的监控项值（这可能比使用 last() 更精确，尤其是在
```

timeleft(/host/key,(sec|#num)<:time shift>,threshold,<fit>)

监控项达到指定阈值所需的时间（以秒为单位）。
 支持的值类型：*Float*、*Integer*。

参数：

- 请参阅通用参数；
- **threshold** - 要达到的值（可以使用单位后缀）；
- **fit**（可选；必须使用双引号）- 请参阅forecast()。

注释：

- 如果要返回的值大于 1.7976931348623158E+308，则返回值将截断为 1.7976931348623158E+308；
- 如果无法达到阈值，则返回 1.7976931348623158E+308；
- 只有在表达式中使用不当（监控项类型错误、参数无效）时才会变为不支持；否则，发生错误时返回 -1；
- 另请参阅预测性触发器函数的其他信息。

示例：

```
timeleft(/host/key,#10,0) #基于最近 10 个值，监控项值达到零所需的时间
timeleft(/host/key,1h,100) #基于最近一小时的数据，监控项值达到 100 所需的时间
timeleft(/host/key,1h:now-1d,100) #基于一天前一小时的数据，监控项值达到 100 所需的时间
timeleft(/host/key,1h,200,"polynomial2") #基于最近一小时的数据，并假设监控项的行为类似于二次（第二阶）多项
```

请参阅所有支持的函数。

预测性触发器函数

概述

有时，问题发生前会出现一些征兆。可以识别这些征兆，以便提前采取措施来预防问题，或至少将问题的影响降到最低。

Zabbix 提供了基于历史数据预测被监控系统未来行为的工具。这些工具通过预测型触发器函数来实现。

函数

在设置触发器之前，必须先定义什么是问题状态，以及需要多少时间来采取行动。然后，有两种方式可以设置一个用于指示潜在异常情况的触发器。第一种：当系统预计会在“采取行动时间”之后处于问题状态时，触发器必须触发。第二种：当系统将在少于“采取行动时间”的时间内达到问题状态时，触发器必须触发。对应可使用的触发器函数是 **forecast** 和 **timeleft**。请注意，这两个函数所基于的统计分析本质上是相同的。你可以根据自己的偏好选择任意一种方式来设置触发器，并获得类似的结果。

参数

这两个函数使用的参数集几乎相同。请参考支持的函数列表。

时间间隔

首先，您应指定 Zabbix 为得出预测结果而需要分析的历史时间段。您可以像使用 **avg**、**count**、**delta**、**max**、**min** 和 **sum** 函数时一样，通过 `time period` 参数以及可选的时间偏移，以熟悉的方式完成此设置。

预测时间范围

(仅 **forecast**）
 参数 `time` 指定 Zabbix 应根据历史数据中发现的依赖关系向未来外推多远。无论是否使用 `time_shift`，`time` 始终从当前时刻开始计算。

达到阈值

(**timeleft** 专用）
 参数 `threshold` 指定被分析的监控项必须达到的值，无论是从上方还是从下方达到都没有区别。一旦我们确定了 f(t)（见下文），就应求解方程 f(t) = `threshold`，并返回距离当前时刻最近且位于当前时刻右侧的根；如果不存在这样的根，则返回 1.7976931348623158E+308。

Note:
当监控项的值接近阈值并随后穿过该阈值时，**timeleft** 会假定该交点已经发生在过去，因此会切换到与 `threshold` 水平的下一个交点（如果存在）。最佳实践是将预测作为常规问题诊断的补充，而不是替代。[1]

拟合函数

默认的 `fit` 是 *linear* 函数。但如果您的被监控系统更复杂，您还有更多选项可供选择。

`fit`	x = f(t)
linear	$x = a + b*t$
polynomialN[2]	$x = a_0 + a_1*t + a_2*t^2 + ... + a_n*t^n$
exponential	$x = a*exp(b*t)$
logarithmic	$x = a + b*log(t)$
power	$x = a*t^b$

模式

(**forecast** 专用）
 每次对触发器函数进行求值时，它都会从指定的历史周期中获取数据，并将指定的函数拟合到这些数据上。因此，如果数据略有不同，拟合后的函数也会略有不同。如果我们只是简单地计算该拟合函数在未来某个指定时间点的值，那么你将无法了解从现在到未来该时刻之间，被分析监控项的预期行为。对于某些 `fit` 选项（如 *polynomial*），未来的单一数值可能会产生误导。

`mode`	**forecast** 结果
value	f(now + `time`)
max	$\max_{now <= t <= now + time} f(t)$
min	$\min_{now <= t <= now + time} f(t)$
delta	*max - min*
avg	根据定义，f(t) 在 (now <= t <= now + `time`) 区间内的平均值

详细信息

为避免使用超大数字进行计算，我们将指定时间段内第一个值的时间戳加上 1 ns 视为新的零时间（当前 epoch 时间的量级为 10^9，epoch 的平方为 10^{18}，双精度约为 10^{-16}）。增加 1 ns 是为了给 *logarithmic* 和 *power* 拟合提供全部为正的时间值，因为它们涉及计算 log(t)。时间偏移不会影响 *linear*、*polynomial*、*exponential*（除了使计算更容易且更精确）拟合，但会改变 *logarithmic* 和 *power* 函数的形状。

可能的错误

在以下情况下，函数会返回 -1：

- 指定的评估周期不包含任何数据；

- 数学运算结果未定义[3]；
- 数值计算问题（遗憾的是，对于某些输入数据集，双精度浮点格式的范围和精度会变得不足）[4]。

Note:
如果所选拟合方式不能很好地描述所提供的数据，或者数据量过少而无法进行准确预测，则不会标记任何警告或错误。

示例与错误处理

要在主机即将耗尽可用磁盘空间时收到警告，您可以使用如下触发器表达式：

```
timeleft(/host/vfs.fs.size[/,free],1h,0)<1h
```

但是，错误代码 -1 可能会生效，并使您的触发器进入问题状态。通常来说，这是有益的，因为您会收到一个警告，表明您的预测未能正确工作，您应该更仔细地检查它们以找出原因。但有时这也不好，因为 -1 可能仅仅意味着在过去一小时内没有获取到有关主机可用磁盘空间的数据。如果您收到了过多误报，请考虑使用更复杂的触发器表达式[5]：

```
timeleft(/host/vfs.fs.size[/,free],1h,0)<1h and timeleft(/host/vfs.fs.size[/,free],1h,0)<>-1
```

对于 **forecast**，情况会稍微复杂一些。首先，-1 可能会或可能不会使触发器进入问题状态，这取决于您使用的是类似 `forecast(/host/item,(...))<...` 的表达式，还是类似 `forecast(/host/item,(...))>...` 的表达式。

此外，如果监控项值为负数是正常情况，那么 -1 也可能是一个有效的预测值。但这种情况在现实世界中的概率可以忽略不计（请参见运算符 = 的工作方式）。因此，如果您希望将 -1 视为问题或不视为问题，请分别添加 `... or forecast(/host/item,(...))=-1` 或 `... and forecast(/host/item,(...))<>-1`。

脚注

[1] 例如，像 `timeleft(/host/item,1h,X) < 1h` 这样一个简单的触发器，可能会在监控项值接近 X 时进入问题状态，然后在达到值 X 后突然恢复。如果问题是监控项值低于 X，请使用：`last(/host/item) < X or timeleft(/host/item,1h,X) < 1h` 如果问题是监控项值高于 X，请使用：`last(/host/item) > X or timeleft(/host/item,1h,X) < 1h`

[2] 多项式阶数可以为 1 到 6，*polynomial1* 等同于 *linear*。但是，请谨慎使用高阶多项式。如果评估周期内包含的点少于确定多项式系数所需的数量，则会降低多项式阶数（例如，请求的是 *polynomial5*，但只有 4 个点，因此将拟合 *polynomial3*）。

[3] 例如，拟合 *exponential* 或 *power* 函数时，需要计算监控项值的 log()。如果数据包含零或负数，您将收到错误，因为 log() 仅对正值有定义。

[4] 对于 *linear*、*exponential*、*logarithmic* 和 *power* 拟合，所有必要的计算都可以显式写出。对于 *polynomial*，只有 *value* 可以在不增加额外步骤的情况下计算。计算 *avg* 需要计算多项式的原函数（解析地）。计算 *max*、*min* 和 *delta* 需要计算多项式的导数（解析地）并找到其根（数值地）。求解 f(t) = 0 需要找到多项式的根（数值地）。

[5] 但在这种情况下，-1 可能会导致您的触发器从问题状态恢复。要获得完整保护，请使用：`timeleft(/host/vfs.fs.size[/,free],1h,0)<1h and ({TRIGGER.VALUE}=0 and timeleft(/host/vfs.fs.size[/,free],1h,0)<>-1 or {TRIGGER.VALUE}=1)`

字符串函数

此处列出的所有函数均支持用于：

- 触发器表达式
- 计算监控项公式
- 表达式宏

下方列出了函数名称及简要说明。点击函数可查看完整详细信息。

函数	描述
ascii	值中最左侧字符的 ASCII 代码。
bitlength	值的位长度。
bytelength	值的字节长度。
char	将值解释为 ASCII 代码并返回对应字符。
concat	连接所引用监控项的值或常量值后得到的字符串。
contains	如果值包含 `search value` 指定的子字符串，则返回 1，否则返回 0。
insert	从字符串中指定的位置开始插入指定的字符或空格。
jsonpath	返回 JSONPath 结果。
left	返回值中最左侧的字符。
length	值的字符长度。
ltrim	从字符串开头移除指定的字符。
mid	从由 `start` 指定的字符位置开始，返回包含 N 个字符的子字符串。此函数是 `substring()` 函数的别名。
repeat	重复字符串。
replace	在值中查找模式，并将其替换为指定的替换内容。

函数	描述
right	返回值中最右侧的字符。
rtrim	从字符串末尾移除指定的字符。
substring	从由 `start` 指定的字符位置开始，返回包含 N 个字符的子字符串。
trim	从字符串开头和末尾移除指定的字符。
xmlxpath	返回 XML XPath 结果。

函数详情

有关函数参数的一些一般说明：

- 函数参数以逗号分隔
- 表达式可作为参数
- 字符串参数必须使用双引号；否则可能会被误解
- 可选的函数参数（或参数部分）以 < > 表示

ascii(value)

值最左侧字符的 ASCII 代码。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值

例如，像'Abc' 这样的值将返回'65'（'A' 的 ASCII 代码）。

示例：

```
ascii(last(/host/key))
```

bitlength(value)

value 的位长度。
 支持的值类型：*String*、*Text*、*Log*、*Integer*。

参数：

- **value** - 要检查的值

示例：

```
bitlength(last(/host/key))
```

bytelength(value)

value 的字节长度。
 支持的值类型：*String*、*Text*、*Log*、*Integer*。

参数：

- **value** - 要检查的值

示例：

```
bytelength(last(/host/key))
```

char(value)

通过将该值解释为 ASCII 代码来返回相应字符。
 支持的值类型：*Integer*。

参数：

- **value** - 要检查的值

该值必须在 0-255 范围内。例如，像 "65" 这样的值（解释为 ASCII 代码）将返回 "A"。

示例：

```
char(last(/host/key))
```

concat(<value1>,<value2>,...)

通过连接引用的监控项值或常量值而得到的字符串。
 支持的值类型：*String*、*Text*、*Log*、*Float*、*Integer*。

参数：

- **valueX** - 由某个历史函数返回的值或常量值（字符串、整数或浮点数）。必须至少包含两个参数。

例如，将值 "Zab" 与 "bix"（常量字符串）连接后，将返回 "Zabbix"。

示例：

```
concat(last(/host/key),"bix")
concat("1 min: ",last(/host/system.cpu.load[all,avg1]),", 15 min: ",last(/host/system.cpu.load[all,avg15]))
```

contains(value, search value)

如果 value 包含 search value 指定的子字符串，则返回 1，否则返回 0。| 支持的 value 类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **search value** - 要搜索的值。

注释：

- 搜索区分大小写。
- 当 value 有效且 `search value` 为空时，函数返回 1。

例如，如果 value 包含 “Zabbix”，则函数返回 1。

示例：

```
contains(last(/host/key),"Zabbix")
```

insert(value,start,length,replacement)

从字符串中指定的位置开始，将指定字符或空格插入到该字符串中。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **start** - 起始位置；
- **length** - 要替换的位置数；
- **replacement** - 替换字符串。

例如，如果将值’Zabbbix’ 中的’bb’（起始位置为 3，要替换的位置数为 2）替换为’b’，则该值将变为’Zabbix’。

示例：

```
insert(last(/host/key),3,2,"b")
```

jsonpath(value,path,<default>)

返回 JSONPath 结果。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **path** - 路径（必须加引号）；
- **default** - 可选的回退值，如果 JSONPath 查询未返回数据则使用该值。请注意，发生其他错误时将返回失败（例如 “unsupported construct”）。

示例：

```
jsonpath(last(/host/proc.get[zabbix_agentd,,,summary]),"$..size")
```

left(value,count)

返回值最左侧的字符。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **count** - 要返回的字符数。

例如，通过指定返回最左侧 3 个字符，您可以从 “Zabbix” 中返回 “Zab”。另请参见right()。

示例：

```
left(last(/host/key),3) #返回最左侧三个字符
```

length(value)

value 的字符长度。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值。

示例：

```
length(last(/host/key)) #最新值的长度
length(last(/host/key,#3)) #倒数第三个最新值的长度
length(last(/host/key,#1:now-1d)) #一天前最近值的长度
```

ltrim(value,<chars>)

从字符串开头移除指定字符。
 支持的值类型：字符串、文本、日志。

参数：

- **value** - 要检查的值；
- **chars**（可选）- 指定要移除的字符。

默认会移除开头的空白字符（如果未指定可选字符）。另请参见：rtrim()、trim()。

示例：

```
ltrim(last(/host/key)) #从字符串开头移除空白字符
ltrim(last(/host/key),"Z") #从字符串开头移除所有 'Z'
ltrim(last(/host/key)," Z") #从字符串开头移除所有空格和 'Z'
```

mid(value,start,length)

返回从 “start” 指定的字符位置开始的 N 个字符组成的子字符串。
 此函数是 `substring()` 函数的别名。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **start** - 子字符串的起始位置；
- **length** - 要在子字符串中返回的字符数。

例如，如果起始位置为 2，要返回的字符数为 4，则可以从类似 “Zabbix” 的值中返回 “abbi”。

示例：

```
mid(last(/host/key),2,4)="abbi"
```

repeat(value,count)

重复一个字符串。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **count** - 重复的次数。

示例：

```
repeat(last(/host/key),2) #将该值重复两次
```

replace(value,pattern,replacement)

在值中查找模式，并替换为 replacement。模式的所有匹配项都会被替换。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **pattern** - 要查找的模式；
- **replacement** - 用于替换该模式的字符串。

示例：

```
replace(last(/host/key),"ibb","abb") #将所有 'ibb' 替换为 'abb'
```

right(value,count)

返回值最右侧的字符。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **count** - 要返回的字符数。

例如，通过指定返回最右侧的 3 个字符，您可以从 “Zabbix” 中返回 “bix”。另请参见left()。

示例：

```
right(last(/host/key),3) #返回最右侧的三个字符
```

rtrim(value,<chars>)

从字符串末尾移除指定字符。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **chars**（可选）- 指定要移除的字符。

默认会移除右侧空白字符（如果未指定可选字符）。另请参见：ltrim()、trim()。

示例：

```
rtrim(last(/host/key)) #从字符串末尾移除空白字符
rtrim(last(/host/key),"x") #从字符串末尾移除所有 'x'
rtrim(last(/host/key),"x ") #从字符串末尾移除所有 'x' 和空格
```

substring(value,start,length)

返回从 “start” 指定的字符位置开始、包含 N 个字符的子字符串。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **start** - 子字符串的起始位置；
- **length** - 子字符串中要返回的字符数。

例如，如果起始位置为 2，要返回的字符数为 4，则可以从类似 “Zabbix” 的值中返回 “abbi”。

示例：

```
substring(last(/host/key),2,4)="abbi"
```

trim(value,<chars>)

从字符串的开头和结尾移除指定字符。
 支持的值类型：字符串、文本、日志。

参数：

- **value** - 要检查的值；
- **chars**（可选）- 指定要移除的字符。

默认会从两端去除空白字符（如果未指定可选字符）。另请参见：ltrim()、rtrim()。

示例：

```
trim(last(/host/key)) #从字符串的开头和结尾移除空白字符
trim(last(/host/key),"_") #从字符串的开头和结尾移除 '_'
```

xmlxpath(value,path,<default>)

返回 XML XPath 结果。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **value** - 要检查的值；
- **path** - 路径（必须加引号）；
- **default** - 可选的回退值，如果 XML XPath 查询返回空节点集，则使用该值。如果空结果不是节点集（即空字符串），则不会返回该值。发生其他错误时将返回失败（例如 “invalid expression”）。

示例：

```
xmlxpath(last(/host/xml_result),"/response/error/status")
```

参见所有支持的函数。

触发器最佳实践

触发器是一个强大的工具，但它们也可能产生不必要的告警噪音。若要获得更多真实信号、减少噪音，请参考以下建议：

1. 降低触发器的敏感度。不要针对最新值（过高/过低）进行告警，而是使用 **avg**、**min** 和 **max** 等函数，对较长时间段内的平均值进行分析。
2. 如果你想避免因随机峰值和骤降而触发告警，可以考虑在触发器中使用 **percentile** 函数（将其设置为 95% 或 5%）。
3. 使用hysteresis 来避免触发器抖动——即触发器状态频繁变化（OK ↔ Problem）。可以通过添加恢复表达式（用于问题恢复的单独条件）来定义问题状态的连续区间。

4. 使用触发器dependencies 来避免与根本原因无关的告警。
5. 使用触发器severity 仅对更严重的问题进行告警。
6. 定义maintenance 窗口。

滞后

有时，问题状态与恢复状态之间需要一个区间，而不是一个简单的阈值。例如，如果我们想定义一个触发器：当服务器机房温度高于 20°C 时报告问题，并且希望它保持在问题状态，直到温度降到 15°C 以下，那么仅设置一个 20°C 的简单触发器阈值是不够的。

相反，我们需要先为问题事件定义一个触发器表达式（温度高于 20°C）。然后还需要定义一个额外的恢复条件（温度低于 15°C）。这可以通过在配置触发器时定义一个恢复表达式来实现。

在这种情况下，问题恢复分两步进行：

- 首先，问题表达式（温度高于 20°C）必须计算为 FALSE
- 其次，恢复表达式（温度低于 15°C）必须计算为 TRUE

恢复表达式仅在问题事件被解决后才会进行计算。仅恢复表达式为 TRUE 并不会解决问题，如果问题表达式仍然为 TRUE！

示例

服务器机房温度过高。

问题表达式：

```
last(/server/temp)>20
```

恢复表达式：

```
last(/server/temp)<=15
```

> **Note:**
> 在恢复表达式中使用 {TRIGGER.VALUE} 宏没有实际意义，因为该表达式仅在触发器处于“问题”状态时才会被求值。因此，在对表达式求值时，{TRIGGER.VALUE} 始终会解析为“1”（表示“问题”状态）。

触发器依赖关系

触发器依赖关系可用于避免与根本原因无关的告警。

请参见所有最佳实践。

概述

有时，一个主机的可用性取决于另一个主机。位于路由器后面的服务器将在路由器宕机时变得不可达。如果为这两个主机都配置了触发器，您可能会收到两个主机宕机的通知，而实际上只有路由器是故障源。

这时，设置主机之间的某种依赖关系可能会很有用。设置依赖关系后，可以暂缓发送依赖项的通知，只发送根问题的通知。

虽然 Zabbix 不直接支持主机之间的依赖关系，但可以通过另一种更灵活的方法来定义依赖关系，即触发器依赖关系。一个触发器可以依赖一个或多个其他触发器。

在这个简单示例中，我们打开服务器触发器配置表单，并将其设置为依赖路由器的相应触发器。设置此依赖关系后，只要其依赖的触发器处于“PROBLEM”状态，服务器触发器就不会更改其状态，因此不会执行依赖于该触发器的操作，也不会发送通知。

如果服务器和路由器都宕机，并且已在“服务器宕机”触发器与“路由器宕机”触发器之间设置依赖关系，Zabbix 将不会为依赖触发器执行操作。

当父触发器处于 PROBLEM 状态时，其依赖触发器可能会报告不可信的值。因此，在父触发器（上述示例中的路由器）发生以下情况之前，不会重新评估依赖触发器：

- 从“PROBLEM”状态恢复为“OK”状态；
- 将其状态从“PROBLEM”更改为“UNKNOWN”；
- 通过手动方式、关联或借助date and time 和/或nodata() 函数关闭；
- 由不参与依赖触发器的监控项的值解决；
- 被禁用、具有已禁用的监控项，或具有已禁用监控项的主机。

在上述所有情况下，只有在收到该依赖触发器的新指标后，才会重新评估依赖触发器（服务器）。这意味着依赖触发器可能不会立即更新。

此外：

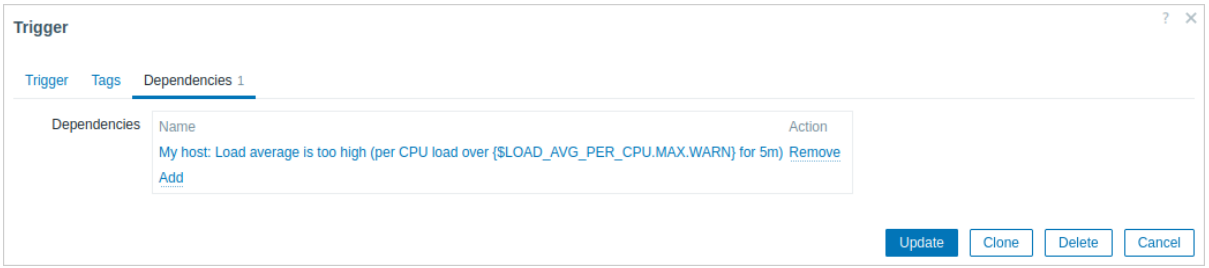
- 只要不会导致循环依赖，就可以将触发器依赖关系添加到任意主机的触发器与其他任意主机的触发器之间。

- 可以在一个模板与另一个模板之间添加触发器依赖关系。如果模板 A 中的某个触发器依赖于模板 B 中的某个触发器，则模板 A 只能与模板 B 一起链接到主机（或另一个模板），而模板 B 可以单独链接到主机（或另一个模板）。
- 可以将模板触发器添加为对主机触发器的依赖。在这种情况下，将此类模板链接到主机时，会创建一个主机触发器，该触发器依赖于原触发器所依赖的同一个模板触发器。例如，这允许您创建一个模板，其中某些触发器依赖于路由器（主机）的触发器。所有链接到此模板的主机都将依赖于该特定路由器。
- 不可以将主机触发器添加为对模板触发器的依赖。
- 可以将触发器原型添加为对另一个触发器原型（位于同一个低级别发现规则中）或实际触发器的依赖。触发器原型不能依赖于其他 LLD 规则中的触发器原型，也不能依赖于由触发器原型创建的触发器。主机触发器原型不能依赖于模板中的触发器。

配置

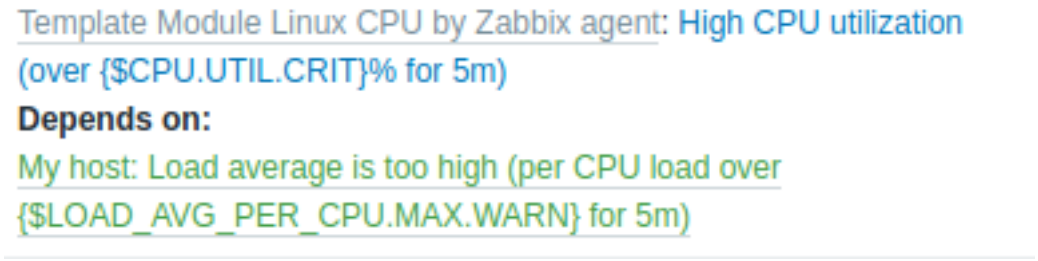
要定义依赖关系：

1. 在触发器配置表单中打开依赖关系选项卡。
2. 在依赖关系部分中点击添加，然后选择一个或多个该触发器所依赖的触发器。



3. 点击更新。

现在，触发器列表中会显示其依赖关系指示。



多个依赖关系示例

例如，主机位于 Router2 后面，而 Router2 位于 Router1 后面。

```
Zabbix - Router1 - Router2 - Host
```

如果 Router1 宕机，那么显然主机和 Router2 也都不可达，但同时收到关于主机、Router1 和 Router2 全部宕机的三条通知就显得过多了。

因此，在这种情况下，我们定义两个依赖关系：

```
the 'Host is down' trigger depends on the 'Router2 is down' trigger
the 'Router2 is down' trigger depends on the 'Router1 is down' trigger
```

在更改'Host is down' 触发器的状态之前，Zabbix 会检查相应的触发器依赖关系。如果找到了此类依赖关系，并且其中某个触发器处于 **Problem** 状态，那么触发器状态将不会更改，也不会执行操作，并且不会发送任何通知。

Zabbix 会递归执行此检查。如果 Router1 或 Router2 不可达，Host 触发器将不会更新。

触发器严重性

触发器严重性可用于避免因不太严重的问题而产生告警噪音。

请参见所有最佳实践。

概述

严重性表示触发器的相对重要程度。

分配严重性不仅有助于可视化展示，也有助于避免针对琐碎问题发送通知。

触发器严重性用于：

- 可视化展示（根据原始触发器严重性，为每个已创建的问题显示不同颜色）
- 根据严重性限制通知（请参见动作条件）
- 用户介质 - 针对不同严重性使用不同介质（通知渠道）（例如，对 *High* 和 *Disaster* 严重性的触发器使用 SMS，而对其他严重性的触发器使用 Email）
- 全局告警中的音频 - 针对不同严重性使用不同音频

默认支持以下触发器严重性：

严重性	颜色	描述
未分类	灰色	可用于事件严重性级别未知、尚未确定、不属于常规监控范围等情况，例如在初始配置期间，作为后续评估的占位符，或作为集成
信息	浅蓝色	可用于不需要立即关注、但仍可提供有价值见解的信息类事件。
警告	黄色	可用于表示可能需要调查或处理、但尚不严重的潜在问题。
一般	橙色	可用于表示较为重要的问题，应在相对较短时间内处理，以防止进一步问题发生。
高	浅红色	可用于表示需要立即关注的关键问题，以避免造成重大中断。
灾难	红色	可用于表示需要立即采取行动的严重事件，以防止例如系统中断或数据丢失。

可在 *Administration* > *General* > *Trigger displaying options* 中自定义触发器严重性的名称和颜色。
颜色在所有前端主题之间共享。

翻译自定义严重性名称

> **Attention:**
> 如果使用 Zabbix 前端翻译，则默认情况下，自定义严重性名称将覆盖已翻译的名称。

默认的触发器严重性名称可在所有语言环境中进行翻译。
如果更改了严重性名称，则所有语言环境都会使用自定义名称，因此需要额外进行手动翻译。

自定义严重性名称翻译步骤：

- 设置所需的自定义严重性名称，例如 “Important”
- 编辑 <frontend_dir>/locale/<required_locale>/LC_MESSAGES/frontend.po
- 添加 2 行：

```
msgid "Important"
msgstr "<翻 译 字 符 串>"
```

并保存文件。

- 按照 <frontend_dir>/locale/README 中的说明创建.mo 文件

这里，**msgid** 应与新的自定义严重性名称一致，而 **msgstr** 应为该名称在特定语言中的翻译。

每次更改严重性名称后，都应执行此过程。

可视化

概述

可视化提供了多种方式来显示 Zabbix 收集的监控数据。图形展示趋势和指标，网络拓扑图显示基础设施拓扑，而仪表板则通过组合可自定义的小部件来提供全面的监控视图。

可视化工具

- 图形 - 以简单的自动生成图形查看监控项数据，创建自定义图形，或在临时图形中比较多个监控项。
- 网络拓扑图 - 创建网络基础架构的可视化表示，并可自定义元素和指示器。
- 仪表板 - 自定义布局，组合各种仪表板小部件，以不同格式可视化数据。

图形

概述

随着大量数据不断流入 Zabbix，如果用户能够查看当前状况的可视化表示，而不仅仅是数字，使用起来会更加方便。

这正是图形发挥作用的地方。图形可以帮助用户一目了然地掌握数据流，关联问题，发现问题何时开始，或说明某种情况何时可能演变为问题。

Zabbix 为用户提供：

- 内置的单个监控项数据简单图形。
- 创建更复杂的自定义图形的功能。
- 在临时图形中快速比较多个监控项的功能。
- 现代化且可自定义的矢量图形和饼图。

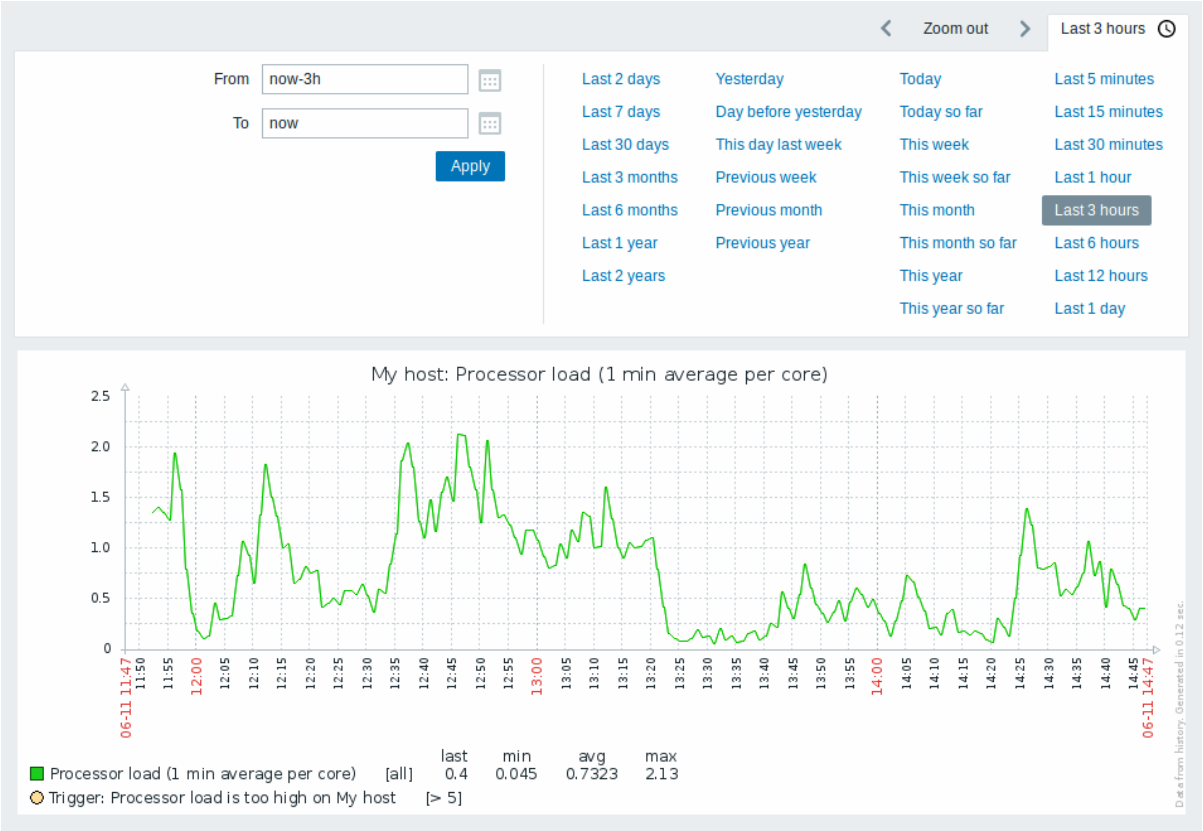
简单图形

概述

系统会为监控项收集的数据提供简单图形，以便进行可视化。

用户无需进行任何配置即可查看简单图形。Zabbix 会免费提供这些图形。

只需转到 监控 > 最新数据，然后单击相应监控项的 图形链接，即可显示图形。



Note:
所有数值型监控项都提供简单图形。对于文本型监控项，监控 > 最新数据中会提供指向 历史记录的链接。

时间段选择器

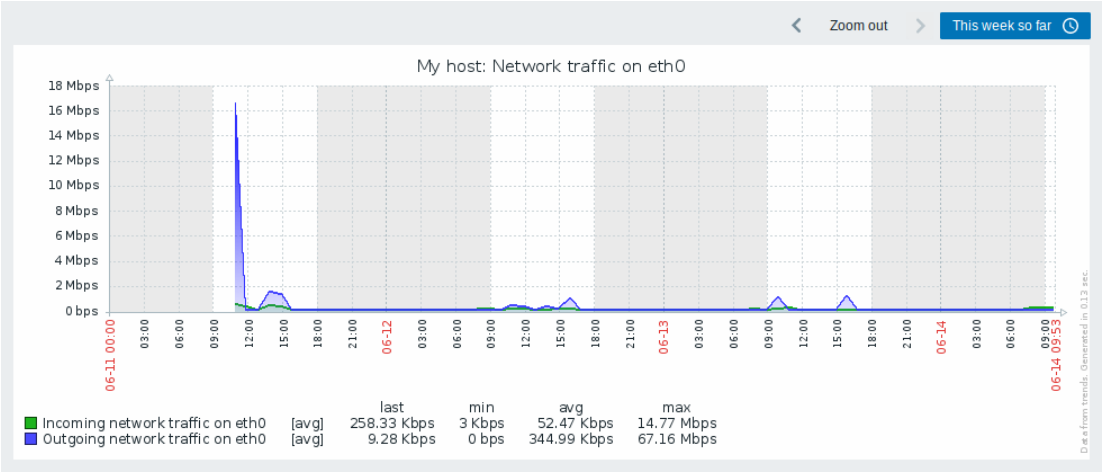
图表上方的时间段选择器允许您通过单击鼠标选择常用的时间段。
有关更多信息，请参阅时间段和主机选择器。

最近数据与较长时间段数据的区别

对于非常新的数据，会绘制一条单独的线来连接每个接收到的值。
只要在水平方向上至少还有一个像素可用于显示一个值，这条单线就会被绘制出来。

对于显示较长时间段的数据，则会绘制三条线——深绿色线表示平均值，而浅粉色线和浅绿色线分别表示该时间点的最大值和最小值。
最高值与最低值之间的区域会用黄色背景填充。

工作时间（工作日）在图形中显示为白色背景，而非工作时间显示为灰色背景（使用默认的 *Original blue* 前端主题时）。

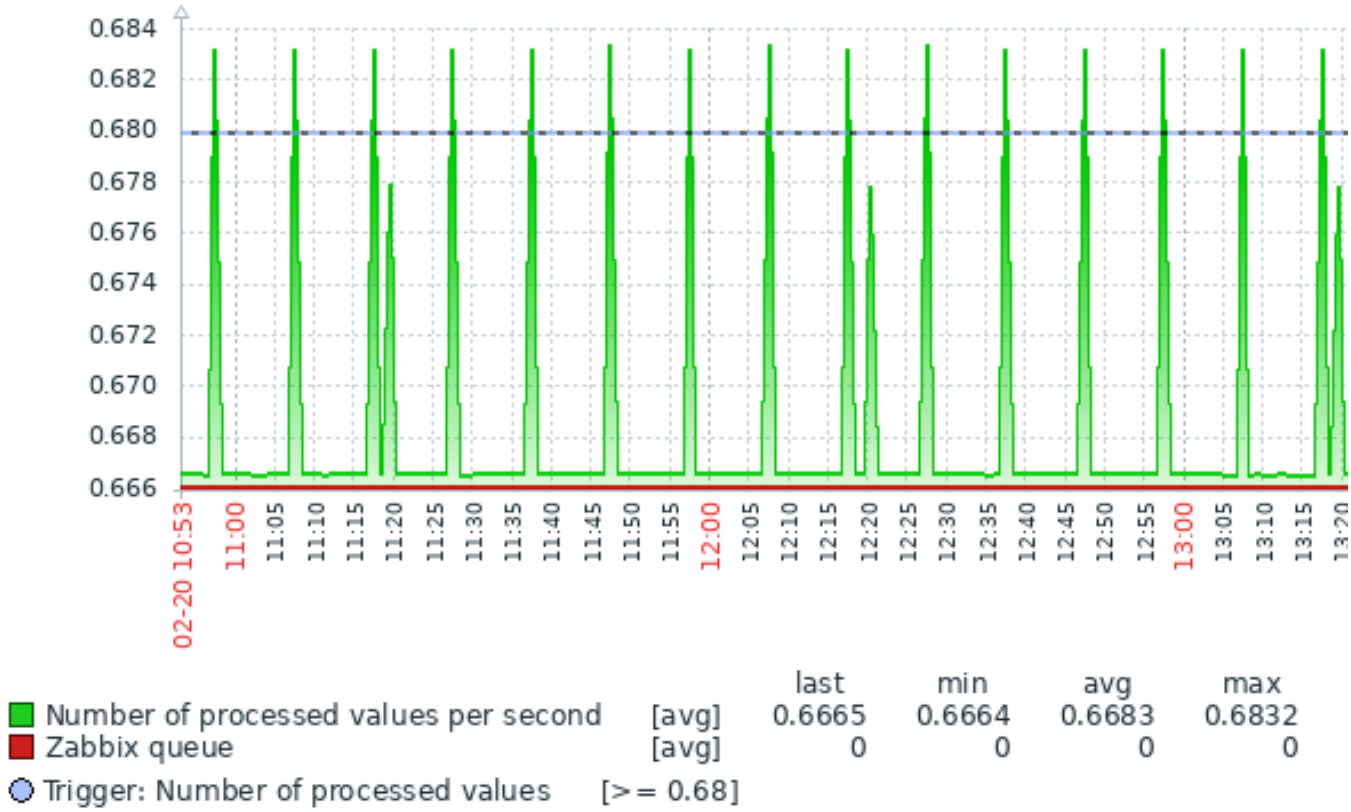


工作时间始终会显示在简单图形中，而在自定义图形中是否显示则取决于用户偏好设置。

如果图形显示的时间跨度超过 3 个月，则不会显示工作时间。

触发器线

简单触发器显示为叠加在触发器严重性颜色之上的黑色虚线——请注意图表中的蓝线以及图例中显示的触发器信息。
图表上最多可显示 3 条触发器线；如果触发器更多，则优先显示严重性较低的触发器。
触发器始终显示在简单图形中，而在自定义图形中是否显示它们则取决于用户偏好。



根据历史数据/趋势数据生成

图形可以基于监控项的历史数据或趋势数据绘制。

对于已启用前端调试模式的用户，图形右下角会显示灰色的垂直说明文字，指示数据来源。

以下因素会影响使用历史数据还是趋势数据：

- 监控项历史数据的保留时长。例如，监控项历史数据可以保留 14 天。在这种情况下，超过 14 天的数据将来自趋势数据。
- 图形中的数据拥挤程度。如果水平图形像素所表示的秒数超过 3600/16，则会显示趋势数据（即使同一时间段的监控项历史数据仍然可用）。
- 如果禁用了趋势数据，则使用监控项历史数据构建图形（如果该时间段的数据可用）。

数据缺失

对于具有常规更新间隔的监控项，如果未采集到监控项数据，则图形中不会显示任何内容。

但是，对于 trapper 监控项以及具有计划更新间隔的监控项（且常规更新间隔设置为 0），会绘制一条直线，从图形起始处延伸到第一个采集到的值，并从最后一个采集到的值延伸到图形末尾；该直线分别位于第一个/最后一个值的水平位置。

切换到原始值

右上角的下拉菜单可用于将简单图形切换为 值/最近 *500* 个值列表。这对于查看构成图形的数值可能很有用。

此处显示的值为原始值，即不使用单位或值的后处理。但是，会应用值映射。

已知的问题

有关图形，请参阅已知问题。

自定义图形

概述

顾名思义，自定义图形提供了自定义功能。

虽然简单图形适合查看单个监控项的数据，但它们不提供配置功能。

因此，例如，如果您想更改图形样式或线条显示方式，或者比较多个监控项（如在单个图形中比较入站和出站流量），则需要使用自定义图形。

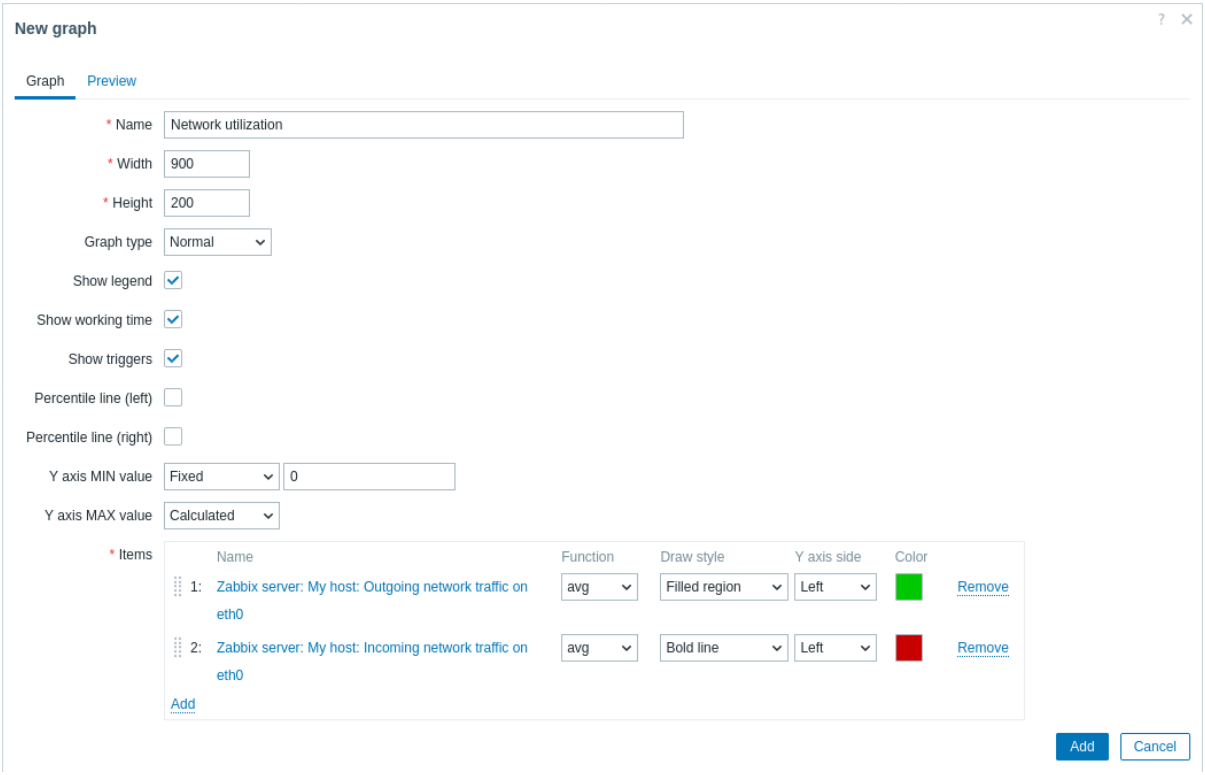
自定义图形需要手动配置。

它们可以为一台主机、多个主机或单个模板创建。

配置自定义图形

要创建自定义图形，请执行以下操作：

1. 转到 *Data collection* > *Hosts*（或 *Templates*）。
2. 在所需主机或模板旁边的行中，点击 *Graphs*。
3. 在 Graphs 屏幕中，点击 *Create graph*。
4. 编辑图形属性。



所有必填输入字段都用红色星号标记。

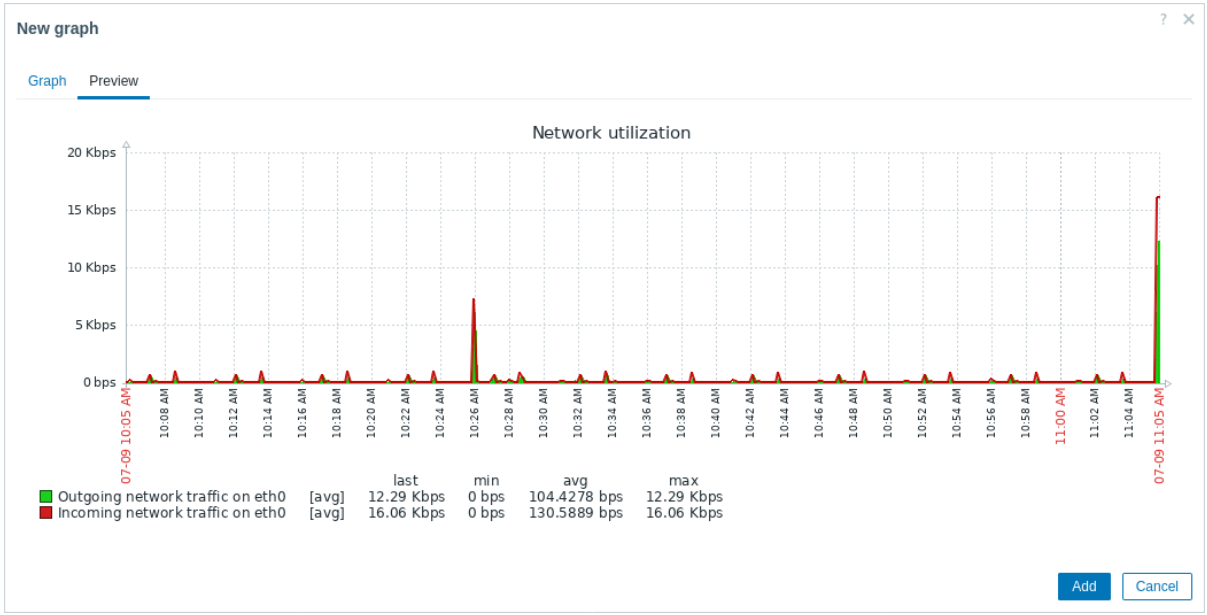
图形属性：

Parameter	Description
Name	唯一的图形名称。 此字段支持表达式macros，但仅支持带时间参数的 `avg`、`last`、`min` 和 `max` 函数（例如，`{?avg(/host/key,1h)}`）。 支持在此宏中使用 {HOST.HOST<1-9>} 宏，用于引用图形中的第一个、第二个、第三个等主机，例如，`{?avg(/{HOST.HOST2}/key,1h)}`。请注意，使用此宏引用第一个主机是多余的，因为可以隐式引用第一个主机，例如，`{?avg(//key,1h)}`。
Width	图形宽度（像素）（仅用于预览以及饼图/分解饼图）。
Height	图形高度（像素）。
Graph type	图形类型： **Normal** - 普通图形，值以线条显示 **Stacked** - 堆叠图形，显示填充区域 **Pie** - 饼图 **Exploded** - “分解” 饼图，各部分显示为从饼图中 “切出”
Show legend	勾选此框将显示图形图例。
Show working time	如果选中，非工作时间将以灰色背景显示。此参数不适用于饼图和分解饼图。
Show triggers	如果选中，简单触发器将以黑色虚线显示在触发器严重性颜色之上。此参数不适用于饼图和分解饼图。
Percentile line (left)	显示左侧 Y 轴的百分位线。例如，如果设置为 95% 百分位，则百分位线将位于 95% 的值低于该线的位置。显示为亮绿色线条。仅适用于普通图形。
Percentile line (right)	显示右侧 Y 轴的百分位线。例如，如果设置为 95% 百分位，则百分位线将位于 95% 的值低于该线的位置。显示为亮红色线条。仅适用于普通图形。
Y axis MIN value	Y 轴最小值： **Calculated** - 将自动计算 Y 轴最小值。 **Fixed** - Y 轴固定最小值。 **Item** - 所选监控项的最后一个值将作为最小值。 此参数不适用于饼图和分解饼图。
Y axis MAX value	Y 轴最大值： **Calculated** - 将自动计算 Y 轴最大值。 **Fixed** - Y 轴固定最大值。 **Item** - 所选监控项的最后一个值将作为最大值 此参数不适用于饼图和分解饼图。
3D view	启用 3D 样式。仅适用于饼图和分解饼图。
Items	要在此图形中显示数据的监控项。点击 *Add* 选择监控项。您还可以选择各种显示选项（函数、绘制样式、左/右轴显示、颜色）。
Sort order (0→100)	绘制顺序。0 将最先处理。可用于将线条或区域绘制在另一条线条或区域的后面（或前面）。 您可以使用行首的图标拖放监控项，以设置排序顺序或确定哪个监控项显示在另一个前面。
Name	所选监控项的名称将显示为链接。点击该链接可打开其他可用监控项列表。
Type	类型（仅适用于饼图和分解饼图）： **Simple** - 监控项的值按比例显示在饼图上 **Graph sum** - 监控项的值表示整个饼图 请注意，“graph sum” 监控项的着色仅在未被 “proportional” 监控项占用的范围内可见。
Function	当某个监控项在每个垂直图形像素上存在多个值时，选择要显示的值： **all** - 在图形中显示所有可能的值（最小值、最大值、平均值）。请注意，对于较短的时间段，此设置无效；仅在较长时间段、垂直图形像素中的数据拥塞增加时，’all’ 才会开始显示最小值、最大值和平均值。此函数仅适用于 *Normal* 图形类型。另请参见：从history/trends 生成图形。 **avg** - 显示平均值 **last** - 显示最新值。仅当图形类型选择 *Pie/Exploded pie* 时，此函数才可用。 **max** - 显示最大值 **min** - 显示最小值
Draw style	选择绘制样式（仅适用于普通图形；对于堆叠图形始终使用填充区域）以应用于监控项数据 - *Line*、*Bold line*、*Filled region*、*Dot*、*Dashed line*、*Gradient line*。

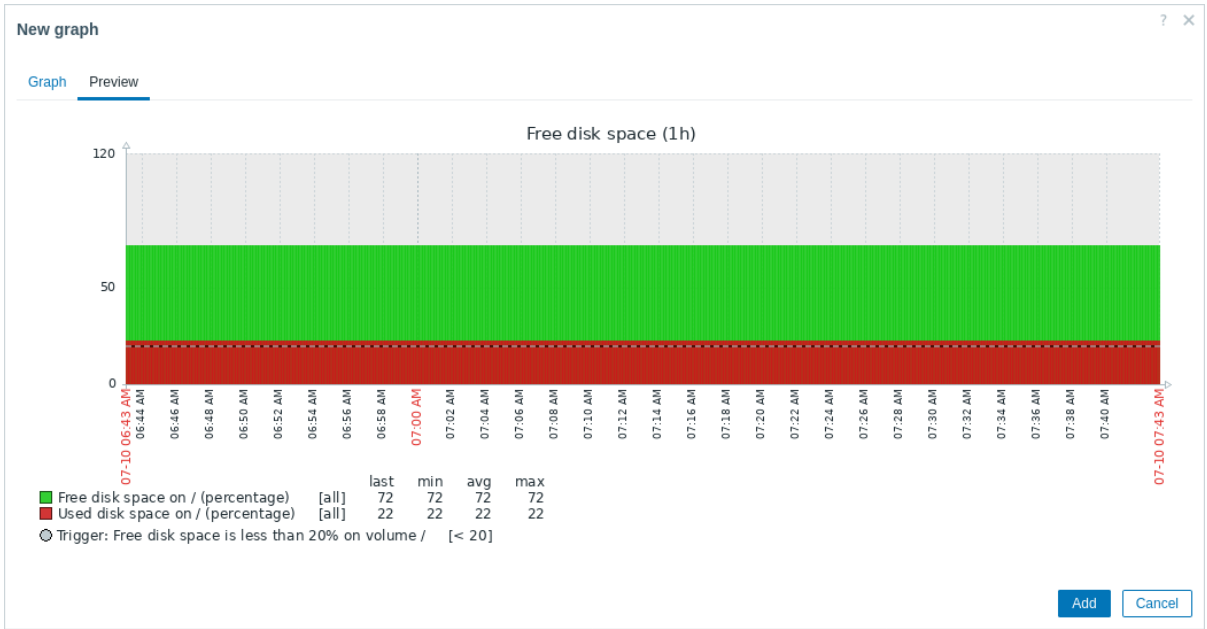
Parameter	Description
Y axis side	选择显示监控项数据的 Y 轴侧 - *Left*、*Right*。
Color	选择要应用于监控项数据的颜色。

图形预览

在 预览选项卡中，会显示图形的预览，这样您就可以立即看到正在创建的内容。



请注意，对于模板监控项，预览中不会显示任何数据。



在此示例中，请注意显示触发器级别的粗虚线，以及图例中显示的触发器信息。

Note:
最多只能显示 3 条触发器线。如果触发器数量更多，则会优先显示严重性较低的触发器。

 如果图形高度设置为小于 120 像素，则图例中不会显示任何触发器。

临时图形

概述

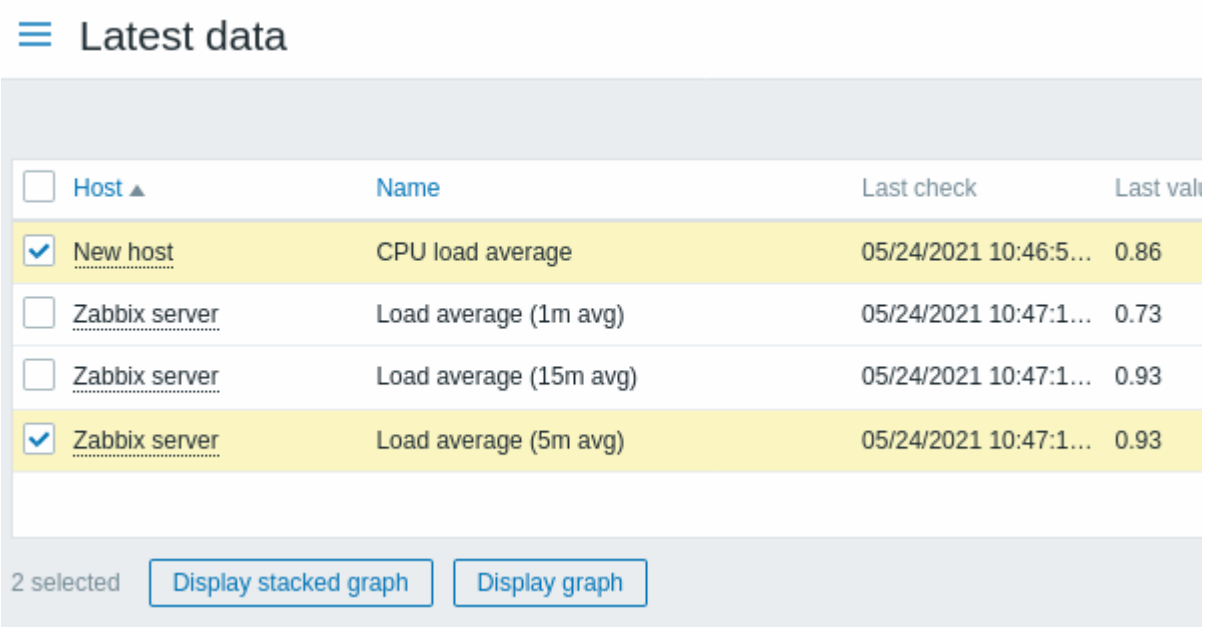
简单图形适用于查看单个监控项的数据，而自定义图形提供了自定义选项，但二者都无法以最少的操作和维护工作，快速创建多个监控项的对比图形。

为解决此问题，您可以通过非常快捷的方式为多个监控项创建临时图形。

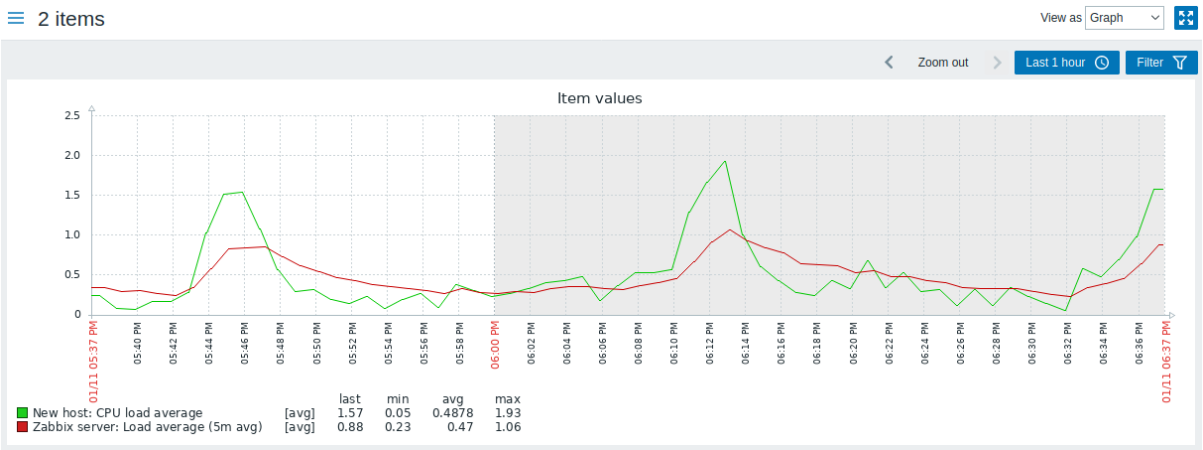
配置

要创建临时图形，请执行以下操作：

1. 转到 *Monitoring* > *Latest data*。
2. 使用筛选器显示您想要的监控项。
3. 勾选要绘制图形的监控项复选框。
4. 单击 *Display stacked graph* 或 *Display graph* 按钮。

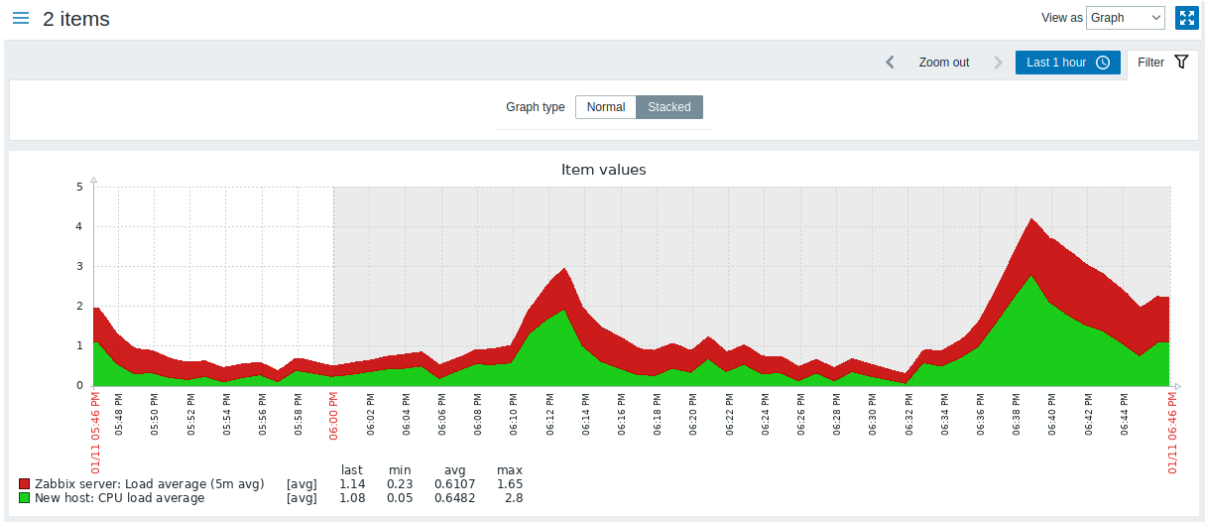


图形会立即创建：



请注意，为了避免在图形中显示过多线条，每个监控项只显示平均值（不显示最小/最大值线）。图形中不显示触发器和触发器信息。

在创建的图形窗口中，您可以使用时间段选择器，并且可以将“普通”折线图切换为堆叠图（也可以切换回来）。



网络地图

概述

如果你有一个网络需要管理，你可能想要对你的基础设施有一个全面的了解。为此，你可以在 Zabbix 中创建网络拓扑图以及任何你喜欢的图。

任何用户都可以创建网络拓扑图. 拓扑图可以是公开的 (所有用户可见) 或者私有的 (部分用户可见)。

请参考配置网络拓扑.

配置网络地图

概述

在 Zabbix 中配置地图时，首先需要通过定义其通用参数来创建一个地图，然后再开始用元素及其链接填充实际的地图。

你可以用以下元素来填充地图：主机、主机组、触发器、图像或其他地图。

图标用于表示地图元素。你可以定义随图标显示的信息，并设置最近问题以特殊方式显示。你还可以链接这些图标，并定义在链接上显示的信息。

你可以添加自定义 URL，以便通过单击图标进行访问。因此，你可以将主机图标链接到主机属性，或将地图图标链接到另一张地图。

地图中的问题计数仅显示根本原因问题。

地图在 监控 > 地图 中进行管理，可在此进行配置、管理和查看。在监控视图中，你可以单击图标，并利用指向某些脚本和 URL 的链接。

网络地图基于矢量图形（SVG）。

公共和私有拓扑图

Zabbix 中的所有用户（包括非管理员用户）都可以创建网络拓扑图。拓扑图有一个所有者——即创建它们的用户。拓扑图可以设为公共或私有。

- 公共拓扑图对所有用户可见，但要查看它们，用户必须对至少一个拓扑图元素具有读权限。公共拓扑图可以被编辑，前提是某个用户/用户组对此拓扑图具有读写权限，并且对相应拓扑图的所有元素（包括链接中的触发器）至少具有读权限。
- 私有拓扑图仅对其所有者以及所有者与之共享 的用户/用户组可见。普通（非超级管理员）用户只能与其所属的组以及属于这些组的用户共享。管理员级别用户无论是否为所有者或是否属于共享用户列表，都可以查看私有拓扑图。私有拓扑图可以由其所有者编辑；如果某个用户/用户组对此拓扑图具有读写权限，并且对相应拓扑图的所有元素（包括链接中的触发器）至少具有读权限，也可以进行编辑。

用户没有读权限的拓扑图元素会以灰显图标显示，并且该元素上的所有文本信息都会被隐藏。但是，即使用户对触发器没有权限，触发器标签仍然可见。

要向拓扑图添加元素，用户还必须对该元素至少具有读权限。

创建地图

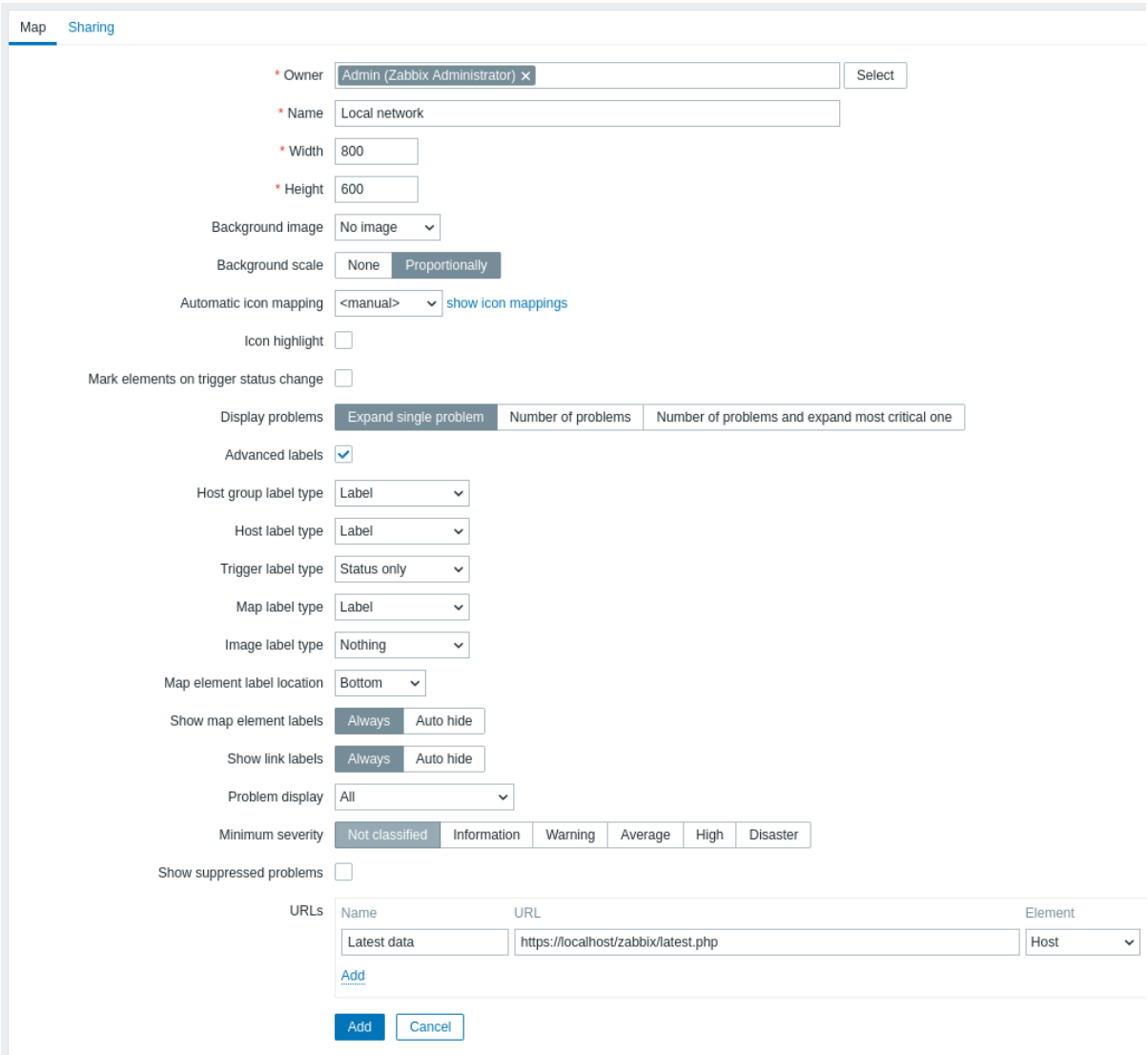
要创建地图，请执行以下操作：

1. 转到 监测 > 地图。
2. 转到包含所有地图的视图。

3. 单击 创建地图。
4. 配置地图详细信息。

您也可以在现有地图的配置表单中使用 克隆按钮来创建新地图。此地图将具有现有地图的所有属性，包括常规布局属性以及现有地图的元素。

地图选项卡包含常规地图属性：



所有必填输入字段均以红色星号标记。

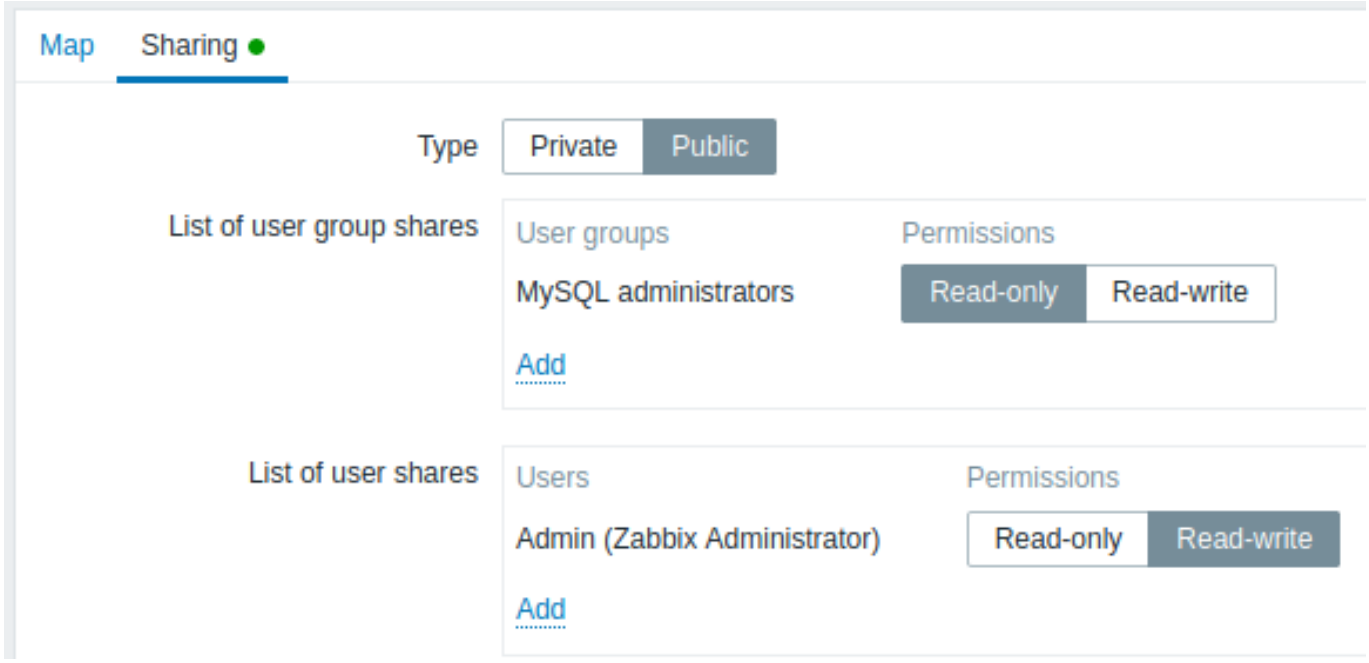
常规地图属性：

参数	描述
所有者	地图所有者的名称。
名称	唯一的地图名称。
宽度	地图宽度，单位为像素。
高度	地图高度，单位为像素。
背景图像	使用背景图像： 无图像 - 不使用背景图像（白色背景） 图像 - 选择要用作背景图像的图像。您可以使用地理地图或任何其他图像来增强地图效果。
背景缩放	使用背景缩放： 无 - 不进行缩放； 按比例 - 覆盖整个地图背景，同时保持图像比例（默认）。
自动图标映射	您可以设置使用在 管理 > 常规 > 图标映射中配置的自动图标映射。图标映射允许将特定图标与特定主机资产字段进行映射。

参数	描述
图标高亮	选中此复选框后，地图元素将突出显示。 具有活动触发器的元素将显示圆形背景，其颜色与最高严重性触发器的颜色相同。此外，如果所有问题都已确认，圆圈周围还会显示一条粗绿色线条。 状态为 “已禁用” 或 “维护中” 的元素将分别显示灰色和橙色方形背景。 另请参阅：查看地图
在触发器状态更改时标记元素	触发器状态的最近更改（最近发生的问题或恢复）将通过标记突出显示。标记显示在元素图标上没有标签的三个边上，形状为指向内侧的红色三角形。标记显示 30 分钟。
显示问题	选择如何通过地图元素显示问题： 展开单个问题 - 如果只有一个问题，则显示问题名称。否则，显示问题总数。 问题数量 - 显示问题总数 问题数量并展开最严重的问题 - 显示最严重问题的名称和问题总数。 “最严重” 根据问题严重性确定；如果严重性相同，则根据问题事件 ID 确定（ID 较高或较晚发生的问题优先显示）。对于触发器地图元素，则根据问题严重性确定；如果严重性相同，则根据触发器在触发器列表中的位置确定。如果同一触发器存在多个问题，则显示最近的问题。
高级标签	选中此复选框后，您可以为不同的元素类型定义单独的标签类型。
主机组标签类型	用于地图元素的标签类型： 标签 - 地图元素标签 **IP** 地址 - IP 地址 元素名称 - 元素名称（例如，主机名称） 仅状态 - 仅显示状态（正常或问题） 无 - 不显示标签
主机标签类型	
触发器标签类型	
地图标签类型	
图像标签类型	
地图元素标签位置	标签相对于地图元素的位置： 底部 - 位于地图元素下方 左侧 - 位于左侧 右侧 - 位于右侧 顶部 - 位于地图元素上方
显示地图元素标签	选择如何显示地图元素标签： 始终 - 始终显示（默认）； 自动隐藏 - 当标签未被悬停或选中时隐藏。
显示链接标签	选择如何显示链接标签： 始终 - 始终显示（默认）； 自动隐藏 - 当标签未被悬停或选中时隐藏。
问题显示	将问题数量显示为： 全部 - 显示完整的问题数量 分开 - 将未确认的问题数量作为问题总数中的一个单独数字显示 仅未确认 - 仅显示未确认的问题数量
最低严重性	低于所选最低严重性级别的问题不会显示在地图上。 例如，选择警告后，具有信息和未分类级别触发器的更改不会反映在地图中。
显示被抑制的问题	选中此复选框，以显示原本会因主机维护而被抑制（不显示）的问题。
URL	可以在此处为每种元素类型定义 URL，其中： 名称 - URL 的标签； **URL** - URL（最多 2048 个字符）； 元素 - 元素类型（例如，主机、主机组等）。 用户在地图查看模式下单击元素时，这些 URL 将显示为链接。 地图 URL 名称和值中可以使用宏。完整列表请参阅支持的宏，并搜索 “map URL names and values”。

共享

共享选项卡包含地图类型以及私有地图的共享选项（用户组、用户）：



Parameter	Description
Type	选择地图类型： **Private** - 地图仅对所选用户组和用户可见 **Public** - 地图对所有人可见
List of user group shares	选择该地图可访问的用户组。 您可以允许只读或读写访问。
List of user shares	选择该地图可访问的用户。 您可以允许只读或读写访问。

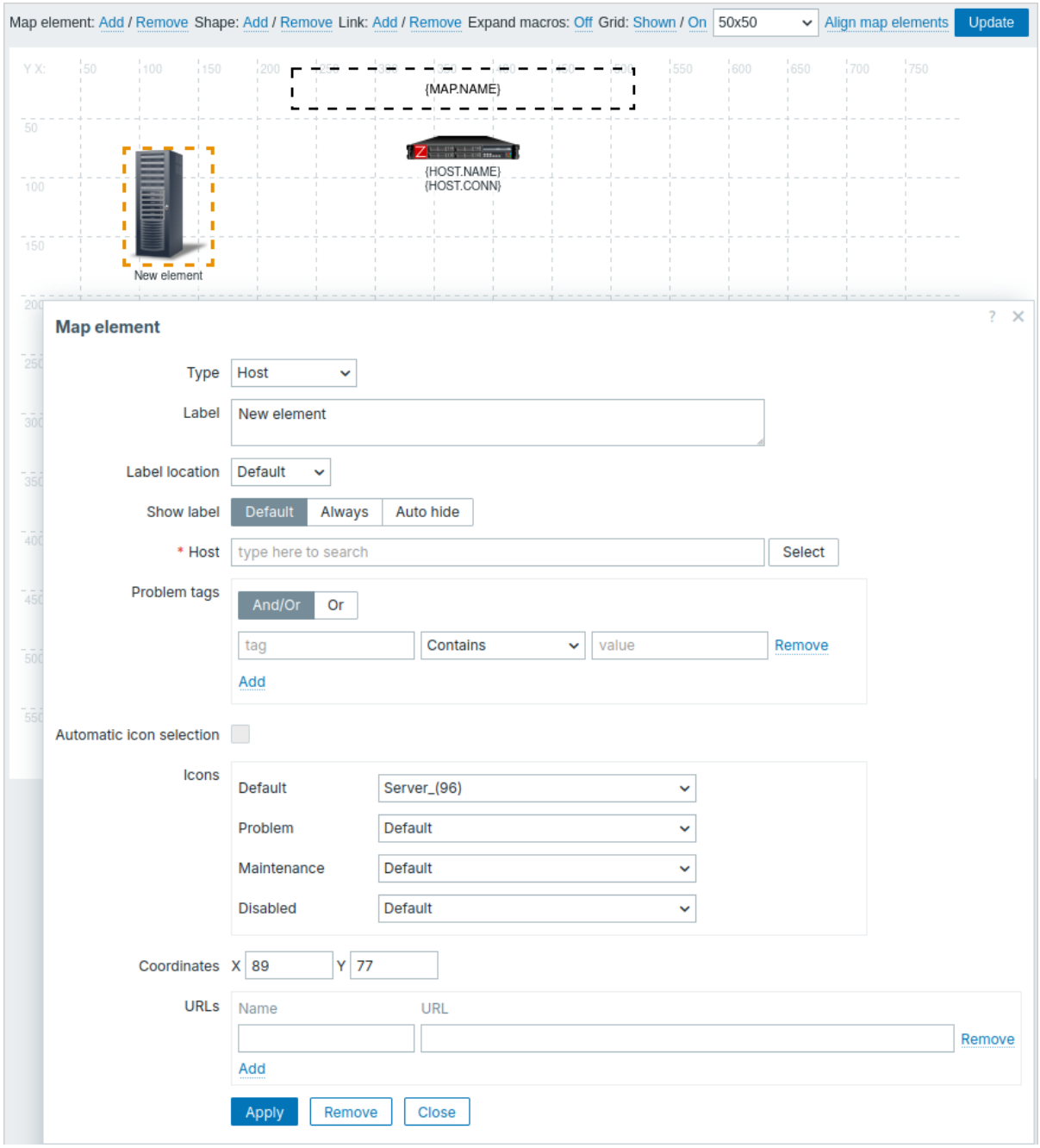
当您点击 *Add* 保存此地图时，您就创建了一个空地图，其中包含名称、尺寸和某些首选项。现在您需要添加一些元素。为此，请在地图列表中点击 *Edit* 以打开可编辑区域。

添加元素

要添加一个元素，请点击 `Map element` 旁边的 *Add*。新元素将出现在地图的左上角。将其拖放到你喜欢的任何位置。

请注意，当 Grid 选项为"On" 时，元素将始终与网格对齐（你可以从下拉菜单中选择不同的网格大小，也可以隐藏/显示网格）。如果你希望将元素放置在任意位置而不对齐，请将该选项设为"Off"。（之后你仍然可以通过点击 *Align map elements* 将随机放置的元素与网格对齐。）

现在你已经放置了一些元素，可能希望通过为它们命名等方式来区分它们。点击元素后会显示一个表单，你可以设置元素类型、指定名称、选择不同的图标等。



地图元素属性：

Parameter	Description
Type	元素类型： **Host** - 表示所选主机所有触发器状态的图标 **Map** - 表示地图中所有元素状态的图标 **Trigger** - 表示一个或多个触发器状态的图标 **Host group** - 表示所选组中所有主机的所有触发器状态的图标 **Image** - 不关联任何资源的图标
Label	图标标签，任意字符串。 可使用宏和多行字符串。 此字段支持表达式macros，但仅支持带时间参数的 `avg`、`last`、`min` 和 `max` 函数（例如，`{?avg(/host/key,1h)}`）。 有关支持的宏完整列表，请参见supported macros，并搜索'map element labels'。
Label location	标签相对于图标的位置： **Default** - 地图的默认标签位置； **Bottom** - 位于图标下方； **Left** - 位于左侧； **Right** - 位于右侧； **Top** - 位于图标上方。

Parameter	Description
Show label	选择元素标签的显示方式： **Default** - 使用地图配置中的 *Show map element labels* 设置（默认）； **Always** - 始终显示（默认）； **Auto hide** - 当未悬停或未选中时隐藏标签。
Host	选择一个主机；或者输入主机名。此字段支持自动补全，因此开始输入主机名时会显示匹配主机的下拉列表。向下滚动以选择一个。点击删除（x）图标可移除所选项。 此字段仅适用于类型为 *Host* 的元素。
Map	选择一个地图；或者输入地图名称。此字段支持自动补全，因此开始输入地图名称时会显示匹配地图的下拉列表。向下滚动以选择一个。点击删除（x）图标可移除所选项。 此字段仅适用于类型为 *Map* 的元素。
Triggers	在下面的 *New triggers* 字段中选择一个或多个触发器，然后点击 *Add*。或者，开始输入触发器名称，并从匹配触发器的下拉列表中选择一个。点击删除（x）图标可移除所选项。 此字段仅适用于类型为 *Trigger* 的元素。 所选触发器的顺序可以更改，但仅限于相同严重级别的触发器。多触发器选择还会影响编辑模式和查看模式下 {HOST.*} 宏的解析。 在编辑模式下，首先显示的 {HOST.*} 宏将根据列表中的第一个触发器进行解析（基于触发器严重级别）。 查看模式取决于常规地图属性中的Display problems 参数： - 如果选择 *Expand single problem* 模式，首先显示的 {HOST.*} 宏将根据最近检测到的问题触发器进行解析（不考虑严重级别）；如果未检测到问题，则根据列表中的第一个触发器进行解析。 - 如果选择 *Number of problems and expand most critical one* 模式，首先显示的 {HOST.*} 宏将根据触发器严重级别进行解析。
Host group	选择一个主机组；或者输入主机组名称。此字段支持自动补全，因此开始输入主机组名称时会显示匹配主机组的下拉列表。向下滚动以选择一个。点击删除（x）图标可移除所选项。 此字段仅适用于类型为 *Host group* 的元素。
Problem tags	指定标签以限制小组件中显示的问题数量。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含标签值中含有输入字符串的指定标签名称（子串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除标签值中含有输入字符串的指定标签名称（子串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足任一条件即可 此字段仅适用于类型为 *Host* 和 *Host group* 的元素。
Automatic icon selection	勾选后，将使用图标映射来确定显示哪个图标。
Icons	你可以为元素在以下情况下选择显示不同的图标：*Default*、*Problem*、*Maintenance* 和 *Disabled*。 此字段仅适用于类型为 *Host* 的元素。
Coordinates	地图元素的 **X** 和 **Y** 坐标。
URLs	可在此为元素设置特定 URL，其中： **Name** - URL 的标签； **URL** - 一个 URL（最多 2048 个字符）。 当用户在地图查看模式下点击元素时，这些内容将作为链接显示。如果该元素有自己的 URL，并且已为其类型定义了地图级 URL，则它们会合并到同一菜单中。 可在地图元素名称和值中使用宏。完整列表请参见supported macros，并搜索'map URL names and values'。

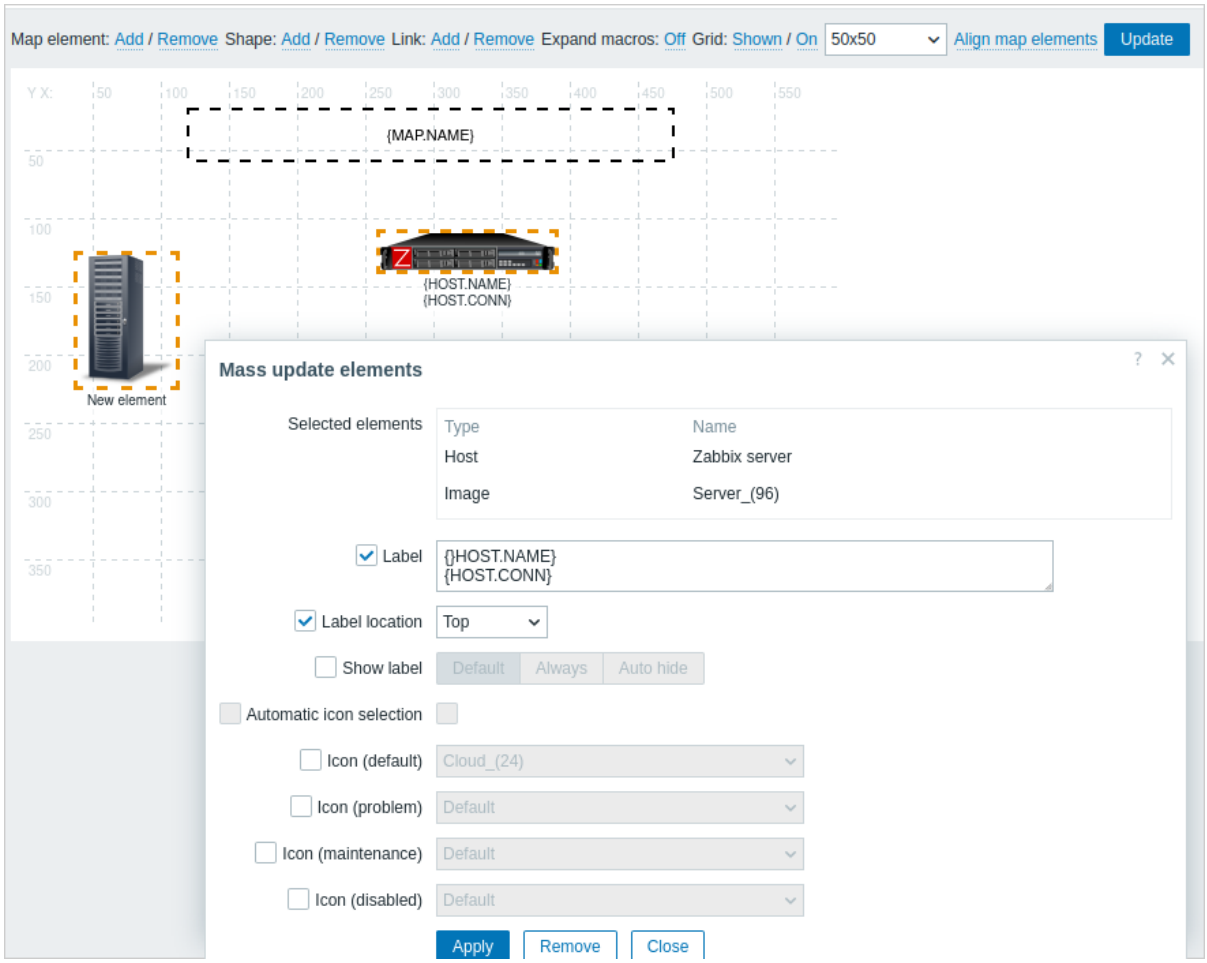
Attention:
新增的元素不会自动保存。如果你离开此页面，所有更改都可能丢失。
因此，建议点击右上角的 **Update** 按钮。点击后，无论你在随后弹出的窗口中选择什么，修改都会被保存。
所选的网格选项也会随每张地图一起保存。

选择元素

要选择多个元素，请选择一个，然后按住 *Ctrl* 选择其他元素。

您也可以通过在可编辑区域拖动一个矩形并选择其中的所有元素来选择多个元素。

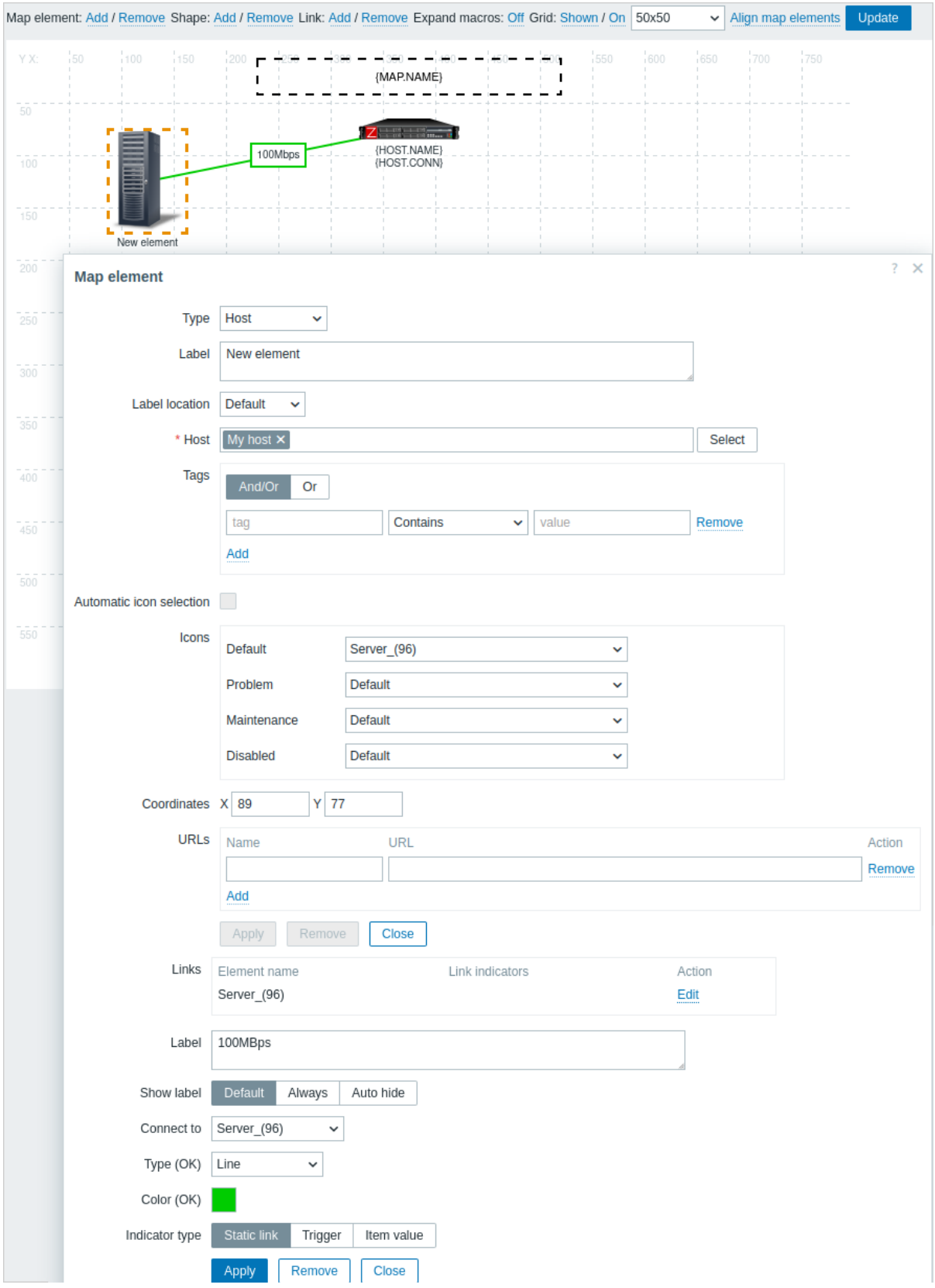
一旦选择了多个元素，元素属性形式就会发生变化到批量更新模式，以便您可以一次性更改选定的属性元素。为此，请使用复选框标记属性，然后为其输入一个新值。您可以在此处使用宏（例如，下图标签中使用的 host.name 宏）。



关联元素

在地图上放置一些元素后，就可以开始将它们关联起来了。要关联两个元素，必须先选中它们。选中元素后，点击 Link 旁边的 *Add*。

创建关联后，单个元素表单现在会包含一个额外的 *Links* 部分。点击 *Edit* 可编辑关联属性。



关联属性：

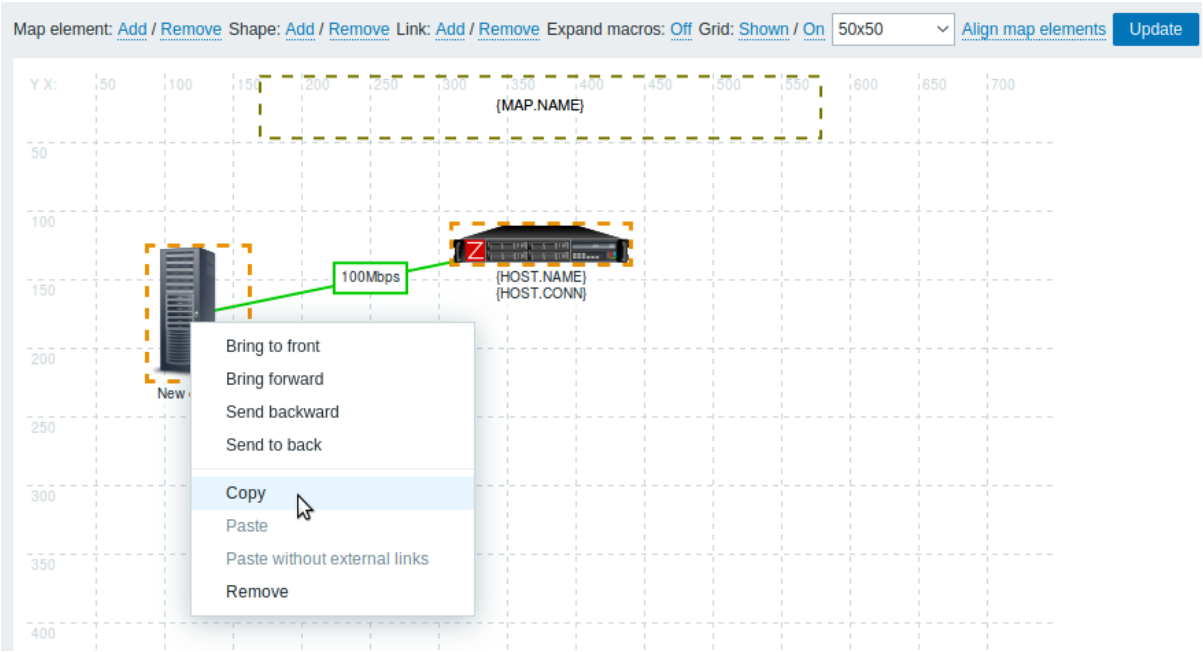
Parameter	Description
Label	显示在关联上方的标签。 此字段支持表达式macros，但仅支持带时间参数的 `avg`、`last`、`min` 和 `max` 函数（例如，`{?avg(/host/key,1h)}`）。
Show label	选择如何显示关联标签： **Default** - 使用地图配置中的 *Show map element labels* 设置（默认）； **Always** - 始终显示（默认）； **Auto hide** - 如果未悬停或未选中，则隐藏标签。

Parameter	Description
Connect to	该关联连接到的元素。
Type (OK)	默认关联样式。选择样式：*Line*、*Bold line*、*Dot* 或 *Dashed line*。
Color (OK)	默认关联颜色。点击颜色样本可选择其他颜色。
Indicator type	选择关联指示器类型： **Static link** - 无指示器； **Trigger** - 允许基于触发器的关联指示器； **Item value** - 允许基于监控项值的关联指示器。
Item	选择监控项。此监控项的值将影响关联样式。 当 *Indicator type* 选择 *Item value* 时，此字段可用。
Indicators	关联指示器列表。 当 *Indicator type* 设置为 *Trigger* 或 *Item value* 时，此字段可用。 如果设置为 *Trigger*，则可以将触发器分配为关联指示器。当触发器处于 *Problem* 状态时，其样式（所选颜色和线型）将应用于该关联。 如果设置为 *Item value* 且在 *Item* 字段中选择了监控项，则可以将监控项值分配为关联指示器。当监控项值达到指定阈值（对于数值监控项）或匹配模式（对于文本监控项）时，其样式（所选颜色和线型）将应用于该关联。

移动和复制粘贴元素

可以通过单击任一已选中的元素，按住鼠标按钮，并将光标移动到所需位置，将多个已选中的元素移动到地图中的其他位置。

可以通过选择这些元素，然后用鼠标右键单击一个已选中的元素，并从菜单中选择 *Copy*，来复制一个或多个元素。

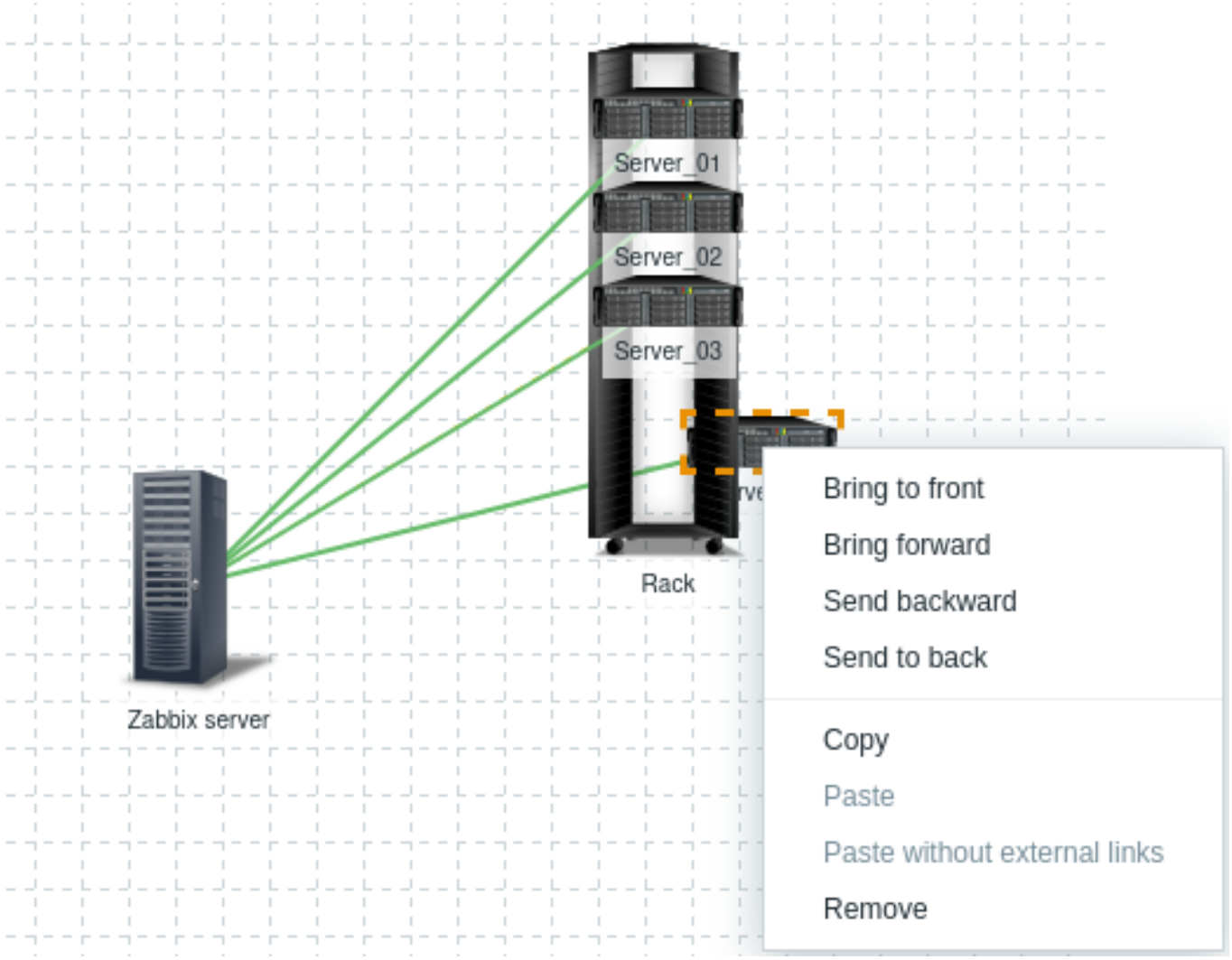


要粘贴这些元素，请用鼠标右键单击地图区域，并从菜单中选择 *Paste*。*Paste without external links* 选项会粘贴这些元素，但仅保留所选元素之间的链接。

复制粘贴仅在同一个浏览器窗口内有效。不支持键盘快捷键。

元素排序

要将一个元素置于另一个元素前面（或反之），请用鼠标右键单击该元素，然后选择 前移一层/置于顶层或 后移一层/置于底层。

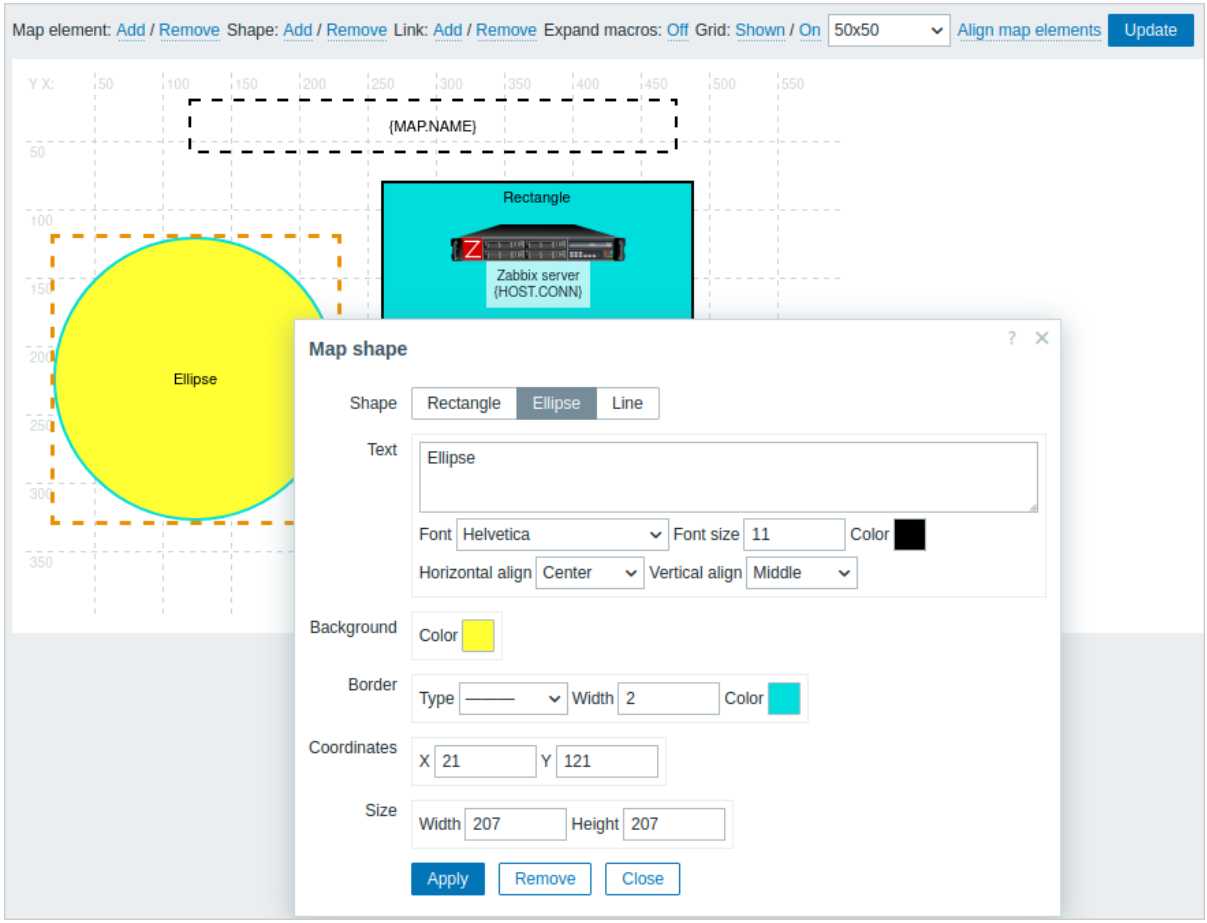


添加形状

除了地图元素之外，还可以添加一些形状。形状不是地图元素；它们只是视觉表示。例如，矩形形状可用作背景，以分组一些主机。可以添加矩形和椭圆形状。

要添加形状，请单击 Shape 旁边的 *Add*。新形状将出现在地图的左上角。将其拖放到任意位置即可。

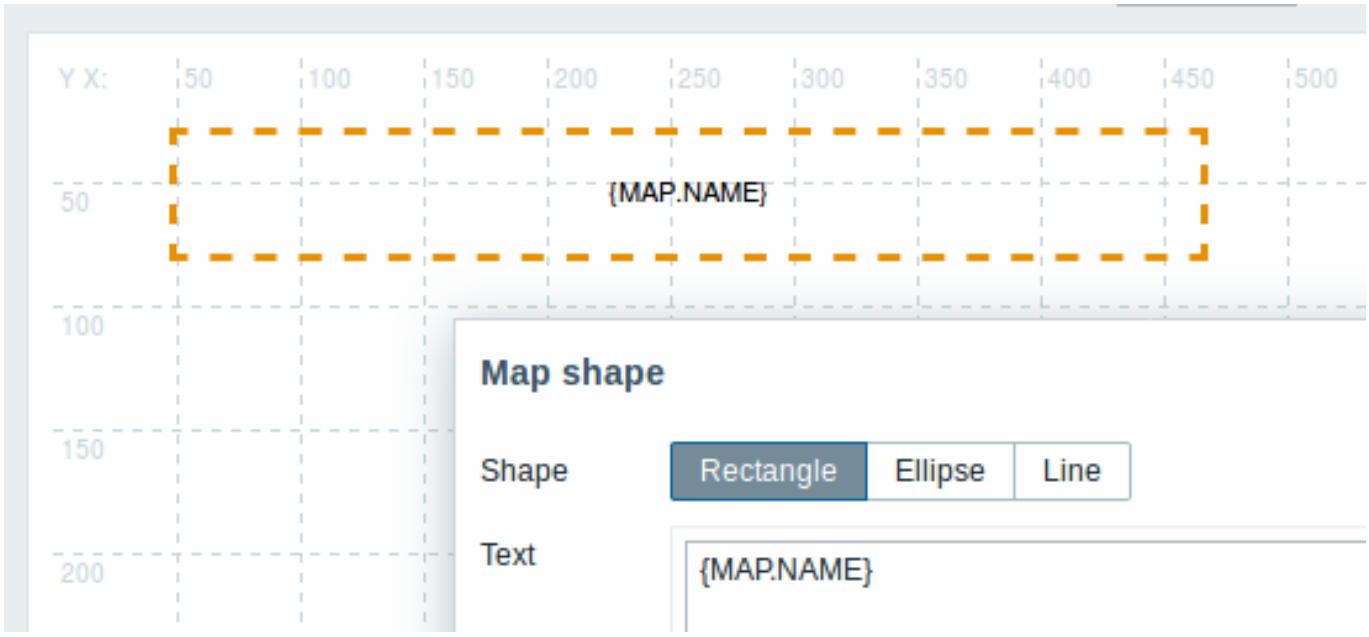
新形状会以默认颜色添加。单击该形状后，会显示一个表单，您可以自定义形状的外观、添加文本等。



要选择多个形状，先选中一个，然后按住 *Ctrl* 再选择其他形状。选中多个形状后，可以像处理元素一样批量更新公共属性。

可以在形状中添加文本。文本中支持表达式macros，但仅支持带时间参数的 `avg`、`last`、`min` 和 `max` 函数（例如，`{?avg(/host/key,1h)}`）。

如果只想显示文本，可以通过移除形状边框来使形状不可见（在 *Border* 字段中选择'None'）。例如，请注意上方截图中可见的 {MAP.NAME} 宏实际上是一个带文本的矩形形状，单击该宏即可看到这一点：



在查看地图时，{MAP.NAME} 会解析为已配置的地图名称。

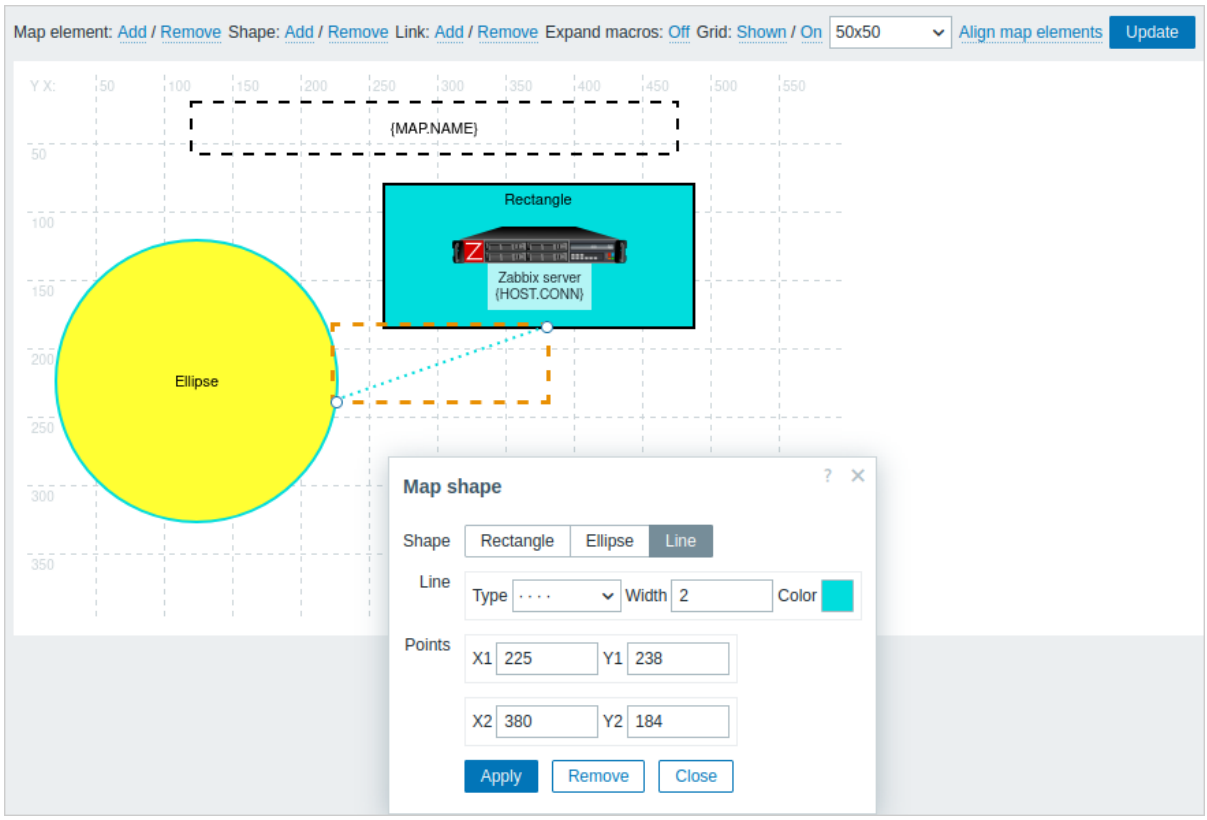
如果文本中使用了超链接，在查看地图时它们会变为可点击。

在形状内，文本换行始终处于 "开启" 状态。不过，在椭圆形中，换行方式会像椭圆是矩形一样进行处理。当前未实现按词换行，因此较长的单词（无法放入形状中的单词）不会换行，而是会被遮罩（在地图编辑页面）或裁剪（其他带地图的页面）。

添加线条

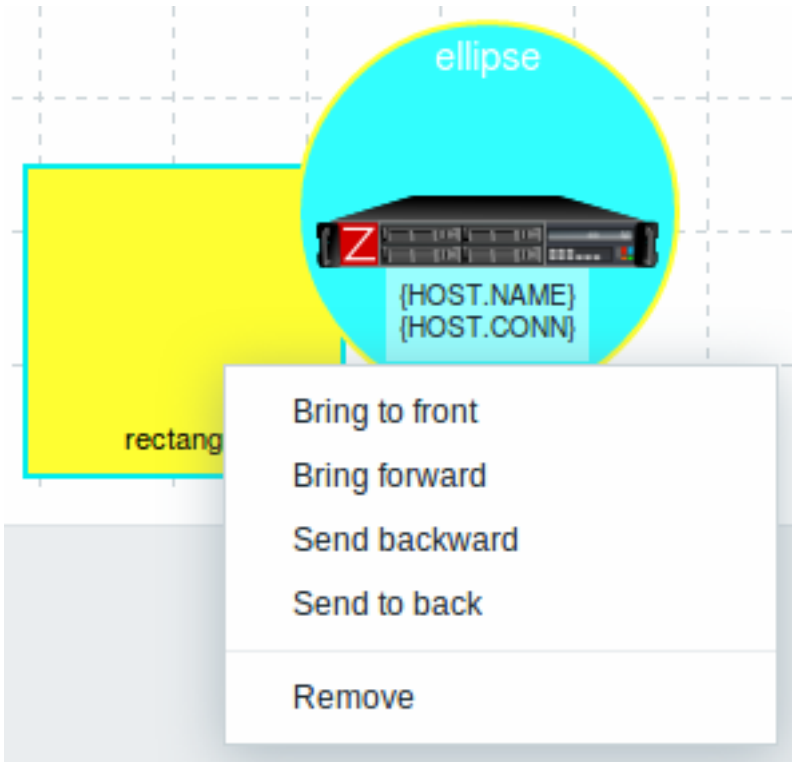
除了形状之外，还可以添加一些线条。线条可用于在地图中链接元素或形状。

要添加线条，请单击 Shape 旁边的 *Add*。一个新的形状将出现在地图的左上角。选中它，然后在编辑表单中单击 *Line*，将该形状更改为线条。接着调整线条属性，例如线条类型、宽度、颜色等。



排列形状和线条

要将一个形状置于另一个形状前面（或反之），请用鼠标右键单击该形状以打开地图形状菜单。

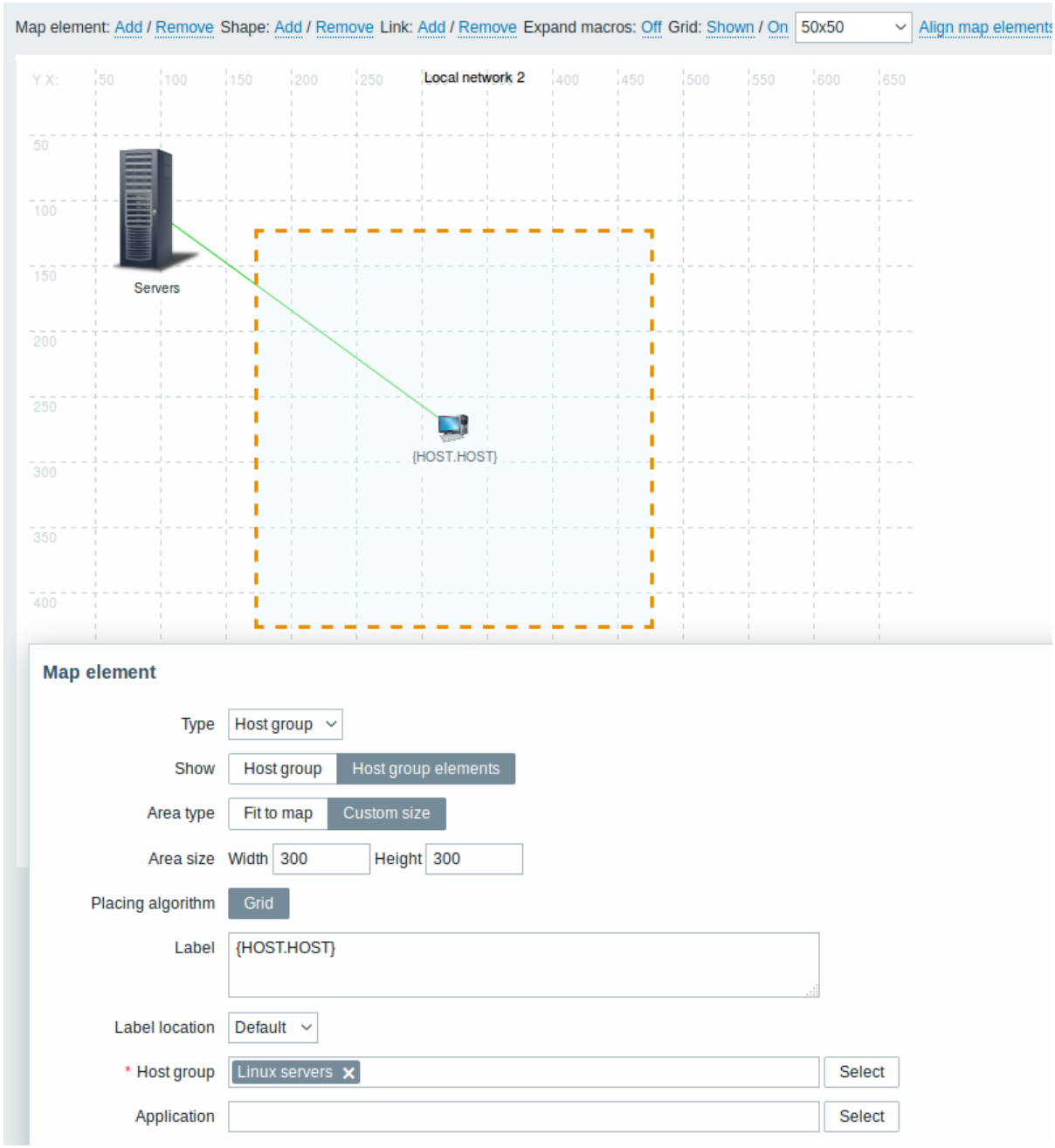


主机组元素

概述

本节介绍在配置网络拓扑图时，如何添加主机群组类型的元素。

配置



所有必填输入字段均以红色星号标记。

此表包含 主机组元素类型的典型参数：

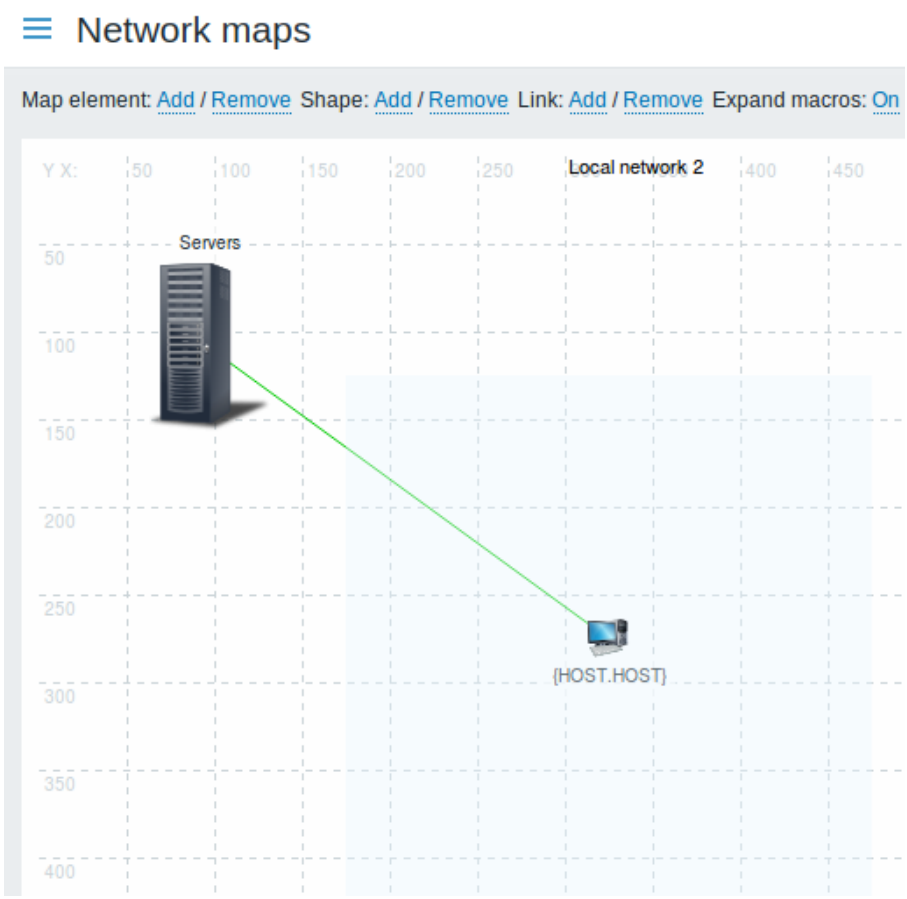
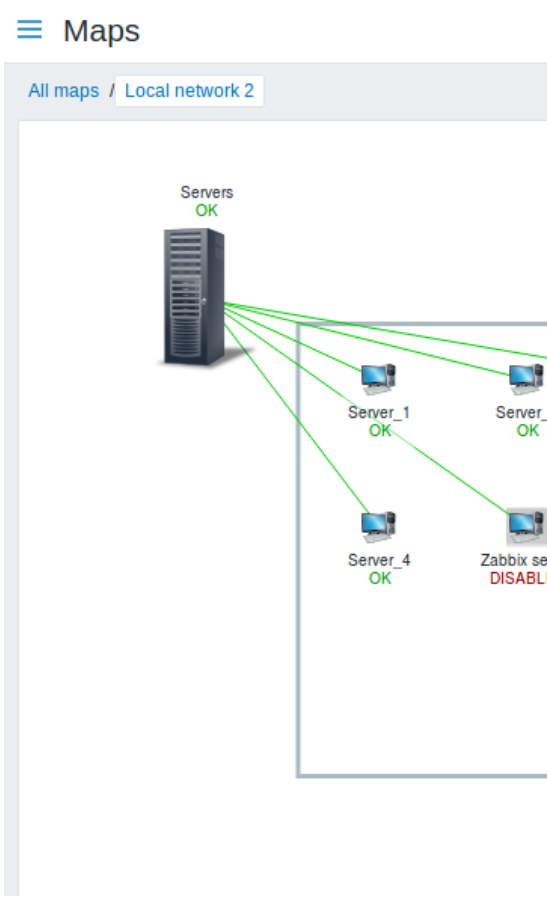
参数	描述
类型	选择元素的类型： 主机组 - 表示所选组中所有主机的所有触发器状态的图标。
显示	显示选项： 主机组 - 选择此选项后，将显示一个图标，其中包含有关特定主机组的相应信息。 主机组元素 - 选择此选项后，将显示多个图标，其中每个图标包含有关特定主机组中每个元素（主机）的相应信息。
区域类型	当 显示设置为 主机组元素时，此设置可用： 适应地图 - 所有主机组元素将在地图中均匀放置。 自定义大小 - 手动设置用于显示所有主机组元素的地图区域。
区域大小	当 区域类型设置为 自定义大小时，此设置可用： 宽度 - 以像素为单位指定地图区域宽度。 高度 - 以像素为单位指定地图区域高度。
放置算法	网格 - 显示所有主机组元素时唯一可用的选项。

参数	描述
标签	图标标签，可以是任意字符串。 标签中可以使用宏和多行字符串。 如果地图元素的类型为 主机组，指定特定宏会影响地图视图，使其显示有关每个主机的相应信息。例如，如果使用{HOST.IP}宏，编辑地图视图中将仅显示宏 {HOST.IP} 本身，而地图视图将包含并显示每个主机的唯一 IP 地址。

查看主机组元素

如果选择了 主机组元素显示选项，则可以使用此选项。

将 主机组元素选作 显示选项时，最初只能看到代表该主机组的一个图标。
但是，保存地图并转到地图视图后，您会看到地图包含指定主机组的所有元素（主机）：

地图编辑视图	地图视图
	

请注意{HOST.HOST}宏的使用方式。
在地图编辑视图中，宏名称尚未解析；而在地图视图中，会显示所有主机的唯一名称。

链接指示器

概述

您可以为网络地图中元素之间的链接分配指示器。

这些指示器可以基于触发器或监控项值。可以显示不同的链接样式和颜色：

- 当触发器进入问题状态时；
- 当监控项值：
  - 达到阈值时（对于数值型监控项）；
  - 匹配正则表达式时（对于文本型监控项）。

配置链接时，您需要设置默认的链接类型和颜色。通过为链接分配指示器，
可以使链接样式和颜色取决于触发器状态或监控项值。

例如，如果任何已分配的触发器进入问题状态，链接样式和颜色将发生变化以反映该状态。因此，您的默认链接可能是一条绿色线条。现在，当触发器处于问题状态时，您的链接可能会变为粗红线（如果您已这样定义）。

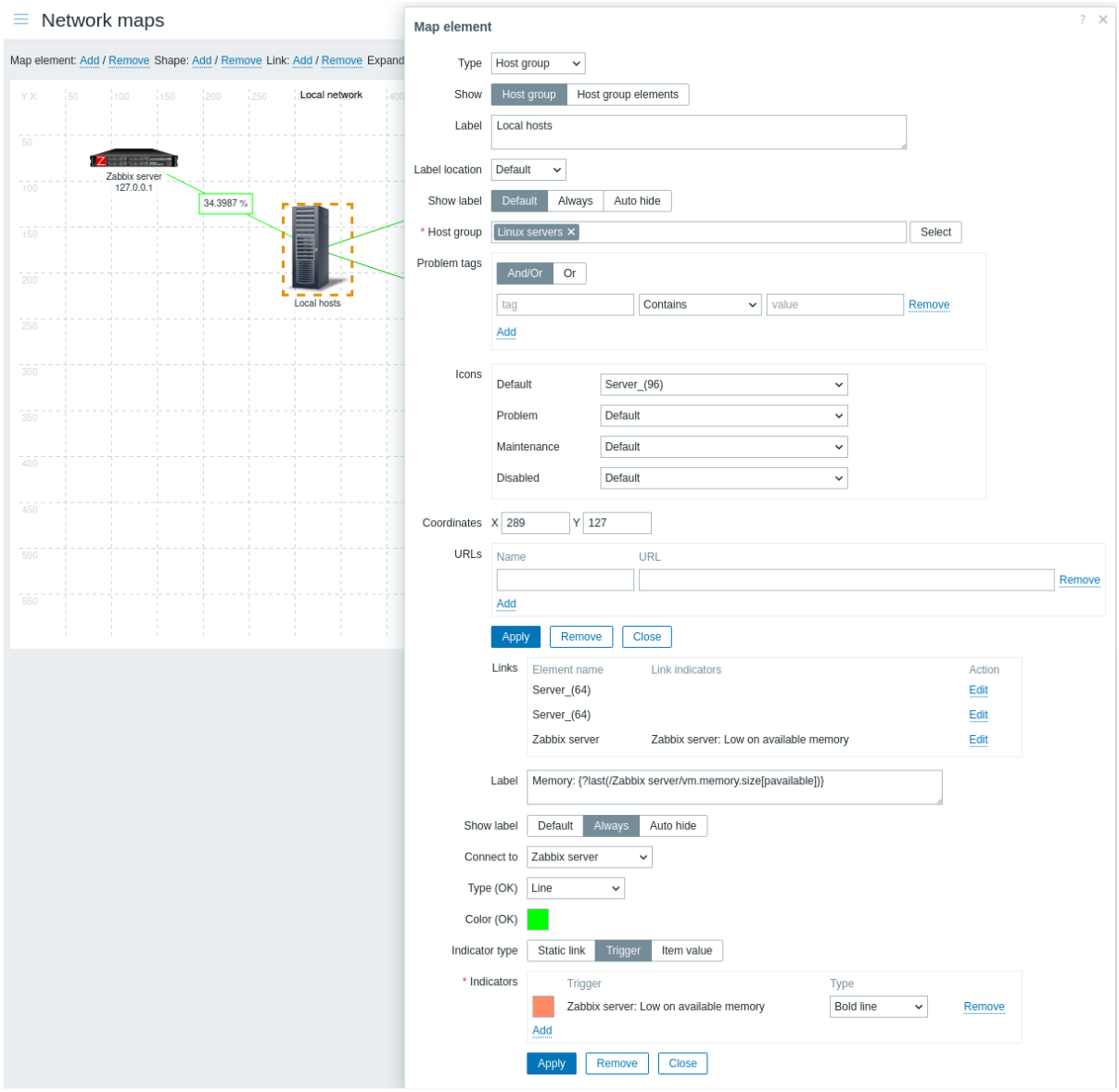
同样，如果监控项值达到指定阈值或匹配指定的正则表达式，链接样式也可以反映这一点。

配置

触发器

要将触发器作为链接指示器，请执行以下操作：

- 选择一个地图元素
- 在相应链接的 *Links* 部分点击 *Edit*
- 将指示器类型选择为 *Trigger*
- 在 *Indicators* 区块中点击 *Add*，并选择一个或多个触发器

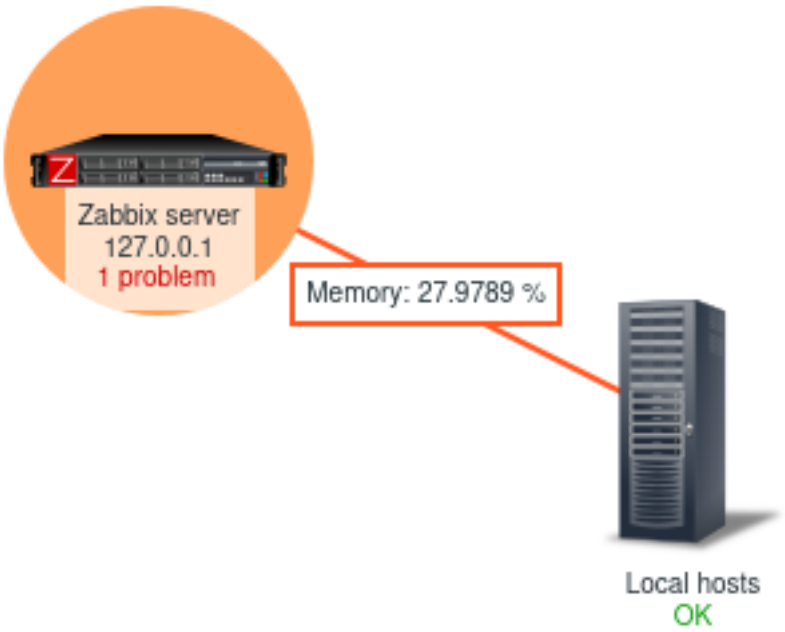


所有必填输入字段都用红色星号标记。

已添加的触发器可在 *Indicators* 列表中查看。

您可以直接在列表中为每个指示器设置链接类型和颜色。完成后，点击 *Apply*，关闭表单，然后点击 *Update* 以保存地图更改。

在 *Monitoring* > *Maps* 中，如果 **trigger** 进入问题状态，将显示相应的链接类型和颜色。



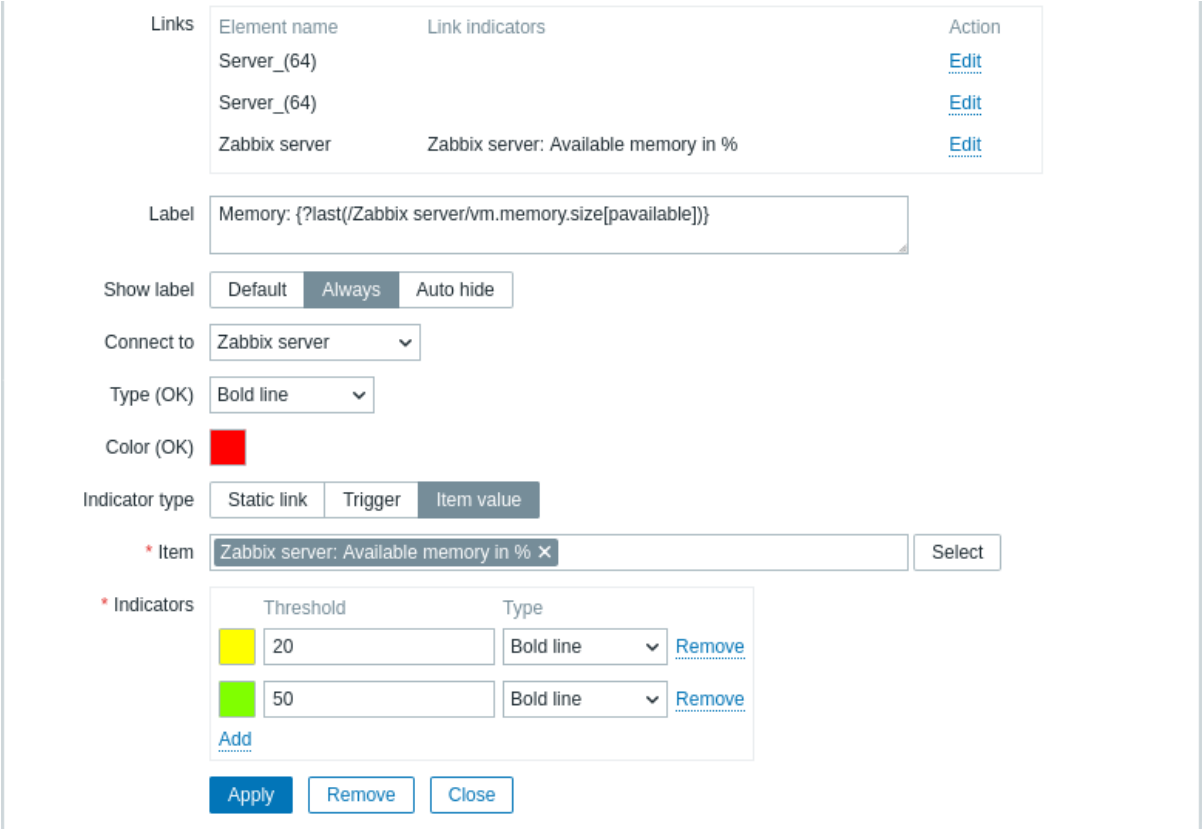
Note:
如果多个触发器进入问题状态，则严重性最高的问题将决定链接样式和颜色。如果多个具有相同严重性的触发器分配给同一个地图链接，则 ID 最低的那个优先。另请注意：

1. 地图配置中的 *Minimum trigger severity* 和 *Show suppressed problem* 设置会影响哪些问题会被纳入考虑。
2. 对于具有多个问题的触发器（多问题生成），每个问题的严重性可能与触发器严重性不同（可手动更改），也可能具有不同的标签（由于宏），并且可能被抑制。

监控项值

要将监控项值作为链接指示器，请执行以下操作：

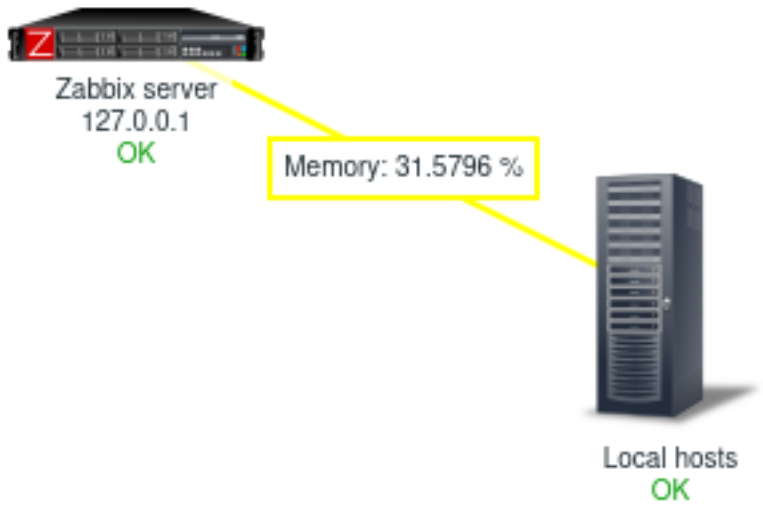
- 选择一个地图元素
- 在相应链接的 *Links* 部分点击 *Edit*
- 将指示器类型选择为 *Item value*
- 选择该监控项
- 在 *Indicators* 区块中添加一个或多个监控项值阈值或模式



已添加的监控项阈值/模式可在 *Indicators* 列表中查看。

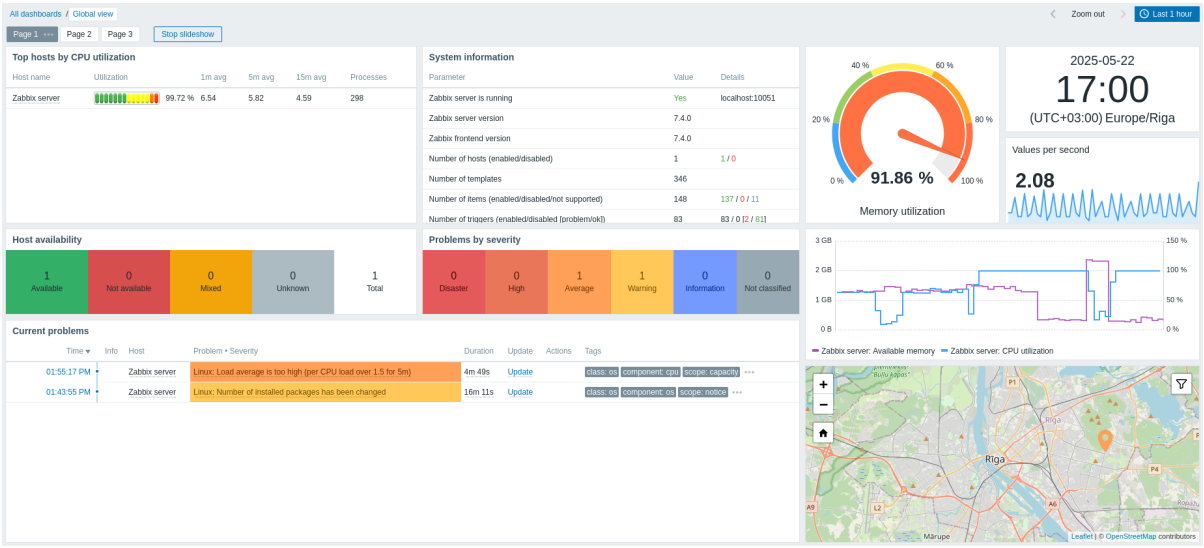
您可以直接在列表中为每个指示器设置链接类型和颜色。完成后，点击 *Apply*，关闭表单，然后点击 *Update* 以保存地图更改。

在 *Monitoring* > *Maps* 中，如果 **item value** 达到设置的阈值（适用于数值数据类型）或匹配正则表达式模式（适用于文本数据类型），将显示相应的链接类型和颜色。



仪表板

仪表板 - 包括全局仪表板和主机仪表板 - 通过现代图形、地图、幻灯片放映等小部件和工具，提供强大的可视化平台。



模板和模板组

概述

使用模板是减少工作量并简化 Zabbix 配置的绝佳方式。模板是一组可以方便地应用到多个主机的实体。

这些实体可以是：

- 监控项
- 触发器
- 图形
- 仪表板
- 低级别发现规则
- Web 场景

由于现实中许多主机是相同或相当相似的，因此很自然地，你为一台主机创建的一组实体（监控项、触发器、图形……）也可能对许多其他主机有用。当然，你可以将它们复制到每一台新主机上，但那将需要大量手动工作。相反，使用模板时，你可以先将它们复制到一个模板中，然后根据需要将该模板应用到任意数量的主机。

当模板链接到某台主机时，模板中的所有实体（监控项、触发器、图形……）都会被添加到该主机。模板是直接分配给每台单独的主机的（而不是分配给主机组）。

模板通常用于将特定服务或应用程序（如 Apache、MySQL、PostgreSQL、Postfix 等）的实体分组，然后应用到运行这些服务的主机上。

使用模板的另一个好处是，当需要对所有主机进行某项更改时，只需在模板级别更改一次，该更改就会传播到所有已链接的主机。

模板按模板组进行组织。

继续阅读创建和配置模板。

开箱即用的模板

概述

Zabbix 提供了不断扩充的预配置模板，用于简化并加快监控目标的设置。

所有开箱即用的模板都可在数据采集 > 模板中找到。

升级 Zabbix 时，模板不会自动升级。这样可以避免覆盖您所做的任何自定义更改。如需手动升级或添加模板，请参阅下面的模板升级部分。

请使用侧边栏访问有关特定模板类型和操作要求的信息。

专为与现有 Zabbix 模板配合使用而设计的全局仪表板可从仪表板存储库下载，并导入到您的 Zabbix 环境中。每个仪表板的详细说明（包括前提条件）可在该仪表板的 README 文件中找到。

另请参阅：

- 链接模板
- 模板的已知问题

模板升级

要升级或添加单个模板：

1. 在 Zabbix Git 仓库中打开 `templates` 目录。

2. 选择与您的 Zabbix 版本匹配的分支（位于 *Source* 标签下），例如，开发版本选择 *master*。

3. 找到要添加或升级的模板（例如，MySQL by Zabbix agent）。

4. 复制模板文件的内容（例如，`template_db_mysql_agent.yaml`），并将其保存到本地文件。

5. 在 Zabbix 用户界面中，转到数据采集 > 模板，然后导入本地文件。

> **Note:**
> Webhook 媒体类型也可以采用类似方式升级：从 `media` 目录中获取它们，然后在告警 > 媒体类型中导入。

要升级或添加多个模板：

1. 根据您的 Web 服务器下载模板包存档：

   - Nginx - `templates_bundles-8.0-1M-nginx.zip`
   - Apache - `templates_bundles-8.0-2M-apache.zip`

2. 解压存档。存档中包含多个 YAML 文件，每个文件包含多个模板；还包含一个索引文件，用于列出每个 YAML 文件中的模板。

3. 在 Zabbix 用户界面中，转到数据采集 > 模板，然后逐个导入 YAML 文件。

> **Note:**
> Webhook 媒体类型也可以采用类似方式升级：从模板包存档中获取 `media_bundle-8.0-000.yaml` 文件，然后在告警 > 媒体类型中导入。

要一次性升级或添加所有模板，可以使用模板包存档中包含的 `import_templates.sh` 脚本。

该脚本需要 `curl` 和 `jq` 工具，并使用 Zabbix API 的`configuration.import` 方法。

运行脚本时，请提供包含 YAML 文件的目录、Zabbix API URL 和身份验证令牌。

> **Warning:**
> `import_templates.sh` 脚本会覆盖所有现有模板，且无法查看或撤销模板更改。

示例：

```
/tmp/templates_bundles-8.0-2M-apache/import_templates.sh \
  /tmp/templates_bundles-8.0-2M-apache \
  https://example.com/zabbix/api_jsonrpc.php \
  d8d6b5c78ee2a8333db11bfcd3e448c49c239d8ce00702759e2b56f42e24dbe4
```

与主机向导的模板兼容性

主机向导与所有开箱即用的模板兼容。

您也可以使自定义模板与主机向导兼容：

1. 在 Zabbix 前端中，转到数据采集 > 模板，然后导出自定义模板。

2. 编辑导出的文件：

   - 添加 `wizard_ready` 模板元素，以表示该模板与主机向导兼容。
   - 根据需要，可选添加 `readme` 或 `config` 模板元素。

3. 在 Zabbix 前端中，转到数据采集 > 模板，然后导入更新后的模板文件。

Zabbix agent 模板操作

请执行以下步骤，确保使用Zabbix agent 收集指标的模板正常运行：

1. 确保主机上已安装 Zabbix agent。对于主动检查，还需确保已将 Zabbix 服务器/proxy 地址添加到agent 配置文件 的 `ServerActive` 参数中。

2. 将模板链接到目标主机（如果您的 Zabbix 安装中没有该模板，可能需要先导入模板）。
3. 如有必要，调整模板宏的值。
4. 配置被监控的实例，使其允许与 Zabbix 共享数据。

模板的详细说明，包括宏、监控项和触发器的完整列表，可在模板的 README 文件中找到（点击模板名称即可访问）。

以下模板可用：

- Apache by Zabbix agent
- Apache by Zabbix agent active
- HAProxy by Zabbix agent
- IIS by Zabbix agent
- IIS by Zabbix agent active
- MariaDB by Zabbix agent
- MariaDB by Zabbix agent active
- Microsoft Exchange Server 2016 by Zabbix agent
- Microsoft Exchange Server 2016 by Zabbix agent active
- MySQL by Zabbix agent
- MySQL by Zabbix agent active
- Nginx by Zabbix agent
- Nginx by Zabbix agent active
- Percona by Zabbix agent
- Percona by Zabbix agent active
- PHP-FPM by Zabbix agent
- PHP-FPM by Zabbix agent active
- Podman by Zabbix agent
- Podman by Zabbix agent active
- PostgreSQL by Zabbix agent
- PostgreSQL by Zabbix agent active
- RabbitMQ cluster by Zabbix agent

Zabbix agent 2 模板操作

确保使用Zabbix agent 2 收集指标的模板正常运行的步骤：

1. 确保主机上已安装 agent 2，并且已安装的版本包含所需的插件。在某些情况下，您可能需要先升级 agent 2。
2. 将模板链接到目标主机（如果您的 Zabbix 安装中没有该模板，您可能需要先导入该模板）。
3. 如有必要，调整模板宏的值。请注意，可以使用用户宏来覆盖配置参数。
4. 配置被监控的实例，以允许与 Zabbix 共享数据。

Attention:
Zabbix agent 2 模板与插件配合工作。虽然基本配置只需调整用户宏即可完成，但更深入的自定义可以通过配置插件本身来实现。例如，如果某个插件支持命名会话，则可以在配置文件中为每个实体指定带有各自 URI、用户名和密码的命名会话，从而监控多个同类实体（例如 MySQL1 和 MySQL2）。

模板的详细说明，包括宏、监控项和触发器的完整列表，可在该模板的 README 文件中查看（点击模板名称即可访问）。

可用的模板如下：

- Ceph by Zabbix agent 2
- Docker
- Memcached
- MongoDB cluster by Zabbix agent 2
- MongoDB node by Zabbix agent 2
- MySQL by Zabbix agent 2
- MySQL by Zabbix agent 2 active
- Oracle by Zabbix agent 2
- PostgreSQL by Zabbix agent 2
- PostgreSQL by Zabbix agent 2 active
- Redis
- SMART by Zabbix agent 2
- SMART by Zabbix agent 2 active
- Systemd by Zabbix agent 2
- Website certificate by Zabbix agent 2

HTTP 模板操作

请按照以下步骤，确保使用HTTP agent 收集指标的模板正常运行：

1. 在 Zabbix 中创建与监控目标对应的主机。将监控目标的 IP 地址/DNS 名称指定为主机的主要主机接口。这是为了确保模板监控项中的 {HOST.CONN} 宏能够正确解析。
2. 将模板关联到步骤 1 中创建的主机（如果您的 Zabbix 安装中没有该模板，可能需要先导入模板）。
3. 如有必要，调整模板级别的宏值。
4. 配置被监控的实例，以允许与 Zabbix 共享数据。

模板的详细说明，包括宏、监控项和触发器的完整列表，可在模板的 README 文件中找到（点击模板名称即可访问）。

以下模板可用：

- Acronis Cyber Protect Cloud by HTTP
- Apache by HTTP
- Asterisk by HTTP
- AWS by HTTP
- AWS Cost Explorer by HTTP
- AWS EC2 by HTTP
- AWS ECS Cluster by HTTP
- AWS ECS Serverless Cluster by HTTP
- AWS ELB Application Load Balancer by HTTP
- AWS ELB Network Load Balancer by HTTP
- AWS Lambda by HTTP
- AWS RDS instance by HTTP
- AWS S3 bucket by HTTP
- Azure by HTTP
- Cisco Meraki organization by HTTP
- Cisco SD-WAN by HTTP
- Cisco Secure Firewall Threat Defense by HTTP
- Claude API by HTTP
- ClickHouse by HTTP
- Cloudflare by HTTP
- CockroachDB by HTTP
- Control-M enterprise manager by HTTP
- Control-M server by HTTP
- Cradlepoint NCM v2 by HTTP
- DELL PowerEdge R720 by HTTP
- DELL PowerEdge R740 by HTTP
- DELL PowerEdge R820 by HTTP
- DELL PowerEdge R840 by HTTP
- Domain RDAP by HTTP
- Elasticsearch Cluster by HTTP
- Envoy Proxy by HTTP
- Etcd by HTTP
- FortiGate by HTTP
- GitHub repository by HTTP
- GitLab by HTTP
- GLPI by HTTP
- Google Cloud Platform (GCP) by HTTP
- Hadoop by HTTP
- HAProxy by HTTP
- HashiCorp Consul Cluster by HTTP
- HashiCorp Consul Node by HTTP
- HashiCorp Nomad by HTTP
- HashiCorp Vault by HTTP
- Hikvision camera by HTTP
- HPE iLO by HTTP
- HPE MSA 2040 Storage by HTTP
- HPE MSA 2060 Storage by HTTP
- HPE Primera by HTTP
- HPE Synergy by HTTP
- InfluxDB by HTTP

- Jenkins by HTTP
- Kubernetes API server by HTTP, Kubernetes cluster state by HTTP, Kubernetes Controller manager by HTTP, Kubernetes kubelet by HTTP, Kubernetes nodes by HTTP, Kubernetes Scheduler by HTTP 已被弃用，并已由 *Kubernetes Cluster by HTTP* 模板取代。
- Kubernetes Cluster by HTTP
- MantisBT by HTTP
- Microsoft 365 reports by HTTP
- Microsoft SharePoint by HTTP
- NetApp AFF A700 by HTTP
- Nextcloud by HTTP
- NGINX by HTTP
- NGINX Plus by HTTP
- Nutanix Prism Element by HTTP
- OpenAI Platform by HTTP
- OpenStack by HTTP
- OpenWeatherMap by HTTP
- Oracle Cloud by HTTP
- Palo Alto PA-440 by HTTP
- PHP-FPM by HTTP
- Podman by HTTP
- Proxmox VE by HTTP
- Pure Storage FlashArray v1 and v2 by HTTP
- RabbitMQ cluster by HTTP
- Ribbon SBC Edge by HTTP
- Ribbon SBC SWe CE by HTTP
- Ribbon SBC SWe Core by HTTP
- TiDB by HTTP
- TiDB PD by HTTP
- TiDB TiKV by HTTP
- Travis CI by HTTP
- Veeam Backup Enterprise Manager by HTTP
- Veeam Backup and Replication by HTTP
- VeloCloud SD-WAN by HTTP
- VeloCloud SD-WAN Edge by HTTP
- YugabyteDB by HTTP
- ZooKeeper by HTTP

IPMI 模板操作

IPMI 模板不需要任何特定的设置。要开始监控，请将模板链接 到目标主机（如果该模板在您的 Zabbix 安装中不可用，您可能需要先导入该模板）。

模板的详细说明，包括宏、监控项和触发器的完整列表，可在模板的 README 文件中查看（可通过单击模板名称访问）。

可用模板：

- Chassis by IPMI

JMX 模板操作

确保通过JMX 收集指标的模板能够正常工作的步骤：

1. 确保 Zabbix Java 网关 已正确安装并配置。
2. 将模板链接 到目标主机。该主机应已设置 JMX 接口。如果您的 Zabbix 安装中没有该模板，可能需要先导入 该模板。
3. 如有必要，调整模板宏的值。
4. 配置被监控的实例，以允许与 Zabbix 共享数据。

模板的详细说明，包括宏、监控项和触发器的完整列表，可在模板的 README 文件中查看（点击模板名称即可访问）。

可用的模板如下：

- Apache ActiveMQ by JMX
- Apache Cassandra by JMX
- Apache Kafka by JMX
- Apache Tomcat by JMX

- GridGain by JMX
- Ignite by JMX
- Jira Data Center by JMX
- WildFly Domain by JMX
- WildFly Server by JMX

ODBC 模板操作

确保通过ODBC 监控 收集指标的模板能够正确运行，请执行以下步骤：

1. 确保所需的 ODBC 驱动已安装在 Zabbix 服务器或 proxy 上。
2. 将该模板链接 到目标主机（如果该模板在您的 Zabbix 安装中不可用，您可能需要先导入 该模板）。
3. 如有必要，调整模板宏的值。
4. 配置被监控的实例，以允许与 Zabbix 共享数据。

模板的详细说明，包括宏、监控项和触发器的完整列表，可在模板的 README 文件中查看（点击模板名称即可访问）。

可用的模板如下：

- MariaDB by ODBC
- MSSQL by ODBC
- MySQL by ODBC
- Oracle by ODBC
- Percona by ODBC
- PostgreSQL by ODBC

网络设备的标准化模板

概述

为了为交换机和路由器等网络设备提供监控，我们创建了两个所谓的模型：一个用于网络设备本身（基本上就是它的机箱），另一个用于网络接口。

提供了许多网络设备系列的模板。所有模板都涵盖了（在能够从设备获取这些监控项的情况下）：

- 机箱故障监控（电源、风扇和温度、整体状态）
- 机箱性能监控（CPU 和内存监控项）
- 机箱库存采集（序列号、型号名称、固件版本）
- 使用 IF-MIB 和 EtherLike-MIB 的网络接口监控（接口状态、接口流量负载、以太网双工状态）

如果你的 Zabbix 安装中没有该模板，你可能需要先导入该模板。

如果你正在导入新的开箱即用模板，你可能还需要将 `@Network interfaces for discovery` 全局正则表达式更新为：

```
Does not match regular expression: ^Software Loopback Interface
Does not match regular expression: ^(In)?[lL]oop[bB]ack[0-9._]*$
Does not match regular expression: ^NULL[0-9.]*$
Does not match regular expression: ^[lL]o[0-9.]*$
Does not match regular expression: ^[sS]ystem$
Does not match regular expression: ^Nu[0-9.]*$
```

以便在大多数系统上过滤掉回环接口和空接口。

设备

可用模板所支持的设备系列列表：

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
Alcatel Timetra TiMOS SNMP	Alcatel	Alcatel Timetra	ALCATEL SR 7750	TiMOS	TIMETRA-SYSTEM-MIB,TIMETRA-CHASSIS-MIB	已认证

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
Aruba CX 8300s by SNMP	HPE Aruba Network-ing	Aruba CX 交换机系列	Aruba JL636A 8325, Aruba JL717A 8360	Aruba AOS-CX	ARUBAWIRED-FAN-MIB, ARUBAWIRED-POWERSUPPLY-MIB, ARUBAWIRED-SYSTEMINFO-MIB, ARUBAWIRED-TEMPSENSOR-MIB, OSPF-MIB	已认证
Brocade FC SNMP	Brocade	Brocade FC 交换机	Brocade 300 SAN Switch-	-	SW-MIB,ENTITY-MIB	性能 故障
Brocade_Foundry Stackable SNMP	Brocade	Brocade ICX	Brocade ICX6610, Brocade ICX7250-48, Brocade ICX7450-48F		FOUNDRY-SN-AGENT-MIB, FOUNDRY-SN-STACKING-MIB	已认证
Brocade_Foundry Nonstackable SNMP	Brocade, Foundry	Brocade MLX, Foundry	Brocade MLXe, Foundry FLS648, Foundry FWSX424		FOUNDRY-SN-AGENT-MIB	性能 故障
Check Point Next Generation Firewall by SNMP	Check Point	下一代防火墙	-	Gaia	HOST-RESOURCES-MIB, CHECKPOINT-MIB, UCD-SNMP-MIB, SNMPv2-MIB, IF-MIB	已认证
Ciena 3906 by SNMP	Ciena	Ciena CPE	Ciena 3906	SAOS	WWP-LEOS-BLADE-MIB, WWP-LEOS-CHASSIS-MIB, WWP-LEOS-SYSTEM-CONFIG-MIB	已认证
Cisco Catalyst 3750<device model> SNMP	Cisco	Cisco Catalyst 3750	Cisco Catalyst 3750V2-24FS, Cisco Catalyst 3750V2-24PS, Cisco Catalyst 3750V2-24TS, Cisco Catalyst SNMP, Cisco Catalyst SNMP		CISCO-MEMORY-POOL-MIB, IF-MIB, EtherLike-MIB, SNMPv2-MIB, CISCO-PROCESS-MIB, CISCO-ENVMON-MIB, ENTITY-MIB	已认证
Cisco IOS SNMP	Cisco	Cisco IOS 版本 > 12.2 3.5	Cisco C2950	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	已认证
Cisco IOS versions 12.0_3_T-12.2_3.5 SNMP	Cisco	Cisco IOS > 12.0 3 T 且 12.2 3.5	-	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	已认证
Cisco IOS prior to 12.0_3_T SNMP	Cisco	Cisco IOS 12.0 3 T	-	IOS	OLD-CISCO-CPU-MIB,CISCO-MEMORY-POOL-MIB	已认证
D-Link DES_DGS Switch SNMP	D-Link	DES/DGX 交换机	D-Link DES-xxxx/DGS-xxxx,DLINK DGS-3420-26SC	-	DLINK-AGENT-MIB,EQUIPMENT-MIB,ENTITY-MIB	已认证

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
D-Link DES 7200 SNMP	D-Link	DES-7xxx	D-Link DES 7206	-	ENTITY-MIB,MY-SYSTEM-MIB,MY-PROCESS-MIB,MY-MEMORY-MIB	性能 故障 接口
Dell Force S-Series SNMP	Dell	Dell Force S 系列	S4810		F10-S-SERIES-CHASSIS-MIB	已认证
Extreme Exos SNMP	Extreme	Extreme EXOS	X670V-48x	EXOS	EXTREME-SYSTEM-MIB,EXTREME-SOFTWARE-MONITOR-MIB	已认证
FortiGate by SNMP	Fortinet	FortiGate（NGFW）	-	FortiOS	HOST-RESOURCES-MIB FORTINET-FORTIGATE-MIB FORTINET-CORE-MIB SNMPv2-MIB IF-MIB ENTITY-MIB	性能 资产清单
HP Comware HH3C SNMP	HP	HP（H3C）Comware	HP A5500-24G-4SFP HI Switch		HH3C-ENTITY-EXT-MIB,ENTITY-MIB	已认证
HP Enterprise Switch SNMP	HP	HP 企业交换机	HP ProCurve J4900B Switch 2626, HP J9728A 2920-48G Switch		STATISTICS-MIB,NETSWITCH-MIB,HP-ICF-CHASSIS,ENTITY-MIB,SEMI-MIB	已认证
Huawei AR600 by SNMP	Huawei	Huawei NetEngine AR600 系列路由器	搭载 VRP 软件（V300R019/V300R021/V300R023 或更高版本）的 Huawei AR600/AR6100/AR6200/AR6300 系列路由器		HUAWEI-CBQOS-MIB, HUAWEI-NQA-MIB, ENTITY-MIB, IF-MIB, HOST-RESOURCES-MIB, SNMPv2-MIB	已认证
Huawei OceanStor 5300 V5 by SNMP	Huawei	Huawei OceanStor Dorado	搭载 V5 软件的 Huawei OceanStor Dorado		HUAWEI-STORAGE-HARDWARE-MIB, HUAWEI-STORAGE-SPACE-MIB, ISM-PERFORMANCE-MIB, HOST-RESOURCES-MIB, SNMPv2-MIB	已认证
Huawei OceanStor Dorado by SNMP	Huawei	Huawei OceanStor, Huawei OceanStor Dorado	搭载 V6/V700 软件的 Huawei OceanStor Dorado 3000/5000/6000/8000/18000/5300 V5/5500 V5/6800 V5/18500/18800		HUAWEI-STORAGE-HARDWARE-MIB, HUAWEI-STORAGE-SPACE-MIB, ISM-PERFORMANCE-MIB, HOST-RESOURCES-MIB, SNMPv2-MIB	已认证

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
Huawei VRP by SNMP	Huawei	Huawei VRP	S2352P-EI		ENTITY-MIB,HUAWEI-ENTITY-EXTENT-MIB	已认证
Intel_Qlogic Infiniband SNMP	Intel/QLogic	Intel/QLogic Infini-band 设备	Infiniband 12300		ICS-CHASSIS-MIB	故障 资产清单
Juniper SNMP	Juniper	MX、SRX、EX 型号	Juniper MX240, Juniper EX4200-24F	JunOS	JUNIPER-MIB	已认证
Juniper MX NETCONF	Juniper	MX 型号	Juniper MX204 Edge Router	JunOS 24.2R1-S1.10	不适用（通过 *SSH* 使用 *NETCONF*）	已认证
Juniper MX SNMP	Juniper	MX 型号	Juniper MX204 Edge Router	JunOS 24.2R1-S1.10	OSPF-MIB, JUNIPER-DOM-MIB, JUNIPER-MIB, BGP4-V2-MIB-JUNIPER, OSPFV3-MIB-JUNIPER	已认证
Mellanox SNMP	Mellanox	Mellanox Infini-band 设备	SX1036	MLNX-OS	HOST-RESOURCES-MIB,ENTITY-MIB,ENTITY-SENSOR-MIB,MELLANOX-MIB	已认证
MikroTik CCR<device model> SNMP	MikroTik	MikroTik Cloud Core 路由器 (CCR 系列)	可用的独立专用模板包括 MikroTik CCR1009-7G-1C-1S+, MikroTik CCR1009-7G-1C-1S+PC, MikroTik CCR1009-7G-1C-PC, MikroTik CCR1016-12G, MikroTik CCR1016-12S-1S+, MikroTik CCR1036-12G-4S-EM, MikroTik CCR1036-12G-4S, MikroTik CCR1036-8G-2S+, MikroTik CCR1036-8G-2S+EM, MikroTik CCR1072-1G-8S+, MikroTik CCR2004-16G-2S+, MikroTik CCR2004-1G-12S+2XS	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
MikroTik CRS<device model> SNMP	MikroTik	MikroTik Cloud Router 交换机 (CRS 系列)	可用的独立专用模板包括 MikroTik CRS106-1C-5S, MikroTik CRS109-8G-1S-2HnD-IN, MikroTik CRS112-8G-4S-IN, MikroTik CRS112-8P-4S-IN, MikroTik CRS125-24G-1S-2HnD-IN, MikroTik CRS212-1G-10S-1S+IN, MikroTik CRS305-1G-4S+IN, MikroTik CRS309-1G-8S+IN, MikroTik CRS312-4C+8XG-RM, MikroTik CRS317-1G-16S+RM, MikroTik CRS326-24G-2S+IN, MikroTik CRS326-24G-2S+RM, MikroTik CRS326-24S+2Q+RM, MikroTik CRS328-24P-4S+RM, MikroTik CRS328-4C-20S-4S+RM, MikroTik CRS354-48G-4S+2Q+RM, MikroTik CRS354-48P-4S+2Q+RM	RouterOS/SwitchOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
MikroTik CSS<device model> SNMP	MikroTik	MikroTik Cloud Smart 交换机 (CSS 系列)	可用的独立专用模板包括 MikroTik CSS326-24G-2S+RM, MikroTik CSS610-8G-2S+IN	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
MikroTik FiberBox SNMP	MikroTik	MikroTik FiberBox	MikroTik FiberBox	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
MikroTik hEX <device model> SNMP	MikroTik	MikroTik hEX	可用的独立专用模板包括 MikroTik hEX, MikroTik hEX lite, MikroTik hEX PoE, MikroTik hEX PoE lite, MikroTik hEX S	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
MikroTik netPower <device model> SNMP	MikroTik	MikroTik net-Power	可用的独立专用模板包括 MikroTik netPower 15FR, MikroTik netPower 16P SNMP, MikroTik netPower Lite 7R	RouterOS/SwitchOS, SwitchOS Lite	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
MikroTik PowerBox <device model> SNMP	MikroTik	MikroTik Power-Box	可用的独立专用模板包括 MikroTik PowerBox, MikroTik PowerBox Pro	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
MikroTik RB<device model> SNMP	MikroTik	MikroTik RB 系列路由器	可用的独立专用模板包括 MikroTik RB1100AHx4, MikroTik RB1100AHx4 Dude Edition, MikroTik RB2011iL-IN, MikroTik RB2011iL-RM, MikroTik RB2011iLS-IN, MikroTik RB2011UiAS-IN, MikroTik RB2011UiAS-RM, MikroTik RB260GS, MikroTik RB3011UiAS-RM, MikroTik RB4011iGS+RM, MikroTik RB5009UG+S+IN	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
MikroTik SNMP	MikroTik	MikroTik RouterOS 设备	MikroTik CCR1016-12G, MikroTik RB2011UAS-2HnD, MikroTik 912UAG-5HPnD, MikroTik 941-2nD, MikroTik 951G-2HnD, MikroTik 1100AHx2	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	已认证
Netgear Fastpath SNMP	Netgear	Netgear Fastpath	M5300-28G		FASTPATH-SWITCHING-MIB,FASTPATH-BOXSERVICES-PRIVATE-MIB	故障 资产清单
QTech QSW SNMP	QTech	Qtech 设备	Qtech QSW-2800-28T	-	QTECH-MIB,ENTITY-MIB	性能 资产清单

模板名称	厂商	设备系列	已知型号	操作系统	使用的 MIB	标签
Stormshield SNS by SNMP	Stormshield	Stormshield Network Security (SNS) 防火墙	SN3100	Stormshield SNS	HOST-RESOURCES-MIB, UCD-SNMP-MIB, STORMSHIELD-ASQ-STATS-MIB, STORMSHIELD-AUTOUPDATE-MIB, STORMSHIELD-HA-MIB, STORMSHIELD-PROPERTY-MIB, STORMSHIELD-HEALTH-MONITOR-MIB, STORMSHIELD-IF-MIB, STORMSHIELD-SYSTEM-MONITOR-MIB, STORMSHIELD-IPSEC-STATS-MIB	已认证
TP-LINK SNMP	TP-LINK	TP-LINK	T2600G-28TS v2.0		TPLINK-SYSMONITOR-MIB,TPLINK-SYSINFO-MIB	性能 资产清单
Ubiquiti AirOS SNMP	Ubiquiti	Ubiquiti AirOS 无线设备	NanoBridge,NanoStation,Unifi	AirOS	FROGFOOT-RESOURCES-MIB,IEEE802dot11-MIB	性能
Vyatta Virtual Router by SNMP	Ciena	Vyatta	Vyatta Virtual Router 1908e	Vyatta 1908e	SNMPv2-MIB, HOST-RESOURCES-MIB, UCD-SNMP-MIB, IF-MIB, DISMAN-EVENT-MIB	性能 资产清单

模板设计

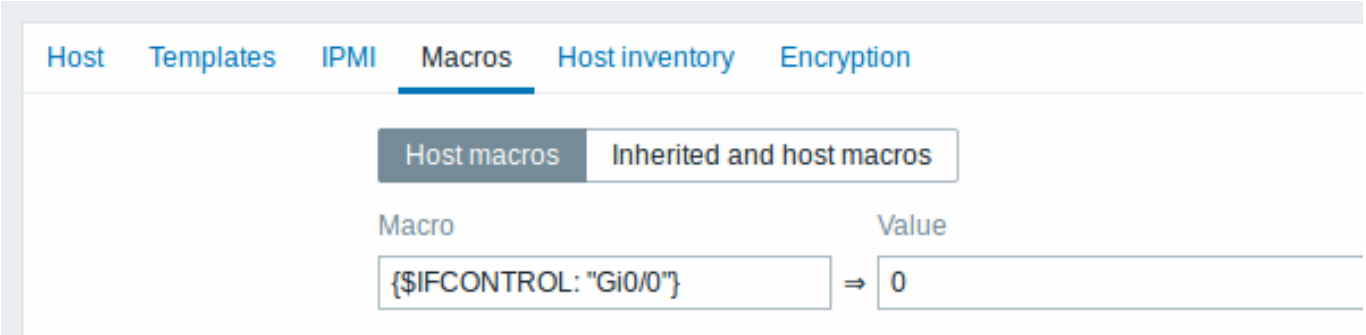
模板的设计考虑了以下几点：

- 尽可能多的使用用户宏，以便用户可以调整触发器；
- 尽可能使用底层自动发现，以尽量减少不受支持的监控项的数量；
- 所有模板都依赖于模板 ICMP Ping，因此所有设备也由 ICMP 检查；
- 监控项不使用任何 MIB - SNMP OID 用于监控项和底层自动发现。因此，无需将任何 MIB 加载到 Zabbix 中即可使模板正常工作；
- 环回网络接口在发现时被过滤以及 ifAdminStatus = down(2) 的接口;
- 尽可能使用 IF-MIB::ifXTable 中的 64 位计数器。如果不支持，则使用默认的 32 位计数器。

所有发现的网络接口都有一个触发器来监控其运行状态（链接），例如：

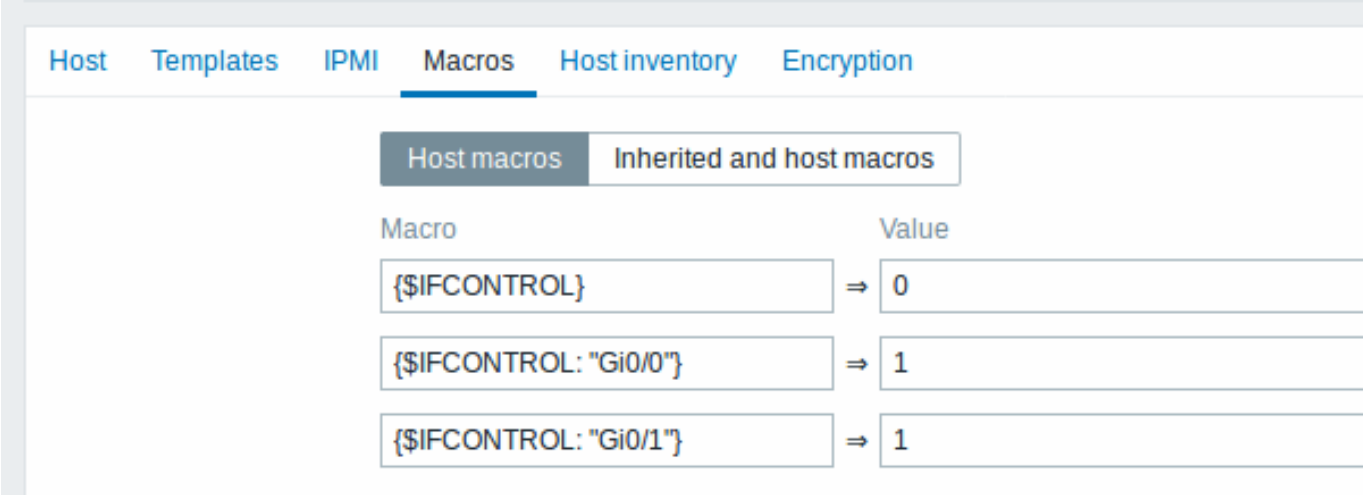
```
{$IFCONTROL:"{#IFNAME}"}=1 and last(/Alcatel Timetra TiMOS SNMP/net.if.status[ifOperStatus.{#SNMPINDEX}])
```

- 如果您不想监控特定接口的这种情况，请创建一个上下文值为 0 的用户宏。例如：



其中 Gi0/0 是 {#IFNAME}。这样触发器就不再用于此特定接口。

- 您还可以更改所有触发器的默认行为，使其不触发并仅对有限数量的接口（如上行链路）激活此触发器：



标签

- 性能 - 设备系列 MIB 提供了监控 CPU 和内存监控项的方法；
- 故障 - 设备系列 MIB 提供了监控至少一个温度传感器的方法；
- 资产清单 - 设备系列 MIB 提供了采集至少设备序列号和型号名称的方法；
- 已认证 - 以上三个主要类别均已覆盖。

VMware 模板操作

概述

Zabbix 包含一系列开箱即用的虚拟机监控模板，专为 VMware vCenter 和 ESXi 管理程序设计。可用模板包含在两套独立的模板集中：

- VMware - 对应宏使用 UUID 数据
- VMware FQDN - 对应宏使用 FQDN 数据

为监控 VMware vCenter 或 ESXi 管理程序而设计的模板包含预配置的低级别发现规则，以及用于监控虚拟安装环境的各种内置检查。

Note:
为了使 *VMware FQDN* 模板正常工作，每台被监控的虚拟机都应具有符合 FQDN 规则的唯一操作系统名称。此外，每台机器上都必须安装 VMware Tools/Open Virtual Machine tools。如果满足这些前提条件，建议使用 *VMware FQDN* 模板。*VMware FQDN* 模板自 Zabbix 5.2 起可用，当时引入了使用自定义接口创建主机的功能。

 还提供了经典的 *VMware* 模板，可在不满足 FQDN 要求时使用。但是，*VMware* 模板存在一个已知问题。已发现虚拟机的主机会使用保存在 vCenter 中的名称创建（例如 “VM1”、“VM2” 等）。如果这些主机上安装了 Zabbix agent，并且启用了主动式 Zabbix agent 自动注册，自动注册过程将读取主机在启动时注册的主机名（例如 “vm1.example.com”、“vm2.example.com” 等）。这可能会导致为现有虚拟机创建新的主机（因为未找到名称匹配项），从而产生名称不同的重复主机。

请注意：

- 模板 *VMware Hypervisor* 既可以手动链接到主机，也可以在发现中使用；
- 模板 *VMware Guest* 不应手动链接到主机，只能在发现中使用。

主机宏配置

要使用 VMware 简单检查，主机必须定义以下用户宏：

- {$VMWARE.URL} - VMware 服务（vCenter 或 ESXi 管理程序）SDK URL (https://servername/sdk)
- {$VMWARE.USERNAME} - VMware 服务用户名
- {$VMWARE.PASSWORD} - VMware 服务 {$VMWARE.USERNAME} 用户的密码

OTLP 模板操作

Zabbix 提供了一系列开箱即用的模板，可使用 *Telemetry query* 监控项收集和监控 OpenTelemetry 跟踪、指标和日志。

确保模板正常运行的步骤：

1. 查看模板的 README 文件，了解模板特定的配置要求。
2. 在 Zabbix 中创建一个与监控目标对应的主机。
3. 确保在 Zabbix 服务器或 Zabbix proxy配置文件中配置了 TelemetryProvider 参数。
4. 将模板链接到创建的主机（如果您的 Zabbix 安装中没有该模板，可能需要先导入模板）。
5. 如有必要，调整模板级别的宏值。

模板的详细说明（包括宏、监控项和触发器的完整列表）位于模板的 README 文件中。您可以通过单击模板名称访问该文件。

以下模板可用：

- Envoy Proxy by OTLP
- Generic OpenTelemetry by OTLP
- OpenRouter traces by OTLP
- Proxmox VE by OTLP

事件通知

概述

在配置了诸多监控项和触发器，并且在触发器状态发生变化的情况下，用户已经接收到了一些告警信息，那么接下来就要考虑通过配置动作（actions）来响应事件的发生。

值得强调的是，用户不可能一直观察触发器或者事件的状态变化。更好的解决办法是，当发生明显的变化时（例如出现紧急问题）用户可以接收到系统发送的通知。当然，问题发生时，所有相关人员都可以接收到该事件通知是最好的。

可以说，事件通知是 Zabbix 首要的动作配置之一。该配置可以细致到对特定事件所要通知的对象和时间进行定义。

开启 Zabbix 事件通知的发送和接收功能，您需要做到：

- 定义媒介
- 配置动作 向指定的定义媒介发送消息

标准的动作由 条件和 操作两个元素构成。基本上，每当设定的条件达成时，就会执行相对应的设定操作。最重要的两个操作分别为：发送消息（事件提醒）和执行远程命令。

对于发现和自动注册所创建的事件，用户可以配置一些额外的操作。这些操作包括添加或删除主机，链接一个监控模板等。

媒体类型

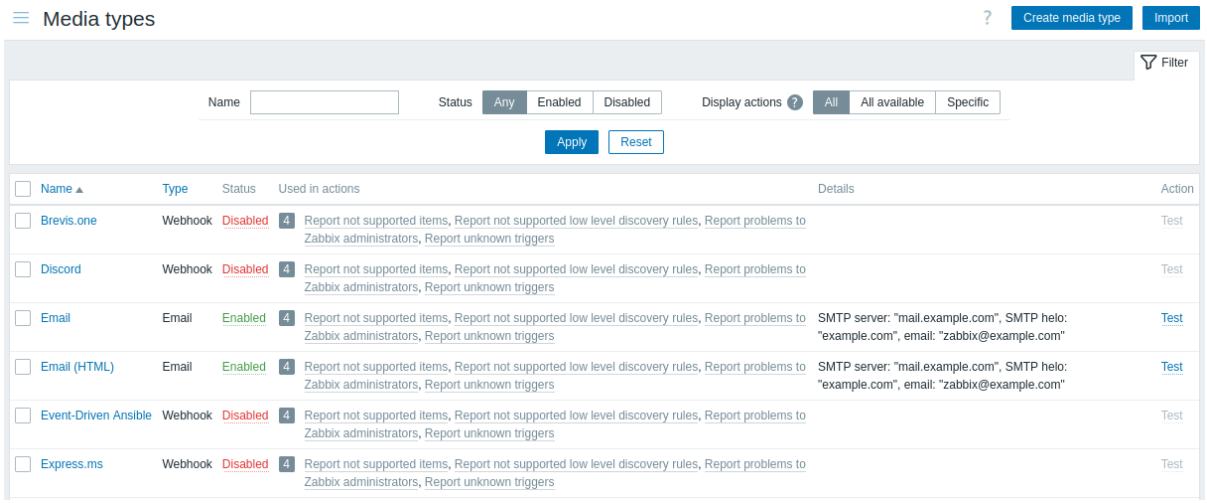
概述

媒体类型是 Zabbix 用于发送通知和告警的传递渠道。

媒体类型支持以下传递方式：

- 电子邮件
- 推送通知
- 短信
- 自定义脚本
- Webhook

媒体类型在告警 > 媒体类型中进行维护。默认数据集中预定义了一些媒体类型。您只需调整其参数即可使其正常工作。



要了解媒体类型在告警流程中的作用，请看一下将通知从 Zabbix 发送给最终用户所需的三个条件：

1. 必须定义一个用于发送通知的操作操作步骤。
2. 必须定义一个可正常工作的媒体类型（例如使用 SMTP 发送告警的电子邮件）。
3. 必须在用户媒体中定义用户级别的传递详细信息（例如电子邮件地址、电话号码等）。

媒体类型测试

要测试已配置的媒体类型是否正常工作，请在媒体类型列表中点击 测试。

测试请求将发送到 Zabbix 服务器。Zabbix 服务器将尝试使用指定的媒体类型发送告警，并将结果返回给前端。前端将等待服务器返回结果。媒体类型测试默认超时时间为 65 秒（可在 管理 > 常规 > 超时 中配置）。

另请参阅以下媒体类型的测试详情：

- 电子邮件
- 推送通知
- webhook
- 脚本

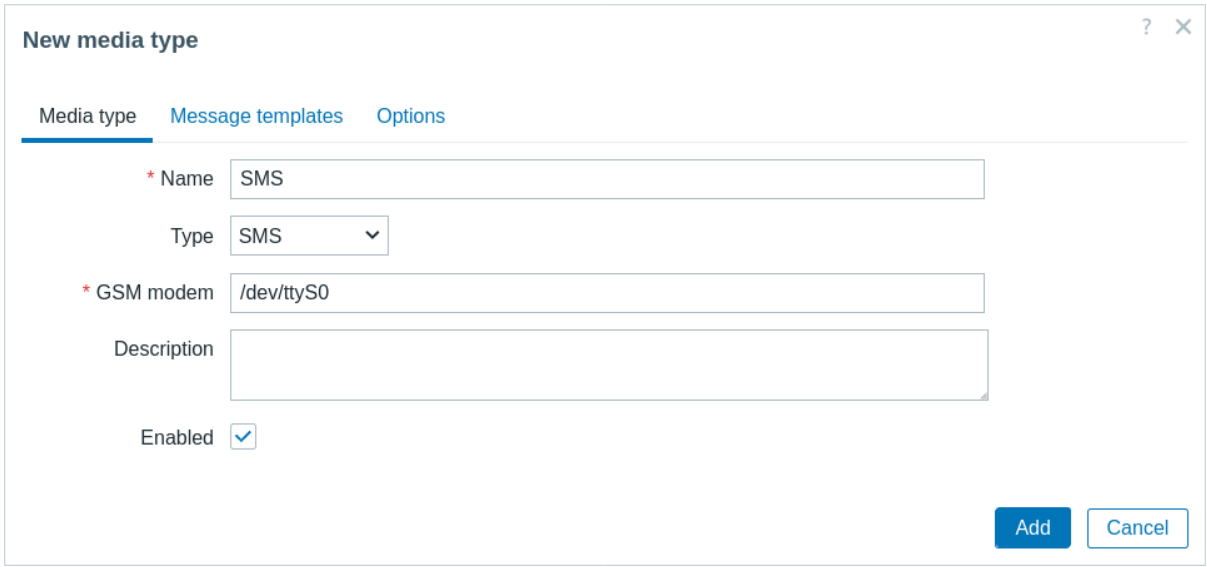
配置

要在 Zabbix 前端中创建媒体类型：

1. 转到 告警 > 媒体类型。
2. 单击 创建媒体类型。
3. 在表单中输入媒体类型的参数。

某些参数对所有传递方式都是通用的。

Note:
如果在前端配置文件（zabbix.conf.php）的 $ZBX_FEATURE_FLAGS['media_type_denylist'] 功能标志中指定了一个或多个传递方式，则这些传递方式不会显示在 类型列表中。

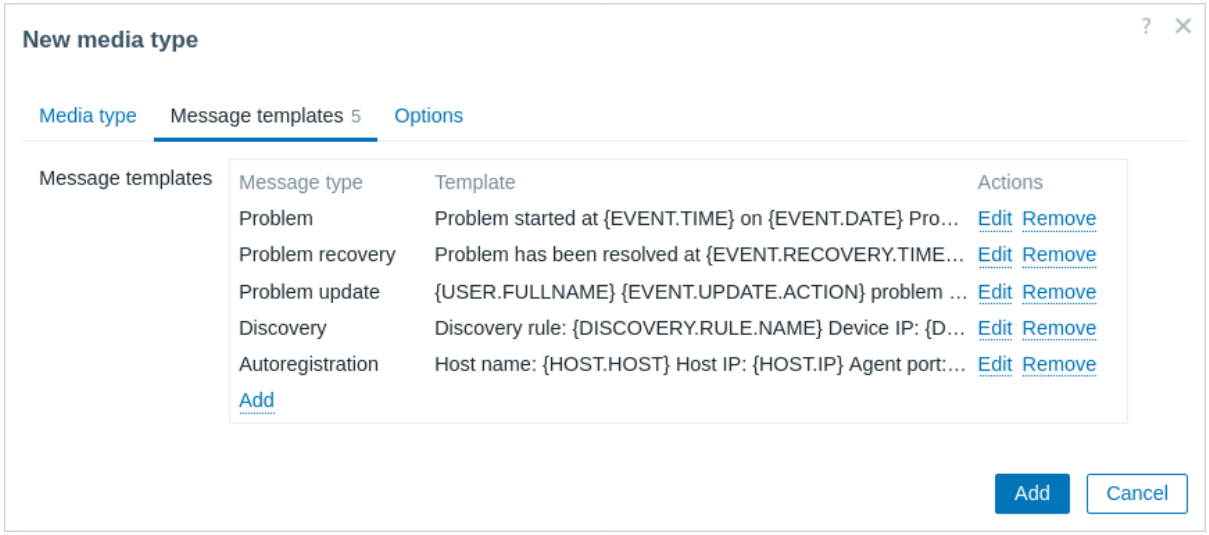


参数	描述
名称	媒体类型的名称。
类型	选择媒体类型的传递方式。
描述	输入媒体类型的描述。
启用	选中复选框以启用媒体类型。

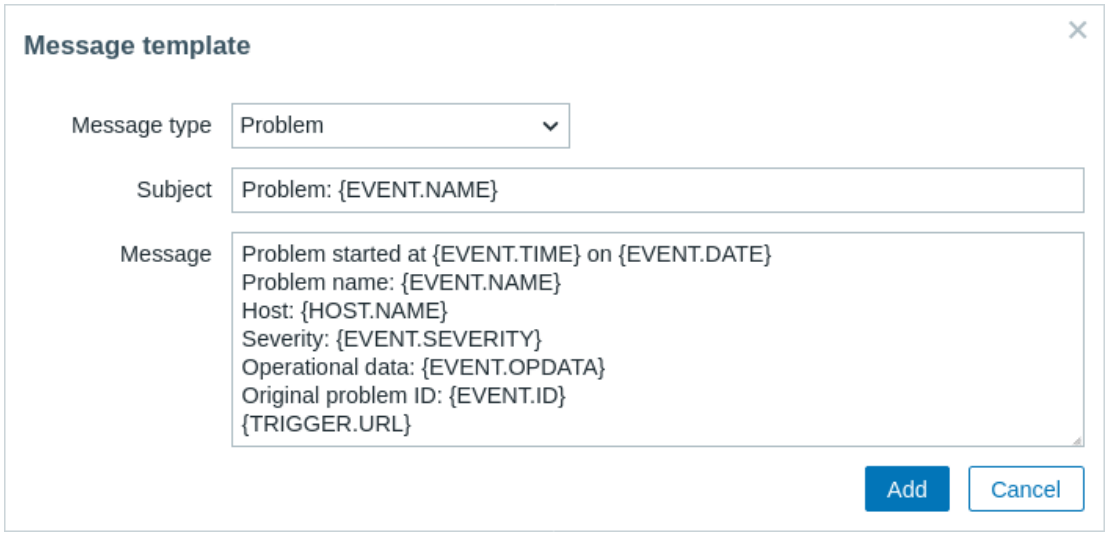
有关特定于传递方式的参数，请参阅电子邮件、推送通知、SMS、自定义 alertscript或webhook页面。

消息模板

消息模板选项卡包含按事件类型（问题、问题恢复、发现等）划分的默认消息。



单击 **Add** 以定义默认消息（或单击 **Edit** 更新现有消息）：



Parameter	Description
Message type	应使用默认消息的事件类型。 每种事件类型只能定义一条默认消息。
Subject	默认消息的主题。主题中可以包含宏。长度限制为 255 个字符。 SMS 媒体类型不支持主题。
Message	默认消息。其可包含的字符数量受数据库类型限制（更多信息请参见发送消息）。 消息中可以包含受支持的宏。 在问题和问题更新消息中，支持表达式宏（例如，`{?avg(/host/key,1h)}`）。

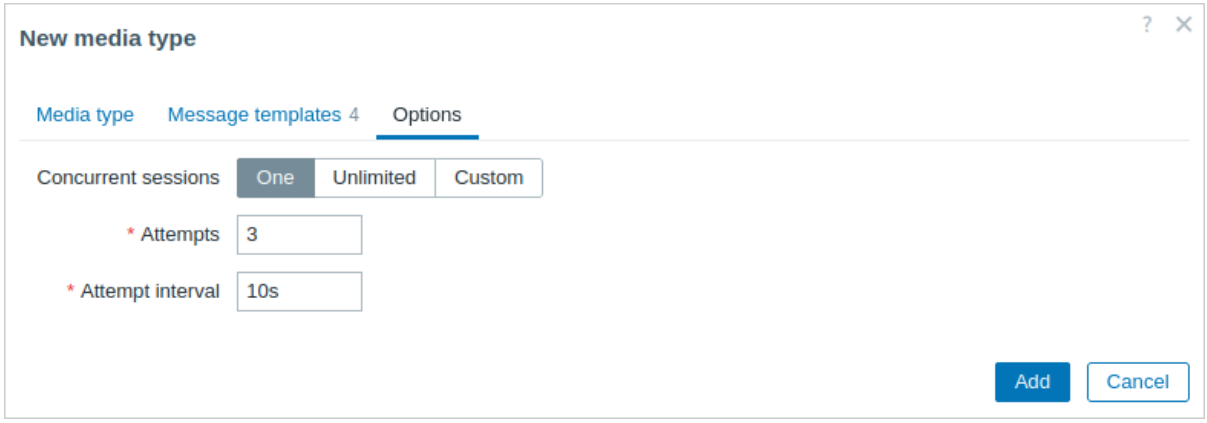
请注意，如果在action operations 中定义了自定义消息，则默认消息会被覆盖。

Warning:
为所有传递方式定义消息模板都是必需的，包括不使用默认消息进行通知的 webhook 或自定义告警脚本。例如，action “Send message to Pushover webhook” 如果未定义 Pushover webhook 的问题消息，将无法发送问题通知。

选项

选项选项卡包含告警处理设置。
每种媒介类型都可以配置相同的一组选项。

所有媒介类型均会并行处理。
虽然可以按媒介类型配置并发会话的最大数量，但服务器上的 alerter 进程总数只能通过 StartAlerters 参数进行限制。
由一个触发器生成的告警会按顺序处理。
因此，只有在多个触发器生成通知时，才可能同时处理多个通知。



参数	描述
并发会话	选择该媒介类型的并行 alerter 会话数： 一个 - 一个会话 无限制 - 不限制会话数 自定义 - 选择自定义的会话数 无限制或较高的值意味着更多的并行会话，以及更高的通知发送容量。应在需要同时发送大量通知的大型环境中使用无限制或较高的值。 如果需要发送的通知数量超过并发会话数，剩余通知将进入队列；这些通知不会丢失。
尝试次数	尝试发送通知的次数。最多可指定 100 次尝试；默认值为 “3”。如果指定 “1”，Zabbix 将只发送一次通知；如果发送失败，则不会重试。
尝试间隔	发送失败时尝试重新发送通知的频率，单位为秒（0-3600）。如果指定 “0”，Zabbix 将立即重试。 支持时间后缀（例如 30s、1m、1h）。

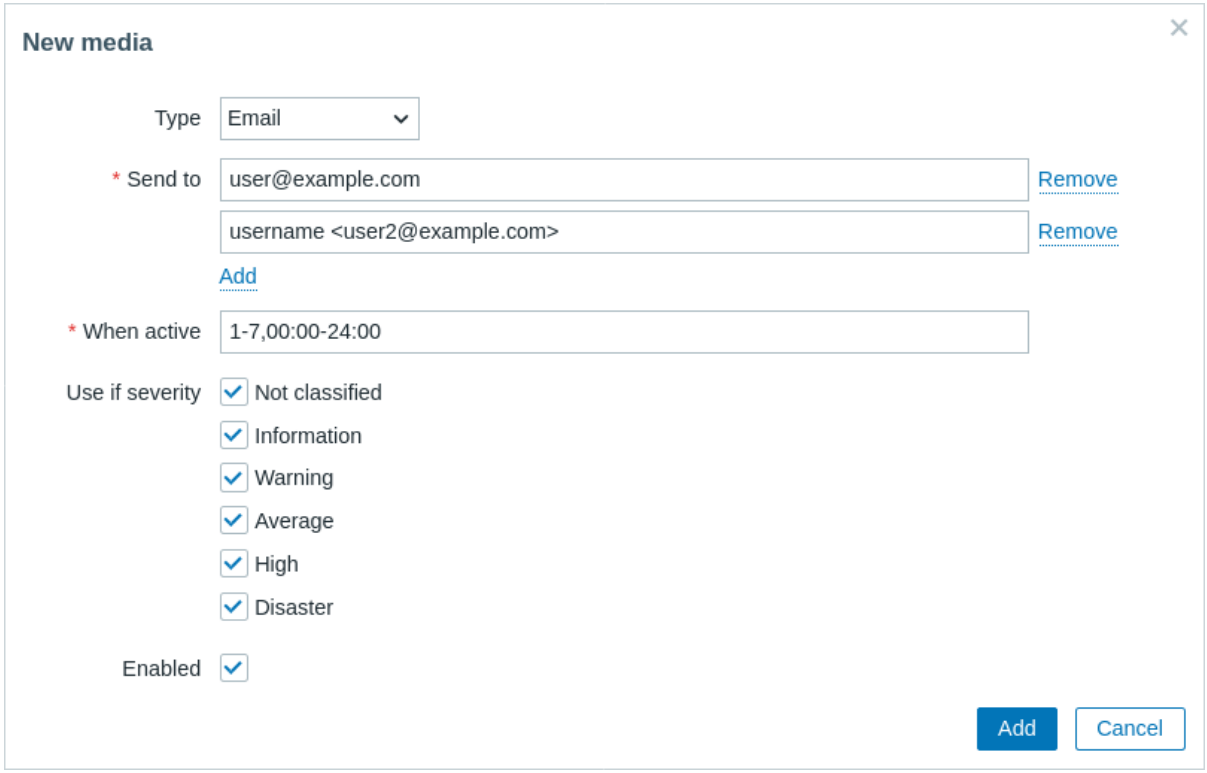
用户媒介

媒介类型定义通知将如何发送，而用户媒介定义通知必须发送到哪里。

无论采用何种发送方式，都必须在用户配置文件中定义用户媒介（例如，电子邮件地址、webhook 用户 ID 等）。如果 webhook *X* 的发送详细信息未在 Admin 用户配置文件中定义，则使用 webhook *X* 向 *Admin* 用户发送消息的动作将无法成功发送。

要定义用户媒介：

1. 转到 *Users* > *Users*，打开用户属性表单。
   或者，转到您自己的用户配置文件中的 *User settings* > *Notifications*。
2. 在 *Media* 选项卡中单击 *Add*。
3. 在表单中输入用户媒介参数。



Parameter	Description
Type	下拉列表包含已启用媒介类型的名称。 请注意，编辑已禁用媒介类型的媒介时，类型将以红色显示。
Send to	输入应接收消息的联系信息。 对于电子邮件媒介类型，可以通过单击地址字段下方的 Add 按钮添加多个地址。在这种情况下，通知将发送到所有列出的地址。有关地址示例，请参阅email 媒介类型的 *Email* 参数说明。 对于推送通知媒介类型，可以从用户的活动移动设备中进行选择，也可以添加一个或多个特定设备。
When active	您可以限制发送消息的时间，例如，仅设置工作日（1-5,09:00-18:00）。请注意，此限制基于用户的时区。如果更改用户时区，且该时区与系统时区不同，则可能需要相应调整此限制，以免错过重要消息。 有关格式说明，请参阅时间段规范页面。 支持用户宏。
Use if severity	选中您希望接收通知的触发器严重性对应的复选框。 注意，如果希望接收非触发器事件的通知，则必须选中默认严重性（“Not classified”）。 保存后，选中的触发器严重性将以相应的严重性颜色显示，而未选中的严重性将显示为灰色。
Status	用户媒介的状态。 **Enabled** - 正在使用。 **Disabled** - 未使用。

电子邮件

概述

要将电子邮件配置为消息的传递渠道，您需要将电子邮件配置为媒体类型，并为用户分配特定的电子邮件地址。

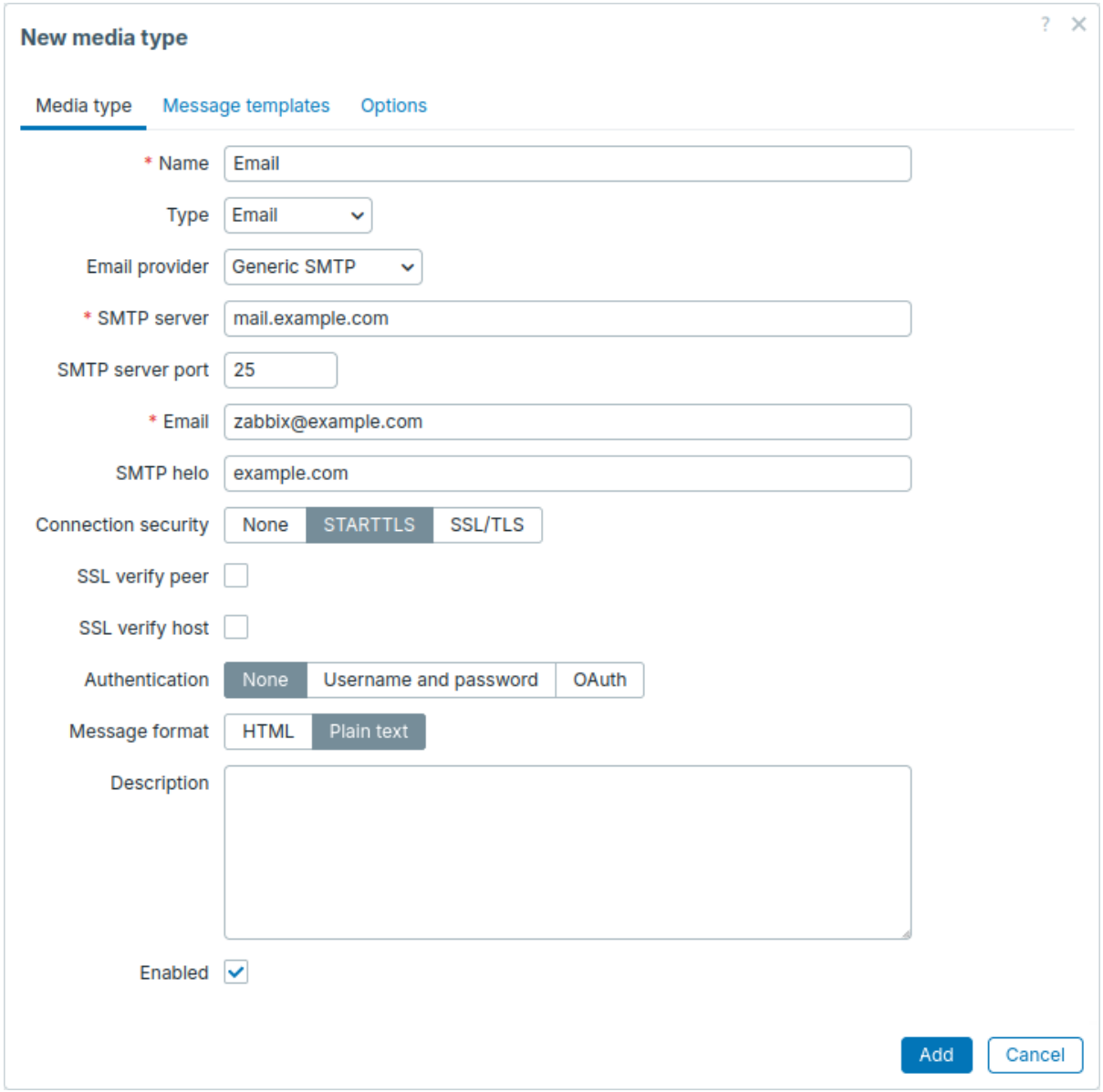
> **Note:**
> 单个事件的多个通知将归并到同一个电子邮件会话线程中。

配置

要将电子邮件配置为媒介类型：

1. 转到 告警 > 媒介类型。
2. 单击 创建媒介类型（或在预定义媒介类型列表中单击 *Email*）。
3. 在表单中输入电子邮件媒介类型参数。

媒介类型选项卡包含常规媒介类型属性：



所有必填输入字段均以红色星号标记。

Warning:
导出电子邮件媒介类型配置时，密码将以明文形式导出。

以下参数专用于电子邮件媒介类型：

参数	描述
Email provider	选择电子邮件提供商：*Generic SMTP*、*Gmail*、*Gmail relay*、*Office365* 或 *Office365 relay*。 如果选择与 Gmail/Office365 相关的选项，则只需提供发件人电子邮件地址和密码；*SMTP server*、*SMTP server port*、*SMTP helo* 和 *Connection security* 等选项将由 Zabbix 自动填充。另请参阅：自动化 Gmail/Office365 媒介类型。
SMTP server	设置用于处理外发消息的 SMTP 服务器。 当选择 *Generic SMTP* 作为电子邮件提供商时，此字段可用。
SMTP server port	设置用于处理外发消息的 SMTP 服务器端口。 当选择 *Generic SMTP* 作为电子邮件提供商时，此字段可用。

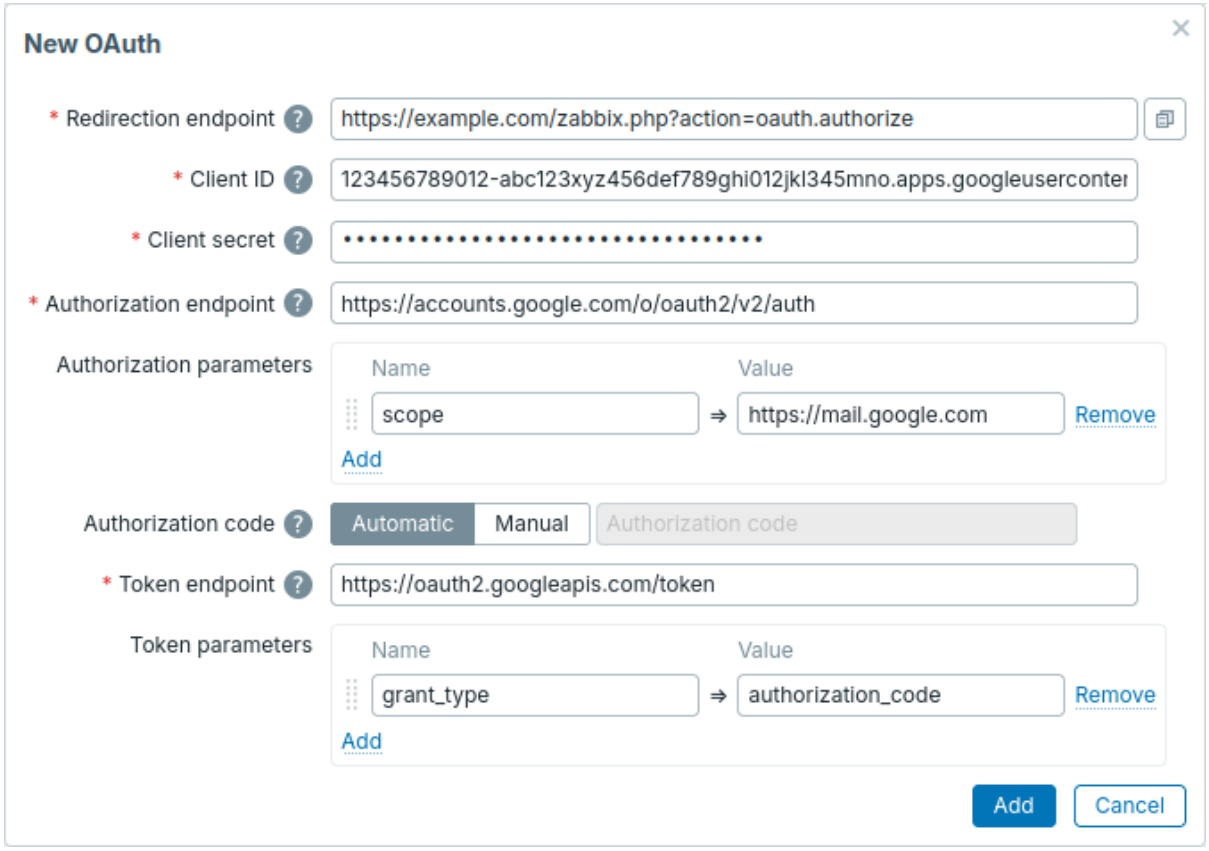
参数	描述
Email	此处输入的地址将用作所发送消息的 **From** 地址。 支持在实际电子邮件地址中添加发件人显示名称（例如，`Zabbix_info <zabbix@company.com>`）。 与 RFC 5322 允许的显示名称相比，Zabbix 电子邮件中的显示名称存在一些限制，示例如下。 有效示例： `zabbix@company.com`（仅电子邮件地址；不需要尖括号） `Zabbix_info <zabbix@company.com>`（显示名称后跟尖括号中的电子邮件地址） `Ω-monitoring <zabbix@company.com>`（显示名称中包含 UTF-8 字符） 无效示例： `Zabbix HQ zabbix@company.com`（存在显示名称，但电子邮件地址未使用尖括号括起） `"Zabbix@<H(comment)Q>" <zabbix@company.com>`（尽管根据 RFC 5322 有效，但 Zabbix 电子邮件不支持带引号的字符对和注释）。
SMTP helo	设置正确的 SMTP helo 值，通常为域名。 如果为空，将发送电子邮件的域名（即 *Email* 字段中 @ 后面的内容）。如果无法获取域名，将记录调试级别的警告，并将服务器主机名作为 HELO 命令的域发送。 当选择 *Generic SMTP* 作为电子邮件提供商时，此字段可用。
Connection security	选择连接安全级别： **None** - 不使用 CURLOPT_USE_SSL 选项 **STARTTLS** - 使用值为 CURLUSESSL_ALL 的 CURLOPT_USE_SSL 选项 **SSL/TLS** - CURLOPT_USE_SSL 的使用是可选的。
SSL verify peer	选中此复选框以验证 SMTP 服务器的 SSL 证书。 应将服务器配置指令"SSLCALocation" 的值放入 CURLOPT_CAPATH，以进行证书验证。 这会设置 cURL 选项 CURLOPT_SSL_VERIFYPEER。
SSL verify host	选中此复选框，以验证 SMTP 服务器证书的 *Common Name* 字段或 *Subject Alternate Name* 字段是否匹配。 这会设置 cURL 选项 CURLOPT_SSL_VERIFYHOST。
Authentication	选择身份验证级别： **None** - 不设置 cURL 选项 **Username and password** - 表示使用"AUTH=*"，由 cURL 选择身份验证机制 **OAuth** - OAuth 身份验证 *Office365 relay* 电子邮件提供商不支持 OAuth 身份验证。
Username	用于身份验证的用户名。 这会设置 CURLOPT_USERNAME 的值。 支持用户宏。
Password	用于身份验证的密码。 这会设置 CURLOPT_PASSWORD 的值。 支持用户宏。 媒介类型配置导出时，此处输入的值将按原样导出。为避免在导出文件中暴露敏感凭据，请使用用户宏（例如 `{$EMAIL_SMTP_PASSWORD}`）代替明文密码；导出时将导出宏引用，而机密值应在目标系统上重新配置。
OAuth tokens	单击 *Configure*，在新窗口中设置用于获取OAuth 令牌 的参数。 仅当在 *Authentication* 字段中选择"OAuth" 时，此字段才可用。
Message format	选择消息格式： **HTML** - 以 HTML 格式发送； **Plain text** - 以纯文本格式发送。

Attention:
要启用 SMTP 身份验证，Zabbix 服务器必须使用 `--with-libcurl` 编译选项进行编译，该选项需要 libcurl（有关版本详细信息，请参阅服务器要求）。

有关如何配置默认消息和告警处理选项的详细信息，请参阅通用媒介类型参数。

OAuth 令牌

检索 OAuth 令牌需要以下参数：



Warning:
导出电子邮件媒体类型配置时，Client secret 会以明文形式导出。

Parameter	Description
Redirection endpoint	输入 Zabbix 前端 URL，OAuth 服务将通过该 URL 重定向回 OAuth 授权（格式为 `https://<zabbix-frontend-url>/zabbix.php?action=oauth.authorize`）。 当定义了frontend URL 时，新建电子邮件媒体类型会自动设置该值。
Client ID	输入在 OAuth 授权服务器中注册的客户端应用的唯一标识符。
Client secret	输入在 OAuth 授权服务器中注册的客户端应用的私密密钥。
Authorization endpoint	输入用于请求用户授权的 OAuth 授权服务器 URL。
Authorization parameters	输入授权端点的参数。
Authorization code	输入授权码： **Automatic** - 代码将通过重定向页面自动获取； **Manual** - 如果自动获取失败，请手动输入代码。
Token endpoint	输入 OAuth 授权服务器 URL，用于将授权码交换为访问令牌和刷新令牌。
Token parameters	输入访问令牌的参数。

Note:
OAuth 令牌的检索为 **Gmail**、**Gmail relay** 和 **Office365** 电子邮件提供商提供了自动化功能。此时只需提供 *Redirection endpoint*、*Client ID* 和 *Client secret* 参数值。Zabbix 会自动填充其他所需值（请参见按提供商划分的 OAuth URL 默认值）。

OAuth 访问令牌和刷新令牌获取

提交 OAuth 参数后：

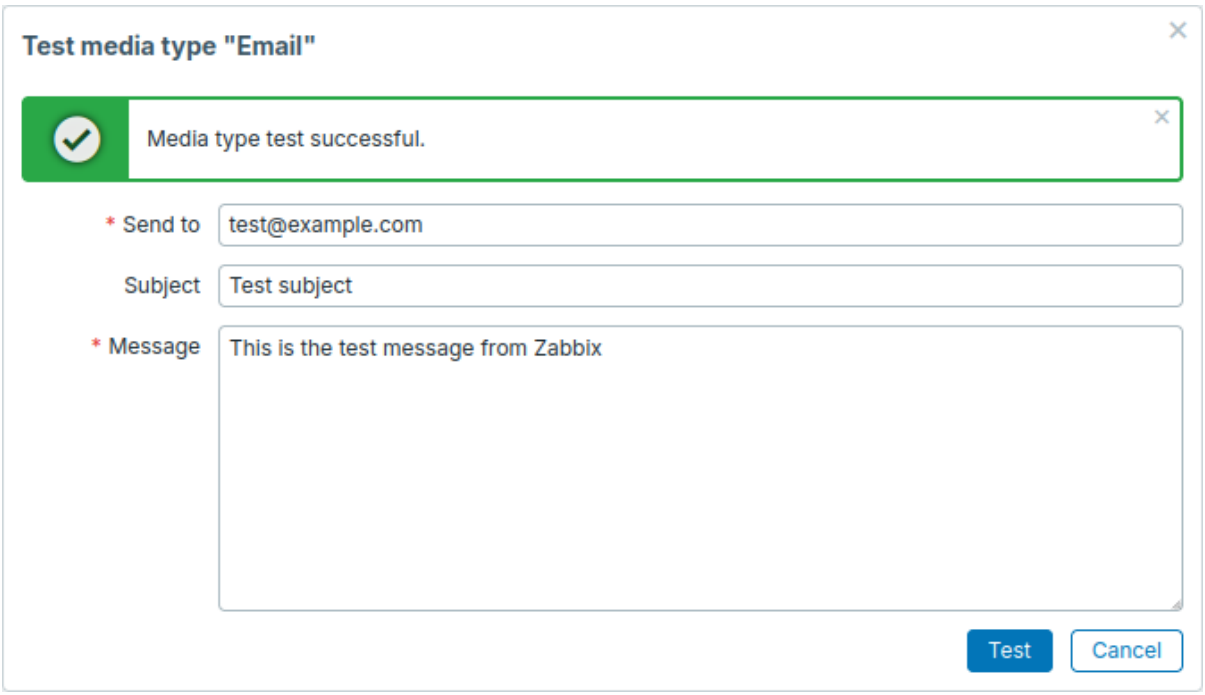
1. 将打开一个浏览器弹窗，引导用户访问授权端点。
2. 用户在 OAuth 服务中授权 Zabbix。
3. OAuth 服务会将用户重定向到 Zabbix 前端操作 `oauth.authorize`，并附带授权码和作用域值。
4. 随后，Zabbix 将向令牌端点发出请求，以使用授权码换取访问令牌和刷新令牌。

测试

要测试已配置的 email 媒介类型是否正常工作：

1. 在媒介类型的列表 中找到相关的 email。
2. 单击列表最后一列中的 *Test*（将打开一个测试窗口）。
3. 输入 *Send to* 收件人地址、消息正文，以及可选的主题。
4. 单击 *Test* 发送测试消息。

测试成功或失败的消息将显示在同一窗口中：



用户媒介

配置好电子邮件媒介类型后，转到 *Users* > *Users* 部分，编辑用户配置文件，为该用户分配电子邮件媒介。有关设置用户媒介的步骤（这些步骤适用于所有媒介类型），请参见Media types 页面。

自动化 Gmail/Office365 媒体类型

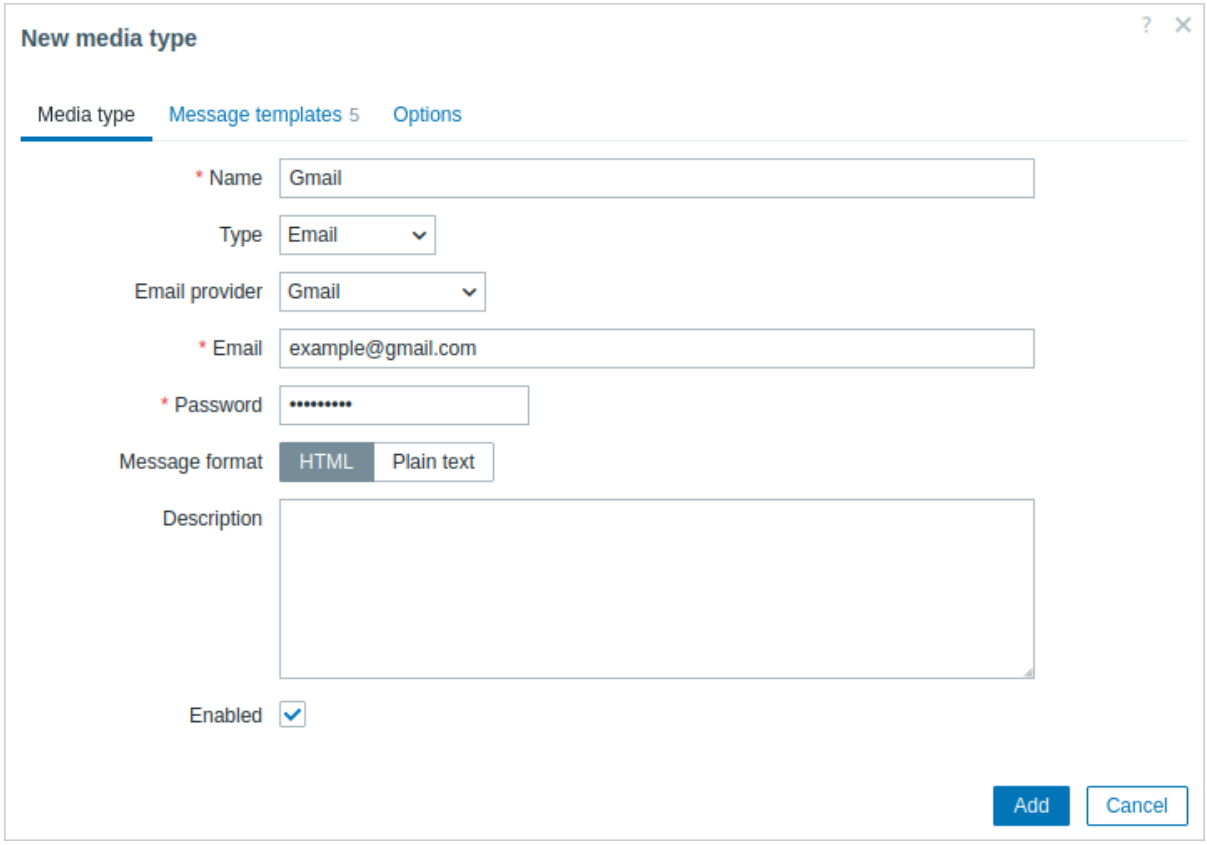
概述

Gmail 或 Office365 用户可以从媒体类型配置中的自动化功能中受益。

配置

电子邮件媒介类型配置中的电子邮件提供商字段允许您选择 Gmail 和 Office 365 的预配置选项。

选择与 Gmail/Office365 相关的选项时，只需提供发件人电子邮件地址和密码，即可创建可正常工作的媒介类型。



提供电子邮件地址和密码后，Zabbix 将能够使用实际值/推荐值自动填充 Gmail/Office365 媒介类型所需的所有设置，即 *SMTP* 服务器、*SMTP* 服务器端口、*SMTP helo* 和连接安全性。由于采用了此自动化机制，这些字段甚至不会显示；不过，您可以在媒介类型列表中查看 SMTP 服务器和电子邮件详细信息（请参见详细信息列）。

另请注意：

- 中继选项不需要密码。
- 对于 Office365 中继，将使用所提供电子邮件地址的域名动态填充 SMTP 服务器(即，将 `example-com.mail.protection.outlook.com` 中的 “example.com” 替换为实际值)。

OAuth 令牌

OAuth 令牌 的获取功能为 *Gmail*、*Gmail relay* 和 *Office365* 电子邮件提供商提供了自动化特性。



只需提供一组有限的参数即可获取 OAuth 访问令牌和刷新令牌——重定向端点、客户端 *ID* 和 客户端密钥。Zabbix 将自动填充其他必需的值（请参见按提供商划分的 OAuth URL 默认值）。

请注意，也可以使用Generic SMTP 表单为这些提供商设置 OAuth 授权。

Note:
在 Office365 中，必须为每个用户/共享邮箱或整个租户启用 SmtpClientAuthentication（默认设置下未启用）。

按提供商划分的 OAuth URL 默认值

下表列出了 Zabbix 使用的各提供商默认 OAuth URL 值和参数。

参数		Gmail	Office365	通用 SMTP
授权端点		https://accounts.google.com/o/oauth2/auth	https://login.microsoftonline.com/common/oauth2/v2.0/authorize	无默认值
	scope	https://mail.google.com	https://outlook.office.com/SMTP.Send, offline_access	无默认值
	access_type	offline	未使用	无默认值
	prompt	consent	未使用	无默认值
	redirect_uri	重定向端点字段的值会由代码自动添加。不会列在 授权端点参数中。		
	state	用于标识正在使用 OAuth 令牌更新的媒介类型的唯一哈希值会由代码自动添加。不会列在 授权端点参数中。		
令牌端点		https://oauth2.googleapis.com/token	https://login.microsoftonline.com/common/oauth2/v2.0/token	无默认值
	grant_type	authorization_code	authorization_code	无默认值
	redirect_uri	重定向端点字段的值会由代码自动添加。不会列在 令牌端点参数中。		
	state	用于标识正在使用 OAuth 令牌更新的媒介类型的唯一哈希值会由代码自动添加。不会列在 令牌端点参数中。		

适用于个人账户的 Office365 变通方法

Office365 个人账户不支持在重定向 URL 中使用查询字符串。

为了解决此问题，Apache Web 服务器管理员可以将以下重写规则添加到 .htaccess 文件中：

```
RewriteEngine On
RewriteRule ^/oauth\.authorize$ /zabbix.php?action=oauth.authorize [QSA,L,PT]
```

这样即可使用如下简化的重定向 URL：

```
http://server-name/zabbix/oauth.authorize
```

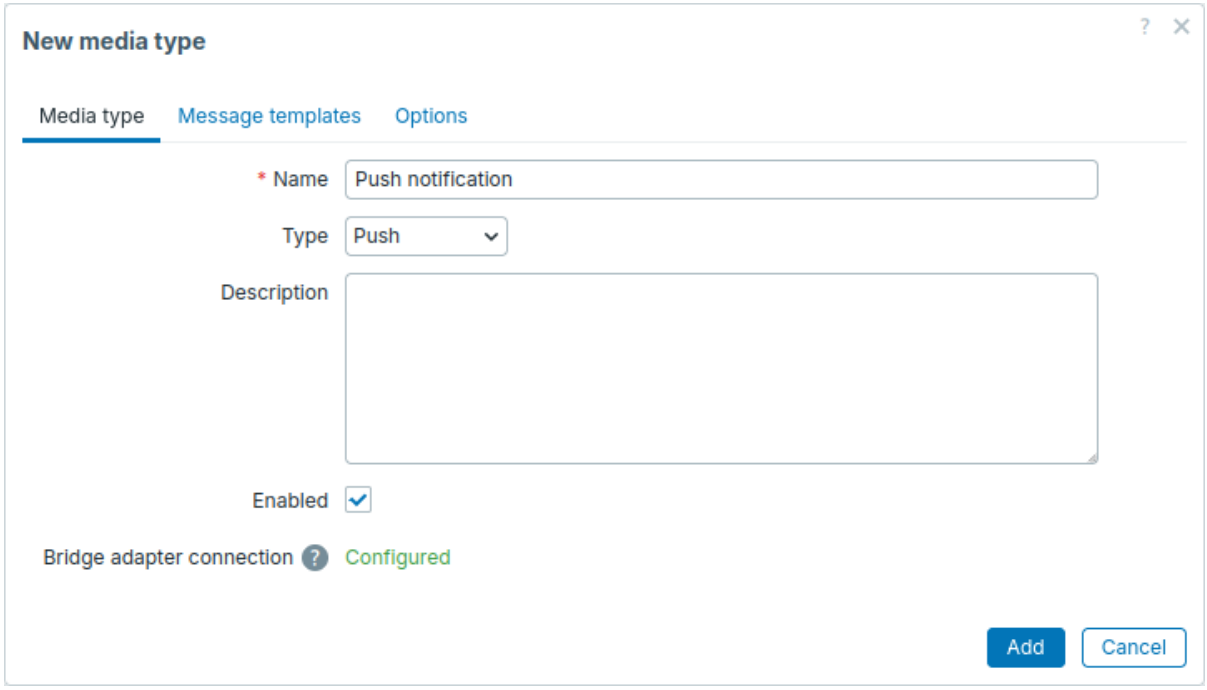
而不是使用带有查询参数的 URL，从而确保与 Office365 个人账户兼容。

推送通知

此媒介类型允许 Zabbix 服务器向移动设备上的Zabbix 移动应用 发送推送通知。

配置

创建媒体类型时，将 *Type* 选择为 *Push*：



配置表单包含通用媒体类型参数、消息模板和选项。

表单还会显示 Zabbix bridge adapter 是否已配置。
 如果未配置该适配器，您仍然可以创建媒体类型，但它将无法正常工作。

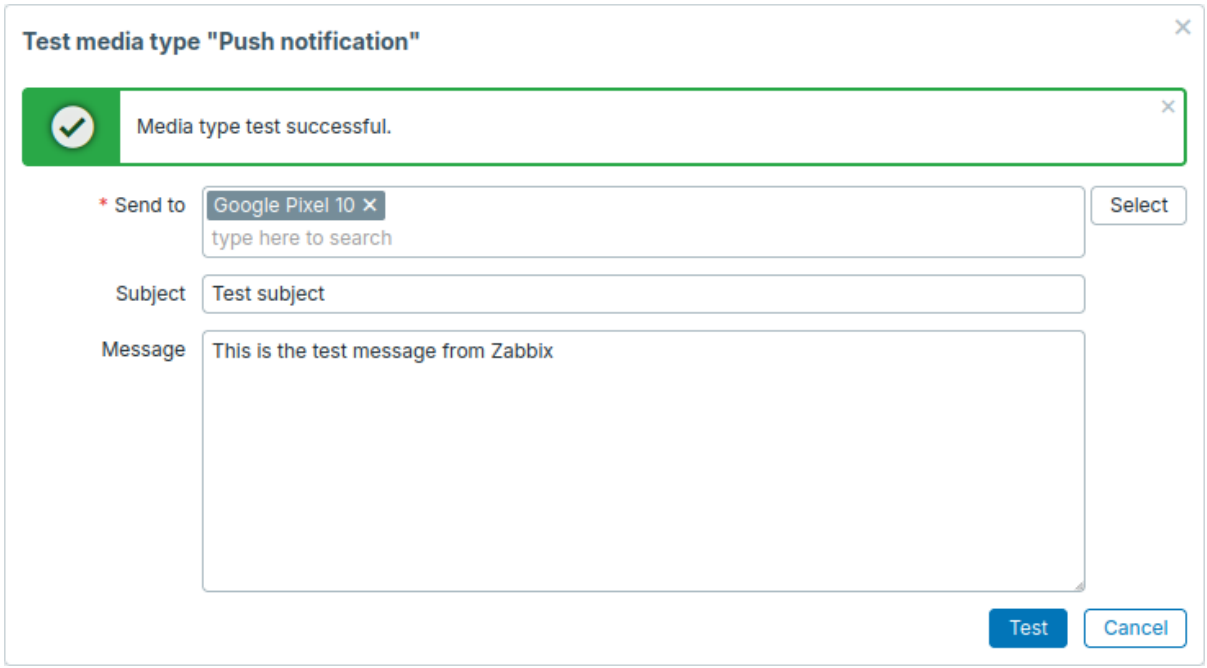
创建并启用媒体类型后，请为用户添加设备，以便自动将该媒体类型添加到该用户（或手动添加用户媒体）。这样，用户便可以在移动设备上接收推送通知。

测试

要测试此媒体类型是否正常工作：

1. 在媒体类型列表中找到该媒体类型。
2. 点击列表最后一列中的测试，打开测试表单。
3. 在发送至字段中，选择一个设备。
4. 点击测试，向所选设备发送测试推送通知。

成功或失败消息将显示在同一窗口中：



短信

概述

Zabbix 支持使用连接到 Zabbix 服务器串口的串行 GSM 调制解调器发送短信。

请确保：

- 串行设备的速率（在 Linux 下通常为 /dev/ttyS0）与 GSM 调制解调器的速率一致。Zabbix 不会设置串行链路的速率，而是使用默认设置。
- 'zabbix' 用户对串行设备具有读/写访问权限。运行命令 ls -l /dev/ttyS0 以查看串行设备的当前权限。
- GSM 调制解调器已输入 PIN 码，并且在断电重启后仍能保留该设置。或者，您也可以在 SIM 卡上禁用 PIN。可以在终端软件中执行命令 AT+CPIN="NNNN"（NNNN 为您的 PIN 码，且必须保留引号）来输入 PIN，例如 Unix minicom 或 Windows HyperTerminal。

Zabbix 已使用以下 GSM 调制解调器进行过测试：

- Siemens MC35
- Teltonika ModemCOM/G10

要将短信配置为消息的传递渠道，您还需要将短信配置为媒体类型，并为用户输入相应的电话号码。

配置

要将 SMS 配置为媒介类型：

1. 转到 *Alerts* > *Media types*。
2. 单击 *Create media type*（或在预定义媒介类型列表中单击 *SMS*）。
3. 在表单中输入 SMS 媒介类型参数。

以下参数专用于 SMS 媒介类型：

Parameter	Description
GSM modem	设置 GSM 调制解调器的串行设备名称。 此处输入的路径将根据SMSDevices 服务器参数（如果已指定）进行验证。

有关如何配置默认消息和告警处理选项的详细信息，请参阅通用媒介类型参数。请注意，无法并行处理 SMS 通知的发送。

用户媒介

配置 SMS 媒介类型后，转到 用户 > 用户部分，编辑用户配置文件，为用户分配 SMS 媒介。

所有媒介类型通用的用户媒介设置步骤，请参阅媒介类型页面。

自定义告警脚本

概述

如果您对现有用于发送告警的媒体类型不满意，还有一种替代方法。您可以创建一个脚本，以您自己的方式处理通知。

自定义告警脚本在 Zabbix 服务器上执行。这些脚本必须位于服务器配置文件中AlertScriptsPath 参数指定的目录下。

下面是一个自定义告警脚本示例：

```
#####!/bin/bash

to=$1
subject=$2
body=$3
host=$4
value=$5

cat <<EOF | mail -s "$subject" "$to"
$body

Host: $host
Value: $value
EOF
```

Attention:
Zabbix 会检查已执行命令和脚本的退出码。任何不等于 **0** 的退出码都将被视为命令执行 错误。在这种情况下，Zabbix 将尝试重新执行失败的操作。

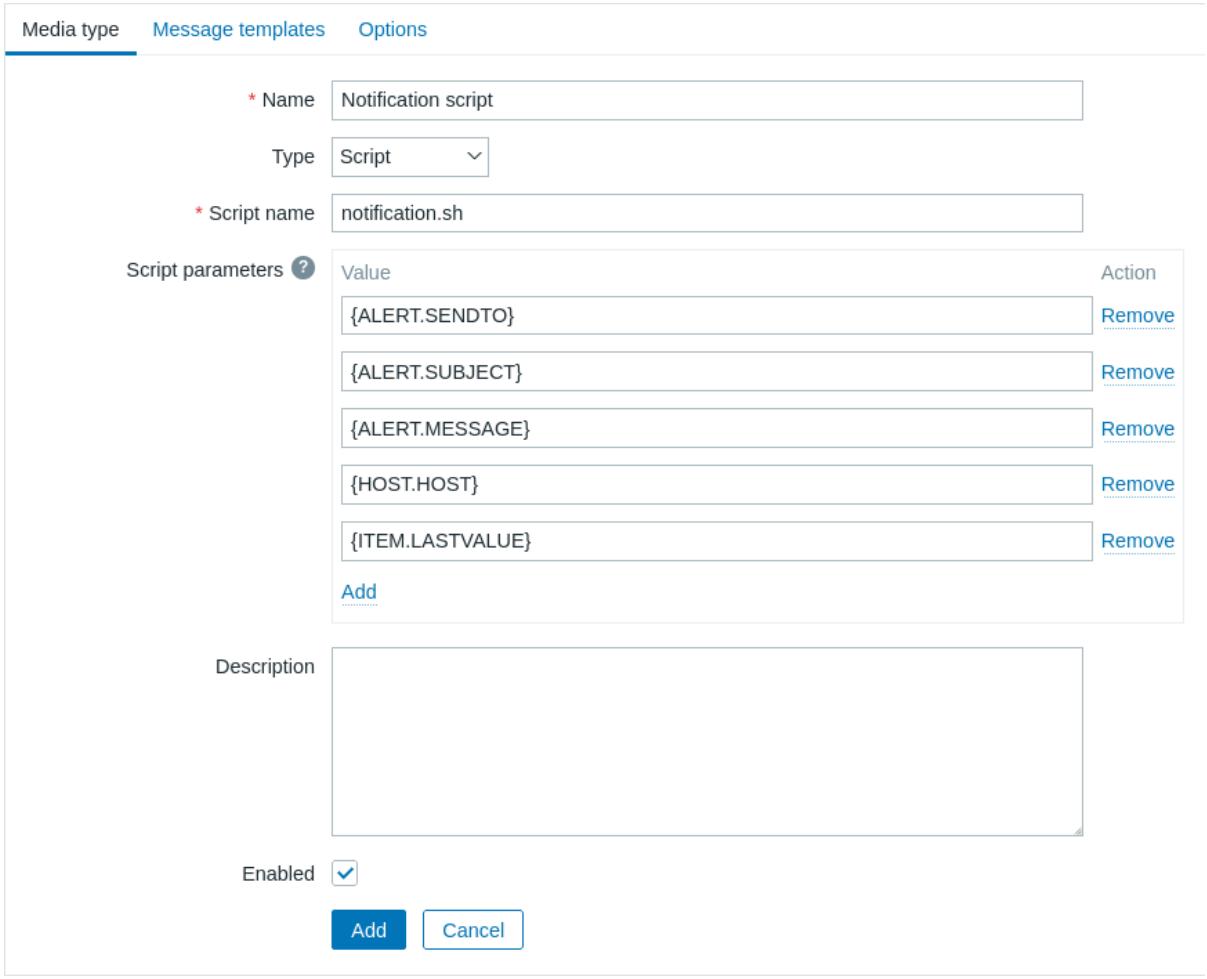
脚本不会保留或创建环境变量，因此应显式处理这些变量。

配置

要将自定义告警脚本配置为媒体类型：

1. 转到 告警 > 媒体类型。
2. 单击 创建媒体类型。
3. 在表单中输入脚本媒体类型参数。

媒体类型选项卡包含常规媒体类型属性：



所有必填输入字段均以红色星号标记。

以下参数专用于脚本媒体类型：

参数	描述
脚本名称	输入脚本文件的名称（例如 notification.sh）。该文件位于AlertScriptsPath 服务器配置参数指定的目录中。
脚本参数	添加可选脚本参数。这些参数将按照定义顺序作为命令行参数传递给脚本。 脚本参数支持 {ALERT.SENDTO}、{ALERT.SUBJECT}、{ALERT.MESSAGE} 宏，以及通知中支持的所有宏 和用户宏。

有关如何配置默认消息和告警处理选项的详细信息，请参阅通用媒体类型参数。

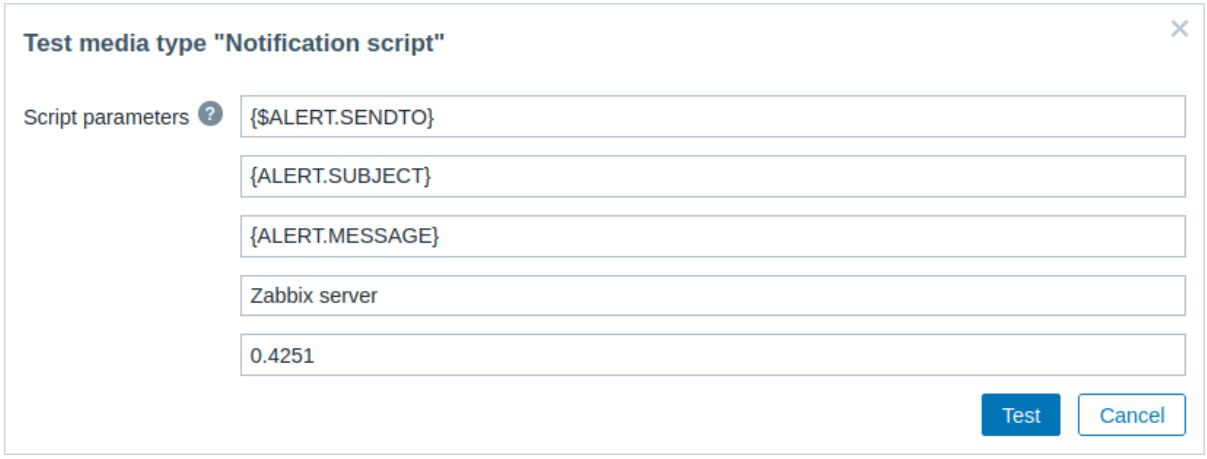
Warning:
即使告警脚本不使用默认消息，仍必须定义此媒体类型所使用的操作类型的消息模板。否则，将不会发送通知。

Attention:
如果配置了多个脚本媒体类型，alerter 进程可能会并行处理这些脚本。alerter 进程的总数受服务器配置文件中的`StartAlerters` 参数限制。

测试

要测试已配置的脚本 media type：

1. 在 media type 的列表 中找到相关脚本。
2. 在列表的最后一列点击 *Test*；测试表单将会在弹出窗口中打开。测试表单将包含与该脚本 media type 配置相同数量的参数。
3. 如有需要，编辑脚本参数值。编辑仅影响测试过程；实际值不会被更改。
4. 点击 *Test*。



Note:
在测试已配置的脚本 media type 时，{ALERT.SENDTO}、{ALERT.SUBJECT}、{ALERT.MESSAGE} 和用户宏将解析为其值，但与事件相关的宏（例如 {HOST.HOST}、{ITEM.LASTVALUE} 等）不会被解析，因为在测试期间没有相关事件可用于获取详细信息。请注意，{ALERT.SUBJECT} 和 {ALERT.MESSAGE} 中的宏也不会被解析。例如，如果 {ALERT.SUBJECT} 的值由"Problem: {EVENT.NAME}" 组成，那么 {EVENT.NAME} 宏将不会被解析。

用户媒介

配置好媒介类型后，转到 *Users* > *Users* 部分，通过将此媒介类型分配给用户来编辑用户配置文件。适用于所有媒介类型的用户媒介设置步骤，请参见Media types 页面。

请注意，在定义用户媒介时，*Send to* 字段不能为空。如果警报脚本中未使用此字段，请输入任意支持字符的组合，以绕过校验要求。

webhook

概述

webhook 媒介类型可用于使用自定义 JavaScript 代码发起 HTTP 调用，从而与帮助台系统、聊天工具或即时通讯软件等外部软件进行直接集成。您可以选择导入由 Zabbix 提供的集成，或从头开始创建自定义集成。

集成

以下集成可用，允许使用预定义的 webhook 媒体类型将 Zabbix 通知推送到：

- brevis.one
- Discord
- Event-Driven Ansible
- Express.ms messenger
- GitHub
- GLPI
- IBM Maximo Service Request
- iLert
- iTop
- Jira
- Jira Service Management
- ManageEngine ServiceDesk
- Mantis Bug Tracker
- Mattermost
- MS Teams Workflows
- LINE
- Opsgenie
- OTRS CE

- Pagerduty
- Pushover
- Redmine
- Rocket.Chat
- ServiceNow
- SIGNL4
- Slack
- SolarWinds
- SysAid
- Telegram
- TOPdesk
- VictorOps
- Zammad
- Zendesk

Note:
除此处列出的服务外，Zabbix 还可以与 **Spiceworks** 集成（无需 webhook）。要将 Zabbix 通知转换为 Spiceworks 工单，请创建一个电子邮件媒体类型，并在指定 Zabbix 用户的个人资料设置中输入 Spiceworks 服务台电子邮件地址（例如 help@zabbix.on.spiceworks.com）。

配置

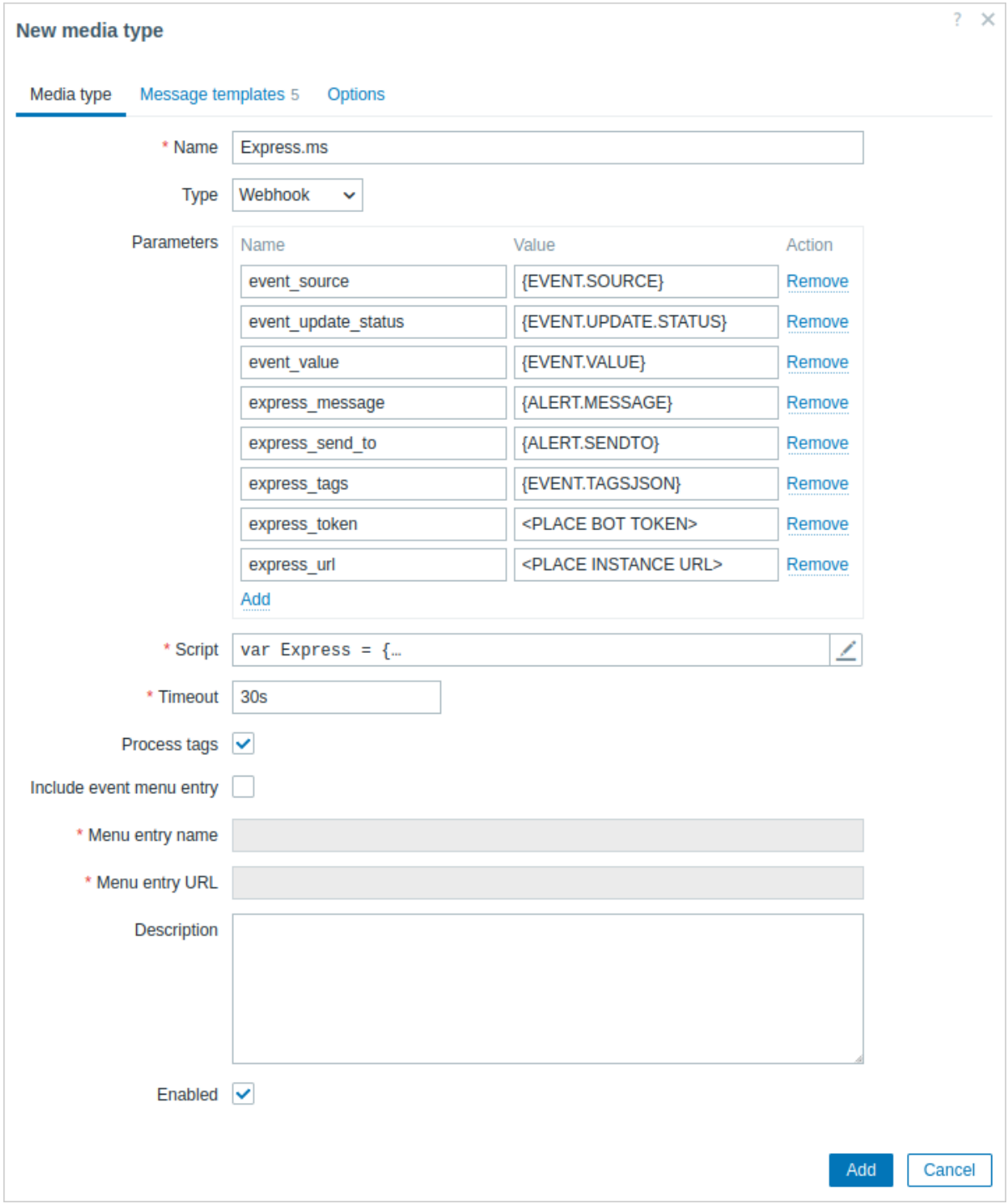
要开始使用 webhook 集成：

1. 在下载的 Zabbix 版本的 `templates/media` 目录中找到所需的 `.yaml` 文件，或从 Zabbix git 仓库下载。
2. 将文件导入到您的 Zabbix 安装中。
   webhook 将显示在媒体类型列表中。
3. 根据 `Readme.md` 文件中的说明配置 webhook（您可以点击上方的 webhook 名称，以快速访问 `Readme.md`）。

要从头创建自定义 webhook：

1. 转到告警 > 媒体类型。
2. 点击创建媒体类型。
3. 在表单中输入 webhook 媒体类型参数。

媒体类型选项卡包含此媒体类型特有的各种属性：



所有必填输入字段均以红色星号标记。

以下参数是 webhook 媒体类型特有的：

参数	描述
参数	将 webhook 变量指定为属性和值对。 对于预配置的 webhook，参数列表因服务而异。有关参数说明，请查看 webhook 的 *Readme.md* 文件。 对于新 webhook，默认包含一些常用变量（URL:<empty>、HTTPProxy:<empty>、To:{ALERT.SENDTO}、Subject:{ALERT.SUBJECT}、Message:{ALERT.MESSAGE}），您可以保留或删除这些变量。 Webhook 参数支持用户宏、问题通知中支持的所有宏，以及 {ALERT.SENDTO}、{ALERT.SUBJECT} 和 {ALERT.MESSAGE} 宏。 如果指定 HTTP proxy，此字段支持与监控项配置中的HTTP proxy 字段相同的功能。proxy 字符串可以添加 [scheme]:// 前缀，以指定所使用的 proxy 类型（例如 https、socks4、socks5；请参阅文档）。

参数	描述
脚本	在点击参数字段或其旁边的铅笔图标时打开的模态编辑器中输入 JavaScript 代码。此代码将执行 webhook 操作。 该脚本是一个接受参数-值对的函数代码。应使用 JSON.parse() 方法将值转换为 JSON 对象，例如：`var params = JSON.parse(value);`。 该代码可以访问所有参数，可以执行 HTTP GET、POST、PUT 和 DELETE 请求，支持 CONNECT、PATCH、HEAD、OPTIONS 和 TRACE 等其他方法，并可控制 HTTP 标头和请求正文。 脚本必须包含 return 运算符，否则将无效。它可以返回 OK 状态以及可选的标签和标签值列表（请参阅处理标签选项），也可以返回错误字符串。 恢复事件（无论是自动生成的，还是手动关闭后生成的）由服务器创建，并包含已解决的事件标签（包括从模板、主机和触发器继承的标签）。Webhook 脚本在告警创建后执行；因此，webhook 脚本返回的标签仅会在初始告警创建后添加，不会出现在初始问题消息或即时恢复消息的{EVENT.TAGS}和{EVENT.RECOVERY.TAGS}宏中。 注意：建议使用局部变量（例如 `var local = 1`），而不是全局变量（例如 `global = 1`），以确保每个脚本使用自己的数据，并避免同时调用之间发生冲突（请参阅已知问题）。 另请参阅：Webhook 开发指南、Webhook 脚本示例、其他 JavaScript 对象。
超时	JavaScript 执行超时时间（1-60 秒，默认 30 秒）。 支持时间后缀（例如 30s、1m）。
处理标签	选中复选框，将返回的 JSON 属性值作为标签进行处理。这些标签会添加到现有的问题标签中。 请注意，使用 webhook 标签时，webhook 必须返回一个至少包含空标签对象的 JSON 对象：`var result = {tags: {}};` 可返回的标签示例：*jira-id:prod-1234*、*responsible:John Smith*、*processed:<no value>*
包含事件菜单项	选中复选框，在事件菜单中包含一个指向已创建外部工单的条目。 每个已启用且选中此复选框的 webhook 都会包含一个条目。请注意，如果菜单项名称和菜单项 *URL* 参数包含任何{EVENT.TAGS.<tag name>}宏，则仅当这些宏可以解析时（即事件已定义这些标签）才会包含条目。 选中后，不应使用此 webhook 向不同用户发送通知（请考虑改为创建一个专用用户），也不应在多个告警操作中用于单个问题事件。
菜单项名称	指定菜单项名称。 支持{EVENT.TAGS.<tag name>}宏。 仅当选中包含事件菜单项时，此字段才为必填字段。
菜单项 *URL*	指定菜单项的底层 URL。 支持{EVENT.TAGS.<tag name>}宏。 仅当选中包含事件菜单项时，此字段才为必填字段。

有关如何配置默认消息和告警处理选项的详细信息，请参阅常用媒体类型参数。

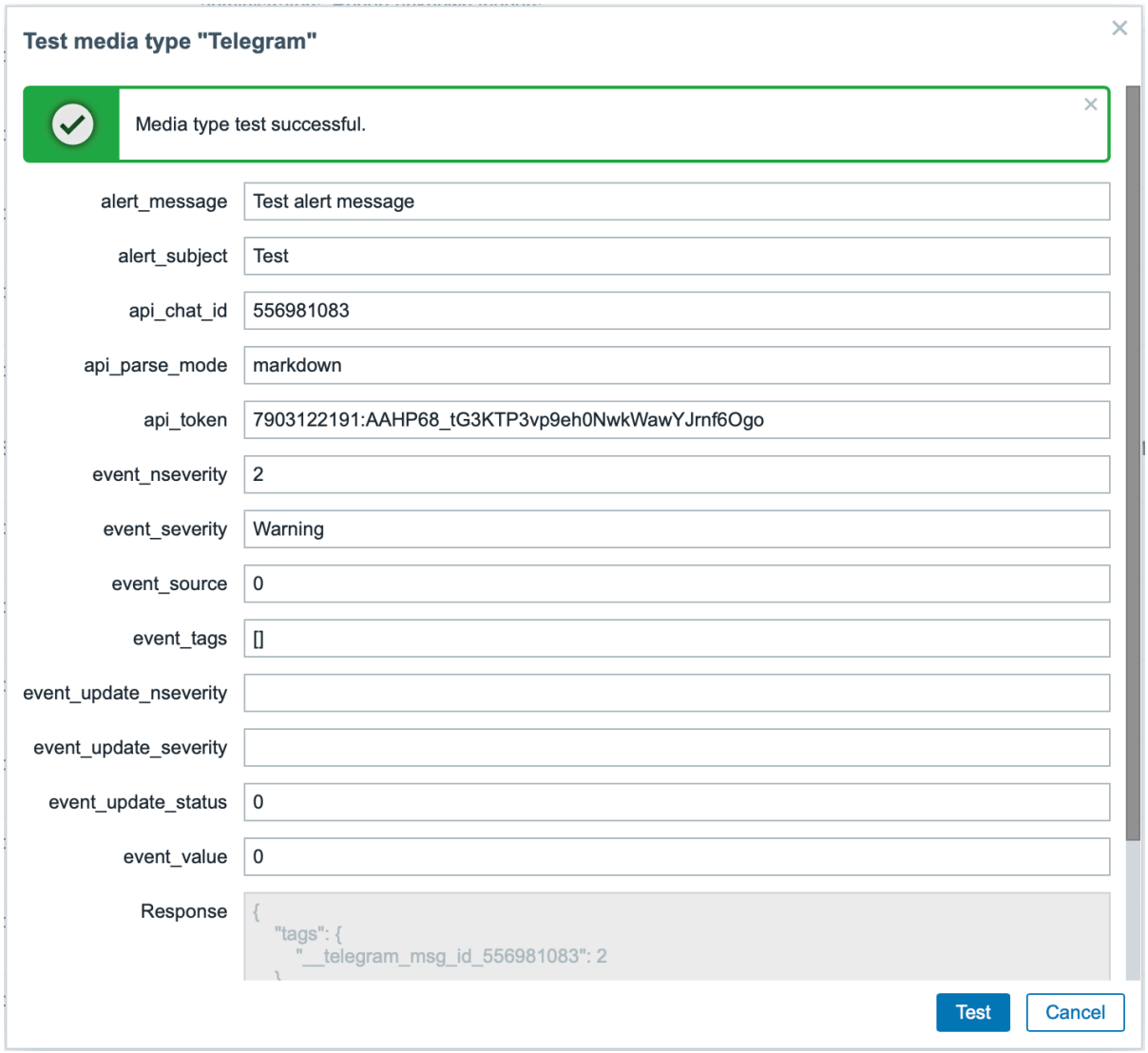
Warning:
即使 webhook 不使用默认消息，仍必须定义此 webhook 所使用操作类型的消息模板。

测试

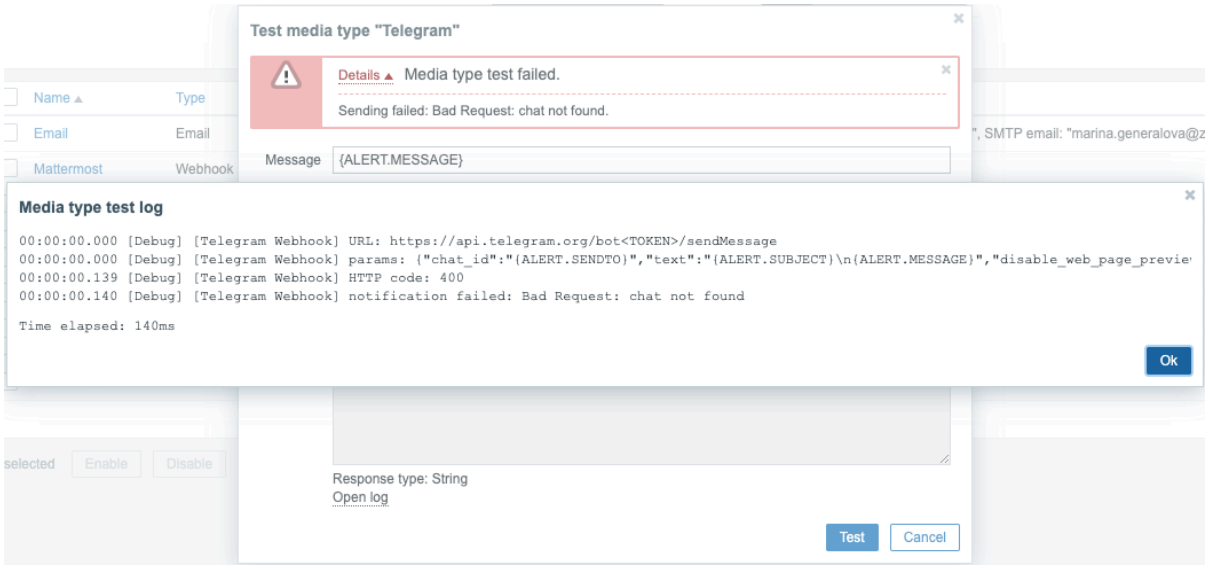
要测试已配置的 webhook 媒体类型：

1. 在媒体类型的列表中找到相关 webhook。
2. 单击列表最后一列中的测试（将打开测试窗口）。
3. 根据需要编辑 webhook 参数值。
   将宏替换为示例值；否则，宏将无法解析，测试也会失败。
4. 单击测试。

在测试窗口中替换或删除值只会影响测试过程，实际的 webhook 属性值不会发生变化。



若要在不离开测试窗口的情况下查看媒体类型测试日志条目，请单击打开日志（将打开新的弹出窗口）。



如果 **webhook** 测试成功：

- 将显示 *"Media type test successful."* 消息。
- 服务器响应将显示在灰色的 *Response* 字段中。
- *Response* 字段下方会标明响应类型（JSON 或 String）。

如果 **webhook** 测试失败：

- 将显示 *"Media type test failed."* 消息，后跟其他失败详情。

用户媒体

配置媒体类型后，转到 用户 > 用户部分，将 webhook 媒体分配给现有用户，或创建一个代表该 webhook 的新用户。为现有用户设置用户媒体的步骤适用于所有媒体类型，详见媒体类型页面。

如果 webhook 使用标签存储工单\消息 ID，请避免将同一个 webhook 作为媒体分配给不同用户，因为这可能会导致 webhook 错误（适用于大多数使用 *Include event menu entry* 选项的 webhook）。在这种情况下，最佳做法是创建一个专用用户来代表该 webhook：

1. 配置 webhook 媒体类型后，转到 用户 > 用户部分，创建一个专用的 Zabbix 用户来代表该 webhook - 例如，为 Slack webhook 创建用户名为 *Slack* 的用户。除媒体以外的所有设置均可保留默认值，因为该用户不会登录 Zabbix。
2. 在用户配置文件中，转到 媒体选项卡，并添加 webhook，填写所需的联系信息。如果 webhook 不使用 *Send to* 字段，请输入任意组合的受支持字符，以绕过验证要求。
3. 至少授予该用户对所有需要发送告警的主机的读取权限。

配置告警动作时，在操作详情的 *Send to users* 字段中添加该用户 - 这将告知 Zabbix 使用该 webhook 发送此动作的通知。

配置告警动作

动作决定应通过 webhook 发送哪些通知。涉及 webhook 的配置动作步骤与所有其他媒体类型相同，但有以下例外：

- 如果某个 webhook 使用webhook 标签来存储工单\消息 ID 并处理更新\解决操作，请避免在同一个问题事件的多个告警动作中使用同一个 webhook。如果 {EVENT.TAGS.<tag name>} 存在并在 webhook 中被更新，则其结果值将是未定义的。为避免这种情况，请在 webhook 中使用新的标签名称来存储更新后的值。这适用于 Zabbix 提供的 Jira、Jira Service Desk、Mattermost、Opsgenie、OTRS、Redmine、ServiceNow、Slack、Zammad 和 Zendesk webhooks，以及大多数使用 *Include event menu entry* 选项的 webhooks。不过请注意，同一个 webhook 可以在同一动作的多个操作或升级步骤中使用，也可以用于不同的动作，只要这些动作不会因不同的条件而由同一个问题事件触发。
- 在针对内部事件的动作中使用 webhook 时，请确保勾选 *Custom message* 复选框，并在动作操作配置中定义自定义消息。否则，将不会发送通知。

webhook 脚本示例

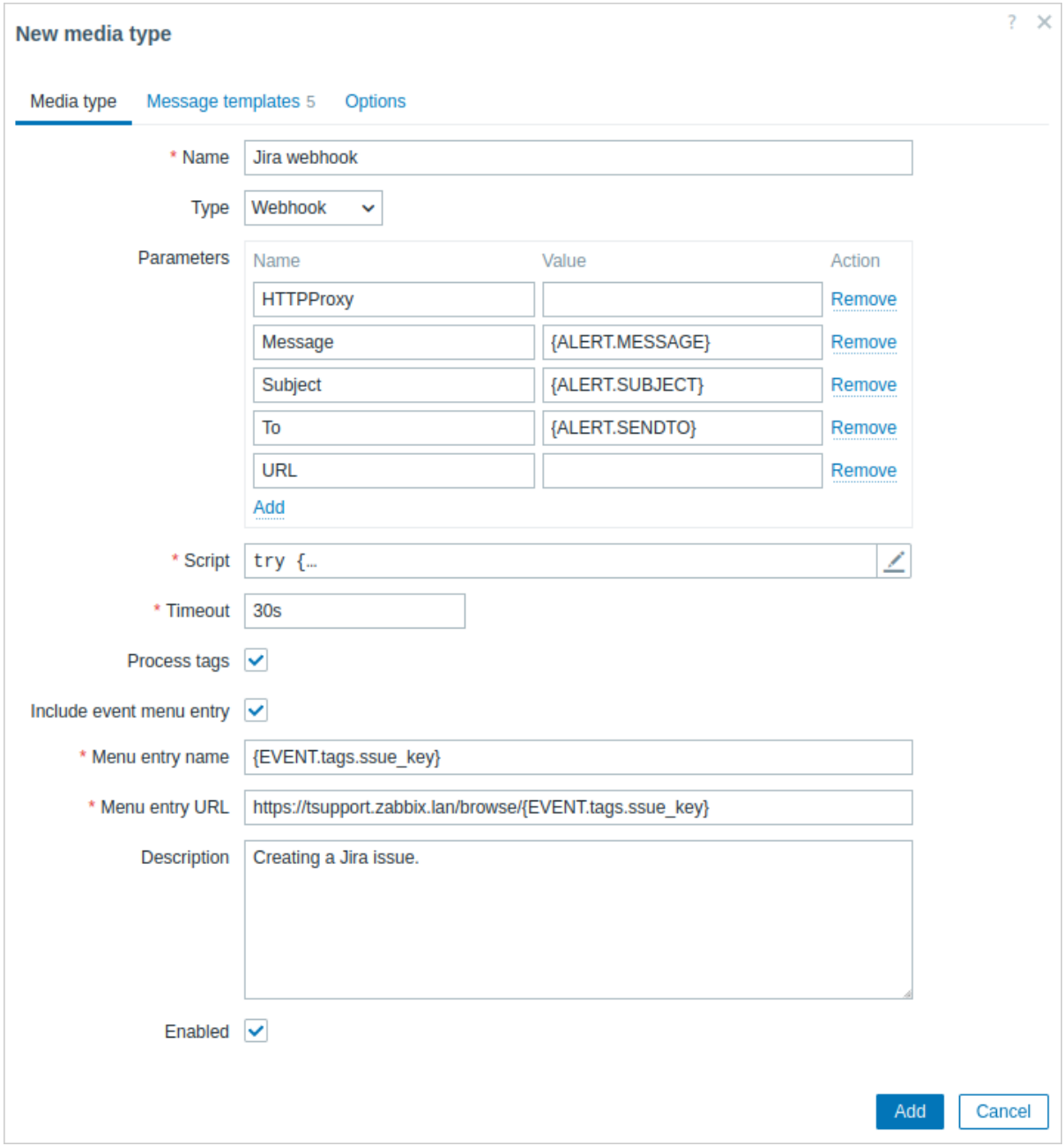
概述

尽管 Zabbix 开箱即提供了大量可用的 webhook 集成，您仍可能希望自行创建 webhook。本节提供自定义 webhook 脚本的示例（用于 *Script* 参数）。有关其他 webhook 参数的说明，请参见webhook 部分。

Attention:
不要在预处理 JavaScript 中使用未声明的赋值。请使用 `var` 声明局部变量。

恢复事件（无论是自动生成，还是在手动关闭 后生成）都包含已解决的事件标签（包括从模板、主机和触发器继承的标签）。webhook 脚本在告警创建后执行；因此，webhook 脚本返回的标签会在初始告警创建之后才应用，可能不会出现在触发该 webhook 的告警中。如果某个集成要求 webhook 生成的标签必须出现在初始通知中，请从服务器获取事件标签（例如，使用Event API），或者将 webhook 生成的标签存储在外部持久化存储中并在那里进行关联。

Jira webhook（自定义）



此脚本将创建一个 JIRA 问题，并返回一些有关已创建问题的信息。

```
try {
    Zabbix.log(4, '[ Jira webhook ] Started with params: ' + value);

    var result = {
            'tags': {
                'endpoint': 'jira'
            }
        },
        params = JSON.parse(value),
        req = new HttpRequest(),
        fields = {},
        resp;

    if (params.HTTPProxy) {
        req.setProxy(params.HTTPProxy);
    }

    req.addHeader('Content-Type: application/json');
    req.addHeader('Authorization: Basic ' + params.authentication);
```

```
    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {key: params.project_key};
    fields.issuetype = {id: params.issue_id};

    resp = req.post('https://jira.example.com/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.getStatus() != 201) {
        throw 'Response code: ' + req.getStatus();
    }

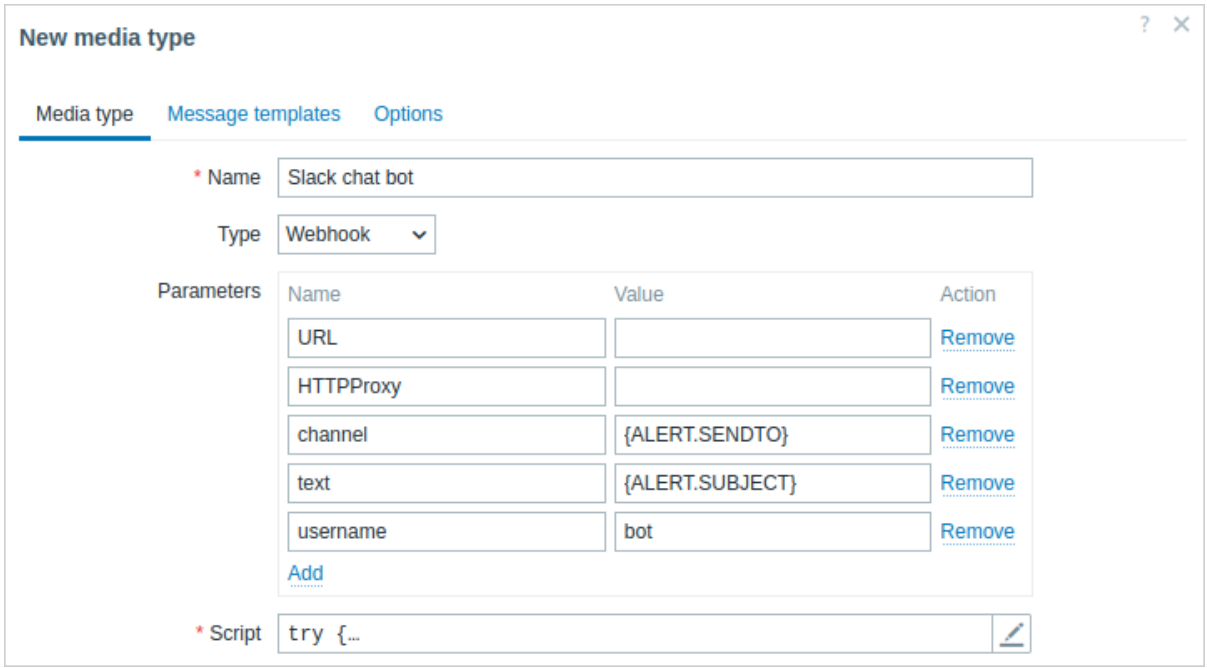
    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;

    return JSON.stringify(result);
}
catch (error) {
    Zabbix.log(4, '[ Jira webhook ] Issue creation failed json : ' + JSON.stringify({"fields": fields}));
    Zabbix.log(3, '[ Jira webhook ] issue creation failed : ' + error);

    throw 'Failed with error: ' + error;
}
```

Slack webhook（自定义）

此 webhook 会将来自 Zabbix 的通知转发到 Slack 频道。




```
try {
    var params = JSON.parse(value),
        req = new HttpRequest(),
        response;

    if (params.HTTPProxy) {
        req.setProxy(params.HTTPProxy);
    }

    req.addHeader('Content-Type: application/x-www-form-urlencoded');

    Zabbix.log(4, '[ Slack webhook ] Webhook request with value=' + value);
```

```
    response = req.post(params.URL, 'payload=' + encodeURIComponent(value));
    Zabbix.log(4, '[ Slack webhook ] Responded with code: ' + req.getStatus() + '. Response: ' + response)

    try {
        response = JSON.parse(response);
    }
    catch (error) {
        if (req.getStatus() < 200 || req.getStatus() >= 300) {
            throw 'Request failed with status code ' + req.getStatus();
        }
        else {
            throw 'Request success, but response parsing failed.';
        }
    }

    if (req.getStatus() !== 200 || !response.ok || response.ok === 'false') {
        throw response.error;
    }

    return 'OK';
}
catch (error) {
    Zabbix.log(3, '[ Slack webhook ] Sending failed. Error: ' + error);

    throw 'Failed with error: ' + error;
}
```

操作

概述

如果您希望某些操作作为事件的结果而发生（例如，发送通知），则需要配置动作。

可以针对所有受支持类型的事件定义动作：

- 触发器动作 - 用于触发器状态在 **OK** 和 **Problem** 之间变化时产生的事件
- 服务动作 - 用于服务状态在 **OK** 和 **Problem** 之间变化时产生的事件
- 发现动作 - 用于发生网络发现时产生的事件
- 自动注册动作 - 用于新的活动 agent 自动注册时产生的事件（或已注册主机的主机元数据发生变化时）
- 内部动作 - 用于监控项变为不支持或触发器进入未知状态时产生的事件

服务动作的主要区别在于：

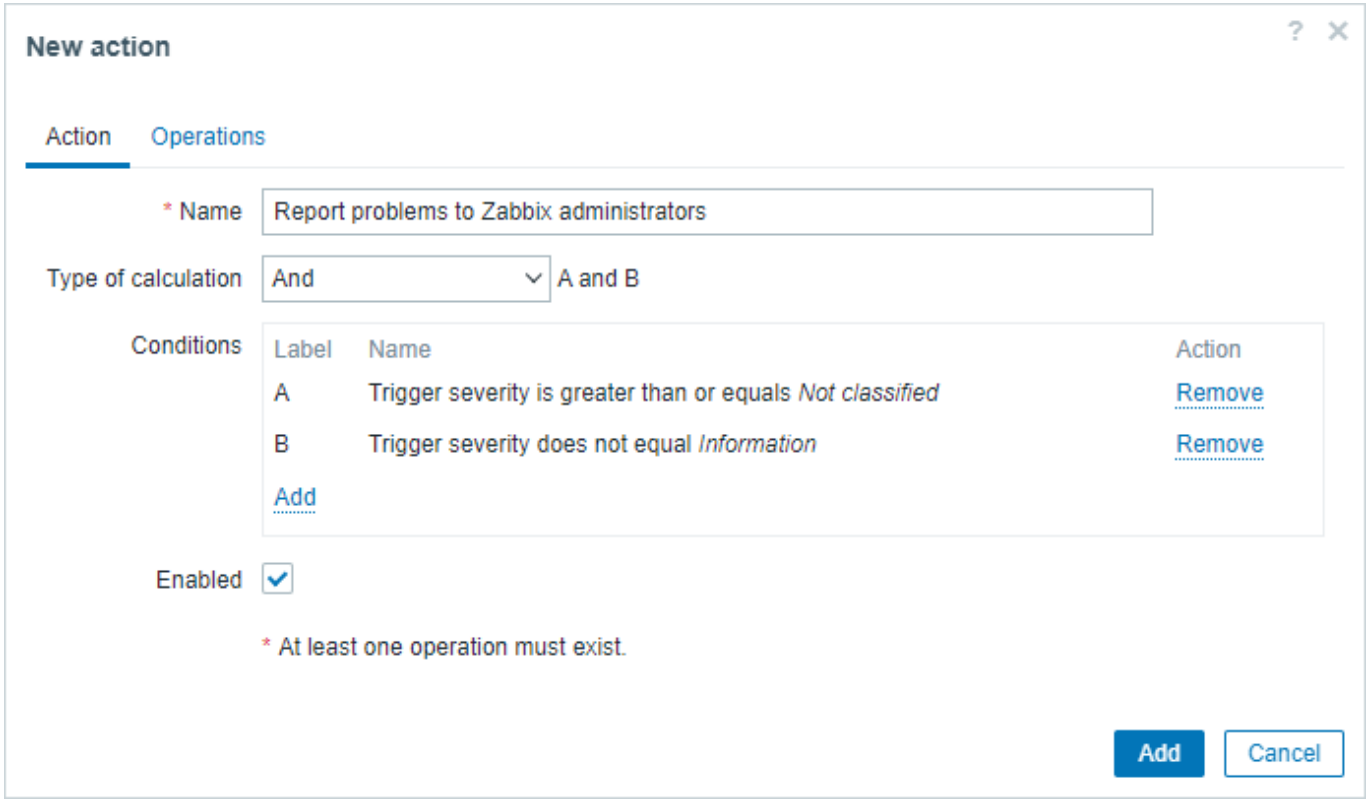
- 用户对服务动作的访问取决于用户角色中授予的服务访问权限
- 服务动作支持不同的一组条件

配置一个 action

要配置一个 action，请执行以下操作：

1. 转到 *Alerts* > *Actions*，并从子菜单中选择所需的 action 类型（之后可以使用标题下拉菜单切换到其他类型）。
2. 单击 *Create action*。
3. 为 action 命名。
4. 选择要执行操作所依据的conditions。
5. 选择要执行的operations。

常规 action 属性：



所有必填输入字段都用红色星号标记。

Parameter	Description
Name	唯一的 action 名称。
Type of calculation	为 action conditions 选择评估option（适用于多个条件）： **And** - 必须满足所有条件。 **Or** - 满足其中一个条件即可。 **And/Or** - 两者的组合：不同条件类型之间使用 AND，相同条件类型之间使用 OR。 **Custom expression** - 用户自定义的计算公式，用于评估 action conditions。
Conditions	action conditions 列表。 单击 *Add* 可添加新的condition。 如果未配置任何 conditions，则该 action 将针对与正在配置的action type 相对应的每个事件运行。
Enabled	勾选复选框以启用该 action。否则，它将被禁用。

条件

概述

可以定义仅当事件匹配一组已定义条件时才执行某个操作。条件是在配置action时设置的。

条件匹配区分大小写。

触发动作

以下条件可用于基于触发器的动作：

条件类型	支持的运算符	说明
Host group	等于 不等于	指定要排除的主机组或主机组。 等于 - 事件属于此主机组。 不等于 - 事件不属于此主机组。 指定父主机组会隐式选择所有嵌套的主机组。要仅指定父组，必须使用 不等于运算符另外设置所有嵌套组。
Template	等于 不等于	指定模板或要排除的模板。 等于 - 事件属于从该模板继承的触发器。 不等于 - 事件不属于从该模板继承的触发器。

条件类型	支持的运算符	说明
Host	等于 不等于	指定主机或要排除的主机。 等于 - 事件属于此主机。 不等于 - 事件不属于此主机。
Tag name	等于 不等于 包含 不包含	指定事件标签或要排除的事件标签。 等于 - 事件有此标签 * * 不等于 **-** 事件没有这个标签 ** ** 包含 **-** 事件有一个包含这个字符串的标签 ** ** 不包含 ** - 事件没有包含这个字符串的标签
Tag value	等于 不等于 包含 不包含	指定标签和值组合或要排除的事件标签和值组合。 等于 - 事件有这个标签和值 不等于 - 事件没有这个标签和值 包含 - 事件有一个包含这些字符串的标签和值 不包含 - 事件没有包含这些字符串的标签和值
Trigger	等于 不等于	指定触发器或要排除的触发器。 等于 - 事件由此触发器生成。 不等于 - 事件通过任何其他触发器生成，除了这个。
Trigger name	包含 不包含	指定触发器名称中的字符串或要排除的字符串。 包含 - 事件由触发器生成，名称中包含此字符串。 不包含 - 在触发器名称中找不到此字符串。 注意：输入的值将与展开所有宏的触发器名称进行比较。
Trigger severity	等于 不等于 大于或等于 小于或等于	指定触发器严重性。 等于 - 等于触发器严重性 不等于 - 不等于触发严重性 大于或等于 - 大于或等于触发严重性 小于或等于 - 小于或等于触发严重程度
Time period	在 不在	指定时间段或要排除的时间段。 在 - 事件时间在时间段内。 不在 - 事件时间不在时间段内。 请参阅时间段规范页面了解格式说明。 自 Zabbix 3.4.0 起开始支持用户宏 .
Problem is suppressed	否 是	如果由于主机维护问题被抑制（未显示），请指定。 否 - 问题未被抑制。 是 - 问题被抑制了。

服务动作

以下条件可用于服务动作：

Condition type	Supported operators	Description
Service	equals does not equal	指定一个服务或要排除的服务。 **equals** - 事件属于此服务。 **does not equal** - 事件不属于此服务。 指定父服务会隐式选择所有子服务。若只指定父服务，则还必须使用 **does not equal** 运算符额外设置所有嵌套服务。
Service name	contains does not contain	指定服务名称中的字符串或要排除的字符串。 **contains** - 事件由某个服务生成，该服务名称中包含此字符串。 **does not contain** - 在服务名称中找不到此字符串。
Service tag name	equals does not equal contains does not contain	指定事件标签或要排除的事件标签。服务事件标签可在服务配置部分 *Tags* 中定义。 **equals** - 事件具有此标签。 **does not equal** - 事件不具有此标签。 **contains** - 事件具有包含此字符串的标签。 **does not contain** - 事件不具有包含此字符串的标签。
Service tag value	equals does not equal contains does not contain	指定事件标签和值的组合，或要排除的标签和值组合。服务事件标签可在服务配置部分 *Tags* 中定义。 **equals** - 事件具有此标签和值。 **does not equal** - 事件不具有此标签和值。 **contains** - 事件具有包含这些字符串的标签和值。 **does not contain** - 事件不具有包含这些字符串的标签和值。

Attention:
请确保在 *Alerts* > *Media types* 菜单中为 Service actions 定义message templates。否则，将不会发送通知。

发现动作

以下条件可用于基于发现的事件：

条件类型	支持的运算符	描述
主机 *IP*	等于 不等于	指定已发现主机的 IP 地址范围，或要排除的范围。 等于 - 主机 IP 在该范围内。 不等于 - 主机 IP 不在该范围内。 支持以下格式： 单个 IP：192.168.1.33 IP 地址范围：192.168.1-10.1-254 IP 掩码：192.168.4.0/24 列表：192.168.1.1-254, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 支持在列表格式中使用空格。
服务类型	等于 不等于	指定已发现服务的服务类型，或要排除的服务类型。 等于 - 与已发现的服务匹配。 不等于 - 与已发现的服务不匹配。 可用的服务类型：SSH、LDAP、SMTP、FTP、HTTP、HTTPS、POP、NNTP、IMAP、TCP、Zabbix agent、SNMPv1 agent、SNMPv2 agent、SNMPv3 agent、ICMP ping、telnet。
服务端口	等于 不等于	指定已发现服务的 TCP 端口范围，或要排除的范围。 等于 - 服务端口在该范围内。 不等于 - 服务端口不在该范围内。
发现规则	等于 不等于	指定发现规则，或要排除的发现规则。 等于 - 使用此发现规则。 不等于 - 使用除该规则之外的任何其他发现规则。
发现检查	等于 不等于	指定发现检查，或要排除的发现检查。 等于 - 使用此发现检查。 不等于 - 使用除该检查之外的任何其他发现检查。
发现对象	等于	指定已发现的对象。 等于 - 等于已发现的对象（设备或服务）。
发现状态	等于	**Up** - 匹配 “主机 Up” 和 “服务 Up” 事件。 **Down** - 匹配 “主机 Down” 和 “服务 Down” 事件。 **Discovered** - 匹配 “主机已发现” 和 “服务已发现” 事件。 **Lost** - 匹配 “主机丢失” 和 “服务丢失” 事件。
运行时间/停机时间	大于或等于 小于或等于	对于 “主机 Up” 和 “服务 Up” 事件，表示运行时间；对于 “主机 Down” 和 “服务 Down” 事件，表示停机时间。 大于或等于 - 大于或等于指定值。参数以秒为单位。 小于或等于 - 小于或等于指定值。参数以秒为单位。
接收值	等于 不等于 大于或等于 小于或等于 包含 不包含	指定从发现规则中的 agent（Zabbix、SNMP）检查接收的值。进行字符串比较。如果为某个规则配置了多个 Zabbix agent 或 SNMP 检查，则会检查每个检查的接收值（每个检查都会生成一个新事件，并根据所有条件进行匹配）。 等于 - 等于指定值。 不等于 - 不等于指定值。 大于或等于 - 大于或等于指定值。 小于或等于 - 小于或等于指定值。 包含 - 包含该子字符串。参数以字符串形式提供。 不包含 - 不包含该子字符串。参数以字符串形式提供。
Proxy	等于 不等于	指定 proxy，或要排除的 proxy。 等于 - 使用此 proxy。 不等于 - 使用除该 proxy 之外的任何其他 proxy。
Proxy 组	等于 不等于	指定 proxy 组，或要排除的 proxy 组。 等于 - 使用此 proxy 组。 不等于 - 使用除该 proxy 组之外的任何其他 proxy 组。

Note:
发现规则中的服务检查会生成发现事件，但这些检查不会同时执行。因此，如果在动作中为 `Service type`、`Service port` 或 `Received value` 条件配置了多个值，则这些值会逐个与发现事件进行比较，而不会同时与多个事件进行比较。因此，对于相同检查类型配置了多个值的动作可能无法正确执行。

自动注册动作

以下条件可用于基于 active agent 自动注册的动作：

条件类型	支持的运算符	描述
主机元数据	包含 不包含 匹配 不匹配	指定主机元数据或要排除的主机元数据。 包含 - 主机元数据包含该字符串。 不包含 - 主机元数据不包含该字符串。 可在agent 配置文件 中指定主机元数据。 匹配 - 主机元数据匹配正则表达式。 不匹配 - 主机元数据不匹配正则表达式。
主机名称	包含 不包含 匹配 不匹配	指定主机名称或要排除的主机名称。 包含 - 主机名称包含该字符串。 不包含 - 主机名称不包含该字符串。 匹配 - 主机名称匹配正则表达式。 不匹配 - 主机名称不匹配正则表达式。
proxy	等于 不等于	指定 proxy 或要排除的 proxy。 等于 - 使用此 proxy。 不等于 - 使用除该 proxy 之外的任何其他 proxy（或服务器）。
proxy 组	等于 不等于	指定 proxy 组或要排除的 proxy 组。 等于 - 使用此 proxy 组。 不等于 - 使用除该 proxy 组之外的任何其他 proxy 组。

内部事件动作

可以为基于内部事件的动作设置以下条件:

条件类型	支持的操作符	说明
事件类型	等于	处于 **“不支持”** 状态的监控项 - 匹配监控项从’ 正常’ 变为’ 不支持’ 状态的事件 处于 **“不支持”** 状态的底层自动发现规则 - 匹配底层自动发现规则从’ 正常’ 变为’ 不支持’ 状态的事件 处于 **“未知”** 状态的触发器 - 匹配触发器从’ 正常’ 变为’ 未知’ 状态的事件
主机组	等于 不等于	主机组或要排除的主机组 等于 - 属于该主机组的事件 不等于 - 不属于该主机组的事件
标签名称	等于 不等于 包含 不包含	标签名称或者要排除的标签名称 等于 - 具备该标签的事件 不等于 - 不具备该标签的事件 包含 - 标签名称中包含该字符串的事件 不包含 - 标签名称中不包含该字符串的事件

条件类型	支持的操作符	说明
标签值	等于 不等于 包含 不包含	事件标签和值组合或要排除的标签和值组合 等于 - 具备该标签并且值等于给定值的事件 不等于 - 具备该标签但值不等于给定值的事件 包含 - 具备该标签并且值包含指定字符串的事件 不包含 - 具备该标签但值不包含指定字符串的事件
模板	等于 不等于	模板或要排除的模板 等于 - 属于从此模板继承的监控项/触发器/底层自动发现规则的事件 不等于 - 不属于从此模板继承的监控项/触发器/底层自动发现规则的事件
主机	等于 不等于	主机或要排除的主机 等于 - 属于此主机的事件 不等于 - 不属于此主机的事件

计算类型

可用的条件计算选项如下：

- **And** - 必须满足所有条件

请注意，当多个触发器被选为 `Trigger=` 条件时，不允许在它们之间使用 “And” 计算。操作只能基于一个触发器的事件来执行。

- **Or** - 只需满足一个条件即可
- **And/Or** - 两者的组合：不同条件类型之间使用 AND，相同条件类型之间使用 OR，例如：

主机组等于 Oracle servers
 主机组等于 MySQL servers
 事件名称包含’Database is down’
 事件名称包含’Database is unavailable’

其计算结果为

(主机组等于 Oracle servers **or** 主机组等于 MySQL servers**) and (**事件名称包含’Database is down’ **or** 事件名称包含’Database is unavailable’**)**

- **Custom expression** - 用户定义的计算公式，用于评估操作条件。它必须包含所有条件（以大写字母 A、B、C、... 表示），并且可以包含空格、制表符、括号 ( )、**and**（区分大小写）、**or**（区分大小写）、**not**（区分大小写）。

虽然前面的 `And/Or` 示例可表示为 (A or B) and (C or D)，但在自定义表达式中，你也可以使用多种其他计算方式：

(A and B) and (C or D)
 (A and B) or (C and D)
 ((A or B) and C) or D
 (not (A or B) and C) or not D
 等等。

删除在动作中使用的对象

当尝试删除在动作条件/操作中使用的对象时，用户将收到相应的错误消息。

如果动作条件或操作：

- 包含已删除的对象；
- 包含不再由低级别发现发现并已被移除的对象（在 *Delete lost resources* 周期之后）。

动作不会被禁用。

如果动作条件/操作中存在缺失的对象，则不会将其移除。相反，已删除的对象会显示为 *Deleted host group*、*Deleted host*、*Deleted trigger* 等。在动作列表的 “Info” 列中，会显示一个警告图标，并附带消息 “This action has conditions or operations referencing deleted object(s).”。

操作

概述

您可以为所有事件定义以下操作：

- 发送消息

- 执行远程命令

Attention:
如果为操作接收者定义的用户被明确 “拒绝” 访问该主机，或者该用户根本没有对该主机定义任何权限，Zabbix 服务器将不会创建告警。

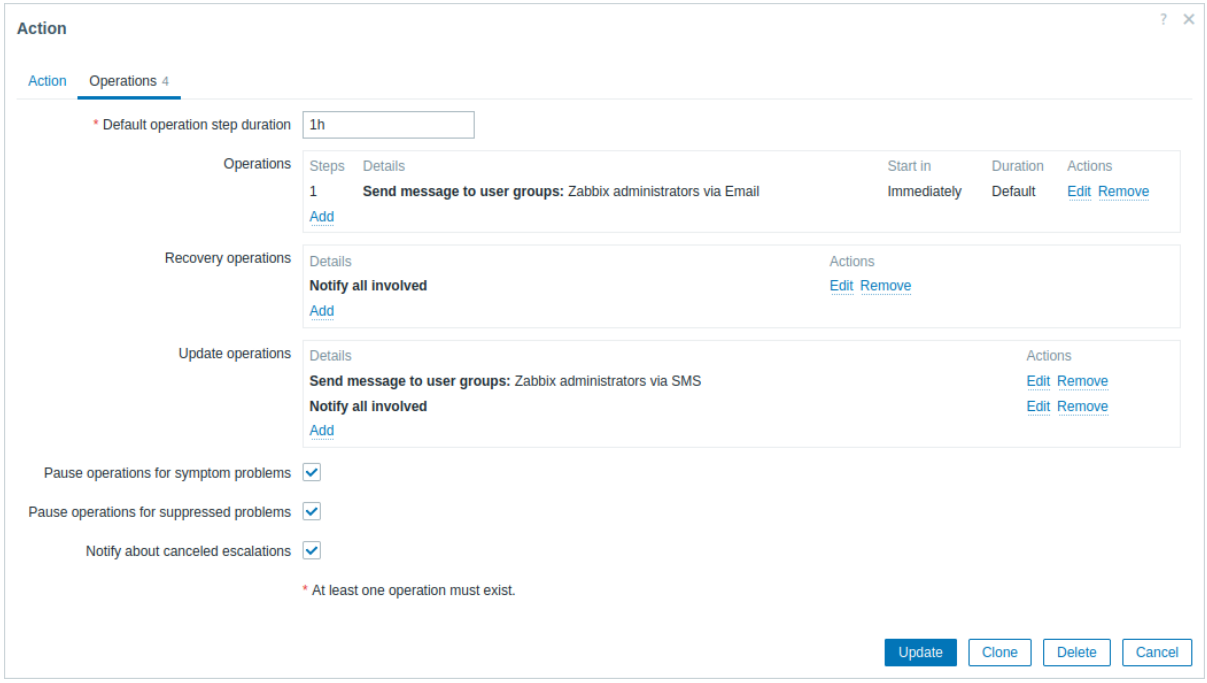
对于发现和自动注册事件，还可使用其他操作：

- 添加主机
- 删除主机
- 启用主机
- 禁用主机
- 添加到主机组
- 从主机组中移除
- 添加主机标签
- 移除主机标签
- 链接模板
- 取消链接模板
- 设置主机资产记录模式

配置操作

要配置操作：

1. 转到操作 配置中的 操作选项卡。
2. 配置操作详细信息。



操作常规属性：

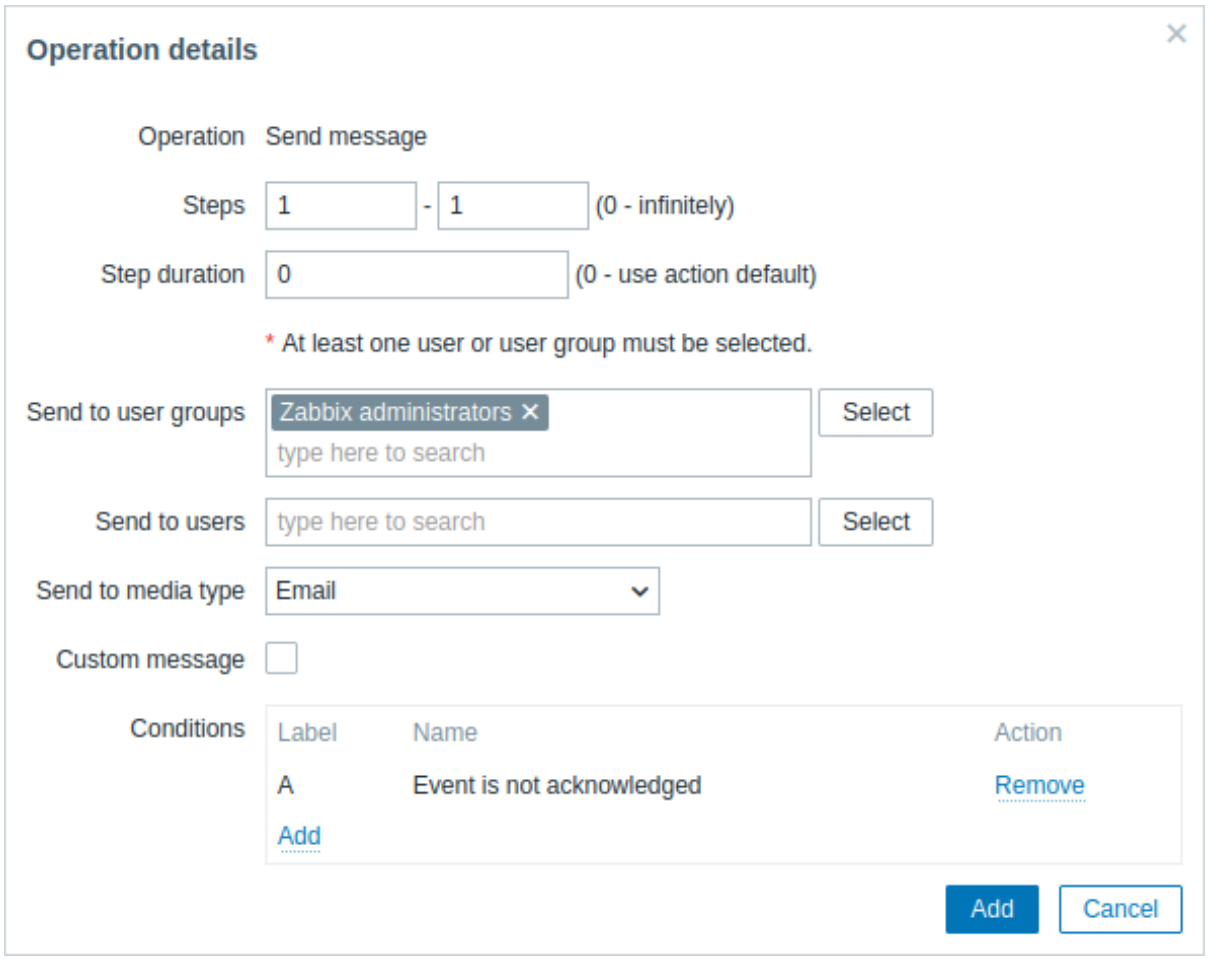
参数	描述
默认操作步骤持续时间	默认情况下一个操作步骤的持续时间（60 秒至 1 周）。 例如，步骤持续时间为 1 小时表示，如果执行了某个操作，则需要经过 1 小时才会执行下一步骤。 支持时间后缀，例如 60s、1m、2h、1d。 支持用户宏。
操作	显示操作（如果有），包括以下详细信息： 步骤 - 分配给操作的升级步骤。 详细信息 - 操作类型及其接收者/目标。 操作列表还会显示所使用的媒介类型（电子邮件、SMS 或脚本），以及通知接收者的名字和姓氏（显示在用户名后的括号中）。 开始于 - 事件发生后多长时间执行操作。 持续时间（秒） - 显示步骤持续时间。如果步骤使用默认持续时间，则显示 默认；如果使用自定义持续时间，则显示相应时间。 操作 - 显示用于编辑和删除操作的链接。

参数	描述
恢复操作	显示操作（如果有），包括以下详细信息： 详细信息 - 操作类型及其接收者/目标。 操作列表还会显示所使用的媒介类型（电子邮件、SMS 或脚本），以及通知接收者的名字和姓氏（显示在用户名后的括号中）。 操作 - 显示用于编辑和删除操作的链接。
更新操作	显示操作（如果有），包括以下详细信息： 详细信息 - 操作类型及其接收者/目标。 操作列表还会显示所使用的媒介类型（电子邮件、SMS 或脚本），以及通知接收者的名字和姓氏（显示在用户名后的括号中）。 操作 - 显示用于编辑和删除操作的链接。
暂停症状问题的操作	选中此复选框可暂停症状问题的操作（首次操作之后）。 请注意，此设置仅影响问题升级；恢复和更新操作不受影响。 此选项仅适用于 触发器操作。
暂停被抑制问题的操作	选中此复选框可在维护期间延迟操作的开始时间。维护结束后开始执行操作时，将执行所有操作，包括维护期间事件对应的操作。 请注意，此设置仅影响问题升级；恢复和更新操作不受影响。 如果取消选中此复选框，即使在维护期间也会立即执行操作，不会延迟。 此选项不适用于 服务操作。
通知已取消的升级	取消选中此复选框可禁用有关已取消升级的通知（当主机、监控项、触发器或操作被禁用时）。

所有必填输入字段均以红色星号标记。

要配置新操作的详细信息，请单击 Add（位于 操作块中）。要编辑现有操作，请单击操作旁边的 Edit。此时将打开一个弹出窗口，您可以在其中编辑操作步骤的详细信息。

操作详细信息



参数		描述
操作		选择操作： 发送消息 - 向用户发送消息。 **<** 远程命令名称 **>** - 执行远程命令。如果之前已在全局脚本中定义，并选择操作执行作为其作用域，则可执行这些命令。 基于发现和自动注册事件的操作也可用（见上文）。
步骤		在升级计划中，选择要将操作分配到的步骤： 从 - 从此步骤开始执行。 到 - 执行至此步骤（0= 无限，执行不受限制）。
步骤持续时间		这些步骤的自定义持续时间（0= 使用默认步骤持续时间）。 支持时间后缀，例如 60s、1m、2h、1d。 支持用户宏。 可以将多个操作分配到同一个步骤。如果这些操作定义了不同的步骤持续时间，则采用最短的持续时间并将其应用于该步骤。
操作类型：发送消息		
	发送到用户组	选择要向其发送消息的用户组。 用户组必须至少拥有对该主机的 “读取”权限，才能接收通知。
	发送到用户	选择要向其发送消息的用户。 用户必须至少拥有对该主机的 “读取”权限，才能接收通知。
	发送到媒介类型	向所有可用的（已配置且已启用）媒介类型发送消息，或仅向特定媒介类型发送消息。
	自定义消息	选中后，可以配置自定义消息。 通过webhook发送内部事件通知时，必须使用自定义消息。
	主题	自定义消息的主题。主题可以包含宏。长度限制为 255 个字符。
	消息	自定义消息。消息可以包含宏。长度限制取决于数据库类型（有关更多信息，请参阅发送消息）。
操作类型：远程命令		

参数	描述
目标列表	选择要在其上执行命令的目标： 当前主机 - 在导致问题事件的触发器所属的主机上执行命令。如果触发器中包含多个主机，此选项将不起作用。 主机 - 选择要在其上执行命令的主机。 主机组 - 选择要在其上执行命令的主机组。指定父主机组会隐式选择所有嵌套的主机组。因此，也会在嵌套组中的主机上执行远程命令。 即使某个主机匹配多次（例如来自多个主机组，或分别匹配主机和主机组），也只会在该主机上执行一次命令。 如果在 Zabbix 服务器上执行自定义脚本，则目标列表没有意义。在这种情况下选择更多目标，只会导致脚本在服务器上执行更多次。 请注意，对于全局脚本，目标选择还取决于全局脚本配置中的主机组设置。 目标列表选项不适用于服务操作，因为在这种情况下，远程命令始终在 Zabbix 服务器上执行。
条件	执行操作的条件： 事件未确认 - 仅当事件未确认时执行。 事件已确认 - 仅当事件已确认时执行。 条件选项仅适用于触发器操作。

完成后，单击添加，将操作添加到操作列表中。

发送消息

概述

发送消息是将问题通知给相关人员的最佳方式之一。这也是它成为 Zabbix 提供的主要操作之一的原因。

配置

要能够从 Zabbix 发送和接收通知，您必须：

- 定义媒体，用于发送消息。

如果操作发生在用户配置中为所选媒体定义的活动时间 之外，则不会发送消息。

如果您希望接收发现、agent 主动自动注册或内部事件等非触发器事件的通知，则必须在用户媒体配置中勾选默认触发器严重性（“未分类”）。

- 配置操作步骤，将消息发送到已定义的某个媒体。

Attention:
Zabbix 仅向对生成事件的主机至少具有 “读取” 权限的用户发送通知。触发器表达式中的至少一个主机必须可访问。

您可以使用升级配置发送消息的自定义场景。

要成功接收和阅读来自 Zabbix 的电子邮件，电子邮件服务器/客户端必须支持标准的 “SMTP/MIME 电子邮件” 格式，因为 Zabbix 发送 UTF-8 数据（如果主题仅包含 ASCII 字符，则不会进行 UTF-8 编码）。消息的主题和正文会进行 base64 编码，以符合 “SMTP/MIME 电子邮件” 格式标准。

所有宏展开后的消息限制与远程命令的消息限制相同。

跟踪消息

您可以在 *Monitoring* > *Problems* 中查看已发送消息的状态。

在 *Actions* 列中，您可以看到已执行操作的汇总信息。其中绿色数字表示已发送的消息，红色数字表示发送失败的消息。*In progress* 表示某个操作已启动。*Failed* 表示没有任何操作成功执行。

如果您单击事件时间以查看事件详情，您将能够在 *Actions* 区块中查看由于该事件而发送（或未发送）的消息详情。

在 *Reports* > *Action log* 中，您将看到针对已配置操作的这些事件所执行的所有操作的详情。

远程命令

概述

通过远程命令，您可以定义在满足某些条件时，自动在被监控主机上执行某个预定义命令。

因此，远程命令是实现智能主动监控的强大机制。

在此功能最常见的用途中，您可以尝试：

- 如果某个应用程序（Web 服务器、中间件、CRM）无响应，则自动重启该应用程序
- 如果某台远程服务器不响应请求，则使用 IPMI 的 “reboot” 命令重启该服务器
- 如果磁盘空间不足，则自动释放磁盘空间（删除较旧的文件、清理 /tmp）
- 根据 CPU 负载，将虚拟机从一台物理机迁移到另一台物理机
- 当 CPU（磁盘、内存或其他资源）不足时，向云环境添加新节点

为远程命令配置操作与为发送消息配置操作类似；唯一的区别是，Zabbix 执行命令，而不是发送消息。

远程命令可以由 Zabbix 服务器、Zabbix proxy 或 Zabbix agent 执行。请注意以下事项：

- Zabbix agent 上的远程命令可以由 Zabbix 服务器直接执行，也可以通过 Zabbix proxy 执行。
- Zabbix 服务器执行的远程命令按照命令执行中所述的方式运行，包括检查退出代码。
- 即使目标主机处于维护模式，远程命令仍会执行。

默认情况下，Zabbix agent 和 Zabbix proxy 上的远程命令均处于禁用状态。可以按如下方式启用：

- 对于由 Zabbix agent 执行的远程命令，请在agent 配置中为每个允许的命令添加 `AllowKey=system.run[<command>,*]` 或 `AllowKeyRegexp=^system\.run\[<command>,.*\]$` 参数，其中 * 匹配任一执行模式（`wait` 或 `nowait`）。
- 对于由 Zabbix proxy 执行的远程命令，请在proxy 配置中将 `EnableRemoteCommands` 参数设置为 1。请注意，当远程命令由 proxy 监控的主机上的 Zabbix agent 执行时，无需在 Zabbix proxy 上启用远程命令。

Attention:
除非确有必要，否则不要使用 `AllowKey=system.run[*]` 或 `AllowKeyRegexp=^system\.run\[.*\]$`，因为这会允许在主机上执行任意命令。出于安全原因，建议仅允许执行精确的命令，而 `system.run[*]` 默认处于拒绝状态（`DenyKey=system.run[*]`）。

另请参阅限制 agent 检查。

远程命令限制

在解析所有宏之后，远程命令的限制取决于数据库类型和字符集（非 ASCII 字符存储时需要超过一个字节）：

Database	字符数限制	字节数限制
MySQL	65535	65535
PostgreSQL	65535	不受限制
SQLite（仅 Zabbix proxy）	65535	不受限制

远程命令执行输出（返回值）限制为 16MB（包括会被截断的尾随空白字符）。IPMI 远程命令的限制取决于已安装的 IPMI 库。请注意，数据库限制适用于所有远程命令。

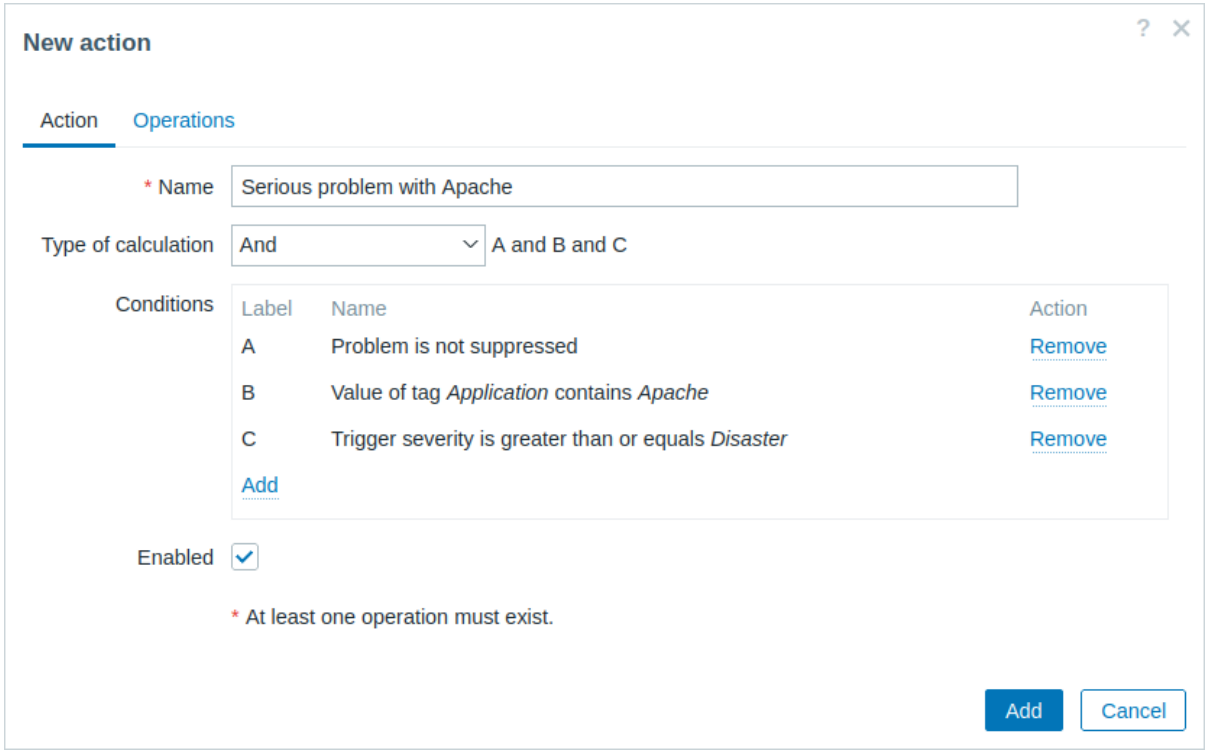
配置

Zabbix agent 执行的远程命令必须首先在agent 配置中使用 `AllowKey` 或 `AllowKeyRegexp` 参数进行允许。

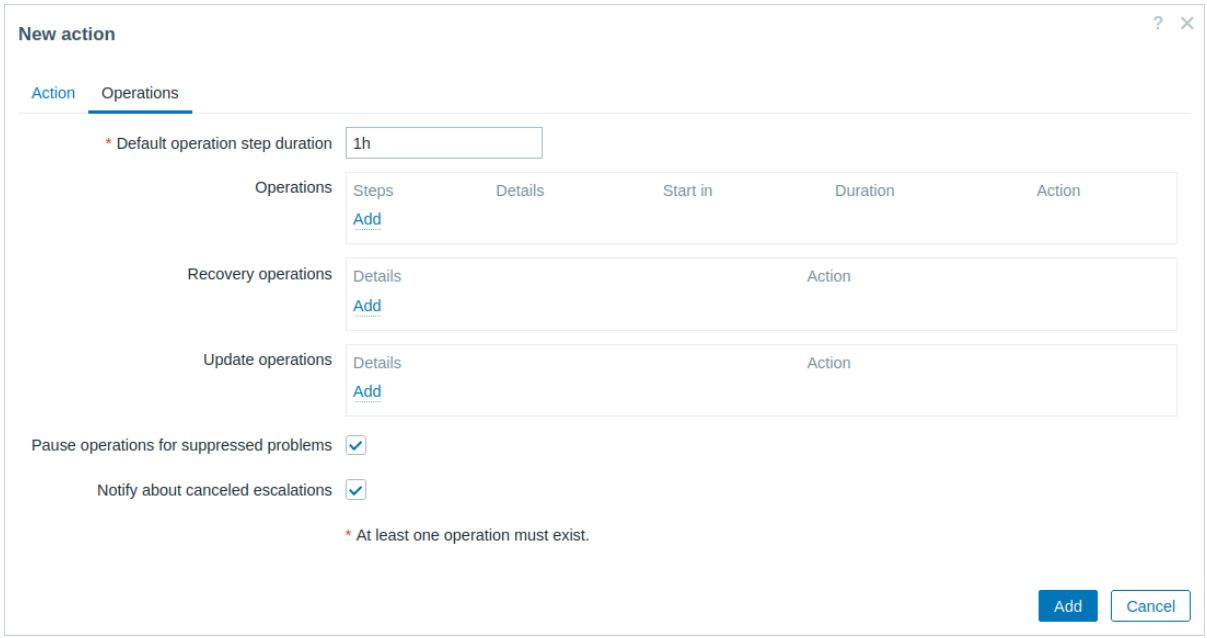
请确保在 agent 配置中为每个允许执行的命令添加 `AllowKey=system.run[<command>,*]` 或 `AllowKeyRegexp=^system\.run\[<comma` 参数，以允许其在 `nowait` 模式下执行。更改参数后，请重启 agent 守护进程。

然后，在告警 > 操作 > 触发器操作中配置新操作时：

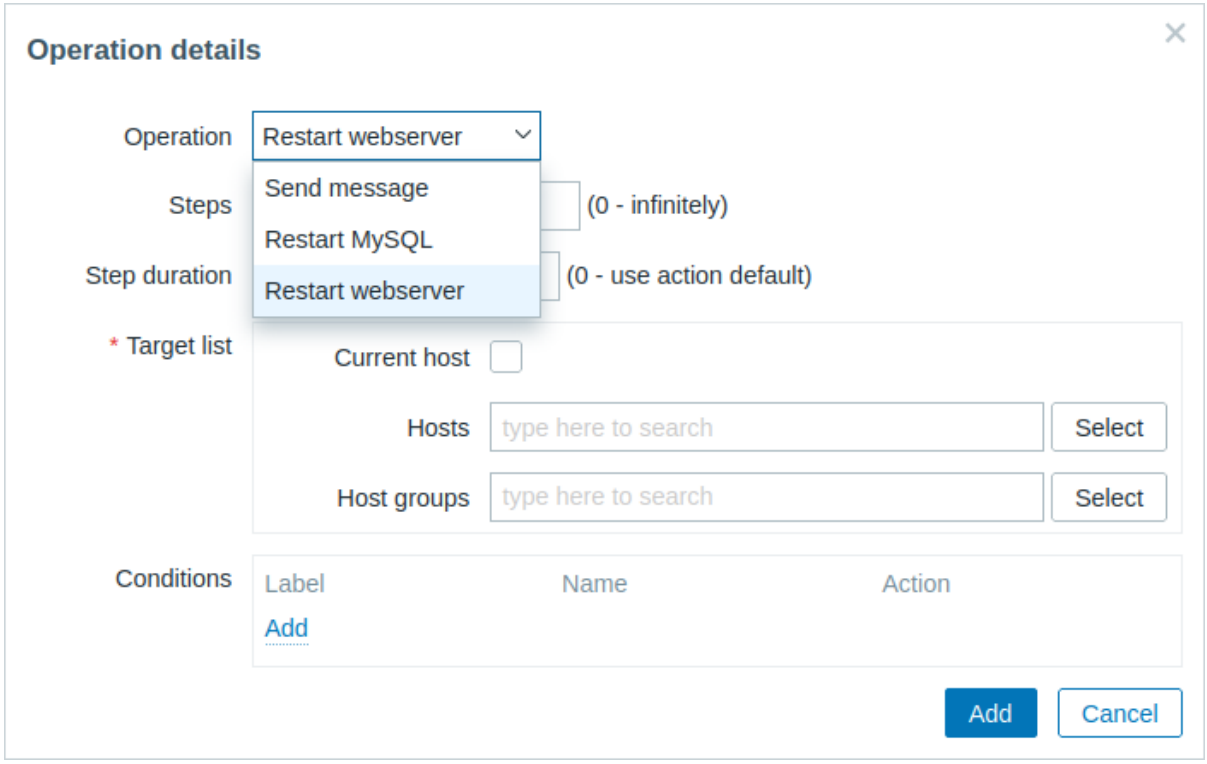
1. 定义适当的条件，例如，设置当任意一个 Apache 应用程序出现灾难级别问题时激活该操作。



2. 在操作选项卡中，在操作、恢复操作或更新操作部分单击添加。



3. 从操作下拉列表中选择一个预定义脚本，并为该脚本设置目标列表。



预定义脚本

可用于动作操作的脚本（webhook、脚本、SSH、Telnet、IPMI）在全局脚本 中定义，并将 *Action operation* 选为其作用域。

例如：

```
sudo /etc/init.d/apache restart
```

在这种情况下，Zabbix 将尝试重启 Apache 进程。使用此命令时，请确保该命令是在 Zabbix agent 上执行的（点击 *Execute on* 旁边的 *Zabbix agent* 按钮）。

Attention:
注意 sudo 的使用 - 默认情况下，Zabbix 用户没有重启系统服务的权限。有关如何配置 sudo 的提示，请参见下文。

Note:
从 Zabbix agent 7.0 开始，远程命令也可以在以主动模式运行的 agent 上执行。Zabbix agent - 无论是主动还是被动 - 都应运行在远程主机上，并在后台执行这些命令。

在 Zabbix agent 上，远程命令通过 `system.run[,nowait]` 键执行，不设置超时，也不会检查执行结果。
在 Zabbix 服务器和 Zabbix proxy 上，远程命令会按照 `zabbix_server.conf` 或 `zabbix_proxy.conf` 文件中`TrapperTimeout` 参数设置的超时时间执行，并会检查执行结果。
更多信息，请参见*Script timeout*。

访问权限

确保 `zabbix` 用户对已配置的命令具有执行权限。可以考虑使用 `sudo` 来授予对特权命令的访问权限。要配置访问权限，请以 root 身份执行：

```
visudo
```

`sudoers` 文件中可使用的示例行：

```
# 允许 'zabbix' 用户在无需密码的情况下运行所有命令。
zabbix ALL=NOPASSWD: ALL

# 允许 'zabbix' 用户在无需密码的情况下重启 apache。
zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```

Note:
在某些系统上，`sudoers` 文件会阻止非本地用户执行命令。要更改这一点，请在 `/etc/sudoers` 中注释掉 `requiretty` 选项。

使用多个接口的远程命令

如果目标系统具有所选类型的多个接口（Zabbix agent 或 IPMI），远程命令将在默认接口上执行。

可以通过 SSH 和 Telnet 使用不同于 Zabbix agent 接口的其他接口来执行远程命令。可用接口将按以下顺序选择：

- Zabbix agent 默认接口
- SNMP 默认接口
- JMX 默认接口
- IPMI 默认接口

IPMI 远程命令

对于 IPMI 远程命令应使用如下语法：

```
<command> [<value>]
```

其中

- <command>——表示一个没有空格的 IPMI 命令。
- <value>——其值为'on'、'off' 或任何无符号整数，<value> 是可选参数。

示例

可在操作中用作远程命令的全局脚本示例。

示例 **1**

在特定条件下重启 Windows。

为了在 Zabbix 检测到问题时自动重启 Windows，请定义以下脚本：

Script parameter	Value
Scope	'Action operation'
Type	'Script'
Command	`c:\windows\system32\shutdown.exe -r -f`

示例 2

用 IPMI 控制重启主机。

脚本参数	值
范围	'Action operation'
类型	'IPMI'
命令	reset

示例 3

使用 IPMI 控制关闭主机电源。

脚本参数	值
范围	'Action operation'
类型	'IPMI'
命令	power off

其他操作

概述

在本节中，您可以找到有关发现/自动注册事件的附加操作的一些详细信息。

添加主机

主机会在发现过程中被添加，也就是说，主机一旦被发现，就会立即添加，而不是等到整个发现过程结束后再添加。

Note:
由于网络发现可能会因为大量不可用的主机/服务而耗费一些时间，因此建议保持耐心，并使用合理的 IP 范围。

添加主机时，其名称由标准的 **gethostbyname** 函数决定。如果主机可以被解析，则使用解析后的名称；如果不能，则使用 IP 地址。此外，如果主机名必须使用 IPv6 地址，则所有":"（冒号）都会被替换为"_"（下划线），因为主机名中不允许使用冒号。

Attention:
如果通过 proxy 执行发现，目前主机名查找仍然是在 Zabbix 服务器上进行。

Attention:
如果 Zabbix 配置中已经存在与新发现主机同名的主机，Zabbix 会在主机名后附加 **_N**，其中 **N** 为递增数字，从 2 开始。

在消息中使用宏

概述

在消息主题和消息文本中，您可以使用宏来更高效地报告问题。

除了若干内置宏之外，还支持用户宏 和表达式宏。Zabbix 支持的完整宏列表 可供查阅。

示例

此处的示例说明了如何在消息中使用宏。

示例 1

消息主题：

```
Problem: {TRIGGER.NAME}
```

当您收到该消息时，消息主题将被替换为类似如下内容：

```
Problem: Zabbix 服务器上的处理器负载过高
```

示例 2

消息：

```
处理器负载为：{?last(/zabbix.zabbix.com/system.cpu.load[,avg1])}
```

当您收到该消息时，消息将被替换为类似如下内容：

```
处理器负载为：1.45
```

示例 3

消息：

```
最新值：{?last(/{HOST.HOST}/{ITEM.KEY})}
15 分钟内的最大值：{?max(/{HOST.HOST}/{ITEM.KEY},15m)}
15 分钟内的最小值：{?min(/{HOST.HOST}/{ITEM.KEY},15m)}
```

当您收到该消息时，消息内容将被替换为类似如下内容：

```
最新值：1.45
15 分钟内的最大值：2.33
15 分钟内的最小值：1.01
```

示例 4

消息：

```
http://<server_ip_or_name>/zabbix/tr_events.php?triggerid={TRIGGER.ID}&eventid={EVENT.ID}
```

当您收到该消息时，其中将包含一个指向事件详情页面的链接，该页面提供有关该事件、其触发器以及由同一触发器生成的最新事件列表的信息。

示例 5

在触发器表达式中通知来自多个主机的值。

消息：

```
Problem name: {TRIGGER.NAME}
Trigger expression: {TRIGGER.EXPRESSION}
```

1. 主机 {HOST.NAME1} 上的监控项值：{ITEM.VALUE1} ({ITEM.NAME1})
2. 主机 {HOST.NAME2} 上的监控项值：{ITEM.VALUE2} ({ITEM.NAME2})

当您收到该消息时，消息将被替换为类似如下内容：

```
Problem name: 本地主机上的处理器负载过高
Trigger expression: last(/Myhost/system.cpu.load[percpu,avg1])>5 or last(/Myotherhost/system.cpu.load[perc

1. 主机 Myhost 上的监控项值：0.83（处理器负载（每核 1 分钟平均值））
2. 主机 Myotherhost 上的监控项值：5.125（处理器负载（每核 1 分钟平均值））
```

示例 6

在恢复 消息中同时接收问题事件和恢复事件的详细信息：

消息：

```
问题：

事件 ID：{EVENT.ID}
事件值：{EVENT.VALUE}
事件状态：{EVENT.STATUS}
事件时间：{EVENT.TIME}
事件日期：{EVENT.DATE}
事件持续时间：{EVENT.AGE}
事件确认：{EVENT.ACK.STATUS}
事件更新历史：{EVENT.UPDATE.HISTORY}

恢复：

事件 ID：{EVENT.RECOVERY.ID}
事件值：{EVENT.RECOVERY.VALUE}
事件状态：{EVENT.RECOVERY.STATUS}
事件时间：{EVENT.RECOVERY.TIME}
事件日期：{EVENT.RECOVERY.DATE}
运行数据：{EVENT.OPDATA}
```

当您收到该消息时，这些宏将被替换为如下内容：

```
问题：

事件 ID：21874
事件值：1
事件状态：PROBLEM
事件时间：13:04:30
事件日期：2018.01.02
事件持续时间：5m 0s
事件确认：是
事件更新历史：2018.01.02 13:05:51 "John Smith (Admin)"
操作：已确认。

恢复：

事件 ID：21896
事件值：0
事件状态：OK
事件时间：13:10:07
事件日期：2018.01.02
运行数据：当前值为 0.83
```

恢复操作

概述

恢复操作允许您在问题得到解决时收到通知。

恢复操作支持消息和远程命令。虽然可以添加多个操作，但不支持升级 - 所有操作都分配给一个步骤，因此将同时执行。

用例

恢复操作的一些用例如下：

1. 通知所有收到问题通知的用户恢复：

- 选择通知所有相关人员作为操作类型。

2. 恢复后有多个操作：发送通知和执行远程命令：

- 添加发送消息和执行命令的操作类型。

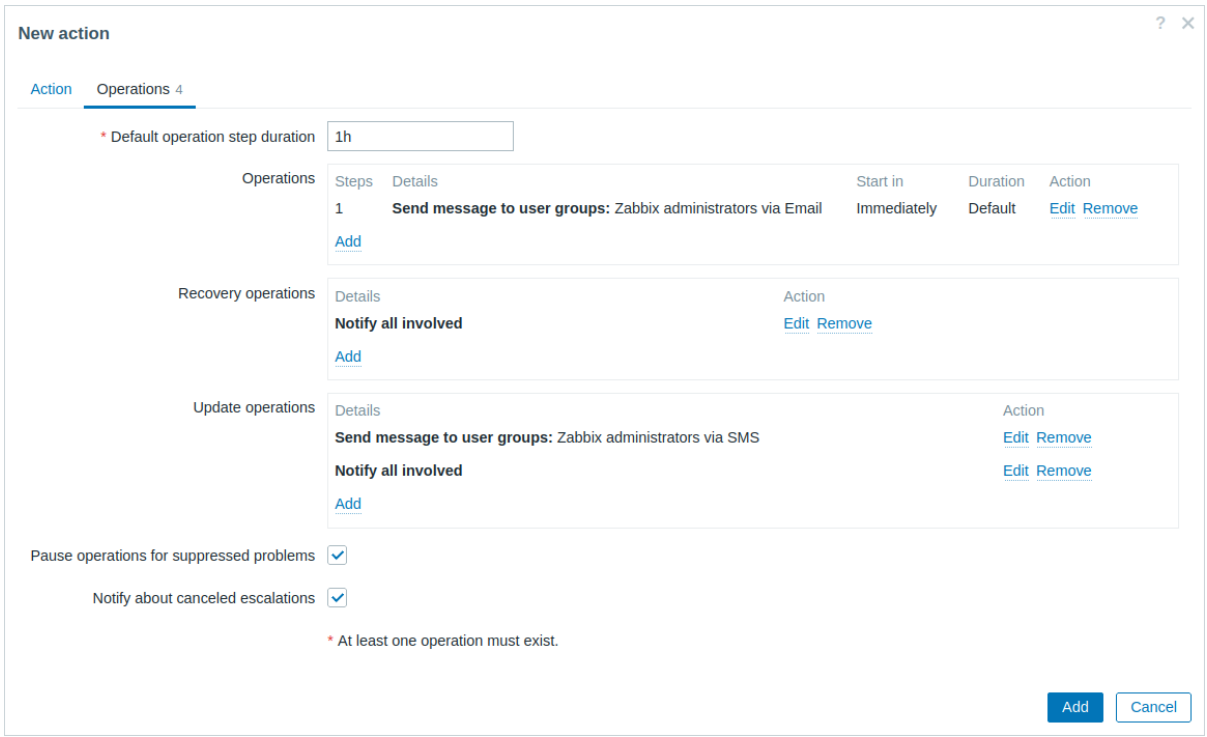
3. 在外部帮助台/工单系统中打开一个工单，并在问题解决后将其关闭：

- 创建一个与服务台系统通信的外部脚本。
- 创建一个具有执行此脚本并因此打开工单的操作的动作。
- 进行恢复操作，使用其他参数执行此脚本并关闭票证。
- 使用 {EVENT.ID} 宏来引用原始问题。

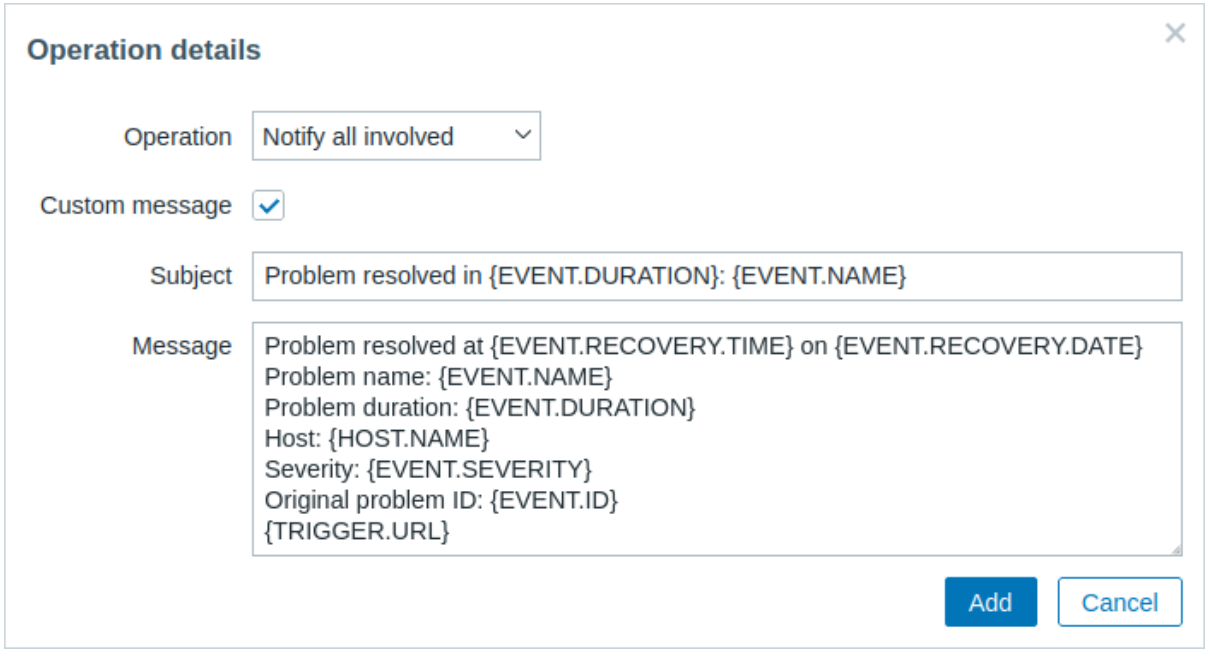
配置恢复操作

要配置恢复操作，请按以下步骤进行：

1. 在action 配置表单中，转到 操作选项卡。
2. 要配置新的恢复操作，请在 恢复操作部分中单击 添加。要编辑现有操作，请单击该操作旁边的 编辑。
3. 配置操作详细信息。



恢复操作详情



恢复事件可使用三种操作类型：

- 发送消息 - 向指定用户发送恢复消息；
- 通知所有相关人员 - 向在问题事件中已收到通知的所有用户发送恢复消息；
- **<** 远程命令名称 **>** - 执行远程命令。如果此前已在全局脚本中定义命令，并将其作用域选择为 *Action operation*，则该命令可供执行。

下面将介绍每种操作类型的参数。所有必填输入字段都以红色星号标记。完成后，单击 *Add* 将操作添加到 *Recovery operations* 列表中。

Note:
请注意，如果在多个操作类型中定义了相同的收件人，且未指定 *Custom message*，则不会发送重复通知。

操作类型：发送消息

参数		描述
发送给用户组		选择要向其发送恢复消息的用户组。 要收到通知，用户组必须对主机至少具有 “读取”权限。
发送给用户		选择要向其发送恢复消息的用户。 要收到通知，用户必须对主机至少具有 “读取”权限。
发送到媒介类型		将默认恢复消息发送到所有可用的（已配置并已启用）媒介类型，或仅发送到某一个特定媒介类型。
自定义消息		如果选中，则可以定义自定义消息。
	主题	自定义消息的主题。主题中可以包含宏。
	消息	自定义消息内容。消息中可以包含宏。

操作类型：远程命令

参数	描述
目标列表	选择要在其上执行命令的目标： 当前主机 - 命令将在触发问题事件的触发器所属主机上执行。如果触发器中包含多个主机，则此选项不起作用。 主机 - 选择要在其上执行命令的主机。 主机组 - 选择要在其上执行命令的主机组。指定父主机组会隐式选择所有嵌套的主机组。因此，远程命令也将在嵌套组中的主机上执行。 即使某个主机匹配多次（例如，来自多个主机组，或分别匹配主机和主机组），在该主机上也只执行一次命令。 如果命令在 Zabbix 服务器上执行，则目标列表没有意义。在这种情况下选择更多目标，只会导致命令在服务器上执行更多次。 请注意，对于全局脚本，目标选择还取决于全局脚本配置中的主机组设置。

操作类型：通知所有相关人员

参数		描述
自定义消息		如果选择此项，可以定义自定义消息。
	主题	自定义消息的主题。主题中可以包含宏。
	消息	自定义消息。消息中可以包含宏。

更新操作

概述

更新操作可用于具有以下事件来源的动作中：

- 触发器 - 当问题被其他用户更新时，即被评论、确认、严重性已更改、已关闭（手动）；
- 服务 - 当某个服务的严重性已发生变化，但该服务仍未恢复时。

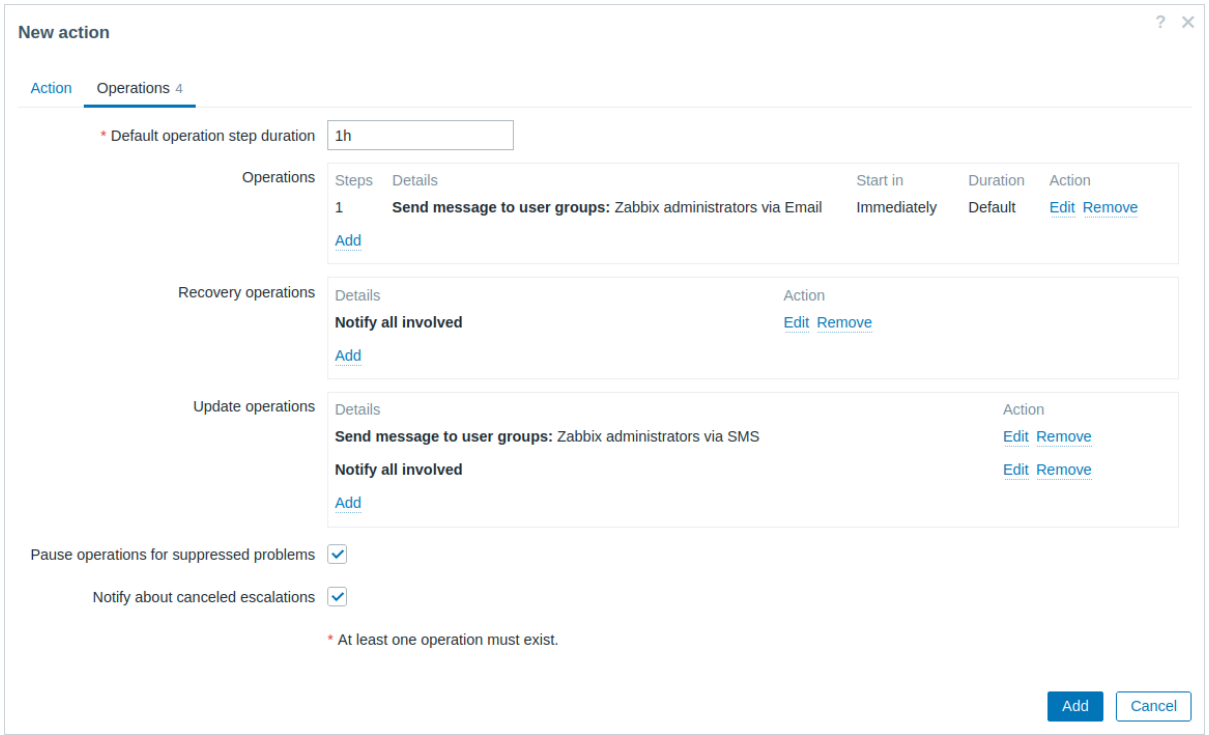
请注意，用户不会收到关于自己所做更新的通知。

更新操作同时支持消息和远程命令。虽然可以添加多个操作，但不支持升级 - 所有操作都分配到同一个步骤，因此会同时执行。

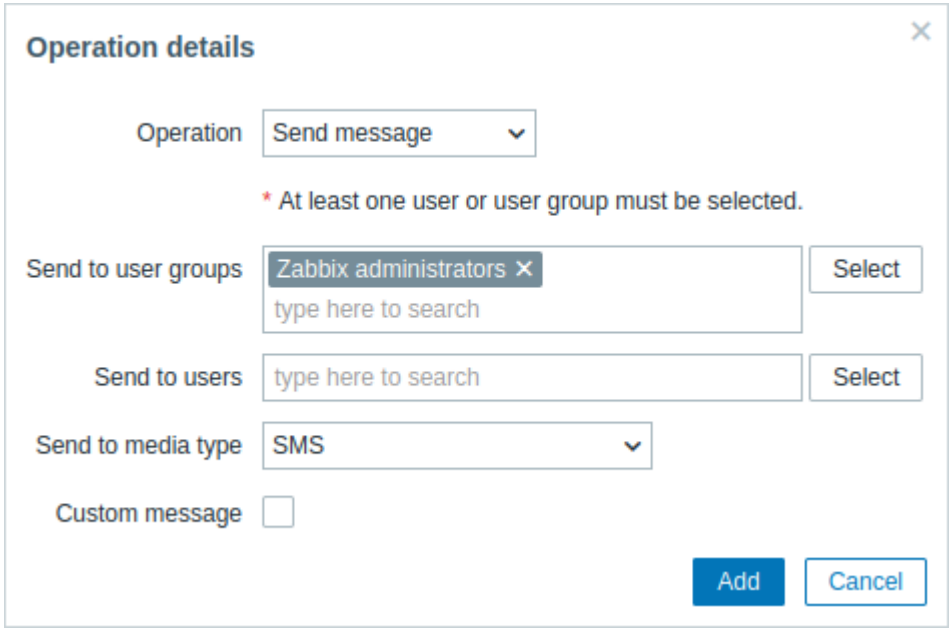
配置更新操作

要配置更新操作，请按以下步骤操作：

1. 在action 配置表单中，转到 操作选项卡。
2. 要配置新的更新操作，请在 更新操作部分中单击 添加。要编辑现有操作，请单击该操作旁边的 编辑。
3. 配置操作详细信息。



更新操作详情



更新操作提供与恢复操作相同的一组参数。

升级操作

概述

通过升级，您可以创建用于发送通知或执行远程命令的自定义场景。

实际来说，这意味着：

- 用户可以立即获知新问题。
- 通知可以重复发送，直到问题被解决。
- 发送通知可以延迟。
- 通知可以升级到另一个 “更高” 的用户组。
- 远程命令可以立即执行，或者在问题长时间未解决时执行。

操作会根据升级步骤进行升级。每个步骤都有一个持续时间。

您既可以定义默认持续时间，也可以为单个步骤定义自定义持续时间。一个升级步骤的最短持续时间为 60 秒。

您可以从任意步骤开始执行操作，例如发送通知或执行命令。第一步用于立即执行的操作。如果您希望延迟某个操作，可以将其分配到后续步骤。对于每个步骤，都可以定义多个操作。

升级步骤的数量没有限制。

升级是在配置操作时定义的。仅问题操作支持升级，不支持恢复操作。

升级行为的其他方面

下面介绍在一个动作包含多个升级步骤时，不同情况下会发生什么。

情况	行为
相关主机在发送初始问题通知后进入维护状态	根据动作配置中的暂停被抑制问题的操作设置，所有剩余的升级步骤要么因维护期间而延迟执行，要么立即执行。维护期间不会取消操作。
初始通知发送后，时间段动作条件中定义的时间段结束	所有剩余的升级步骤都会执行。时间段条件无法停止操作；它只影响动作是否启动以及何时启动，而不影响操作。
问题在维护期间开始，并在维护结束后仍持续存在（未恢复）	根据动作配置中的暂停被抑制问题的操作设置，所有升级步骤要么从维护结束时开始执行，要么立即执行。
问题在无数据维护期间开始，并在维护结束后仍持续存在（未恢复）	必须等待触发器触发后，所有升级步骤才会执行。
不同的升级接连发生并相互重叠	每个新升级的执行都会取代之前的升级，但之前的升级始终至少会执行一个升级步骤。此行为适用于针对触发器每次问题评估都创建的事件执行的动作。

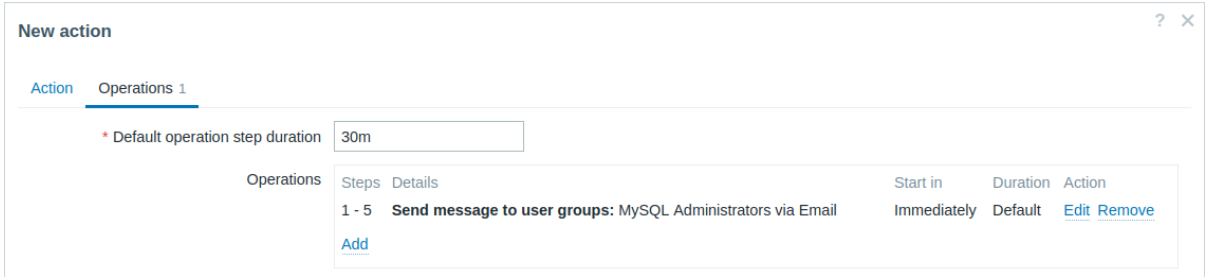
情况	行为
在升级进行期间（例如正在发送消息时），基于任意类型的事件： - 动作被禁用 基于触发器事件： - 触发器被禁用 - 主机或监控项被禁用 基于有关触发器的内部事件： - 触发器被禁用 基于有关监控项/低级别发现规则的内部事件： - 监控项被禁用 - 主机被禁用	正在发送的消息会发送完成，随后还会针对该升级发送一条消息。后续消息将在消息正文开头包含取消文本（注意：升级已取消），并说明原因（例如，注意：升级已取消：动作 "< 动作名称 >" 已禁用）。这样，接收者可以获知升级已取消，且不会再执行其他步骤。此消息会发送给之前收到通知的所有人员。取消原因也会记录到服务器日志文件中（从调试级别 3=Warning 开始）。
在升级进行期间（例如正在发送消息时）删除动作	请注意，即使操作已完成，但恢复操作已配置且尚未执行，也会发送升级已取消消息。 不会再发送其他消息。相关信息会记录到服务器日志文件中（从调试级别 3=Warning 开始），例如：`escalation canceled: action id:334 deleted`

升级示例

示例 1

每 30 分钟向 "MySQL Administrators" 组发送一次重复通知（总共 5 次）。配置方法如下：

- 在 *Operations* 选项卡中，将 *Default operation step duration* 设置为 "30m"（30 分钟）。
- 将升级的 *Steps* 设置为从 "1" 到 "5"。
- 选择 "MySQL Administrators" 组作为消息接收者。



通知将在问题开始后的 0:00、0:30、1:00、1:30、2:00 发送（当然，除非问题更早得到解决）。

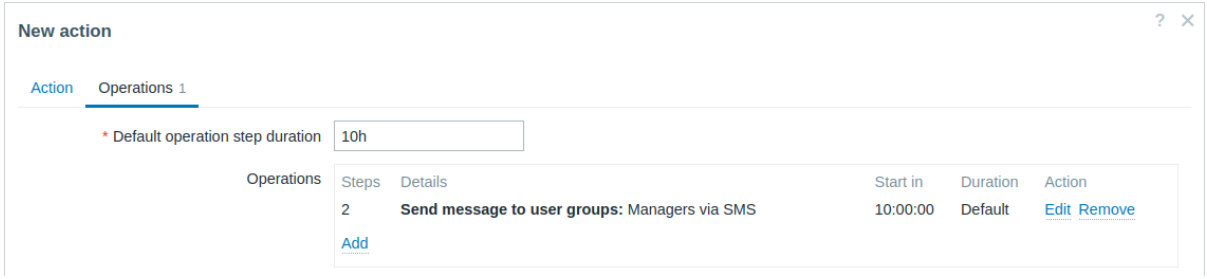
如果问题已解决并且配置了恢复消息，则恢复消息将发送给在此升级场景中至少收到过一条问题消息的人员。

Note:
如果生成活动升级的触发器被禁用，Zabbix 会向所有已经收到通知的人员发送一条相关提示消息。

示例 2

发送关于长期存在问题的延迟通知。配置方法如下：

- 在 操作选项卡中，将 默认操作步骤持续时间设置为 "10h"（10 小时）。
- 将升级的 步骤设置为从 "2" 到 "2"。



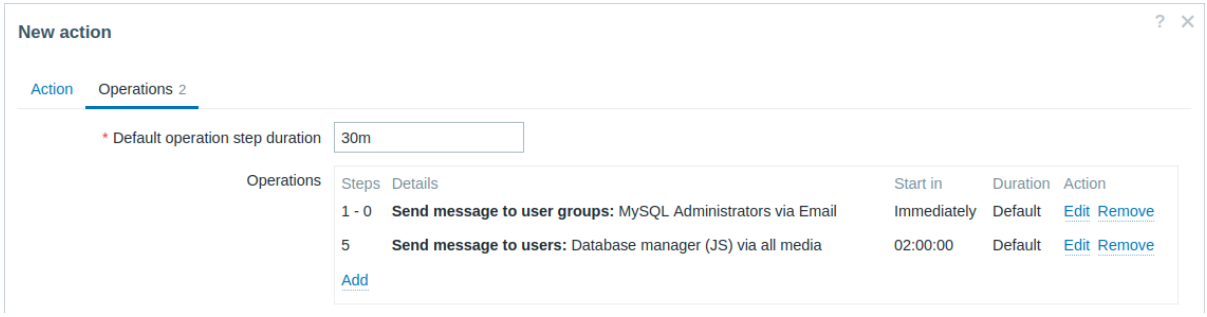
通知只会在升级场景的第 2 步发送，也就是在问题开始 10 小时后发送。

您可以将消息文本自定义为类似 "该问题已持续超过 10 小时" 的内容。

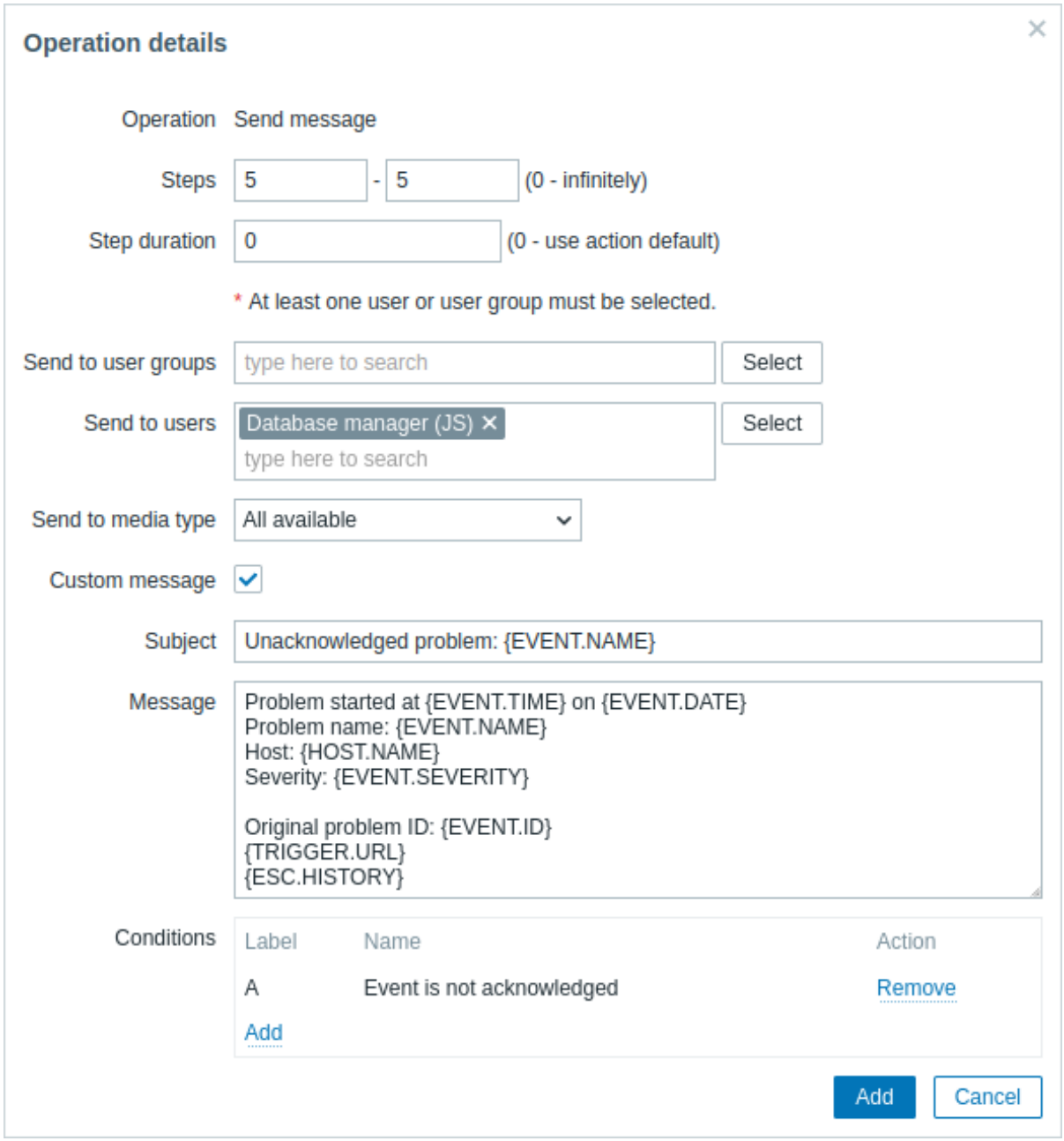
示例 3

将问题升级给老板。

在上面的第一个示例中，我们配置了定期向 MySQL 管理员发送消息。在这种情况下，在问题升级给数据库经理之前，管理员将收到四条消息。请注意，只有在问题尚未被确认的情况下，经理才会收到消息，这意味着可能还没有人正在处理该问题。



操作 2 的详细信息：



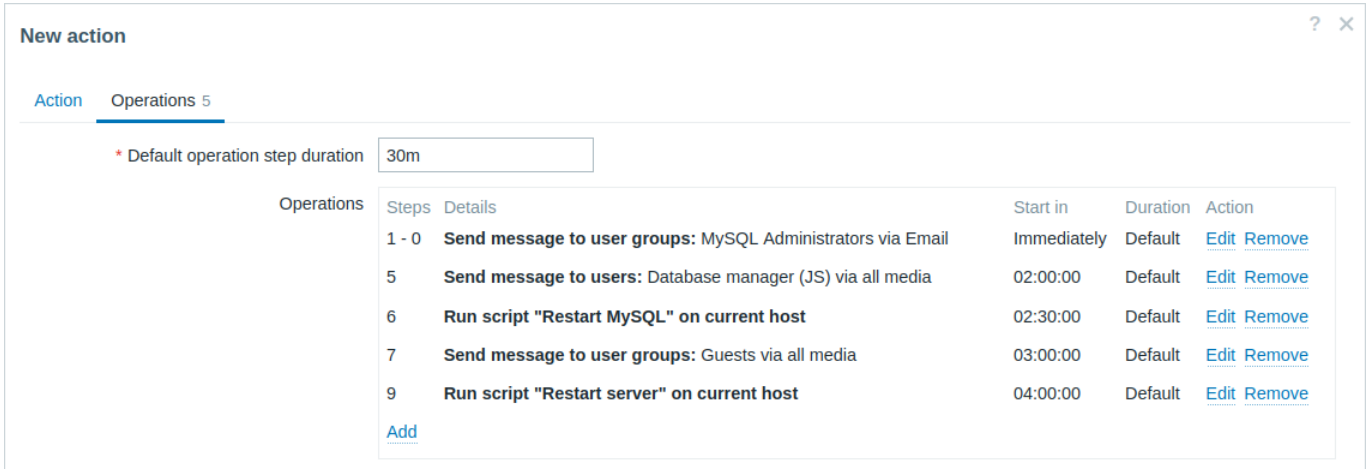
请注意在自定义消息中使用了 {ESC.HISTORY} 宏。该宏将包含有关此升级中之前所有已执行步骤的信息，例如已发送的通知和已执行的命令。

例 4

更复杂的情况。在向 MySQL 管理员发送多条消息并升级到 manager 后，Zabbix 将尝试重新启动 MySQL 数据库。如果问题存在 2:30 小时，但尚未确认，则会发生这种情况。

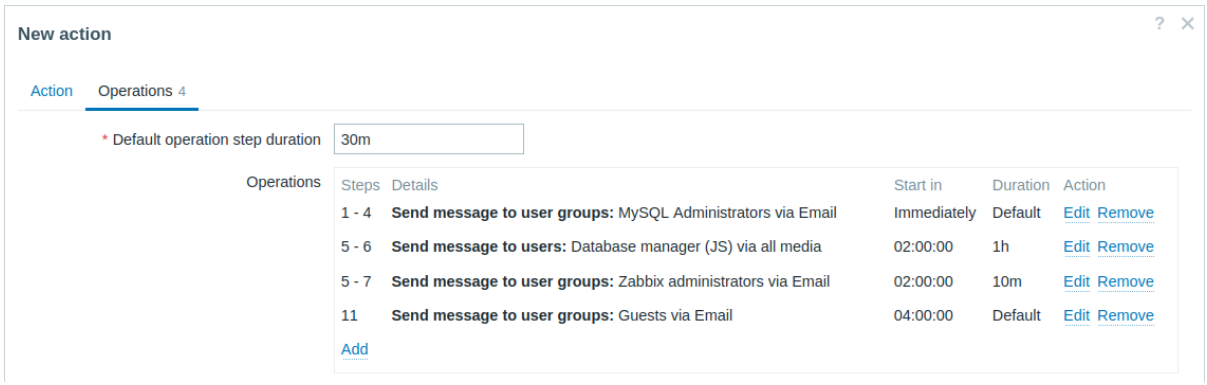
如果问题仍然存在，30 分钟后，Zabbix 将向所有来宾用户发送消息。

如果这没有帮助，再过一个小时，Zabbix 将使用 IPMI 命令使用 MySQL 数据库（第二个远程命令）重新启动服务器。



示例 5

一个升级，包含多个操作，这些操作具有重叠的步骤范围和自定义间隔。默认的操作步骤持续时间为 30 分钟。



通知将按如下方式发送：

- 在问题开始后的 0:00、0:30、1:00 和 1:30，发送给 MySQL 管理员。
- 在 2:00 和 2:10，发送给 Database manager（后续操作中定义的较短自定义步骤持续时间 10 分钟，会覆盖此处配置的较长步骤持续时间 1 小时，正如操作详情 中关于 步骤持续时间在步骤重叠时的说明）。
- 在问题开始后的 2:00、2:10 和 2:20，发送给 Zabbix 管理员（应用 10 分钟的自定义步骤持续时间）。
- 在问题开始后的 4:00，发送给 guest users（在步骤 8 到 11 之间，默认步骤持续时间 30 分钟生效）。

接收不受支持监控项的通知

概述

Zabbix 中可以接收关于不受支持监控项的通知。

这属于 Zabbix 内部事件概念的一部分，允许用户在这些情况下收到通知。内部事件 反映状态变化：

- 当监控项从 “正常” 变为 “不受支持”（以及恢复）时；
- 当触发器从 “正常” 变为 “未知”（以及恢复）时；
- 当低级别发现规则从 “正常” 变为 “不受支持”（以及恢复）时。

本节介绍当监控项变为不受支持时，如何接收通知。

配置

总的来说，以前在 Zabbix 中设置过警报的同学应该熟悉设置通知的过程。

第 1 步

配置用于通知的某种媒介，例如电子邮件、短信或脚本。请参阅手册中相应的章节来完成此任务。

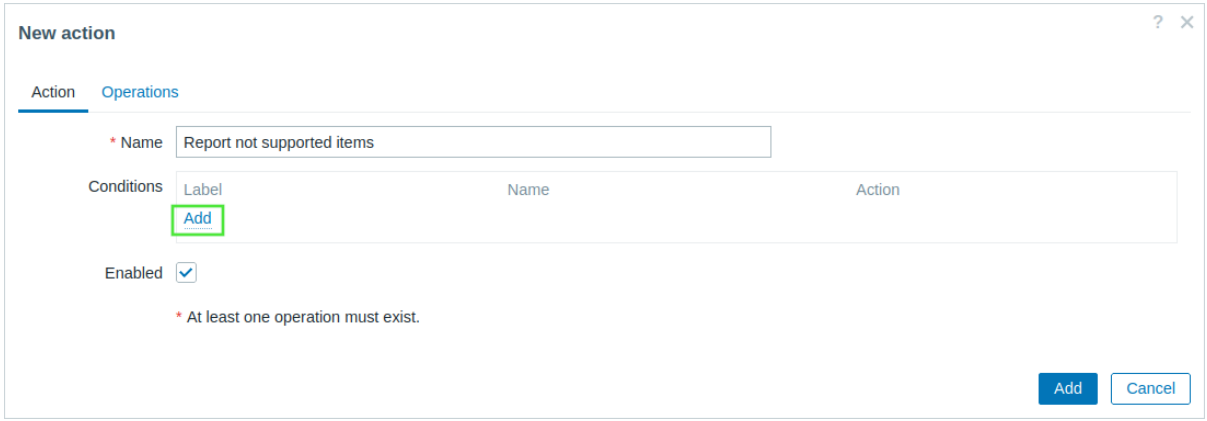
Attention:
对于内部事件通知，将使用默认严重性级别（“未分类”），因此如果您希望接收内部事件的通知，在配置用户媒介时请保持其处于选中状态。

步骤 2

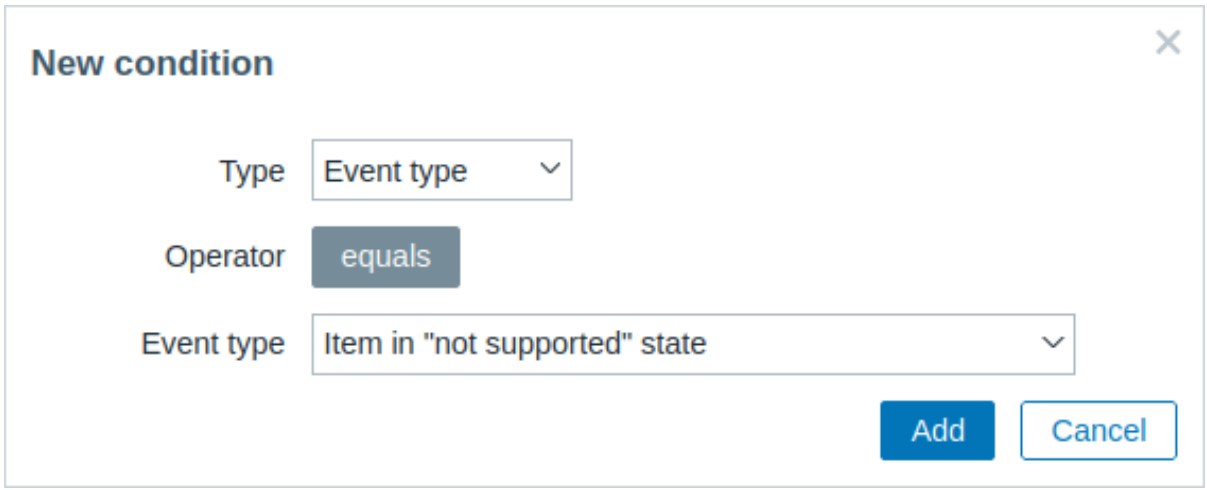
1. 转到 *Alerts* > *Actions* > *Internal actions*。
2. 单击页面右上角的 *Create action*，打开动作配置表单。

步骤 3

1. 在 操作选项卡中输入操作名称。
2. 在 条件区块中点击 添加，以添加新条件。



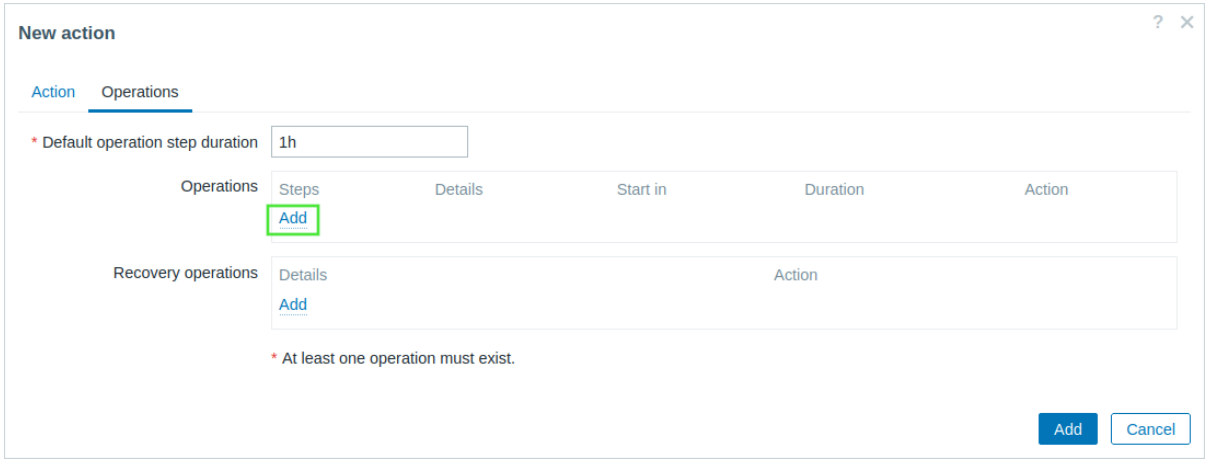
3. 在 新建条件弹出窗口中，将 “事件类型” 选择为条件类型，然后将事件类型选择为 “监控项处于 ‘不支持’ 状态”。



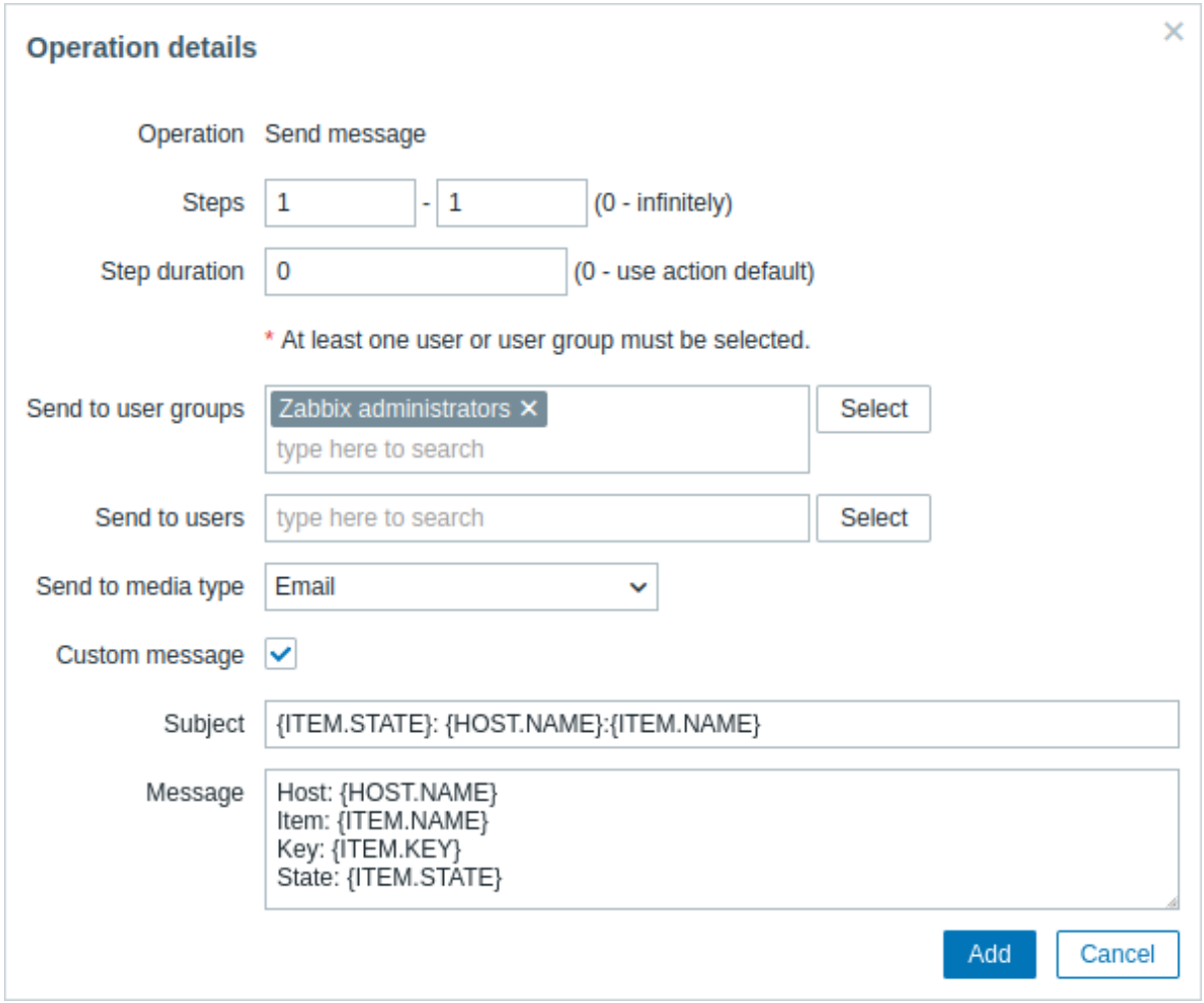
4. 点击 添加，将该条件实际列入 条件区块中。

步骤 4

1. 在 操作选项卡中，单击 操作区块中的 添加，以添加新操作。



2. 选择消息的部分接收者（用户组/用户）以及用于发送的媒介类型（或 “全部”）。
3. 如果希望输入问题消息的自定义主题/内容，请选中 自定义消息复选框。



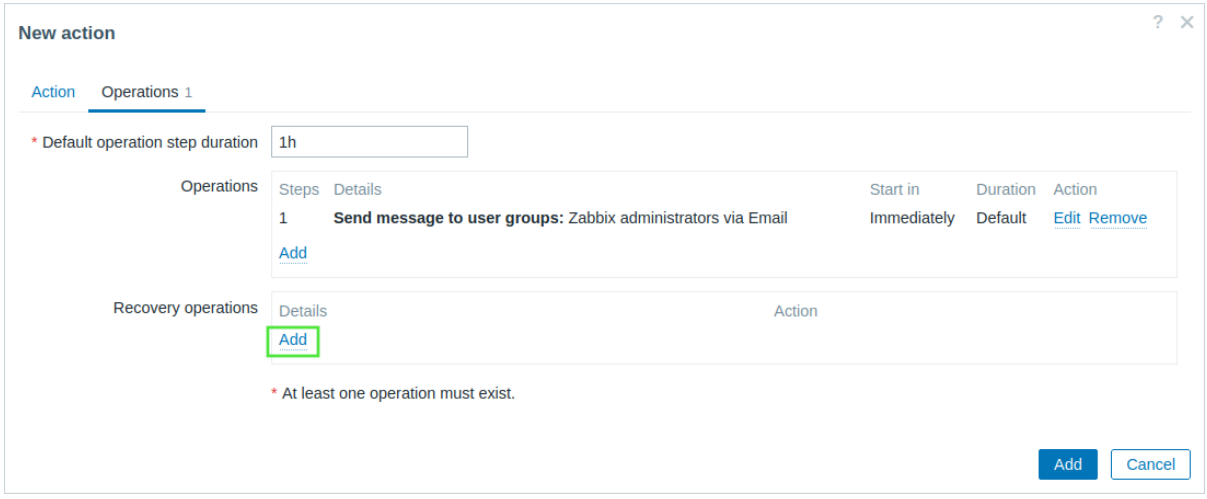
4. 单击 添加，将该操作实际列入 操作区块中。

如果希望接收多条通知，请设置操作步骤持续时间（发送消息之间的间隔），然后添加另一个步骤。

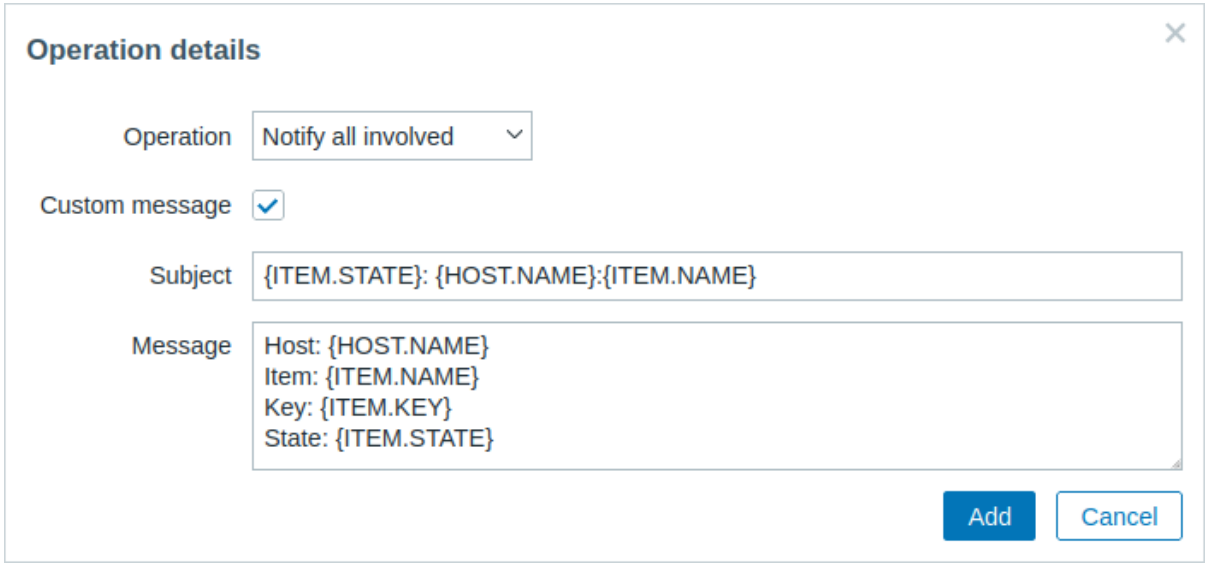
步骤 5

恢复操作块允许您配置监控项恢复正常状态时发送的恢复通知。

1. 在恢复操作块中点击添加，添加新的恢复操作。



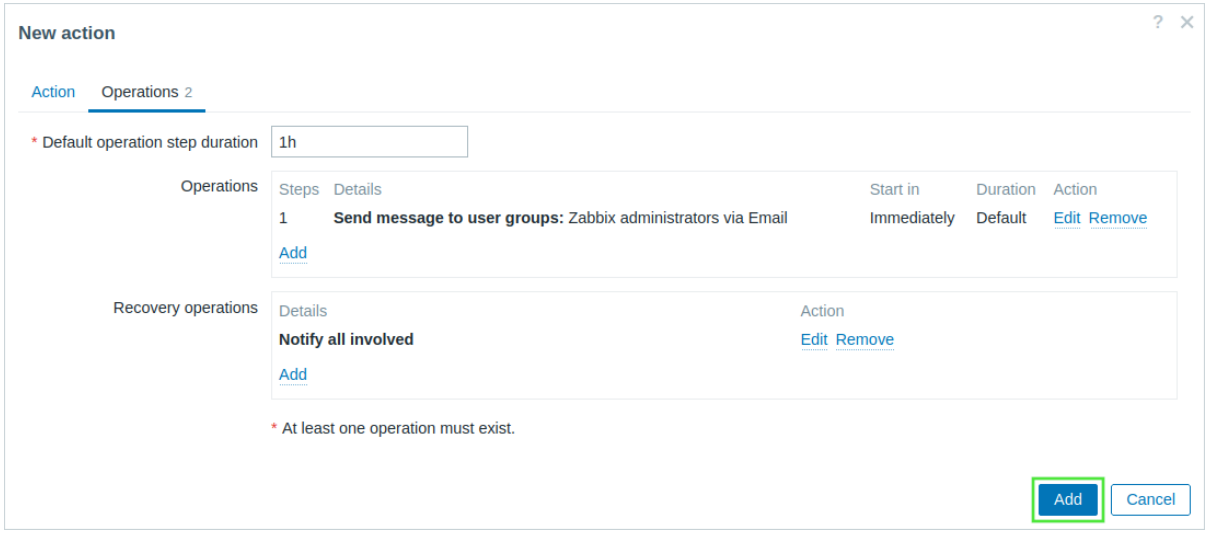
2. 选择操作类型 “通知所有相关人员”。
3. 如果希望输入恢复消息的自定义主题/内容，请选中自定义消息复选框。



4. 在操作详情弹出窗口中点击添加，即可将该操作实际列入恢复操作块中。

第 6 步

完成后，点击表单底部的 *Add* 按钮。



就这样，您已经完成了！现在，如果某个监控项变为不受支持，您就可以期待收到来自 Zabbix 的第一条通知了。

宏

概述

Zabbix 支持多种内置宏，可用于各种场景。这些宏是使用特定语法标识的变量：

`{MACRO}`

宏会根据上下文解析为特定值。

有效使用宏可以节省时间，并使 Zabbix 配置更加清晰。

宏的一个典型用途是在模板中使用宏。例如，模板中的触发器可以命名为 “Processor load is too high on {HOST.NAME}”。将模板应用于主机（例如 Zabbix 服务器）后，在 “监控” 部分显示触发器时，其名称将解析为 “Processor load is too high on Zabbix server”。

宏可以用于监控项键参数中。宏可以只用于参数的一部分，例如 `item.key[server_{HOST.HOST}_local]`。无需使用双引号括起参数，因为如果解析后的宏中存在可能引起歧义的特殊符号，Zabbix 会负责处理。

Zabbix 中还有其他类型的宏。

Zabbix 支持以下宏：

- `{MACRO}` - 内置宏（请参阅完整列表）

- {<macro>.<func>(<params>)} - 宏函数
- {$MACRO} - 用户宏，可选带上下文
- {#MACRO} - 用于低级别发现的宏
- {?EXPRESSION} - 表达式宏

宏函数

概述

宏函数提供了自定义macro 值的能力（例如，缩短或提取特定子字符串），使其更易于使用。

宏函数的语法如下：

```
{macro.func(params)}
```

其中：

- **macro** - 要自定义的宏；
- **func** - 要应用的函数（参见supported functions）；
- **params** - 以逗号分隔的函数参数列表；如果满足以下任一条件，则必须使用双引号括起来：
  - 以空格或双引号开头；
  - 包含右括号或逗号。

例如：

```
{{TIME}.fmttime(format,time_shift)}
{{ITEM.VALUE}.regsub(pattern, output)}
{{$USERMACRO}.regsub(pattern, output)}
{{#LLDMACRO}.regsub(pattern, output)}
```

宏函数支持以下对象：

- 内置宏
- 用户宏
- 低级别发现宏
- 表达式宏

宏函数可用于所有支持所列宏的位置。除非明确说明仅期望使用宏（例如，在配置host macros 或低级别发现规则filters 时），否则均适用。

每个宏仅支持一个函数；不支持将多个宏函数串联使用。

> **Note:**
> 当宏函数用于其他上下文中（函数、监控项键、另一个宏等）时，请参见escaping examples。

支持的函数

函数列表不包含其他信息。
点击函数可查看完整详细信息。

函数	描述
btoa	将宏值编码为 Base64 编码。
fmtnum	数字格式化，用于控制小数点后显示的位数。
fmttime	时间格式化。
htmldecode	从 HTML 编码中解码宏值。
htmlencode	将宏值编码为 HTML 编码。
iregsub	通过正则表达式匹配提取子字符串（不区分大小写）。
jsonpath	返回 JSONPath 结果。
lowercase	将宏值中的字符转换为小写。
regrepl	替换宏值中的字符/子字符串。
regsub	通过正则表达式匹配提取子字符串（区分大小写）。
tr	转换宏值中的字符。
uppercase	将宏值中的字符转换为大写。
urldecode	从 URL 编码中解码宏值。
urlencode	将宏值编码为 URL 编码。
xmlxpath	返回 XML XPath 结果。

函数详细信息

可选的函数参数用 < > 表示。

btoa

将宏值编码为 Base64 编码。Base64 编码是一种将二进制数据表示为文本的方法，有助于通过基于文本的协议存储和安全传输二进制内容。

示例：

```
{{ITEM.VALUE}.btoa()} - 将把类似 "zabbix" 的值进行 Base64 编码为 "emFiYml4"
```

fmtnum(digits)

用于控制小数点后打印位数的数字格式化。

参数：

- **digits** - 小数点后的位数。有效范围：0-20。不会生成末尾的零。

示例：

```
{{ITEM.VALUE}.fmtnum(2)} - 将从接收到的值 "24.3483523" 返回 "24.35"
{{ITEM.VALUE}.fmtnum(0)} - 将从接收到的值 "24.3483523" 返回 "24"
```

fmttime(format,<time_shift>)

时间格式化。
 请注意，此函数可与解析为以下任一时间格式值的宏一起使用：

- hh:mm:ss
- yyyy-mm-ddThh:mm:ss[tz]（ISO8601 标准）
- UNIX 时间戳

参数：

- **format** - 必填的格式字符串，与 strftime 函数的格式化方式兼容；
- **time_shift**（可选）- 在格式化之前应用于时间的时间偏移；应以 -<N><time_unit> 或 +<N><time_unit> 开头，其中：
  - N - 要加上或减去的时间单位数量；
  - time_unit - h（小时）、d（天）、w（周）、M（月）或 y（年）。

注释：

- time_shift 参数支持多步时间操作，并且可以包含 /<time_unit>，用于将时间偏移到该时间单位的起始位置（/d - 午夜，/w - 一周的第一天（星期一），/M - 一个月的第一天，等等）。示例：-1w - 正好向前 7 天；-1w/w - 上一周的星期一；-1w/w+1d - 上一周的星期二。
- 时间操作按从左到右计算，不考虑优先级。例如，-1M/d+1h/w 将被解析为 ((-1M/d)+1h)/w。

示例：

```
{{TIMESTAMP}.fmttime(%B)} - 将从接收到的值 "1633098961" 返回 "October"
{{TIMESTAMP}.fmttime(%d %B,-1M/M)} - 将从接收到的值 "1633098961" 返回 "1 September"
{{EVENT.TIMESTAMP}.fmttime("%x %X")} - 将返回日期和时间的区域设置特定表示；例如，对于事件时间戳 "1784100339"，返回 "07
{{EVENT.TIME}.fmttime("%x %X")} - 将返回日期和时间的区域设置特定表示；例如，对于与该事件关联的时间戳 "1784100339"，返回
```

htmldecode

将宏值从 HTML 编码中解码。

支持以下字符：

Value	解码后的值
&	&
<	<
>	>
"	"
'	'
'	'

示例：

```
{{ITEM.VALUE}.htmldecode()} - 将把类似 "&lt;" 的值进行 HTML 解码为 "<"
```

htmlencode

将宏值编码为 HTML 编码。

支持以下字符：

值	编码后的值
&	&
<	<
>	>
"	"
'	'

示例：

```
{{ITEM.VALUE}.htmlencode()} - 将把类似 "<" 这样的字符进行 HTML 编码为 "&lt;"
```

iregsub(pattern,output)

通过正则表达式匹配提取子字符串（不区分大小写）。

参数：

- **pattern** - 要匹配的正则表达式；
- **output** - 输出选项。支持使用 **\1 - \9** 占位符表示捕获组。**\0** 返回匹配到的文本。

说明：

- 如果正则表达式没有匹配项，该函数将返回空字符串。
- 如果函数中的 pattern 是错误的正则表达式，则该宏将被解析为 “UNKNOWN”（低级发现宏除外，在这种情况下该函数将被忽略，宏将保持未解析状态）。
- 替换字符串中对不存在的捕获组的引用将被替换为空字符串。

示例：

```
{{ITEM.VALUE}.iregsub("fail|error|fault|problem","ERROR")} - 如果接收到 "fail"、"error"、"fault" 或 "problem"
```

jsonpath(pattern,<default>)

返回 JSONPath 结果。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **pattern** - 必需的模式；
- **default** - 可选的回退值，当 JSONPath 查询未返回数据时使用。请注意，发生其他错误时将返回失败。

注释：

- 在以下情况下，该函数会返回错误：
  - JSON 无效。
  - JSONPath 查询失败。
  - JSONPath 结果不包含数据，且未提供 `default` 参数。

示例：

```
{{ITEM.VALUE}.jsonpath("$.columns.servicename", "NA")}
```

小写

将宏值中的所有字符转换为小写。
适用于单字节字符集（如 ASCII），不支持 UTF-8。

示例：

```
{{ITEM.VALUE}.lowercase()} - 将把类似 "Zabbix SERVER" 的值转换为 "zabbix server"（小写）
```

regrepl(pattern,replacement,<pattern2>,<replacement2>,...)

在宏值中替换字符/子字符串。

参数：

- **pattern** - 要匹配的正则表达式；
- **replacement** - 替换字符串。支持在替换字符串中使用 **\1 - \9** 占位符来捕获组。

注释：

- 所有匹配该模式的出现都会被替换。
- 模式和替换会按顺序处理，每一对后续的模式/替换都会根据前一次替换的结果应用。
- 替换字符串中对不存在的捕获组的引用会被替换为空字符串。

示例：

```
{{ITEM.VALUE}.regrepl("oldParam", "newParam")} - 将把 "oldParam" 替换为 "newParam"
{{ITEM.VALUE}.regrepl("([^a-z])","\\\1")} - 所有非字母字符都会用反斜杠进行转义
{$THRESHOLD:"{{#FSNAME}.regrepl(\"\\$\",\"\")}"} - 将移除末尾的反斜杠（例如，将 "C:\" 替换为 "C:"）
{{ITEM.VALUE}.regrepl("_v1\.0", "_v2.0", "\(final\)", "")} - 将替换监控项值中的多个部分
```

regsub(pattern,output)

通过正则表达式匹配提取子字符串（区分大小写）。

参数：

- **pattern** - 要匹配的正则表达式；
- **output** - 输出选项。支持使用 **\1 - \9** 占位符表示捕获组。**\0** 返回匹配到的文本。

说明：

- 如果正则表达式没有匹配项，该函数将返回空字符串。
- 如果函数中的 pattern 是错误的正则表达式，则宏将被解析为 “UNKNOWN”（低级别发现宏除外，在这种情况下该函数将被忽略，宏将保持未解析状态）。
- 替换字符串中对不存在的捕获组的引用将被替换为空字符串。

示例：

```
{{ITEM.VALUE}.regsub("^([0-9]+)", Problem ID: \1)} - 如果接收到类似 "123 Log line" 的值，将解析为 "Problem ID:
{{ITEM.VALUE}.regsub("fail|error|fault|problem","ERROR")} - 如果接收到 "fail"、"error"、"fault" 或 "problem"
```

请参见更多示例。

tr(characters,replacement)

宏值字符的音译转换。

- **characters** - 要替换的字符集；
- **replacement** - 按位置一一对应的替换字符集。

示例：

```
{{ITEM.VALUE}.tr(abc, xyz)} - 将把所有出现的 "a" 替换为 "x"，"b" 替换为 "y"，"c" 替换为 "z"
{{ITEM.VALUE}.tr(abc, xyzq)} - 将把所有出现的 "a" 替换为 "x"，"b" 替换为 "y"，"c" 替换为 "z"（"q" 将被忽略）
{{ITEM.VALUE}.tr(abcde, xyz)} - 将把所有出现的 "a" 替换为 "x"，"b" 替换为 "y"，"c" 替换为 "z"，"d" 替换为 "z"，"e"
{{ITEM.VALUE}.tr("\\'", "\/\"")} - 将把所有出现的反斜杠替换为正斜杠，单引号替换为双引号
{{ITEM.VALUE}.tr(A-Z,a-z)} - 将把所有字母转换为小写
{{ITEM.VALUE}.tr(0-9a-z,*)} - 将把所有数字和小写字母替换为 "*"
{{ITEM.VALUE}.tr(0-9,ab)} - 将把所有出现的 0 替换为 "a"，并将所有出现的 1、2、3、4、5、6、7、8 和 9 替换为 "b"
{{ITEM.VALUE}.tr(0-9abcA-L,*)} - 将把所有数字、"abc" 字符以及 A-L 范围替换为 "*"
{{ITEM.VALUE}.tr("\n","*")} - 将把行尾出现的位置替换为 *
{{ITEM.VALUE}.tr("e", "\n")} - 将把所有 "e" 替换为行尾
```

要包含字面字符：

```
backslash - 必须转义为 \\
single quote - 必须转义为 \'
double quote - 必须转义为 \"
```

反斜杠支持的转义序列：

```
\\\\ => \\ - 双反斜杠变为单反斜杠
\\a  => \a - 响铃
\\b  => \b - 退格
\\f  => \f - 换页
\\n  => \n - 换行
\\r  => \r - 回车
\\t  => \t - 水平制表符
\\v  => \v - 垂直制表符
```

大写

将宏值中的所有字符转换为大写。
适用于单字节字符集（如 ASCII），不支持 UTF-8。

示例：

```
{{ITEM.VALUE}.uppercase()} - 将把类似 "Zabbix Server" 的值转换为 "ZABBIX SERVER"（大写）
```

urldecode

将宏值从 URL 编码中解码。

示例：

```
{{ITEM.VALUE}.urldecode()} - 将把类似“%2F”的值进行 URL 解码为“/”
```

urlencode

将宏值编码为 URL 编码格式。

示例：

```
{{ITEM.VALUE}.urlencode()} - 将把像 "/" 这样的字符进行 URL 编码为 "%2F"
```

xmlxpath(pattern,<default>)

返回 XML XPath 结果。
 支持的值类型：*String*、*Text*、*Log*。

参数：

- **pattern** - 必需的模式；
- **default** - 可选的回退值，当 XPath 查询未返回数据时使用。请注意，发生其他错误时将返回失败。

注释：

- 在以下情况下，该函数会因错误而失败：
  - XML 无效。
  - XPath 查询失败。
  - XPath 结果不包含数据，且未提供 default 参数。
  - Zabbix 在编译时未启用 XML 支持。

示例：

```
{{ITEM.VALUE}.xmlxpath("/response/error/status","unknown")}
```

更多示例

下表展示了使用宏函数的更多示例。

{#IFALIAS} 是一个LLD macros，仅在低级别发现上下文中定义（发现规则、原型以及由它们创建的监控项/触发器）。
在 LLD 之外使用它将导致该标记无法展开。

宏函数	接收值	输出
{{ITEM.VALUE}.regsub(^[0-9]+, Problem)}	123Log line	Problem
{{ITEM.VALUE}.regsub("^([0-9]+)", "Problem")}	123 Log line	Problem
{{ITEM.VALUE}.regsub(".*", "Problem ID: \1")}	Log line	Problem ID:
{{ITEM.VALUE}.regsub("^(\w+).*?([0-9]+)", " Problem ID: \1_\2 ")}	MySQL crashed errno 123	Problem ID: MySQL_123
{{ITEM.VALUE}.regsub("([1-9]+", "Problem ID: \1")}	123 Log line	*UNKNOWN*（无效的正则表达式）
{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}	customername_1	customername
{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}	customername_1	1
{{#IFALIAS}.regsub("(.*)_([0-9]+", \1)}	customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+", \1)}（无效的正则表达式）
{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+)\", \1)}"}	customername_1	{$MACRO:"customername"}

宏函数	接收值	输出
{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+\", \2)}"}	customername_1	{$MACRO:"1"}
{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+\", \1)}"}	customername_1	{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+\", \1)}"}（无效的正则表达式）
"{$MACRO:"\{{#IFALIAS}.regsub("(.*)_([0-9]+", \1)}\"}"	customername_1	"{$MACRO:"\customername\"}"
"{$MACRO:"\{{#IFALIAS}.regsub("(.*)_([0-9]+", \2)}\"}"	customername_1	"{$MACRO:"\1\"}"
"{$MACRO:\"{{#IFALIAS}.regsub(\\"(.*)_([0-9]+\\", \1)}\"}"	customername_1	"{$MACRO:\"{{#IFALIAS}.regsub(\\"(.*)_([0-9]+\\", \1)}\"}"（无效的正则表达式）

查看完整的监控项值

对于文本/log 监控项，已解析的 {ITEM.VALUE} 和 {ITEM.LASTVALUE} 宏的长值在某些前端位置会被截断为 20 个字符。要查看这些宏的完整值，可以使用宏函数，例如：

```
{{ITEM.VALUE}.regsub("(.*)", \1)}
{{ITEM.LASTVALUE}.regsub("(.*)", \1)}
```

另请参见：{ITEM.VALUE} 和 {ITEM.LASTVALUE} 的macro details。

用户宏

概述

除了开箱即用的支持的宏之外，Zabbix 还支持用户宏，以提供更大的灵活性。

用户宏可以在全局、模板和主机级别定义。这些宏具有特殊语法：

{$MACRO}

Zabbix 按照以下优先级解析宏：

1. 主机级别的宏（首先检查）。
2. 为主机的第一级模板定义的宏（即直接链接到主机的模板），按模板 ID 排序。
3. 为主机的第二级模板定义的宏，按模板 ID 排序。
4. 为主机的第三级模板定义的宏，按模板 ID 排序，依此类推。
5. 全局宏（最后检查）。

换句话说，如果主机没有定义某个宏，Zabbix 将尝试在主机的模板中按深度递增的顺序查找该宏。如果仍未找到，则在全局宏存在的情况下使用全局宏。

Warning:
如果同一级别的多个已链接模板中存在名称相同的宏，则使用模板 ID 最低的模板中的宏。因此，在多个模板中使用相同名称的宏存在配置风险。

如果 Zabbix 无法找到某个宏，则不会解析该宏。

Attention:
在“配置”部分（例如触发器列表）中，宏（包括用户宏）会按设计保持未解析状态，以便更透明地显示复杂配置。

用户宏可以用于：

- 监控项名称
- 监控项键参数
- 监控项更新间隔和灵活间隔
- 触发器名称和描述
- 触发器表达式参数和常量（请参阅示例）
- 许多其他位置 - 请参阅完整列表

全局和宿主宏的常见用例

- 在多个位置使用全局宏；然后更改宏值并一键将配置更改应用于所有位置
- 利用具有主机特定属性的模板：密码、端口号、文件名、正则表达式等。

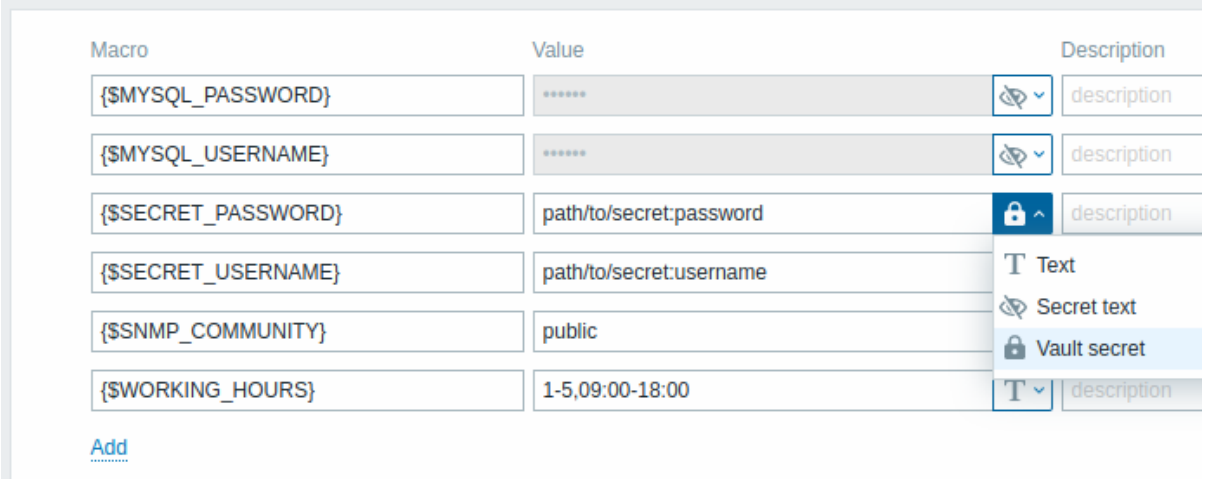
Note:
建议使用主机宏而不是全局宏，因为添加、更新或删除全局宏会强制对所有主机进行增量配置更新。有关更多信息，请参见Passive and active agent checks。

配置

要定义用户宏，请转到前端中的相应位置：

- 对于全局宏，请访问 *Administration* > *Macros*。
- 对于主机和模板级别的宏，请打开主机或模板属性，并查找 *Macros* 选项卡。

用户宏具有以下属性：



Parameter	Description
Macro	宏名称。名称必须用花括号括起来，并以美元符号开头。 示例：{$FRONTEND_URL}。宏名称中允许使用以下字符：**A-Z**（仅大写）、**0-9** 、_ 、.
Value	宏值。支持三种值类型： **Text**（默认）- 纯文本值 **Secret text** - 值会以星号掩码显示 **Vault secret** - 值包含指向vault secret 的路径/查询。 要更改值类型，请单击值输入字段末尾的按钮。 用户宏值的最大长度为 2048 个字符。
Description	用于提供有关此宏更多信息的文本字段。

配置用户宏时，请注意以下与上下文相关的行为：

- 如果在模板监控项或触发器中使用了用户宏，请考虑将这些宏也添加到模板中（即使它们已在全局定义）；这样，在将模板导出为 XML 并导入到其他系统后，*Text* 类型宏仍可按预期工作（secret macro 值不会被导出）
- 如果在触发器表达式中使用了用户宏，则这些宏仅在引用参数或常量时才会解析；如果引用主机、监控项键值、函数、运算符或其他触发器表达式，则不会解析（secret macro 不能用于触发器表达式）
- 如果在具有低级别发现规则和主机原型的主机上使用了用户宏，发现的主机 将继承该主机上定义的所有用户宏

示例

示例 1

在 “SSH 守护进程状态” 监控项键中使用主机级别宏：

```
net.tcp.service[ssh,,{$SSH_PORT}]
```

只要在这些主机上定义了 {$SSH_PORT} 的值，就可以将此监控项分配给多个主机。

示例 2

在 “CPU 负载过高” 触发器中使用主机级别的宏：

```
last(/ca_001/system.cpu.load[,avg1])>{$MAX_CPULOAD}
```

此类触发器应在模板上创建，而不是在各个主机中编辑。

Note:
如果要将值的数量用作函数参数（例如，max(/host/key,#3)），请在宏定义中包含井号，如下所示：SOME_PERIOD => #3。

示例 3

在 “CPU 负载过高” 触发器中使用两个宏：

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

请注意，宏可以用作触发器函数的参数，本例中使用的是 min() 函数。

示例 4

将 agent 不可用条件与监控项更新间隔同步：

- 定义 {$INTERVAL} 宏，并在监控项更新间隔中使用它；
- 将 {$INTERVAL} 用作 agent 不可用触发器的参数：

```
nodata(/ca_001/agent.ping,{$INTERVAL})=1
```

示例 5

集中配置工作时间：

- 创建全局 {$WORKING_HOURS} 宏，并将其设置为 1-5,09:00-18:00；
- 在管理 > 常规 > 前端的工作时间字段中使用该宏；
- 在用户 > 用户中某个用户的媒体选项卡的活动时间字段中使用该宏；
- 使用该宏设置在工作时间内更频繁地轮询监控项：

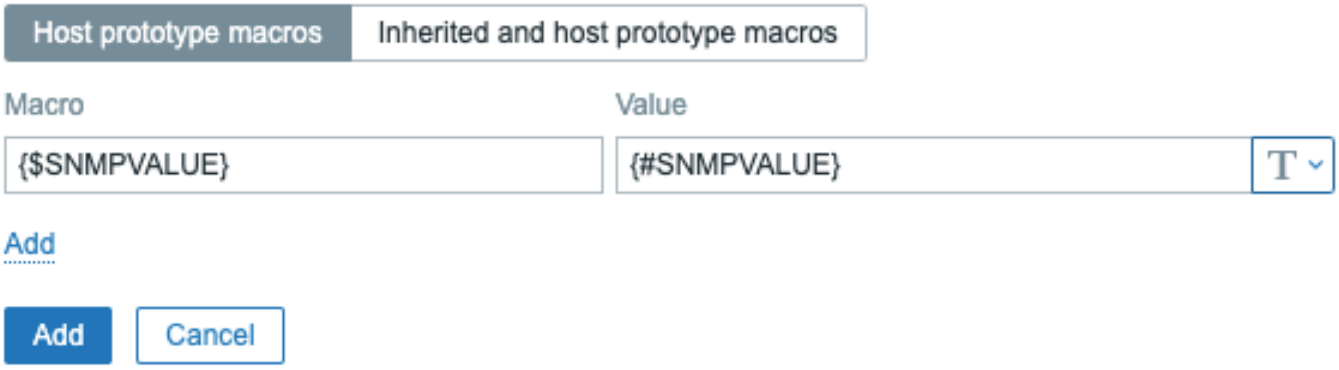


- 在操作条件的时间段中使用该宏；
- 如有需要，在管理 > 宏中调整工作时间。

示例 6

使用主机原型宏为发现的主机配置监控项：

1. 在主机原型上定义用户宏 {$SNMPVALUE}，并将 {#SNMPVALUE} 低级别发现宏设置为其值：



2. 将 *Generic SNMPv2* 模板分配给主机原型。
3. 在 *Generic SNMPv2* 模板监控项的 *SNMP OID* 字段中使用 {$SNMPVALUE}。

用户宏上下文

请参阅带上下文的用户宏。

带上下文的用户宏

概述

用户宏可以使用可选上下文，从而使用特定于上下文的值覆盖默认值。

上下文会附加到宏名称后；具体语法取决于上下文是否为静态文本值：

```
{$MACRO:"static text"}
```

或正则表达式：

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

请注意，带有正则表达式上下文的宏只能在用户宏配置中定义。如果在其他位置（例如触发器表达式中）将 `regex:` 前缀用作用户宏上下文，则会将其视为静态上下文。

上下文引号是可选的（另请参阅重要说明）。

宏上下文示例：

示例	描述
{$LOW_SPACE_LIMIT}	不带上下文的用户宏。
{$LOW_SPACE_LIMIT:/tmp}	带上下文的用户宏（静态字符串）。
{$LOW_SPACE_LIMIT:regex:"^/tmp$"}	带上下文的用户宏（正则表达式）。与 {$LOW_SPACE_LIMIT:/tmp} 相同。
{$LOW_SPACE_LIMIT:regex:"^/var/log/.*$"}	带上下文的用户宏（正则表达式）。匹配所有以 /var/log/ 开头的字符串。

用例

带上下文的用户宏可以用于在触发器表达式中实现更灵活的阈值（基于通过低级别发现检索到的值）。例如，您可以定义以下宏：

- {$LOW_SPACE_LIMIT} = 10
- {$LOW_SPACE_LIMIT:/home} = 20
- {$LOW_SPACE_LIMIT:regex:"^/[a-z]+$"} = 30

然后，低级别发现宏可以作为宏上下文，用于挂载文件系统发现的触发器原型中：

```
last(/host/vfs.fs.size[{#FSNAME},pfree])<{$LOW_SPACE_LIMIT:"{#FSNAME}"}
```

在发现之后，触发器中将根据已发现的挂载点或文件系统类型应用不同的低空间阈值。如果满足以下条件，将生成问题事件：

- /home 目录的可用磁盘空间少于 20%
- 与正则表达式模式匹配的目录（如 /etc、/tmp 或 /var）的可用磁盘空间少于 30%
- 不匹配正则表达式模式且不是 /home 的目录的可用磁盘空间少于 10%

重要笔记

- 如果存在多个具有上下文的用户宏，Zabbix 将尝试首先匹配简单的上下文宏，然后匹配上下文宏未定义顺序的正则表达式。

Warning:
不要创建不同的上下文宏匹配相同的字符串以避免未定义的行为。

- 如果在主机上找不到带有上下文的宏，则链接模板或全局，然后搜索没有上下文的宏。
- 上下文中仅支持低级发现宏。任何其他宏被忽略并被视为纯文本。

从技术上讲，宏上下文是使用类似于item key 参数，除了宏上下文是如果有 **,** 字符，则不被解析为多个参数：

- 如果上下文包含 }，则必须用 " 引用宏上下文字符或以 " 字符开头。引号内的引号 context 必须用 \ 字符转义。
- \ 字符本身没有被转义，这意味着它是不可能的有一个以 \ 字符结尾的引用上下文 - 宏 {$MACRO:"a:\b\c\"} 无效。
- 上下文中的前导空格被忽略，尾随空格是不是：
  - 例如 {$MACRO:A} 与 {$MACRO:A} 相同，但不同 {$ 宏：A}。
- 前导引号和尾随引号后的所有空格都是忽略，但引号内的所有空格都不是：
  - 宏 {$MACRO:"A"}, {$MACRO: "A"}, {$MACRO:"A" } 和 {$MACRO:"A" } 相同，但宏 {$MACRO:"A"} 和 {$MACRO:" A "} 不是。

以下宏都是等价的，因为它们具有相同的上下文：{$MACRO:A}, {$MACRO: A} 和 {$MACRO:"A"}。这是相反的带有监控项键，其中'key[a]', 'key[a]' 和'key["a"]' 是语义上相同，但出于唯一性目的而不同。

机密用户宏

概述

Zabbix 提供了两种用于保护用户宏值中敏感信息的选项：

- 密文
- Vault 密钥

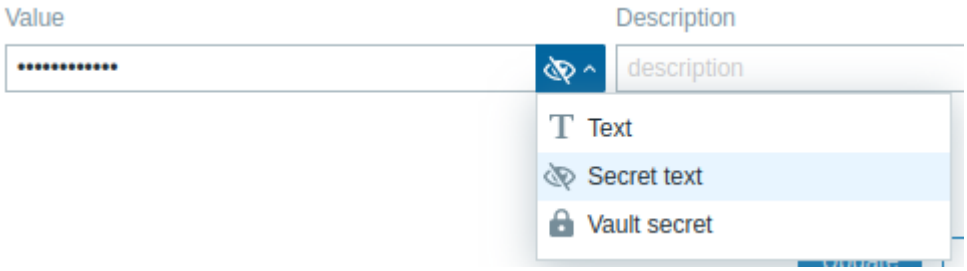
Note:
虽然密钥宏的值是隐藏的，但仍可通过在监控项中使用而被显示出来。例如，在外部脚本中，可使用引用密钥宏的 `echo` 语句将宏值显示到前端，因为 Zabbix 服务器可以访问宏的真实值。请参见密钥宏值会被取消掩码的位置。

密钥宏不能用于触发器表达式。

密文

对于密文宏，宏值会以星号掩码显示。

要将宏值设为密文，请点击 *Value* 字段末尾的按钮并选择 *Secret text* 选项：



一旦配置保存，就将无法再查看该值。

要更改宏值，请将鼠标悬停在 *Value* 字段上，然后点击 *Set new value* 按钮（悬停时显示）：



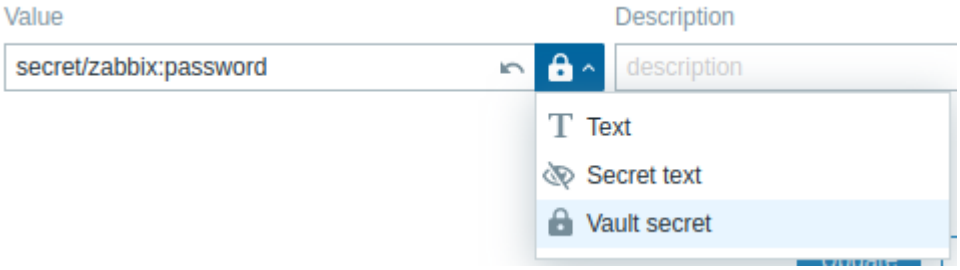
当您点击 *Set new value* 按钮（或更改宏值类型）时，当前值将被清除。您可以通过点击 *Value* 字段末尾的 箭头来恢复原始值（仅在保存新配置之前可用）。请注意，恢复原始值不会暴露其内容。

Note:
包含密文宏的 URL 将无法使用，因为其中的宏会被解析为”******”。

Vault secret

使用 Vault secret 宏时，宏值会存储在外部密钥管理软件（vault）中。

要配置 Vault secret 宏，请单击 *Value* 字段末尾的按钮，并选择 *Vault secret* 选项：



宏值必须指向一个 vault secret。输入格式取决于 vault 提供程序。有关特定提供程序的配置示例，请参见：

- HashiCorp
- CyberArk

Vault 密钥宏值由 Zabbix 服务器（以及 Zabbix proxy，如果 *Resolve secret vault macros by* 设置为 *Zabbix server and proxy*）在每次刷新配置数据时从 vault 中获取，然后存储在配置缓存中。
Zabbix 服务器和 Zabbix proxy 可以使用不同的 vault。

如果 *Resolve secret vault macros by* 设置为 *Zabbix server*，则 vault 密钥仅由服务器获取，而 Zabbix proxy 会在每次配置同步时从 Zabbix 服务器接收 Vault 密钥宏的值，并将其存储在自己的配置缓存中。
这意味着，Zabbix proxy 在重启后必须先收到来自 Zabbix 服务器的配置更新，才能开始采集数据。

要手动从 vault 刷新密钥值，请使用 `secrets_reload` 运行时控制 选项（仅服务器）。

必须在 Zabbix 服务器与 proxy 之间启用加密；否则会记录一条服务器警告消息。

Warning:
如果某个宏值无法成功获取，使用该值的相应监控项将变为不支持。

未脱敏的位置

此列表提供了参数位置，在这些位置中，密钥宏值不会被脱敏。

Note:
如果通过间接方式引用，下面位置中的密钥宏值仍将保持脱敏。例如，在媒体类型（Script 或 webhook 参数）中使用的 {ITEM.KEY}, {ITEM.KEY<1-9>}, {LLDRULE.KEY} 内置宏 将解析为包含已脱敏密钥宏的监控项键，例如 `net.tcp.port[******,******]`，而不是 `net.tcp.port[192.0.2.1,80]`。

Context		Parameter
监控项、监控项原型、**LLD** 规则		
	监控项	监控项键参数
	监控项原型	监控项原型键参数
	低级别发现规则	发现监控项键参数
	SNMP agent	*SNMP community*
		Context name (SNMPv3)
		Security name (SNMPv3)
		Authentication passphrase (SNMPv3)
		Privacy passphrase (SNMPv3)
	HTTP agent	*URL*
		Query fields
		Request body
		Headers
		User name
		Password
		SSL key password
	Script	*Parameters*
		Script
	Browser	*Parameters*
		Script
	Database monitor	*SQL query*
	TELNET agent	*Script*
		User name
		Password
	SSH agent	*Script*
		User name
		Password
	Simple check	*User name*
		Password
	JMX agent	*User name*
		Password
监控项值预处理		
	JavaScript 预处理步骤	*Script*
Web 场景		
	Web 场景	*Variable value*
		Header value
		URL
		Query field value
		Post field value
		Raw post
	Web 场景认证	*User*
		Password
		SSL key password
连接器		
	连接器	*URL*
		Username
		Password
		Token
		HTTP proxy

Context		Parameter
		SSL certificate file
		SSL key file
		SSL key password
网络发现		
	SNMP	*SNMP community*
		Context name (SNMPv3)
		Security name (SNMPv3)
		Authentication passphrase (SNMPv3)
		Privacy passphrase (SNMPv3)
全局脚本		
	Webhook	*JavaScript script*
		JavaScript script parameter value
	Telnet	*Username*
		Password
	SSH	*Username*
		Password
	Script	*Script*
媒体类型		
	Script	*Script parameters*
	Webhook	*Parameters*
IPMI 管理		
	主机	*Username*
		Password

低级别发现宏

概述

在低级发现（LLD）功能中，会使用一种宏：

```
{#MACRO}
```

它是在 LLD 规则中使用的宏，返回文件系统名称、网络接口、SNMP OID 等的实际值。

这些宏可用于创建实体原型（监控项、触发器、图形、LLD 规则、主机和主机组原型）。在发现实际的文件系统、网络接口、虚拟机等时，这些宏会被替换为实际值，并作为创建实际实体的基础。

有些低级发现宏是 Zabbix 中 LLD 功能 “预置” 的——{#FSNAME}、{#FSTYPE}、{#IFNAME}、{#SNMPINDEX}、{#SNMPVALUE}。但是，在创建自定义低级发现规则时，并不强制要求遵循这些名称。此时，您可以使用任何其他 LLD 宏名称，并引用该名称。

支持的数据类型

在定义自定义发现规则时，JSON 对象中为 LLD 宏返回的属性值必须是以下原始类型之一：

- string
- number
- boolean

不支持数组、对象和 null 值。
任何引用此类值的 LLD 宏都将保持未展开状态，并会在监控项预处理和创建期间按字面显示（例如 '{#MY_MACRO}'）。

支持的位置

LLD 宏可用于：

- 低级别发现规则过滤器中
- 嵌套的低级别发现规则中，在
  - JSONPath 预处理参数中
  - 用于自定义 LLD 宏的 JSONPath 字段中
- 用于监控项原型和发现原型，在以下位置中
  - 名称
  - 键参数
  - 单位
  - 更新间隔[1]
  - 超时[1]
  - 历史数据存储期[1]

  - 趋势数据存储期[1]
  - 监控项值预处理步骤
  - SNMP OID
  - IPMI 传感器字段
  - calculated/aggregate 监控项表达式中，在：
    * 表达式常量和函数参数中
    * 监控项键参数中
  - aggregate 监控项过滤条件（主机组名称和标签名称）
  - SSH 脚本和 Telnet 脚本
  - 数据库监控 SQL 查询
  - JMX 监控项端点字段
  - 描述
  - HTTP agent URL 字段
  - HTTP agent HTTP 查询字段字段
  - HTTP agent 请求体字段
  - HTTP agent 所需状态码字段
  - HTTP agent 标头字段键和值
  - HTTP agent HTTP 认证用户名字段
  - HTTP agent HTTP 认证密码字段
  - HTTP agent HTTP proxy 字段
  - HTTP agent HTTP SSL 证书文件字段
  - HTTP agent HTTP SSL 密钥文件字段
  - HTTP agent HTTP SSL 密钥密码字段
  - 标签
- 用于触发器原型，在以下位置中
  - 名称
  - 运行数据
  - 表达式（仅限常量和函数参数中）
  - URL
  - 描述
  - 标签
- 用于图形原型，在以下位置中
  - 名称
- 用于主机原型，在以下位置中
  - 名称
  - 可见名称
  - 自定义接口字段：IP、DNS、端口、SNMP v1/v2 community、SNMP v3 上下文名称、SNMP v3 安全名称、SNMP v3 认证口令、SNMP v3 隐私口令
  - 主机组原型名称
  - 主机标签值
  - 主机宏值
  - (请参见完整列表)

在所有这些位置中，除了低级别发现规则过滤器外，LLD 宏还可在静态用户宏上下文中使用。

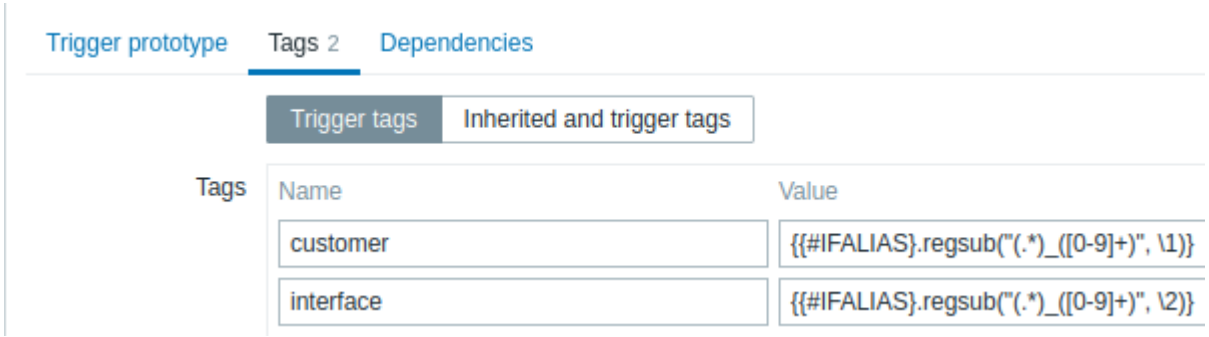
使用宏函数

低级别发现宏支持宏函数（低级别发现规则的过滤器中除外），允许使用正则表达式提取宏值的特定部分。

例如，您可能希望从以下 LLD 宏中提取客户名称和接口编号，以便用于事件标记：

```
{#IFALIAS}=customername_1
```

为此，可以在触发器原型的事件标签值字段中，将 `regsub` 宏函数与该宏结合使用：



请注意，未加引号的监控项key 参数中不允许使用逗号，因此包含宏函数的参数必须加引号。参数中的双引号应使用反斜杠（\）字符进行转义。示例：

```
net.if.in["{{#IFALIAS}.regsub(\"(.*)_([0-9]+)\", \1)}",bytes]
```

有关宏函数语法的更多信息，请参阅：宏函数

从 Zabbix 4.0 开始，低级别发现宏支持宏函数。

不带键参数的监控项原型

当未使用监控项键参数时，请将 LLD 宏放在监控项键参数方括号 [...] 内，这样它会被视为一个参数，并在发现期间被替换：

```
v_[{#MACRO}]
```

注释

[1] 在标有 [1] 的字段中，单个宏必须填充整个字段。不支持在一个字段中使用多个宏，也不支持将宏与文本混合使用。

表达式宏

概述

表达式宏可让你在字段中执行计算。

其值的计算方式是先解析任何内部宏，然后再对得到的表达式求值。

语法：

```
{?EXPRESSION}
```

EXPRESSION 使用与触发器表达式 相同的语法，并支持相同的函数。

示例：

```
{?trendavg(/host/item1,1M:now/M)/trendavg(/host/item1,1M:now/M-1y)*100}
```

对于支持表达式宏的位置，请在支持的宏 表中搜索"{?EXPRESSION}"。

使用说明：

- 使用{FUNCTION.*} 宏引用触发器表达式/恢复表达式中的函数值。
- 使用{HOST.HOST<1-9>} 和{ITEM.KEY<1-9>} 宏引用主机和监控项。
- 在模板中，使用{HOST.HOST<1-9>} 宏，或者对第一个主机直接省略主机名，例如 {?avg(//item1,1h)}，而不要使用模板名称，因为在模板链接 期间，模板名称不会替换为主机名称。

```
{?{FUNCTION.VALUE2} - {FUNCTION.VALUE3}}
{?max(/{HOST.HOST}/{ITEM.KEY},3h)}
```

另请参见触发器表达式示例，其中展示了如何在事件名称中使用表达式宏。

用户和用户组

概述

Zabbix 中的所有用户都通过基于 Web 的前端访问 Zabbix 应用程序。每个用户都会分配一个唯一的登录名和密码。

所有用户密码都会被加密并存储在 Zabbix 数据库中。除非用户也已按 UNIX 方式进行了相应设置，否则不能使用其用户 ID 和密码直接登录 UNIX 服务器。Web 服务器与用户浏览器之间的通信可以使用 SSL 进行保护。

借助灵活的用户权限方案，您可以限制并区分以下权限：

- 访问管理 Zabbix 前端功能。
- 在前端执行某些操作。
- 访问主机组中的受监控主机。
- 使用特定的 API 方法。

配置用户

概述

初始 Zabbix 安装包含两个预定义用户：

- *Admin* - 具有完整权限的 Zabbix 超级用户。
- *guest* - 特殊的 Zabbix 用户。

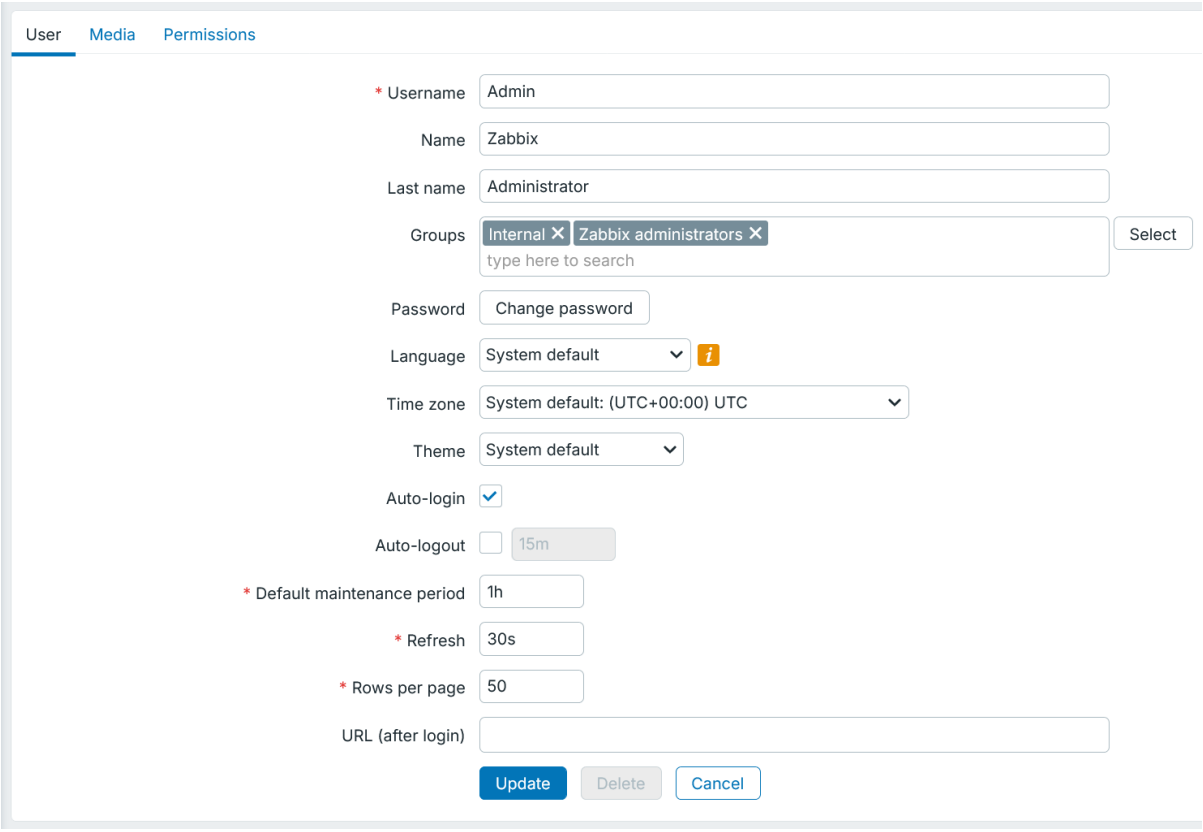
默认情况下，'guest' 用户处于禁用状态。
要启用该用户，请从用户组中移除预定义的 *Disabled* 组。
您可以使用此用户登录并访问 Zabbix 中的监控页面。
请注意，默认情况下，'guest' 对 Zabbix 对象没有任何权限。

要配置用户：

1. 转到 用户 > 用户。
2. 单击 创建用户（或单击用户名称以编辑现有用户）。
3. 在表单中编辑用户属性。

常规属性

用户选项卡包含常规用户属性：



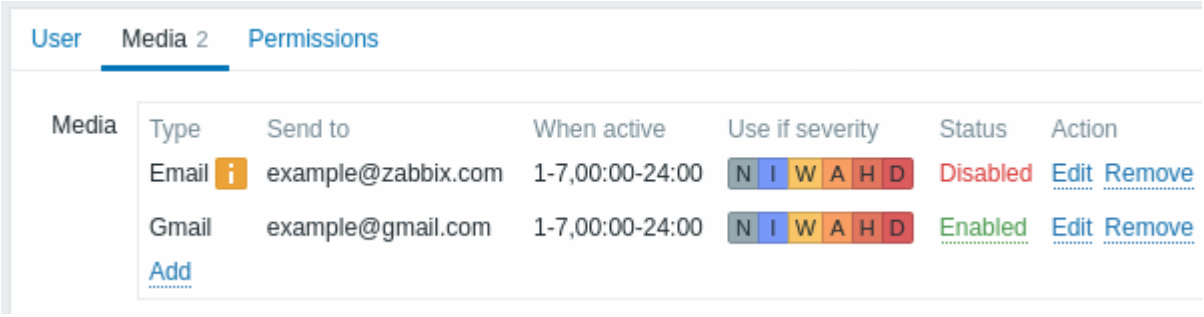
所有必填字段均以红色星号标记。

参数	描述
用户名	唯一的用户名，用作登录名。
名字	用户的名字（可选）。 如果不为空，则会显示在确认信息和通知接收者信息中。
姓氏	用户的姓氏（可选）。 如果不为空，则会显示在确认信息和通知接收者信息中。
用户组	选择用户所属的用户组。此字段支持自动完成，因此开始输入用户组名称时，将显示包含匹配用户组的下拉列表。向下滚动以进行选择。也可以点击选择来添加用户组。点击 "x" 可移除已选择的用户组。 用户组决定用户可以访问哪些主机组和主机。

参数	描述
密码	用于输入用户密码的两个字段；如果用户已存在，则显示更改密码按钮。 点击更改密码按钮后，会打开两个用于输入新密码的字段。 对于具有超级管理员角色且正在更改自身密码的用户，点击更改密码按钮后，还会打开一个用于输入当前（旧）密码的字段。 密码成功更改后，密码被更改的用户将从所有活动会话中退出。 请注意，仅可更改使用 Zabbix 内部身份验证的用户的密码。
语言	Zabbix 前端的语言。 要使翻译正常工作，需要 php gettext 扩展。
时区	选择要在用户级别覆盖全局时区的时区，或选择系统默认以使用全局时区设置。
主题	定义前端的外观： 系统默认 - 使用默认系统设置 蓝色 - 标准蓝色主题 蓝色（经典） - 使用旧版字体的标准蓝色主题 深色 - 备用深色主题 深蓝色 - 深蓝色主题 深色（经典） - 使用旧版字体的备用深色主题 高对比度浅色 - 高对比度浅色主题 高对比度深色 - 高对比度深色主题
自动登录	选中此复选框后，Zabbix 将记住该用户，并在 30 天内自动登录。 使用记住 **30** 天登录时： - 自动注销将被重置（会话持续 30 天）。 - 启用自动登录，以实现无缝重新身份验证。 不使用记住 *30* 天登录时： - 启用自动登录，以实现无缝重新身份验证。 - 自动注销仍由标准超时设置控制。 此功能使用浏览器 Cookie。
自动注销	选中此复选框后，用户将在设定的秒数后自动注销（最少 90 秒，最多 1 天）。 请注意，如果启用了记住 **30** 天，则此设置会被覆盖，因为会话会延长至整个期限。 支持时间后缀，例如 90s、5m、2h、1d。 请注意，在以下情况下，此选项不会生效： * 启用了 “如果 Zabbix 服务器停机则显示警告” 全局配置选项，并且 Zabbix 前端保持打开。 * 监控菜单页面执行后台信息刷新时。 * 使用选中的记住我 *30* 天选项登录时。
默认维护周期	设置临时维护的默认维护周期。
刷新	设置图形、纯文本数据等的刷新频率。可设置为 0 以禁用。 支持时间后缀，例如 90s、5m、1h。
每页行数	可以确定列表中每页显示的行数。
URL（登录后）	可以让 Zabbix 在成功登录后将用户转到特定 URL，例如问题页面。

用户媒体

媒体选项卡列出了为用户定义的所有媒体。媒体用于发送通知。



单击添加，为用户分配媒体。

如果媒体类型已被禁用：

- 名称后会显示黄色信息图标。
- 已禁用会显示在 “状态” 列中。

Note:
可以根据用户的用户角色授予或撤销用户更改自己的媒体详细信息的权限（请参阅创建和编辑自己的媒体选项）。也可以根据超级管理员用户的用户角色授予或撤销其更改其他用户媒体详细信息的权限（请参阅创建和编辑用户媒体选项）。

有关配置用户媒体的详细信息，请参阅媒体类型部分。

权限

权限选项卡包含以下元素的信息：

- 用户角色（任何新创建的用户都必须设置），且只能由 *Super admin* 类型用户更改。

Attention:
创建用户时必须指定用户角色（通过 Zabbix User API 创建的情况除外）。之前创建且未分配角色的用户，仍可在不分配角色的情况下进行编辑。但是，一旦分配了角色，就只能更改，不能移除。

 请注意，没有角色的用户只能通过LDAP 或SAML 身份验证登录 Zabbix，前提是其 LDAP/SAML 信息与 Zabbix 中配置的用户组映射相匹配。

- 用户类型（*User*、*Admin*、*Super admin*），在用户角色配置中定义。
- 用户有权访问的主机组和模板组。
  - 默认情况下，*User* 和 *Admin* 类型用户无权访问任何组、模板和主机。要授予此类访问权限，必须将用户加入已配置了相关实体权限的用户组。
- 对 Zabbix 前端各部分和元素、模块以及 API 方法的访问权限。
  - 允许访问的元素以绿色显示，而拒绝访问的元素以浅灰色显示。
- 执行特定操作的权限。
  - 用户被允许执行的操作以绿色显示，而被拒绝的操作以浅灰色显示。

详情请参见权限页面。

权限

概述

在 Zabbix 中，权限取决于用户类型、定制的用户角色以及基于用户组指定的主机访问权限。

用户类型

Zabbix 中的权限主要取决于用户类型：

- *User* - 对菜单部分的访问权限有限（见下文），默认情况下无权访问任何资源。必须明确分配对主机组或模板组的权限。
- *Admin* - 对菜单部分的访问权限不完整（见下文）。默认情况下，用户无权访问任何主机组。必须明确授予对主机组或模板组的权限。
- *Super admin* - 可访问所有菜单部分。用户对所有主机组和模板组具有读写权限。不能通过拒绝访问特定组来撤销权限。

菜单访问权限

Note:
限制对某些 UI 元素的访问仅会阻止打开相应页面，不会移除在界面的其他部分访问底层数据的能力。

下表说明了不同用户类型对 Zabbix 菜单部分的访问权限：

菜单部分		User	Admin	Super admin
仪表板		+	+	+
监控		+	+	+
	问题	+	+	+
	主机	+	+	+
	最新数据	+	+	+
	拓扑图	+	+	+
	自动发现		+	+
APM		+	+	+
	追踪	+	+	+
	指标	+	+	+
	日志	+	+	+
服务		+	+	+
	服务	+	+	+
	SLA		+	+
	SLA 报告	+	+	+

菜单部分		User	Admin	Super admin
资产清单		+	+	+
	概览	+	+	+
	主机	+	+	+
报告		+	+	+
	系统信息		+	+
	计划报告		+	+
	可用性报告	+	+	+
	前 *100* 个触发器	+	+	+
	审计日志			+
	操作日志			+
	通知		+	+
数据采集			+	+
	模板组		+	+
	主机组		+	+
	模板		+	+
	主机		+	+
	维护		+	+
	事件关联			+
	自动发现		+	+
告警			+	+
	触发器操作		+	+
	服务操作		+	+
	自动发现操作		+	+
	自动注册操作		+	+
	内部操作		+	+
	媒介类型			+
	脚本			+
用户				+
	用户组			+
	用户角色			+
	用户			+
	API 令牌			+
	身份验证			+
	设备			+
管理				+
	数据源			+
	常规			+
	审计日志			+
	内务处理			+
	proxy 组			+
	proxy			+
	宏			+
	队列			+

用户角色

用户角色允许根据用户类型定义的权限进行自定义调整。
虽然无法添加权限（因为这会超出用户类型所拥有的权限），但可以撤销某些权限。

此外，用户角色不仅决定对菜单部分的访问权限，还决定对前端中的服务、模块、API 方法和各种操作的访问权限。

用户角色由超级管理员用户在用户 > 用户角色部分进行配置。

超级管理员用户可在用户配置表单的权限选项卡中为用户分配用户角色。

Users

User | Media | Permissions

* Role: Admin role ✕ | Select

User type: Admin

Permissions

User group permissions

Group	Type	Permissions
All groups	Hosts	None
All groups	Templates	None

Proxies: None

Proxy groups: None

Permissions can be assigned for user groups only.

Access to UI elements

Dashboards: Dashboards

Monitoring: Problems | Hosts | Latest data | Maps | Discovery

APM: Traces | Metrics | Logs

Services: Services | SLA | SLA report

Inventory: Overview | Hosts

Reports: System information | Scheduled reports | Availability report | Top 100 triggers | Notifications

Data collection: Template groups | Host groups | Templates | Hosts | Maintenance | Discovery

Alerts: Trigger actions | Service actions | Discovery actions | Autoregistration actions | Internal actions

User profile settings

Redirect URL after login

Access to services

Read-write access to services: None

Read-only access to services: All

Access to modules

Action log | Clock | Discovery status | Favorite graphs | Favorite maps | Gauge | Geomap | Graph | Graph (classic) | Graph prototype | Honeycomb | Host availability | Host card | Host navigator | Item card | Item history | Item navigator | Item value | Map | Map navigation tree | Pie chart | Problem hosts | Problems | Problems by severity | Scatter plot | SLA report | System information | Top hosts | Top items | Top triggers | Trigger overview | URL | Web monitoring

Access to API

Disabled

Devices

Enabled | Manage own devices

Access to actions

Create and edit dashboards | Create and edit maps | Create and edit maintenance | Add problem comments | Change severity | Acknowledge problems | Suppress problems | Close problems | Execute scripts | Manage API tokens | Manage scheduled reports | Manage SLA | Invoke "Execute now" on read-only hosts | Change problem ranking | Create and edit own media | Select "Server" for monitoring and discovery

Add | Cancel

访问主机的权限

在 Zabbix 中，对任何主机和模板数据的访问权限只能在主机/模板组级别授予给用户组。

这意味着，不能直接向单个用户授予对某个主机（或主机组）的访问权限。只能通过将用户加入某个已被授予访问包含该主机的主机组权限的用户组，来授予其对该主机的访问权限。

同样，只有当用户属于某个已被授予访问包含该模板的模板组权限的用户组时，才能授予其对该模板的访问权限。

用户组

概述

用户组可用于出于组织管理目的对用户进行分组，也可用于为数据分配权限。主机组和模板组的数据查看及配置权限分配给用户组，而不是单个用户。

通常，将一组用户可访问的信息与另一组用户可访问的信息区分开来是有意义的。这可以通过对用户进行分组，然后为主机组和模板组分配不同的权限来实现。

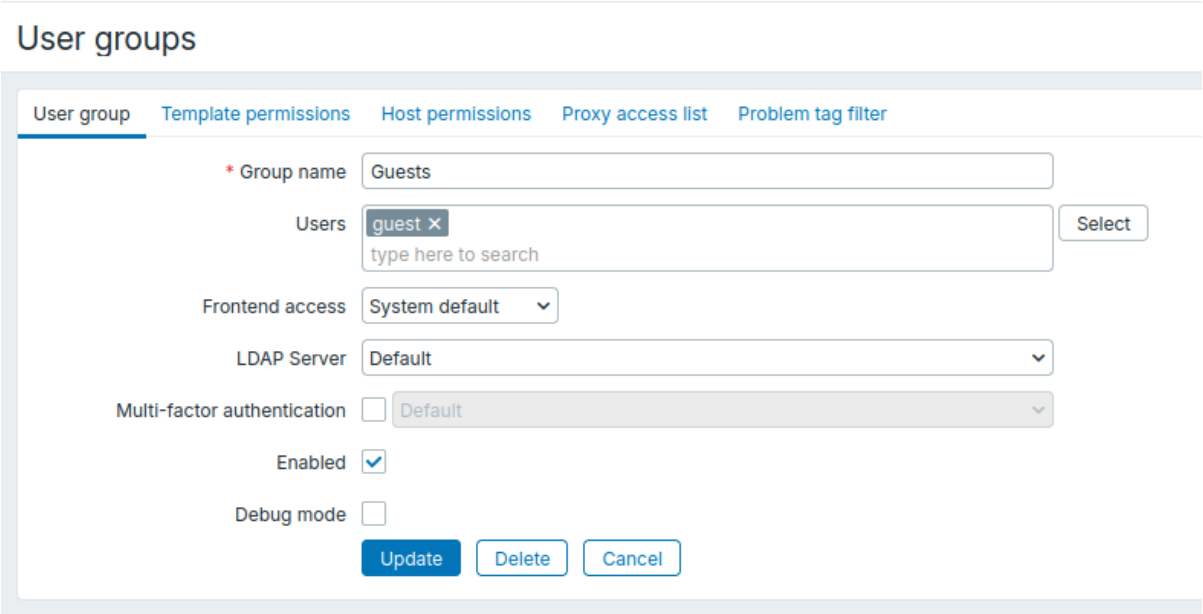
一个用户可以属于任意数量的用户组。

配置

要配置用户组：

1. 转到 用户 > 用户组。
2. 单击 创建用户组（或单击组名称以编辑现有组）。
3. 在表单中编辑组属性。

用户组选项卡包含常规组属性：

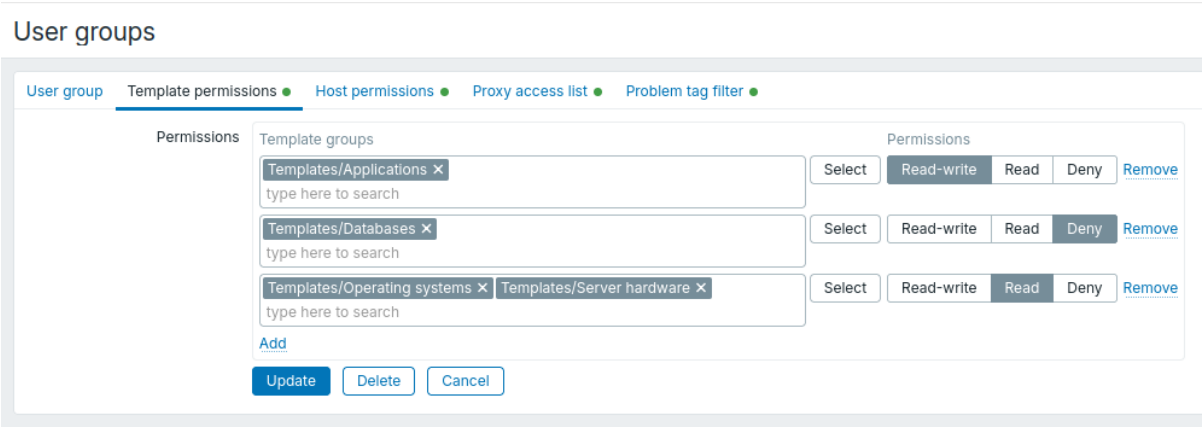


所有必填输入字段均以红色星号标记。

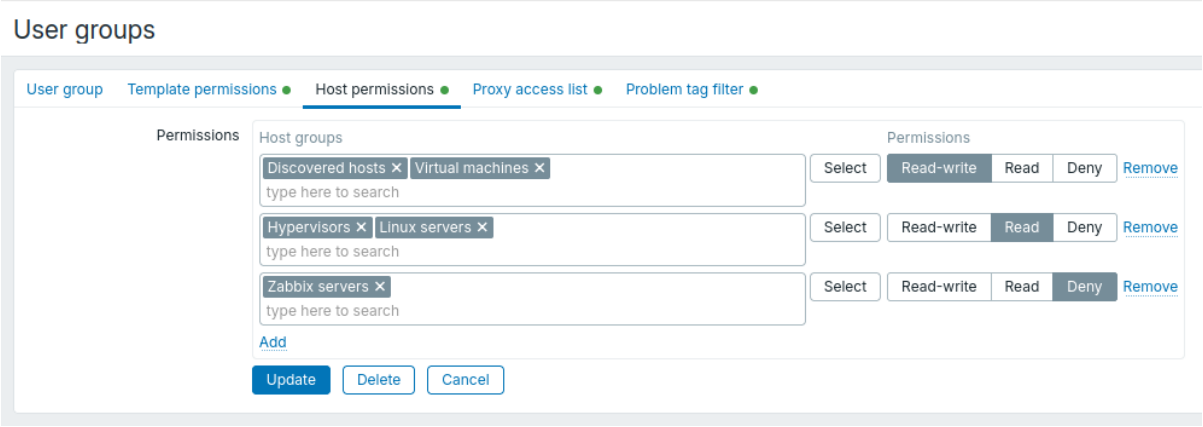
参数	描述
组名称	唯一的组名称。
用户	要将用户添加到组中，请开始输入现有用户的名称。当显示包含匹配用户名的下拉列表时，向下滚动并选择用户。 或者，您可以单击 选择按钮，在弹出窗口中选择用户。
前端访问	组用户的身份验证方式。 系统默认 - 使用默认身份验证方法（在全局设置中设置） 内部 - 使用 Zabbix 内部身份验证（即使全局使用 LDAP 身份验证）。 如果 HTTP 身份验证是全局默认设置，则忽略此选项。 **LDAP** - 使用 LDAP 身份验证（即使全局使用内部身份验证）。 如果 HTTP 身份验证是全局默认设置，则忽略此选项。 禁用 - 禁止此组访问 Zabbix 前端
LDAP 服务器	选择用于对用户进行身份验证的LDAP 服务器。 仅当 前端访问设置为 LDAP 或系统默认时，此字段才可用。

参数	描述
多重身份验证	选择用于对用户进行身份验证的多重身份验证方法： 默认 - 使用 MFA 配置中设置为默认的方法；如果启用了 MFA，则新建用户组时默认选择此选项； < 方法名称 > - 使用选定的方法（例如 “Zabbix TOTP”）； 禁用 - 此组禁用 MFA；如果禁用了 MFA，则新建用户组时默认选择此选项。 请注意，如果用户属于多个启用了 MFA 的用户组（或至少有一个组启用了 MFA），则适用以下身份验证规则：如果任意组使用 “默认”MFA 方法，则使用该方法对用户进行身份验证；否则，使用按字母顺序排列的第一个方法进行身份验证。
启用	用户组及组成员的状态。 选中 - 用户组和用户已启用 未选中 - 用户组和用户已禁用
调试模式	选中此复选框，为用户启用调试模式。

模板权限选项卡允许您指定用户组对模板组的访问权限，以及通过模板组对模板数据的访问权限：



主机权限选项卡允许您指定用户组对主机组的访问权限，以及通过主机组对主机数据的访问权限：



单击 Add 选择模板/主机组（包括父组或嵌套组），并为其分配权限。开始输入组名称（将显示包含匹配组的下拉列表），或单击 选择打开列出所有组的弹出窗口。

然后使用选项按钮为选定的组分配权限。可用权限如下：

- 读写 - 对组具有读写访问权限
- 读取 - 对组具有只读访问权限
- 拒绝 - 拒绝访问组

如果同一模板/主机组在多个行中添加，并设置了不同的权限，则应用最严格的权限。

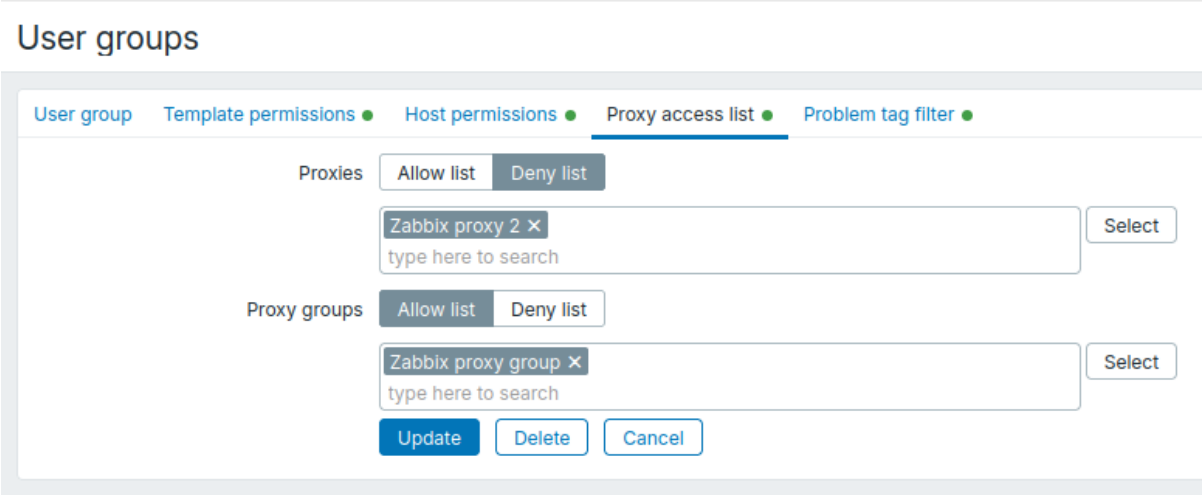
请注意，超级管理员用户可以强制嵌套组使用与父组相同级别的权限；可以在主机/模板组配置表单中进行设置。

模板权限和主机权限选项卡支持相同的参数集。

组的当前权限显示在 权限区块中，并且可以修改或删除这些权限。

Note:
如果用户组对主机具有读写权限，而对链接到该主机的模板具有拒绝权限或没有权限，则该组用户将无法编辑主机上的模板化监控项，并且模板名称将显示为无法访问的模板。

Proxy 访问列表选项卡允许您指定用户组对 proxy 的访问权限，可以单独指定，也可以通过 proxy 组指定：



使用 *Proxy* 组字段将权限应用于选定的组。属于某个组的 proxy 只能通过该组允许或拒绝访问。使用 *Proxy* 字段添加不属于任何 proxy 组的单个 proxy。

Proxy 和 *Proxy* 组支持相同的参数集。

单击 *Proxy* 或 *Proxy* 组字段旁的 选择按钮，分别选择 proxy 或 proxy 组。或者，在相应字段中开始输入 proxy 或 proxy 组的名称。此时会显示包含匹配 proxy 或 proxy 组的下拉列表。

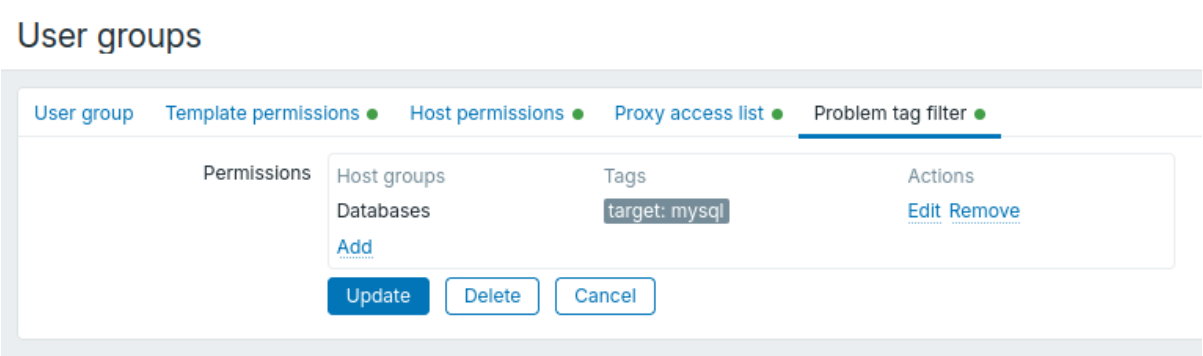
选择 允许列表，仅允许用户组访问选定的 proxy 和 proxy 组。
 如果 允许列表中的 *Proxy* 选择为空，则用户组无法访问任何不属于 proxy 组的 proxy。如果 允许列表中的 *Proxy* 组选择为空，则用户组无法访问任何 proxy 组。

选择 拒绝列表，拒绝用户组访问选定的 proxy 和 proxy 组。允许访问所有其他 proxy 和 proxy 组。
 如果 拒绝列表中的 *Proxy* 选择为空，则用户组可以完全访问所有不属于 proxy 组的 proxy。如果 拒绝列表中的 *Proxy* 组选择为空，则用户组可以完全访问所有 proxy 组。

属于多个用户组的用户的权限优先级规则：

- 拒绝列表优先于 允许列表。如果任何组拒绝访问某个 proxy/proxy 组，则用户无法访问该 proxy/proxy 组。
- 允许列表权限会累积。如果多个组允许访问，则用户可以访问这些组所允许的所有 proxy。

问题标签过滤器选项卡允许为用户组设置基于标签的权限，以便按标签名称和值筛选可查看的问题：

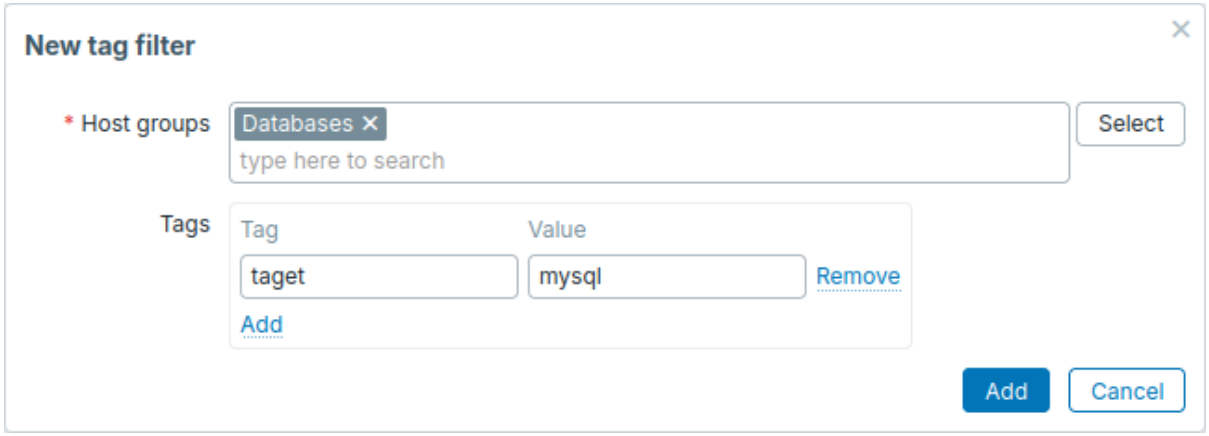


单击 Add 选择主机组。要选择应用标签过滤器的主机组，请单击 选择获取现有主机组的完整列表，或开始输入主机组名称以获取包含匹配组的下拉列表。这里只会显示主机组，因为问题标签过滤器无法应用于模板组。

可以添加不带值的标签名称，但不能添加不带名称的值。权限区块中只显示前三个标签（如果有值，则同时显示值）；如果标签超过三个，可以通过单击或将鼠标悬停在 ••• 图标上查看其他标签。

标签过滤器允许将对主机组的访问权限与查看问题的权限分开。

例如，如果数据库管理员只需要查看 “MySQL” 数据库问题，则首先需要为数据库管理员创建用户组，然后指定 “target” 标签名称和 “mysql” 值。



如果指定了 “target” 标签名称，而值字段留空，则用户组将看到选定主机组中所有标签名称为 “target” 的问题。

请确保正确指定标签名称和标签值，否则用户组将看不到任何问题。

下面查看一个用户属于多个选定用户组时的示例。在这种情况下，标签筛选将使用 OR 条件。

用户组 **A**			用户组 **B**			同时属于两个组的用户可见结果
标签过滤器						
主机组	标签名称	标签值	主机组	标签名称	标签值	
数据库	target	mysql	数据库	target	oracle	可见 target:mysql 或 target:oracle 问题
未在问题标签过滤器中配置			数据库	target	oracle	可见 target:oracle 问题

来自多个用户组的访问

一个用户可以属于任意数量的用户组。这些组可能对主机或模板具有不同的访问权限。

因此，了解低权限用户最终能够访问哪些实体非常重要。在下面的示例中，考虑对于同时属于用户组 A 和 B 的用户，在各种情况下，对主机 **X**（位于主机组 1 中）的访问将如何受到影响。

- 如果组 A 对主机组 1 只有 读权限，而组 B 对主机组 1 具有 读写权限，则该用户将获得对 “X” 的 读写访问权限。

Attention:
“读写” 权限优先于 “读” 权限。

- 在与上述相同的场景中，如果 “X” 同时也位于主机组 2 中，而主机组 2 对组 A 或 B 是拒绝的，则对 “X” 的访问将不可用，尽管对主机组 1 具有 读写访问权限。
- 如果组 A 未定义任何权限，而组 B 对主机组 1 具有 读写访问权限，则该用户将获得对 “X” 的 读写访问权限。
- 如果组 A 对主机组 1 具有 拒绝访问权限，而组 B 对主机组 1 具有 读写访问权限，则该用户对 “X” 的访问将被拒绝。

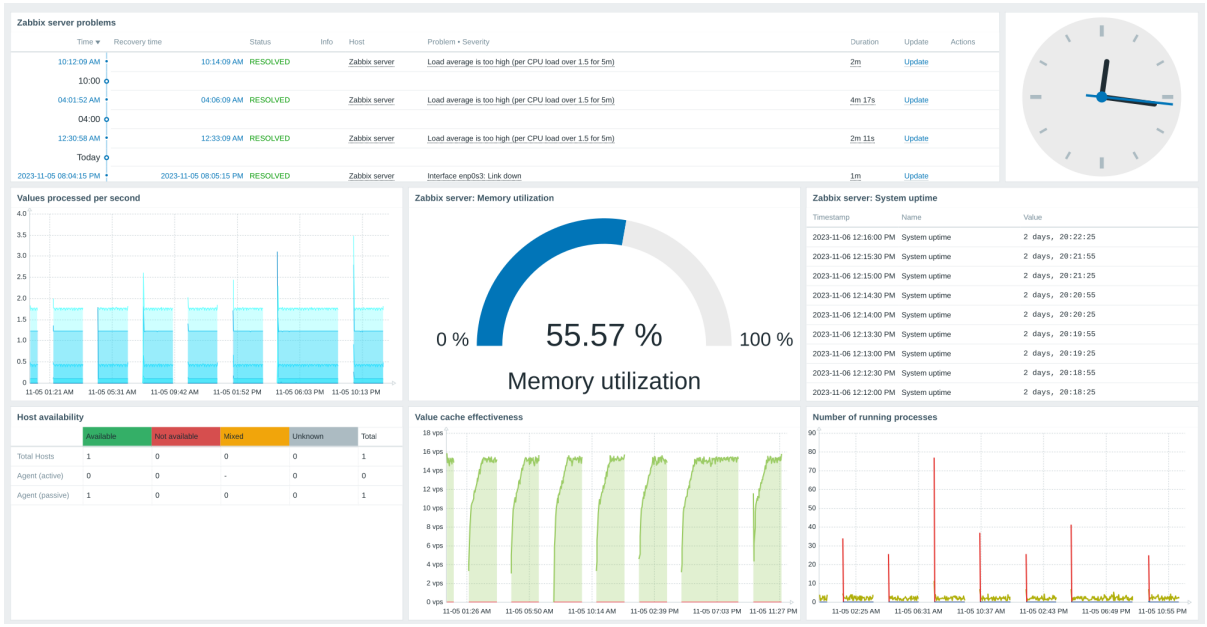
其他细节

- 对某主机具有 读写访问权限的 Admin 级别用户，如果对模板所属的模板组没有访问权限，则无法链接/取消链接模板。如果对模板组具有 读取访问权限，则他将能够将模板链接到/从主机取消链接模板；但是，他不会在模板列表中看到任何模板，也无法在其他位置对模板进行操作。
- 对某主机具有 读取访问权限的 Admin 级别用户，将不会在配置部分的主机列表中看到该主机；但是，该主机的触发器可在 IT 服务配置中访问。
- 任何非 Super Admin 用户（包括 “guest”）都可以查看网络拓扑图，只要该拓扑图为空或仅包含图像。当向拓扑图中添加主机、主机组或触发器时，将遵循权限设置。
- 如果对相关主机的访问被显式 “拒绝”，Zabbix 服务器将不会向定义为动作操作接收人的用户发送通知。

定时报表

概述

借助计划报表功能，您可以设置将指定仪表板的 PDF 版本按固定时间间隔发送给指定的收件人。



前提条件：

- 必须正确安装和配置 Zabbix web service，才能启用计划报表生成。有关说明，请参阅设置计划报表。
- 用户必须具有类型为管理员或超级管理员的用户角色，并拥有以下权限：
  - 访问 *UI* 元素区域中的计划报表权限（用于查看报表设置）
  - 访问操作区域中的管理计划报表权限（用于创建/编辑报表）

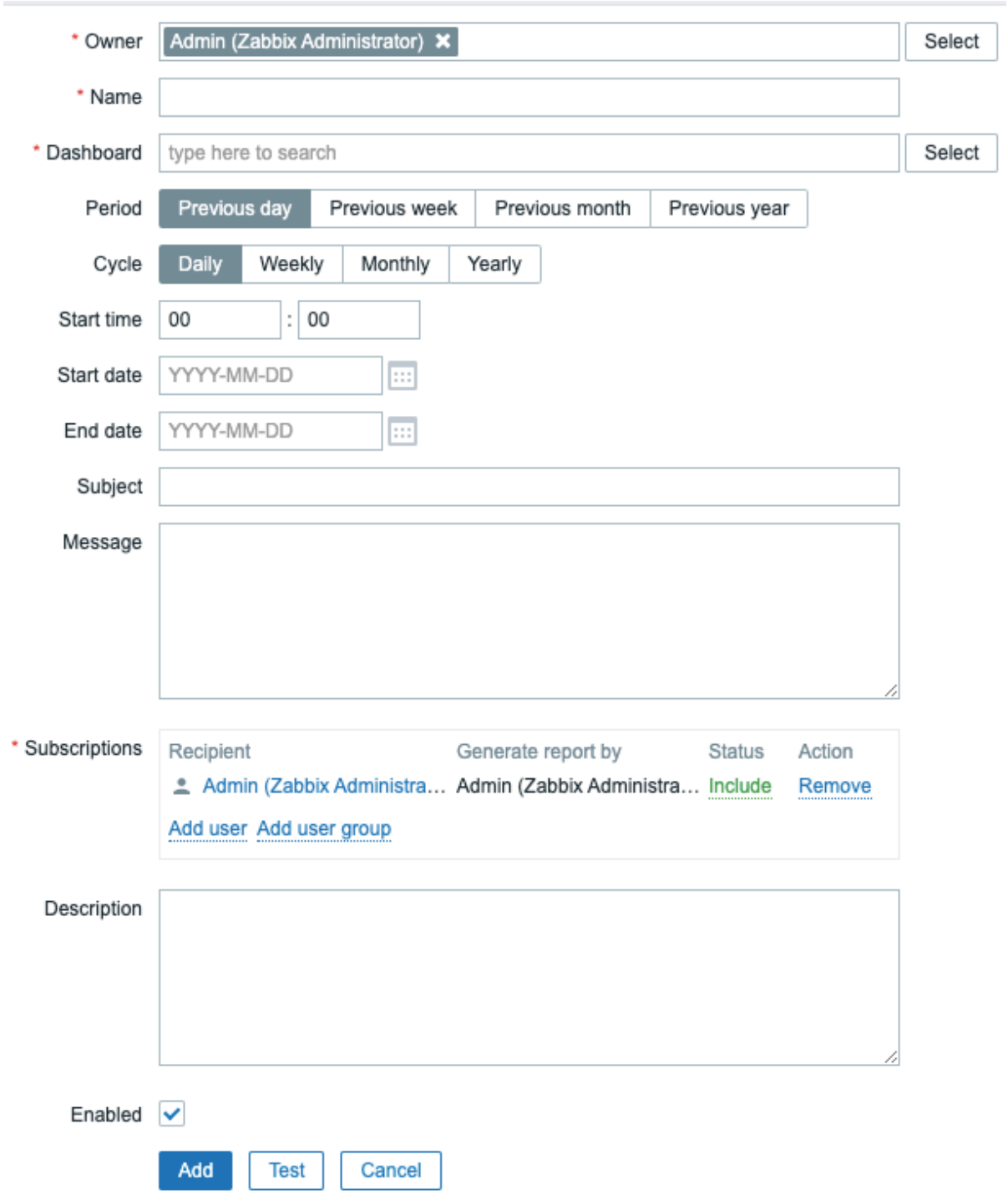
要在 Zabbix 前端中创建计划报表，请执行以下操作：

1. 转到报表 > 计划报表。
2. 单击屏幕右上角的创建报表。
3. 在表单中输入报表参数。

或者，您也可以打开现有报表，单击克隆按钮，然后使用其他名称保存该报表。

配置

计划报表选项卡包含报表的一般属性。



所有必填输入字段均标有红色星号。

参数	描述
所有者	创建报表的用户。超级管理员级别的用户可以更改所有者。对于管理员级别的用户，此字段为只读。
名称	报表名称；必须唯一。
仪表板	报表所基于的仪表板；一次只能选择一个仪表板。要选择仪表板，请开始输入其名称，将显示匹配的仪表板列表；向下滚动以进行选择。或者，可以单击字段旁边的选择，然后从显示的列表中选择仪表板。
周期	准备报表的时间段。选择前一天、前一周、前一个月或前一年。
循环	报表生成频率。报表可以按天、按周、按月或按年发送。“每周”模式允许您选择发送报表的星期几。
开始时间	准备报表的时间，格式为 hh:mm。请注意，将使用 Zabbix 服务器的时区。
重复日期	发送报表的星期几。仅当循环设置为“每周”时，此字段才可用。
开始日期	开始定期生成报表的日期。
结束日期	停止定期生成报表的日期。
主题	报表电子邮件的主题。支持的宏：{TIME}、{TIMESTAMP}。
消息	报表电子邮件的正文。支持的宏：{TIME}、{TIMESTAMP}。

参数	描述
订阅	报表收件人列表。默认情况下，仅包括报表所有者。任何已配置电子邮件媒介的 Zabbix 用户都可以指定为报表收件人。 单击添加用户或添加用户组以添加更多收件人。 单击用户名以编辑设置： 生成报表依据 - 选择是否应根据当前用户或收件人的仪表板权限生成报表数据。 状态 - 选择 “包括” 以向用户发送报表，或选择 “排除” 以阻止向该用户发送报表。至少有一个用户必须具有 “包括” 状态。可以使用 “排除” 状态，将特定用户从已包含的用户组中排除。 请注意，权限不足的用户（即，角色基于管理员用户类型，且不属于与收件人或报表所有者相同的用户组的用户）将在收件人和生成报表依据字段中看到 “无法访问的用户” 或 “无法访问的用户组”，而不是实际名称；状态和操作字段将显示为只读。
启用	报表状态。清除该复选框将禁用报表。
描述	报表的可选描述。此描述仅供内部使用，不会发送给报表收件人。

表单按钮

表单底部的按钮可用于执行多项操作。

Add	添加报告。此按钮仅适用于新报告。
Update	更新报告的属性。
Clone	根据当前报告的属性创建另一个报告。
Test	通过向当前用户发送报告，测试报告配置是否正确。
Delete	删除报告。
Cancel	取消编辑报告属性。

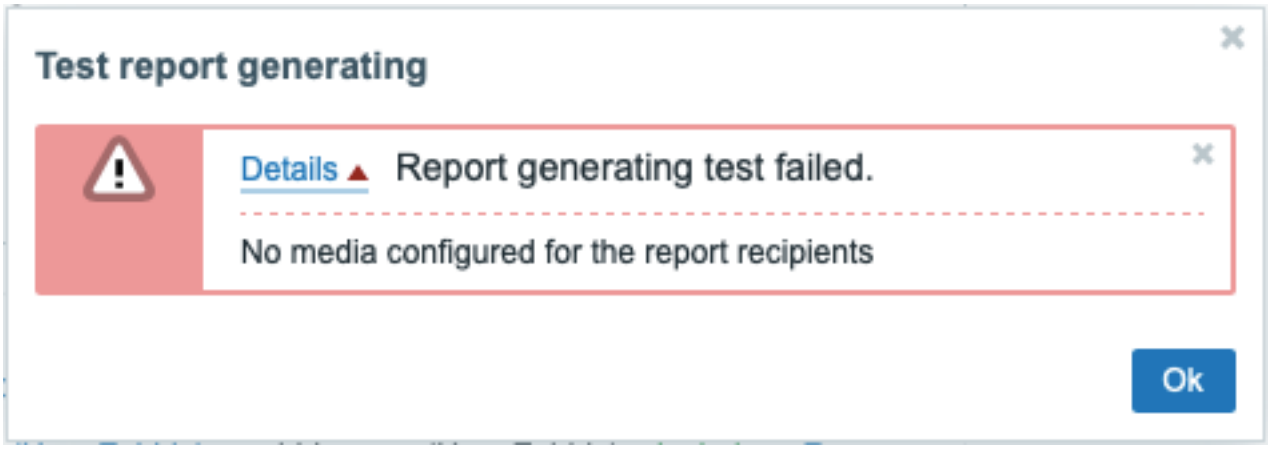
测试

要测试报告，请单击报告配置表单底部的 测试按钮。

Note:
如果报告配置表单是从仪表板操作菜单打开的，则 测试按钮不可用。

如果配置正确，测试报告会立即发送给当前用户。对于测试报告，将忽略订阅者和 生成报告的用户设置。

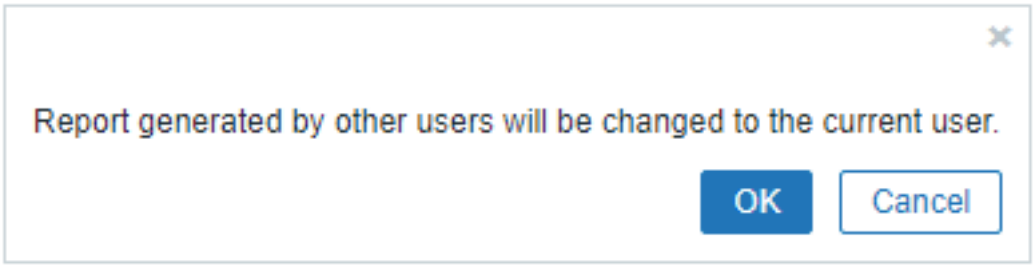
如果配置不正确，则会显示一条错误消息，说明可能的原因。



更新报告

要更新现有报告，请单击报告名称，进行所需的配置更改，然后单击 更新按钮。

如果现有报告已被其他用户更新，并且该用户更改了仪表板，那么在单击 更新按钮时，将显示一条警告消息：“由其他用户生成的报告将更改为当前用户”。



在此步骤中单击 确定将导致以下更改：

- 报告生成者设置将更新为显示最后编辑该报告的用户（除非 报告生成者设置为收件人）。
- 显示为 “无法访问的用户” 或 “无法访问的用户组” 的用户将从报告订阅者列表中删除。

单击 取消将关闭配置表单并取消报告更新。

克隆报表

要快速克隆现有报表，请点击现有报表配置表单底部的 *Clone* 按钮。
克隆由其他用户创建的报表时，当前用户将成为新报表的所有者。

报表设置将复制到新的报表配置表单中，并遵循用户权限：

- 如果克隆报表的用户对某个仪表板没有权限，*Dashboard* 字段将被清空。
- 如果克隆报表的用户对 *Subscriptions* 列表中的某些用户或用户组没有权限，则无法访问的收件人不会被克隆。
- *Generate report by* 设置将更新为显示当前用户（除非 *Generate report by* 设置为收件人）。

更改所需设置和报表名称，然后点击 *Add*。

数据导出

概述

Zabbix 支持两种实时数据导出方式：

- 导出到文件
- 流式导出到外部系统

可以导出以下实体数据：

- 触发器事件
- 监控项数值
- 趋势数据（仅限导出到文件）

导出到文件

概述

可以配置以换行分隔的 JSON 格式实时导出触发器事件、监控项值和趋势数据。

导出会写入文件，其中导出文件的每一行都是一个 JSON 对象。不会应用值映射。

如果发生错误（数据无法写入导出文件，或者导出文件无法重命名，或者在重命名后无法创建新的导出文件），则该数据项会被丢弃，并且永远不会写入导出文件。它只会写入 Zabbix 数据库。当写入问题解决后，向导出文件写入数据将会恢复。

导出文件仅为文件所有者创建读写权限。此外，文件所有者所属组也可读取该文件。其他所有权限均被拒绝。

请注意，如果主机/监控项在接收到数据后、但在服务器导出数据之前被移除，则主机/监控项可能没有元数据（主机组、主机名、监控项名称）。

配置

通过为导出文件指定目录，可以配置触发器事件、监控项值和趋势的实时导出 - 请参阅服务器配置中的ExportDir 参数。

还可以使用另外两个参数：

- `ExportFileSize` 可用于设置单个导出文件允许的最大大小。当进程需要写入文件时，会先检查文件大小。如果文件大小超过配置的限制，系统会在文件名后追加 `.old` 来重命名该文件，然后创建一个使用原始文件名的新文件。

Attention:
每个将写入数据的进程都会创建一个文件（即大约 4-30 个文件）。由于每个导出文件的默认大小为 1G，保留大型导出文件可能会迅速耗尽磁盘空间。

- ExportType 允许您指定要导出的实体类型（事件、历史数据、趋势）。

换行分隔 JSON 导出协议

本节介绍以换行分隔的 JSON 格式实现的导出协议的详细信息，该格式用于：

- 导出到文件
- 流式传输到外部系统

所有文件都使用 .ndjson 扩展名。导出文件中的每一行都是一个 JSON 对象。

触发器事件导出

以下信息会为问题事件导出：

Field			Type	Description
clock			number	自 Epoch 起到检测到问题时刻的秒数（整数部分）。
ns			number	要加到 clock 上的纳秒数，以获得精确的问题检测时间。
value			number	1（始终如此）。
eventid			number	问题事件 ID。
name			string	问题事件名称。
severity			number	问题事件严重性（0 - 未分类，1 - 信息，2 - 警告，3 - 一般，4 - 高，5 - 灾难）。
hosts			array	触发器表达式中涉及的主机列表；数组中至少应有一个元素。
	-		object	
		host	string	主机名称。
		name	string	可见主机名称。
groups			array	触发器表达式中涉及的所有主机所属的主机组列表；数组中至少应有一个元素。
	-		string	主机组名称。
tags			array	问题标签列表（可以为空）。
	-		object	
		tag	string	标签名称。
		value	string	标签值（可以为空）。

以下信息会为恢复事件导出：

Field	Type	Description
clock	number	自 Epoch 起到问题解决时刻的秒数（整数部分）。
ns	number	要加到 clock 上的纳秒数，以获得精确的问题解决时间。
value	number	0（始终如此）。
eventid	number	恢复事件 ID。
p_eventid	number	问题事件 ID。

触发器事件导出示例

问题：

```
{"clock":1519304285,"ns":123456789,"value":1,"name":"Either Zabbix agent is unreachable on Host B or polle
```

恢复：

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

问题（生成多个问题事件）：

```
{"clock":1519304286,"ns":123456789,"value":1,"eventid":43,"name":"Either Zabbix agent is unreachable on Ho

{"clock":1519304286,"ns":123456789,"value":1,"eventid":43,"name":"Either Zabbix agent is unreachable on Ho
```

恢复：

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

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

监控项值导出

以下信息会针对已采集的监控项值导出：

Field	Type	Description
host	object	监控项所属主机的主机名。
host	string	主机名。
name	string	可见主机名。
groups	array	监控项所属主机的主机组列表；数组中至少应有一个元素。
-	string	主机组名称。
item_tags	array	监控项标签列表（可以为空）。
-		object
tag	string	标签名称。
value	string	标签值（可以为空）。
itemid	number	监控项 ID。
name	string	可见监控项名称。
clock	number	采集值时距 Epoch 的秒数（整数部分）。
ns	number	要加到 `clock` 上以获得精确采集时间的纳秒数。
timestamp (*Log* only)	number	如果不可用，则为 0。
source (*Log* only)	string	如果不可用，则为空字符串。
severity (*Log* only)	number	如果不可用，则为 0。
eventid (*Log* only)	number	如果不可用，则为 0。
value	number (for numeric items) or string (for text items)	采集到的监控项值。
type	number	采集值类型： 0 - 数值浮点，1 - 字符，2 - 日志，3 - 无符号数值，4 - 文本，5 - 二进制，6 - JSON

监控项值导出示例

数值（无符号）：

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

数值（浮点）：

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

字符、文本值：

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

日志值：

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

趋势导出

以下信息会为计算出的趋势值导出：

Field		Type	Description
host		object	监控项主机的主机名。
	host	string	主机名。
	name	string	可见主机名。
groups		array	监控项主机的主机组列表；数组中至少应有一个元素。
	-	string	主机组名称。

Field			Type	Description
item_tags			array	监控项标签列表（可以为空）。
	-		object	
		tag	string	标签名称。
		value	string	标签值（可以为空）。
itemid			number	监控项 ID。
name			string	可见监控项名称。
clock			number	采集该值时距 Epoch 的秒数（整数部分）。
count			number	在给定小时内采集到的值数量。
min			number	给定小时内的最小监控项值。
avg			number	给定小时内的平均监控项值。
max			number	给定小时内的最大监控项值。
type			number	值类型： 0 - 数值浮点，3 - 数值无符号

趋势导出示例

数值（无符号）值：

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

数值（浮点）值：

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

流式传输到外部系统

概述

可以通过 HTTP 将监控项值和事件从 Zabbix 流式传输到外部系统（请参阅协议详细信息）。

可以使用标签过滤器来流式传输部分监控项值或事件。

两种 Zabbix 服务器进程类型负责数据流式传输：`connector manager` 和 `connector worker`。Zabbix 内部监控项 `zabbix[connector_queue]` 可用于监控连接器队列中排队的值数量。

配置

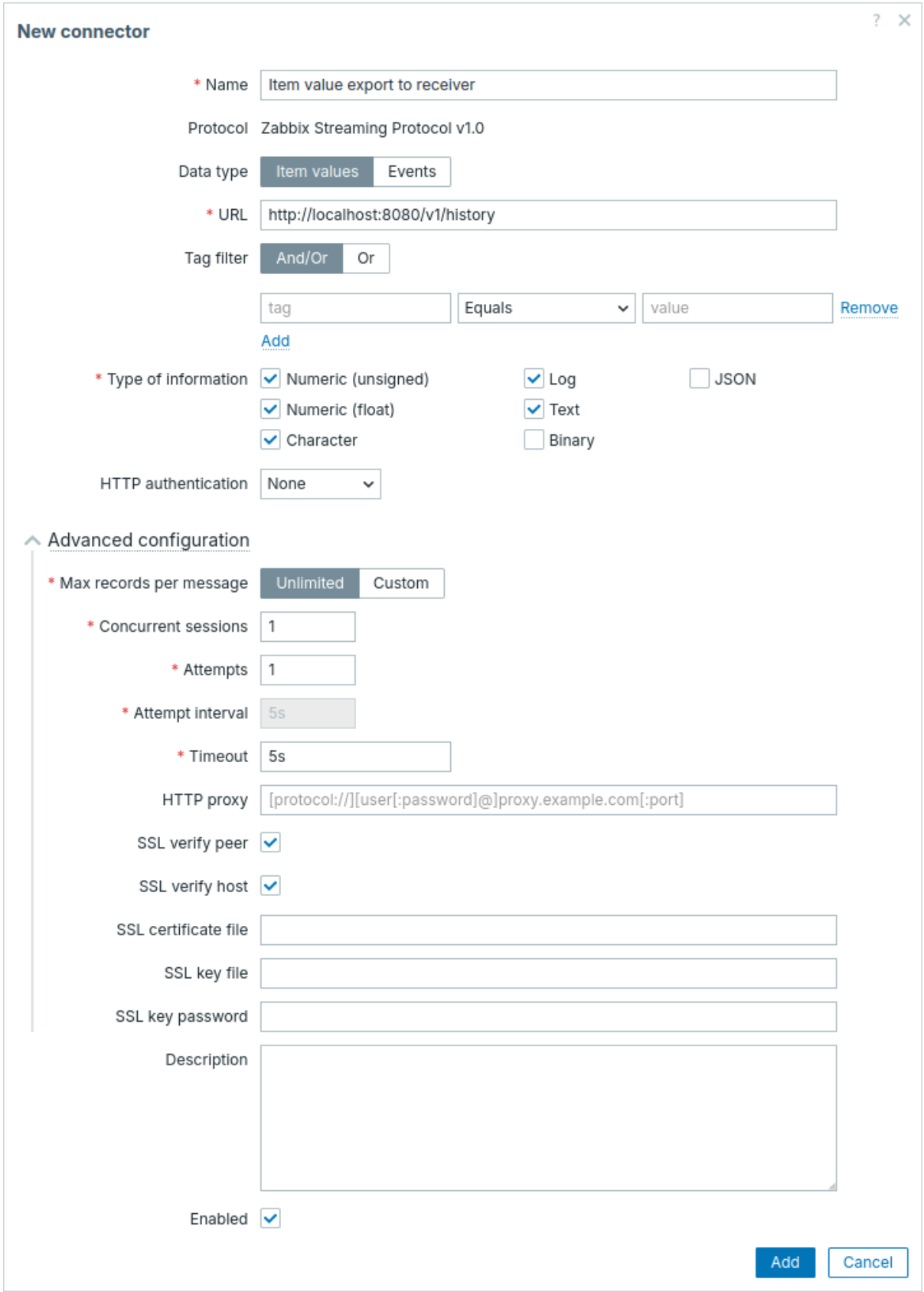
需要执行以下步骤，将数据流式传输到外部系统：

1. 设置远程系统，以接收来自 Zabbix 的数据。为此，可以使用以下工具：

- 一个简单的 接收器 示例，可将接收到的信息记录在 `events.ndjson` 和 `history.ndjson` 文件中。
- Zabbix 服务器的 Kafka 连接器 - 使用 Go 编写的轻量级服务器，用于将监控项值和事件从 Zabbix 服务器转发到 Kafka broker。

2. 在 Zabbix 中，通过调整 `zabbix_server.conf` 中的`StartConnectors` 参数，设置所需的连接器工作进程数量。连接器工作进程的数量应与 Zabbix 前端中配置的连接器数量相匹配（如果并发会话数大于 1，则应大于或等于该数量）。然后，重启 Zabbix 服务器。

3. 在 Zabbix 前端中配置新的连接器（管理 > 常规 > 连接器），并使用 `zabbix_server -R config_cache_reload` 命令重新加载服务器缓存。



必填字段以星号标记。

参数	描述
名称	输入连接器名称。
数据类型	选择要流式传输的数据类型： 监控项值 - 将监控项值从 Zabbix 流式传输到外部系统； 事件 - 将事件从 Zabbix 流式传输到外部系统。
URL	输入接收器 URL。支持用户宏。

参数	描述
标签过滤器	仅导出与标签过滤器匹配的监控项值或事件。如果未设置，则导出所有内容。 可以包含或排除特定标签及标签值。可以设置多个条件。标签名称匹配始终区分大小写。 每个条件有以下运算符可用： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件；具有相同标签名称的条件将通过 Or 条件进行分组； **Or** - 满足一个条件即可。
信息类型	选择信息类型（无符号数值、浮点数值、字符等），连接器将根据此类型筛选要流式传输的监控项值。 当数据类型设置为“监控项值”时，此字段可用。
HTTP 身份验证	选择身份验证选项： 无 - 不使用身份验证； **Basic** - 使用基本身份验证； **NTLM** - 使用 NTLM（Windows NT LAN Manager）身份验证； **Kerberos** - 使用 Kerberos 身份验证（另请参阅：配置 Kerberos 与 Zabbix）； **Digest** - 使用摘要身份验证； **Bearer** - 使用 Bearer 身份验证。
用户名	输入用户名（最多 255 个字符）。支持用户宏。 当 *HTTP* 身份验证设置为“Basic”、“NTLM”、“Kerberos”或“Digest”时，此字段可用。
密码	输入用户密码（最多 255 个字符）。支持用户宏。 当 *HTTP* 身份验证设置为“Basic”、“NTLM”、“Kerberos”或“Digest”时，此字段可用。
Bearer 令牌	输入 Bearer 令牌。支持用户宏。 当 *HTTP* 身份验证设置为“Bearer”时，此字段可用且为必填项。
高级配置	单击高级配置标题以显示高级配置选项（见下文）。
每条消息的最大记录数	指定一条消息中可以流式传输的值或事件的最大数量。
并发会话	选择为此连接器运行的发送器进程数量。最多可指定 100 个会话；默认值为“1”。
尝试次数	数据流式传输的尝试次数。最多可指定 5 次；默认值为“1”。
尝试间隔	指定连接器在数据流式传输尝试失败后应等待的时间。最多可指定 10s；默认值为“5s”。 当尝试次数设置为“2”或更大值时，此字段可用。 建立连接失败，或 HTTP 响应代码不是 200、201、202、203、204，均属于失败尝试。在发生通信错误，或 HTTP 响应代码不是 200、201、202、203、204、400、401、403、404、405、415、422 时，将触发重试。系统会跟随重定向，因此 302 -> 200 属于成功响应；而 302 -> 503 将触发重试。
超时	指定消息超时时间（1-60 秒，默认值为 5 秒）。 支持时间后缀（例如 30s、1m）。支持用户宏。
HTTP proxy	可以使用以下格式指定要使用的 HTTP proxy： `[protocol://][username[:password]@]proxy.example.com[:port]` 支持用户宏。 可以使用可选的 `protocol://` 前缀指定备用 proxy 协议（cURL 7.21.7 增加了协议前缀支持）。如果未指定协议，则将 proxy 视为 HTTP proxy。默认使用 1080 端口。 如果指定了 *HTTP proxy*，proxy 将覆盖 `http_proxy`、`HTTPS_PROXY` 等与 proxy 相关的环境变量。如果未指定，则 proxy 不会覆盖与 proxy 相关的环境变量。输入的值将按原样传递，不会进行合理性检查。 也可以输入 SOCKS proxy 地址。如果指定了错误的协议，连接器将无法从 Zabbix 流式传输监控项值或事件。 请注意，HTTP proxy 仅支持简单身份验证。
SSL 验证对等方	选中复选框以验证 Web 服务器的 SSL 证书。 服务器证书将根据系统范围 CA 位置中的受信任证书颁发机构（CA）证书进行验证。可以使用 Zabbix 服务器或 proxy 配置参数`SSLCALocation` 覆盖 CA 文件的位置。
SSL 验证主机	选中复选框以验证 Web 服务器证书的通用名称字段或主题备用名称字段是否匹配。 此选项设置 `CURLOPT_SSL_VERIFYHOST` cURL 选项。
SSL 证书文件	用于客户端身份验证的 SSL 证书文件名称。证书文件必须为 PEM[1] 格式。支持用户宏。 如果证书文件还包含私钥，请将 *SSL* 密钥文件字段留空。如果密钥已加密，请在 *SSL* 密钥密码字段中指定密码。包含此文件的目录由 Zabbix 服务器或 proxy 配置参数`SSLCertLocation` 指定。

参数	描述
SSL 密钥文件	用于客户端身份验证的 SSL 私钥文件名称。私钥文件必须为 PEM[1] 格式。支持用户宏。 包含此文件的目录由 Zabbix 服务器或 proxy 配置参数SSLKeyLocation 指定。
SSL 密钥密码	SSL 私钥文件密码。支持用户宏。
描述	输入连接器描述。
启用	选中复选框以启用连接器。

当 Kafka 连接器配置为使用以逗号分隔的 bootstrap broker 地址列表时(例如 `Kafka.URL=kafka1.example.com:9093,kafka2.example.co`
Kafka 客户端会连接到最先响应的 broker，并使用其集群元数据。如果列表中包含来自不同 Kafka 集群的地址，则只会使用响应最快的集群，其他地址将记录为不可用；因此，即使连接器已连接，也可能出现如下启动警告：

```
kafka cluster connected, but broker(s) "kafka1.example.com:9093, kafka2.example.com:9093" unavailable; wil
```

在某些环境中（专用网络、容器网络或非标准 DNS/hosts 设置），主机名或 IP 可能解析为回环地址（例如 `127.0.0.1/localhost`），或被客户端规范化，这可能导致此类警告产生误导。为减少混淆，请确保所有 `Kafka.URL` 地址都属于同一个 Kafka 集群，验证连接器主机上的 DNS 解析以及 broker 的 `advertised.listeners`，并优先使用可解析为 broker 广播地址的地址。

协议

服务器与接收器之间的通信通过 HTTP 完成，使用 REST API、NDJSON 和"Content-Type: application/x-ndjson"。

有关更多详细信息，请参见Newline-delimited JSON export protocol。

服务器请求

流式传输监控项值的示例：

```
POST /v1/history HTTP/1.1
Host: localhost:8080
Accept: */*
Accept-Encoding: deflate, gzip, br, zstd
Content-Length: 628
Content-Type: application/x-ndjson

{"host":{"host":"Zabbix server","name":"Zabbix server"},"groups":["Zabbix servers"],"item_tags":[{"tag":"f
{"host":{"host":"Zabbix server","name":"Zabbix server"},"groups":["Zabbix servers"],"item_tags":[{"tag":"f
{"host":{"host":"Zabbix server","name":"Zabbix server"},"groups":["Zabbix servers"],"item_tags":[{"tag":"b
```

流式传输事件的示例：

```
POST /v1/events HTTP/1.1
Host: localhost:8080
Accept: */*
Accept-Encoding: deflate, gzip, br, zstd
Content-Length: 333
Content-Type: application/x-ndjson

{"clock":1673454303,"ns":800155804,"value":1,"eventid":5,"name":"trigger for foo being 0","severity":0,"ho
{"clock":1673454303,"ns":832290669,"value":0,"eventid":6,"p_eventid":5}
```

接收器响应

响应由 HTTP 响应状态码和 JSON 负载组成。
对于已成功处理的请求，HTTP 响应状态码必须为 "200"、"201"、"202"、"203" 或 "204"；其他状态码表示请求失败。

成功示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Content-Type-Options: nosniff
Date: Tue, 21 Apr 2026 10:13:04 GMT
Content-Length: 23

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

错误示例：

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
X-Content-Type-Options: nosniff
Date: Tue, 21 Apr 2026 12:15:01 GMT
Content-Length: 55

{"error":"invalid character '{' after top-level value"}
```

SNMP 网关

概述

Zabbix SNMP gateway 是 snmpd 的一个 AgentX 扩展，同时支持 SNMP 轮询和 trap。

使用 Zabbix SNMP gateway，可以通过 SNMP 协议获取：

- 触发器数据；
- 故障触发器数据；
- 主机组状态（每个组中按触发器状态统计的触发器数量）

数据通过 OID 获取，OID 由一个通用基础部分和一个特定后缀组合而成。通用的 **base** 在 SNMP gateway 的配置文件中设置，例如：

- BaseOID=1.3.6.1.4.1.3043.7.55 - 用于任意触发器数据；
- ProblemBaseOID=1.3.6.1.4.1.3047.7.55 - 用于故障触发器数据；
- BaseOID=1.3.6.1.4.1.3046.7.55 - 用于主机组状态。

OID 的 **suffix** 在前端中通过主机触发器配置为一个 **tag** 来设置（例如，OIDSuffix:3）。

在这种情况下，该触发器的所有信息都可通过 OID=1.3.6.1.4.1.3043.7.55.X.3 获取。这里的 "X" 表示触发器数据字段编号（即 1 - 后缀，2 - ID，3 - 表达式，4 - 描述，等等）。

有关更详细的说明和配置文件示例，请参见 SNMP gateway readme file。

安装和设置

有关以下内容的说明，请参阅 SNMP gateway 的 readme 文件：

- 安装和配置 snmpd；
- 启用 AgentX 支持；
- 配置 Zabbix SNMP gateway；
- 为触发器状态更改配置 SNMP trap。

获取数据

正确完成所有设置后，您可以使用 snmpwalk 和 snmpget 命令来获取数据：

```
[user@localhost ~]# snmpget -v2c -c public 127.0.0.1 1.3.6.1.4.1.3043.7.55.2.3
SNMPv2-SMI::enterprises.3043.7.55.2.3 = INTEGER: 15247
```

```
[user@localhost ~]# snmpwalk -v2c -c public 127.0.0.1 1.3.6.1.4.1.3043.7.55
SNMPv2-SMI::enterprises.3043.7.55.1.1 = INTEGER: 1
SNMPv2-SMI::enterprises.3043.7.55.1.3 = INTEGER: 3
SNMPv2-SMI::enterprises.3043.7.55.1.4 = INTEGER: 4
SNMPv2-SMI::enterprises.3043.7.55.1.5 = INTEGER: 5
SNMPv2-SMI::enterprises.3043.7.55.1.6 = INTEGER: 6
SNMPv2-SMI::enterprises.3043.7.55.1.10 = INTEGER: 10
SNMPv2-SMI::enterprises.3043.7.55.2.1 = INTEGER: 15367
SNMPv2-SMI::enterprises.3043.7.55.2.3 = INTEGER: 15247
SNMPv2-SMI::enterprises.3043.7.55.2.4 = INTEGER: 15365
SNMPv2-SMI::enterprises.3043.7.55.2.5 = INTEGER: 15366
SNMPv2-SMI::enterprises.3043.7.55.2.6 = INTEGER: 13493
SNMPv2-SMI::enterprises.3043.7.55.2.10 = INTEGER: 13503
...
```

过滤选项

您可以在 SNMP 网关配置中限制问题触发器信息：

- 按严重性限制（默认 ProblemMinSeverity=-1）
- 通过隐藏已确认的问题进行限制（默认 ProblemHideAck=false）

您可以在 SNMP 网关配置中限制每个主机组的问题计数：

- 按未知状态的触发器进行限制（默认 `CountUnknown=false`）
- 按具有已确认/未确认/全部问题的触发器进行限制（默认 `CountAcknowledgeStatus=all`）

配置导出/导入

概述

Zabbix 的导出/导入功能可以在一个 Zabbix 系统与另一个系统之间交换各种配置实体。

此功能的典型使用场景包括：

- 共享模板或网络地图 - Zabbix 用户可以共享其配置参数
- 将模板上传到 Zabbix Community templates。然后其他人可以下载该模板并将文件导入 Zabbix。
- 与第三方工具集成 - 通用的 YAML、XML 和 JSON 格式使得与第三方工具和应用程序进行集成以及数据导入/导出成为可能

可导出/导入的内容

可导出/导入的对象包括：

- 主机组（仅通过 *Zabbix API*）
- 模板组（仅通过 *Zabbix API*）
- 模板
- 主机
- 网络地图
- 媒体类型
- 图像
- 仪表板
- 正则表达式

导出格式

数据可以使用 Zabbix 前端或Zabbix API 导出。支持的导出格式为 YAML、XML 和 JSON。

导出详情

- 所有受支持的元素都会导出到一个文件中。
- 从关联模板继承的主机和模板实体（监控项、触发器、图形、发现规则）不会被导出。对这些实体在主机级别所做的任何更改（例如更改监控项间隔、修改正则表达式或向低级别发现规则添加原型）在导出时都会丢失；在导入时，来自关联模板的所有实体都会按原始关联模板重新创建。
- 由低级别发现创建的实体以及依赖于这些实体的任何实体都不会被导出。例如，为 LLD 规则生成的监控项创建的触发器不会被导出。
- 当导出的主机/模板包含支持超时的实体时，如果这些实体已配置各自的超时值，则会导出这些超时值。

导入详情

- 导入会在遇到第一个错误时停止。
- 在导入图像时更新现有图像，”imagetype” 字段会被忽略，也就是说，无法通过导入更改图像类型。
- 使用”Delete missing” 选项导入主机/模板时，导入文件中不存在的主机/模板宏将在导入后从主机/模板中删除。
- 监控项、触发器、图形、discoveryRules、itemPrototypes、triggerPrototypes、graphPrototypes 的空标签没有意义，也就是说，与缺失时相同。
- 如果导入的主机/模板中的实体配置了各自的超时，将应用这些超时；否则，将应用 proxy/全局超时。
- 导入支持 YAML、XML 和 JSON，导入文件必须具有正确的文件扩展名：YAML 使用.yaml 和.yml，XML 使用.xml，JSON 使用.json。有关支持的 XML 版本，请参见兼容性信息。
- 导入仅支持 UTF-8 编码的配置文件（可带或不带 BOM）；其他编码（UTF16LE、UTF16BE、UTF32LE、UTF32BE 等）都会导致导入转换错误。

YAML 基本格式

YAML 导出格式包含以下节点：

- Zabbix YAML 导出的根节点
- 导出版本

```
zabbix_export:
  version: '8.0'
```

其他节点取决于导出的对象。

XML 格式

XML 导出格式包含以下标签：

- XML 文档的默认头部
- Zabbix XML 导出的根标签
- 导出版本

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>8.0</version>
</zabbix_export>
```

其他标签取决于导出的对象。

JSON 格式

JSON 导出格式包含以下对象：

- Zabbix JSON 导出的根对象
- 导出版本

```
{
    "zabbix_export": {
        "version": "8.0"
    }
}
```

其他对象取决于被导出的对象。

模板组

概述

在前端中，模板组只能通过模板导出进行导出。当导出一个模板时，它所属的所有组也会自动随之导出。

API 允许独立于模板导出模板组。

导出格式

```
  template_groups:
    - uuid: 36bff6c29af64692839d077febfc7079
      name: 'Network devices'
```

导出的元素

Element	Type	Description
uuid	string	此模板组的唯一标识符。
name	string	组名称。

主机群组

概述

在前端中，主机群组只能通过主机导出一起导出。当导出主机时，它所属的所有群组也会自动随之导出。

API 允许独立于主机导出主机群组。

导出格式

```
   host_groups:
    - uuid: 6f6799aa69e844b4b3918f779f2abf08
      name: 'Zabbix servers'
```

导出的元素

Element	Type	Description
uuid	string	此主机组的唯一标识符。
name	string	组名称。

模板

概述

模板会与许多相关对象及对象关系一起导出。

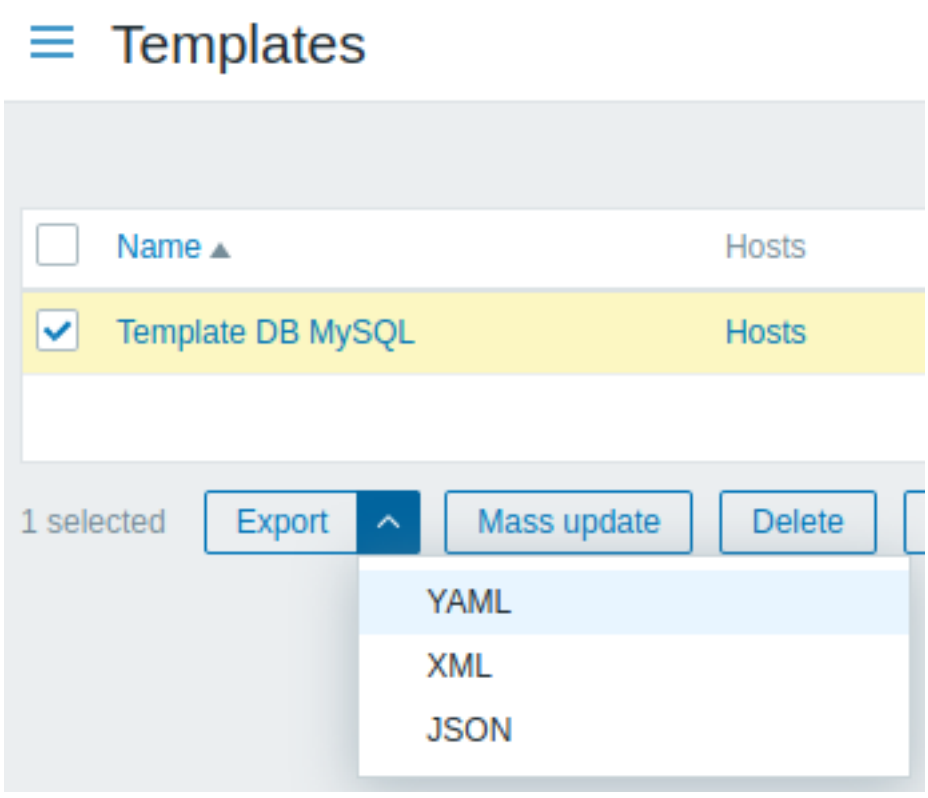
模板导出包含：

- 关联的模板组
- 关联的主机组（如果在主机原型配置中使用）
- 模板数据
- 与其他模板的关联
- 与模板组的关联
- 直接关联的监控项
- 直接关联的触发器
- 直接关联的图形
- 直接关联的仪表板
- 直接关联的发现规则及其所有原型
- 直接关联的 web 场景
- 值映射

导出

要导出模板，请执行以下操作：

1. 转到 *Data collection* > *Templates*。
2. 勾选要导出的模板复选框。
3. 单击列表下方的 *Export* 以默认 YAML 格式导出，或单击箭头选择其他导出格式。



根据所选格式，模板会导出到本地文件，并使用默认名称：

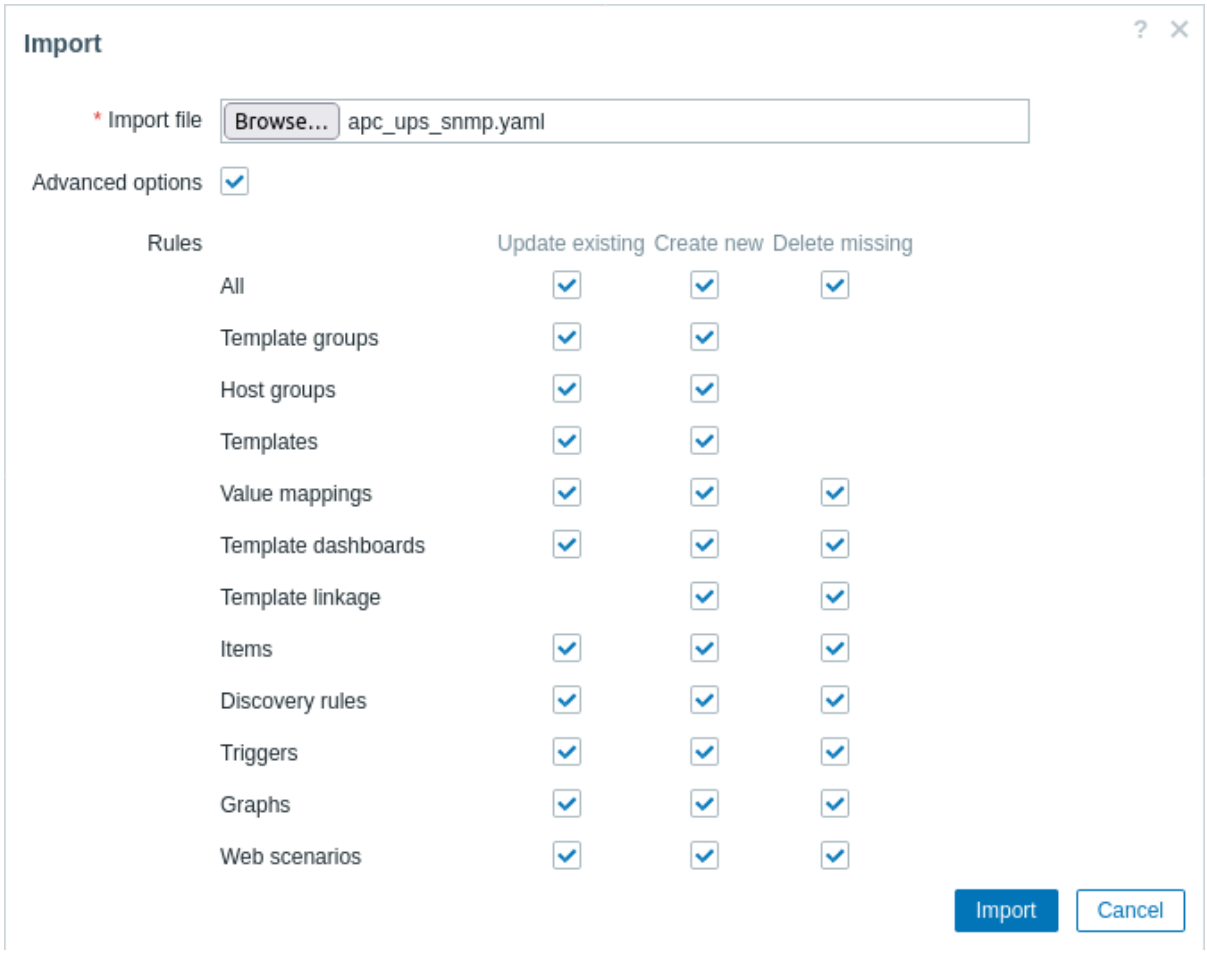
- `zabbix_export_templates.yaml` - YAML 导出；
- `zabbix_export_templates.xml` - XML 导出；
- `zabbix_export_templates.json` - JSON 导出。

导入

要导入模板，请执行以下操作：

1. 转到 数据采集 > 模板。
2. 点击右上角的 导入。
3. 选择导入文件。

4. 勾选要应用的导入规则（见下文）。
5. 点击配置表单右下角的 导入。



如果勾选 高级选项，将显示所有可导入元素的详细列表 - 根据需要勾选或取消勾选各项导入规则。

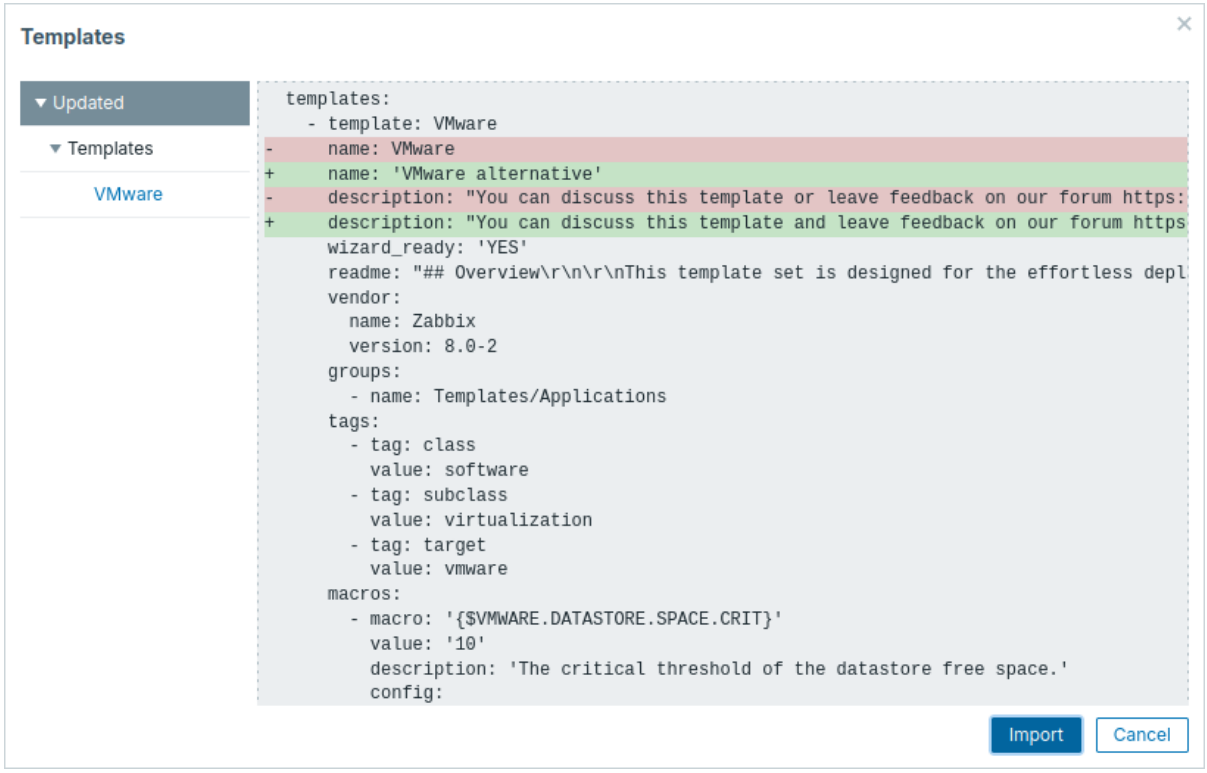
如果点击 全部行中的复选框，则其下方的所有元素都会被勾选或取消勾选。

导入规则：

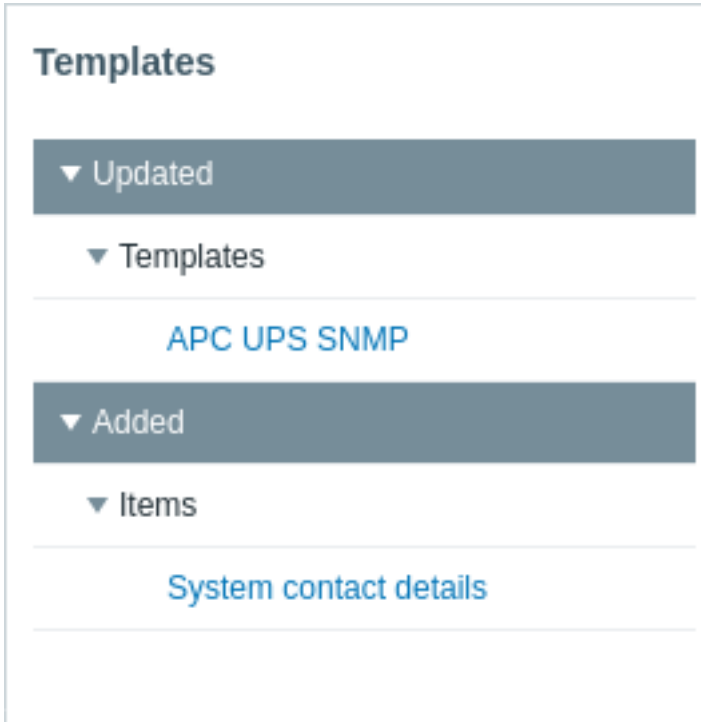
规则	描述
更新现有项	使用导入文件中的数据更新现有元素。否则，不会更新这些元素。
创建新项	使用导入文件中的数据创建新元素。否则，不会创建这些元素。
删除缺失项	删除导入文件中不存在的现有元素。否则，不会删除这些元素。 如果为 模板链接勾选了 删除缺失项，则会取消链接导入文件中不存在的当前模板链接。不会删除从已取消链接的模板继承的实体（监控项、触发器、图形等），除非同时为每个实体选择了 删除缺失项选项。

Attention:
如果已存在同名模板，则导入时应勾选 删除缺失项选项，以实现干净导入。这样，更新后的模板中不再存在的旧监控项将被删除（请注意，这意味着会丢失这些旧监控项的历史数据）。

在下一个屏幕中，您可以查看正在导入的模板内容。如果这是一个新模板，所有元素都会以绿色列出。如果正在更新现有模板，新的模板元素将以绿色突出显示；已删除的模板元素将以红色突出显示；未发生变化的元素将显示在灰色背景上。



可以使用左侧的菜单浏览更改列表。已更新部分突出显示对现有模板元素所做的所有更改。已添加部分列出新的模板元素。每个部分中的元素按元素类型分组；点击灰色箭头可展开或折叠元素组。



查看模板更改，然后点击 导入以执行模板导入。导入成功或失败的消息将显示在前端中。

导出格式　当模板被导出时，Zabbix 会生成结构化的 YAML、JSON 或 XML 格式。导出内容包括模板元素，例如模板元数据、监控项、宏、触发器、仪表板等。

每个元素都有特定用途，并且可能包含嵌套元素。

以下各节将介绍导出格式中的每个元素。示例使用 Linux by Zabbix agent 模板。

省略号（`...`）表示为简洁起见省略的元素。注释（`see table below`）用于在后续章节中对该元素进行更详细说明时。

```
zabbix_export:
  version: '8.0'
```

```
  template_groups:
    - uuid: 846977d1dfed4968bc5f8bdb363285bc
      name: 'Templates/Operating systems'
  templates: (see table below)
  triggers: (see table below)
  graphs: (see table below)
```

模板元数据

Element		Type	Description
version		string	(必填) 导出版本。
template_groups			(必填) 模板组的根元素。
	uuid	string	(必填) 此模板组的唯一标识符。
	name	string	(必填) 模板组名称。
host_groups			由主机原型使用的主机组的根元素。
	uuid	string	(必填) 此主机组的唯一标识符。
	name	string	(必填) 主机组名称。
templates			模板 的根元素。
triggers			模板触发器元素的根元素，与模板监控项触发器 相同。
graphs			模板图形元素的根元素，与主机图形 相同。

模板

```
zabbix_export:
  (...)
  templates:
    - uuid: f8f7908280354f2abeed07dc788c3747
      template: 'Linux by Zabbix agent'
      name: 'Linux by Zabbix agent'
      description: |
        这是一个官方 Linux 模板。它需要 Zabbix agent 8.0 或更高版本。(...)
      wizard_ready: 'YES'
      readme: |
        ## 概述

        这是一个官方 Linux 模板。它需要 Zabbix agent 8.0 或更高版本。(...)
      vendor:
        name: Zabbix
        version: 8.0-2
      groups:
        - name: 'Templates/Operating systems'
      items: (see table below)
      discovery_rules: (see table below)
      tags:
        - tag: class
          value: os
        - tag: target
          value: linux
      macros:
        - macro: '{$AGENT.TIMEOUT}'
          value: 3m
          description: '超时后将 agent 视为不可用。仅适用于可从 Zabbix 服务器/proxy 访问的 agent（被动模式）。'
          config: (see table below)
        - macro: '{$CPU.UTIL.CRIT}'
          value: '90'
          description: '以 % 表示的 CPU 利用率严重阈值。'
          config: (see table below)
        (...)
      dashboards: (see table below)
      valuemaps: (see table below)
    (...)
```

Element	Type	Description
uuid	string	(required) 此模板的唯一标识符。
template	string	(required) 唯一的模板名称。
name	string	可见的模板名称。
description	text	模板描述。
wizard_ready	text	此模板是否可在主机向导中选择。 可能的值：[1] NO (0, 默认), YES (1). 另请参见：Template object (`wizard_ready`).
readme	text	在主机向导中显示的模板特定配置说明。支持 Markdown 格式。
vendor		模板供应商的根元素（当导出的模板包含供应商数据时存在）。
name	string	(required) 模板供应商名称。
version	string	(required) 模板版本。 对于开箱即用模板，版本显示格式如下：Zabbix 的主版本号、分隔符（"-"）、修订号（每发布一个新模板版本递增，并在每个 Zabbix 主版本中重置）。例如，7.0-0、7.0-3、8.0-0、8.0-3。
templates		关联模板的根元素。
name	string	(required) 模板名称。
groups		模板组的根元素。
name	string	(required) 模板组名称。
items		模板监控项的根元素。
discovery_rules		模板低级别发现规则的根元素。
httptests		模板 Web 场景的根元素。
tags		模板标签的根元素。
tag	string	(required) 标签名称。
value	string	标签值。
macros		模板用户宏的根元素。
macro	string	(required) 用户宏名称。
type	string	用户宏类型。 可能的值：[1] TEXT (0, 默认), SECRET_TEXT (1), VAULT (2). 另请参见：User macro object (`type`).
value	string	用户宏值。
description	string	用户宏描述。
config		负责控制宏在主机向导中显示方式的模板宏配置根元素。
dashboards		模板仪表板的根元素。
valuemaps		模板值映射的根元素。

模板监控项

```yaml
zabbix_export:
  (...)
  templates:
    (...)
      items:
        - uuid: f94f9f4699e94c369e6c98b2a2f485ce
          name: 'Zabbix agent ping'
          key: agent.ping
          description: 'agent 对此监控项始终返回 "1"。可与 `nodata()` 结合使用来检查可用性。'
          valuemap:
            name: 'Zabbix agent ping status'
          tags:
            - tag: component
              value: system
        (...)
        - uuid: 58818005e76d46dda14d6592f601ab00
          name: '已安装软件包数量'
          key: system.sw.packages.get
          delay: 1h
          preprocessing: （请参见下表）
          tags:
            - tag: component
              value: os
          triggers: （请参见下表）
        - uuid: 403cebed115441369e94d35d070ca7b8
```

```
      name: '内存利用率'
      type: DEPENDENT
      key: vm.memory.utilization
      value_type: FLOAT
      units: '%'
      description: '已使用内存的百分比按 `100-pavailable` 计算。'
      preprocessing: （请参见下表）
      master_item:
        key: 'vm.memory.size[pavailable]'
      tags:
        - tag: component
          value: memory
      triggers: （请参见下表）
    (...)
```

元素	类型	描述
uuid	string	（必需）此监控项的唯一标识符。
name	string	（必需）监控项名称。
type	string	监控项类型。 可能的值：[1] ZABBIX_PASSIVE（0，默认）、TRAP（2）、SIMPLE（3）、INTERNAL（5）、ZABBIX_ACTIVE（7）、EXTERNAL（10）、ODBC（11）、IPMI（12）、SSH（13）、TELNET（14）、CALCULATED（15）、JMX（16）、SNMP_TRAP（17）、DEPENDENT（18）、HTTP_AGENT（19）、SNMP_AGENT（20）、ITEM_TYPE_SCRIPT（21）、ITEM_TYPE_BROWSER（22）、TELEMETRY_QUERY（24）。
snmp_oid	string	（SNMP_AGENT 监控项必需）SNMP 对象 ID。
key	string	（必需）监控项键值。
delay	string	监控项的更新间隔。 默认值：1m。对于 TRAP 监控项，该值始终为 0。
history	string	历史数据的保存时长（使用时间后缀、用户宏或LLD 宏）。 默认值：31d。
trends	string	趋势数据的保存时长（使用时间后缀、用户宏或LLD 宏）。 默认值：365d。
status	string	监控项状态。 可能的值：[1] ENABLED（0，默认）、DISABLED（1）。
value_type	string	接收值类型。 可能的值：[1] FLOAT（0）、CHAR（1）、LOG（2）、UNSIGNED（3，默认）、TEXT（4）、BINARY（5）、JSON（6）。
allowed_hosts	string	允许为此监控项发送数据的主机 IP 地址列表，地址之间以逗号分隔。 支持 TRAP 和 HTTP_AGENT 监控项。
units	string	接收值的单位（bps、B 等）。
params	text	根据监控项类型而定的其他参数（SSH 和 TELNET 监控项执行的脚本；ODBC 监控项的 SQL 查询；CALCULATED 监控项的公式；ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项的脚本）。
ipmi_sensor	string	IPMI 传感器。 支持 IPMI 监控项。
authtype	string	身份验证类型。 支持 SSH 和 HTTP_AGENT 监控项。 SSH 监控项的可能值：[1] PASSWORD（0，默认）、PUBLIC_KEY（1）。 HTTP_AGENT 监控项的可能值：[1] NONE（0，默认）、BASIC（1）、NTLM（2）、Kerberos（3）或 Digest（4）。
username	string	（SSH 和 TELNET 监控项必需）用于身份验证的用户名。 支持 SIMPLE、ODBC、JMX 和 HTTP_AGENT 监控项。 用于 JMX 监控项时，还应指定 `password`（见下文），或者将这两个元素都留空。
password	string	（SSH 和 TELNET 监控项必需）用于身份验证的密码。 支持 SIMPLE、ODBC、JMX 和 HTTP_AGENT 监控项。 用于 JMX 监控项时，还应指定 `username`（见上文），或者将这两个元素都留空。
publickey	string	（SSH 监控项必需）公钥文件名称。
privatekey	string	（SSH 监控项必需）私钥文件名称。
description	text	监控项描述。
inventory_link	string	由监控项填充的主机资产清单字段。 可能的值：[1] NONE（0）、ALIAS（4）等（支持的字段请参见主机资产清单）。

元素	类型	描述
valuemap		监控项值映射的根元素。
name	string	（必需）用于此监控项的值映射名称。
logtimefmt	string	日志条目中的时间格式。 支持 LOG 值类型的监控项。
preprocessing		监控项值预处理的根元素。
step		模板监控项值预处理步骤的根元素。
jmx_endpoint	string	JMX 端点。 支持 JMX 监控项。
master_item		（DEPENDENT 监控项必需）依赖监控项的主监控项的根元素。
key	string	（必需）依赖监控项的主监控项键值。
timeout	string	监控项数据轮询请求超时时间。 支持超时列表中的监控项类型。
url	string	（HTTP_AGENT 监控项必需）URL 字符串。
query_fields		查询参数的根元素。 支持 HTTP_AGENT 监控项。
name	string	（HTTP_AGENT 监控项必需）查询参数名称。
value	string	查询参数值。 支持 HTTP_AGENT 监控项。
parameters		用户定义参数的根元素。 支持 ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项。
name	string	（ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项必需）用户定义参数名称。
value	string	用户定义参数值。 支持 ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项。
posts	string	HTTP(S) 请求正文数据。 支持 HTTP_AGENT 监控项。
status_codes	string	必需的 HTTP 状态码范围，使用逗号分隔。 支持 HTTP_AGENT 监控项。
follow_redirects	string	轮询数据时跟随响应重定向。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO（0）、YES（1，默认）。
post_type	string	POST 数据正文类型。 支持 HTTP_AGENT 监控项。 可能的值：[1] RAW（0，默认）、JSON（2）、XML（3）。
http_proxy	string	HTTP(S) proxy 连接字符串。 支持 HTTP_AGENT 监控项。
headers		HTTP(S) 请求标头的根元素。 支持 HTTP_AGENT 监控项。
name	string	（HTTP_AGENT 监控项必需）标头名称。
value	string	（HTTP_AGENT 监控项必需）标头值。
retrieve_mode	string	应存储响应的哪一部分。 支持 HTTP_AGENT 监控项。 可能的值：[1] BODY（0，默认）、HEADERS（1）、BOTH（2）。
request_method	string	请求方法类型。 支持 HTTP_AGENT 监控项。 可能的值：[1] GET（0，默认）、POST（1）、PUT（2）、HEAD（3）。
output_format	string	响应处理方式。 支持 HTTP_AGENT 监控项。 可能的值：[1] RAW（0，默认）、JSON（1）。
allow_traps	string	允许以与 trapper 监控项类似的方式填充值。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO（0，默认）、YES（1）。
ssl_cert_file	string	SSL 公钥文件路径。 支持 HTTP_AGENT 监控项。
ssl_key_file	string	SSL 私钥文件路径。 支持 HTTP_AGENT 监控项。
ssl_key_password	string	SSL 私钥文件密码。 支持 HTTP_AGENT 监控项。
verify_peer	string	是否验证主机证书的真实性。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO（0，默认）、YES（1）。

元素	类型	描述
verify_host	string	是否验证连接的主机名与主机证书中的主机名匹配。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO（0，默认）、YES（1）。
time_shift	string	监控项查询的时间范围结束时间相对于当前时间向前偏移的时长。 支持 TELEMETRY_QUERY 监控项。
lookback_limit	string	在没有可继续使用的先前查询时（例如监控项首次运行时），或先前查询早于 `delay` 时（例如 Zabbix 服务器停止运行期间），监控项查询数据时相对于当前时间向前回溯的时长。 支持 TELEMETRY_QUERY 监控项。
granularity	string	查询完整时间范围拆分后每个片段的长度。 支持 TELEMETRY_QUERY 监控项。
query		（TELEMETRY_QUERY 监控项必需）遥测查询的根元素。定义监控项从数据源查询的内容、数据过滤方式以及结果聚合方式。
signal_type	string	（必需）监控项查询的信号类型。 支持 TELEMETRY_QUERY 监控项。 可能的值：[1] TRACES（0）、METRICS（1）、LOGS（2）。
metric_point_type	string	（METRICS 信号类型必需）监控项查询的指标点类型。 支持 TELEMETRY_QUERY 监控项。 可能的值：[1] SUM（0）、GAUGE（1）、HISTOGRAM（2）、EXPONENTIAL_HISTOGRAM（3）。
columns		（必需）监控项查询的遥测数据列的根元素。 此元素包含以下元素： `column`（string）- （必需）监控项查询的遥测数据列名称。 `attribute_key`（string）- （存储键值对的列必需）监控项在存储键值对的列中查询的键名称。
aggregated_columns		（必需）监控项查询的遥测数据聚合列的根元素。 此元素包含以下元素： `column`（string）- （MIN、MAX、AVG、SUM 和 PERCENTILE 函数必需）监控项查询的遥测数据列名称。 `function`（string）- （必需）监控项应用于列记录的函数。可能的值：[1] MIN（1）、MAX（2）、AVG（3）、COUNT（4）、SUM（5）、PERCENTILE（8）。 `parameters`（array）- （PERCENTILE 函数必需）监控项从列记录中计算的百分位数。 `alias`（string）- （必需）计算值的唯一名称，用于在监控项值中标识该值。
filter		（必需）查询中列记录的遥测查询过滤器的根元素。 此元素包含以下元素： `evaltype`（string）- （必需）过滤条件评估方法。可能的值：[1] AND_OR（0）、AND（1）、OR（2）、FORMULA（3）。 `formula`（string）- （FORMULA 评估方法必需）用于通过自定义表达式评估过滤条件的用户定义表达式。 `conditions`（string）- （必需）遥测数据列中的记录必须满足的遥测数据过滤条件的根元素。 `conditions` 元素包含以下元素： `column`（string）- （必需）过滤条件所应用的遥测数据列名称。 `attribute_key`（string）- （存储键值对的列必需）条件在存储键值对的列中应用的键名称。 `operator`（string）- （必需）过滤条件运算符。单值列的可能值：[1] EQUAL（0）、NOT_EQUAL（1）、LIKE（2）、NOT_LIKE（3）。存储键值对的列的可能值：[1] EQUAL（0）、NOT_EQUAL（1）、EXISTS（12）。 `formulaid`（string）- （FORMULA 评估方法必需）用于从自定义表达式引用条件的任意唯一 ID。 `value`（string）- （EQUAL、NOT_EQUAL、LIKE 和 NOT LIKE 操作必需）用于比较的值。
tags		监控项标签的根元素。
tag	string	（必需）标签名称。
value	string	标签值。
triggers		模板监控项触发器的根元素。

Note:
另请参见：监控项对象（参阅名称匹配的相关属性）。

模板监控项值预处理步骤

```
zabbix_export:
  (...)
  templates:
    (...)
      items:
        (...)
        - uuid: 58818005e76d46dda14d6592f601ab00
          (...)
          preprocessing:
            - type: JSONPATH
              parameters:
                - $.length()
            - type: DISCARD_UNCHANGED_HEARTBEAT
              parameters:
                - 12h
          (...)
```

Element	Type	Description
type	string	(必填) 监控项值预处理步骤类型。 可能的值：[1] MULTIPLIER (1), RTRIM (2), LTRIM (3), TRIM (4), REGEX (5), BOOL_TO_DECIMAL (6), OCTAL_TO_DECIMAL (7), HEX_TO_DECIMAL (8), SIMPLE_CHANGE (9, calculated: received value - previous value), CHANGE_PER_SECOND (10, calculated: (received value - previous value)/(time now - time of last check)), XMLPATH (11), JSONPATH (12), IN_RANGE (13), MATCHES_REGEX (14), NOT_MATCHES_REGEX (15), CHECK_JSON_ERROR (16), CHECK_XML_ERROR (17), CHECK_REGEX_ERROR (18), DISCARD_UNCHANGED (19), DISCARD_UNCHANGED_HEARTBEAT (20), JAVASCRIPT (21), PROMETHEUS_PATTERN (22), PROMETHEUS_TO_JSON (23), CSV_TO_JSON (24), STR_REPLACE (25), CHECK_NOT_SUPPORTED (26), XML_TO_JSON (27), SNMP_WALK_VALUE (28), SNMP_WALK_TO_JSON (29), SNMP_GET_VALUE (30).
parameters		(必填) 监控项值预处理步骤参数的根元素。
parameter	string	监控项值预处理步骤的单个参数。
error_handler	string	预处理步骤失败时使用的操作类型。 可能的值：[1] ORIGINAL_ERROR (0, 默认), DISCARD_VALUE (1), CUSTOM_VALUE (2), CUSTOM_ERROR (3).
error_handler_params	string	错误处理器参数。

Note:
另请参见：监控项预处理对象（请参阅名称匹配的相应属性）。

模板监控项触发器

```
zabbix_export:
  (...)
  templates:
    (...)
      items:
        (...)
        - uuid: 58818005e76d46dda14d6592f601ab00
          (...)
          triggers:
            - uuid: b950c306394f4b3c902060a8273cbcde
              expression: 'change(/Linux by Zabbix agent/system.sw.packages.get)<>0'
              name: 'Linux: Number of installed packages has been changed'
              priority: WARNING
              manual_close: 'YES'
              tags:
```

```
            - tag: scope
              value: notice
      (...)
      - uuid: 403cebed115441369e94d35d070ca7b8
        (...)
        triggers:
          - uuid: cfd395b1cde74ef18a5e5f840bd5142a
            expression: 'min(/Linux by Zabbix agent/vm.memory.utilization,5m)>{$MEMORY.UTIL.MAX}'
            name: 'Linux: High memory utilization'
            event_name: 'Linux: High memory utilization (>{$MEMORY.UTIL.MAX}% for 5m)'
            priority: AVERAGE
            description: 'The system is running out of free memory.'
            dependencies:
              - name: 'Linux: Lack of available memory'
                expression: 'max(/Linux by Zabbix agent/vm.memory.size[available],5m)<{$MEMORY.AVAILABLE
            tags:
              - tag: scope
                value: capacity
              - tag: scope
                value: performance
      (...)
```

Element	Type	Description
uuid	string	(required) 该触发器的唯一标识符。
expression	string	(required) 触发器表达式。
recovery_mode	string	生成 OK 事件的依据。 Possible values:[1] EXPRESSION (0, default), RECOVERY_EXPRESSION (1), NONE (2).
recovery_expression	string	触发器恢复表达式。
correlation_mode	string	关联模式（无事件关联或按标签进行事件关联）。 Possible values:[1] DISABLED (0, default), TAG_VALUE (1).
correlation_tag	string	用于事件关联的标签名称。
name	string	(required) 触发器名称。
event_name	string	事件名称。
opdata	string	运行时数据。
url_name	string	与触发器关联的 URL 标签。
url	string	与触发器关联的 URL。
status	string	触发器状态。 Possible values:[1] ENABLED (0, default), DISABLED (1).
priority	string	触发器严重性。 Possible values:[1] NOT_CLASSIFIED (0, default), INFO (1), WARNING (2), AVERAGE (3), HIGH (4), DISASTER (5).
description	text	触发器描述。
type	string	事件生成类型（单个问题事件或多个问题事件）。 Possible values:[1] SINGLE (0, default), MULTIPLE (1).
manual_close	string	手动关闭问题事件。 Possible values:[1] NO (0, default), YES (1).
dependencies		依赖项根元素。
name	string	(required) 依赖触发器名称。
expression	string	(required) 依赖触发器表达式。
recovery_expression	string	依赖触发器恢复表达式。
tags		触发器标签根元素。
tag	string	(required) 标签名称。
value	string	标签值。

Note:
另请参见：Trigger object（请参考名称匹配的相关属性）。

模板低级发现规则

```
zabbix_export:
  (...)
```

```
templates:
  (...)
    discovery_rules:
      - uuid: acfdea9c46ef48c68e6636f43b8f96a2
        name: 'Network interface discovery'
        key: net.if.discovery
        delay: 1h
        filter: (see table below)
        description: 'The discovery of network interfaces.'
        item_prototypes: (see table below)
        trigger_prototypes: (see table below)
        graph_prototypes: (see table below)
      (...)
```

Attention:
大多数模板低级发现规则元素与模板监控项 相同。下表描述了与模板监控项不同的元素。

Element	Type	Description
type	string	监控项类型。 可选值:[1] ZABBIX_PASSIVE (0, default), TRAP (2), SIMPLE (3), INTERNAL (5), ZABBIX_ACTIVE (7), EXTERNAL (10), ODBC (11), IPMI (12), SSH (13), TELNET (14), JMX (16), DEPENDENT (18), HTTP_AGENT (19), SNMP_AGENT (20), ITEM_TYPE_SCRIPT (21), ITEM_TYPE_BROWSER (22).
key	string	(必填) 低级发现规则键值。
filter		模板低级发现规则过滤器 的根元素。
lifetime	string	时间周期（使用秒、时间后缀 或用户宏），超过该时间后，将删除不再发现的资源。 默认值: 7d。
lifetime_type	string	删除丢失的 LLD 资源的场景。 可选值: DELETE_NEVER, DELETE_IMMEDIATELY, DELETE_AFTER。
enabled_lifetime	string	时间周期（使用秒、时间后缀 或用户宏），超过该时间后，将禁用不再发现的资源。
enabled_lifetime_type	string	禁用丢失的 LLD 资源的场景。 可选值: DISABLE_NEVER, DISABLE_IMMEDIATELY, DISABLE_AFTER。
item_prototypes		模板监控项原型元素的根元素，与模板监控项 相同。
trigger_prototypes		模板触发器原型元素的根元素，与模板监控项触发器 相同。
graph_prototypes		模板图形原型元素的根元素，与主机图形 相同。
host_prototypes		模板主机原型元素的根元素，与主机 相同。
parent_discovery_rule		低级发现规则原型的父级低级发现规则（或规则原型）的根元素。此属性表示它是一个 LLD 规则原型，是所引用规则/规则原型的直接子级。
key	string	(必填) 父级低级发现规则（或规则原型）键值。
master_item	string	(DEPENDENT 规则必填) 依赖规则主监控项的根元素。
lld_macro_paths		低级发现规则宏路径的根元素。
lld_macro	string	(必填) 低级发现规则宏名称。
path	string	(必填) 值选择器，该值将被分配给相应的宏。
preprocessing		低级发现规则值预处理的根元素。
step		低级发现规则值预处理步骤元素的根元素，与模板监控项值预处理步骤 相同，但可选值更少。 另请参见: LLD 规则预处理对象。
overrides		低级发现规则覆盖规则的根元素。
name	string	(必填) 唯一的覆盖名称。
step	string	(必填) 覆盖的唯一顺序号。
stop	string	如果匹配，则停止处理后续覆盖。
filter		模板低级发现规则覆盖规则过滤器元素的根元素，与模板低级发现规则过滤器 相同。
operations		模板低级发现规则覆盖操作 的根元素。

Note:
另请参见: LLD 规则对象（请参考名称匹配的相关属性）。

模板低级别发现规则过滤器

```
zabbix_export:
  (...)
```

```yaml
templates:
  (...)
    discovery_rules:
      - uuid: acfdea9c46ef48c68e6636f43b8f96a2
        (...)
        filter:
          evaltype: AND
          conditions:
            - macro: '{#IFNAME}'
              value: '{$NET.IF.IFNAME.MATCHES}'
            - macro: '{#IFNAME}'
              value: '{$NET.IF.IFNAME.NOT_MATCHES}'
              operator: NOT_MATCHES_REGEX
        (...)
      - uuid: 2bbdc79f082d4c618e01bec625e9c90a
        (...)
        filter:
          evaltype: AND
          conditions:
            - macro: '{#DEVNAME}'
              value: '{$VFS.DEV.DEVNAME.MATCHES}'
            - macro: '{#DEVNAME}'
              value: '{$VFS.DEV.DEVNAME.NOT_MATCHES}'
              operator: NOT_MATCHES_REGEX
            - macro: '{#DEVTYPE}'
              value: disk
        (...)
```

Element	Type	Description
evaltype	string	覆盖过滤器条件的计算方法。 可能的值：[1] AND_OR (0，默认)、AND (1)、OR (2)、FORMULA (3)。
formula	string	用于过滤器条件的自定义计算公式。
conditions		过滤器条件的根元素。
macro	string	(必填) 用于执行检查的低级别发现宏名称。
value	string	要比较的值。
operator	string	条件运算符。 可能的值：[1] MATCHES_REGEX (8，默认)、NOT_MATCHES_REGEX (9)。
formulaid	string	(必填) 用于在自定义表达式中引用某个条件的任意唯一 ID。只能包含大写字母。在修改过滤器条件时必须由用户定义，但在之后再次请求时会重新生成。

Note:
另请参见：LLD 规则过滤器对象（请参阅名称匹配的相关属性）。

模板低级别发现规则覆盖操作

Element	Type	Description
operationobject	string	操作所应用到的对象。 可能的值:[1] ITEM_PROTOTYPE (0), TRIGGER_PROTOTYPE (1), GRAPH_PROTOTYPE (2), HOST_PROTOTYPE (3).
operator	string	覆盖条件运算符。 可能的值:[1] EQUAL (1), NOT_EQUAL (2), LIKE (3), NOT_LIKE (4), REGEXP (5), NOT_REGEXP (6).
value	string	用于覆盖条件运算符的正则表达式或字符串。
status	string	覆盖操作后对象的状态。
discover	string	是否将该对象作为发现结果添加。
delay	string	在覆盖操作后为监控项原型设置的更新间隔。
history	string	在覆盖操作后为监控项原型设置的历史数据保存周期。
trends	string	在覆盖操作后为监控项原型设置的趋势数据保存周期。
severity	string	在覆盖操作后为触发器原型设置的严重性。
tags		覆盖操作后为对象设置的标签根元素。

Element	Type	Description
tag	string	(必填) 标签名称。
value	string	标签值。
templates		覆盖操作后链接到主机原型的模板根元素。
name	string	(必填) 模板名称。
inventory_mode	string	覆盖操作后设置的主机原型清单模式。

Note:
另请参见: LLD 规则覆盖操作对象（请参阅名称匹配的相关属性）。

模板 web 场景

Element	Type	Description
uuid	string	(必填) 此 web 场景的唯一标识符。
name	string	(必填) web 场景名称。
delay	string	执行 web 场景的频率（使用秒、时间后缀 或用户宏）。 默认值：1m。
attempts	integer	执行 web 场景步骤的尝试次数。 可能的值：[1] 1-10（默认值：1）。
agent	string	客户端 agent。Zabbix 将伪装成所选浏览器。 当网站针对不同浏览器返回不同内容时，这很有用。 默认值：Zabbix。
http_proxy	string	web 场景使用的 proxy，格式为： `http://[username[:password]@]proxy.example.com[:port]`
variables		web 场景变量的根元素，这些变量可在场景步骤中使用。
name	string	(必填) 变量名称。
value	text	(必填) 变量值。
headers		执行请求时要发送的 HTTP 标头根元素。 标头应使用与 HTTP 协议中相同的语法列出。
name	string	(必填) 标头名称。
value	text	(必填) 标头值。
status	string	web 场景状态。 可能的值：[1] ENABLED (0，默认值)、DISABLED (1)。
authentication	string	认证方法。 可能的值：[1] NONE (0，默认值)、BASIC (1)、NTLM (2)、Kerberos (3) 或 Digest (4)。
http_user	string	用于 BASIC (HTTP)、NTLM、Kerberos 或 Digest 认证的用户名。
http_password	string	用于 BASIC (HTTP)、NTLM、Kerberos 或 Digest 认证的密码。
verify_peer	string	验证 web 服务器的 SSL 证书。 可能的值：[1] NO (0，默认值)、YES (1)。
verify_host	string	验证 web 服务器证书的 *Common Name* 字段或 *Subject Alternate Name* 字段是否匹配。 可能的值：[1] NO (0，默认值)、YES (1)。
ssl_cert_file	string	用于客户端认证的 SSL 证书文件名（必须为 PEM 格式）。
ssl_key_file	string	用于客户端认证的 SSL 私钥文件名（必须为 PEM 格式）。
ssl_key_password	string	SSL 私钥文件密码。
steps		(必填) 模板 web 场景步骤 的根元素。
tags		web 场景标签的根元素。
tag	string	(必填) 标签名称。
value	string	标签值。

Note:
另请参见：Web scenario object（请参阅名称匹配的相关属性）。

模板 web 场景步骤

Element	Type	Description
name	string	(required) Web 场景步骤名称。
url	string	(required) 用于监控的 URL。
query_fields		查询参数的根元素（在执行请求时要添加到 URL 的 HTTP 字段数组）。

Element	Type	Description
name	string	(required) 查询参数名称。
value	string	查询参数值。
posts		HTTP POST 变量的根元素（字符串（原始 post 数据）或 HTTP 字段数组（表单字段数据））。
name	string	(required) Post 字段名称。
value	string	(required) Post 字段值。
variables		步骤级变量（宏）的根元素，这些变量应在此步骤之后应用。 如果变量值带有'regex:' 前缀，则会根据该前缀后面的正则表达式模式，从此步骤返回的数据中提取其值
name	string	(required) 变量名称。
value	text	(required) 变量值。
headers		执行请求时要发送的 HTTP 标头的根元素。
name	string	(required) 标头名称。
value	text	(required) 标头值。
follow_redirects	string	跟随 HTTP 重定向。 可选值:[1] NO (0), YES (1, 默认)。
retrieve_mode	string	HTTP 响应获取模式。 可选值:[1] BODY (0, 默认), HEADERS (1), BOTH (2)。
timeout	string	步骤执行的超时时间（使用秒、time suffix 或user macro）。 默认值: 15s。
required	string	响应中必须存在的文本（如果为空则忽略）。
status_codes	string	以逗号分隔的可接受 HTTP 状态码列表（例如 200-201,210-299；如果为空则忽略）。

Note:
另请参见: Web 场景步骤对象（请参阅名称匹配的相关属性）。

模板宏配置

```
zabbix_export:
  (...)
  templates:
    - uuid: f8f7908280354f2abeed07dc788c3747
      (...)
      macros:
        - macro: '{$AGENT.TIMEOUT}'
          (...)
          config:
            type: TEXT
            priority: '1'
            label: '自上次看到 Zabbix agent 以来的秒数'
            description: '超出此时间后，agent 将被视为不可用。'
        - macro: '{$CPU.UTIL.CRIT}'
          (...)
          config:
            type: TEXT
            priority: '2'
            section_name: Thresholds
            label: 'CPU 利用率阈值'
            description: 'CPU 利用率的严重阈值，以 % 表示。取值范围为 0 到 100（含）。'
            regex: '^-?([0-9]+|(([0-9]+)\.([0-9]+)))$'
        (...)
        - macro: '{$IFCONTROL}'
          (...)
          config:
            type: CHECKBOX
            priority: '19'
            label: '接口控制'
            description: '当接口运行状态变为 "Link down" 时触发触发器。'
            options:
              - checked: '1'
                unchecked: '0'
        (...)
```

Element	Type	Description
type	string	(必需) 宏输入字段的类型。 可能的值：[1] NOCONF (0), TEXT (1), LIST (2), CHECKBOX (3).
priority	string	宏在宏列表中的位置。
section_name	string	宏分组所在的可折叠部分的标签。
label	string	(TEXT、LIST、CHECKBOX 宏必需) 宏标签。
description	text	宏帮助文本。支持 Markdown 格式。
required	string	标记该宏为必填。 可能的值：[1] NO (0), YES (1). 支持 TEXT 和 LIST 宏。
regex	string	用于验证文本框字段中用户输入的正则表达式。 支持 TEXT 宏。
options		LIST 项或 CHECKBOX 值的根元素。
value	string	(LIST 宏必需) LIST 项值。
text	string	(LIST 宏必需) LIST 项文本。
checked	string	(CHECKBOX 宏必需) 表示选中状态的值。 可能的值：[1] 0 (false), 1 (true). 支持 CHECKBOX 宏。
unchecked	string	(CHECKBOX 宏必需) 表示未选中状态的值。 可能的值：[1] 0 (false), 1 (true). 支持 CHECKBOX 宏。

Note:
另请参见：宏配置对象（请参阅名称匹配的相关属性）。

模板仪表板

```
zabbix_export:
  (...)
  templates:
    (...)
      dashboards:
        - uuid: c689ad3115fd46a4b927d1f70ee2e5a4
          name: Filesystems
          pages:
            - name: Overview
              widgets: (see table below)
        (...)
```

Element		Type	Description
uuid		string	(必填) 此仪表板的唯一标识符。
name		string	(必填) 模板仪表板名称。
display period		integer	仪表板页面的显示时长。
auto_start		string	幻灯片自动开始。 可能的值：[1] NO (0), YES (1, default).
pages			模板仪表板页面的根元素。
	name	string	页面名称。
	display period	integer	幻灯片间隔。
	sortorder	integer	页面排序顺序。
widgets			模板仪表板小部件的根元素。

Note:
另请参见：模板仪表板对象（请参阅名称匹配的相应属性）。

模板仪表板小部件

```
zabbix_export:
  (...)
```

```yaml
templates:
  (...)
    dashboards:
      - uuid: c689ad3115fd46a4b927d1f70ee2e5a4
        name: Filesystems
        pages:
          - name: Overview
            widgets:
              - type: graphprototype
                width: '72'
                height: '5'
                fields:
                  - type: INTEGER
                    name: columns
                    value: '3'
                  - type: GRAPH_PROTOTYPE
                    name: graphid.0
                    value:
                      host: 'Linux by Zabbix agent'
                      name: 'FS [{#FSTYPE}({#FSNAME})]: Space utilization chart (relative to total)'
                  - type: STRING
                    name: reference
                    value: AAAEB
              - type: graphprototype
                'y': '5'
                width: '72'
                height: '5'
                fields:
                  - type: INTEGER
                    name: columns
                    value: '1'
                  - type: GRAPH_PROTOTYPE
                    name: graphid.0
                    value:
                      host: 'Linux by Zabbix agent'
                      name: 'FS [{#FSTYPE}({#FSNAME})]: Space usage graph, in % (relative to max availab
                  - type: STRING
                    name: reference
                    value: AAAEC
      (...)
```

Element	Type	Description
type	string	(required) 小部件类型。
name	string	小部件名称。
x	integer	从模板仪表板左侧开始的水平位置。 Possible values:[1] 0-71.
y	integer	从模板仪表板顶部开始的垂直位置。 Possible values:[1] 0-63.
width	integer	小部件宽度。 Possible values:[1] 1-72.
height	integer	小部件高度。 Possible values:[1] 1-64.
hide_header	string	隐藏小部件标题。 Possible values:[1] NO (0, default), YES (1).
fields		模板仪表板小部件字段的根元素。
type	string	(required) 小部件字段类型。 Possible values:[1] INTEGER (0), STRING (1), ITEM (4), ITEM_PROTOTYPE (5), GRAPH (6), GRAPH_PROTOTYPE (7), MAP (8), SERVICE (9), SLA (10), USER (11), ACTION (12), MEDIA_TYPE (13).
name	string	(required) 小部件字段名称。
value	mixed	(required) 小部件字段值，具体取决于字段类型。

Note:
另请参见：Template dashboard widget object（请参阅名称匹配的相关属性）。

模板值映射

```
zabbix_export:
  (...)
  templates:
    (...)
      valuemaps:
        - uuid: 8c048c6cca8248f2860c208e8db0f59e
          name: ifOperStatus
          mappings:
            - value: '0'
              newvalue: unknown
            - value: '1'
              newvalue: notpresent
            - value: '2'
              newvalue: down
            - value: '3'
              newvalue: lowerlayerdown
            - value: '4'
              newvalue: testing
            - value: '5'
              newvalue: dormant
            - value: '6'
              newvalue: up
        (...)
```

Element	Type	Description
uuid	string	(required) 此值映射的唯一标识符。
name	string	(required) 值映射名称。
mapping		映射的根元素。
type	string	映射匹配类型。 可能的值:[1] EQUAL (0, default), GREATER_OR_EQUAL (2), LESS_OR_EQUAL (3), IN_RANGE (4), REGEXP (5), DEFAULT (6).
value	string	原始值。
newvalue	string	(required) 原始值映射到的值。

Note:
另请参见: Value map object（请参阅名称匹配的相应属性）。

脚注 [1] 括号中的 API 整数值，例如 ENABLED (0)，仅供参考。有关更多信息，请参见表项中的链接 API 对象页面，或每个章节末尾的相关页面。

主机

概述

主机会连同许多相关对象及对象关系一起导出。

主机导出包含：

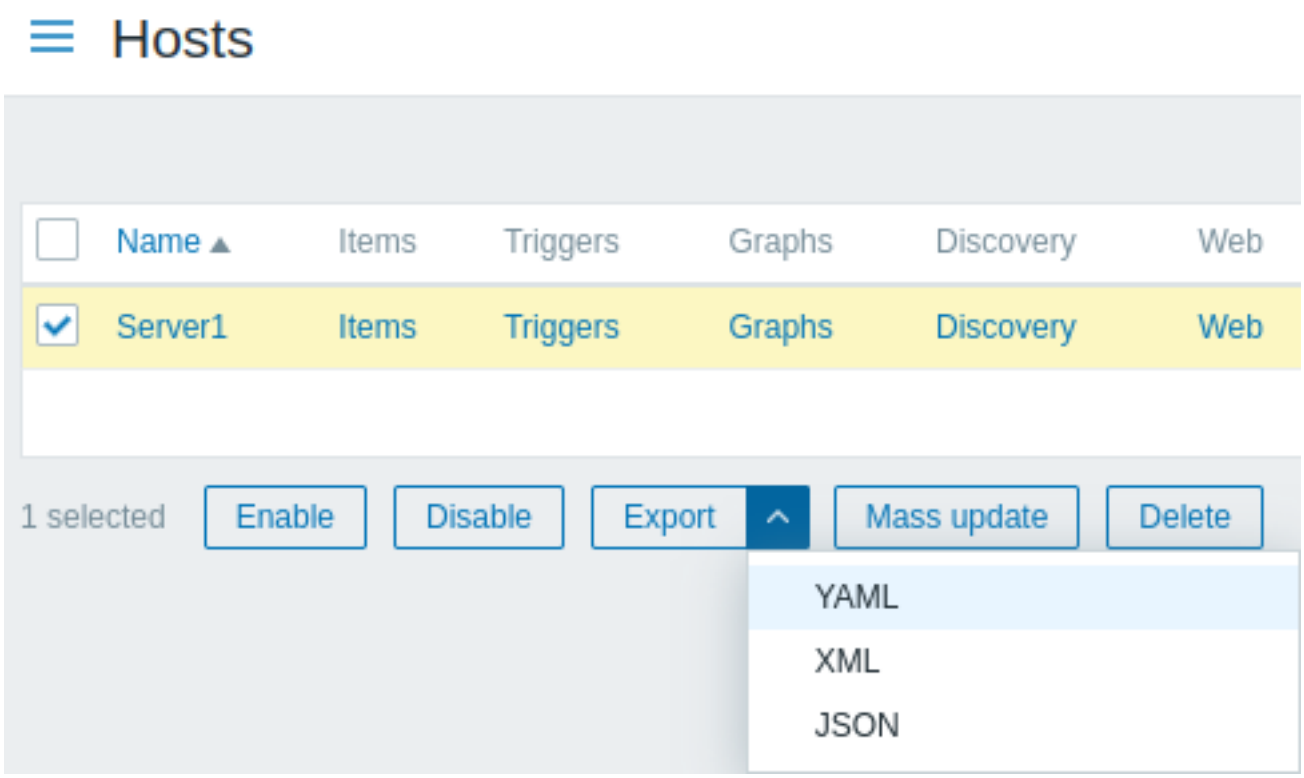
- 关联的主机群组
- 主机数据
- 模板关联
- 主机群组关联
- 主机接口
- 直接关联的监控项
- 直接关联的触发器
- 直接关联的发现规则及其所有原型

- 直接关联的 web 场景
- 主机宏
- 主机清单数据
- 值映射
- 关联的图形

导出

要导出主机，请执行以下操作：

1. 转到 *Data collection* > 主机。
2. 勾选要导出的主机对应的复选框。
3. 单击列表下方的 *Export*，以默认 YAML 格式导出，或单击箭头选择其他导出格式。



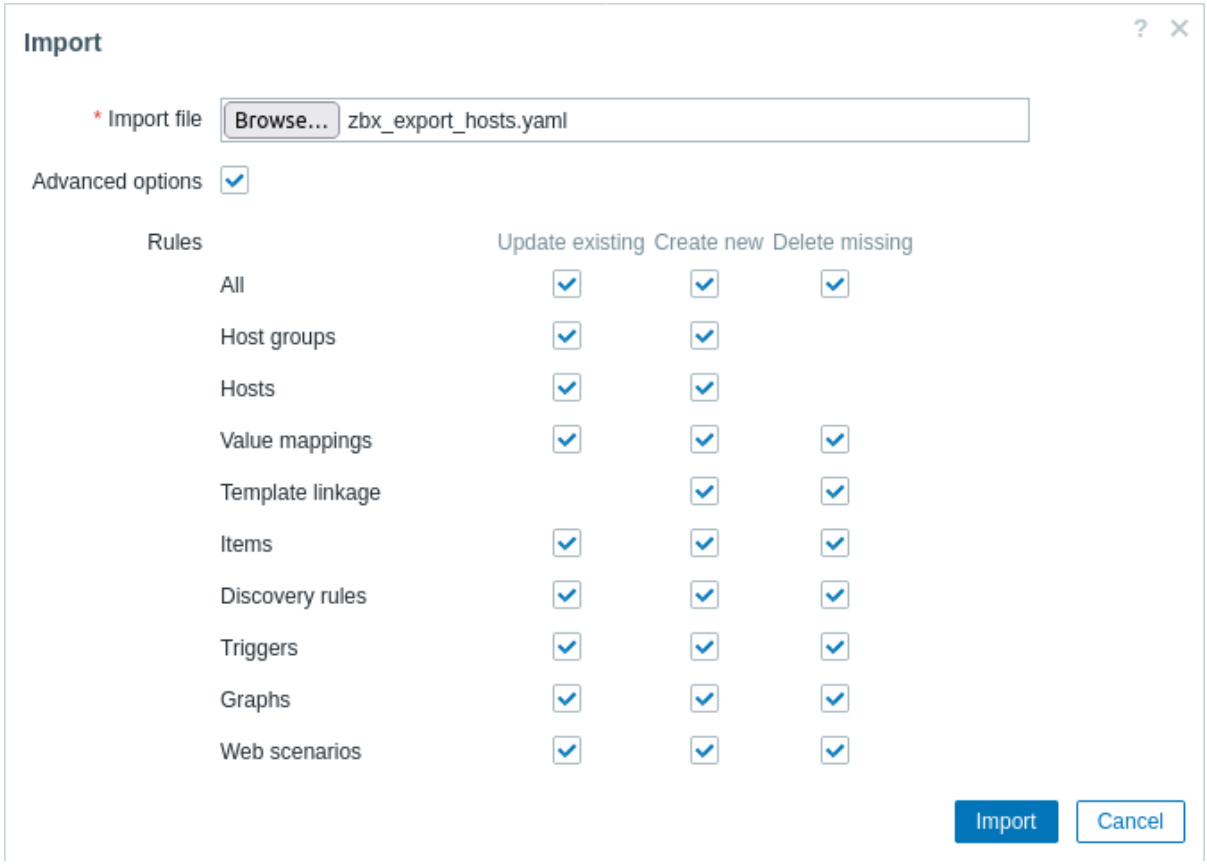
根据所选格式，主机会导出到本地文件，默认文件名如下：

- `zabbix_export_hosts.yaml` - YAML 导出；
- `zabbix_export_hosts.xml` - XML 导出；
- `zabbix_export_hosts.json` - JSON 导出。

导入

要导入主机，请执行以下操作：

1. 转到 数据采集 > 主机。
2. 点击右上角的 导入。
3. 选择导入文件。
4. 勾选要应用的导入规则（见下文）。
5. 点击配置表单右下角的 导入。



如果勾选 高级选项，将显示所有可导入元素的详细列表 - 根据需要勾选或取消勾选每条导入规则。

如果点击 全部行中的复选框，其下的所有元素都将被勾选或取消勾选。

导入规则：

规则	描述
更新现有项	将使用导入文件中的数据更新现有元素。否则，不会更新这些元素。
创建新项	将使用导入文件中的数据创建新元素。否则，不会创建这些元素。
删除缺失项	将删除导入文件中不存在的现有元素。否则，不会删除这些元素。 如果为 模板链接勾选了 删除缺失项，则会取消链接导入文件中不存在的当前模板链接。不会删除从已取消链接的模板继承的实体（监控项、触发器、图形等），除非同时为每个实体选择了 删除缺失项选项。

导入期间无法查看主机的更改。导入成功或失败的消息将显示在前端中。

导出格式

当导出主机时，Zabbix 会生成结构化的 YAML、JSON 或 XML 格式。导出内容包括主机元素，例如主机元数据、监控项、宏、触发器等。

每个元素都有特定用途，并且可能包含嵌套元素。

以下各节将介绍导出格式中的每个元素。示例使用带有 MySQL by Zabbix agent 2 模板的主机。为便于演示元素，在创建主机后会取消链接该模板。取消链接会保留配置中的所有元素（不同于取消链接并清除，后者会移除这些元素）。

省略号（...）表示为简洁起见省略的元素。注释（see table below）用于在后续章节中对该元素进行更详细说明时。

```
zabbix_export:
  version: '8.0'
  host_groups:
    - uuid: 748ad4d098d447d492bb935c907f652f
      name: Databases
  hosts: （见下表）
  graphs: （见下表）
```

主机元数据

Element		Type	Description
version		string	(必需) Zabbix 版本。
host_groups			(必需) 主机组的根元素。
	uuid	string	(必需) 此主机组的唯一标识符。
	name	string	(必需) 主机组名称。
hosts			主机 的根元素。
graphs			主机图形 的根元素。

主机

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      name: 'MySQL server'
      groups:
        - name: Databases
      interfaces: (see table below)
      items: (see table below)
      discovery_rules: (see table below)
      tags:
        - tag: location
          value: Riga
      macros:
        (...)
        - macro: '{$MYSQL.DSN}'
          value: 192.0.2.1
        (...)
        - macro: '{$MYSQL.PASSWORD}'
          type: SECRET_TEXT
          description: 'MySQL user password.'
        (...)
      valuemaps: (see table below)
  (...)
```

Element	Type	Description
host	string	(必填) 唯一的主机名称。
name	string	可见的主机名称。
description	text	主机描述。
monitored_by	string	主机的监控方式。 可能的值:[1] SERVER (0，默认)、PROXY (1) 或 PROXY_GROUP (2)。
proxy		proxy 的根元素。
name	string	(必填) 用于监控主机的 proxy 名称（如果有）。
proxy_group		proxy 组的根元素。
name	string	(必填) 用于监控主机的 proxy 组名称（如果有）。
status	string	主机状态。 可能的值:[1] ENABLED (0，默认)、DISABLED (1)。
ipmi_authtype	string	IPMI 会话认证类型。 可能的值:[1] DEFAULT (-1，默认)、NONE (0)、MD2 (1)、MD5 (2)、STRAIGHT (4)、OEM (5)、RMCP_PLUS (6)。
ipmi_privilege	string	IPMI 会话权限级别。 可能的值:[1] CALLBACK (1)、USER (2，默认)、OPERATOR (3)、ADMIN (4)、OEM (5)。
ipmi_username	string	IPMI 检查的用户名。
ipmi_password	string	IPMI 检查的密码。
templates		已关联模板的根元素。
name	string	(必填) 模板名称。
groups		主机所属主机组的根元素。
name	string	(必填) 主机组名称。
interfaces		主机接口 的根元素。
items		主机监控项 的根元素。
discovery_rules		主机低级别发现规则 的根元素。

Element	Type	Description
http_tests		主机 web 场景 的根元素。
tags		主机标签的根元素。
tag	string	(必填) 标签名称。
value	string	标签值。
macros		主机宏的根元素。
macro	string	(必填) 用户宏名称。
type	string	用户宏类型。 可能的值:[1] TEXT (0，默认)、SECRET_TEXT (1)、VAULT (2)。
value	string	用户宏值。
description	string	用户宏描述。
inventory		主机清单的根元素。
<inventory_property>	string	清单属性。所有属性都有各自对应的元素（type、name、os 等；例如，参见导出格式）。
inventory_mode	string	清单模式。 可能的值:[1] DISABLED (-1)、MANUAL (0，默认)、AUTOMATIC (1)。
valuemaps		主机值映射 的根元素。

Note:
另请参见：Host object（请参考名称匹配的相关属性）。

主机接口

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      (...)
      interfaces:
        - ip: 192.0.2.1
          interface_ref: if1
      (...)
```

Element	Type	Description
default	string	是否为主机的主接口。请注意，同一主机上同一类型只能有一个主接口。 可能值:[1] NO (0), YES (1, default).
type	string	接口类型。 可能值:[1] ZABBIX (1, default), SNMP (2), IPMI (3), JMX (4).
useip	string	是否使用 IP 作为连接主机的接口（否则将使用 DNS）。 可能值:[1] NO (0), YES (1, default).
ip	string	(IP 连接必填）IP 地址（IPv4 或 IPv6）。
dns	string	(DNS 连接必填）DNS 名称。
port	string	端口号。
details		接口详细信息的根元素。
version	string	使用此 SNMP 版本。 可能值:[1] SNMPV1 (1), SNMP_V2C (2, default), SNMP_V3 (3).
community	string	(SNMPv1 和 SNMPv2 监控项必填）SNMP community。
max_repetitions	string	原生 SNMP 批量请求（GetBulkRequest-PDUs）的最大重复值。 支持 SNMPv2 和 SNMPv3 监控项（discovery[] 和 walk[] 监控项）。 默认值: 10。
contextname	string	SNMPv3 上下文名称。 支持 SNMPv3 监控项。
securityname	string	SNMPv3 安全名称。 支持 SNMPv3 监控项。
securitylevel	string	SNMPv3 安全级别。 支持 SNMPv3 监控项。 可能值:[1] NOAUTHNOPRIV (0, default), AUTHNOPRIV (1), AUTHPRIV (2).
authprotocol	string	SNMPv3 身份验证协议。 支持 SNMPv3 监控项。 可能值:[1] MD5 (0, default), SHA1 (1), SHA224 (2), SHA256 (3), SHA384 (4), SHA512 (5).
authpassphrase	string	SNMPv3 身份验证口令。 支持 SNMPv3 监控项。

Element	Type	Description
privprotocol	string	SNMPv3 隐私协议。 支持 SNMPv3 监控项。 可能值:[1] DES (0, default), AES128 (1), AES192 (2), AES256 (3), AES192C (4), AES256C (5).
privpassphrase	string	SNMPv3 隐私口令。 支持 SNMPv3 监控项。
bulk	string	对 SNMP 使用批量请求。 可能值:[1] NO (0), YES (1, default).
interface_ref	string	在监控项中使用的接口引用名称（格式: `if<N>`）。

Note:
另请参见: Host interface object（请参阅名称匹配的相关属性）。

主机监控项

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      (...)
      items:
        (...)
        - name: 'Binlog cache disk use'
          type: DEPENDENT
          key: mysql.binlog_cache_disk_use
          value_type: FLOAT
          description: '由于事务大小超过 `binlog_cache_size`，无法放入常规二进制日志缓存，因此使用临时磁盘缓存的事务数量。'
          preprocessing: （见下表）
          master_item:
            key: 'mysql.get_status_variables["{$MYSQL.DSN}","{$MYSQL.USER}","{$MYSQL.PASSWORD}"]'
          tags:
            - tag: component
              value: cache
        (...)
        - name: 'Buffer pool utilization'
          type: CALCULATED
          key: mysql.buffer_pool_utilization
          value_type: FLOAT
          units: '%'
          params: |
            ( last(//mysql.innodb_buffer_pool_pages_total) -
            last(//mysql.innodb_buffer_pool_pages_free) ) /
            ( last(//mysql.innodb_buffer_pool_pages_total) +
            ( last(//mysql.innodb_buffer_pool_pages_total) = 0 ) ) * 100 *
            ( last(//mysql.innodb_buffer_pool_pages_total) > 0 )
          description: '缓冲池中已使用页数与总页数的比率。'
          tags:
            - tag: component
              value: memory
          triggers: （见下表）
        (...)
        - name: Uptime
          type: DEPENDENT
          key: mysql.uptime
          units: uptime
          description: '服务器已运行的秒数。'
          preprocessing: （见下表）
          master_item:
            key: 'mysql.get_status_variables["{$MYSQL.DSN}","{$MYSQL.USER}","{$MYSQL.PASSWORD}"]'
          tags:
            - tag: component
```

```
            value: application
        triggers: (见下表)
    (...)
```

Element	Type	Description
name	string	(必需) 监控项名称。
type	string	监控项类型。 可能的值：[1] ZABBIX_PASSIVE (0，默认)、TRAP (2)、SIMPLE (3)、INTERNAL (5)、ZABBIX_ACTIVE (7)、EXTERNAL (10)、ODBC (11)、IPMI (12)、SSH (13)、TELNET (14)、CALCULATED (15)、JMX (16)、SNMP_TRAP (17)、DEPENDENT (18)、HTTP_AGENT (19)、SNMP_AGENT (20)、ITEM_TYPE_SCRIPT (21)、ITEM_TYPE_BROWSER (22)、ITEM_TYPE_NESTED (23)、TELEMETRY_QUERY (24)。
snmp_oid	string	(SNMP_AGENT 监控项必需) SNMP 对象 ID。
key	string	(必需) 监控项键值。
delay	string	监控项的更新间隔。 默认值：1m。对于 TRAP 监控项，该值始终为 0。
history	string	历史数据应保留多长时间（使用时间后缀、用户宏或LLD 宏）。 默认值：31d。
trends	string	趋势数据应保留多长时间（使用时间后缀、用户宏或LLD 宏）。 默认值：365d。
status	string	监控项状态。 可能的值：[1] ENABLED (0，默认)、DISABLED (1)。
value_type	string	接收值类型。 可能的值：[1] FLOAT (0)、CHAR (1)、LOG (2)、UNSIGNED (3，默认)、TEXT (4)、BINARY (5)、JSON (6)。
allowed_hosts	string	允许为该监控项发送数据的主机 IP 地址列表，地址之间使用逗号分隔。 支持 TRAP 和 HTTP_AGENT 监控项。
units	string	接收值的单位（bps、B 等）。
params	text	根据监控项类型而定的其他参数（SSH 和 TELNET 监控项执行的脚本；ODBC 监控项的 SQL 查询；CALCULATED 监控项的公式；ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项的脚本）。
ipmi_sensor	string	IPMI 传感器。 支持 IPMI 监控项。
authtype	string	身份验证类型。 支持 SSH 和 HTTP_AGENT 监控项。 SSH 监控项的可能值：[1] PASSWORD (0，默认)、PUBLIC_KEY (1)。 HTTP_AGENT 监控项的可能值：[1] NONE (0，默认)、BASIC (1)、NTLM (2)、Kerberos (3) 或 Digest (4)。
username	string	(SSH 和 TELNET 监控项必需) 用于身份验证的用户名。 支持 SIMPLE、ODBC、JMX 和 HTTP_AGENT 监控项。 用于 JMX 监控项时，还应指定 `password`（见下文），或者将这两个元素都留空。
password	string	(SSH 和 TELNET 监控项必需) 用于身份验证的密码。 支持 SIMPLE、ODBC、JMX 和 HTTP_AGENT 监控项。 用于 JMX 监控项时，还应指定 `username`（见上文），或者将这两个元素都留空。
publickey	string	(SSH 监控项必需) 公钥文件名称。
privatekey	string	(SSH 监控项必需) 私钥文件名称。
description	text	监控项描述。
inventory_link	string	由监控项填充的主机资产清单字段。 可能的值：[1] NONE (0)、ALIAS (4) 等（支持的字段请参见主机资产清单）。
valuemap		监控项值映射的根元素。
name	string	(必需) 用于该监控项的值映射名称。
logtimefmt	string	日志条目中的时间格式。 支持 LOG 值类型的监控项。
preprocessing		监控项值预处理的根元素。
step		主机监控项值预处理步骤的根元素。
interface_ref	string	对主机接口的引用（格式：`if<N>`）。
jmx_endpoint	string	JMX 端点。 支持 JMX 监控项。
master_item		(DEPENDENT 监控项必需) 依赖监控项的主监控项根元素。
key	string	(必需) 依赖监控项的主监控项键值。
timeout	string	监控项数据轮询请求超时时间。 支持监控项类型的超时列表。

Element	Type	Description
url	string	(HTTP_AGENT 监控项必需) URL 字符串。
query_fields		查询参数的根元素。 支持 HTTP_AGENT 监控项。
name	string	(HTTP_AGENT 监控项必需) 查询参数名称。
value	string	查询参数值。 支持 HTTP_AGENT 监控项。
parameters		用户定义参数的根元素。 支持 ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项。
name	string	(ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项必需) 用户定义参数名称。
value	string	用户定义参数值。 支持 ITEM_TYPE_SCRIPT 和 ITEM_TYPE_BROWSER 监控项。
posts	string	HTTP(S) 请求正文数据。 支持 HTTP_AGENT 监控项。
status_codes	string	所需 HTTP 状态码的范围，使用逗号分隔。 支持 HTTP_AGENT 监控项。
follow_redirects	string	轮询数据时跟随响应重定向。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO (0)、YES (1，默认)。
post_type	string	POST 数据正文的类型。 支持 HTTP_AGENT 监控项。 可能的值：[1] RAW (0，默认)、JSON (2)、XML (3)。
http_proxy	string	HTTP(S) proxy 连接字符串。 支持 HTTP_AGENT 监控项。
headers		HTTP(S) 请求标头的根元素。 支持 HTTP_AGENT 监控项。
name	string	(HTTP_AGENT 监控项必需) 标头名称。
value	string	(HTTP_AGENT 监控项必需) 标头值。
retrieve_mode	string	应存储响应的哪一部分。 支持 HTTP_AGENT 监控项。 可能的值：[1] BODY (0，默认)、HEADERS (1)、BOTH (2)。
request_method	string	请求方法类型。 支持 HTTP_AGENT 监控项。 可能的值：[1] GET (0，默认)、POST (1)、PUT (2)、HEAD (3)。
output_format	string	如何处理响应。 支持 HTTP_AGENT 监控项。 可能的值：[1] RAW (0，默认)、JSON (1)。
allow_traps	string	允许以与 trapper 监控项相同的方式填充值。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO (0，默认)、YES (1)。
ssl_cert_file	string	SSL 公钥文件路径。 支持 HTTP_AGENT 监控项。
ssl_key_file	string	SSL 私钥文件路径。 支持 HTTP_AGENT 监控项。
ssl_key_password	string	SSL 密钥文件密码。 支持 HTTP_AGENT 监控项。
verify_peer	string	是否验证主机证书的真实性。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO (0，默认)、YES (1)。
verify_host	string	是否验证连接的主机名与主机证书中的主机名匹配。 支持 HTTP_AGENT 监控项。 可能的值：[1] NO (0，默认)、YES (1)。
time_shift	string	监控项查询的时间范围结束点相对于当前时间向前偏移的时长。 支持 TELEMETRY_QUERY 监控项。
lookback_limit	string	在没有可继续使用的上一次查询（例如监控项首次运行时），或上一次查询早于 `delay`（例如由于 Zabbix 服务器停机）时，监控项向前查询数据的时间范围。 支持 TELEMETRY_QUERY 监控项。
granularity	string	查询完整时间范围拆分后每个片段的长度。 支持 TELEMETRY_QUERY 监控项。
query		(TELEMETRY_QUERY 监控项必需) 遥测查询的根元素。定义监控项从数据源查询的内容、筛选数据的方式以及聚合结果的方式。

Element	Type	Description
signal_type	string	(必需) 监控项查询的信号类型。 支持 TELEMETRY_QUERY 监控项。 可能的值：[1] TRACES (0)、METRICS (1)、LOGS (2)。
metric_point_type	string	(METRICS 信号类型必需) 监控项查询的指标点类型。 支持 TELEMETRY_QUERY 监控项。 可能的值：[1] SUM (0)、GAUGE (1)、HISTOGRAM (2)、EXPONENTIAL_HISTOGRAM (3)。
columns		(必需) 监控项查询的遥测数据列的根元素。 此元素包含以下元素： `column` (`string`) - (必需) 监控项查询的遥测数据列名称。 `attribute_key` (`string`) - (存储键值对的列必需) 监控项在存储键值对的列中查询的键名称。
aggregated_columns		(必需) 监控项查询的遥测数据聚合列的根元素。 此元素包含以下元素： `column` (`string`) - (MIN、MAX、AVG、SUM 和 PERCENTILE 函数必需) 监控项查询的遥测数据列名称。 `function` (`string`) - (必需) 监控项应用于列记录的函数。可能的值：[1] MIN (1)、MAX (2)、AVG (3)、COUNT (4)、SUM (5)、PERCENTILE (8)。 `parameters` (`array`) - (PERCENTILE 函数必需) 监控项根据列记录计算的百分位数。 `alias` (`string`) - (必需) 计算值的唯一名称，用于在监控项值中标识该值。
filter		(必需) 查询中列记录的遥测查询筛选器根元素。 此元素包含以下元素： `evaltype` (`string`) - (必需) 筛选条件评估方法。可能的值：[1] AND_OR (0)、AND (1)、OR (2)、FORMULA (3)。 `formula` (`string`) - (FORMULA 评估方法必需) 用于通过自定义表达式评估筛选条件的用户定义表达式。 `conditions` (`string`) - (必需) 遥测数据列中的记录必须满足的遥测数据筛选条件根元素。 `conditions` 元素包含以下元素： `column` (`string`) - (必需) 应用筛选条件的遥测数据列名称。 `attribute_key` (`string`) - (存储键值对的列必需) 存储键值对的列中应用条件的键名称。 `operator` (`string`) - (必需) 筛选条件运算符。单值列的可能值：[1] EQUAL (0)、NOT_EQUAL (1)、LIKE (2)、NOT_LIKE (3)。存储键值对的列的可能值：[1] EQUAL (0)、NOT_EQUAL (1)、EXISTS (12)。 `formulaid` (`string`) - (FORMULA 评估方法必需) 用于从自定义表达式引用条件的任意唯一 ID。 `value` (`string`) - (EQUAL、NOT_EQUAL、LIKE 和 NOT LIKE 操作必需) 用于比较的值。
tags		监控项标签的根元素。
tag	string	(必需) 标签名称。
value	string	标签值。
triggers		主机监控项触发器的根元素。

Note:
另请参见：监控项对象（参阅名称匹配的相关属性）。

主机监控项值预处理步骤

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      (...)
      items:
        (...)
        - name: 'Binlog cache disk use'
          (...)
          preprocessing:
            - type: JSONPATH
              parameters:
```

```
              - $.Binlog_cache_disk_use
            - type: DISCARD_UNCHANGED_HEARTBEAT
              parameters:
                - 6h
          (...)
```

Element	Type	Description
type	string	(必填) 监控项值预处理步骤类型。 可能的值：[1] MULTIPLIER (1), RTRIM (2), LTRIM (3), TRIM (4), REGEX (5), BOOL_TO_DECIMAL (6), OCTAL_TO_DECIMAL (7), HEX_TO_DECIMAL (8), SIMPLE_CHANGE (9, calculated: received value - previous value), CHANGE_PER_SECOND (10, calculated: (received value - previous value)/(time now - time of last check)), XMLPATH (11), JSONPATH (12), IN_RANGE (13), MATCHES_REGEX (14), NOT_MATCHES_REGEX (15), CHECK_JSON_ERROR (16), CHECK_XML_ERROR (17), CHECK_REGEX_ERROR (18), DISCARD_UNCHANGED (19), DISCARD_UNCHANGED_HEARTBEAT (20), JAVASCRIPT (21), PROMETHEUS_PATTERN (22), PROMETHEUS_TO_JSON (23), CSV_TO_JSON (24), STR_REPLACE (25), CHECK_NOT_SUPPORTED (26), XML_TO_JSON (27), SNMP_WALK_VALUE (28), SNMP_WALK_TO_JSON (29), SNMP_GET_VALUE (30).
parameters		(必填) 监控项值预处理步骤参数的根元素。
parameter	string	监控项值预处理步骤的单个参数。
error_handler	string	预处理步骤失败时使用的操作类型。 可能的值：[1] ORIGINAL_ERROR (0, default), DISCARD_VALUE (1), CUSTOM_VALUE (2), CUSTOM_ERROR (3).
error_handler_params	string	错误处理程序参数。

Note:
另请参见：监控项预处理对象（请参阅名称匹配的相关属性）。

主机监控项触发器

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      (...)
      items:
        (...)
        - name: Uptime
          (...)
          triggers:
            - expression: 'nodata(/MySQL server/mysql.uptime,30m)=1'
              name: 'MySQL: Failed to fetch info data'
              event_name: 'MySQL: Failed to fetch info data (or no data for 30m)'
              priority: INFO
              description: 'Zabbix 在过去 30 分钟内未收到任何监控项的数据。'
              dependencies:
                - name: 'MySQL: Service is down'
                  expression: 'last(/MySQL server/mysql.ping["{$MYSQL.DSN}","{$MYSQL.USER}","{$MYSQL.PASSW
              tags:
                - tag: scope
                  value: availability
            - expression: 'last(/MySQL server/mysql.uptime)<10m'
              name: 'MySQL: Service has been restarted'
              event_name: 'MySQL: Service has been restarted (uptime < 10m)'
              priority: INFO
              description: 'MySQL 运行时间少于 10 分钟。'
              tags:
                - tag: scope
                  value: notice
        (...)
```

Element	Type	Description
uuid	string	(必填) 此触发器的唯一标识符。
expression	string	(必填) 触发器表达式。
recovery_mode	string	生成 OK 事件的依据。 可能的值:[1] EXPRESSION (0, 默认), RECOVERY_EXPRESSION (1), NONE (2).
recovery_expression	string	触发器恢复表达式。
correlation_mode	string	关联模式（无事件关联或按标签进行事件关联）。 可能的值:[1] DISABLED (0, 默认), TAG_VALUE (1).
correlation_tag	string	用于事件关联的标签名称。
name	string	(必填) 触发器名称。
event_name	string	事件名称。
opdata	string	运维数据。
url_name	string	与触发器关联的 URL 标签。
url	string	与触发器关联的 URL。
status	string	触发器状态。 可能的值:[1] ENABLED (0, 默认), DISABLED (1).
priority	string	触发器严重性。 可能的值:[1] NOT_CLASSIFIED (0, 默认), INFO (1), WARNING (2), AVERAGE (3), HIGH (4), DISASTER (5).
description	text	触发器描述。
type	string	事件生成类型（单个问题事件或多个问题事件）。 可能的值:[1] SINGLE (0, 默认), MULTIPLE (1).
manual_close	string	手动关闭问题事件。 可能的值:[1] NO (0, 默认), YES (1).
dependencies		依赖项的根元素。
name	string	(必填) 依赖触发器名称。
expression	string	(必填) 依赖触发器表达式。
recovery_expression	string	依赖触发器恢复表达式。
tags		触发器标签的根元素。
tag	string	(必填) 标签名称。
value	string	标签值。

Note:
另请参见: Trigger object（请参阅名称匹配的相应属性）。

主机低级别发现规则

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      (...)
      discovery_rules:
        - name: 'Database discovery'
          key: 'mysql.db.discovery["{$MYSQL.DSN}","{$MYSQL.USER}","{$MYSQL.PASSWORD}"]'
          delay: 1h
          filter: (see table below)
          description: 'Scanning databases in DBMS.'
          interface_ref: if1
          item_prototypes:
            - name: 'Size of database {#DATABASE}'
              key: 'mysql.db.size["{$MYSQL.DSN}","{$MYSQL.USER}","{$MYSQL.PASSWORD}","{#DATABASE}"]'
              delay: 5m
              units: B
              description: 'Database size.'
              preprocessing:
                - type: DISCARD_UNCHANGED_HEARTBEAT
                  parameters:
                    - 1h
              interface_ref: if1
              tags:
                - tag: component
```

```
                value: storage
              - tag: database
                value: '{#DATABASE}'
        lld_macro_paths:
          - lld_macro: '{#DATABASE}'
            path: $.Database
        preprocessing:
          - type: DISCARD_UNCHANGED_HEARTBEAT
            parameters:
              - 1d
      (...)
```

Attention:
大多数主机低级别发现规则元素与主机监控项 相同。下表描述了与主机监控项不同的元素。

Element	Type	Description
type	string	监控项类型。 可选值：[1] ZABBIX_PASSIVE (0, default), TRAP (2), SIMPLE (3), INTERNAL (5), ZABBIX_ACTIVE (7), EXTERNAL (10), ODBC (11), IPMI (12), SSH (13), TELNET (14), JMX (16), DEPENDENT (18), HTTP_AGENT (19), SNMP_AGENT (20), ITEM_TYPE_SCRIPT (21), ITEM_TYPE_BROWSER (22).
key	string	（必填）低级别发现规则键值。
filter		主机低级别发现规则过滤器 的根元素。
lifetime	string	时间周期（使用秒、时间后缀 或用户宏），超过该时间后，将删除不再发现的资源。 默认值：7d。
lifetime_type	string	删除丢失的 LLD 资源的场景。 可选值：DELETE_NEVER, DELETE_IMMEDIATELY, DELETE_AFTER。
enabled_lifetime	string	时间周期（使用秒、时间后缀 或用户宏），超过该时间后，将禁用不再发现的资源。
enabled_lifetime_type	string	禁用丢失的 LLD 资源的场景。 可选值：DISABLE_NEVER, DISABLE_IMMEDIATELY, DISABLE_AFTER。
item_prototypes		主机监控项原型元素的根元素，与主机监控项 相同。
trigger_prototypes		主机触发器原型元素的根元素，与主机监控项触发器 相同。
graph_prototypes		主机图形原型的根元素，与主机图形 相同。
host_prototypes		主机原型的根元素，与主机 相同。
parent_discovery_rule		低级别发现规则原型的父级低级别发现规则（或规则原型）的根元素。此属性表示它是一个 LLD 规则原型，且是所引用规则/规则原型的直接子级。
key	string	（必填）父级低级别发现规则（或规则原型）键值。
master_item	string	（DEPENDENT 规则必填）依赖规则主监控项的根元素。
lld_macro_paths		低级别发现规则宏路径的根元素。
lld_macro	string	（必填）低级别发现规则宏名称。
path	string	（必填）值选择器，该值将被分配给相应的宏。
preprocessing		低级别发现规则值预处理的根元素。
step		低级别发现规则值预处理步骤元素的根元素，与主机监控项值预处理步骤 相同，但可选值更少。另请参见：LLD 规则预处理对象。
overrides		低级别发现规则覆盖规则的根元素。
name	string	（必填）唯一的覆盖名称。
step	string	（必填）覆盖的唯一顺序号。
stop	string	如果匹配，则停止处理后续覆盖。
filter		低级别发现规则覆盖规则过滤器元素的根元素，与主机低级别发现规则过滤器 相同。
operations		主机低级别发现规则覆盖操作 的根元素。

Note:
另请参见：LLD 规则对象（请参考名称匹配的相关属性）。

主机低级别发现规则过滤器

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
```

```
    (...)
    discovery_rules:
      - name: 'Database discovery'
        (...)
        filter:
          evaltype: AND
          conditions:
            - macro: '{#DATABASE}'
              value: '{$MYSQL.DBNAME.MATCHES}'
            - macro: '{#DATABASE}'
              value: '{$MYSQL.DBNAME.NOT_MATCHES}'
              operator: NOT_MATCHES_REGEX
        (...)
```

Element	Type	Description
evaltype	string	覆盖过滤条件的求值方式。 可选值:[1] AND_OR (0, 默认), AND (1), OR (2), FORMULA (3).
formula	string	用于过滤条件的自定义计算公式。
conditions		过滤条件的根元素。
macro	string	(必填) 要执行检查的低级别发现宏名称。
value	string	要比较的值。
operator	string	条件运算符。 可选值:[1] MATCHES_REGEX (8, 默认), NOT_MATCHES_REGEX (9).
formulaid	string	(必填) 用于在自定义表达式中引用某个条件的任意唯一 ID。只能包含大写字母。修改过滤条件时必须由用户定义该 ID，但在之后再次请求时会重新生成。

Note:
另请参见: LLD 规则过滤器对象（请参阅名称匹配的相关属性）。

主机低级别发现规则覆盖操作

Element	Type	Description
operationobject	string	应用该操作的对象。 可能的值：[1] ITEM_PROTOTYPE (0), TRIGGER_PROTOTYPE (1), GRAPH_PROTOTYPE (2), HOST_PROTOTYPE (3), DISCOVERY_PROTOTYPE (4).
operator	string	覆盖条件运算符。 可能的值：[1] EQUAL (1), NOT_EQUAL (2), LIKE (3), NOT_LIKE (4), REGEXP (5), NOT_REGEXP (6).
value	string	用于覆盖条件运算符的正则表达式或字符串。
status	string	覆盖操作后对象的状态。
discover	string	对象是否作为发现结果被添加。
delay	string	覆盖操作后为监控项原型设置的更新间隔。
history	string	覆盖操作后为监控项原型设置的历史数据保存周期。
trends	string	覆盖操作后为监控项原型设置的趋势数据保存周期。
severity	string	覆盖操作后设置的触发器原型严重性。
tags		覆盖操作后为对象设置的标签根元素。
tag	string	(必填) 标签名称。
value	string	标签值。
templates		覆盖操作后链接到主机原型的模板根元素。
name	string	(必填) 模板名称。
inventory_mode	string	覆盖操作后设置的主机原型清单模式。

Note:
另请参见：LLD 规则覆盖操作对象（请参考名称匹配的相关属性）。

主机 web 场景

Element	Type	Description
uuid	string	(required) 此 web 场景的唯一标识符。
name	string	(required) web 场景名称。
delay	string> 执行 web 场景的频率（使用秒、time suffix 或user macro）。默认值：1m。	
attempts	integer	执行 web 场景步骤的尝试次数。 可选值：1-10（默认值：1）。
agent	string	客户端 agent。Zabbix 将伪装为所选浏览器。 当网站针对不同浏览器返回不同内容时，这很有用。 默认值：Zabbix。
http_proxy	string	web 场景使用的 proxy，格式为： `http://[username[:password]@]proxy.example.com[:port]`
variables		web 场景变量的根元素，这些变量可在场景步骤中使用。
name	string	(required) 变量名称。
value	text	(required) 变量值。
headers		执行请求时要发送的 HTTP 标头根元素。 标头应使用与 HTTP 协议中相同的语法列出。
name	string	(required) 标头名称。
value	text	(required) 标头值。
status	string	web 场景状态。 可选值：[1] ENABLED (0, default), DISABLED (1).
authentication	string	认证方法。 可选值：[1] NONE (0, default), BASIC (1), NTLM (2), Kerberos (3), or Digest (4).
http_user	string	用于 BASIC (HTTP)、NTLM、Kerberos 或 Digest 认证的用户名。
http_password	string	用于 BASIC (HTTP)、NTLM、Kerberos 或 Digest 认证的密码。
verify_peer	string	验证 web 服务器的 SSL 证书。 可选值：[1] NO (0, default), YES (1).
verify_host	string	验证 web 服务器证书的 *Common Name* 字段或 *Subject Alternate Name* 字段是否匹配。 可选值：[1] NO (0, default), YES (1).
ssl_cert_file	string	用于客户端认证的 SSL 证书文件名（必须为 PEM 格式）。
ssl_key_file	string	用于客户端认证的 SSL 私钥文件名（必须为 PEM 格式）。
ssl_key_password	string	SSL 私钥文件密码。
steps		主机 web 场景步骤 的根元素。（必填）
tags		web 场景标签的根元素。
tag	string	(required) 标签名称。
value	string	标签值。

Note:
另请参见：Web 场景对象（请参阅名称匹配的相关属性）。

主机 web 场景步骤

Element	Type	Description
name	string	(required) Web 场景步骤名称。
url	string	(required) 用于监控的 URL。
query_fields		查询参数的根元素（在执行请求时要添加到 URL 的 HTTP 字段数组）。
name	string	(required) 查询参数名称。
value	string	查询参数值。
posts		HTTP POST 变量的根元素（字符串（原始 post 数据）或 HTTP 字段数组（表单字段数据））。
name	string	(required) Post 字段名称。
value	string	(required) Post 字段值。

Element	Type	Description
variables		步骤级变量（宏）的根元素，这些变量应在此步骤之后应用。 如果变量值带有'regex:' 前缀，则会根据'regex:' 前缀后面的正则表达式模式，从此步骤返回的数据中提取其值
name	string	(required) 变量名称。
value	text	(required) 变量值。
headers		执行请求时要发送的 HTTP 标头的根元素。
name	string	(required) 标头名称。
value	text	(required) 标头值。
follow_redirects	string	跟随 HTTP 重定向。 可选值:[1] NO (0), YES (1, 默认)。
retrieve_mode	string	HTTP 响应获取模式。 可选值:[1] BODY (0, 默认), HEADERS (1), BOTH (2)。
timeout	string	步骤执行的超时时间（使用秒、时间后缀 或用户宏）。 默认值: 15s。
required	string	响应中必须存在的文本（如果为空则忽略）。
status_codes	string	以逗号分隔的可接受 HTTP 状态码列表（例如 200-201,210-299；如果为空则忽略）。

Note:
另请参见: Web 场景步骤对象（请参考名称匹配的相关属性）。

主机图形

```
zabbix_export:
  (...)
  graphs:
    - name: 'MySQL: Bandwidth'
      graph_items: (see table below)
    (...)
```

Element	Type	Description
uuid	string	图形的唯一标识符。
name	string	(必填) 图形名称。
width	integer	图形宽度，单位为像素。 用于预览以及饼图/展开图。 可能的值:[1] 20-65535（默认值: 900）。
height	integer	图形高度，单位为像素。 用于预览以及饼图/展开图。 可能的值:[1] 20-65535（默认值: 900）。
yaxismin	double	Y 轴最小值。 支持 Y 轴 FIXED 最小值。 默认值: 0。
yaxismax	double	Y 轴最大值。 支持 X 轴 FIXED 最小值。 默认值: 0。
show_work_period	string	高亮显示非工作时间。 支持 NORMAL 和 STACKED 图形。 可能的值:[1] NO (0), YES (1, 默认)。
show_triggers	string	将简单触发器值显示为一条线。 支持 NORMAL 和 STACKED 图形。 可能的值:[1] NO (0), YES (1, 默认)。
type	string	图形类型。 可能的值:[1] NORMAL (0, 默认), STACKED (1), PIE (2), EXPLODED (3)。
show_legend	string	显示图形图例。 可能的值:[1] NO (0), YES (1, 默认)。
show_3d	string	启用 3D 样式。 支持 NORMAL 和 STACKED 图形。 可能的值:[1] NO (0, 默认), YES (1)。
percent_left	double	显示左轴的百分位线。 支持 NORMAL 图形。 默认值: 0。

Element	Type	Description
percent_right	double	显示右轴的百分位线。 支持 NORMAL 图形。 默认值: 0。
ymin_type_1	string	Y 轴最小值。 支持 NORMAL 和 STACKED 图形。 可能的值:[1] CALCULATED (0, 默认), FIXED (1), ITEM (2)。
ymin_item_1		(如果 `ymin_type_1` 设置为 `ITEM`，则必填) 单个监控项详细信息的根元素。
host	string	(必填) 监控项主机。
key	string	(必填) 监控项键值。
ymax_type_1	string	Y 轴最大值。 支持 NORMAL 和 STACKED 图形。 可能的值:[1] CALCULATED (0, 默认), FIXED (1), ITEM (2)。
ymax_item_1		(如果 `ymax_type_1` 设置为 `ITEM`，则必填) 单个监控项详细信息的根元素。
host	string	(必填) 监控项主机。
key	string	(必填) 监控项键值。
graph_items		(必填) 主机图形项 的根元素。

Note:
另请参见: Graph object（请参阅名称匹配的相关属性）。

主机图形监控项

```
zabbix_export:
  (...)
  graphs:
    - name: 'MySQL: Bandwidth'
      graph_items:
        - drawtype: GRADIENT_LINE
          color: 199C0D
          item:
            host: 'MySQL server'
            key: mysql.bytes_received.rate
        - sortorder: '1'
          drawtype: GRADIENT_LINE
          color: F63100
          item:
            host: 'MySQL server'
            key: mysql.bytes_sent.rate
    (...)
```

Element	Type	Description
sortorder	integer	绘制顺序。数值越小，越先绘制。可用于将线条或区域绘制在另一条线或区域的后面（或前面）。
drawtype	string	图形监控项的绘制样式。 支持 NORMAL 图形。 可能的值:[1] SINGLE_LINE (0, default), FILLED_REGION (1), BOLD_LINE (2), DOTTED_LINE (3), DASHED_LINE (4), GRADIENT_LINE (5).
color	string	元素颜色（6 个字符，十六进制）。
yaxisside	string	绘制图形监控项 Y 轴刻度的一侧。 支持 NORMAL 和 STACKED 图形。 可能的值:[1] LEFT (0, default), RIGHT (1).
calc_fnc	string	当某个监控项存在多个值时，用于绘制的数据。 可能的值:[1] MIN (1), AVG (2, default), MAX (4), ALL (7; minimum, average, and maximum; supported for simple graphs), LAST (9, supported for pie/exploded graphs).
type	string	图形监控项类型。 可能的值:[1] SIMPLE (0, default), GRAPH_SUM (2; value of the item represents the whole pie; supported for pie/exploded graphs).
item		（必填）单个监控项。
host	string	（必填）监控项主机。

Element	Type	Description
key	string	（必填）监控项键值。

Note:
另请参见: Graph item object（请参阅名称匹配的相应属性）。

主机值映射

```
zabbix_export:
  (...)
  hosts:
    - host: 'MySQL server'
      (...)
      valuemaps:
        - name: Example value map
          mappings:
            - value: '1'
              newvalue: Example value
  (...)
```

Element	Type	Description
uuid	string	(required) 此值映射的唯一标识符。
name	string	(required) 值映射名称。
mapping		映射的根元素。
type	string	映射匹配类型。 可能的值：[1] EQUAL (0, default), GREATER_OR_EQUAL (2), LESS_OR_EQUAL (3), IN_RANGE (4), REGEXP (5), DEFAULT (6).
value	string	原始值。
newvalue	string	(required) 原始值映射到的值。

Note:
另请参见：Value map object（请参考具有匹配名称的相关属性）。

脚注 [1] 括号中的 API 整数值，例如 ENABLED (0)，仅供参考。有关更多信息，请参见表格条目中的链接 API 对象页面，或每个章节末尾的链接。

网络地图

概述

网络地图导出 包含：

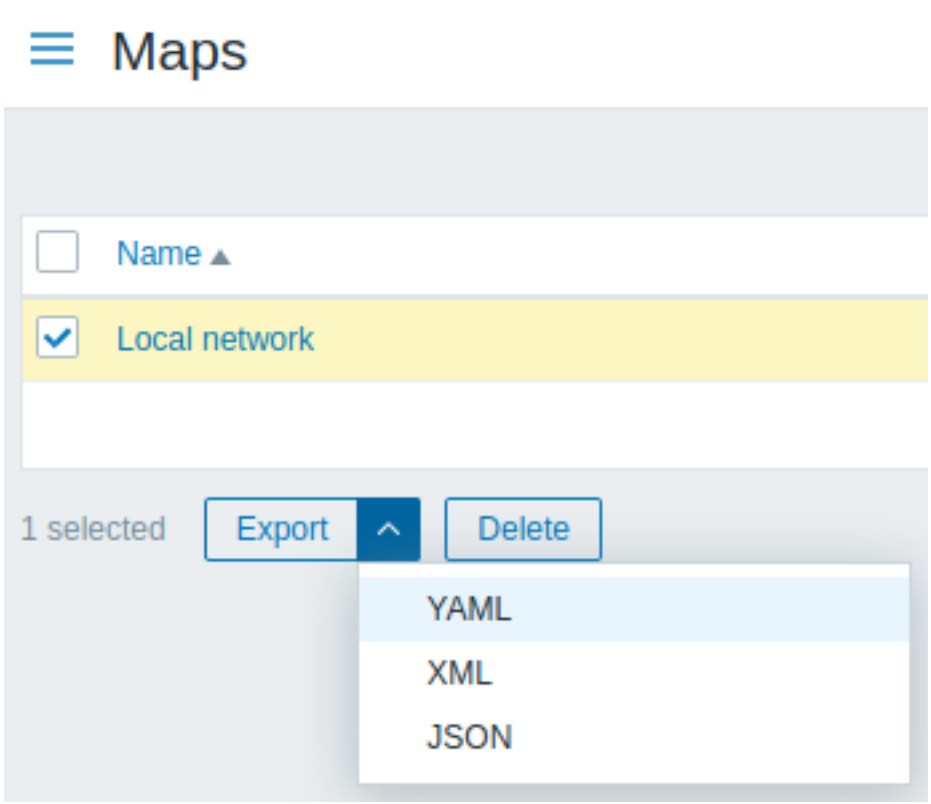
- 所有相关图像
- 地图结构（所有地图设置、所有包含的元素及其设置、地图链接和地图链接状态指示器）

Warning:
任何可能与已导出的地图相关的主机组、主机、触发器、其他地图或其他元素都不会被导出。因此，如果地图引用的元素中至少有一个缺失，导入将失败。

导出

要导出网络地图，请执行以下操作：

1. 转到 *Monitoring* > *Maps*。
2. 勾选要导出的网络地图复选框。
3. 单击列表下方的 *Export*，以默认 YAML 格式导出，或者单击箭头选择其他导出格式。



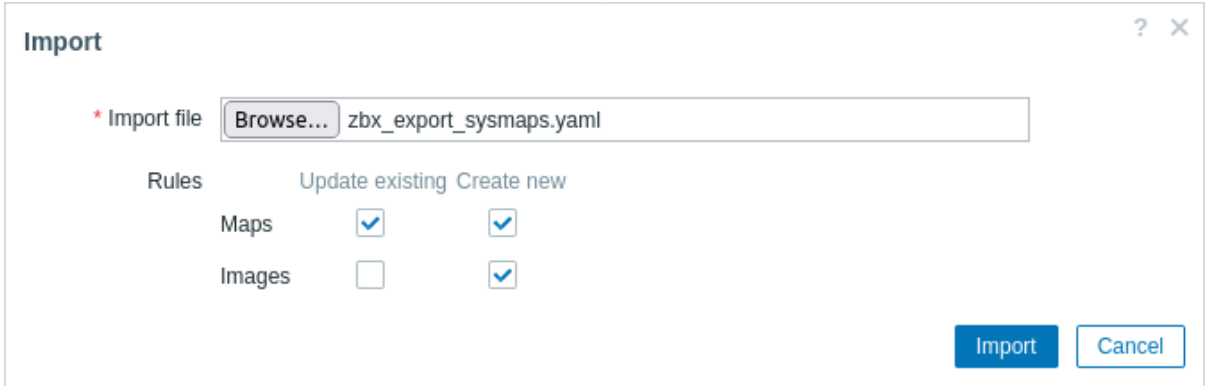
根据所选格式，地图会导出到本地文件，默认文件名如下：

- `zabbix_export_maps.yaml` - YAML 导出；
- `zabbix_export_maps.xml` - XML 导出；
- `zabbix_export_maps.json` - JSON 导出。

导入

要导入网络拓扑图，请执行以下操作：

1. 转到 监测 > 拓扑图。
2. 单击右上角的 导入。
3. 选择导入文件。
4. 勾选要应用的导入规则（见下文）。
5. 单击配置表单右下角的 导入。



导入规则：

规则	描述
更新现有项	将使用导入文件中的数据更新现有拓扑图。否则，不会更新这些拓扑图。
创建新项	将使用导入文件中的数据创建新的拓扑图。否则，不会创建这些拓扑图。

如果取消选中两个拓扑图选项，并选中图像对应的选项，则只会导入图像。只有 *Super admin* 用户可以导入图像。

在导入期间，您无法查看网络拓扑图的更改。导入成功或失败的消息将显示在前端中。

Warning:
如果替换现有图像，将影响所有使用该图像的拓扑图。

导出格式

导出为 YAML：

```yaml
zabbix_export:
  version: '8.0'
  images:
    - name: Zabbix_server_3D_(128)
      imagetype: '1'
      encodedImage: iVBOR...5CYII=
  maps:
    - name: 'Local network'
      width: '680'
      height: '200'
      label_type: '0'
      label_location: '0'
      highlight: '1'
      expandproblem: '1'
      markelements: '1'
      show_unack: '0'
      severity_min: '0'
      show_suppressed: '0'
      grid_size: '50'
      grid_show: '1'
      grid_align: '1'
      label_format: '0'
      label_type_host: '2'
      label_type_hostgroup: '2'
      label_type_trigger: '2'
      label_type_map: '2'
      label_type_image: '2'
      label_string_host: ''
      label_string_hostgroup: ''
      label_string_trigger: ''
      label_string_map: ''
      label_string_image: ''
      expand_macros: '1'
      background: {  }
      iconmap: {  }
      urls: {  }
      selements:
        - elementtype: '0'
          elements:
            - host: 'Zabbix server'
          label: |
            {HOST.NAME}
            {HOST.CONN}
          label_location: '0'
          x: '111'
          'y': '61'
          elementsubtype: '0'
          areatype: '0'
          width: '200'
          height: '200'
          viewtype: '0'
          use_iconmap: '0'
          selementid: '1'
          icon_off:
            name: Zabbix_server_3D_(128)
          icon_on: {  }
```

```
        icon_disabled: {  }
        icon_maintenance: {  }
        urls: {  }
        evaltype: '0'
    shapes:
      - type: '0'
        x: '0'
        'y': '0'
        width: '680'
        height: '15'
        text: '{MAP.NAME}'
        font: '9'
        font_size: '11'
        font_color: '000000'
        text_halign: '0'
        text_valign: '0'
        border_type: '0'
        border_width: '0'
        border_color: '000000'
        background_color: ''
        zindex: '0'
    lines: {  }
    links: {  }
```

导出的元素

下表说明了导出的元素。

Element		Type	Description
images			图像的根元素。
	name	string	唯一的图像名称。
	imagetype	integer	图像类型。 可能的值： 1 - 图像； 2 - 背景。
	encodedImage	string	Base64 编码的图像。
maps			maps 的根元素。

地图

Element	Type	Description
name	string	唯一的地图名称。
width	integer	地图宽度，单位为像素。
height	integer	地图高度，单位为像素。
label_type	integer	地图元素标签类型。 可能的值： 0 - 标签； 1 - 主机 IP 地址； 2 - 元素名称； 3 - 仅状态； 4 - 无。
label_location	integer	默认的地图元素标签位置。 可能的值： 0 - 底部； 1 - 左侧； 2 - 右侧； 3 - 顶部。
highlight	integer	为活动触发器和主机状态启用图标高亮显示。 可能的值： 0 - 否； 1 - 是。

Element	Type	Description
expandproblem	integer	为具有单个问题的元素显示问题触发器。 可能的值： 0 - 否； 1 - 是。
markelements	integer	高亮显示最近更改状态的地图元素。 可能的值： 0 - 否； 1 - 是。
show_unack	integer	问题显示。 可能的值： 0 - 所有问题的数量； 1 - 未确认问题的数量； 2 - 已确认和未确认问题分别计数。
severity_min	integer	默认在地图上显示的最低触发器严重性。 可能的值： 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。
show_suppressed	integer	显示原本会因主机维护而被抑制（不显示）的问题。 可能的值： 0 - 否； 1 - 是。
grid_size	integer	地图网格单元大小，单位为像素。 当 `grid_show` 设置为 0 时支持。 可能的值：20、40、50、75 或 100。
grid_show	integer	在地图配置中显示网格。 可能的值： 0 - 是； 1 - 否。
grid_align	integer	在地图配置中自动对齐图标。 可能的值： 0 - 是； 1 - 否。
label_format	integer	使用高级标签配置。 可能的值： 0 - 否； 1 - 是。
label_type_host	integer	将标签显示为主机标签。 当 `label_format` 设置为 1 时支持。 可能的值： 0 - 标签； 1 - 主机 IP 地址； 2 - 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义标签。
label_type_hostgroup	integer	将标签显示为主机组标签。 当 `label_format` 设置为 1 时支持。 可能的值： 0 - 标签； 2 - 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义标签。

Element	Type	Description
label_type_trigger	integer	将标签显示为触发器标签。 当 `label_format` 设置为 1 时支持。 可能的值： 0 - 标签； 2 - 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义标签。
label_type_map	integer	将标签显示为地图标签。 当 `label_format` 设置为 1 时支持。 可能的值： 0 - 标签； 2 - 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义标签。
label_type_image	integer	将标签显示为图像标签。 当 `label_format` 设置为 1 时支持。 可能的值： 0 - 标签； 2 - 元素名称； 4 - 无； 5 - 自定义标签。
label_string_host	string	主机元素的自定义标签。 当 `label_type_host` 设置为 5 时支持。
label_string_hostgroup	string	主机组元素的自定义标签。 当 `label_type_hostgroup` 设置为 5 时支持。
label_string_trigger	string	触发器元素的自定义标签。 当 `label_type_trigger` 设置为 5 时支持。
label_string_map	string	地图元素的自定义标签。 当 `label_type_map` 设置为 5 时支持。
label_string_image	string	图像元素的自定义标签。 当 `label_type_image` 设置为 5 时支持。
expand_macros	integer	在地图配置的标签中展开宏。 可能的值： 0 - 否； 1 - 是。
background		背景图像的根元素（如果有）。 当 `imagetype` 设置为 2 时支持。
name	string	背景图像名称。
iconmap		图标映射的根元素（如果有）。
name	string	图标映射名称。
urls		地图或各个地图元素使用的 URL 根元素。
name	string	链接名称。
url	string	链接 URL。
elementtype	integer	链接所属的地图项类型。 可能的值： 0 - 主机； 1 - 地图； 2 - 触发器； 3 - 主机组； 4 - 图像。
selements		地图 selements 的根元素。
shapes		地图形状的根元素。
type	integer	形状类型。 可能的值： 0 - 矩形； 1 - 椭圆。
x	integer	形状的 X 坐标，单位为像素。
y	integer	形状的 Y 坐标，单位为像素。
width	integer	形状宽度。

Element		Type	Description
	height	integer	形状高度。
	text	string	形状内的文本。
	font	integer	文本字体样式。 可能的值： 0 - Georgia, serif； 1 - ”Palatino Linotype”, ”Book Antiqua”, Palatino, serif； 2 - ”Times New Roman”, Times, serif； 3 - Arial, Helvetica, sans-serif； 4 - ”Arial Black”, Gadget, sans-serif； 5 - ”Comic Sans MS”, cursive, sans-serif； 6 - Impact, Charcoal, sans-serif； 7 - ”Lucida Sans Unicode”, ”Lucida Grande”, sans-serif； 8 - Tahoma, Geneva, sans-serif； 9 - ”Trebuchet MS”, Helvetica, sans-serif； 10 - Verdana, Geneva, sans-serif； 11 - ”Courier New”, Courier, monospace； 12 - ”Lucida Console”, Monaco, monospace。
	font_size	integer	字体大小，单位为像素。
	font_color	string	以十六进制代码表示的字体颜色。
	text_halign	integer	文本水平对齐方式。 可能的值： 0 - 居中； 1 - 左对齐； 2 - 右对齐。
	text_valign	integer	文本垂直对齐方式。 可能的值： 0 - 中间； 1 - 顶部； 2 - 底部。
	border_type	integer	形状边框类型。 可能的值： 0 - 无； 1 - 粗线； 2 - 点线； 3 - 虚线。
	border_width	integer	边框宽度，单位为像素。
	border_color	string	以十六进制代码表示的边框颜色。
	background_color	string	以十六进制代码表示的背景（填充）颜色。
	zindex	integer	用于对所有形状和线条排序的值（z-index）。
lines			地图线条的根元素。
	x1	integer	线条点 1 的 X 坐标，单位为像素。
	y1	integer	线条点 1 的 Y 坐标，单位为像素。
	x2	integer	线条点 2 的 X 坐标，单位为像素。
	y2	integer	线条点 2 的 Y 坐标，单位为像素。
	line_type	integer	线条类型。 可能的值： 0 - 无； 1 - 粗线； 2 - 点线； 3 - 虚线。
	line_width	integer	线条宽度，单位为像素。
	line_color	string	以十六进制代码表示的线条颜色。
	zindex	integer	用于对所有形状和线条排序的值（z-index）。
links			地图元素之间链接的根元素。
	drawtype	integer	链接样式。 可能的值： 0 - 线条； 2 - 粗线； 3 - 点线； 4 - 虚线。
	color	string	链接颜色（6 个字符，十六进制）。

Element	Type	Description
label	string	链接标签。
selementid1	id	要连接的一个元素的 ID。
selementid2	id	要连接的另一个元素的 ID。
linktriggers		链接状态指示器 的根元素。

Note:
另请参见：地图对象（请参阅名称匹配的相应属性）。

地图元素

Element	Type	Description
elementtype	integer	地图元素类型。 可能的值： 0 - 主机； 1 - 地图； 2 - 触发器； 3 - 主机组； 4 - 图像。
elements		Zabbix 实体（主机、主机组、地图等）在地图上表示时的根元素。 所有实体都有各自对应的元素（host 等；例如，参见导出格式）。
label	string	图标标签。
label_location	integer	标签位置。 可能的值： -1 - 使用地图默认值； 0 - 底部； 1 - 左侧； 2 - 右侧； 3 - 顶部。
x	integer	X 轴上的位置。
y	integer	Y 轴上的位置。
elementsubtype	integer	元素子类型。 当 elementtype 设置为 3 时支持。 可能的值： 0 - 单个主机组； 1 - 所有主机组。
areatype	integer	区域大小。 当 elementtype 设置为 1 时支持。 可能的值： 0 - 与整个地图相同； 1 - 自定义大小。
width	integer	区域宽度。 当 areatype 设置为 1 时支持。
height	integer	区域高度。 当 areatype 设置为 1 时支持。
viewtype	integer	区域放置算法。 当 elementsubtype 设置为 1 时支持。 可能的值： 0 - 在区域内均匀放置。
use_iconmap	integer	为此元素使用图标映射。仅当在地图级别启用了图标映射时相关。 可能的值： 0 - 否； 1 - 是。
selementid	id	唯一的元素记录 ID。
icon_off		当元素处于 OK 状态时使用的图像根元素。
name	string	唯一的图像名称。
icon_on		当元素处于 Problem 状态时使用的图像根元素。
name	string	唯一的图像名称。
icon_disabled		当元素被禁用时使用的图像根元素。
name	string	唯一的图像名称。

Element	Type	Description
icon_maintenance		当元素处于维护中时使用的图像根元素。
name	string	唯一的图像名称。
urls		地图或各个地图元素使用的 URL 根元素。
name	string	链接名称。
url	string	链接 URL。
evaltype	integer	标签的评估类型。
tags		问题标签的根元素（用于主机和主机组元素）。 如果提供了标签，则地图上只会显示带有这些标签的问题。
tag	string	标签名称。
value	string	标签值。
operator	integer	运算符。

Note:
另请参见：地图元素对象（请参阅名称匹配的相应属性）。

地图链接状态指示器

Element	Type	Description
drawtype	integer	当触发器处于 “Problem” 状态时的链接样式。 可能的值： 0 - 直线； 2 - 粗线； 3 - 点线； 4 - 虚线。
color	string	当触发器处于 “Problem” 状态时的链接颜色（6 个字符，十六进制）。
trigger		用于指示链接状态的触发器根元素。
description	string	触发器名称。
expression	string	触发器表达式。
recovery_expression	string	触发器恢复表达式。

Note:
另请参见：Map link trigger object（请参阅名称匹配的相关属性）。

媒体类型

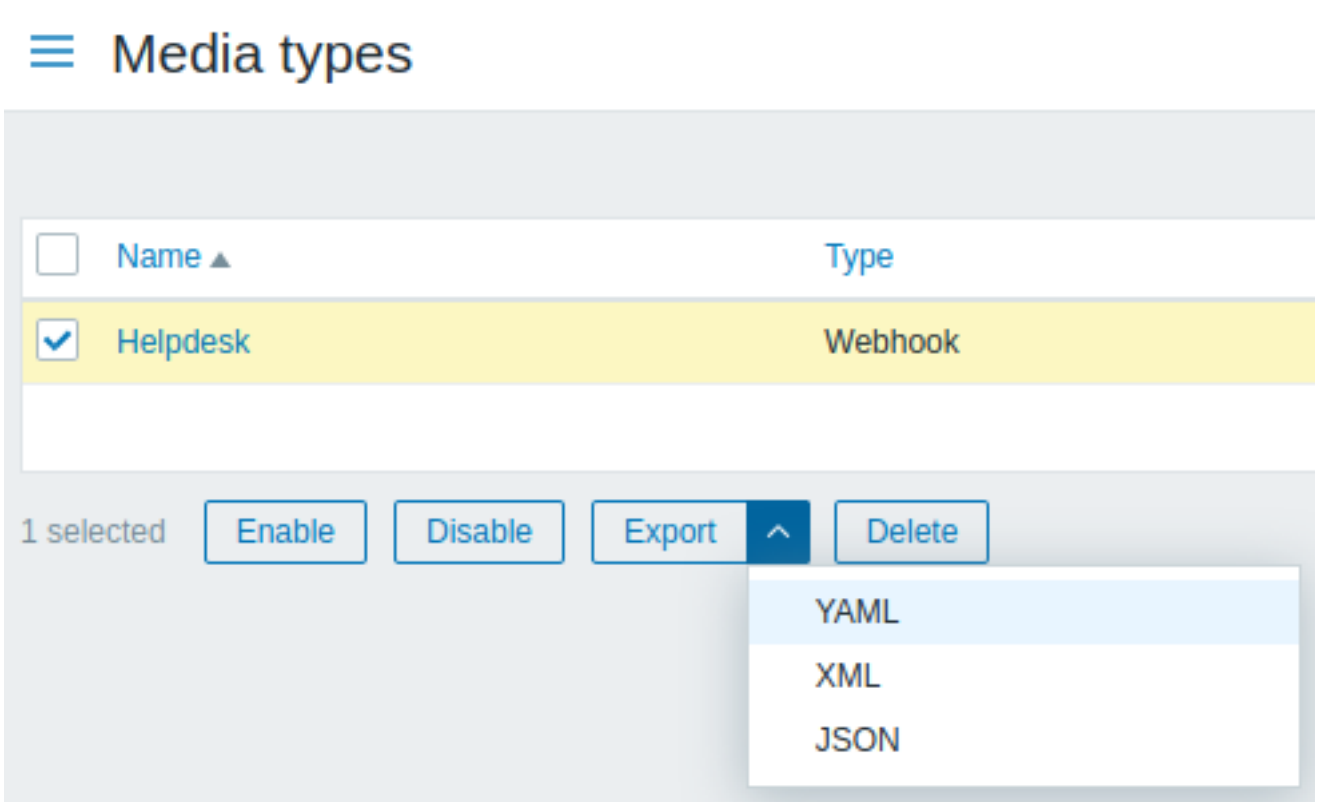
概述

媒体类型会与所有相关对象及对象关系一起导出。

导出

要导出媒体类型，请执行以下操作：

1. 转到 告警 > 媒体类型。
2. 勾选要导出的媒体类型复选框。
3. 单击列表下方的 导出以默认 YAML 格式导出，或单击箭头选择其他导出格式。



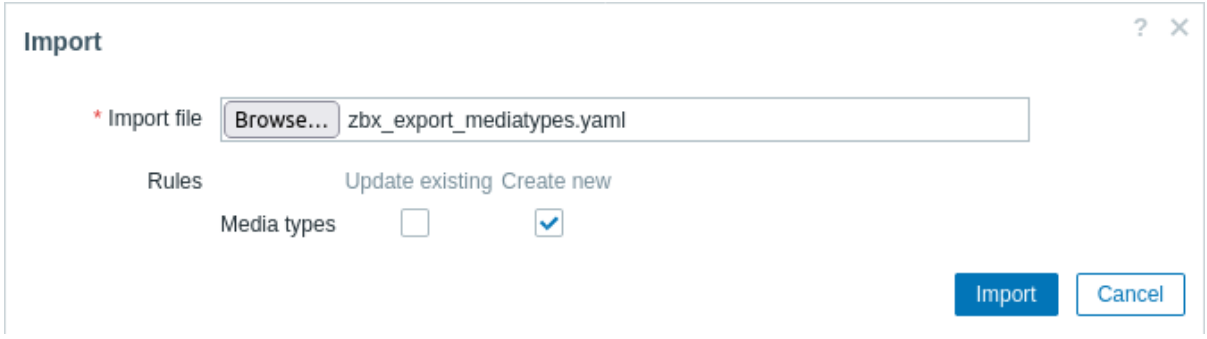
根据所选格式，媒体类型会导出到本地文件，并使用默认名称：

- `zabbix_export_mediatypes.yaml` - YAML 导出；
- `zabbix_export_mediatypes.xml` - XML 导出；
- `zabbix_export_mediatypes.json` - JSON 导出。

导入

要导入媒介类型，请执行以下操作：

1. 转到 告警 > 媒介类型。
2. 单击右上角的 导入。
3. 选择导入文件。
4. 勾选要应用的导入规则（见下文）。
5. 单击配置表单右下角的 导入。



导入规则：

规则	描述
更新现有项	勾选此复选框，使用导入文件中的数据更新现有媒介类型。
创建新项	勾选此复选框，使用导入文件中的数据创建新的媒介类型。如果导入文件包含已存在的媒介类型（例如 *Email*），则不会创建新的媒介类型，也不会更新现有媒介类型。

导入期间无法查看媒介类型的更改。导入成功或失败的消息将显示在前端中。

导出格式

导出为 YAML：

```yaml
zabbix_export:
  version: '8.0'
  media_types:
    - name: Pushover
      type: WEBHOOK
      parameters:
        - name: endpoint
          value: 'https://api.pushover.net/1/messages.json'
        - name: eventid
          value: '{EVENT.ID}'
        - name: event_nseverity
          value: '{EVENT.NSEVERITY}'
        - name: event_source
          value: '{EVENT.SOURCE}'
        - name: event_value
          value: '{EVENT.VALUE}'
        - name: expire
          value: '1200'
        - name: message
          value: '{ALERT.MESSAGE}'
        - name: priority_average
          value: '0'
        - name: priority_default
          value: '0'
        - name: priority_disaster
          value: '0'
        - name: priority_high
          value: '0'
        - name: priority_information
          value: '0'
        - name: priority_not_classified
          value: '0'
        - name: priority_warning
          value: '0'
        - name: retry
          value: '60'
        - name: title
          value: '{ALERT.SUBJECT}'
        - name: token
          value: '<PUSHOVER TOKEN HERE>'
        - name: triggerid
          value: '{TRIGGER.ID}'
        - name: url
          value: '{$ZABBIX.URL}'
        - name: url_title
          value: Zabbix
        - name: user
          value: '{ALERT.SENDTO}'
      status: DISABLED
      max_sessions: '0'
      script: |
        try {
            var params = JSON.parse(value),
                request = new HttpRequest(),
                data,
                response,
                severities = [
                    {name: 'not_classified', color: '#97AAB3'},
                    {name: 'information', color: '#7499FF'},
                    {name: 'warning', color: '#FFC859'},
                    {name: 'average', color: '#FFA059'},
                    {name: 'high', color: '#E97659'},
```

```
            {name: 'disaster', color: '#E45959'},
            {name: 'resolved', color: '#009900'},
            {name: 'default', color: '#000000'}
        ],
        priority;

    if (typeof params.HTTPProxy === 'string' && params.HTTPProxy.trim() !== '') {
        request.setProxy(params.HTTPProxy);
    }

    if ([0, 1, 2, 3].indexOf(parseInt(params.event_source)) === -1) {
        throw 'Incorrect "event_source" parameter given: "' + params.event_source + '".\nMust be 0
    }

    if (params.event_value !== '0' && params.event_value !== '1'
        && (params.event_source === '0' || params.event_source === '3')) {
        throw 'Incorrect "event_value" parameter given: ' + params.event_value + '\nMust be 0 or 1
    }

    if ([0, 1, 2, 3, 4, 5].indexOf(parseInt(params.event_nseverity)) === -1) {
        params.event_nseverity = '7';
    }

    if (params.event_value === '0') {
        params.event_nseverity = '6';
    }

    priority = params['priority_' + severities[params.event_nseverity].name] || params.priority_de

    if (isNaN(priority) || priority < -2 || priority > 2) {
        throw '"priority" should be -2..2';
    }

    if (params.event_source === '0' && isNaN(params.triggerid)) {
        throw 'field "triggerid" is not a number';
    }

    if (isNaN(params.eventid)) {
        throw 'field "eventid" is not a number';
    }

    if (typeof params.message !== 'string' || params.message.trim() === '') {
        throw 'field "message" cannot be empty';
    }

    data = {
        token: params.token,
        user: params.user,
        title: params.title,
        message: params.message,
        url: (params.event_source === '0')
            ? params.url + '/tr_events.php?triggerid=' + params.triggerid + '&eventid=' + params.e
            : params.url,
        url_title: params.url_title,
        priority: priority
    };

    if (priority == 2) {
        if (isNaN(params.retry) || params.retry < 30) {
            throw 'field "retry" should be a number with value of at least 30 if "priority" is set
        }
```

```
            if (isNaN(params.expire) || params.expire > 10800) {
                throw 'field "expire" should be a number with value of at most 10800 if "priority" is
            }

            data.retry = params.retry;
            data.expire = params.expire;
        }

        data = JSON.stringify(data);
        Zabbix.log(4, '[ Pushover Webhook ] Sending request: ' + params.endpoint + '\n' + data);

        request.addHeader('Content-Type: application/json');
        response = request.post(params.endpoint, data);

        Zabbix.log(4, '[ Pushover Webhook ] Received response with status code ' + request.getStatus()

        if (response !== null) {
            try {
                response = JSON.parse(response);
            }
            catch (error) {
                Zabbix.log(4, '[ Pushover Webhook ] Failed to parse response received from Pushover');
                response = null;
            }
        }

        if (request.getStatus() != 200 || response === null || typeof response !== 'object' || respons
            if (response !== null && typeof response === 'object' && typeof response.errors === 'objec
                    && typeof response.errors[0] === 'string') {
                throw response.errors[0];
            }
            else {
                throw 'Unknown error. Check debug log for more information.';
            }
        }

        return 'OK';
    }
    catch (error) {
        Zabbix.log(4, '[ Pushover Webhook ] Pushover notification failed: ' + error);
        throw 'Pushover notification failed: ' + error;
    }
description: |
    请参阅此处的设置指南：https://git.zabbix.com/projects/ZBX/repos/zabbix/browse/templates/media/pushover

    将 token 参数设置为您的 Pushover 应用密钥。
    将 Pushover 媒体分配给 Zabbix 用户时，请将用户密钥添加到“发送到”字段中。
message_templates:
    - event_source: TRIGGERS
      operation_mode: PROBLEM
      subject: '问题：{EVENT.NAME}'
      message: |
        问题开始于 {EVENT.TIME}，日期为 {EVENT.DATE}
        问题名称：{EVENT.NAME}
        主机：{HOST.NAME}
        严重性：{EVENT.SEVERITY}
        运行数据：{EVENT.OPDATA}
        原始问题 ID：{EVENT.ID}
        {TRIGGER.URL}
    - event_source: TRIGGERS
      operation_mode: RECOVERY
      subject: '已在 {EVENT.DURATION} 内解决：{EVENT.NAME}'
```

```
      message: |
        问题已于 {EVENT.RECOVERY.TIME}，日期 {EVENT.RECOVERY.DATE} 解决
        问题名称：{EVENT.NAME}
        问题持续时间：{EVENT.DURATION}
        主机：{HOST.NAME}
        严重性：{EVENT.SEVERITY}
        原始问题 ID：{EVENT.ID}
        {TRIGGER.URL}
    - event_source: TRIGGERS
      operation_mode: UPDATE
      subject: '已更新的问题，持续 {EVENT.AGE}：{EVENT.NAME}'
      message: |
        {USER.FULLNAME} 于 {EVENT.UPDATE.DATE} {EVENT.UPDATE.TIME} {EVENT.UPDATE.ACTION} 了问题。
        {EVENT.UPDATE.MESSAGE}

        当前问题状态为 {EVENT.STATUS}，持续时间为 {EVENT.AGE}，已确认：{EVENT.ACK.STATUS}。
    - event_source: DISCOVERY
      operation_mode: PROBLEM
      subject: '发现：{DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}'
      message: |
        发现规则：{DISCOVERY.RULE.NAME}

        设备 IP：{DISCOVERY.DEVICE.IPADDRESS}
        设备 DNS：{DISCOVERY.DEVICE.DNS}
        设备状态：{DISCOVERY.DEVICE.STATUS}
        设备运行时间：{DISCOVERY.DEVICE.UPTIME}

        设备服务名称：{DISCOVERY.SERVICE.NAME}
        设备服务端口：{DISCOVERY.SERVICE.PORT}
        设备服务状态：{DISCOVERY.SERVICE.STATUS}
        设备服务运行时间：{DISCOVERY.SERVICE.UPTIME}
    - event_source: AUTOREGISTRATION
      operation_mode: PROBLEM
      subject: '自动注册：{HOST.HOST}'
      message: |
        主机名：{HOST.HOST}
        主机 IP：{HOST.IP}
        agent 端口：{HOST.PORT}
```

导出的元素

导出的元素在下表中说明。

Element	Type	Description
name	string	(必填) 媒体类型名称。
type	string	(必填) 媒体类型使用的传输方式。 可能的值:[1] EMAIL (0), SMS (1), SCRIPT (2), WEBHOOK (4).
status	string	媒体类型是否启用。 可能的值:[1] ENABLED (0, 默认), DISABLED (1)
max_sessions	integer	可并行处理的告警最大数量。 SMS 的可能值:[1] 1 (默认). 其他媒体类型的可能值:[1] 0-100 (其中 0 表示无限制).
attempts	integer	发送告警的最大尝试次数。 可能的值:[1] 1-10 (默认: 3).
attempt_interval	string	重试尝试之间的间隔（使用秒或time suffix）。 可能的值:[1] 0-60s (默认: 10s).
description	string	媒体类型描述。
message_templates		媒体类型消息模板的根元素。
event_source	string	(必填) 事件源。 可能的值:[1] TRIGGERS (0), DISCOVERY (1), AUTOREGISTRATION (2), INTERNAL (3), SERVICE (4).
operation_mode	string	操作模式。 可能的值:[1] PROBLEM (0), RECOVERY (1), UPDATE (2).

Element	Type	Description
subject	string	消息主题。
message	string	消息正文。

Note:
另请参见: Media type object（请参阅名称匹配的相关属性）。

Email

以下附加元素仅针对 *Email* 媒体类型导出。

Element	Type	Description
provider	string	Email 提供商。
smtp_server	string	SMTP 服务器。
smtp_port	integer	要连接的 SMTP 服务器端口。 默认值：25。
smtp_helo	string	SMTP helo。
smtp_email	string	将从中发送通知的电子邮件地址。
smtp_security	string	要使用的 SMTP 连接安全级别。 可选值：[1] NONE (0, 默认), STARTTLS (1), SSL_OR_TLS (2)。
smtp_verify_host	string	SMTP 的 SSL 主机验证。 可选值：[1] NO (0, 默认), YES (1)。
smtp_verify_peer	string	SMTP 的 SSL 对等方验证。 可选值：[1] NO (0, 默认), YES (1)。
smtp_authentication	string	要使用的 SMTP 身份验证方法。 可选值：[1] NONE (0, 默认), PASSWORD (1), OAUTH (2)。
username	string	用户名。
password	string	身份验证密码。
redirection_url	string	用于重定向回 OAuth 授权的 Zabbix 前端 URL。
client_id	string	在 OAuth 授权服务器中注册的客户端标识符。
authorization_url	string	用于获取访问令牌和刷新令牌的 OAuth URL，带参数。
token_url	string	用于将授权令牌交换为访问令牌和刷新令牌的 OAuth URL。
message_format	string	消息格式。 可选值：[1] TEXT (0), HTML (1, 默认)。

Note:
另请参见：Media type object（请参阅名称匹配的相关属性）。

SMS

以下附加元素仅针对 *SMS* 媒体类型导出。

元素	类型	描述
gsm_modem	string	（必需）GSM 调制解调器的串行设备名称。

Note:
另请参见：媒体类型对象（请参考名称匹配的相关属性）。

脚本

以下附加元素仅针对 *Script* 媒体类型导出。

元素		类型	描述
脚本名称		string	(必填) 脚本名称。
参数			脚本参数的根元素。
	sortorder	string	(必填) 传递给脚本的脚本参数顺序，作为命令行参数。
	值	string	脚本参数值。

Note:
另请参见：媒体类型对象（请参阅名称匹配的相关属性）。

webhook

以下附加元素仅针对 *webhook* 媒体类型导出。

Element	Type	Description
script	string	脚本。
timeout	string	Javascript 脚本 HTTP 请求超时间隔。 可选值:[1] 1-60s（默认值: 30s）。
process_tags	string	是否处理返回的标签。 可选值:[1] NO (0, 默认), YES (1)。
show_event_menu	string	如果在 `event_menu_url` 和 `event_menu_name` 字段中成功解析了 {EVENT.TAGS.*} 宏，则表示事件菜单中存在一个条目。 可选值:[1] NO (0, 默认), YES (1)。
event_menu_url	string	事件菜单条目的 URL。支持 {EVENT.TAGS.*} 宏。
event_menu_name	string	事件菜单条目的名称。支持 {EVENT.TAGS.*} 宏。
parameters		webhook 媒体类型参数的根元素。
name	string	(必填) webhook 参数名称。
value	string	webhook 参数值。

Note:
另请参见: Media type object（请参阅具有匹配名称的相应属性）。

脚注

[1] 括号中的 API 整数值，例如 ENABLED (0)，仅供参考。有关更多信息，请参见表项中链接的 API 对象页面，或每个章节末尾的相关页面。

仪表板

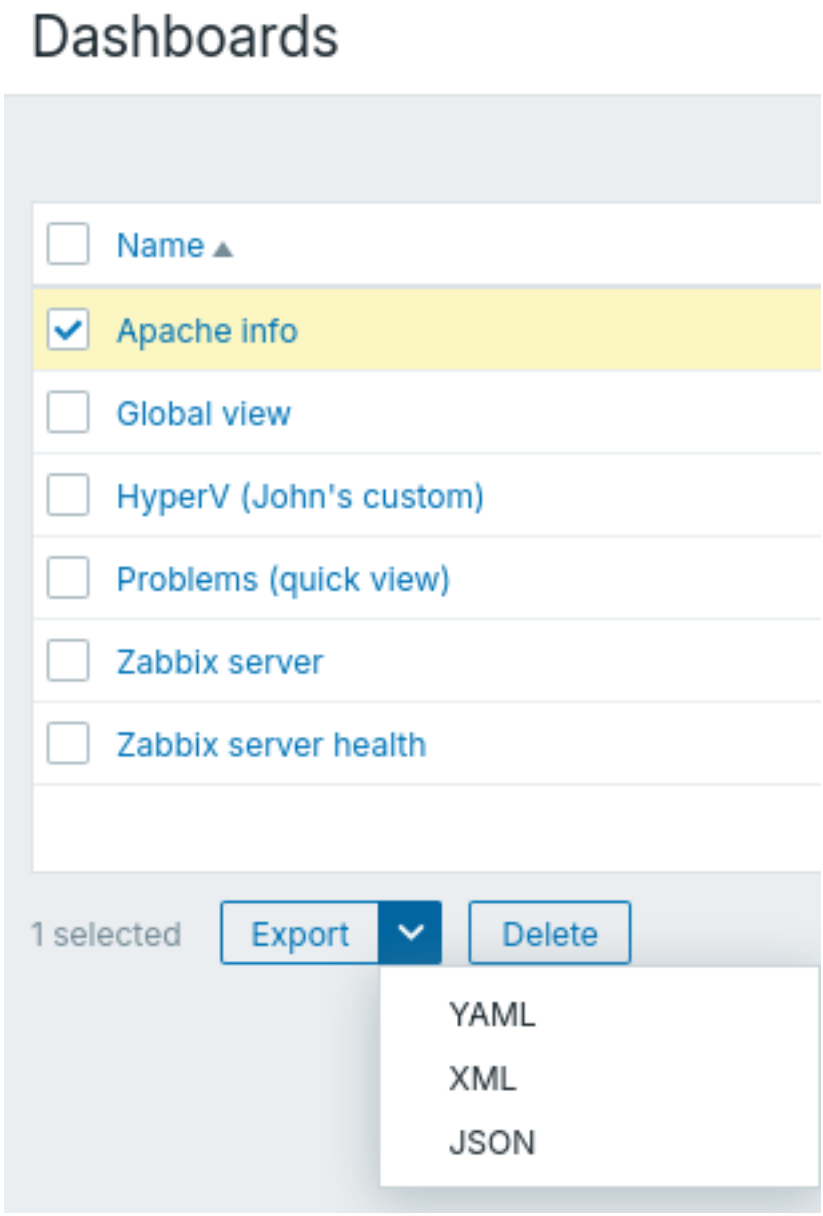
概述

仪表板会与所有相关对象及对象关系一起导出。

导出

要导出仪表板，请执行以下操作：

1. 转到 *Dashboards* > *All dashboards*。
2. 勾选要导出的仪表板复选框。
3. 单击列表下方的 *Export* 以默认 YAML 格式导出，或单击箭头选择其他导出格式。



根据所选格式，仪表板会导出到本地文件，默认文件名如下：

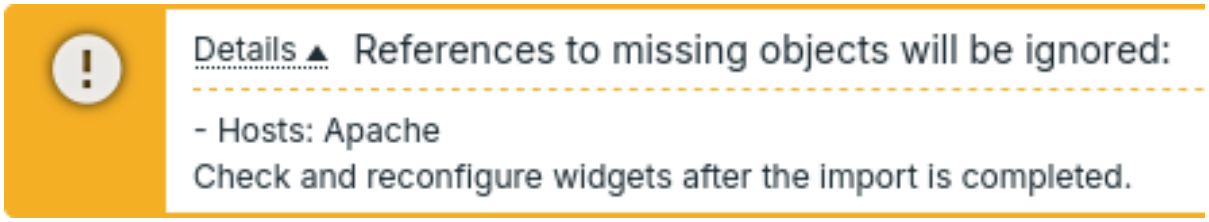
- `zbx_export_dashboards.yaml` - YAML 导出；
- `zbx_export_dashboards.xml` - XML 导出；
- `zbx_export_dashboards.json` - JSON 导出。

导入

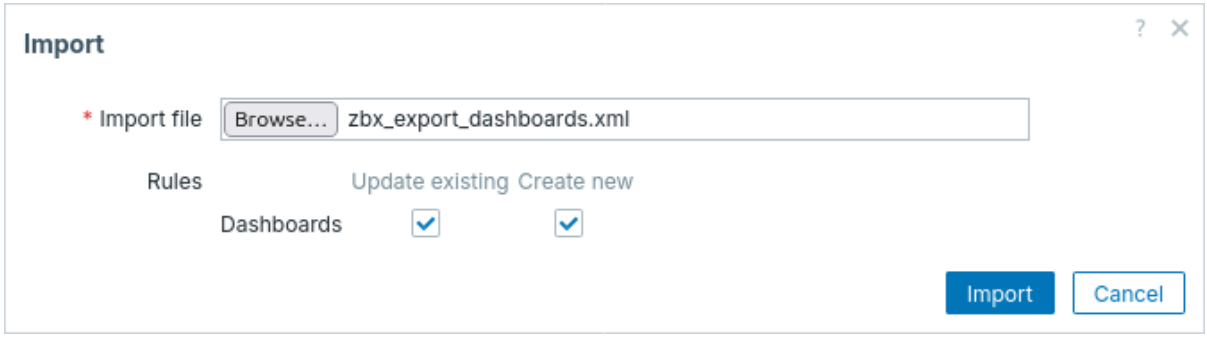
要导入仪表板，请执行以下操作：

1. 转到 仪表板 > 所有仪表板。
2. 点击右上角的 导入。如果用户没有创建和编辑仪表板权限，则此选项将被禁用。
3. 选择导入文件。
4. 勾选要应用的导入规则（见下文）。
5. 点击配置表单右下角的 导入。

如果导入文件引用了目标系统中不存在的对象，则会显示警告消息。对于引用特定主机、监控项或其他对象的小组件，这是预期行为。在这种情况下，Zabbix 允许导入仪表板，清除缺失的引用；如果数据为必填项，则将小组件标记为未完全配置；如果字段仅用于筛选，则删除筛选条件。



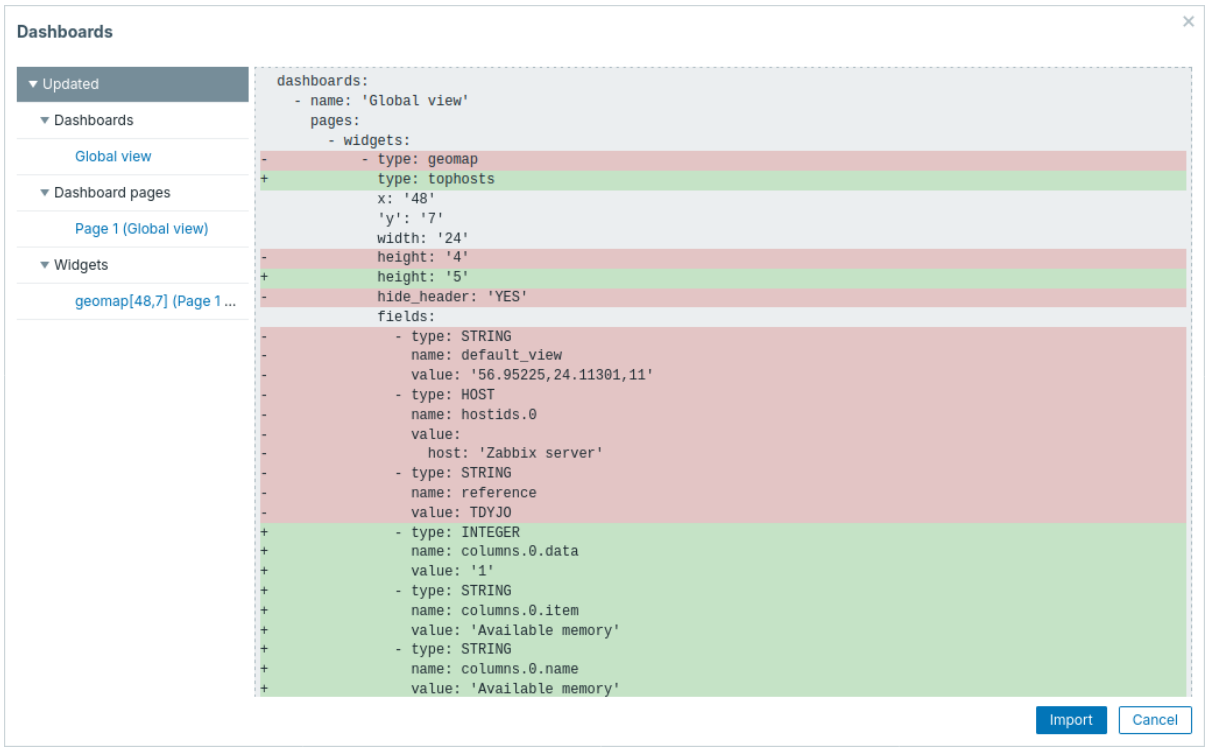
再次点击 导入按钮将添加和/或更新仪表板。



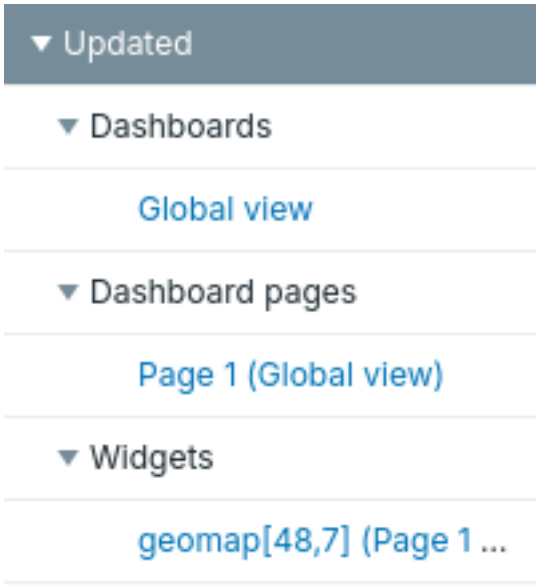
导入规则：

规则	描述
更新现有项	将使用导入文件中的数据更新现有元素。否则，不会更新这些元素。
创建新项	将使用导入文件中的数据创建新元素。否则，不会创建这些元素。

在下一个界面中，您可以查看正在导入的仪表板内容。如果这是一个新仪表板，所有元素都将以绿色列出。如果正在更新现有仪表板，新增的仪表板元素将以绿色突出显示；已删除的仪表板元素将以红色突出显示；未发生变化的元素将列在灰色背景上。



可以使用左侧菜单浏览更改列表。已更新部分突出显示对现有仪表板元素所做的所有更改。已添加部分列出新的仪表板元素。每个部分中的元素按元素类型分组；点击灰色箭头可展开或折叠元素组。



查看仪表板更改，然后点击 导入以执行仪表板导入。导入成功或失败的消息将显示在前端中。

导出格式

导出为 YAML：

```yaml
zabbix_export:
  version: '8.0'
  dashboards:
    - name: 'Global view'
      pages:
        - widgets:
            - type: tophosts
              name: 'CPU 利用率最高的主机'
              width: '25'
              height: '4'
              fields:
                - type: INTEGER
                  name: column
                  value: '1'
                - type: INTEGER
                  name: columns.0.data
                  value: '2'
                - type: STRING
                  name: columns.0.item
                  value: 'System name'
                - type: STRING
                  name: columns.0.name
                  value: '主机名'
                - type: STRING
                  name: columns.1.base_color
                  value: 4CAF50
                - type: INTEGER
                  name: columns.1.data
                  value: '1'
                - type: INTEGER
                  name: columns.1.display
                  value: '3'
                - type: STRING
                  name: columns.1.item
                  value: 'CPU utilization'
                - type: STRING
```

```
      name: columns.1.max
      value: '100'
    - type: STRING
      name: columns.1.min
      value: '0'
    - type: STRING
      name: columns.1.name
      value: Utilization
    - type: INTEGER
      name: columns.2.data
      value: '1'
    - type: STRING
      name: columns.2.item
      value: 'Load average (1m avg)'
    - type: STRING
      name: columns.2.name
      value: '1m avg'
    - type: INTEGER
      name: columns.3.data
      value: '1'
    - type: STRING
      name: columns.3.item
      value: 'Load average (5m avg)'
    - type: STRING
      name: columns.3.name
      value: '5m avg'
    - type: INTEGER
      name: columns.4.data
      value: '1'
    - type: STRING
      name: columns.4.item
      value: 'Load average (15m avg)'
    - type: STRING
      name: columns.4.name
      value: '15m avg'
    - type: INTEGER
      name: columns.5.data
      value: '1'
    - type: INTEGER
      name: columns.5.decimal_places
      value: '0'
    - type: STRING
      name: columns.5.item
      value: 'Number of processes'
    - type: STRING
      name: columns.5.name
      value: Processes
    - type: STRING
      name: columnsthresholds.1.color.0
      value: FFFF00
    - type: STRING
      name: columnsthresholds.1.color.1
      value: FF8000
    - type: STRING
      name: columnsthresholds.1.color.2
      value: FF4000
    - type: STRING
      name: columnsthresholds.1.threshold.0
      value: '50'
    - type: STRING
      name: columnsthresholds.1.threshold.1
      value: '80'
```

```
        - type: STRING
          name: columnsthresholds.1.threshold.2
          value: '90'
        - type: STRING
          name: reference
          value: NLVMO
      - type: hostavail
        'y': '4'
        width: '25'
        fields:
          - type: INTEGER
            name: interface_type.0
            value: '1'
      - type: problems
        name: '当前问题'
        'y': '6'
        width: '48'
        height: '5'
        fields:
          - type: STRING
            name: reference
            value: BRUQE
          - type: INTEGER
            name: show
            value: '3'
          - type: INTEGER
            name: show_tags
            value: '3'
      - type: systeminfo
        x: '25'
        width: '23'
        height: '4'
      - type: problemsbysv
        name: '按严重性分类的问题'
        x: '25'
        'y': '4'
        width: '23'
        fields:
          - type: STRING
            name: reference
            value: LBSVB
          - type: INTEGER
            name: show_type
            value: '1'
      - type: gauge
        name: 'Zabbix 服务器'
        x: '48'
        width: '12'
        height: '4'
        hide_header: 'YES'
        fields:
          - type: INTEGER
            name: angle
            value: '270'
          - type: INTEGER
            name: desc_size
            value: '8'
          - type: ITEM
            name: itemid.0
            value:
              host: 'Zabbix server'
              key: vm.memory.utilization
```

```
            - type: STRING
              name: max
              value: '100'
            - type: STRING
              name: min
              value: '0'
            - type: INTEGER
              name: scale_size
              value: '11'
            - type: INTEGER
              name: show.0
              value: '1'
            - type: INTEGER
              name: show.1
              value: '2'
            - type: INTEGER
              name: show.2
              value: '3'
            - type: INTEGER
              name: show.3
              value: '4'
            - type: INTEGER
              name: show.4
              value: '5'
            - type: STRING
              name: thresholds.0.color
              value: 42A5F5
            - type: STRING
              name: thresholds.0.threshold
              value: '0'
            - type: STRING
              name: thresholds.1.color
              value: 9CCC65
            - type: STRING
              name: thresholds.1.threshold
              value: '20'
            - type: STRING
              name: thresholds.2.color
              value: FFEE58
            - type: STRING
              name: thresholds.2.threshold
              value: '40'
            - type: STRING
              name: thresholds.3.color
              value: FFA726
            - type: STRING
              name: thresholds.3.threshold
              value: '60'
            - type: STRING
              name: thresholds.4.color
              value: FF7043
            - type: STRING
              name: thresholds.4.threshold
              value: '80'
            - type: INTEGER
              name: th_arc_size
              value: '9'
            - type: INTEGER
              name: th_show_arc
              value: '1'
            - type: INTEGER
              name: th_show_labels
```

```
            value: '1'
          - type: INTEGER
            name: units_bold
            value: '1'
          - type: INTEGER
            name: units_size
            value: '24'
          - type: INTEGER
            name: value_arc_size
            value: '30'
          - type: INTEGER
            name: value_bold
            value: '1'
          - type: INTEGER
            name: value_size
            value: '24'
        - type: svggraph
          x: '48'
          'y': '4'
          width: '24'
          height: '3'
          hide_header: 'YES'
          fields:
            - type: INTEGER
              name: axisx
              value: '0'
            - type: STRING
              name: ds.0.color
              value: AB47BC
            - type: INTEGER
              name: ds.0.fill
              value: '0'
            - type: STRING
              name: ds.0.hosts.0
              value: 'Zabbix server'
            - type: STRING
              name: ds.0.items.0
              value: 'Available memory'
            - type: INTEGER
              name: ds.0.transparency
              value: '2'
            - type: INTEGER
              name: ds.0.type
              value: '2'
            - type: INTEGER
              name: ds.0.width
              value: '2'
            - type: INTEGER
              name: ds.1.axisy
              value: '1'
            - type: STRING
              name: ds.1.color
              value: 42A5F5
            - type: INTEGER
              name: ds.1.fill
              value: '0'
            - type: STRING
              name: ds.1.hosts.0
              value: 'Zabbix server'
            - type: STRING
              name: ds.1.items.0
              value: 'CPU utilization'
```

```
        - type: INTEGER
          name: ds.1.transparency
          value: '0'
        - type: INTEGER
          name: ds.1.type
          value: '2'
        - type: INTEGER
          name: ds.1.width
          value: '2'
        - type: STRING
          name: reference
          value: TOYGS
      - type: geomap
        x: '48'
        'y': '7'
        width: '24'
        height: '4'
        hide_header: 'YES'
        fields:
          - type: STRING
            name: default_view
            value: '56.95225,24.11301,11'
          - type: HOST
            name: hostids.0
            value:
              host: 'Zabbix server'
          - type: STRING
            name: reference
            value: TDYJO
      - type: clock
        name: '本地时间'
        x: '60'
        width: '12'
        hide_header: 'YES'
        fields:
          - type: INTEGER
            name: clock_type
            value: '1'
          - type: INTEGER
            name: show.0
            value: '1'
          - type: INTEGER
            name: show.1
            value: '2'
          - type: INTEGER
            name: show.2
            value: '3'
          - type: INTEGER
            name: time_sec
            value: '0'
          - type: INTEGER
            name: tzone_format
            value: '1'
      - type: item
        name: '服务器性能'
        x: '60'
        'y': '2'
        width: '12'
        hide_header: 'YES'
        fields:
          - type: INTEGER
            name: aggregate_function
```

```
      value: '2'
    - type: INTEGER
      name: decimal_size
      value: '32'
    - type: STRING
      name: description
      value: 'Values per second'
    - type: INTEGER
      name: desc_h_pos
      value: '0'
    - type: INTEGER
      name: desc_v_pos
      value: '0'
    - type: ITEM
      name: itemid.0
      value:
        host: 'Zabbix server'
        key: 'zabbix[wcache,values]'
    - type: INTEGER
      name: show.0
      value: '1'
    - type: INTEGER
      name: show.1
      value: '2'
    - type: INTEGER
      name: show.2
      value: '5'
    - type: INTEGER
      name: sparkline.fill
      value: '2'
    - type: INTEGER
      name: sparkline.width
      value: '2'
    - type: STRING
      name: thresholds.0.color
      value: DCEDC8
    - type: STRING
      name: thresholds.0.threshold
      value: '20'
    - type: STRING
      name: thresholds.1.color
      value: FFF9C4
    - type: STRING
      name: thresholds.1.threshold
      value: '40'
    - type: STRING
      name: thresholds.2.color
      value: FFCCBC
    - type: STRING
      name: thresholds.2.threshold
      value: '60'
    - type: INTEGER
      name: units_bold
      value: '0'
    - type: INTEGER
      name: units_show
      value: '0'
    - type: INTEGER
      name: value_h_pos
      value: '0'
    - type: INTEGER
      name: value_size
```

```
                value: '32'
```

导出的元素

下表说明了导出的元素。

Element	Type	Description
name	string	(必填) 仪表板名称。
pages		仪表板页面的根元素
widgets		仪表板页面的根元素

Note:
另请参见：Dashboard object（请参阅具有相同名称的相关属性）。

仪表板小部件

Element	Type	Description
type	string	(必填) 仪表板小部件的类型。 可选值： actionlog - 操作日志； clock - 时钟； discovery - 发现状态； favgraphs - 收藏图形； favmaps - 收藏地图； gauge - 仪表盘； geomap - 地理地图； graph - 图形（经典）； graphprototype - 图形原型； honeycomb - 蜂窝； hostavail - 主机可用性； hostcard - 主机卡片； hostnavigator - 主机导航器； itemcard - 监控项卡片； itemhistory - 监控项历史； itemnavigator - 监控项导航器； item - 监控项值； map - 地图； navtree - 地图导航树； piechart - 饼图； problemhosts - 存在问题的主机； problems - 问题； problemsbysv - 按严重性划分的问题； scatterplot - 散点图； slareport - SLA 报告； svggraph - 图形； systeminfo - 系统信息； tophosts - Top 主机； topitems - Top 监控项； toptriggers - Top 触发器； trigover - 触发器概览； url - URL； web - Web 监控。
x	integer	仪表板左侧起的水平位置。 可选值：[1] 0-71。
y	integer	仪表板顶部起的垂直位置。 可选值：[1] 0-63。
name	string	自定义小部件名称。
width	integer	小部件宽度。 可选值：[1] 1-72。

Element	Type	Description
height	integer	小部件高度。 可选值：[1] 1-64。
hide_header	string	隐藏小部件标题栏。 可选值：[1] NO (0，默认)，YES (1)。
fields		仪表板小部件字段的根元素。

Note:
另请参见：仪表板小部件（请参阅名称匹配的相关属性）。

仪表板小部件字段

Element	Type	Description
type	string	(必填) 小部件字段类型。 可选值:[1] INTEGER (0), STRING (1), HOST_GROUP (2), HOST (3), ITEM (4), ITEM_PROTOTYPE (5), GRAPH (6), GRAPH_PROTOTYPE (7), MAP (8), SERVICE (9), SLA (10), USER (11), ACTION (12), MEDIA_TYPE (13).
name	string	(必填) 小部件字段名称。
value	mixed	(必填) 小部件字段值，具体取决于字段类型。
host	string	监控项主机。
key	string	监控项键值。

Note:
另请参见: Dashboard widget field（请参阅名称匹配的相关属性）。

脚注

[1] 括号中的 API 整数值，例如 ENABLED (0)，仅供参考。有关更多信息，请参见表项中链接的 API 对象页面，或每个章节末尾的相关页面。

正则表达式

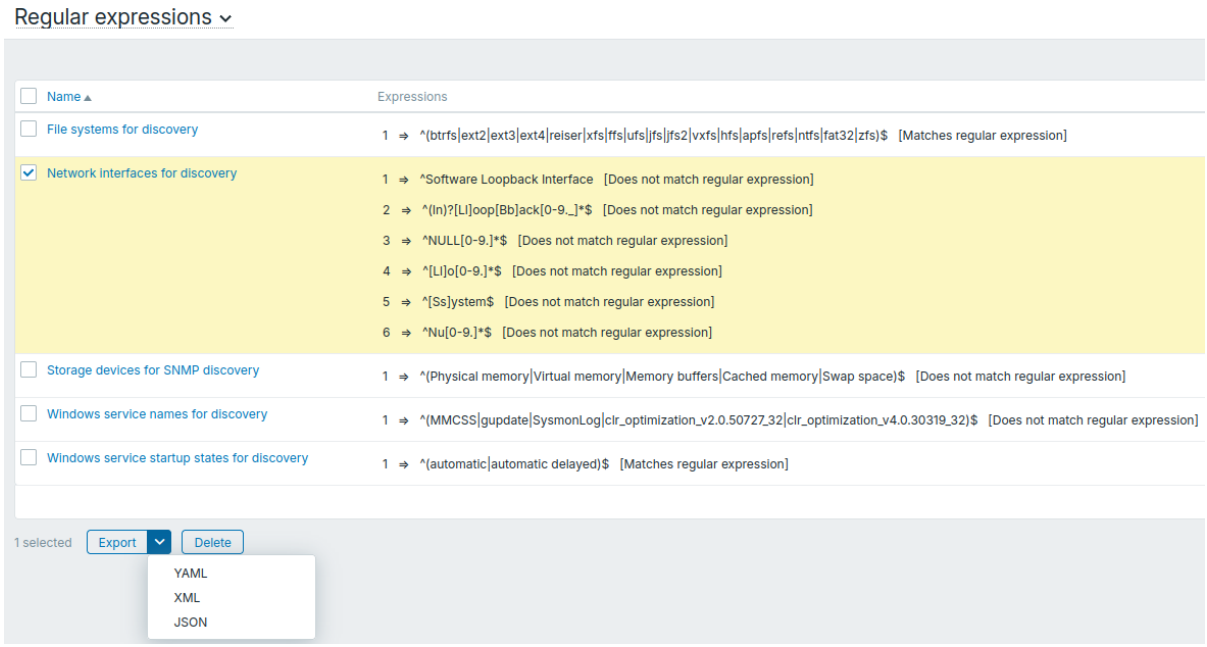
概述

正则表达式会与所有相关对象及对象关系一起导出。

导出

要导出正则表达式，请执行以下操作：

1. 转到 *Administration* > *General* > *Regular expressions*。
2. 勾选要导出的正则表达式复选框。
3. 单击列表下方的 *Export* 以默认 YAML 格式导出，或单击箭头选择其他导出格式。



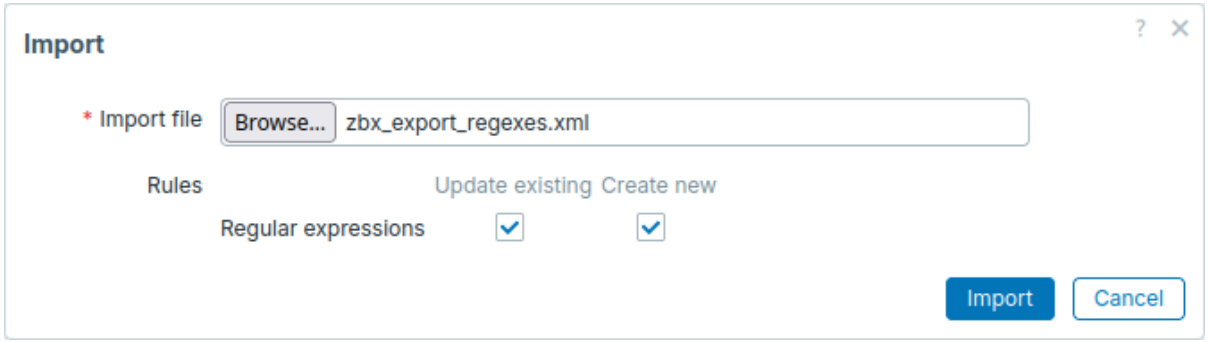
根据所选格式，正则表达式会导出到本地文件，默认文件名如下：

- `zbx_export_regexes.yaml` - YAML 导出；
- `zbx_export_regexes.xml` - XML 导出；
- `zbx_export_regexes.json` - JSON 导出。

导入

要导入正则表达式，请执行以下操作：

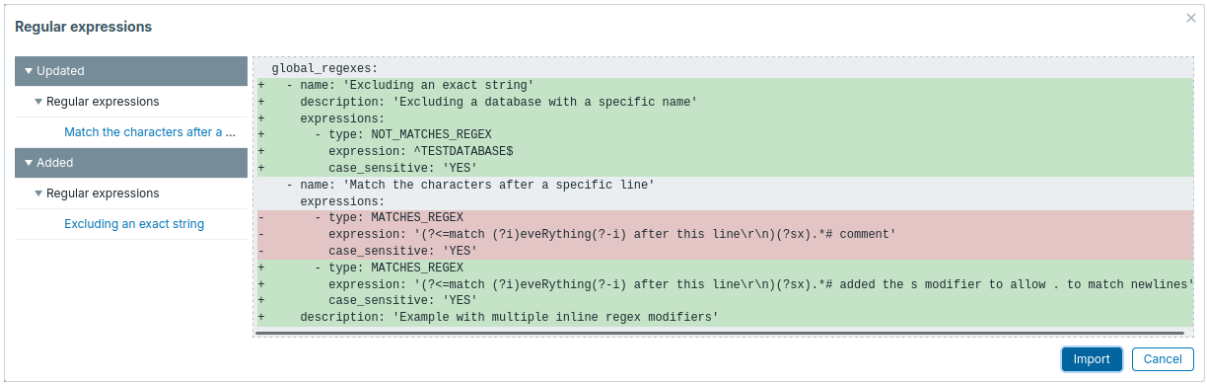
1. 转到 管理 > 常规 > 正则表达式。
2. 单击右上角的 导入。
3. 选择导入文件。
4. 勾选要应用的导入规则（见下文）。
5. 单击配置表单右下角的 导入。



导入规则：

规则	描述
更新现有项	将使用导入文件中的数据更新现有正则表达式，完全覆盖具有相同名称的现有表达式。否则，不会更新这些表达式。
创建新项	将使用导入文件中的数据创建新的正则表达式。否则，不会创建这些表达式。

在下一个界面中，您可以查看正在导入的正则表达式内容。如果这是一个新的正则表达式，所有数据都将以绿色列出。如果正在更新现有正则表达式，新的正则表达式数据将以绿色突出显示；已删除的数据将以红色突出显示；未更改的数据将显示在灰色背景上。



可以使用左侧的菜单浏览更改列表。已更新部分突出显示对现有正则表达式所做的所有更改。已添加部分列出新的正则表达式。单击箭头可展开或折叠正则表达式组。

查看正则表达式更改，然后单击 导入以执行正则表达式导入。导入成功或失败的消息将显示在前端中。

导出格式

导出为 YAML：

```yaml
zabbix_export:
  version: '8.0'
  global_regexes:
    - name: '用于发现的文件系统'
      description: '用于在受监控主机上发现并识别文件系统的全局正则表达式'
      expressions:
        - type: MATCHES_REGEX
          expression: ^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs
    - name: '用于 SNMP 发现的存储设备'
      expressions:
        - type: NOT_MATCHES_REGEX
          expression: '^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$'
          case_sensitive: 'YES'
```

导出元素

正则表达式的导出元素：

Element		Type	Description
global_regexes		object	正则表达式的根元素。
name		string	(必填）正则表达式名称。
description		text	正则表达式的描述。
expressions		object	正则表达式的根元素。
	type	string	正则表达式的类型。
	expression	string	正则表达式。
	exp_delimiter	string	正则表达式分隔符；当类型为 *Contains any substring from list* 时存在。
	case_sensitive	string	正则表达式的大小写敏感选项。

Note:
另请参见：Regular expression object（请参考名称匹配的相关属性）。

正则表达式

概述　Zabbix 支持 Perl 兼容正则表达式（PCRE、PCRE2）。

在 Zabbix 中使用正则表达式有两种方式：

- 手动输入正则表达式
- 使用在 Zabbix 中创建的全局正则表达式

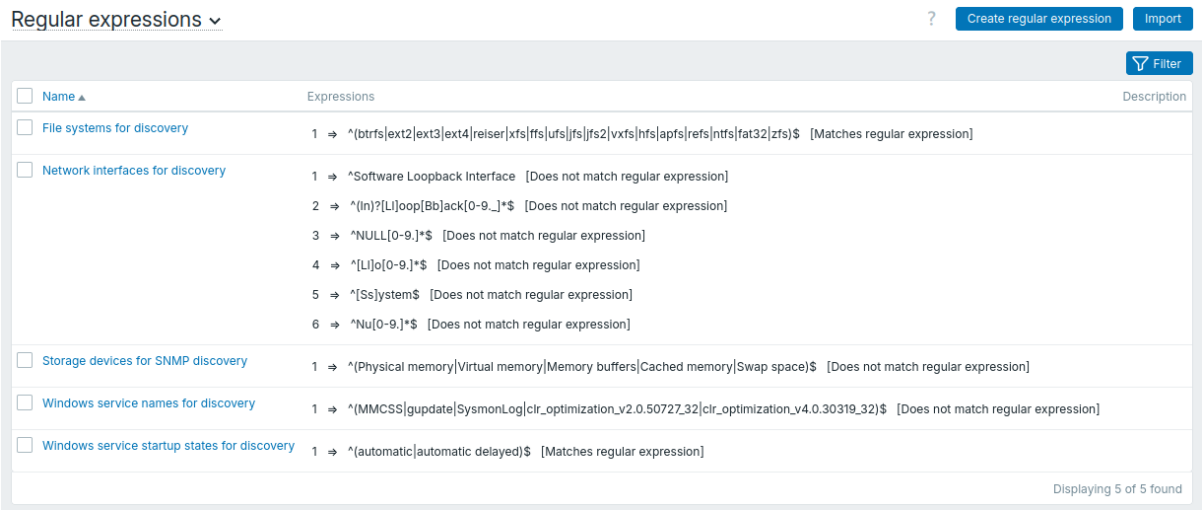
正则表达式　您可以在支持的位置手动输入正则表达式。
 请注意，表达式不能以 @ 开头，因为在 Zabbix 中该符号用于引用全局正则表达式。

Attention:
使用正则表达式时，可能会耗尽栈内存。有关更多信息，请参见 PCRE PERFORMANCE。

请注意，在多行匹配中，^ 和 $ 锚点分别匹配每一行的开头和结尾，而不是整个字符串的开头和结尾。

另请参见在各种上下文中正确转义的示例。

全局正则表达式　Zabbix 在前端提供了一个高级编辑器，用于创建和测试复杂的正则表达式。
 要访问它，请转到 *Administration* > *General* > *Regular expressions*。
 Zabbix 在其默认数据集中包含若干预定义的全局正则表达式。

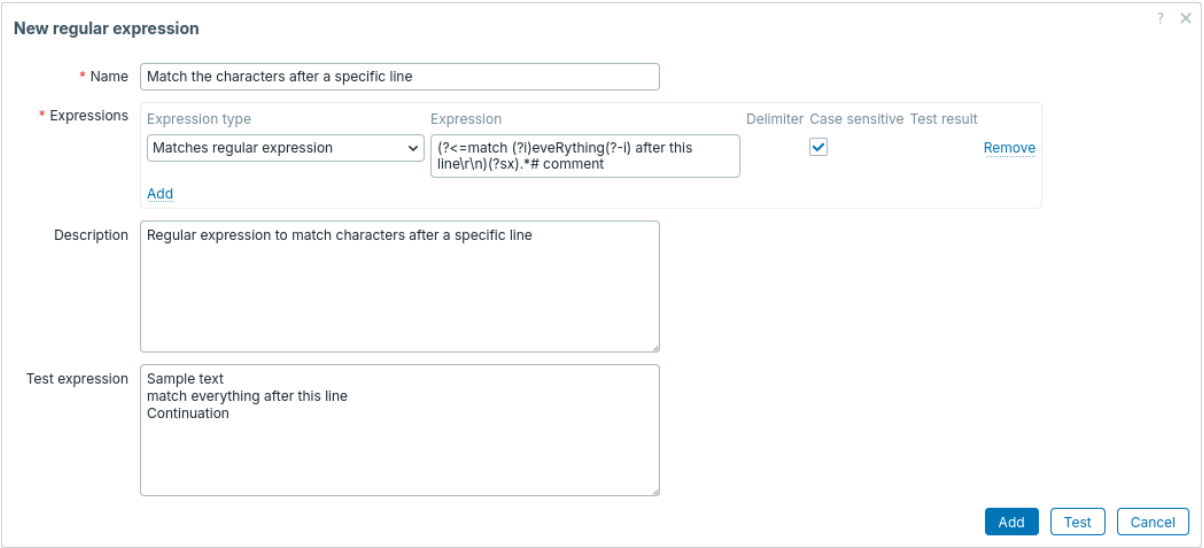


您可以按名称和描述筛选正则表达式列表，以找到所需的表达式。

配置

要创建全局正则表达式：

1. 单击 创建正则表达式。
2. 在表单中输入正则表达式参数。



所有必填输入字段均以红色星号标记。

参数	描述
名称	设置正则表达式名称。允许使用任何 Unicode 字符。
表达式	在 表达式部分单击 添加，以添加新的子表达式。

参数		描述
	表达式类型	选择表达式类型： 包含字符串 - 匹配子字符串。 包含列表中的任意子字符串 - 匹配由分隔符分隔的列表中的任意子字符串。分隔符可以是逗号 ,、句点 . 或正斜杠 /。 不包含字符串 - 匹配除该子字符串之外的任意字符串。 匹配正则表达式 - 匹配该正则表达式。 不匹配正则表达式 - 不匹配该正则表达式。
	表达式	输入子字符串/正则表达式。 最大长度：2048 个字符。
	分隔符	用于在正则表达式中分隔文本字符串的逗号 ,、句点 . 或正斜杠 /。仅当选择 包含列表中的任意子字符串表达式类型时，此参数才处于启用状态。
	区分大小写	用于指定正则表达式是否区分字母大小写的复选框。
描述		输入正则表达式的描述。
测试表达式		用于测试正则表达式的值。

表达式中的正斜杠 / 将按字面值处理，而不是作为分隔符。这样可以在不产生错误的情况下保存包含斜杠的表达式。

Attention:
Zabbix 中的自定义正则表达式名称可以包含逗号、空格等字符。如果在引用时可能导致误解（例如，监控项键参数中包含逗号），可以将整个引用括在引号中，例如：*"@My custom regexp for purpose1, purpose2"*。
 在其他位置（例如 LLD 规则属性中），正则表达式名称不得加引号。

创建正则表达式后，可以在前端的多个位置通过其名称引用它来使用，名称前加 @，例如 `@mycustomregexp`。

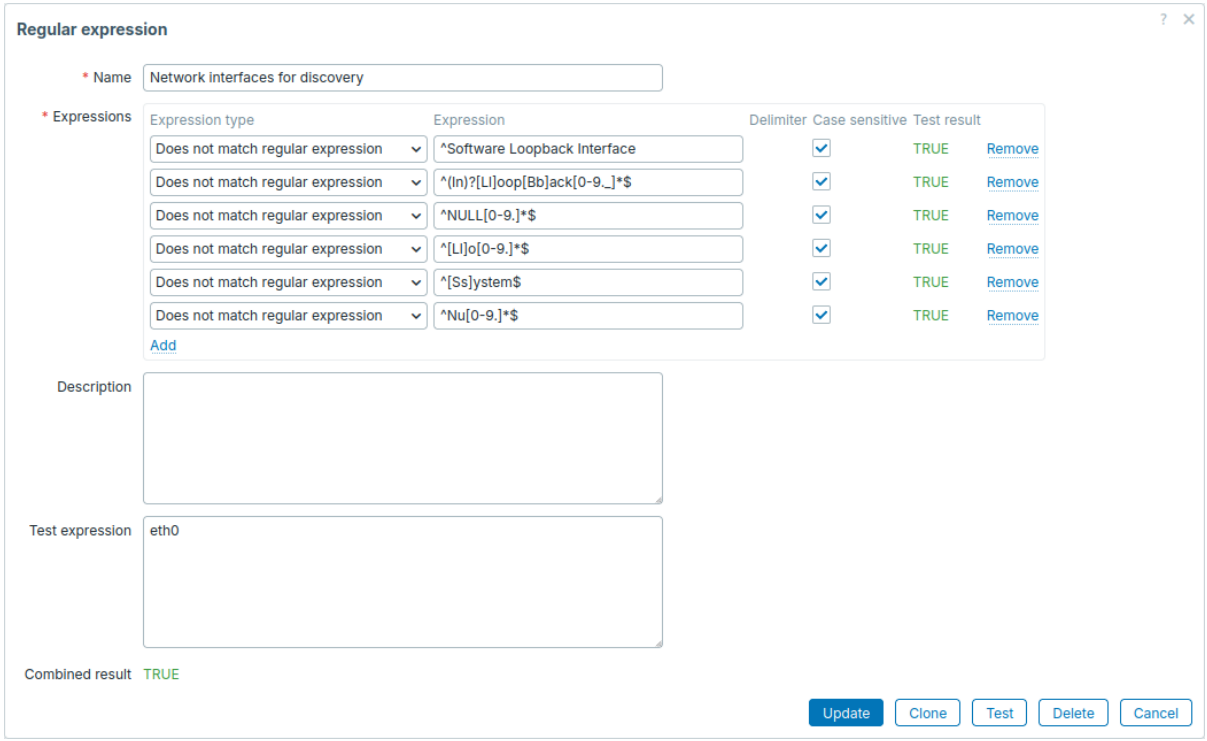
表单按钮

表单底部的按钮可让您执行各种操作：

Button	Description
Add	添加一个正则表达式。此按钮仅适用于新的正则表达式。
Update	更新正则表达式的属性。
Clone	基于当前正则表达式的属性创建另一个正则表达式。
Test	针对提供的值测试正则表达式及其子表达式。
Delete	删除该正则表达式。
Cancel	取消编辑正则表达式属性。

测试正则表达式

您可以通过在 *Test expression* 字段中输入一个值并点击 *Test* 按钮，来测试正则表达式及其子表达式。



在 *Test result* 列中，会显示每个子表达式的状态。*Combined result* 字段显示正则表达式的整体状态。如果定义了多个子表达式，Zabbix 会使用逻辑 AND 运算符来计算 *Combined result*；也就是说，如果至少有一个结果为 `False`，则 *Combined result* 也为 `False`。

默认全局正则表达式

默认数据集包含以下预定义的全局正则表达式：

名称	表达式	匹配
用于发现的文件系统	`^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$`	”btrfs” 或”ext2” 或”ext3” 或”ext4” 或”reiser” 或”xfs” 或”ffs” 或”ufs” 或”jfs” 或”jfs2” 或”vxfs” 或”hfs” 或”apfs” 或”refs” 或”ntfs” 或”fat32” 或”zfs”。
用于发现的网络接口	`^Software Loopback Interface`	以”Software Loopback Interface” 开头的字符串。
	`^lo$`	”lo”。
	`^(In)?[Ll]oop[Bb]ack[0-9._]*$`	可选以”In” 开头，后跟”L” 或”l”，再后跟”oop”，接着是”B” 或”b”，再后跟”ack”，并且后面可选跟任意数量的数字、点或下划线的字符串。
	`^NULL[0-9.]*$`	以”NULL” 开头，后面可选跟任意数量的数字或点的字符串。
	`^[Ll]o[0-9.]*$`	以”Lo” 或”lo” 开头，后面可选跟任意数量的数字或点的字符串。
	`^[Ss]ystem$`	”System” 或”system”。
	`^Nu[0-9.]*$`	以”Nu” 开头，后面可选跟任意数量的数字或点的字符串。
用于 *SNMP* 发现的存储设备	`^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$`	”Physical memory” 或”Virtual memory” 或”Memory buffers” 或”Cached memory” 或”Swap space”。
用于发现的 *Windows* 服务名称	`^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$`	”MMCSS” 或”gupdate” 或”SysmonLog” 或匹配”clr_optimization_v2.0.50727_32” 或”clr_optimization_v4.0.30319_32” 的字符串，其中每个点都可以替换为除换行符之外的任意字符。
用于发现的 *Windows* 服务启动状态	`^(automatic|automatic delayed)$`	”automatic” 或”automatic delayed”。

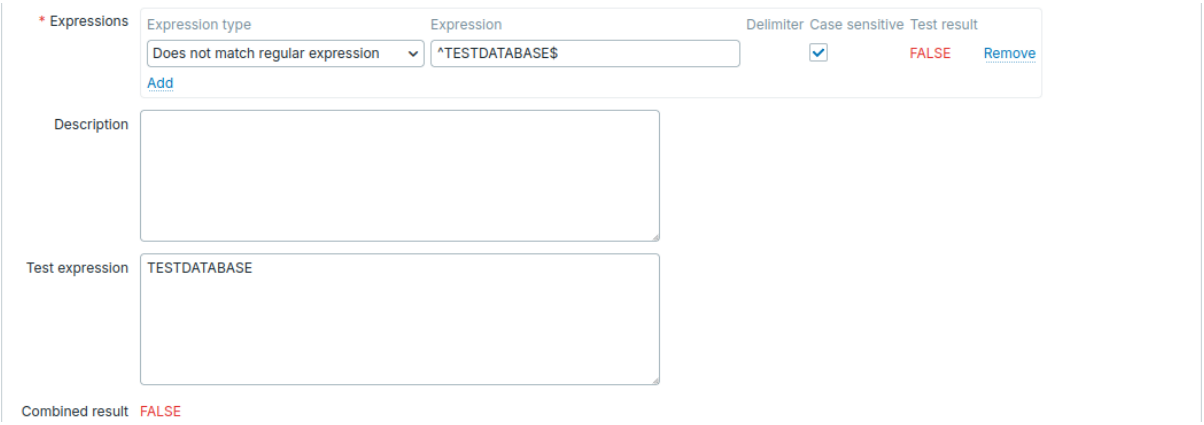
导入/导出正则表达式

您可以按照正则表达式 导出/导入配置页面中的说明，导入或导出全局正则表达式。

示例　排除精确字符串的示例

在低级别发现中使用以下表达式来发现数据库，但排除名称为特定值的数据库：

```
^TESTDATABASE$
```

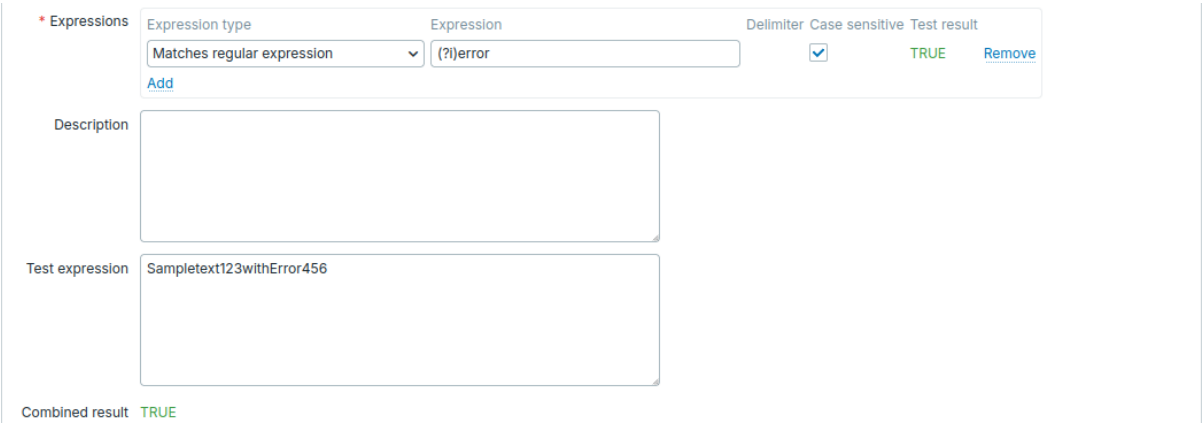


Test result 和 *Combined result* 都显示 FALSE：名称匹配"TESTDATABASE"，因此不满足 *Does not match regular expression* 条件。

使用内联正则修饰符进行不区分大小写匹配的示例

使用以下正则表达式，其中包含内联修饰符（?i），用于对字符"error" 进行不区分大小写的匹配：

```
(?i)error
```



Test result 和 *Combined result* 都显示 TRUE：即使选择了 *Case sensitive*，字符"error" 仍然会被匹配，因为内联修饰符（?i）会使匹配不区分大小写。

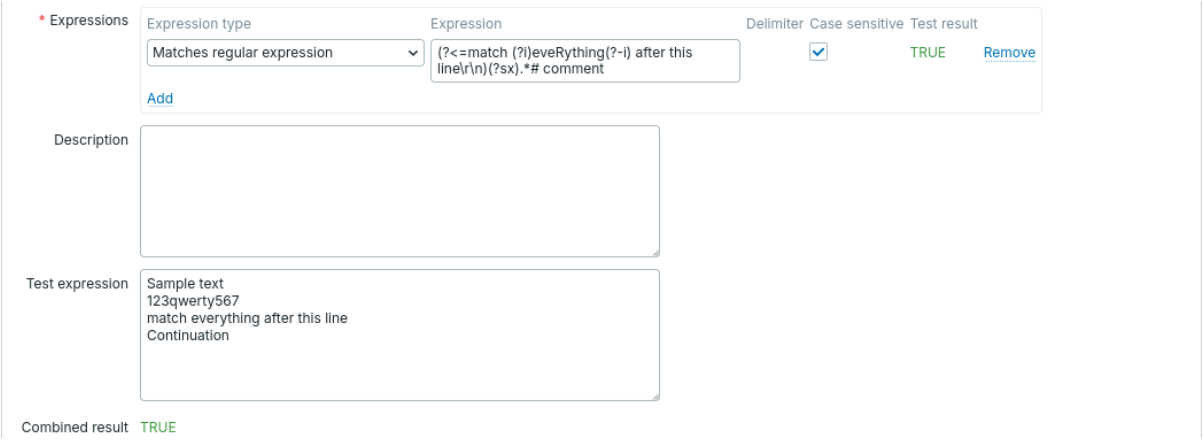
多个内联正则修饰符示例

使用以下正则表达式，其中包含多个内联修饰符，用于匹配特定行之后的字符：

```
(?<=match (?i)eveRything(?-i) after this line\r\n)(?sx).*# comment
```

其中：

- （?i）- 启用不区分大小写匹配；
- （?-i）- 关闭不区分大小写匹配；
- （?s）- 启用 dotall 模式，允许 . 匹配换行符；
- （?x）- 启用扩展模式，允许忽略正则表达式中的空白和注释。



Test result 和 *Combined result* 都显示 TRUE：匹配到了特定行后面的字符。

Attention:
g 修饰符不能以内联方式指定。可用修饰符列表请参见 pcresyntax man page。有关 PCRE 语法的更多信息，请参阅 PCRE HTML documentation。

按位置划分的正则表达式支持情况

位置		正则表达式	全局正则表达式	多行匹配	备注
Agent items					
	eventlog[]	Yes	Yes	Yes	`regexp`、`severity`、`source`、`eventid` 参数
	eventlog.count[]				`regexp`、`severity`、`source`、`eventid` 参数
	log[]				`regexp` 参数
	log.count[]				
	logrt[]		Yes/No		`regexp` 参数同时支持两者，`file_regexp` 参数仅支持非全局表达式
	logrt.count[]				
	proc.cpu.util[]		No	No	`cmdline` 参数
	proc.get[]				
	proc.mem[]				
	proc.num[]				
	sensor[]				Linux 2.4 上的 `device` 和 `sensor` 参数
	system.hw.macaddr[]				`interface` 参数
	system.sw.packages[]				`regexp` 参数
	system.sw.packages.get[]				`regexp` 参数
	vfs.dir.count[]				`regex_incl`、`regex_excl`、`regex_excl_dir` 参数
	vfs.dir.get[]				`regex_incl`、`regex_excl`、`regex_excl_dir` 参数
	vfs.dir.size[]				`regex_incl`、`regex_excl`、`regex_excl_dir` 参数
	vfs.file.regexp[]			Yes	`regexp` 参数
	vfs.file.regmatch[]				
	web.page.regexp[]				
SNMP traps					
	snmptrap[]	Yes	Yes	No	`regexp` 参数
Item value preprocessing		Yes	No	No	`pattern` 参数

位置		正则表达式	全局正则表达式	多行匹配	备注
Functions for triggers/calculated items					
	count()	Yes	Yes	Yes	当 `operator` 参数为 *regexp* 或 *iregexp* 时使用 `pattern` 参数
	countunique()	Yes	Yes		
	find()	Yes	Yes		
	logeventid()	Yes	Yes	No	`pattern` 参数
	logsource()				
Low-level discovery					
	Filters	Yes	Yes	No	*Regular expression* 字段
	Overrides	Yes	No		在 *Operation* 条件的 *matches*、*does not match* 选项中
Action conditions		Yes	No	No	在 *Host name* 和 *Host metadata* 自动注册条件的 *matches*、*does not match* 选项中
Scripts		Yes	Yes	No	*Input validation rule* 字段
Web monitoring		Yes	No	Yes	带有 **regex:** 前缀的 *Variables* *Required string* 字段
User macro context		Yes	No	No	带有 **regex:** 前缀的宏上下文中
Macro functions					
	regsub()	Yes	No	No	`pattern` 参数
	iregsub()				
Link indicators in maps		Yes	No	No	*Pattern* 字段（用于文本项）
Icon mapping		Yes	Yes	No	*Expression* 字段
Value mapping		Yes	No	No	如果映射类型为 `regexp`，则使用 *Value* 字段
Agent configuration parameters					
	AllowKeyRegexp	Yes	No	No	Pattern
	DenyKeyRegexp				

事件和解决

概述　Zabbix 中会生成多种类型的事件：

- 触发器事件 - 每当触发器状态发生变化时（*OK→PROBLEM→OK*）
- 服务事件 - 每当服务状态发生变化时（*OK→PROBLEM→OK*）
- 发现事件 - 当检测到主机或服务时
- 自动注册事件 - 当活动 agent 由服务器自动注册时
- 内部事件 - 当监控项/低级别发现规则变为不受支持，或者触发器进入未知状态时

事件带有时间戳，可作为执行操作的依据，例如发送通知邮件、创建工单或执行远程命令。

要在前端查看事件详情，请转到 *Monitoring* > *Problems*。在那里，您可以单击事件的日期和时间以打开事件详情页面，在该页面中可以查看有关该事件的其他信息，包括它是否因维护而被抑制，或是否被用户手动抑制。

问题解决　由触发器创建的问题事件会在导致问题的条件不再存在时自动解决。

为增强问题事件管理的灵活性和便利性，可使用以下选项：

- 事件关联
- 事件确认和抑制
- 手动关闭问题
- 维护周期 - 定义抑制问题的时间段
- 标记 - 按共同特征为问题添加标签

触发器事件　触发器状态的变化是事件最常见且最重要的来源。每当触发器改变其状态时，都会生成一个事件。该事件包含触发器状态变化的详细信息——发生时间以及新的状态是什么。

触发器会创建两种类型的事件 - **Problem** 和 **OK**。

问题事件

当以下情况时，会创建问题事件：

- 当触发器处于 OK 状态时，如果触发器表达式求值为 TRUE；
- 如果为该触发器启用了多个问题事件生成，则每次触发器表达式求值为 TRUE 时都会创建问题事件。

OK 事件

OK 事件会关闭相关的问题事件，并且可能由 3 个组件创建：

- 触发器 - 基于 “OK 事件生成” 和 “OK 事件关闭” 设置；
- 事件关联
- 任务管理器 - 当事件被手动关闭时

触发器有一个 “OK 事件生成” 设置，用于控制 OK 事件的生成方式：

- *Expression* - 当处于问题状态的触发器的表达式计算结果为 FALSE 时，会生成一个 OK 事件。这是最简单的设置，默认启用。
- *Recovery expression* - 当处于问题状态的触发器的表达式计算结果为 FALSE，且恢复表达式计算结果为 TRUE 时，会生成一个 OK 事件。如果触发器的恢复条件与问题条件不同，可使用此设置。
- *None* - 永远不会生成 OK 事件。可与多个问题事件生成配合使用，仅在发生某些情况时发送通知。

此外，触发器还有一个 “OK 事件关闭” 设置，用于控制哪些问题事件会被关闭：

- *All problems* - OK 事件将关闭由该触发器创建的所有未关闭问题
- *All problems if tag values match* - OK 事件将关闭由该触发器创建且至少有一个匹配标签值的未关闭问题。该标签由触发器设置中的 “Tag for matching” 定义。如果没有可关闭的问题事件，则不会生成 OK 事件。这通常称为触发器级事件关联。

事件关联

事件关联（也称为全局事件关联）是一种设置自定义事件关闭规则（从而生成 OK 事件）的方法。

这些规则定义了如何将新的问题事件与现有问题事件配对，并通过生成相应的 OK 事件来关闭新事件或匹配的事件。

但是，必须非常谨慎地配置事件关联，因为它可能会对事件处理性能产生负面影响；如果配置不当，还可能关闭超出预期数量的事件（在最坏的情况下，甚至可能关闭所有问题事件）。以下是一些配置建议：

1. 始终通过为控制事件（与旧事件配对的事件）设置唯一标签来缩小关联范围，并使用 “新事件标签” 关联条件
2. 使用 “关闭旧事件” 操作时，不要忘记添加基于旧事件的条件，否则可能会关闭所有现有问题
3. 避免使用不同关联配置所使用的通用标签名称

任务管理器

如果为触发器启用了 “允许手动关闭” 设置，则可以手动关闭该触发器生成的问题事件。这是在前端中通过更新问题完成的。事件不会直接关闭，而是会创建一个 “关闭事件” 任务，并由任务管理器在稍后处理。任务管理器将生成相应的 OK 事件，问题事件随后会被关闭。

服务事件　仅当启用了这些事件的服务动作时，才会生成服务事件。在这种情况下，服务状态的每次变更都会创建一个新事件：

- 问题事件 - 当服务状态从 **OK** 更改为 **Problem** 时
- OK 事件 - 当服务状态从 **Problem** 更改为 **OK** 时

事件包含服务状态变更的详细信息 - 变更发生的时间以及新的状态是什么。

发现事件　Zabbix 会定期扫描网络发现规则中定义的 IP 范围。检查频率可针对每条规则单独配置。一旦发现某个主机或服务，就会生成一个发现事件（或多个事件）。

Zabbix 会生成以下事件：

事件	生成时机
服务 Up	每次 Zabbix 检测到活动服务时。
服务 Down	每次 Zabbix 无法检测到服务时。
主机 Up	如果该 IP 的至少一个服务处于 UP 状态。
主机 Down	如果所有服务都无响应。
服务已发现	如果服务在停机后恢复，或首次被发现。
服务丢失	如果服务在处于 UP 状态后丢失。
主机已发现	如果主机在停机后恢复，或首次被发现。
主机丢失	如果主机在处于 UP 状态后丢失。

活动 **agent** 自动注册事件　活动 agent 自动注册会在 Zabbix 中创建事件。

如果已配置，当先前未知的活动 agent 请求检查，或者主机元数据发生更改时，会创建活动 agent 自动注册事件。服务器会使用接收到的 agent IP 地址和端口添加一个新的自动注册主机。

有关更多信息，请参见活动 agent 自动注册 页面。

内部事件　内部事件会在以下情况下发生：

- 监控项的状态从'normal' 变为'unsupported'
- 监控项的状态从'unsupported' 变为'normal'
- 低级别发现规则的状态从'normal' 变为'unsupported'
- 低级别发现规则的状态从'unsupported' 变为'normal'
- 触发器的状态从'normal' 变为'unknown'
- 触发器的状态从'unknown' 变为'normal'

引入内部事件的目的是让用户在发生任何内部事件时都能收到通知，例如，某个监控项变为不受支持并停止采集数据。

只有在启用了这些事件的内部动作时，才会创建内部事件。要停止生成内部事件（例如，监控项变为不受支持时），请在 *Alerts* > *Actions* > *Internal actions* 中禁用所有内部事件动作。

Note:
如果内部动作被禁用，而某个对象处于'unsupported' 状态，则仍会为该对象创建恢复事件。
如果内部动作已启用，而某个对象处于'unsupported' 状态，则即使该对象尚未创建'problem event'，仍会为其创建恢复事件。

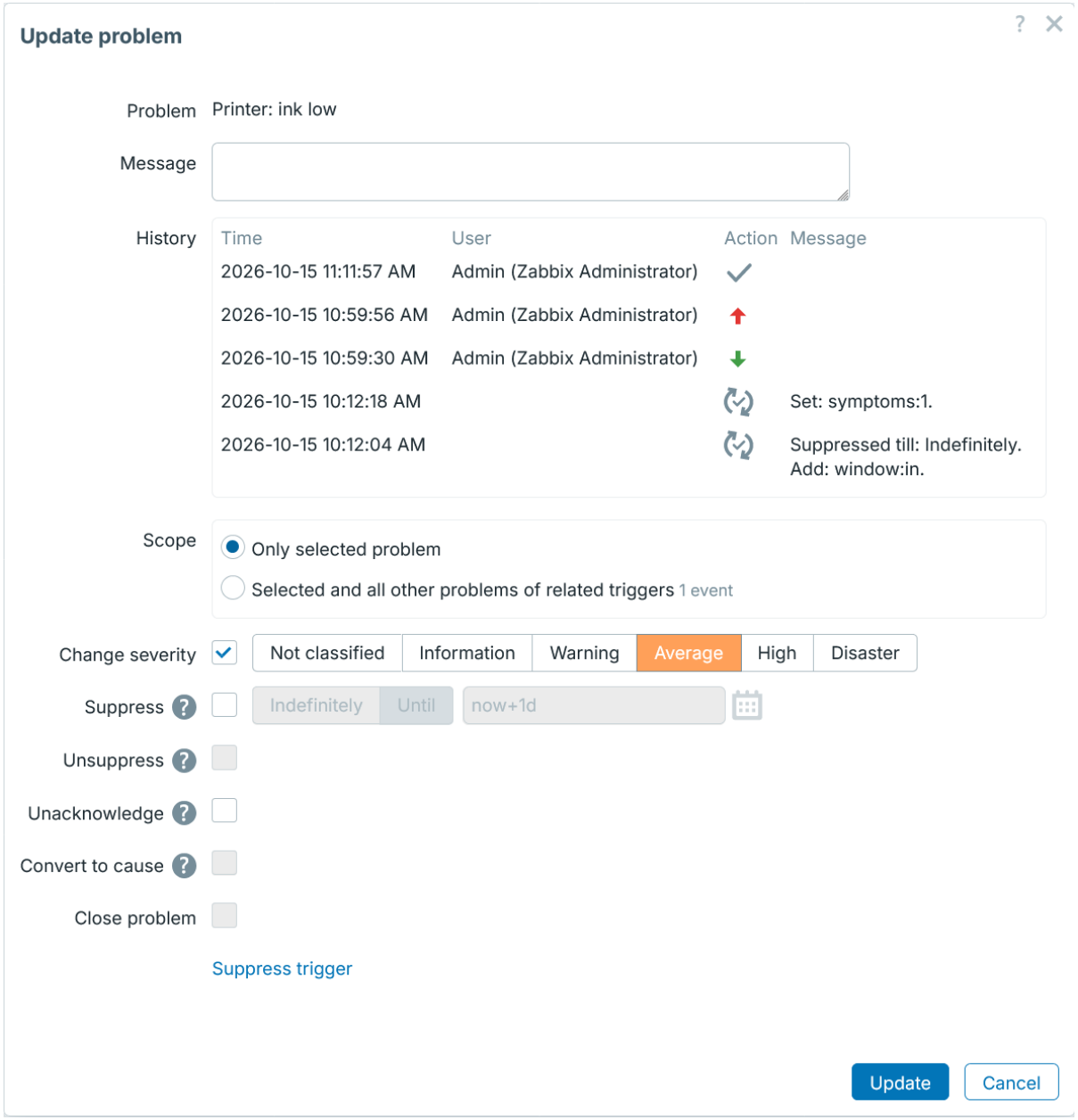
另请参见：接收不受支持监控项的通知

问题确认与抑制

Zabbix 中的问题事件可以由用户确认或抑制。

更新问题　问题更新弹窗允许您：

- 评论问题。
- 查看截至目前的评论和操作。
- 更改问题严重性。
- 抑制/取消抑制问题。
- 确认/取消确认问题。
- 将症状问题更改为原因问题。
- 手动关闭问题。
- 为触发器创建临时维护。



所有必填输入字段均以红色星号标记。

参数	描述
问题	如果仅选择了一个问题，则显示问题名称。 如果选择了多个问题，则显示 已选择 *N* 个问题。
消息	输入用于评论问题的文本（最多 2048 个字符）。
历史记录	列出问题之前的活动和评论，以及时间和用户详细信息。 有关用于表示用户操作的图标含义，请参阅事件详细信息页面。 请注意，仅在更新时选择了一个问题的情况下才会显示历史记录。
范围	定义更改严重性、确认问题或手动关闭问题等操作的范围： 仅选中的问题 - 仅影响此事件 选中的问题以及相关触发器的所有其他问题 - 在确认/关闭问题的情况下，将影响此事件以及截至目前尚未确认/关闭的所有其他问题。如果范围中包含已确认或已关闭的问题，则不会重复确认/关闭这些问题。另一方面，消息和严重性更改操作的次数不受限制。
更改严重性	选中复选框，然后单击严重性按钮以更新问题严重性。 如果至少有一个选中的问题具有读写权限，则可使用更改严重性的复选框。单击更新时，仅会更新具有读写权限的问题。 如果对选中的触发器均不具有读写权限，则复选框处于禁用状态。

参数	描述
抑制	选中复选框以抑制问题： 无限期 - 无限期抑制 直到 - 抑制问题直到指定时间。支持绝对和相对时间格式，例如： now+1d - 从现在起一天（默认） now/w - 直到本周结束 2022-05-28 12:00:00 - 直到指定的绝对日期/时间 请注意，不支持简单时间段（例如 1d、1w）。 此选项是否可用取决于用户角色设置中的 “抑制问题”。 另请参阅：问题抑制
取消抑制	选中复选框以取消抑制问题。仅当至少有一个选中的问题处于抑制状态时，此复选框才处于活动状态。 此选项是否可用取决于用户角色设置中的 “抑制问题”。
确认	选中复选框以确认问题。 当选中的问题中至少有一个未确认问题时，此复选框可用。 无法为已确认的问题添加其他确认（但可以添加其他评论）。
取消确认	选中复选框以取消确认问题。 当选中的问题中至少有一个已确认问题时，此复选框可用。
转换为原因	选中复选框，将症状问题转换为原因问题。
关闭问题	选中复选框以手动关闭选中的问题。 如果在至少一个选中问题的触发器配置中选中了允许手动关闭选项，则可使用关闭问题的复选框。单击更新时，仅会关闭允许关闭的问题。 如果没有可手动关闭的问题，则复选框处于禁用状态。 已关闭的问题不会被重复关闭。
抑制触发器	单击以开始为触发器创建临时维护。 系统将打开维护期配置表单。其中将预先选择从现在开始、持续 1 小时的一次性维护期（可在用户设置和个人资料中配置）。相应的触发器也会被预先选中。维护名称将按 临时：< 事件名称 > 格式预先填充。

问题确认　如果用户收到有关问题事件的通知，他们可以前往 Zabbix 前端，使用下面列出的方法之一打开该问题的 “问题更新” 弹出窗口，并确认该问题。在确认时，他们可以输入评论，例如说明自己正在处理该问题，或者写下他们想对该问题表达的任何其他内容。

这样一来，如果另一位系统用户发现同样的问题，他们就能立即看到该问题是否已被确认，以及目前为止的评论内容。

这种方式可以让多个系统用户协同处理问题的工作流程有序进行。

在定义action operations 时，也会使用确认状态。例如，您可以定义：只有当某个事件在一段时间内未被确认时，才向更高级别的管理者发送通知。

要确认事件并对其添加评论，用户必须至少对相应的触发器具有只读权限。要更改问题严重性或关闭问题，用户必须对相应的触发器具有读写权限。

有多种方式可以访问 “问题更新” 弹出窗口，从而确认问题。

- 您可以在 *Monitoring* > *Problems* 中选择问题，然后点击列表下方的 *Mass update*
- 您可以在以下位置的问题的 *Update* 列中点击 *Update*：
  - *Dashboards*（*Problems* 和 *Problems by severity* 小部件）
  - *Monitoring* > *Problems*
  - *Monitoring* > *Problems* > *Event details*
- 您可以在以下位置点击未解决问题单元格：
  - *Dashboards*（*Trigger overview* 小部件）

弹出菜单包含一个 *Update* 选项，点击后将进入问题更新窗口。

根据确认信息，可以配置仪表板或拓扑图中问题数量的显示方式。为此，您需要在问题显示选项中进行选择，该选项同时位于拓扑图配置和按严重性显示问题仪表板小组件中。

可以显示所有问题数量、将未确认问题数量与总数分开显示，或仅显示未确认问题数量。

根据问题更新信息（确认等），可以配置更新操作，例如发送消息或执行远程命令。

问题抑制　问题抑制提供了一种临时隐藏问题、稍后再处理的方法。这对于清理问题列表很有用，可以优先处理最紧急的问题。例如，有时周末可能会出现某个问题，但其紧急程度不足以立即处理，因此可以将其 “暂缓” 到周一早上。

问题抑制允许隐藏单个问题，这与通过主机维护进行抑制不同；在主机维护期间，维护主机的所有问题都会被隐藏。

对于已抑制的问题，触发器动作的操作将会暂停，其方式与主机维护中的处理方式相同。

问题可以通过 **problem update** 窗口进行抑制，其中抑制是问题更新选项之一，此外还包括添加评论、更改严重性、确认等。

也可以通过同一个问题更新窗口取消抑制问题。

显示

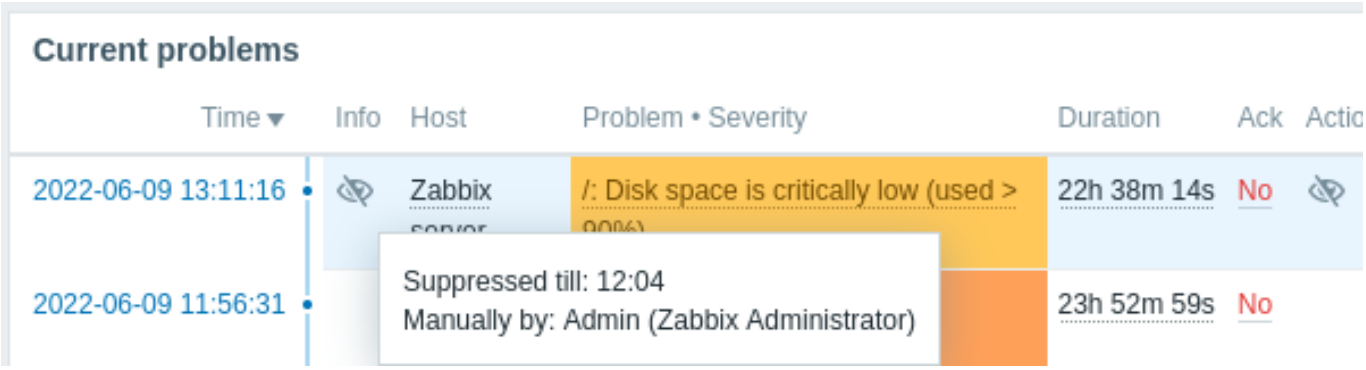
一旦被抑制，问题在被隐藏之前会在 *Info* 列中以闪烁的  抑制图标标记。

如果抑制是由于主机维护导致的，则 *Info* 列中的抑制图标会以橙色显示。如果同时适用维护和手动抑制，则图标颜色以手动抑制为优先。

当抑制任务处于等待列表中时，抑制图标会闪烁。一旦任务管理器已抑制该问题，图标将停止闪烁。如果抑制图标长时间持续闪烁，这可能表示服务器存在问题，例如服务器已停止运行，导致任务管理器无法完成该任务。同样的逻辑也适用于取消抑制。在任务提交后、服务器尚未完成处理的短时间内，取消抑制图标会闪烁。

被抑制的问题可以根据问题筛选器/小组件设置，既可以隐藏，也可以显示。

当在问题列表中显示时，被抑制的问题会用抑制图标标记，并且在鼠标悬停时显示抑制详情：



当鼠标悬停在 *Actions* 列中的抑制图标上时，也会在弹出窗口中显示抑制详情。

手动关闭问题

概述　虽然通常情况下，问题事件会在触发器状态从 **Problem** 变为 **OK** 时自动解决，但在某些情况下，很难仅通过触发器表达式判断问题是否已经解决。在这种情况下，需要手动解决问题。

例如，*syslog* 可能会报告某些内核参数需要调整，以获得最佳性能。在这种情况下，问题会报告给 Linux 管理员，他们修复后再手动关闭该问题。

只有对启用了 *Allow manual close* 选项的触发器，才能手动关闭问题。

当问题被 “手动关闭” 时，Zabbix 会为 Zabbix 服务器生成一个新的内部任务。随后，*task manager* 进程会执行该任务并生成一个 OK 事件，从而关闭问题事件。

生成的 *OK* 事件包含为该事件解析出的完整事件标签集（包括从模板、主机和触发器继承的标签）。这些标签可用于通知，以及诸如`{EVENT.RECOVERY.TAGS}` 和`{EVENT.RECOVERY.TAGSJSON}` 之类的宏中。

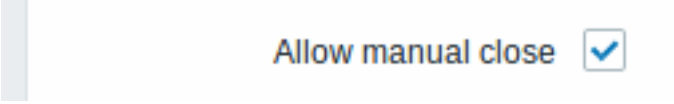
手动关闭的问题并不意味着底层触发器以后不会再次进入 *Problem* 状态。系统会重新评估触发器表达式，并可能再次产生问题：

- 当触发器表达式中包含的任何监控项有新数据到达时（请注意，被节流预处理步骤丢弃的值不视为已接收，因此不会触发重新评估触发器表达式）；
- 当表达式中使用了date and time 和/或nodata() 函数时。

配置　手动关闭问题需要两个步骤。

触发器配置

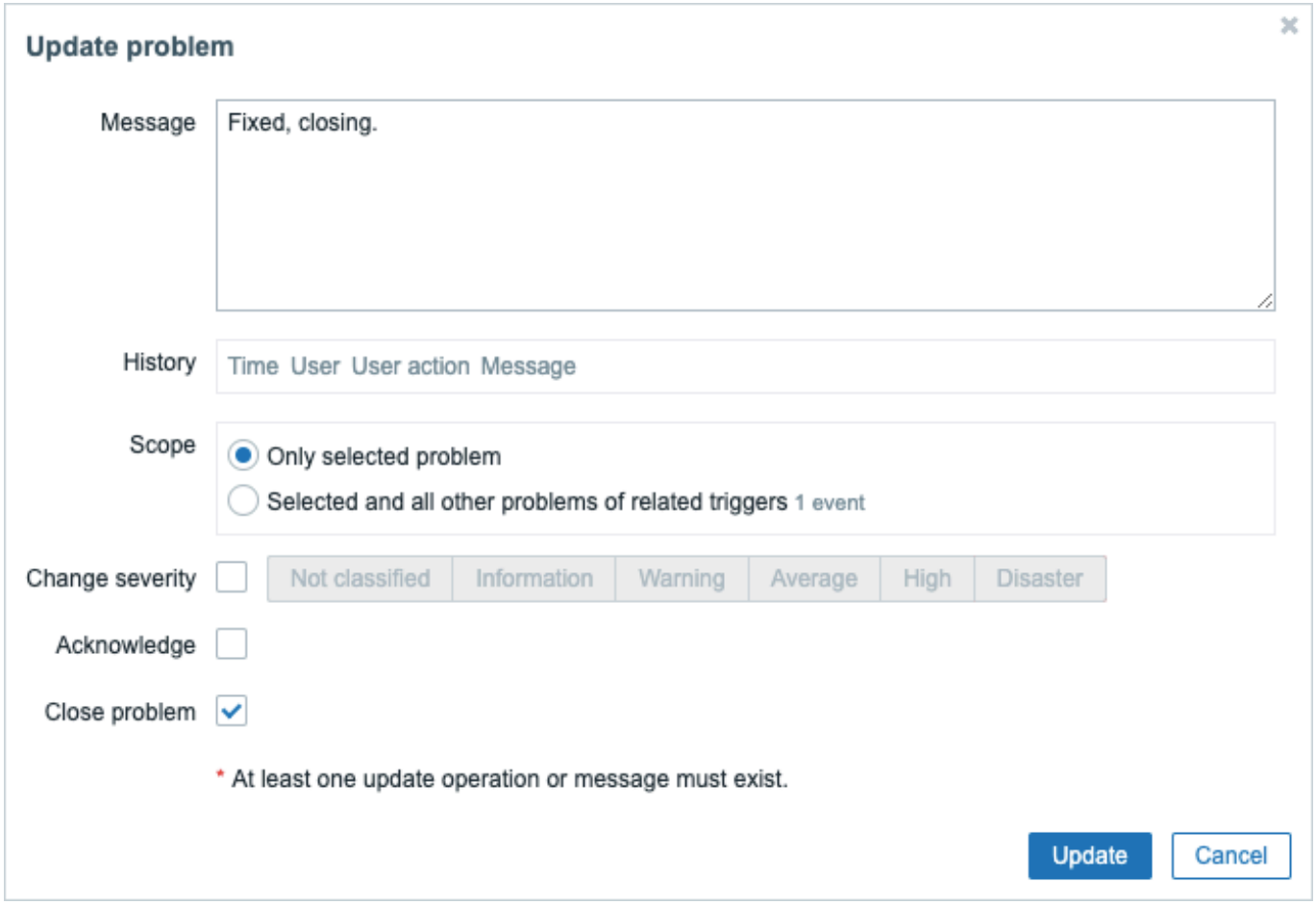
在触发器配置中，启用 允许手动关闭选项。



问题更新窗口

如果带有 *Manual close* 标志的触发器发生问题，您可以打开该问题的问题更新 弹出窗口，并手动关闭该问题。

要关闭该问题，请在表单中勾选 *Close problem* 选项，然后单击 *Update*。



所有必填输入字段都用红色星号标记。

该请求由 Zabbix 服务器处理。通常，关闭该问题需要几秒钟。在此过程中，*Monitoring* > *Problems* 中会将 *CLOSING* 显示为该问题的状态。

验证　可以通过以下方式验证问题已被手动关闭：

- 在事件详情中，可通过 监控 > 问题查看；
- 在通知消息中使用 {EVENT.UPDATE.HISTORY} 宏，该宏会提供此信息。

维护周期

概述

维护用于在预定义的时间段内抑制问题。

您可以为以下实体定义维护时段：

- 主机组
- 主机
- 触发器
- 事件名称
- 标签

如果为特定标签或事件名称定义了维护，则维护仅会对相应的触发器生效；主机或主机组的所有其他触发器不会处于维护状态。

维护有两种类型：收集数据和 不收集数据。

在 收集数据维护期间，触发器会照常处理，并在需要时创建事件。但是，如果在动作配置中选中了 暂停对已抑制问题的操作选项，则处于维护状态的实体会暂停问题升级。在这种情况下，只要维护时段持续，可能包括发送通知或执行远程命令的升级步骤都会被忽略。请注意，维护期间不会抑制问题恢复和更新操作，仅会抑制升级。如果问题在维护期间开始，则不会发送恢复通知。

例如，如果升级步骤安排在问题开始后的第 0、30 和 60 分钟执行，而在实际问题发生后第 10 至 40 分钟之间存在一个持续半小时的维护时段，则第二步和第三步会延迟半小时执行，即在第 60 分钟和第 90 分钟执行（前提是问题仍然存在）。同样，如果问题在维护期间发生，升级将在维护结束后开始。

要在维护期间正常接收问题通知（不延迟），必须在动作配置中取消选中 暂停对已抑制问题的操作选项。

Note:
如果至少有一个主机（用于触发器表达式中）不处于维护模式，Zabbix 将发送问题通知。

维护期间，Zabbix 服务器必须保持运行。维护时段每分钟重新计算一次；如果维护时段发生更改，则会在配置缓存重新加载后立即重新计算。

计时器进程会在每分钟第 0 秒检查是否必须将主机状态更改为维护状态或从维护状态恢复。此外，在配置更新后，计时器进程每秒检查一次是否需要根据 [维护时段] 的更改启动或停止维护。因此，维护时段的启动/停止速度取决于配置更新间隔（默认为 10 秒）。请注意，维护时段更改不包括 生效时间/失效时间设置。此外，如果将某个实体（主机组/主机/触发器等）添加到现有的活动维护时段中，则更改只有在下一分钟开始时由计时器进程激活。

请注意，当主机进入维护状态时，Zabbix 服务器的计时器进程会读取所有未关闭的问题，以检查是否需要抑制这些问题。如果存在大量未关闭的问题，这可能会影响性能。Zabbix 服务器启动时也会读取所有未关闭的问题，即使当时没有配置任何维护。

请注意，无论维护类型如何（包括 不收集数据维护），Zabbix 服务器（或 proxy）始终会收集数据。如果设置了 不收集数据，服务器稍后会忽略这些数据。

当 不收集数据维护结束时，使用 `nodata()` 函数的触发器在其检查周期内不会在下一次检查之前触发。

如果在主机处于维护状态时添加日志监控项，并且维护结束，则只会收集维护结束后新增的日志文件条目。

如果为处于 不收集数据维护状态的主机发送带时间戳的值（例如，使用Zabbix sender），则该值会被丢弃；但是，可以为已过期的维护时段发送带时间戳的值，并且该值会被接受。

如果用户更改了维护时段或维护选项（主机组、主机、触发器等），则更改只有在配置缓存同步后才会生效。

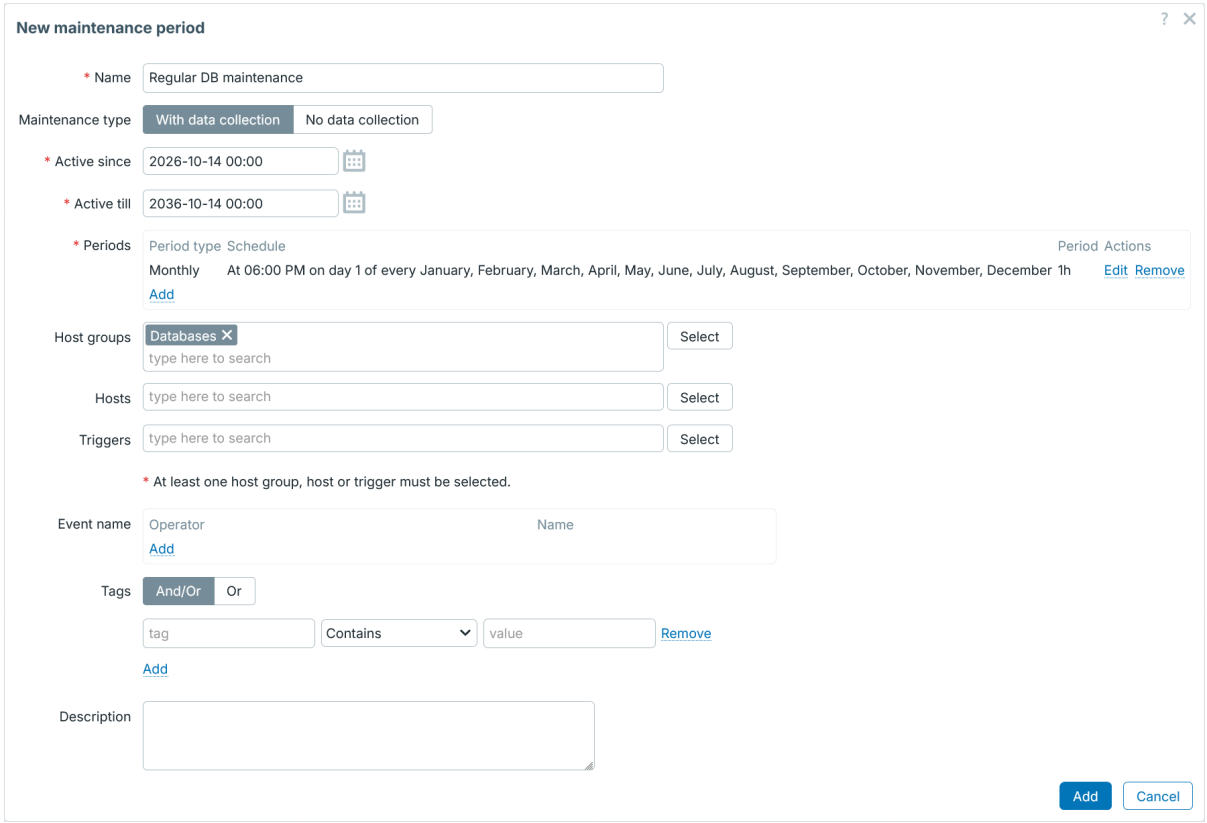
配置

要立即开始维护期：

1. 转到 监测 > 问题。
2. 单击问题名称。
3. 从事件菜单中选择维护选项（主机、触发器、事件名称、标签）。
4. 在表单中修改维护参数。

要配置计划维护期：

1. 转到 数据采集 > 维护。
2. 单击 创建维护期（或现有维护期的名称）。
3. 在表单中输入维护参数。

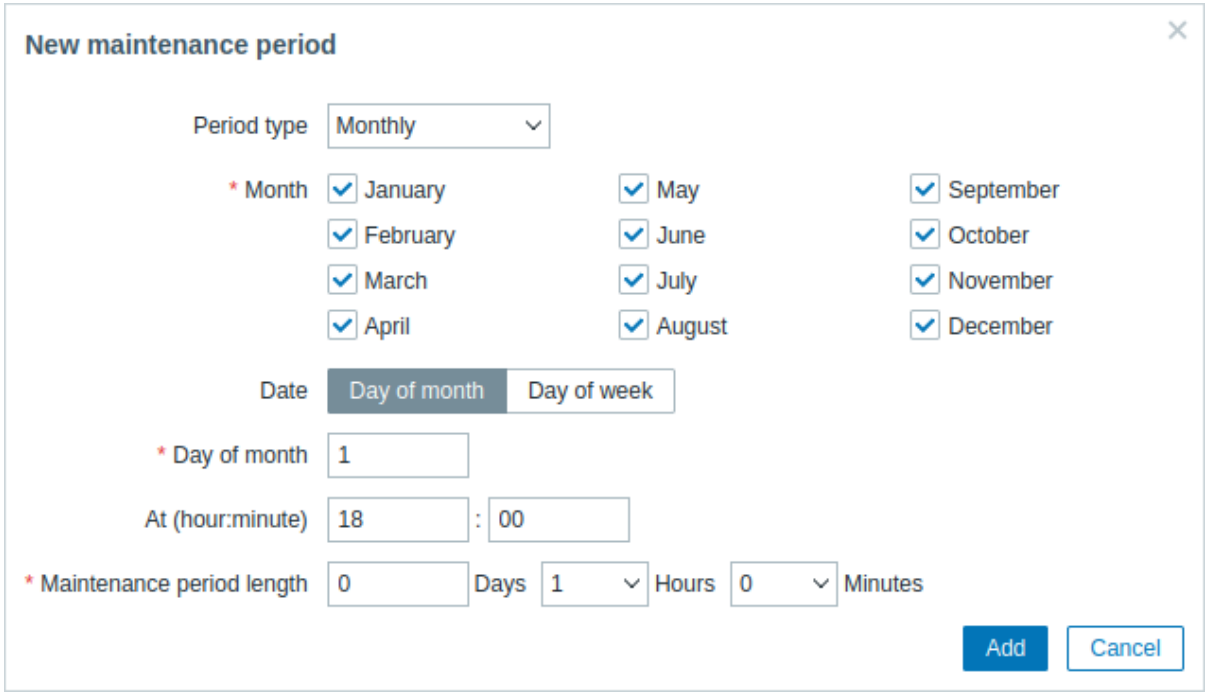


所有必填输入字段均以红色星号标记。

参数	描述
名称	维护期的名称。
维护类型	可以设置两种维护类型： 带数据采集 - 维护期间由服务器采集数据，并处理触发器； 不采集数据 - 仍可能采集数据，但维护期间不会将数据保存到数据库，并且触发器（包括 nodata() 函数）不会触发。 有关每种类型如何影响可用性报告，请参阅维护期的影响。
生效时间	执行维护期开始生效的日期和时间。 注意：仅设置此时间不会激活维护期；必须在 时间段中配置维护期（见下文）。
失效时间	执行维护期停止生效的日期和时间。
时间段	此区域允许您定义维护进行的具体日期和时间。单击 Add 可打开弹出窗口，其中包含灵活的 维护期表单，您可以在其中定义维护计划。有关详细说明，请参阅维护期。
主机群组	选择要激活维护的主机群组。维护将对指定主机群组中的所有主机激活。此字段支持自动补全，因此开始输入时会显示包含所有可用主机群组的下拉列表。 指定父主机群组会隐式选择所有嵌套的主机群组。因此，维护也会对嵌套群组中的主机激活。
主机	选择要激活维护的主机。 此字段支持自动补全，因此开始输入时会显示包含所有可用主机的下拉列表。
触发器	选择要激活维护的触发器。 此字段支持自动补全，因此开始输入时会显示包含所有可用触发器的下拉列表。
事件名称	指定要激活维护的事件名称。 可以设置多个条件。 每个条件有两个可用运算符： 包含 - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写） 不包含 - 排除指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）
标签	指定要激活维护的标签。 可以设置多个条件。标签名称匹配始终区分大小写。 每个条件有多个可用运算符： 包含 - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，区分大小写）； 等于 - 包含指定的标签名称和值（区分大小写）。 不等于 - 排除指定的标签名称和值（区分大小写） 不包含 - 排除指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写） 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将按 “或” 条件分组； 或 - 满足一个条件即可。 仅在选择 带数据采集维护类型时才能指定标签。
描述	维护期的描述。

维护周期

维护周期窗口用于安排周期性或一次性维护的时间。该表单是动态的，所显示的字段会根据所选的 *Period type* 而变化。



Period type	Description
One time only	配置一次性维护周期： *Date* - 维护周期的日期和时间； *Maintenance period length* - 维护将保持活动的时长。
Daily	配置每日维护周期： *Every day(s)* - 维护频率（1 - (默认) 每天，2 - 每两天，等等）； *At (hour:minute)* - 维护开始的当天时间； *Maintenance period length* - 维护将保持活动的时长。 当 *Every day(s)* 参数大于”1” 时，起始日为 *Active since* 时间所在的日期。例如： - 如果 *Active since* 设置为”2021-01-01 12:00”，*Every day(s)* 设置为”2”，且 *At (hour:minute)* 设置为”23:00”，则第一个维护周期将于 1 月 1 日 23:00 开始，而第二个维护周期将于 1 月 3 日 23:00 开始； - 如果 *Active since* 设置为”2021-01-01 12:00”，*Every day(s)* 设置为”2”，且 *At (hour:minute)* 设置为”01:00”，则第一个维护周期将于 1 月 3 日 01:00 开始，而第二个维护周期将于 1 月 5 日 01:00 开始。
Weekly	配置每周维护周期： *Every week(s)* - 维护频率（1 - (默认) 每周，2 - 每两周，等等）； *Day of week* - 维护应在一周中的哪一天进行； *At (hour:minute)* - 维护开始的当天时间； *Maintenance period length* - 维护将保持活动的时长。 当 *Every week(s)* 参数大于”1” 时，起始周为 *Active since* 时间所在的那一周。示例请参见上方 *Daily* 参数说明。
Monthly	配置每月维护周期： *Month* - 选择执行常规维护的所有月份； *Date:* ***Day of month*** - 如果维护应在每个月的同一天进行（例如，每月 1 日），请选择此选项，然后在出现的 *Day of month* 字段中选择所需日期； *Date:* ***Day of week*** - 如果维护只应在某些特定日期进行（例如，每月的第一个星期一），请选择此选项，然后在下拉列表中选择所需的月份中的周次（第一、第二、第三、第四或最后），再勾选维护日期的复选框； *At (hour:minute)* - 维护开始的当天时间； *Maintenance period length* - 维护将保持活动的时长。

Attention:
创建维护周期时，将使用创建该维护周期的用户的时区。但是，在安排重复维护周期（*Daily*、*Weekly*、*Monthly*）时，将使用 Zabbix 服务器的时区。为确保重复维护周期的行为可预测，要求 Zabbix 的所有部分使用统一的时区。

完成后，点击 *Add*，将维护周期添加到 *Periods* 部分。

请注意，夏令时（DST）变更不会影响维护持续的时长。
例如，假设我们配置了一个持续两小时的维护，通常从 01:00 开始，到 03:00 结束：

- 如果在维护进行一小时后（02:00）发生 DST 变更，当前时间从 02:00 变为 03:00，则维护还会继续一小时（直到 04:00）；
- 如果在维护进行两小时后（03:00）发生 DST 变更，当前时间从 03:00 变为 02:00，则维护将停止，因为两小时已经过去；
- 如果维护周期开始于 DST 变更中被跳过的那一小时，则维护不会开始。

如果将维护周期设置为“1 day”（维护的实际周期为 24 小时，因为 Zabbix 按小时计算天数），并且从 00:00 开始，到次日 00:00 结束：

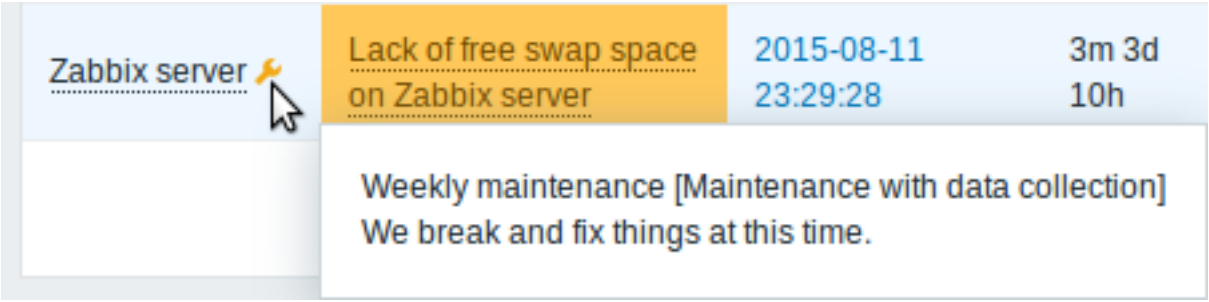
- 如果当前时间向前调整一小时，则维护会在次日 01:00 停止；
- 如果当前时间向后调整一小时，则维护会在当天 23:00 停止。

显示

在维护中显示主机

主机名称旁边的橙色扳手图标 表示该主机处于维护中，显示位置包括：

- 仪表板
- 监控 > 问题
- 资产清单 > 主机 > 主机资产清单详情
- 数据采集 > 主机（参见 状态列）



将鼠标指针悬停在该图标上时，会显示维护详情。

此外，在 监控 > 地图中，处于维护中的主机会显示橙色背景。

显示被抑制的问题

通常，处于维护中的主机的问题会被抑制，也就是说，不会在前端中显示。

被抑制的问题也可能因为被手动抑制而隐藏。

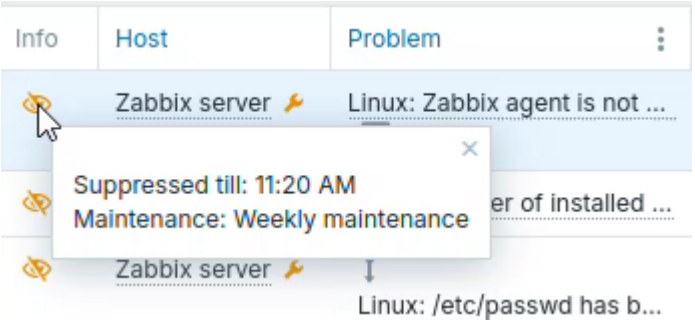
可以在以下位置选择 显示被抑制的问题选项来显示被抑制的问题：

- 仪表板（在 问题主机、问题、按严重性划分的问题、触发器概览小部件配置中）
- 监控 > 问题（在筛选器中）
- 监控 > 地图（在地图配置中）
- 全局通知（在用户配置文件配置中）

显示被抑制的问题时，会显示以下图标：。如果抑制是由于主机维护，则会显示以下图标：

- - 问题因主机维护而被抑制。
- - 问题在主机维护结束后已取消抑制。

将鼠标悬停在图标上可显示更多详细信息。在事件详情中，您可以查看该事件是因为维护而被抑制，还是被手动抑制。如果同时适用维护和手动抑制，则图标颜色以手动抑制为优先。



维护期间的队列计算

Zabbix 前端中显示的队列（Administration > Queue）由 Zabbix 服务器计算。它们不包括处于 *No data collection* 维护中的监控项，并且这些监控项的队列长度始终为零，即使其值已延迟也是如此。处于 *With data collection* 维护中的延迟监控项仍会计入队列。

Zabbix proxy 不会感知维护周期，因为 Zabbix 服务器与 proxy 之间没有维护配置同步。在 Zabbix proxy 上计算的内部检查（例如，`zabbix[queue,,]` 和`zabbix[stats,,,queue,,]`）会报告延迟的监控项，而不考虑 Zabbix 服务器上的维护状态。

因此，对于同一批处于 *No data collection* 维护中的监控项，Zabbix 前端和 Zabbix proxy 上的内部检查可能会报告不同的队列长度。

标记

标签有多种用途，最显著的是用于标记事件。当实体被添加标签后，与该已标记实体相关的任何新事件都会继承其标签。例如：

- 对于已标记的模板 - 任何主机问题（由该模板中的触发器创建）都会继承模板标签。
- 对于已标记的主机 - 任何主机问题都会继承主机标签。
- 对于已标记的监控项/web 场景 - 任何监控项/web 场景问题都会继承监控项/web 场景标签。
- 对于已标记的触发器 - 任何由该触发器创建的问题都会继承触发器标签。

问题事件会继承整个实体链中的所有标签 - 模板、主机、监控项/web 场景、触发器。相同的 `tag:value` 组合（在宏解析后）会合并为一个，从而避免重复。

由手动关闭 生成的恢复事件也会包含从模板、主机、监控项/web 场景和触发器继承的已解析事件标签。这些标签可在通知中使用，也可通过诸如`{EVENT.RECOVERY.TAGS}` 和`{EVENT.RECOVERY.TAGSJSON}` 之类的宏访问。

自定义事件标签提供了更大的灵活性。例如：

- 事件关联 可以基于事件标签进行配置。
- 动作条件 可以基于事件标签进行配置。
- 监控项问题可以基于事件标签进行分组。
- 问题标签可用于将问题映射到服务。

实体可以使用相同的标签名但不同的标签值进行标记（例如，`component:memory` 和 `component:storage`）。同样，实体可以同时拥有一个不带值的标签和一个带值的同名标签（例如，`database` 和 `database:postgresql`）。此类标签不视为重复。

标签由标签名和标签值组成。在为实体添加标签时，您可以只使用名称，也可以将其与值配对使用（例如，`mysql`、`jira`、`target:mysql`、`service:jira` 等）。

标签可为以下各种实体定义：

- 模板
- 主机
- 监控项
- Web 场景
- 触发器
- 服务
- 模板监控项和触发器
- 主机、监控项和触发器原型

在 *Templates*、*Hosts*、*Items*、*Triggers*、*Web scenarios* 列表及其原型中，*Tags* 列同时显示普通标签和继承标签。当悬停在继承标签上或单击继承标签时，会出现一个 JavaScript 提示框，文本为"Inherited tag"。如果继承标签同时也是普通标签，则会根据其所在列表显示不同的提示文本（例如，在templates 列表中显示"Inherited and template tag"，在hosts 列表中显示"Inherited and host tag"）。在前端的其他位置，普通标签和继承标签会一起显示（重复项会被移除），不带图标，也不显示额外的提示文本。

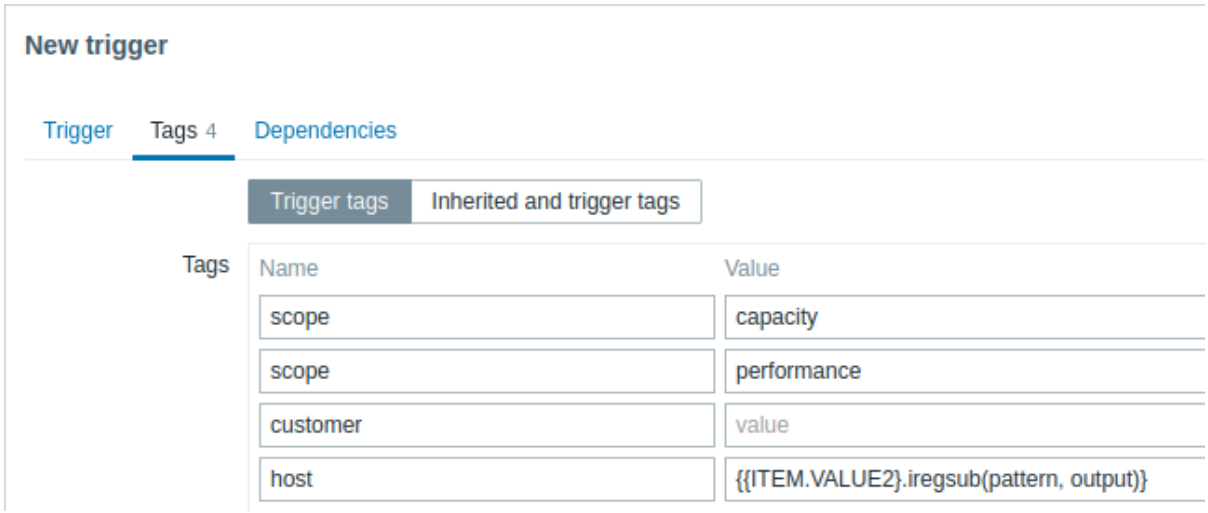
> **Note:**
> 有关定义标签的 一般建议，以及针对 模板、监控项、触发器 和 低级别发现规则 的具体指导，请参阅 Zabbix 官方指南。

用例　标记的常见用例如下：

1. 标记触发器事件：
   - 定义一个触发器标签（例如，`scope:performance`）。
   - 由此触发器创建的问题将带有该触发器标签。
2. 标记继承自模板的问题：
   - 定义一个模板标签（例如，`target:mysql`）。
   - 由此模板中的触发器创建的问题将带有该模板标签。
3. 标记主机问题：
   - 定义一个主机标签（例如，`service:jira`）。
   - 由此主机中的触发器创建的问题将带有该主机标签。

4. 过滤相关监控项：
    - 定义一个监控项标签（例如，`component:cpu`）。
    - 在 *Monitoring* > *Latest data* 中，可以按 `component:cpu` 标签过滤监控项。
5. 使用从监控项值中提取的信息作为标签值：
    - 定义一个将宏作为标签值的标签（例如，`tag-name:{{ITEM.VALUE<N>}.regsub()}`）。
    - 在 *Monitoring* > *Problems* 中，问题的标签值将解析为从监控项值中提取的数据。
6. 识别日志文件中的问题并分别关闭它们：
    - 为日志监控项 的触发器定义一个触发器标签，该标签将使用宏从监控项值中提取值(例如，`service:{{ITEM.VALUE<N>}.regsub()`
    - 在触发器配置 中，设置事件关联：
        - 将 *PROBLEM event generation mode* 设置为"Multiple"；
        - 将 *OK event closes* 设置为"All problems if tag values match"；
        - 设置用于匹配的标签。
    - 由日志监控项触发器创建的问题将带有该触发器标签，并将被单独关闭。
7. 过滤通知：
    - 定义触发器标签（例如，trigger1 使用 `scope:security`，trigger2 使用 `scope:availability`）。
    - 在action conditions 中使用标签过滤，以便仅接收与标签数据匹配的事件通知。
8. 在通知中识别问题：
    - 定义触发器标签。
    - 在问题通知中使用{EVENT.TAGS} 宏。
    - 问题通知将包含触发器标签，从而更容易识别通知属于哪个应用程序/服务。
9. 通过使用模板标签简化配置任务：
    - 定义一个模板触发器标签。
    - 从该模板触发器创建的触发器将带有其标签。
10. 使用低级别发现（LLD）中的标签创建触发器：
    - 在标签名称或值中定义带有 LLD 宏的触发器原型标签（例如，`scope:{#FSNAME}`）。
    - 从该触发器原型创建的触发器将带有其标签。
11. 使用服务标签匹配服务：
    - 定义service tags。
    - 为具有匹配标签的服务配置service actions。
    - 此外，还可以使用服务标签将服务链接到SLA 以进行 SLA 计算。
12. 使用服务问题标签将服务链接到问题：
    - 在service configuration 中定义一个problem tag（例如，`target:mysql`）。
    - 具有匹配标签的问题将自动关联到该服务，并且服务状态将根据已配置的服务状态计算规则进行更改。
13. 当主机处于维护模式时抑制问题：
    - 在maintenance period configuration 中定义标签。
    - 具有所定义标签的问题将被抑制。
14. 授予用户组访问权限：
    - 在user group configuration 中定义标签。
    - 用户组中的用户将只能查看具有所定义标签的问题。

配置　可以在专用选项卡中定义标签，例如在触发器配置 中：



宏支持　标签中的内置宏 和用户宏 会在事件发生时解析。在事件发生之前，这些宏会在 Zabbix 前端中显示为未解析状态。

低级别发现宏 会在发现过程中解析。

以下宏可用于触发器标签名称和值：

- {ITEM.VALUE}、{ITEM.VALUE.AGE}、{ITEM.VALUE.DATE}、{ITEM.VALUE.TIME}、{ITEM.VALUE.TIMESTAMP}、{ITEM.LASTVALUE}、{ITEM.LASTVALUE.AGE}、{ITEM.LASTVALUE.DATE}、{ITEM.LASTVALUE.TIME}、{ITEM.LASTVALUE.TIMESTAMP}、{HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 内置宏
- {INVENTORY.*} 内置宏（用于在触发器表达式中引用一个或多个主机的主机清单值）
- 用户宏和带上下文的用户宏（上下文可包含低级别发现宏）
- 低级别发现宏（仅用于触发器原型标签）

以下宏可用于模板、主机以及 item/web 场景的标签名称和值：

- {HOST.HOST}、{HOST.NAME}、{HOST.CONN}、{HOST.DNS}、{HOST.IP}、{HOST.PORT} 和 {HOST.ID} 内置宏
- {INVENTORY.*} 内置宏
- 用户宏
- 低级别发现宏（仅用于主机和 item 原型标签）

以下宏可用于基于触发器的通知：

- {EVENT.TAGS} 和 {EVENT.RECOVERY.TAGS} 内置宏（这些宏将解析为以逗号分隔的事件标签或恢复事件标签列表）
- {EVENT.TAGSJSON} 和 {EVENT.RECOVERY.TAGSJSON} 内置宏（这些宏将解析为包含事件标签对象 或恢复事件标签对象的 JSON 数组）

触发器标签中的子字符串提取

支持使用宏函数 来填充标签名称或标签值时进行子字符串提取。该函数会将正则表达式应用于由受支持 的宏获取的值。例如：

```
{{ITEM.VALUE}.regsub(pattern, output)}
{{ITEM.VALUE}.iregsub(pattern, output)}

{{#LLDMACRO}.regsub(pattern, output)}
{{#LLDMACRO}.iregsub(pattern, output)}
```

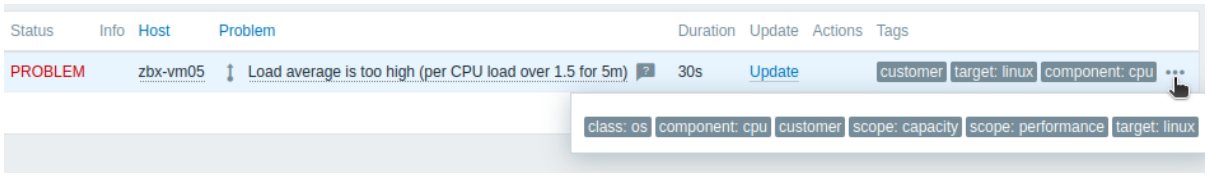
如果标签名称或值在宏解析后超过 255 个字符，则会被截断为 255 个字符。

另请参见：在事件标签中使用低级别发现宏 的宏函数。

查看事件标签　如果已定义标签，可以在以下位置查看新事件中的标签：

- 监控 > 问题
- 监控 > 问题 > 事件详情
- 仪表板 > 问题

显示的标签顺序和数量由 标签显示优先级和 显示标签筛选选项决定，这些选项位于 监控 > 问题或 问题仪表板小部件中。请注意，最多可以显示三个标签；如果标签超过三个，将鼠标悬停在三个点上可在弹出窗口中查看所有标签。



事件关联

概述

事件关联可以非常精确且灵活地将问题事件与其解决事件进行关联。

可以通过以下方式定义事件关联：

- 在触发器级别 - 可以使用一个触发器将不同的问题与其解决方案关联起来；
- 全局 - 可以使用全局关联规则，将问题与来自不同触发器/轮询方法的解决方案关联起来。

基于触发器的事件关联

概述

基于触发器的事件关联允许对同一触发器报告的独立问题进行关联。

通常，一个 OK 事件可以关闭由同一触发器创建的所有问题事件，但在某些情况下，需要采用更细致的方法。例如，在监控日志文件时，您可能希望发现日志文件中的特定问题，并分别关闭这些问题，而不是一次性全部关闭。

对于将 *PROBLEM event generation mode* 参数设置为 *Multiple* 的触发器，就是这种情况。这类触发器通常用于日志监控、trap 处理等场景。

在 Zabbix 中，可以根据标记关联问题事件。标记用于提取值，并为问题事件创建标识。利用这一点，还可以根据匹配的标记分别关闭问题。

换句话说，同一触发器可以创建由事件标记标识的独立事件。因此，可以逐一识别问题事件，并根据事件标记所提供的标识分别关闭这些问题。

工作原理

在日志监控中，您可能会遇到类似以下的行：

```
第1行：服务 1 已停止
第2行：服务 2 已停止
第3行：服务 1 已重启
第4行：服务 2 已重启
```

事件关联的思路是能够将第 1 行中的问题事件与第 3 行中的恢复事件匹配起来，并将第 2 行中的问题事件与第 4 行中的恢复事件匹配起来，然后逐一关闭这些问题：

```
第1行：服务 1 已停止
第3行：服务 1 已重启 #第1行中的问题已关闭

第2行：服务 2 已停止
第4行：服务 2 已重启 #第2行中的问题已关闭
```

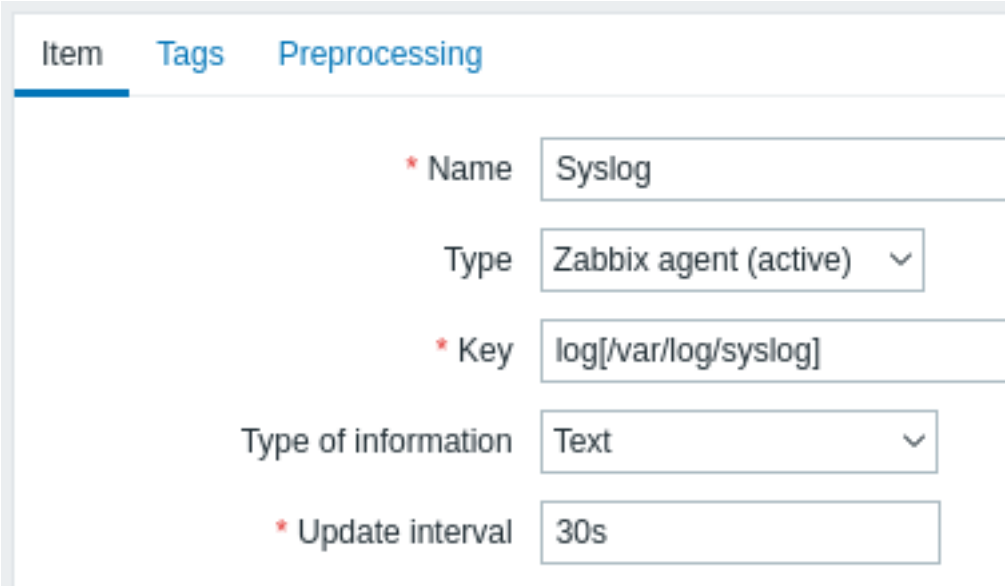
为此，您需要将这些相关事件标记为，例如 “服务 1” 和 “服务 2”。这可以通过对日志行应用正则表达式来提取标签值来实现。然后，在创建事件时，它们会分别被标记为 “服务 1” 和 “服务 2”，从而可以将问题与恢复事件进行匹配。

配置

监控项

首先，您可能需要设置一个监控项来监控日志文件，例如：

```
log[/var/log/syslog]
```



设置好监控项后，请等待一分钟，让配置更改被接收，然后转到最新数据，确认该监控项已开始收集数据。

触发器

在监控项正常工作后，您需要配置触发器。重要的是要决定日志文件中的哪些条目值得关注。例如，下面的触发器表达式会搜索类似于’Stopping’ 的字符串，以提示潜在问题：

```
find(/My host/log[/var/log/syslog],,"regexp","Stopping")=1
```

Attention:
为确保每一行包含字符串”Stopping” 的内容都被视为问题，还需要在触发器配置中将 *Problem event generation mode* 设置为’Multiple’。

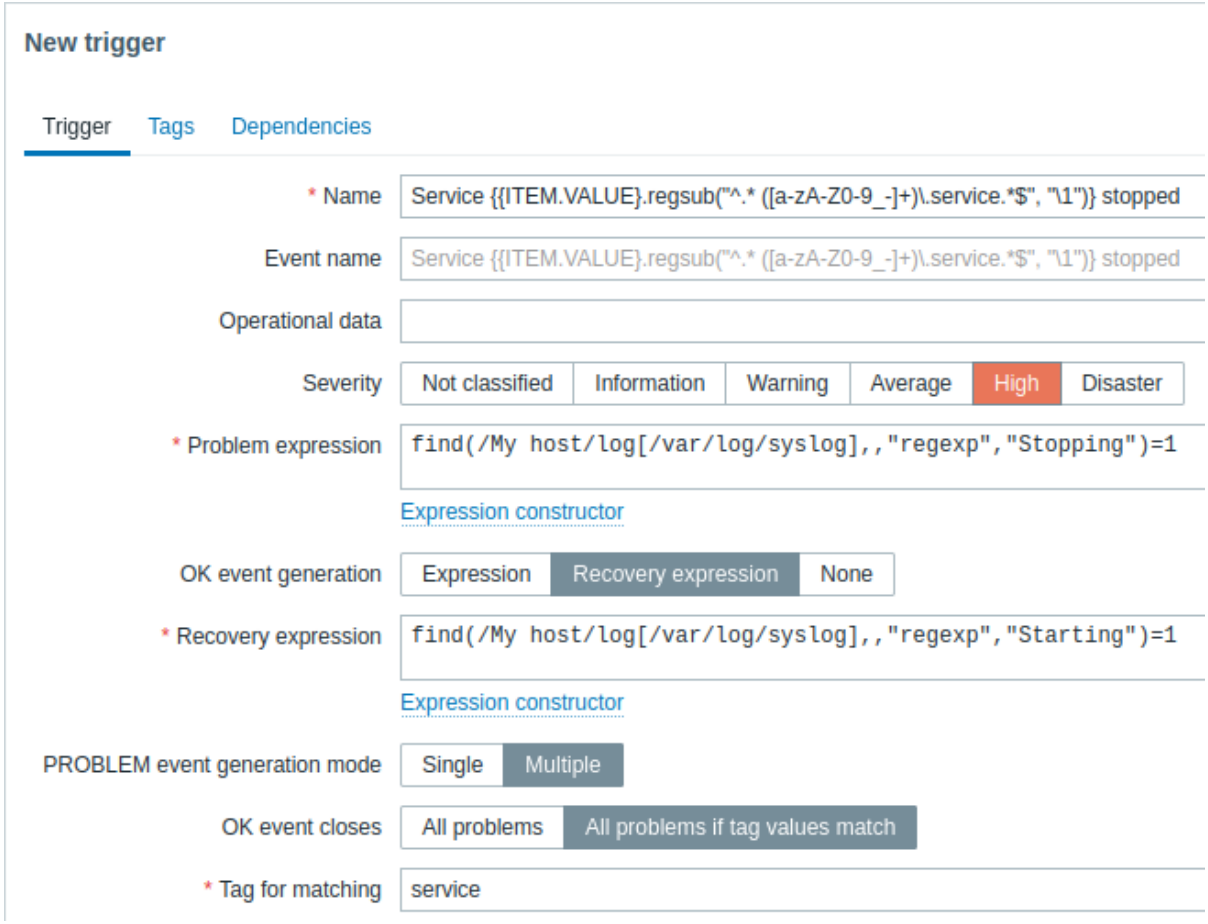
然后定义恢复表达式。下面的恢复表达式会在找到包含字符串"Starting" 的日志行时解决所有问题：

```
find(/My host/log[/var/log/syslog],,"regexp","Starting")=1
```

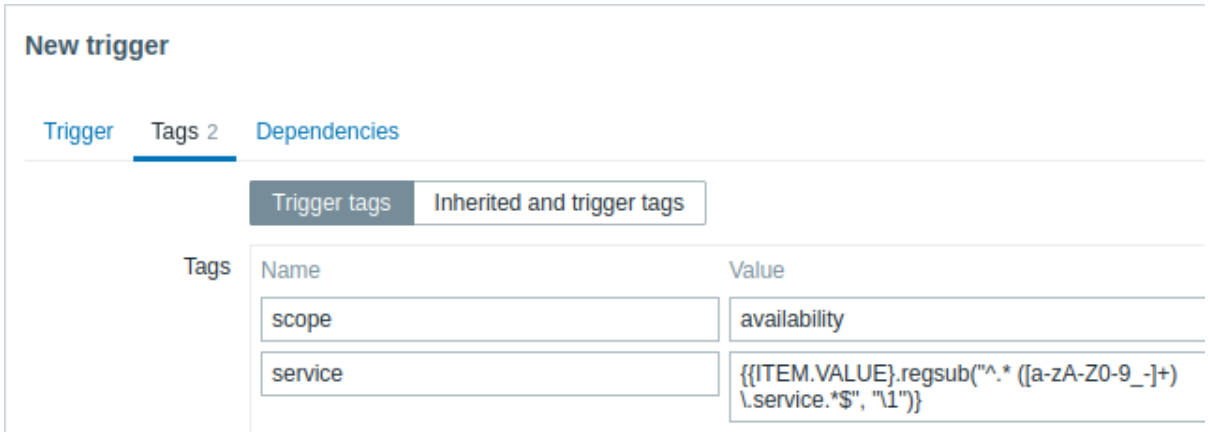
由于我们不希望这样，因此重要的是以某种方式确保相应的根问题被关闭，而不仅仅是所有问题。此时，标签就能派上用场。

可以通过在触发器配置中指定一个标签来匹配问题和恢复。需要进行以下设置：

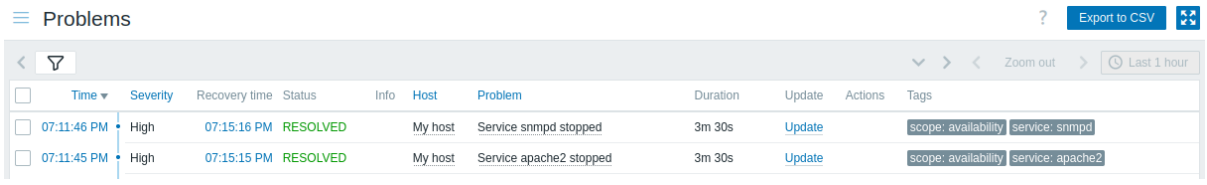
- *Problem event generation mode*: Multiple
- *OK event closes*: All problems if tag values match
- 输入用于事件匹配的标签名称



- 配置标签，从日志行中提取标签值



如果配置成功，您将能够在 *Monitoring* > *Problems* 中看到按应用程序标记并与其恢复相匹配的问题事件。



Warning:
由于可能存在配置错误，当类似的事件标签可能会为无关问题创建时，请务必查看下面列出的情况！

- 索引宏始终引用触发器配置中的 *Expression* 字段，而不是 *Recovery expression*。例如，在恢复事件中，**{ITEM.VALUE1}** 将解析为恢复时刻问题表达式中第一个监控项的最新值。如果恢复表达式基于不同的监控项，并且在恢复时问题表达式对应监控项的值发生了变化，那么这些事件将具有不同的标签，因而不会被关联。

- 当两个应用程序向同一个日志文件写入错误和恢复消息时，用户可能会决定在同一个触发器中使用两个 *service* 标签，并通过在标签值中使用单独的正则表达式来提取服务 A 和服务 B 的名称（例如，当消息格式不同的时候）。但是，如果正则表达式没有匹配，这种做法可能不会按预期工作。不匹配的正则表达式会产生空的标签值，而问题事件和 OK 事件中只要有一个空的标签值，就足以将它们关联起来。因此，来自服务 A 的恢复消息可能会意外关闭来自服务 B 的错误消息。
- 只有在触发器触发时，实际的标签和标签值才会变得可见。如果所使用的正则表达式无效，它会被静默替换为 *UNKNOWN* 字符串。如果带有 *UNKNOWN* 标签值的初始问题事件被遗漏，后续可能会出现具有相同 *UNKNOWN* 标签值的 OK 事件，从而关闭本不应被关闭的问题事件。
- 如果用户将不带宏函数的 {ITEM.VALUE} 宏用作标签值，则适用 255 个字符的限制。当日志消息很长且前 255 个字符没有特异性时，这也可能导致无关问题出现相似的事件标签。

全局事件关联

概述

全局事件关联允许跨越 Zabbix 监控的所有指标，并创建关联关系。

可以关联由完全不同的触发器创建的事件，并对所有这些事件应用相同的操作。通过创建智能关联规则，您可以避免收到数千条重复通知，将精力集中于问题的根本原因！

全局事件关联是一种强大的机制，可以让您摆脱基于单个触发器的问题和恢复逻辑。到目前为止，单个问题事件由一个触发器创建，我们必须依赖同一个触发器来解决该问题。我们无法使用另一个触发器解决由一个触发器创建的问题。但借助基于事件标签的事件关联，我们可以做到这一点。

例如，日志触发器可能会报告应用程序问题，而轮询触发器可能会报告应用程序正在运行。利用事件标签，您可以将日志触发器标记为 *status:down*，同时将轮询触发器标记为 *status:up*。然后，在全局关联规则中，您可以关联这些触发器，并为此关联分配适当的操作，例如关闭旧事件。

在另一种用法中，全局关联可以识别相似的触发器，并对它们应用相同的操作。如果每个网络端口问题只生成一条问题报告，会怎么样？无需报告所有问题。全局事件关联同样可以实现这一点。

全局事件关联在关联规则中进行配置。关联规则定义如何将新问题事件与现有问题事件配对，以及在匹配时执行什么操作（关闭新事件，或通过生成相应的 OK 事件来关闭匹配的旧事件）。如果问题通过全局关联关闭，该信息会显示在监控 > 问题的 *Info* 列中。

只有超级管理员级别的用户可以配置全局关联规则。

Attention:
必须非常谨慎地配置事件关联，因为它可能会对事件处理性能产生负面影响；如果配置不当，还可能关闭超出预期数量的事件（最坏情况下甚至可能关闭所有问题事件）。

要安全地配置全局关联，请遵循以下重要提示：

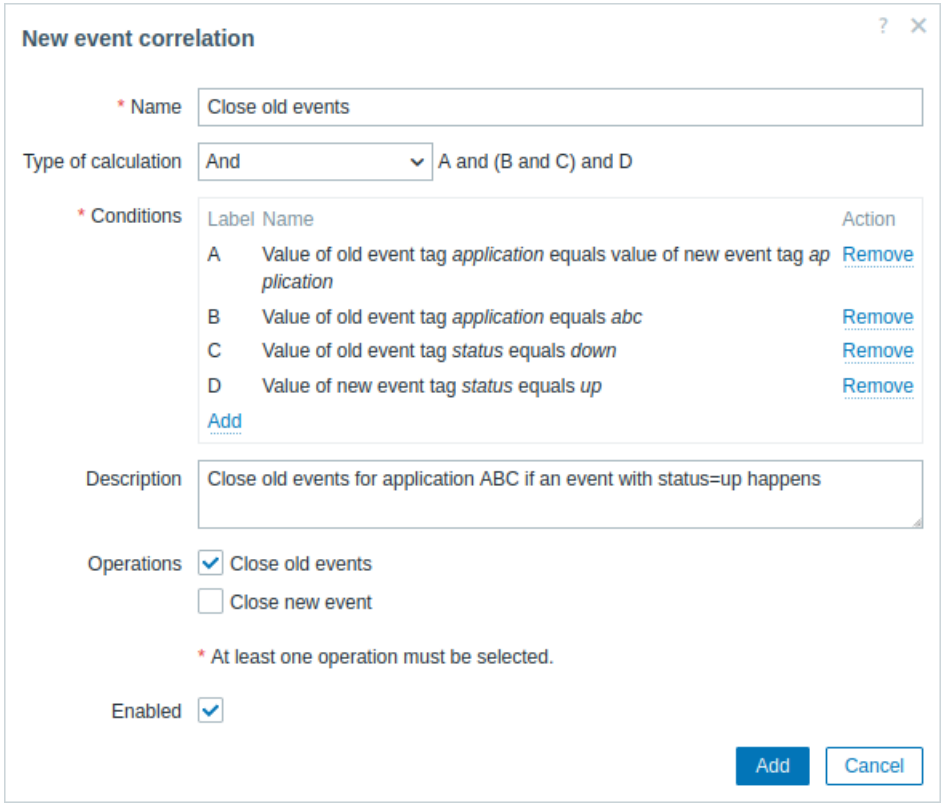
- 缩小关联范围。始终为要与旧事件配对的新事件设置唯一标签，并使用 *New event tag name* 关联条件。
- 使用 **Close old events** 时，添加明确的旧事件条件。选择 **Close old events** 时，始终至少添加一个 *Old event* 条件（例如 *Old event tag name*、*Old event tag value* 或 *Event tag pair*）；否则，该规则可能会匹配并关闭不相关的现有问题事件（最坏情况下会关闭所有问题）。对于匹配运行时值（主机: 端口、会话 ID 等），优先使用 *Event tag pair*；在可能的情况下，再通过主机或主机组进一步缩小匹配范围。
- 避免使用常见的标签名称，以免它们最终被不同的关联配置使用。
- 将关联规则的数量限制在实际需要的范围内。

另请参阅：已知问题。

配置

要全局配置事件关联规则：

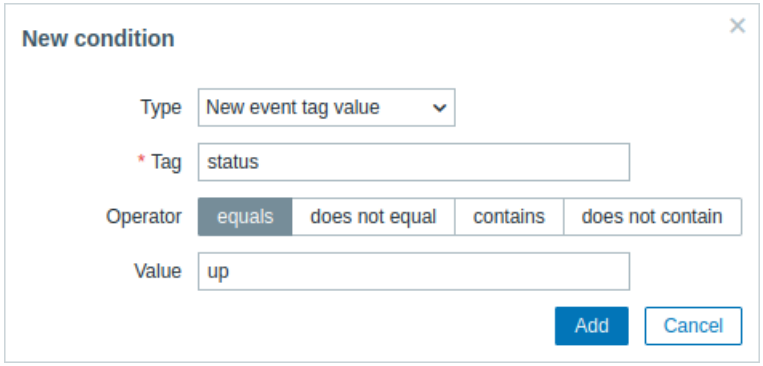
1. 转到 *Data collection* > *Event correlation*。
2. 单击右侧的 *Create event correlation*（或单击关联名称以编辑现有规则）。
3. 在表单中输入关联规则的参数。



所有必填输入字段都用红色星号标记。

Parameter	Description
Name	唯一的关联规则名称。
Type of calculation	可用的条件计算选项如下： **And** - 必须满足所有条件 **Or** - 满足其中一个条件即可 **And/Or** - 不同条件类型之间使用 AND，相同条件类型之间使用 OR **Custom expression** - 用户自定义的计算公式，用于评估动作条件。它必须包含所有条件（以大写字母 A、B、C、... 表示），并且可以包含空格、制表符、括号 ( )、**and**（区分大小写）、**or**（区分大小写）、**not**（区分大小写）。
Conditions	条件列表。有关如何配置条件的详细信息，请参见下文。
Description	关联规则描述。
Operations	勾选在事件关联时要执行的操作。可用的操作如下： **Close old events** - 当新事件发生时关闭旧事件。使用 *Close old events* 操作时，始终要添加一个基于旧事件的条件，否则可能会关闭所有现有问题。 **Close new event** - 当新事件发生时关闭该新事件。 **Warning!** 使用 *Close old events*/*Close new event* 时，不要将旧/新事件条件留空。如果在未添加与旧事件匹配的条件的情况下选择 *Close old events* 操作，Zabbix 可能会匹配所有现有旧事件并将其关闭。使用 *Close old events* 时，始终添加明确的旧事件条件（例如，*Old event tag name* 或 *Event tag pair*）。例如，仅使用 *New event condition* 和 *Close old events operation* 的规则将匹配所有符合（缺失的）旧事件条件的旧事件，从而实际上关闭旧问题。
Enabled	如果勾选此复选框，将启用该关联规则。

要配置新条件的详细信息，请单击 Conditions 区块中的 Add。将打开一个弹出窗口，您可以在其中编辑条件详细信息。



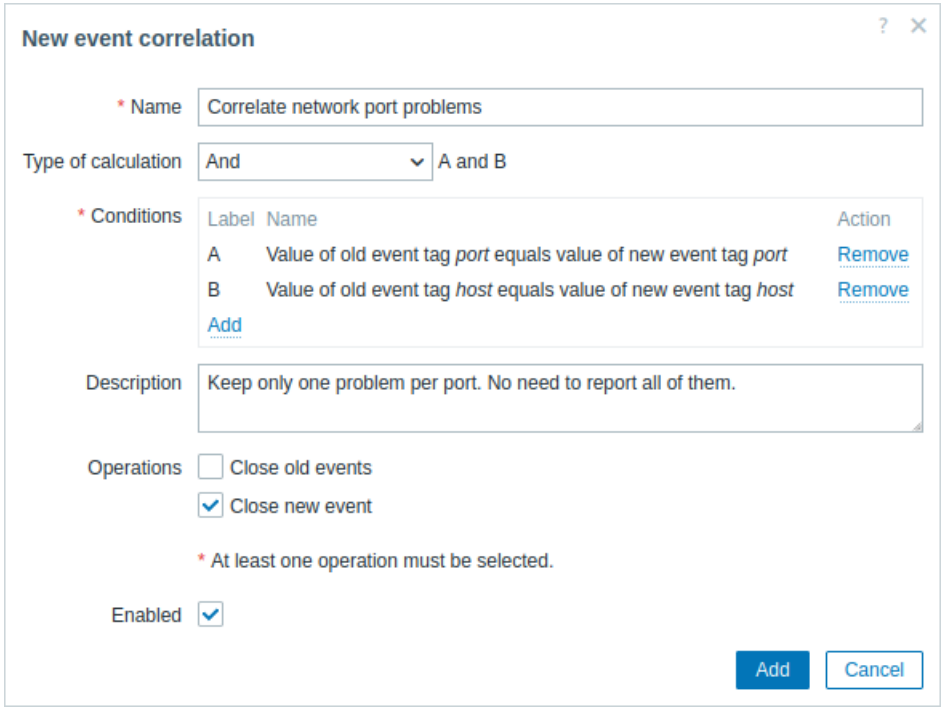
Parameter	Description
New condition	选择一个用于关联事件的条件。 注意：如果未指定旧事件条件，可能会匹配并关闭所有旧事件。同样，如果未指定新事件条件，可能会匹配并关闭所有新事件。 可用的条件如下： **Old event tag name** - 指定用于匹配的旧事件标签名称。 **New event tag name** - 指定用于匹配的新事件标签名称。 **New event host group** - 指定用于匹配的新事件主机组。 **Event tag pair** - 指定用于匹配的新事件标签名称和旧事件标签名称。在这种情况下，如果两个事件中标签的值匹配，则视为匹配。标签名称不必匹配。 此选项适用于匹配运行时值，这些值在配置时可能尚未知晓（另请参见Example）。 **Old event tag value** - 使用以下运算符指定用于匹配的旧事件标签名称和值： *equals* - 具有该旧事件标签值 *does not equal* - 不具有该旧事件标签值 *contains* - 旧事件标签值中包含该字符串 *does not contain* - 旧事件标签值中不包含该字符串 **New event tag value** - 使用以下运算符指定用于匹配的新事件标签名称和值： *equals* - 具有该新事件标签值 *does not equal* - 不具有该新事件标签值 *contains* - 新事件标签值中包含该字符串 *does not contain* - 新事件标签值中不包含该字符串

Warning:
由于可能存在配置错误，当类似的事件标签可能会为无关问题创建时，请查看下面列出的情况！

- 实际标签和标签值只有在触发器触发时才会显示。如果所使用的正则表达式无效，它会被静默替换为 *UNKNOWN* 字符串。如果带有 *UNKNOWN* 标签值的初始问题事件被遗漏，后续可能会出现具有相同 *UNKNOWN* 标签值的 OK 事件，从而关闭本不应被关闭的问题事件。
- 如果用户将 {ITEM.VALUE} 宏在不使用宏函数的情况下作为标签值，则适用 255 个字符的限制。当日志消息很长且前 255 个字符没有特定信息时，这也可能导致无关问题出现相似的事件标签。

示例

阻止来自同一网络端口的重复问题事件。



如果 *host* 和 *port* 标签值存在于触发器中，并且它们在原始事件和新事件中相同，则此全局关联规则将对问题进行关联。

该操作将关闭同一网络端口上的新问题事件，仅保留原始问题处于打开状态。

发现

概述　发现功能提供了自动添加主机并配置监控的方法，无需手动设置。Zabbix 支持多种发现方式，以适应不同场景。

发现方法

- 网络发现 - 自动检测网络中可用的主机和服务。
- 主动 agent 自动注册 - 允许主动 agent 自动向服务器注册自身。
- 低级别发现 - 使用原型和嵌套发现规则，为发现的实体（如文件系统或网络接口）自动创建监控项、触发器、图形和主机。

网络发现

概述

Zabbix 提供自动网络发现功能，既高效又非常灵活。

正确配置网络发现后，您可以：

- 加快 Zabbix 部署。
- 简化管理。
- 在快速变化的环境中使用 Zabbix，而无需过多管理工作。

Zabbix 网络发现基于以下信息：

- IP 范围
- 外部服务的可用性（FTP、SSH、WEB、POP3、IMAP、TCP 等）
- 从 Zabbix agent 接收到的信息（仅支持未加密模式）
- 从 SNMP agent 接收到的信息

它不提供网络拓扑发现。

网络发现基本上由两个阶段组成：发现和动作。

发现

Zabbix 会定期扫描网络发现规则 中定义的 IP 范围。检查频率可针对每条规则单独配置。

每条规则都定义了一组要对该 IP 范围执行的服务检查。

发现规则由发现管理器处理。发现管理器会为每条规则创建一个作业，并附带一个任务列表（网络检查）。网络检查由可用的发现 worker 并行执行（数量可在前端中针对每条规则单独配置）。例外是 SNMPv3 检查，它们由单个 worker 处理。只有具有相同 IP 和端口的检查才会按顺序调度，因为某些设备不接受在同一端口上的并行连接。

网络检查队列大小限制为 2000000，或大约 4 GB 内存。如果队列已满，则会跳过该发现规则，并在日志中输出警告信息。你可以使用 `zabbix[discovery_queue]` 内部监控项来监控队列中的发现检查数量。

发现检查与其他检查独立处理。如果某些检查未找到服务（或失败），其他检查仍会继续处理。

Note:
如果在执行过程中更改了发现规则，则当前的发现执行将被中止。

网络发现模块执行的每次服务和主机（IP）检查都会生成一个发现事件。

事件	服务检查结果
Service Discovered	服务在之前为 “down” 后变为 “up”，或首次被发现。
Service Up	服务在已经为 “up” 的情况下仍然为 “up”。
Service Lost	服务在之前为 “up” 后变为 “down”。
Service Down	服务在已经为 “down” 的情况下仍然为 “down”。
Host Discovered	当某个主机的所有服务都为 “down” 后，该主机至少有一个服务变为 “up”；或者发现了一个属于未注册主机的服务。
Host Up	当至少有一个服务已经为 “up” 后，该主机至少有一个服务变为 “up”。
Host Lost	当至少有一个服务为 “up” 后，该主机的所有服务都变为 “down”。
Host Down	当主机的所有服务已经为 “down” 后，它们仍然全部为 “down”。

操作

发现事件可以作为相关操作的基础，例如：

- 发送通知
- 添加/移除主机
- 启用/禁用主机
- 将主机添加到组
- 将主机从组中移除
- 为主机添加标签
- 移除主机标签
- 将模板链接到主机/将模板与主机取消链接
- 执行远程脚本

这些操作可以根据设备类型、IP、状态、正常运行时间/停机时间等进行配置。
有关为基于网络发现的事件配置操作的完整详细信息，请参见操作执行内容和条件页面。

由于网络发现操作是基于事件的，因此无论发现的主机在线还是离线，都会触发这些操作。
强烈建议添加一个操作条件 发现状态：在线，以避免诸如 添加主机之类的操作在 服务丢失/服务停止事件发生时被触发。
否则，如果手动移除了某个已发现的主机，它仍然会生成 服务丢失/服务停止事件，并会在下一次发现周期中被重新创建。

Note:
如果任一可链接模板中存在唯一实体（例如监控项键值），且该实体与主机上已存在的唯一实体（例如监控项键值）或另一个可链接模板中的唯一实体相同，则将模板链接到已发现主机的操作会整体失败。

主机创建

如果选择了 *Add host* 操作，则会添加一个主机。即使没有 *Add host* 操作，只要选择了会对主机执行操作的操作，也会添加一个主机。此类操作包括：

- 启用主机。
- 禁用主机。
- 将主机添加到主机群组。
- 将模板链接到主机。

创建的主机会被添加到 *Discovered hosts* 群组中（默认情况下，可在 *Administration* > *General* > *Other* 中进行配置）。如果希望将主机添加到其他群组，请添加一个 *Remove from host groups* 操作（指定”Discovered hosts”），并同时添加一个 *Add to host groups* 操作（指定另一个主机群组），因为主机必须属于某个主机群组。

系统会使用已发现设备的 IP 地址，以及发现来源（Zabbix 服务器、Zabbix proxy 或 proxy 组）和接口类型，作为在系统中查找主机的条件。如果已存在具有相同 IP 地址、接口类型和发现来源的主机，则该主机将作为执行操作的目标。如果发现来源不同，则该已发现实体会被视为不同的主机，并且可能会创建一个新主机。

如果已发现主机的 IP 地址发生变化，或者接口被删除，则在下一次发现时会创建一个新主机。

主机命名

添加主机时，主机名来自反向 DNS 查询的结果；如果反向查询失败，则使用 IP 地址。查询由 Zabbix 服务器或 Zabbix proxy 执行，具体取决于是哪一方在执行发现。如果在 proxy 上查询失败，则不会在服务器上重试。如果具有该名称的主机已存在，则下一个主机会在名称后附加 **_2**，然后是 **_3**，依此类推。

也可以覆盖 DNS/IP 查询，改为使用某个监控项的值作为主机名，例如：

- 您可以使用 Zabbix agent 监控项执行发现，发现多个运行 Zabbix agent 的服务器，并根据该监控项返回的字符串值，自动为它们分配合适的名称
- 您可以使用 SNMP agent 监控项执行发现，发现多个 SNMP 网络设备，并根据该监控项返回的字符串值，自动为它们分配合适的名称

如果主机名是使用监控项值设置的，则在后续的发现检查中不会更新。如果无法使用监控项值设置主机名，则使用默认值（DNS 名称）。

如果某个主机已存在且具有发现到的 IP 地址，并且发现源（Zabbix 服务器、proxy 或 proxy 组）未发生变化，则不会创建新主机。如果发现源不同，则会将发现到的实体视为不同对象，并且可能会创建新主机。但是，如果发现动作包含操作（链接模板、添加到主机组等），则这些操作会在与 IP 地址、接口类型和发现源匹配的现有主机上执行。

主机移除

由网络发现规则发现的主机，如果某个已发现实体不再位于该规则的 IP 范围内，则会从 *Monitoring* > *Discovery* 中自动移除。主机会被立即移除。

添加主机时的接口创建

当主机因网络发现而被添加时，会根据以下规则创建接口：

- 检测到的服务 - 例如，如果 SNMP 检查成功，则会创建一个 SNMP 接口。
- 如果主机同时对 Zabbix agent 和 SNMP 请求有响应，则会创建这两种类型的接口。
- 如果唯一性条件是 Zabbix agent 或 SNMP 返回的数据，则会将为主机找到的第一个接口创建为默认接口。其他 IP 地址将被忽略，除非默认接口宕机。如果默认接口宕机，则会使用其他 IP 地址中最小的一个作为默认接口的替代。Action 的条件（例如 Host IP）不会影响接口的添加。注意，如果所有接口都由同一发现规则发现，则此行为才会生效。如果不同的发现规则发现了同一主机的不同接口，则会添加一个额外的主机。
- 如果主机仅对 agent 检查有响应，则只会创建一个 agent 接口。如果之后它开始对 SNMP 有响应，则会添加额外的 SNMP 接口。
- 如果最初创建了 3 个独立主机，并且它们是通过 “IP” 唯一性条件发现的，随后发现规则被修改，使主机 A、B 和 C 的唯一性条件结果相同，则 B 和 C 会作为 A 的额外接口创建，A 是第一个主机。独立的主机 B 和 C 仍然保留。在 *Monitoring* > *Discovery* 中，新增的接口会显示在 “Discovered device” 列中，使用黑色字体并缩进显示，而 “Monitored host” 列只会显示 A，即第一个创建的主机。对于被视为额外接口的 IP，不会统计 “Uptime/Downtime”。

更改 proxy 设置

由不同 proxy 发现的主机并不总是被视为不同的主机。
发现和唯一性检查取决于 proxy 组结构：当某个 proxy 运行发现规则并创建主机时，该主机会被添加到该 proxy 的父 proxy 组中，而不是分配给该 proxy 本身。
当 Zabbix 在发现过程中评估 IP 唯一性时，它会检查由父 proxy 组监控的主机。
该组中由各个 proxy 监控的主机（包括运行发现的 proxy）会在唯一性检查中被忽略，这可能会在多个 proxy 监控重叠子网时导致重复主机。

虽然这种行为允许发现跨越不同子网使用的重叠 IP 范围运行，但更改已监控子网所分配的 proxy 更为复杂，因为 proxy 更改必须同时一致地应用于已发现的主机以及父 proxy 组的成员关系，以避免重复。

例如，在发现规则中替换 proxy 的步骤如下：

1. 禁用发现规则。
2. 同步 proxy 配置。
3. 替换发现规则中的 proxy。
4. 替换此规则发现的所有主机的 proxy（确保更新父 proxy 组中的主机以及该组中由各个 proxy 监控的任何主机，以避免重复）。
5. 启用发现规则。

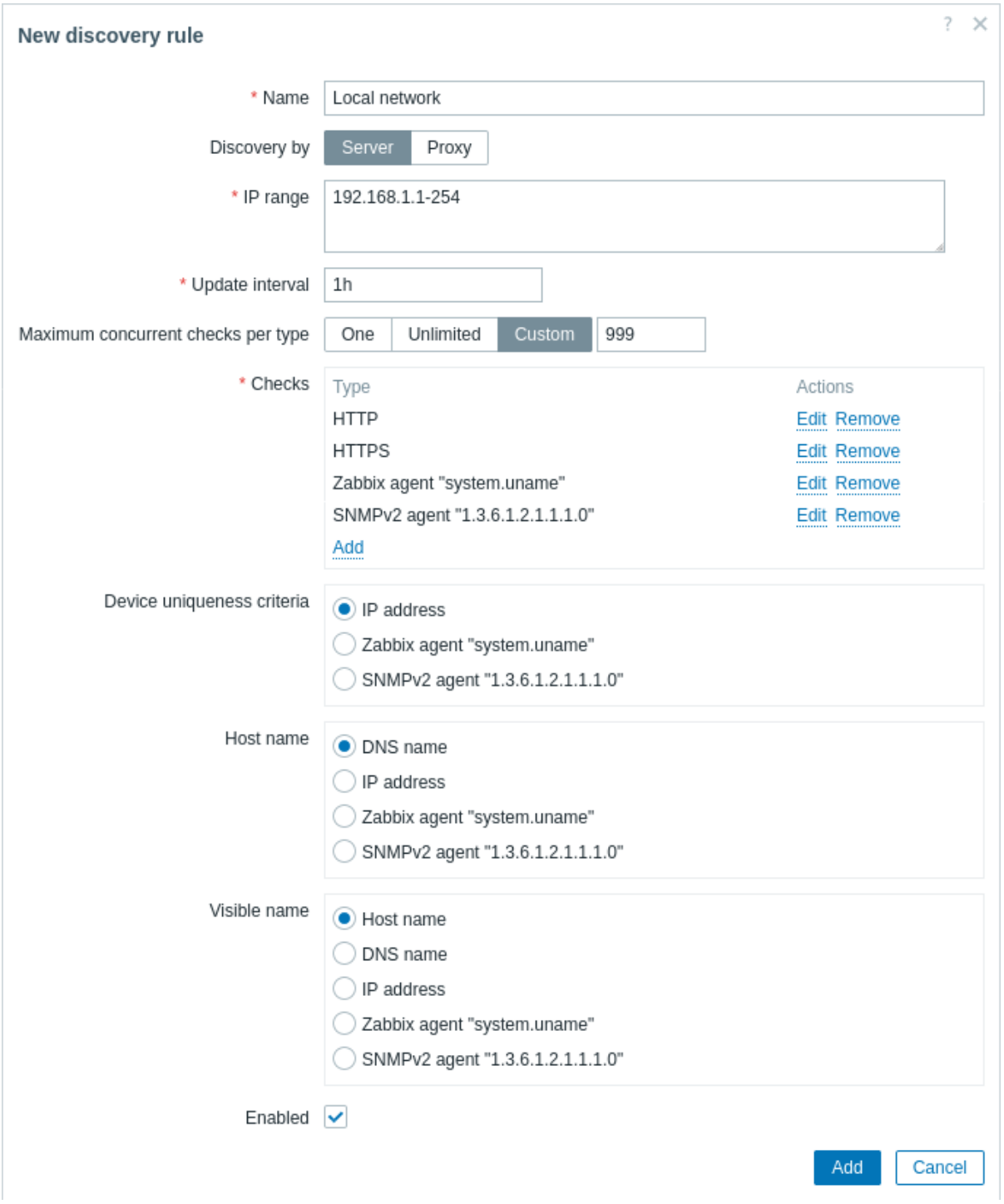
配置网络发现规则

概述

要配置 Zabbix 用于发现主机和服务的网络发现规则：

1. 转到 数据采集 > 发现。
2. 单击 创建发现规则（或单击规则名称以编辑现有规则）。
3. 编辑发现规则属性。

规则属性



所有必填输入字段都用红色星号标记。

Parameter	Description
Name	规则的唯一名称。例如，“Local network”。
Discovery by	发现由以下方式执行： **Server** - 由 Zabbix 服务器执行 **Proxy** - 由 Zabbix proxy 执行（在 proxy name 字段中选择）

Parameter	Description
IP range	用于发现的 IP 地址范围。可采用以下格式： 单个 IP：192.168.1.33 IP 地址范围：192.168.1-10.1-255。该范围受覆盖地址总数限制（少于 64K）。 IP 掩码：192.168.4.0/24 支持的 IP 掩码： /16 - /30 适用于 IPv4 地址 /112 - /128 适用于 IPv6 地址 列表：192.168.1.1-255, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 此字段支持空格、制表符和多行。
Update interval	此参数定义 Zabbix 执行该规则的频率。 该间隔从上一次发现实例执行结束后开始计算，因此不会重叠。 支持时间后缀，例如 30s、1m、2h、1d。 支持用户宏。 注意：如果使用了用户宏且其值发生变化（例如，1w → 1h），下一次检查将按照之前的值执行（按示例值会在更远的未来执行）。
Maximum concurrent checks per type	为每个服务检查设置发现线程（worker）的最大数量，以并行处理发现检查： **One** - 一个线程 **Unlimited** - 不限制线程数量（但不超过StartDiscoverers 参数中的数量） **Custom** - 设置自定义线程数（0-999）。 SNMPv1 和 SNMPv2 异步服务检查由可用 worker 并发处理。由于 libsnmp 实现的特殊性，所有使用 SNMPv3 异步服务检查的发现规则都由一个 worker 处理；因此，增加 worker 数量不会提高发现速度。
Checks	Zabbix 将使用此检查列表进行发现。点击 Add 可在弹出窗口中配置新的检查。 支持的检查：SSH、LDAP、SMTP、FTP、HTTP、HTTPS、POP、NNTP、IMAP、TCP、Telnet、Zabbix agent、SNMPv1 agent、SNMPv2 agent、SNMPv3 agent、ICMP ping。 基于协议的发现使用 **net.tcp.service[]** 功能测试每个主机，SNMP 除外，因为 SNMP 会查询 SNMP OID。Zabbix agent 通过以未加密模式查询监控项进行测试。更多详情请参见agent 监控项。 “Ports” 参数可以是以下之一： 单个端口：22 端口范围：22-45 列表：22-45,55,60-70 自 Zabbix 7.0 起，除 LDAP 检查外，所有服务检查都以异步方式执行。 自 Zabbix 7.0 起，HTTP/HTTPS 检查通过 libcurl 完成。如果 Zabbix 服务器/proxy 编译时未包含 libcurl，则 HTTP 检查将像以前版本一样工作（即作为 TCP 检查），但 HTTPS 检查将无法工作。
Device uniqueness criteria	唯一性判定标准可以是： **IP address** - 不处理多个单 IP 设备。如果已存在具有相同 IP 的设备，则会被视为已发现，不会添加新的主机。 **<discovery check>** - Zabbix agent 或 SNMP agent 检查。 请注意，发现过程中使用的唯一性判定标准与执行动作时系统中的主机标识并不相同。发现过程中的唯一性判定标准用于定义两个或多个已发现设备是否相同（或不同），而在 Zabbix 中用于主机标识的标准只有 IP 地址（请参见主机创建）。
Host name	使用以下方式设置新建主机的技术主机名： **DNS name** - DNS 名称（默认） **IP address** - IP 地址 **<discovery check>** - 发现检查返回的字符串值（例如，Zabbix agent、SNMP agent 检查） 另请参见：主机命名。
Visible name	使用以下方式设置新建主机的可见名称： **Host name** - 技术主机名（默认） **DNS name** - DNS 名称 **IP address** - IP 地址 **<discovery check>** - 发现检查返回的字符串值（例如，Zabbix agent、SNMP agent 检查） 另请参见：主机命名。
Enabled	勾选后，该规则处于活动状态，并将由 Zabbix 服务器执行。 未勾选时，该规则不活动，不会执行。

超出文件描述符限制

在并发检查数量较多的情况下，可能会耗尽discovery manager 的文件描述符限制。

检测所需的文件描述符数量等于 discovery workers 数量 * 1000。默认情况下，有 5 个discovery workers，而系统的软限制大约为 1024。

如果接近此限制，Zabbix 会降低每个 worker 的默认每种类型并发检查数，并向日志文件写入警告。但是，如果用户为 *Maximum concurrent checks per type* 设置的值高于 Zabbix 计算出的值，Zabbix 将对一个 worker 使用用户定义的值。

一个真实场景

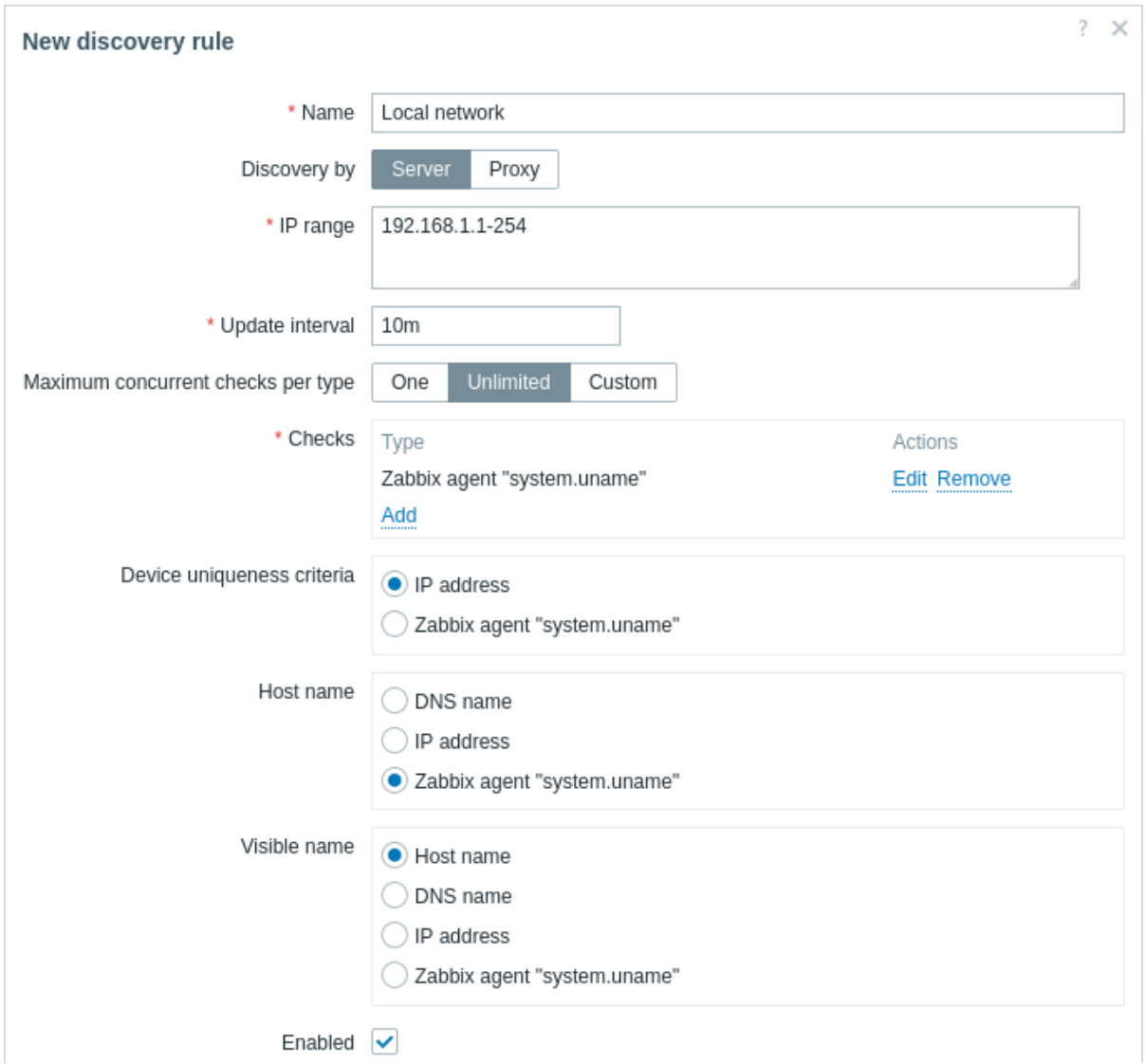
在这个例子中，会演示如何给本地 IP 地址范围 192.168.1.1-192.168.1.254 设置网络发现规则。

需要在这个场景中实现：

- 发现运行了 Zabbix agent 的主机
- 每十分钟执行一次发现
- 如果主机运行时长大于一小时，添加该主机到监控中
- 如果主机停机时长大于二十四小时，则删除主机
- 添加 Linux 主机到”Linux 服务器” 组中
- 添加 Windows 主机到”Windows 服务器” 组中
- 给 Linux 主机使用 *Linux* 模板
- 给 Windows 主机使用 *Windows* 模板

步骤 1

为我们的 IP 范围定义网络发现规则。



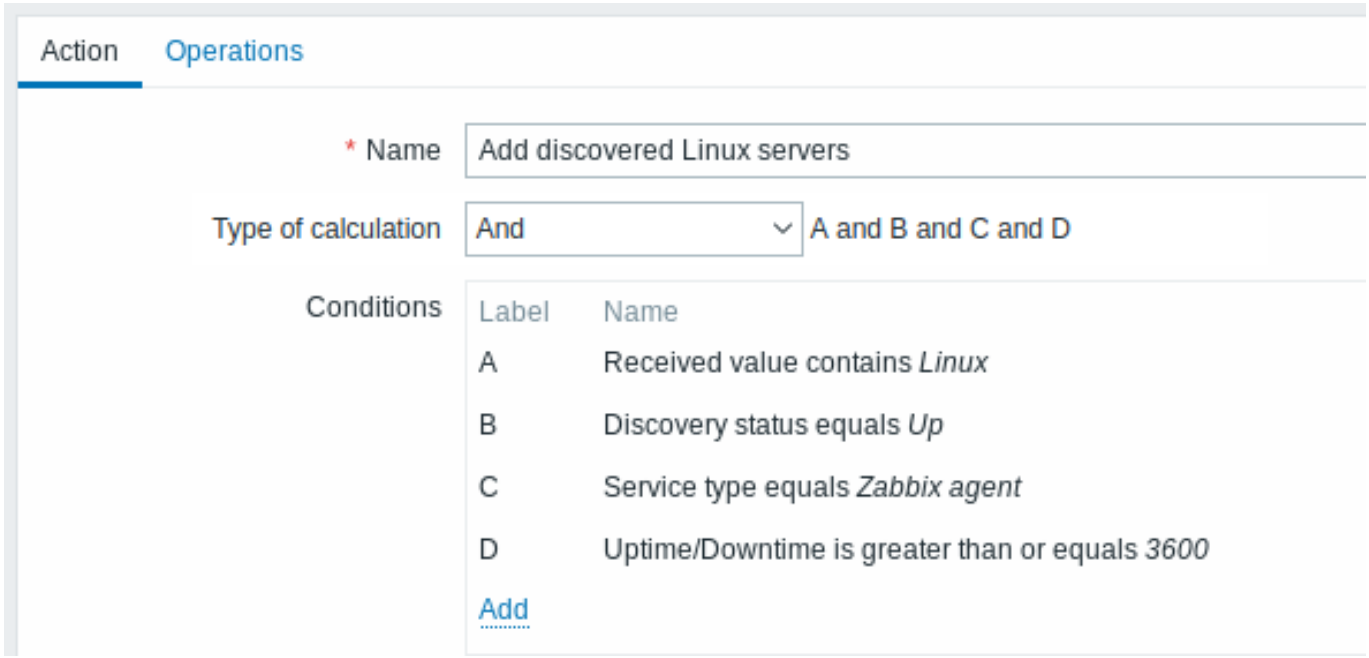
Zabbix 将尝试在 IP 范围 192.168.1.1 到 192.168.1.254 中发现主机，通过连接到 Zabbix agent 并获取 **system.uname** 键的值来完成。从 agent 收到的值可用于命名主机，并根据不同的操作系统应用不同的操作。例如，将 Windows 服务器关联到模板 *Windows*，将 Linux 服务器关联到模板 *Linux*。

该规则将每 10 分钟执行一次。

添加此规则后，Zabbix 将自动启动发现并生成基于发现的事件，以供进一步处理。

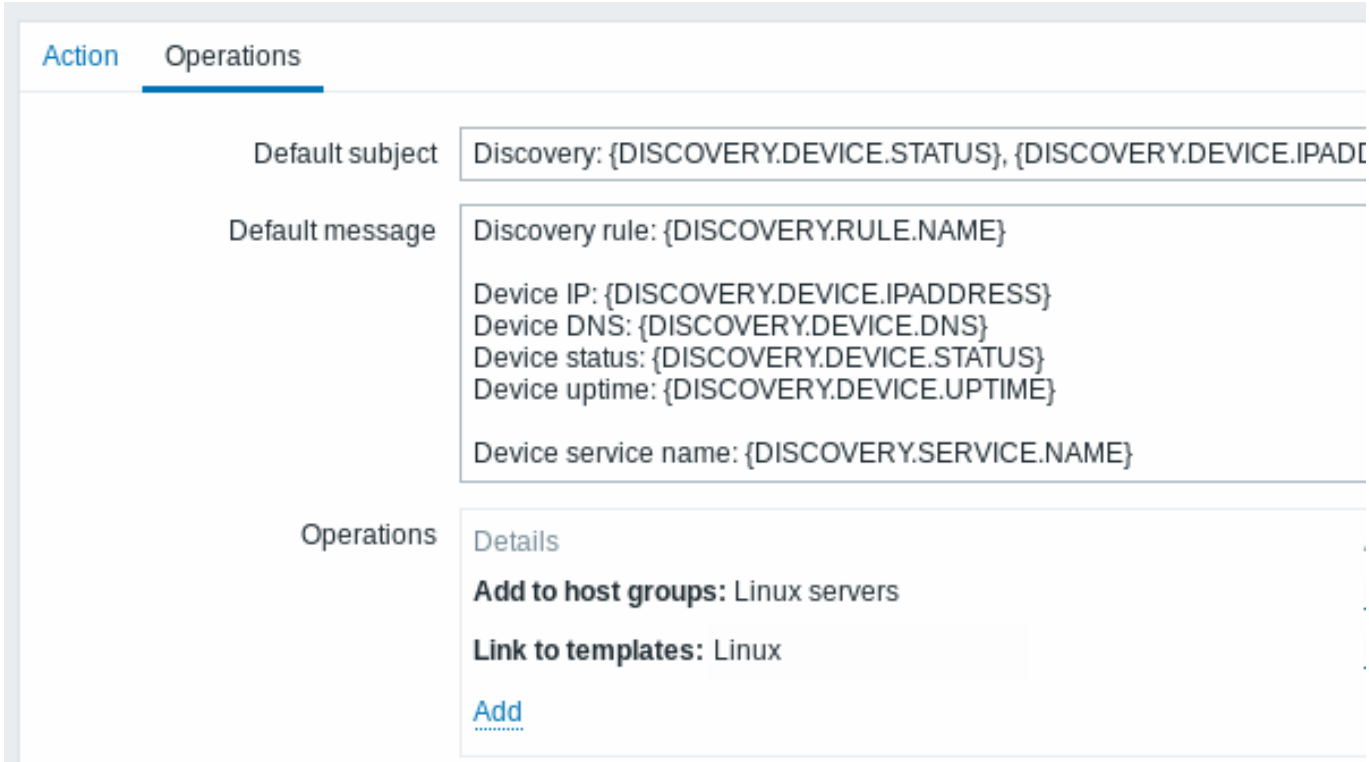
第二步

定义一个网络发现动作 ，用于添加发现的 Linux 服务器到对应的组并链接到对应模板。



同时满足下列条件才会使动作生效:

- ”Zabbix agent” 服务是已启动 (up) 状态的
- system.uname(规则里定义的 Zabbix agent 的键) 的值包含”Linux”
- 运行时长 (uptime) 大于 1 小时 (3600 秒)

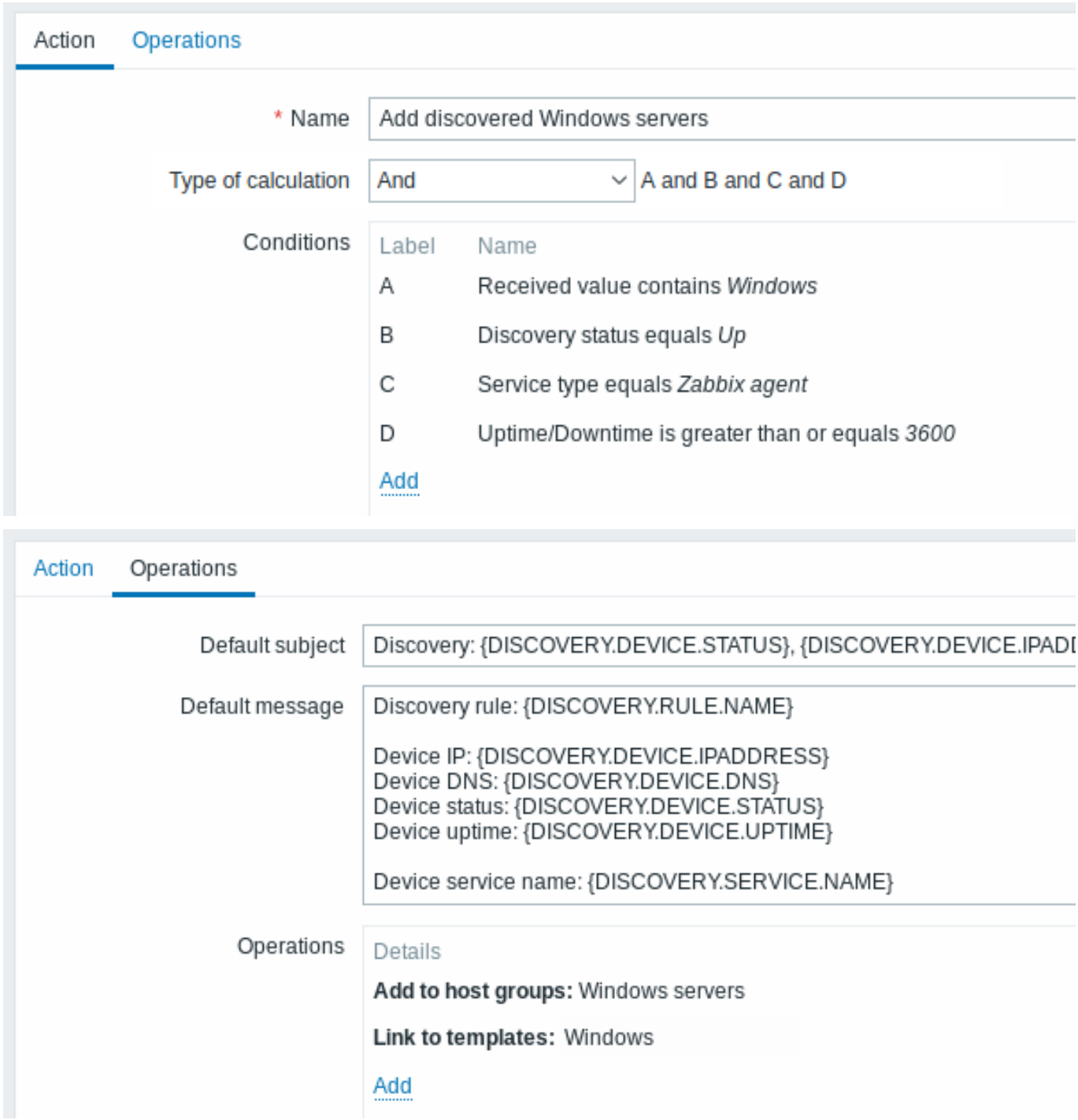


动作会执行下列操作:

- 添加发现的主机到”Linux 服务器” 组 (如果之前主机不存在，则同时创建主机)
- 链接主机到 *Linux* 模板。主机会自动纳入监控，监控使用”Linux” 模板中的监控项和触发器

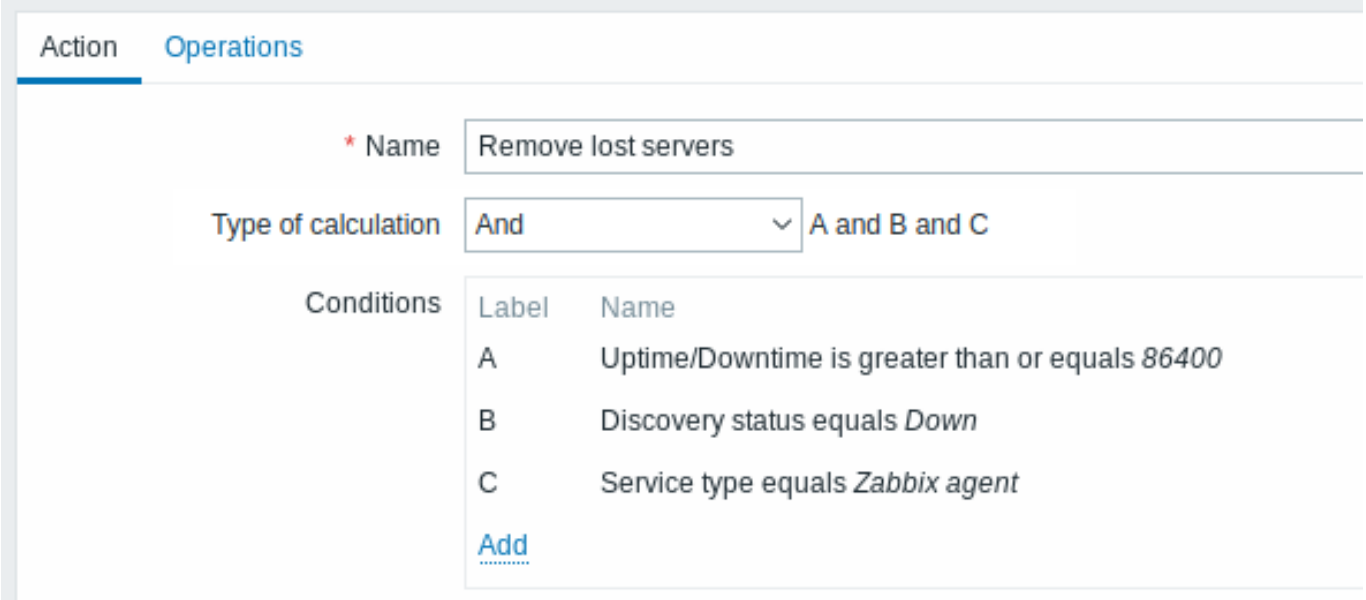
第三步

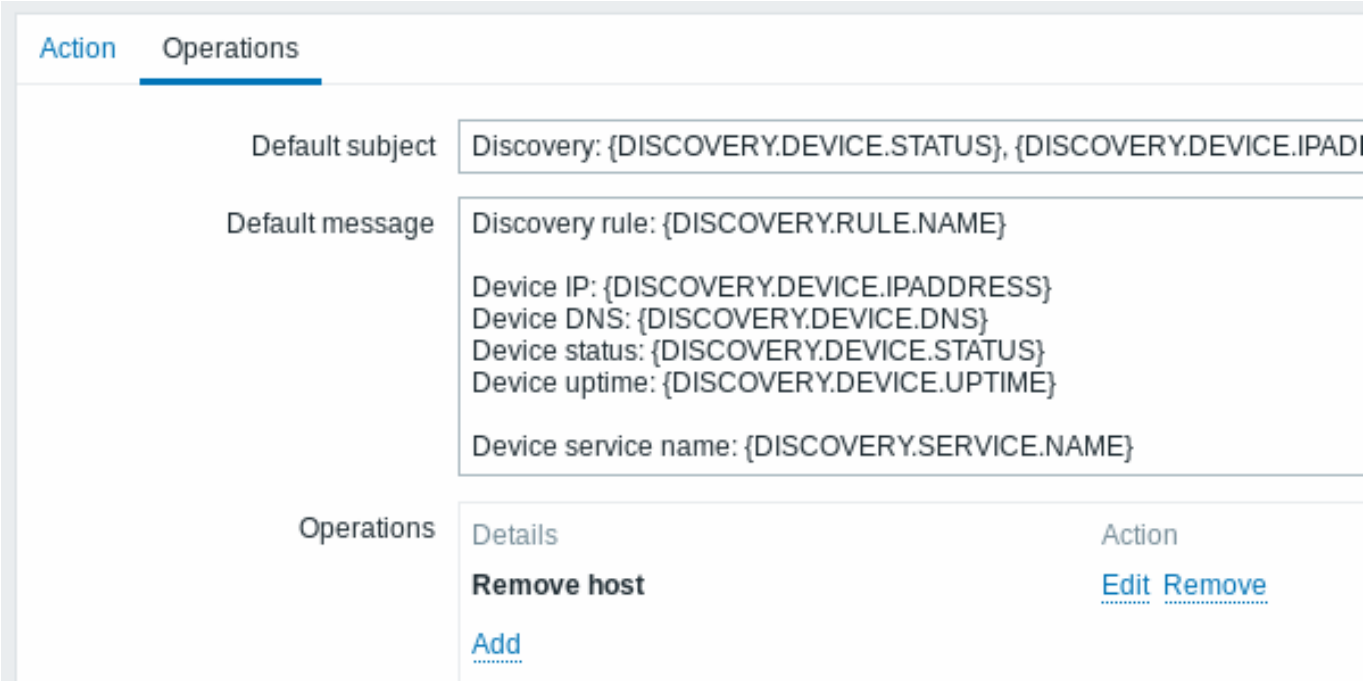
定义网络发现动作，添加发现的 Windows 服务器到对应组并链接到对应模板。



第四步

定义网络发现动作，删除丢失的 (lost) 主机。





如果”Zabbix agent” 服务停止 (down) 时间超过 24 小时 (86400 秒)，则会删除对应主机。

活动 **agent** 自动注册

概述

可以允许 active Zabbix agent 自动注册，之后服务器就可以开始监控它们。这样就无需在服务器上手动配置，即可添加新的主机进行监控。

当之前未知的 active agent 请求检查时，就会发生自动注册。

此功能对于自动监控新的云节点非常有用。一旦有新的节点出现在 Zabbix Cloud 中，系统将自动开始收集该主机的性能和可用性数据。

active agent 自动注册也支持对已添加主机进行被动检查监控。当 active agent 请求检查时，如果其配置文件中定义了`ListenIP`或`ListenPort` 配置参数，这些信息会一并发送到服务器。如果指定了多个 IP 地址，则会将第一个发送到服务器。

服务器在添加新自动注册的主机时，会使用接收到的 IP 地址和端口来配置 agent。如果未收到 IP 地址值，则使用入站连接所使用的 IP 地址。如果未收到端口值，则使用 10050。

可以指定主机在自动注册时使用DNS 名称 作为默认 agent 接口。

自动注册会重新运行：

- 如果主机元数据 信息发生变化：
  - 由于 HostMetadata 发生变化且 agent 已重启
  - 由于 HostMetadataItem 返回的值发生变化
- 对于手动创建但缺少元数据的主机
- 如果某个主机被手动更改为由另一个 Zabbix proxy 监控
- 如果同一主机的自动注册来自新的 Zabbix proxy

Zabbix 服务器和 Zabbix proxy 的 active agent 自动注册心跳间隔为 120 秒。因此，如果某个已发现的主机被删除，自动注册将在 120 秒后重新运行。

配置

指定服务器

请确保已在agent 配置文件（zabbix_agentd.conf）中指定 Zabbix 服务器：

```
ServerActive=10.0.0.1
```

除非您在 zabbix_agentd.conf 中明确指定 *Hostname*，否则服务器将使用 agent 所在位置的系统主机名来命名主机。在 Linux 中，可以通过运行 hostname 命令获取系统主机名。

如果在 Zabbix agent 配置中将 *Hostname* 定义为以逗号分隔的主机列表，则系统将为列出的所有主机名创建主机。

对配置文件进行任何更改后，请重启 agent。

活动 agent 自动注册的操作

当服务器收到来自 agent 的自动注册请求时，会调用一个操作。必须为 agent 自动注册配置事件源为 “自动注册” 的操作。

> **Note:**
> 无需设置网络发现即可让活动 agent 自动注册。

在 Zabbix 前端中：

1. 转到告警 > 操作。
2. 选择自动注册操作。
3. 单击创建操作。
4. 在操作选项卡中，为操作指定名称。
5. 可选：指定条件。在主机名/主机元数据的条件中，可以执行子字符串匹配或正则表达式匹配。如果要使用主机元数据条件类型，请参阅使用主机元数据。
6. 在操作选项卡中，添加相关操作，例如添加主机、添加到主机组（例如已发现的主机）、链接模板等。

> **Note:**
> 如果要自动注册的主机很可能仅支持活动监控（例如，被防火墙阻止而无法访问 Zabbix 服务器的主机），则可以创建一个专用模板，例如 *Template_Linux-active*，并将其链接到这些主机。

创建的主机会被添加到已发现的主机组中（默认情况下，可在管理 > 常规 > 其他中配置）。如果希望将主机添加到其他组，请添加一个从主机组中移除操作（指定 “已发现的主机”），同时再添加一个添加到主机组操作（指定另一个主机组），因为主机必须属于某个主机组。

安全的自动注册

通过配置基于 PSK 身份验证的加密连接，可以实现安全的自动注册。

加密级别在 管理 > 常规 > 自动注册 中进行全局配置。可以选择不加密、使用 PSK 身份验证的 TLS 加密，或同时选择两者（这样，某些主机可以不加密注册，而其他主机则通过加密连接注册）。

Zabbix 服务器会在添加主机之前验证 PSK 身份验证。如果验证成功，则会添加该主机，并将与主机的连接 设置为仅使用 “PSK”，其身份标识和预共享密钥与全局自动注册设置中的相同。

> **Attention:**
> 为确保使用 proxy 的安装环境中的自动注册安全，应启用 Zabbix 服务器与 proxy 之间的加密。

使用 DNS 作为默认接口

HostInterface 和 HostInterfaceItem 配置参数允许您在自动注册期间为主机接口指定自定义值。

更具体地说，如果主机应使用 DNS 名称而不是 IP 地址作为默认 agent 接口进行自动注册，这些参数会非常有用。在这种情况下，应将 DNS 名称指定为 HostInterface 或 HostInterfaceItem 参数的值，或使其返回 DNS 名称。如果其中一个参数的值发生变化，例如从 IP 地址变为 DNS 名称，或反之，则自动注册主机的默认接口也会相应更新。此更新将应用于现有主机，而不是创建新主机。要发送新值，必须重启 agent，使其重新启动自动注册过程。

Note:
如果未配置 HostInterface 或 HostInterfaceItem 参数，则会使用 listen_dns 参数。此值通过对 agent 的 IP 地址执行反向 DNS 查询来确定。如果反向 DNS 未正确配置或返回无效名称，则可能由于接口值无效而导致自动注册信息不正确或失败。

使用主机元数据

当 agent 向服务器发送自动注册请求时，会发送其主机名。在某些情况下（例如 Amazon 云节点），仅凭主机名不足以让 Zabbix 服务器区分发现的主机。可以选择使用主机元数据，将其他信息从 agent 发送到服务器。

主机元数据在agent 配置文件（zabbix_agentd.conf）中配置。在配置文件中指定主机元数据有以下 2 种方式：

```
HostMetadata
HostMetadataItem
```

请参阅上述链接中对这些选项的说明。

HostMetadataItem 参数最多可返回 65535 个 UTF-8 码点。超过此长度的值将被截断。

请注意，在 MySQL 上，如果返回值包含多字节字符，则实际的最大字符数会更少。例如，仅包含 3 字节字符的值总长度将限制为 21844 个字符，而仅包含 4 字节字符的值将限制为 16383 个符号。

Attention:
每当 active agent 向服务器发送请求以刷新 active checks 时，都会尝试进行一次自动注册。请求之间的延迟由 agent 的RefreshActiveChecks 参数指定。agent 重启后会立即发送第一个请求。

示例

使用 HostMetadata 按操作系统自动注册

假设您希望由 Zabbix 服务器自动注册主机。您的网络中部署了主动式 Zabbix agent（参见上文 “Configuration” 部分）。您的网络中既有 Windows 主机，也有 Linux 主机，并且您的 Zabbix 前端中提供了 “Linux by Zabbix agent” 和 “Windows by Zabbix agent” 模板。因此，在主机注册时，您希望将相应的 Linux/Windows 模板应用到正在注册的主机上。默认情况下，在自动注册时只会将主机名发送到服务器，这可能还不够。为了确保将正确的模板应用到主机，您应使用主机元数据。

前端配置

首先要做的是配置前端。
创建 2 个动作。
第一个动作：

- 名称：Linux 主机自动注册
- 条件：主机元数据包含 *Linux*
- 操作：链接模板：Linux by Zabbix agent

Note:
在这种情况下，您可以跳过 “添加主机” 操作。
将模板链接到主机需要先添加主机，因此服务器会自动完成该操作。

第二个动作：

- 名称：Windows 主机自动注册
- 条件：主机元数据包含 *Windows*
- 操作：链接模板：Windows by Zabbix agent

Agent 配置

现在，您需要配置 agent。将以下一行添加到 agent 配置文件中：

```
HostMetadataItem=system.uname
```

这样可以确保主机元数据将根据运行 agent 的主机包含 “Linux” 或 “Windows”。在这种情况下，主机元数据示例如下：

```
Linux: Linux server3 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2 i686 GNU/Linux
Windows: Windows WIN-OPXGGSTYNHO 6.0.6001 Windows Server 2008 Service Pack 1 Intel IA-32
```

请不要忘记在对配置文件进行任何更改后重启 agent。

使用 HostMetadata 控制自动注册和模板

步骤 ***1*** **-** 使用 ***HostMetadata*** 保护注册

使用主机元数据可提供一些基本保护，防止不需要的主机注册。

前端配置

在前端中创建一个动作，使用某个难以猜测的秘密代码来阻止不需要的主机：

- 名称：Linux 自动注册动作
- 条件：
  - 计算类型：AND
  - 条件 (A)：主机元数据包含 //Linux//
  - 条件 (B)：主机元数据包含 //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae//
- 操作：
  - 向用户发送消息：Admin，通过所有媒介
  - 添加到主机群组：Linux servers
  - 链接模板：Linux by Zabbix agent

请注意，单独使用此方法并不能提供强有力的保护，因为数据是以纯文本传输的。要使更改立即生效，需要重新加载配置缓存。

agent 配置

将以下一行添加到 agent 配置文件中：

```
HostMetadata=Linux    21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

其中，“Linux” 是平台，其余字符串是难以猜测的密文。

请不要忘记在对配置文件进行任何更改后重启 agent。

步骤 ***2*** **-** 将模板添加到已注册的主机

可以为已注册的主机添加额外的模板。
在这种情况下，MySQL by Zabbix agent 模板将仅链接到 HostMetadata 包含令牌 MySQL 的主机。

前端配置

在前端中更新该动作：

- 名称：Linux 自动注册动作
- 条件：
  - 计算类型：AND
  - 条件 (A)：主机元数据包含 Linux
  - 条件 (B)：主机元数据包含 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
  - 条件 (C)：主机元数据包含 MySQL
- 操作：
  - 向用户发送消息：Admin，通过所有媒介
  - 添加到主机群组：Linux servers
  - 链接模板：Linux by Zabbix agent
  - 链接模板：MySQL by Zabbix Agent

agent 配置

更新 agent 配置文件中的下一行：

```
HostMetadata=MySQL on Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

请不要忘记在对配置文件进行任何更改后重启 agent。

低级发现

概述　低级发现（LLD）会自动发现主机上的实体（例如文件系统、网络接口），并在无需为每个实体手动配置的情况下创建相应的监控项、触发器和图形。

要使用 LLD，您需要创建一个 发现规则，用于收集描述实体的 JSON 数据，以及针对每个实体的 原型（监控项、触发器 和图形）。

规则返回的 JSON 必须是一个对象数组，其中每个对象使用键值对表示一个已发现的实体。例如，带有 `net.if.discovery` 的发现规则可能返回：

```
[
  {"{#IFNAME}": "lo"},
  {"{#IFNAME}": "eth0"}
]
```

随后会根据这些原型创建相应的一组监控项、触发器和图形——每个接口（lo 和 eth0）各创建一组。

如果对象包含 {#MACRO} 格式的键，则这些宏会直接在原型中使用。如果没有，或者需要额外/自定义宏，则可以手动定义宏，并使用 JSONPath 将其映射到 JSON 字段。

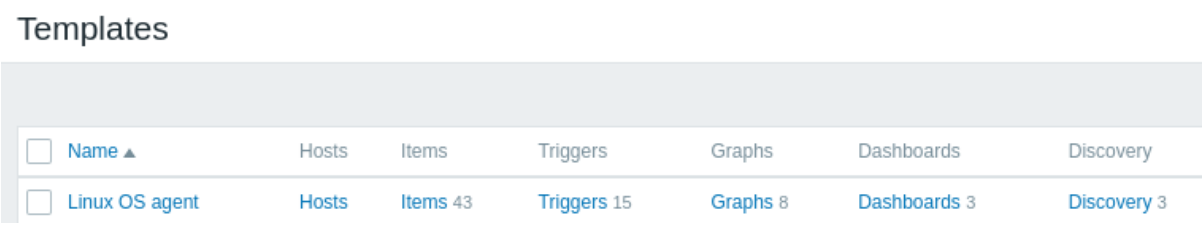
LLD 还可以创建主机（例如，在超管上发现的虚拟机），并支持用于多级发现的嵌套发现规则。

不再被发现的实体可以自动禁用或删除。

配置低级别发现　我们将以文件系统发现为例来说明低级别发现。

要配置发现，请执行以下操作：

1. 转到 *Data collection* > *Templates* 或 *Hosts*。
2. 在相应模板/主机所在行中单击 *Discovery*。



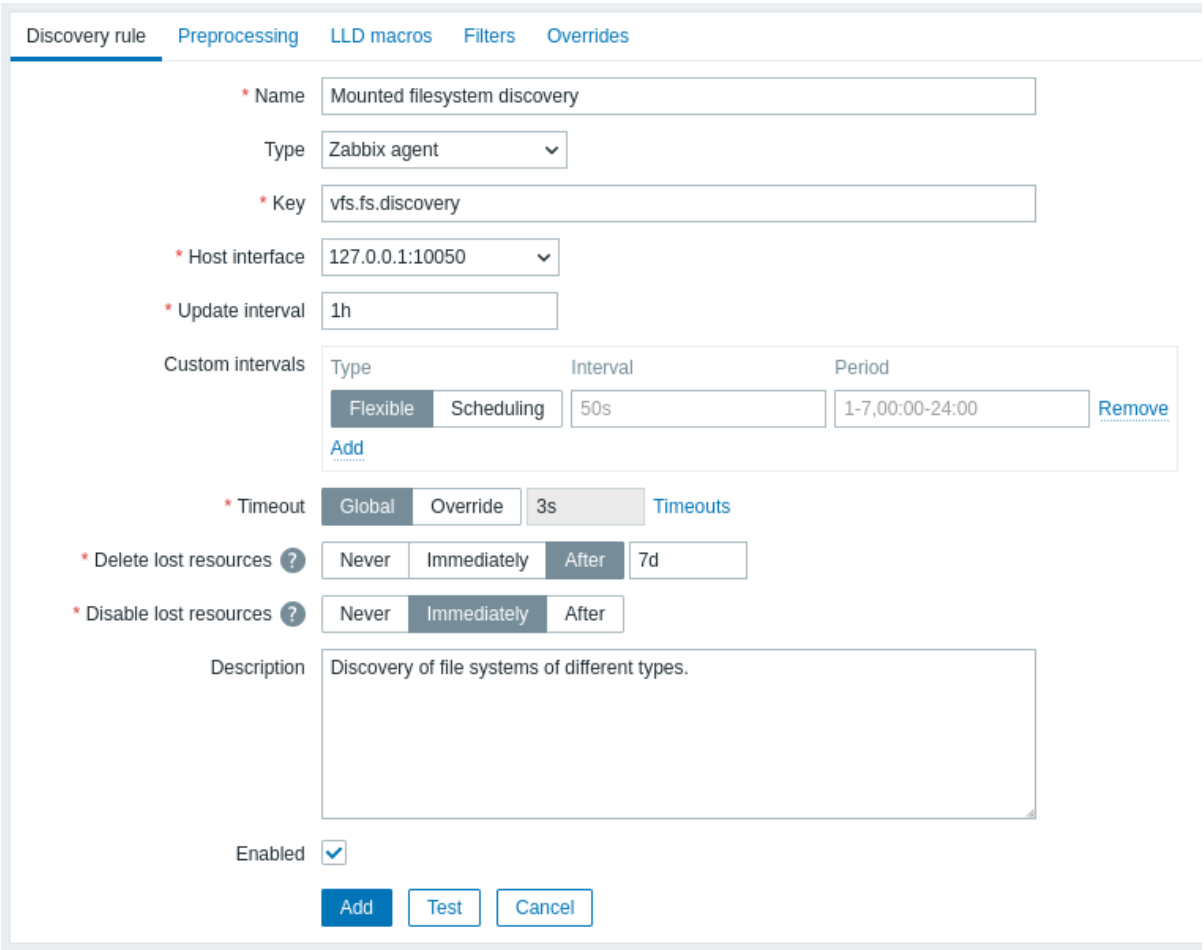
3. 单击屏幕右上角的 *Create discovery rule*。
4. 在发现规则表单中填写所需的详细信息。

发现规则

发现规则表单包含五个选项卡，从左到右分别表示发现期间的数据流：

- 发现规则 - 主要用于指定内置监控项或自定义脚本，以获取发现数据。
- 预处理 - 对发现的数据应用预处理。
- *LLD* 宏 - 提取宏值，以便在发现的监控项、触发器等中使用。
- 过滤器 - 对发现的值进行过滤。
- 覆盖 - 在应用于特定发现对象时，修改监控项、触发器、图形或主机原型。

发现规则选项卡包含用于发现的监控项键值（以及一些常规发现规则属性）：



所有必填输入字段均以红色星号标记。

参数	描述
名称	发现规则的名称。
类型	执行发现时所使用的检查类型。 在此示例中，我们使用 *Zabbix agent* 监控项类型。 发现规则也可以是依赖于常规监控项的依赖监控项。它不能依赖于另一个发现规则。对于依赖监控项，请选择相应的类型（依赖监控项），并在 “主监控项” 字段中指定主监控项。主监控项必须存在。
键值	输入发现监控项键值（最多 2048 个字符）。 例如，可以使用内置的 “vfs.fs.discovery” 监控项键值，返回包含计算机上文件系统列表、文件系统类型及挂载选项的 JSON 字符串。 请注意，文件系统发现还可以使用 “vfs.fs.get” agent 键值的发现结果（请参见示例）。
更新间隔	此字段指定 Zabbix 执行发现的频率。开始设置文件系统发现时，可能希望将其设置为较短的间隔；但确认其正常工作后，可以将其设置为 30 分钟或更长时间，因为文件系统通常不会频繁变化。 支持时间后缀，例如 30s、1m、2h、1d。 支持用户宏。 注意：仅当存在值不为零的自定义间隔时，更新间隔才能设置为 “0”。如果设置为 “0”，且存在值不为零的自定义间隔（灵活或计划），监控项将在自定义间隔的持续时间内进行轮询。 除非新发现规则具有计划或灵活更新间隔且更新间隔设置为 0，否则将在其创建后的 60 秒内检查该规则。 注意：对于现有发现规则，可以通过点击立即执行按钮立即执行发现。
自定义间隔	可以为监控项检查创建自定义规则： 灵活 - 为更新间隔创建例外（使用不同频率的间隔） 计划 - 创建自定义轮询计划。 有关详细信息，请参见自定义间隔。
超时	设置发现检查超时时间。选择超时选项： 全局 - 使用 proxy/全局超时时间（显示在灰显的超时字段中）； 覆盖 - 使用自定义超时时间（在超时字段中设置；允许范围：1 - 600s）。支持时间后缀，例如 30s、1m，以及用户宏。 点击超时链接，可以配置proxy超时或全局超时（未使用 proxy 时）。请注意，只有具有管理 > 常规或管理 > *Proxy*前端部分权限的超级管理员类型用户才能看到超时链接。
删除丢失的资源	指定发现实体的发现状态变为 “不再发现” 后，多久将其删除： 从不 - 不会删除； 立即 - 立即删除； 之后 - 在指定时间段后删除。该值必须大于禁用丢失的资源值。 支持时间后缀，例如 2h、1d。 支持用户宏。 注意：不建议使用 “立即”，因为仅仅错误地编辑过滤器，就可能导致实体及其所有历史数据被删除。 请注意，手动禁用的资源不会被低级别发现删除。
禁用丢失的资源	指定发现实体的发现状态变为 “不再发现” 后，多久将其禁用： 从不 - 不会禁用； 立即 - 立即禁用； 之后 - 在指定时间段后禁用。该值应大于发现规则的更新间隔。 请注意，如果低级别发现再次发现自动禁用的资源，这些资源将重新启用。如果再次发现手动禁用的资源，这些资源不会重新启用。 如果删除丢失的资源设置为 “立即”，则不显示此字段。 支持时间后缀，例如 2h、1d。 支持用户宏。
描述	输入描述。
启用	选中后，将处理该规则。

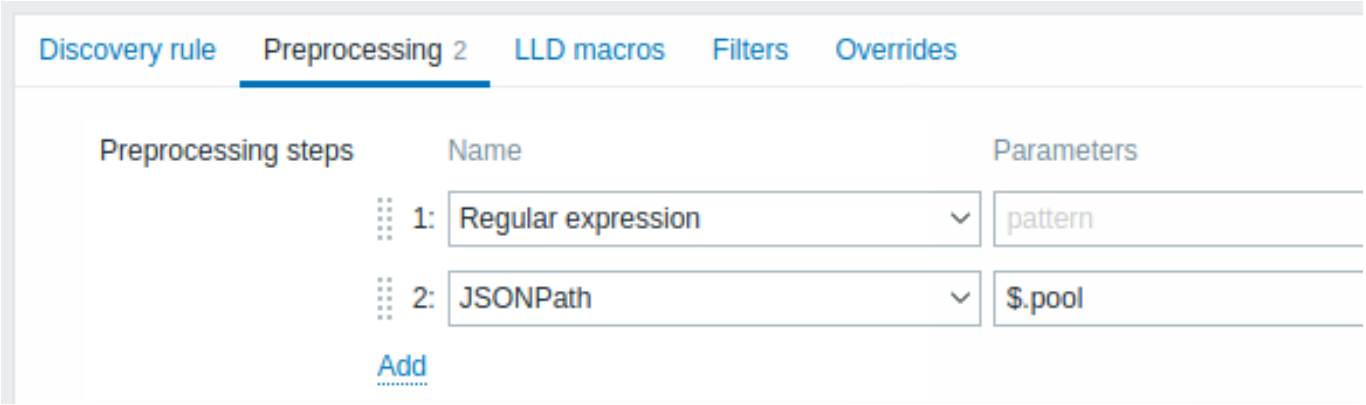
Note:
不会保留发现规则历史记录。

预处理

预处理选项卡允许您定义要应用于发现结果的转换规则。在此步骤中可以执行一个或多个转换。转换将按照定义的顺序执行。所有预处理均由 Zabbix 服务器执行。

另请参阅：

- 预处理详情
- 预处理测试



类型		
	转换	描述
文本		
	正则表达式	将接收到的值与 <pattern> 正则表达式进行匹配，并将值替换为提取出的 <output>。该正则表达式支持使用 \N 序列提取最多 10 个捕获组。 参数： **pattern** - 正则表达式 **output** - 输出格式模板。\N（其中 N=1...9）转义序列将替换为第 N 个匹配组。\0 转义序列将替换为匹配的文本。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	替换	查找搜索字符串并将其替换为其他字符串（或不替换为任何内容）。搜索字符串的所有匹配项都会被替换。 参数： **search string** - 要查找和替换的字符串，区分大小写（必填） **replacement** - 用于替换搜索字符串的字符串。替换字符串也可以为空，这样在找到搜索字符串时即可将其删除。 可以使用转义序列来搜索或替换换行符、回车符、制表符和空格”\n \r \t \s”；反斜杠可以转义为”\\”，转义序列可以转义为”\\n”。在低级别发现期间，换行符、回车符和制表符会自动进行转义。
结构化数据		
	JSONPath	使用JSONPath 功能从 JSON 数据中提取值或片段。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	XML XPath	使用 XPath 功能从 XML 数据中提取值或片段。 要使用此选项，Zabbix 服务器必须在编译时启用 libxml 支持。 示例： `number(/document/item/value)` 将从 `<document><item><value>10</value></item></document>` 中提取 10 `number(/document/item/@attribute)` 将从 `<document><item attribute="10"></item></document>` 中提取 10 `/document/item` 将从 `<document><item><value>10</value></item></document>` 中提取 `<item><value>10</value></item>` 请注意，不支持命名空间。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	CSV 转 JSON	将 CSV 文件数据转换为 JSON 格式。 有关更多信息，请参阅：CSV 转 JSON 预处理。
	XML 转 JSON	将 XML 格式的数据转换为 JSON。 有关更多信息，请参阅：序列化规则。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
SNMP		

类型		
	SNMP walk 值	根据指定的 OID/MIB 名称提取值，并应用格式化选项： 不变 - 将 Hex-STRING 作为未转义的十六进制字符串返回（注意，显示提示仍会应用）； 从 **Hex-STRING** 转换为 **UTF-8** - 将 Hex-STRING 转换为 UTF-8 字符串； 从 **Hex-STRING** 转换为 **MAC** - 将 Hex-STRING 转换为 MAC 地址字符串（其中 ' ' 将替换为 ':'）； 从 **BITS** 转换为整数 - 将以十六进制字符序列表示的位字符串的前 8 个字节（例如"1A 2B 3C 4D"）转换为 64 位无符号整数。对于长度超过 8 个字节的位字符串，将忽略后续字节。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	SNMP walk 转 *JSON*	将 SNMP 值转换为 JSON。在 JSON 中指定字段名称及其对应的 SNMP OID 路径。字段值将由指定 SNMP OID 路径中的值填充。 您可以将此预处理步骤用于SNMP OID 发现。 此步骤提供与 *SNMP walk* 值步骤类似的值格式化选项。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	SNMP get 值	对 SNMP get 值应用格式化选项： 从 **Hex-STRING** 转换为 **UTF-8** - 将 Hex-STRING 转换为 UTF-8 字符串； 从 **Hex-STRING** 转换为 **MAC** - 将 Hex-STRING 转换为 MAC 地址字符串（其中 ' ' 将替换为 ':'）； 从 **BITS** 转换为整数 - 将以十六进制字符序列表示的位字符串的前 8 个字节（例如"1A 2B 3C 4D"）转换为 64 位无符号整数。对于长度超过 8 个字节的位字符串，将忽略后续字节。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
自定义脚本		
	JavaScript	在点击参数字段或其旁边的铅笔图标时打开的模态编辑器中输入 JavaScript 代码。 请注意，可用的 JavaScript 长度取决于所使用的数据库。 有关更多信息，请参阅：Javascript 预处理
验证		
	不匹配正则表达式	指定值不得匹配的正则表达式。 例如：`Error:(.*?)\.` 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	检查 *JSON* 中的错误	检查位于 JSONPath 的应用程序级错误消息。如果检查成功且消息不为空，则停止处理；否则，继续使用此预处理步骤之前的值进行处理。请注意，这些外部服务错误会原样报告给用户，不会添加预处理步骤信息。 例如：`$.errors`。如果收到类似 `{"errors":"e1"}` 的 JSON，则不会执行下一个预处理步骤。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	检查 *XML* 中的错误	检查位于 Xpath 的应用程序级错误消息。如果检查成功且消息不为空，则停止处理；否则，继续使用此预处理步骤之前的值进行处理。请注意，这些外部服务错误会原样报告给用户，不会添加预处理步骤信息。 如果无法解析无效 XML，则不会报告错误。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
	匹配正则表达式	指定值必须匹配的正则表达式。 如果选中失败时自定义复选框，则可以指定自定义错误处理选项：丢弃该值、设置指定值或设置指定的错误消息。
限流		
	在心跳期间丢弃未更改的值	如果值在定义的时间段（以秒为单位）内未发生变化，则丢弃该值。支持使用正整数指定秒数（最小值为 1 秒）。此字段可以使用时间后缀（例如 30s、1m、2h、1d）。此字段可以使用用户宏和低级别发现宏。 对于发现监控项，只能指定一个限流选项。 例如：`1m`。如果在 60 秒内两次将相同文本传递给此规则，则该文本将被丢弃。 注意：更改监控项原型不会重置限流。只有更改预处理步骤时，限流才会重置。
Prometheus		

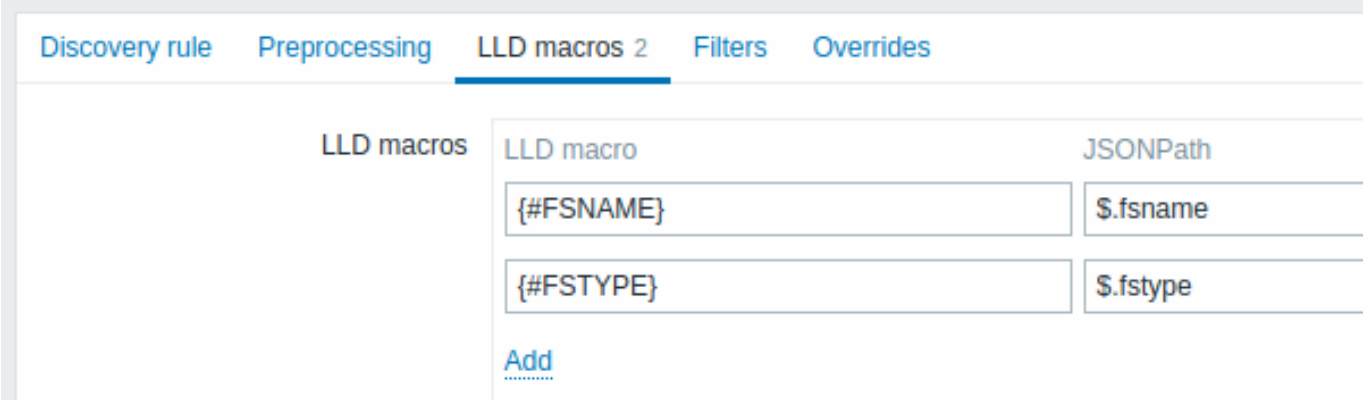
类型	
Prometheus 转 JSON	将所需的 Prometheus 指标转换为 JSON。 有关更多详细信息，请参阅Prometheus 检查。

请注意，如果发现规则通过模板应用于主机，则此选项卡的内容为只读。

自定义宏

LLD 宏选项卡允许您在返回的 JSON 尚未包含 {#MACRO} 格式的键时，指定自定义LLD 宏。

例如，vfs.fs.get agent 监控项键返回的 JSON 文件系统数据 不包含任何预定义的 LLD 宏。
 在这里，您可以手动定义这些宏，并使用JSONPath 表达式将它们映射到 JSON 中的值：



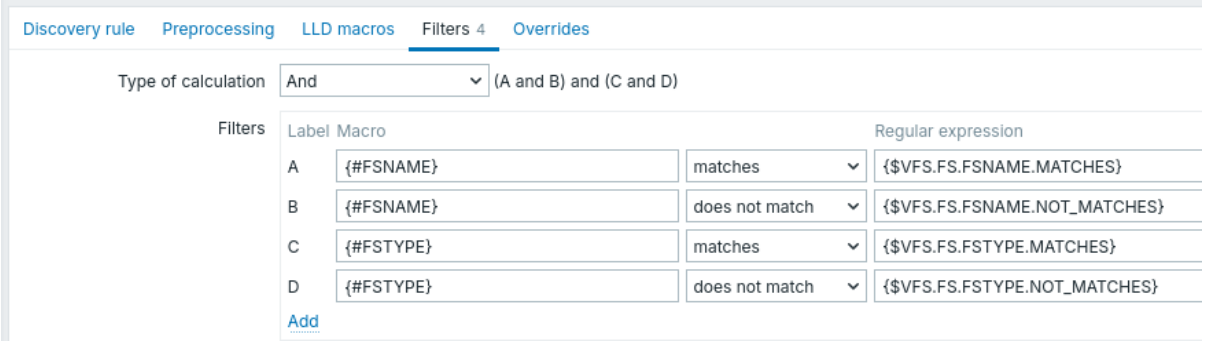
Parameter	Description
LLD macro	LLD 宏的名称，格式为 {#MACRO}。
JSONPath	定义从 JSON 数组中每个已发现实体提取哪个值的表达式。 例如，$.foo 会从 [{"foo":"bar"}, {"foo":"baz"}] 中提取 "bar" 和 "baz"。 JSONPath 同时支持点号和方括号表示法。对于包含特殊字符或 Unicode 的键，必须使用方括号表示法，例如： $['unicode + special chars #1']['unicode + special chars #2']。

发现规则必须在根级别返回一个 JSON 数组（$.）。
 为保持向后兼容，也接受包含单个 "data" 数组的 JSON 对象。在这种情况下，"data" 数组会自动展开：

```
{"data":[{"foo":"bar"}, {"foo":"baz"}]}
```

过滤器

可以使用过滤器，仅为符合条件的实体生成实际的监控项、触发器和图形。过滤器选项卡包含发现规则过滤器定义，可用于过滤发现值：



参数	描述
计算类型	可使用以下选项计算过滤器： **And** - 必须通过所有过滤器； **Or** - 只需通过一个过滤器； **And/Or** - 对不同的宏名称使用 *And*，对相同的宏名称使用 *Or*； 自定义表达式 - 可自定义过滤器的计算方式。公式必须包含列表中的所有过滤器。限制为 255 个字符。
过滤器	可使用以下过滤条件运算符：匹配、不匹配、存在、不存在。 匹配和不匹配运算符要求使用Perl 兼容正则表达式（PCRE）。 您可以在正则表达式字段中输入正则表达式，或引用全局正则表达式。 存在和不存在运算符可根据响应中是否存在指定的 LLD 宏来过滤实体。 请注意，如果响应中缺少过滤器中的某个宏，则会忽略发现的实体，除非为该宏指定了“不存在”条件。 如果宏缺失会影响表达式结果，系统将显示警告。例如，如果 {#B} 在以下表达式中缺失： {#A} 匹配 1 且 {#B} 匹配 2 - 将显示警告 {#A} 匹配 1 或 {#B} 匹配 2 - 不显示警告。

例如，如果您只关注 C:、D: 和 E: 文件系统，可以在宏字段中输入 {#FSNAME}，并在正则表达式字段中输入 ^C|^D|^E。
 也可以使用 {#FSTYPE} 宏按文件系统类型进行过滤（例如 ^ext|^reiserfs），还可以使用 {#FSDRIVETYPE} 宏按驱动器类型进行过滤（仅 Windows agent 支持），例如 fixed。

要测试正则表达式，可以使用 grep -E，例如：

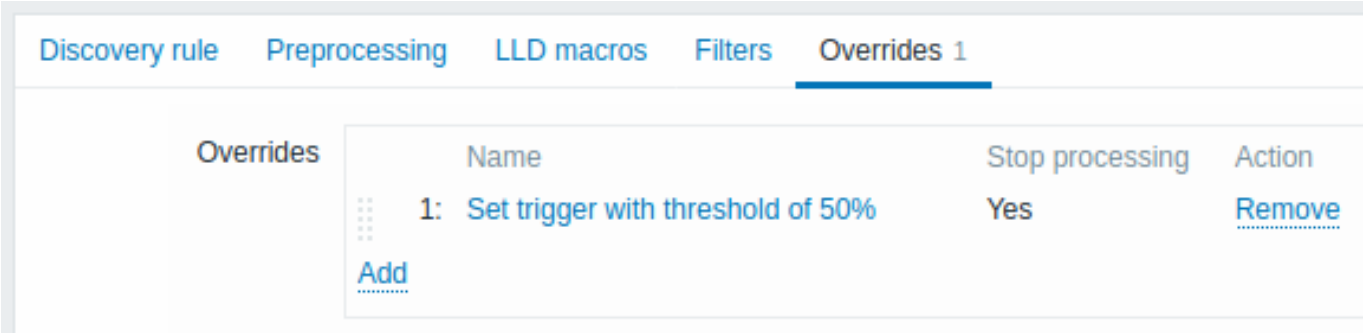
```
for f in ext2 nfs reiserfs smbfs;
do echo $f | grep -E '^ext|^reiserfs' || echo "SKIP: $f";
done
```

Warning:
LLD 规则中使用的正则表达式存在错误或拼写错误（例如，“要发现的文件系统”正则表达式不正确），可能会导致删除许多主机上的数千个配置元素、历史值和事件。

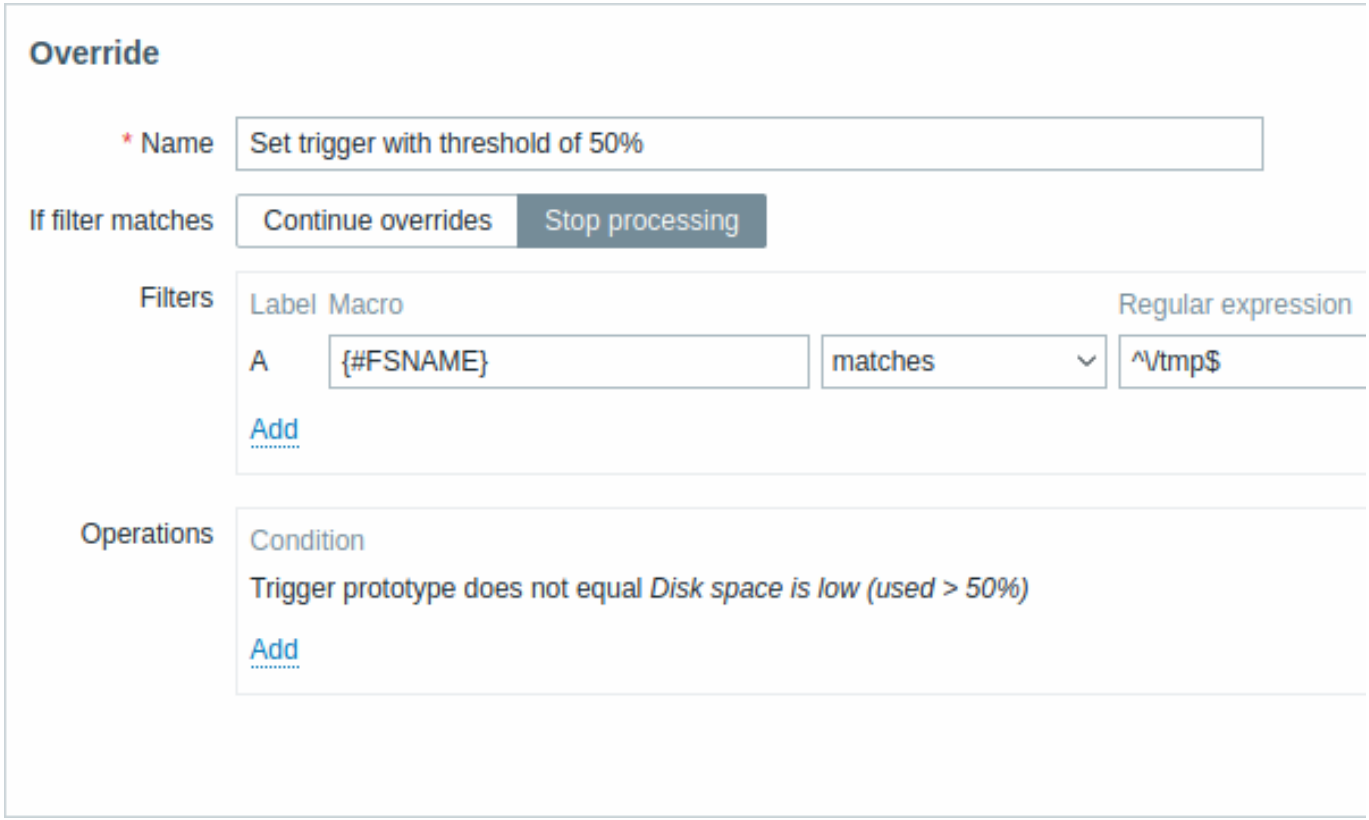
Attention:
如果要正确发现仅大小写不同的文件系统名称，则必须将 MySQL 中的 Zabbix 数据库创建为区分大小写。

覆盖

覆盖选项卡允许设置规则，以修改满足指定条件的已发现对象的监控项、触发器、图形、主机和发现原型列表或其属性。



覆盖（如果有）会显示在可通过拖放重新排序的列表中，并按照定义顺序执行。要配置新覆盖的详细信息，请在覆盖块中单击 Add。要编辑现有覆盖，请单击覆盖名称。此时将打开一个弹出窗口，您可以在其中编辑覆盖规则的详细信息。

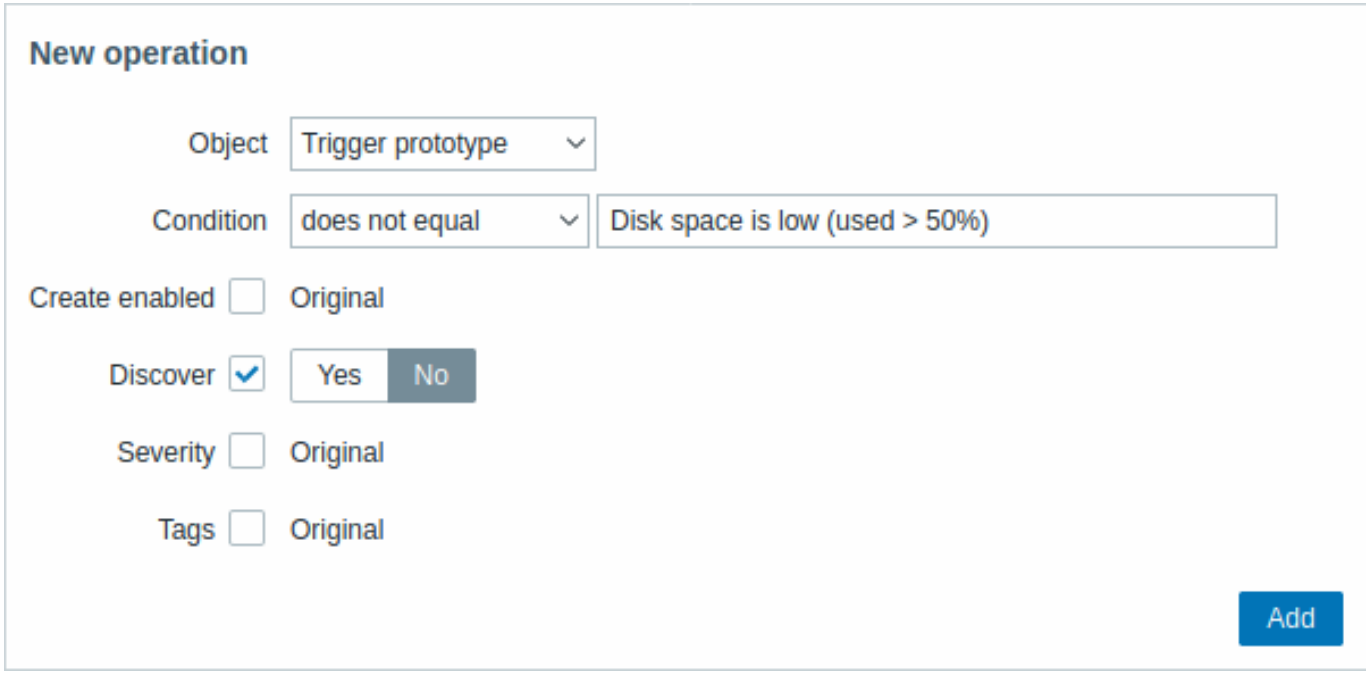


所有必填参数均以红色星号标记。

参数	描述
名称	唯一的（针对每个 LLD 规则）覆盖名称。
如果过滤器匹配	定义满足过滤器条件时是否继续处理后续覆盖： 继续覆盖 - 将处理后续覆盖。 停止处理 - 将执行前面的操作（如果有）和此覆盖，后续覆盖将对匹配的 LLD 行忽略。
过滤器	确定应将覆盖应用于哪些已发现实体。覆盖过滤器在发现规则过滤器之后处理，并具有相同的功能。
操作	覆盖操作会显示以下详细信息： 条件 - 对象类型以及对象名称必须满足的条件；例如：触发器原型不等于磁盘空间不足（已使用 > *50%*）。 操作 - 显示用于编辑和删除操作的链接。

配置操作

要配置新操作的详细信息，请在 “操作” 块中单击 Add。要编辑现有操作，请单击操作旁边的 Edit。此时将打开一个弹出窗口，您可以在其中编辑操作的详细信息。



参数			描述
对象			有五种对象类型可用： 监控项原型 触发器原型 图形原型 主机原型 发现原型
条件			允许筛选应应用操作的实体。
	运算符		支持的运算符： 等于 - 应用于此原型 不等于 - 应用于除该原型之外的所有原型 包含 - 如果原型名称包含此字符串，则应用 不包含 - 如果原型名称不包含此字符串，则应用 匹配 - 如果原型名称匹配正则表达式，则应用 不匹配 - 如果原型名称不匹配正则表达式，则应用
	模式		要搜索的正则表达式或字符串。
	对象：监控项原型		
		创建启用	选中复选框后，将显示按钮，允许您覆盖原始监控项原型设置： 是 - 监控项将以启用状态添加。 否 - 监控项将添加到已发现实体，但处于禁用状态。
		发现	选中复选框后，将显示按钮，允许您覆盖原始监控项原型设置： 是 - 将添加监控项。 否 - 不会添加监控项。
		更新间隔	选中复选框后，将显示两个选项，允许您为监控项设置不同的间隔： 延迟 - 监控项更新间隔。支持用户宏、LLD 宏和时间后缀（例如 30s、1m、2h、1d）。如果使用自定义间隔，应设置为 0。 自定义间隔 - 单击 Add 以指定灵活/计划间隔。有关详细信息，请参阅自定义间隔。
		历史数据	选中复选框后，将显示按钮，允许您为监控项设置不同的历史数据存储期限： 不存储 - 如果选择，则不会存储历史数据。 存储期限 - 如果选择，则右侧会出现用于指定存储期限的输入字段。支持用户宏和LLD 宏。

参数		描述
	趋势数据	选中复选框后，将显示按钮，允许您为监控项设置不同的趋势数据存储期限： 不存储 - 如果选择，则不会存储趋势数据。 存储期限 - 如果选择，则右侧会出现用于指定存储期限的输入字段。支持用户宏和LLD 宏。
	标签	选中复选框后，将显示一个新块，允许您指定标签-值对。支持用户宏和LLD 宏。 这些标签将追加到监控项原型中指定的标签，即使标签名称相同也是如此。
对象：触发器原型		
	创建启用	选中复选框后，将显示按钮，允许您覆盖原始触发器原型设置： 是 - 触发器将以启用状态添加。 否 - 触发器将添加到已发现实体，但处于禁用状态。
	发现	选中复选框后，将显示按钮，允许您覆盖原始触发器原型设置： 是 - 将添加触发器。 否 - 不会添加触发器。
	严重性	选中复选框后，将显示触发器严重性按钮，允许您修改触发器严重性。
	标签	选中复选框后，将显示一个新块，允许您指定标签-值对。支持用户宏和LLD 宏。 这些标签将追加到触发器原型中指定的标签，即使标签名称相同也是如此。
对象：图形原型		
	发现	选中复选框后，将显示按钮，允许您覆盖原始图形原型设置： 是 - 将添加图形。 否 - 不会添加图形。
对象：主机原型		
	创建启用	选中复选框后，将显示按钮，允许您覆盖原始主机原型设置： 是 - 主机将以启用状态创建。 否 - 主机将以禁用状态创建。
	发现	选中复选框后，将显示按钮，允许您覆盖原始主机原型设置： 是 - 将发现主机。 否 - 不会发现主机。
	链接模板	选中复选框后，将显示用于指定模板的输入字段。开始输入模板名称，或单击字段旁的选择，然后在弹出窗口的列表中选择模板。 此覆盖中的模板将添加到已链接到主机原型的所有模板中。
	标签	选中复选框后，将显示一个新块，允许您指定标签-值对。支持用户宏和LLD 宏。 这些标签将追加到主机原型中指定的标签，即使标签名称相同也是如此。
	主机资产清单	选中复选框后，将显示按钮，允许您为主机原型选择不同的资产清单模式： 禁用 - 不填充主机资产清单 手动 - 手动提供详细信息 自动 - 根据收集的指标自动填充主机资产清单数据。
对象：发现原型		
	创建启用	选中复选框后，将显示按钮，允许您覆盖原始发现原型设置： 是 - 发现规则将以启用状态添加。 否 - 发现规则将以禁用状态添加。

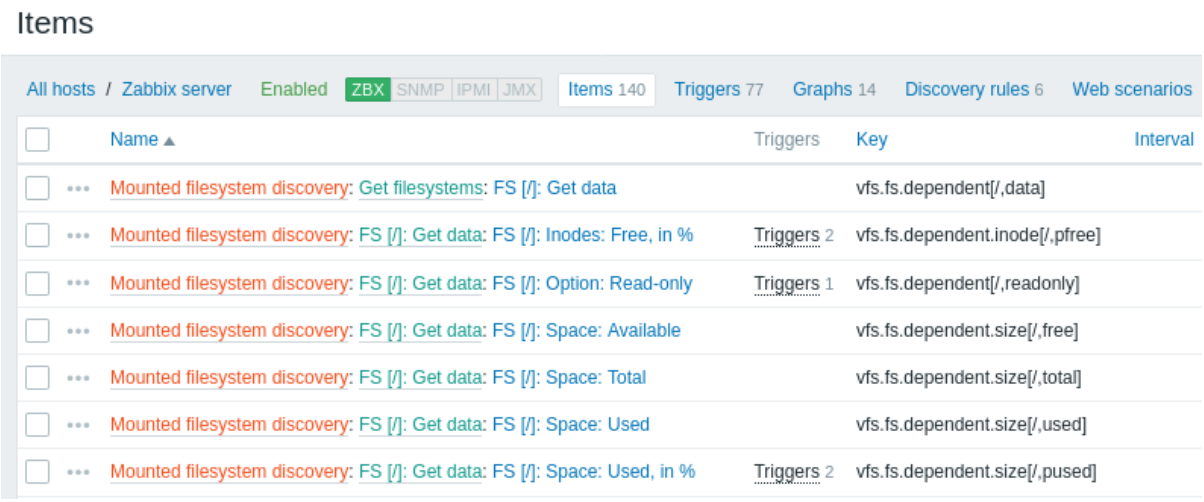
参数		描述
	发现	选中复选框后，将显示按钮，允许您覆盖原始发现原型设置： 是 - 将添加发现规则。 否 - 不会添加发现规则。
	更新间隔	选中复选框后，将显示两个选项，允许您为发现规则设置不同的间隔： 延迟 - 规则更新间隔。支持用户宏、LLD 宏和时间后缀（例如 30s、1m、2h、1d）。如果使用自定义间隔，应设置为 0。 自定义间隔 - 单击 Add 以指定灵活/计划间隔。有关详细信息，请参阅自定义间隔。

表单按钮

表单底部的按钮可用于执行多项操作。

Add	添加发现规则。此按钮仅适用于新建的发现规则。
Update	更新发现规则的属性。此按钮仅适用于现有的发现规则。
Clone	根据当前发现规则的属性创建另一个发现规则。
Execute now	立即根据发现规则执行发现。发现规则必须已存在。请参阅更多详细信息。 注意，立即执行发现时不会更新配置缓存，因此结果不会反映对发现规则配置所做的最新更改。
Test	测试发现规则配置。使用此按钮可验证配置设置（例如连接性和参数正确性），而不会永久应用任何更改。
Delete	删除发现规则。
Cancel	取消编辑发现规则属性。

已发现的实体　下面的截图展示了在主机配置中，已发现的监控项、触发器和图形的显示方式。已发现的实体前面会带有一个橙色链接，指向其来源的发现规则。



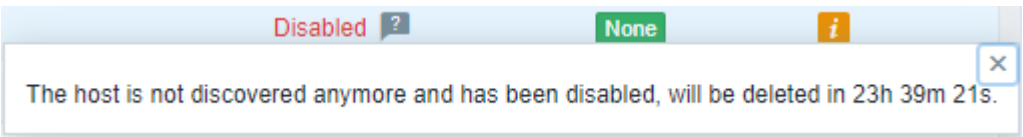
请注意，如果已经存在具有相同唯一性条件的实体，则不会创建已发现的实体，例如具有相同键值的监控项或同名图形。在这种情况下，前端会显示一条错误消息，说明低级别发现规则无法创建某些实体。不过，发现规则本身不会因为某个实体无法创建并被跳过而变为不支持状态。发现规则将继续创建/更新其他实体。

如果某个已发现的实体（主机、文件系统、接口等）不再被发现（或者不再通过过滤器），则基于它创建的实体可能会被自动禁用，并最终删除。

丢失的资源可根据 *Disable lost resources* 参数的值自动禁用。这会影响丢失的主机、监控项和触发器。

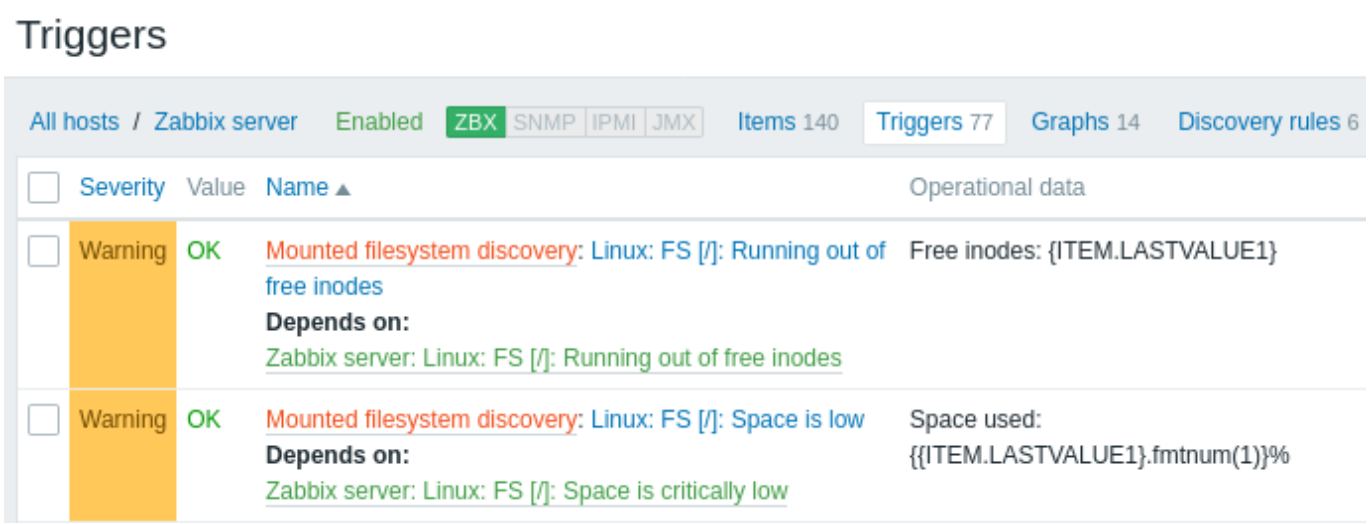
丢失的资源可根据 *Delete lost resources* 参数的值自动删除。这会影响丢失的主机、主机组、监控项、触发器和图形。

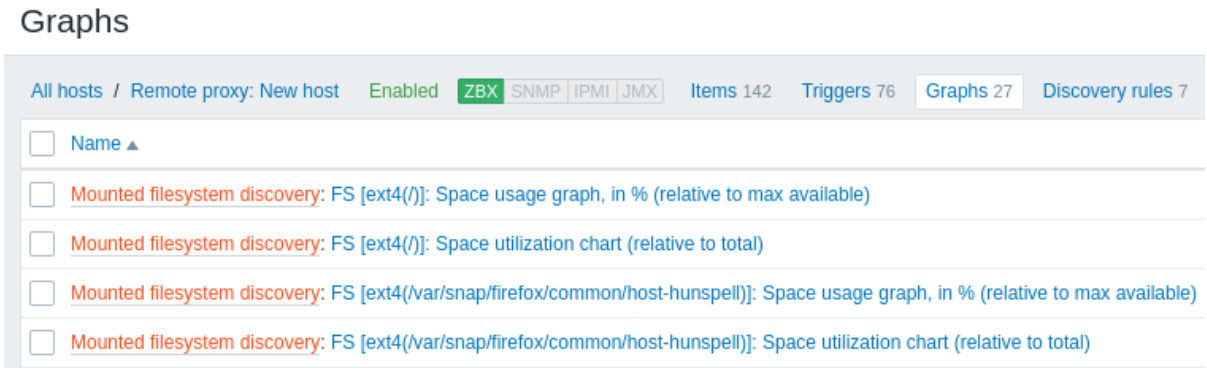
当已发现的实体变为 “Not discovered anymore” 时，实体列表中会显示一个生命周期指示器。将鼠标指针移到其上方时，会显示一条消息，指明其状态详情。



如果实体已被标记为删除，但未在预期时间删除（发现规则或监控项所属主机被禁用），则会在下次处理发现规则时将其删除。

包含其他实体且被标记为删除的实体，如果在发现规则级别发生更改，将不会更新。例如，基于 LLD 的触发器如果包含被标记为删除的监控项，则不会更新。





其他类型的发现　有关其他开箱即用发现类型的更多详细信息和操作指南，请参见以下章节：

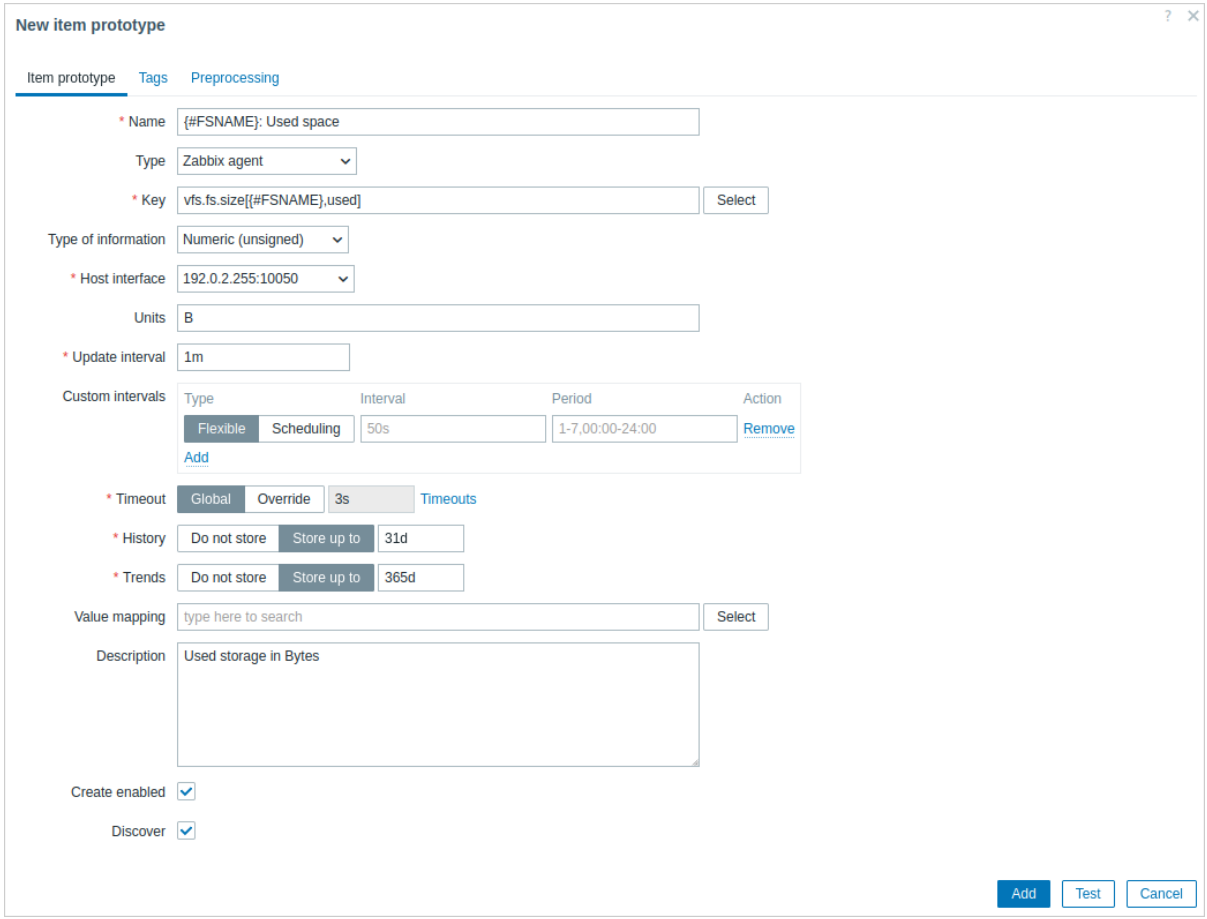
- 网络接口的发现
- CPU 和 CPU 核心的发现
- SNMP OID的发现
- JMX 对象的发现；
- 使用ODBC SQL 查询进行发现
- Windows 服务的发现
- Zabbix 中主机接口的发现

有关发现监控项的 JSON 格式的更多详细信息，以及如何将您自己的文件系统发现器实现为 Perl 脚本的示例，请参见创建自定义 LLD 规则。

监控项原型

创建规则后，进入该规则的监控项页面，点击创建监控项原型以创建监控项原型。

请注意，在需要文件系统名称的位置使用了 {#FSNAME} 宏。监控项键中必须使用低级别发现宏，以确保正确处理发现结果。处理发现规则时，此宏将替换为发现的文件系统。



Note:
当未使用监控项键参数时，请将 LLD 宏放在监控项键的方括号 `[...]` 内（例如 `v_[{#MACRO}]`）。

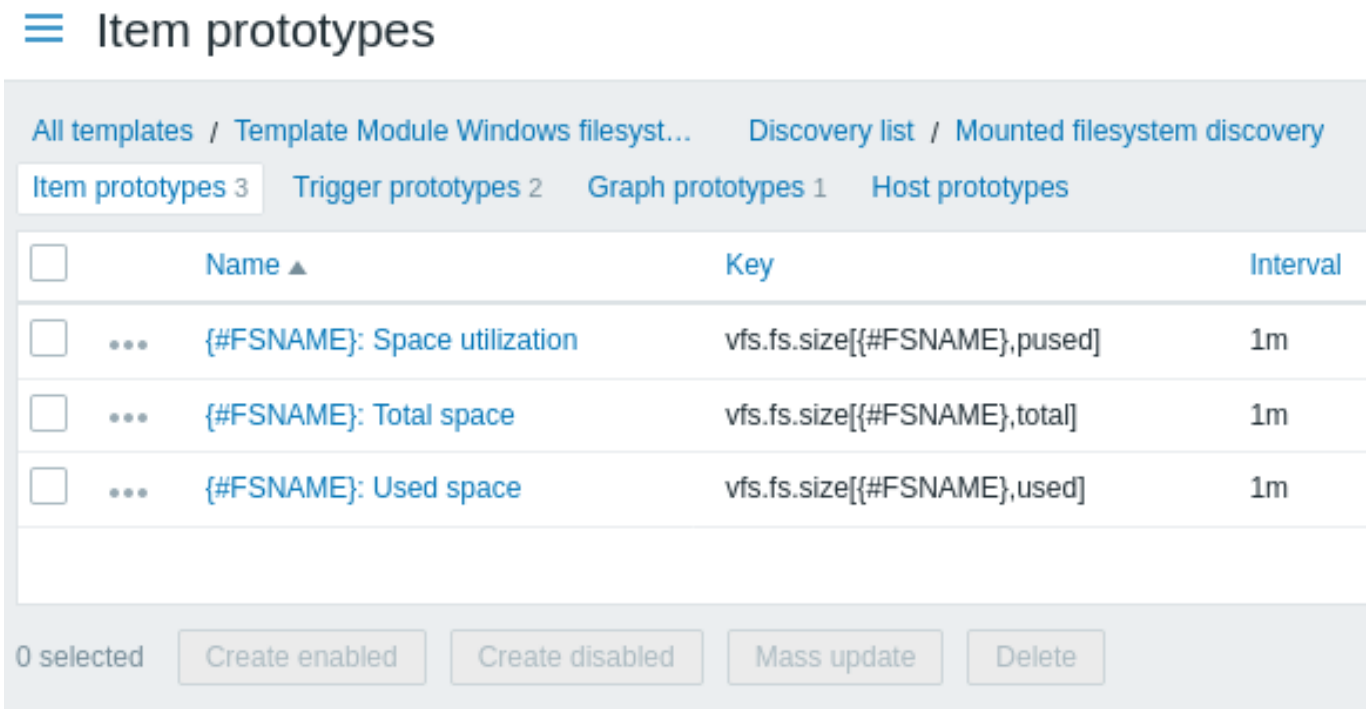
监控项原型配置以及监控项值预处理参数支持低级别发现宏和用户宏。请注意，在更新间隔中使用宏时，单个宏必须填充整个字段。不支持在一个字段中使用多个宏，也不支持将宏与文本混合使用。

Note:
系统会对低级别发现宏执行上下文相关的转义，以便安全地用于正则表达式和 XPath 预处理参数。

监控项原型的特有属性：

参数	描述
创建启用	选中后，监控项将以启用状态添加。 未选中时，监控项将添加到发现的实体，但处于禁用状态。
发现	选中时（默认），监控项将添加到发现的实体。 未选中时，除非在发现规则中覆盖此设置，否则监控项不会添加到发现的实体。

我们可以为所关注的每个文件系统指标创建多个监控项原型：



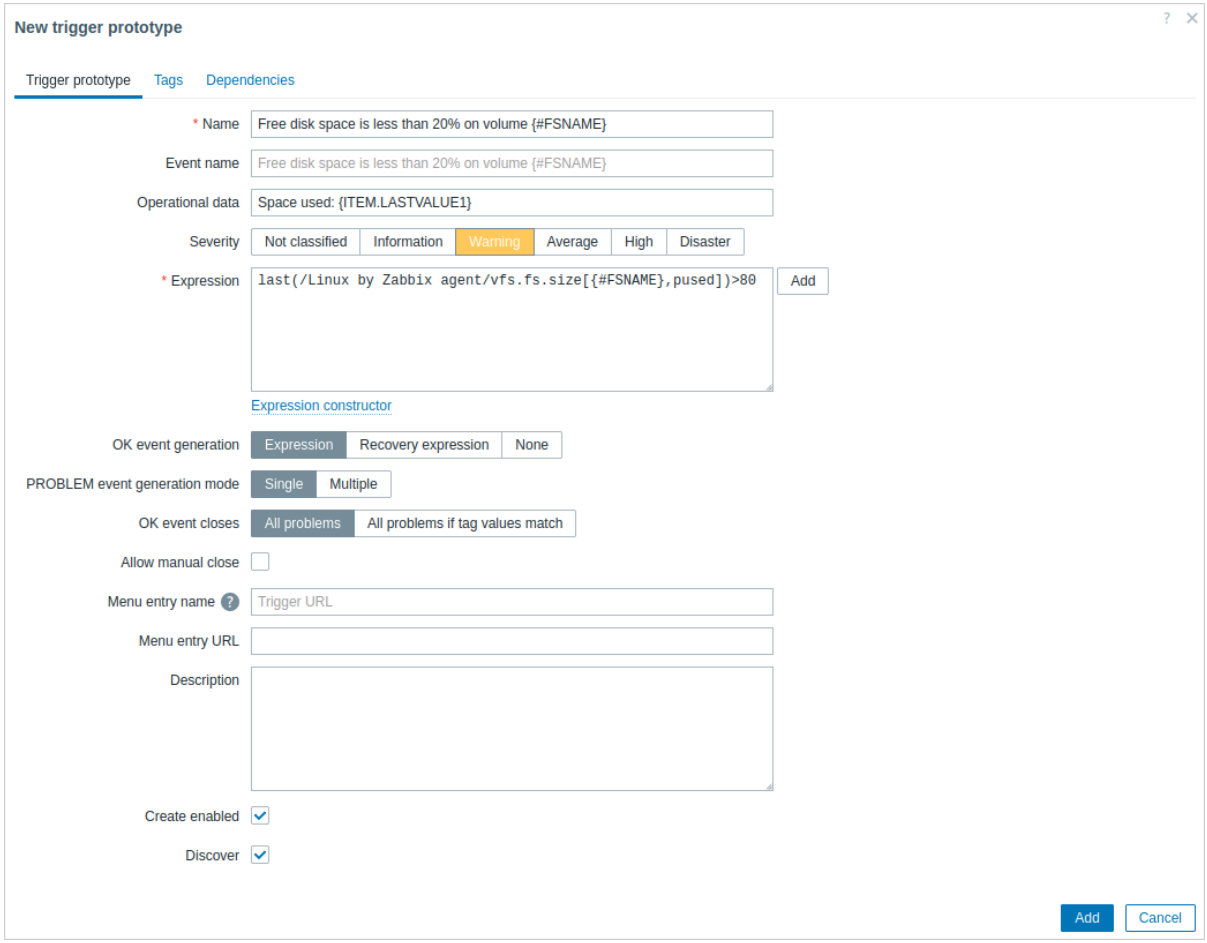
点击三点图标，打开特定监控项原型的菜单，其中包含以下选项：

- 创建触发器原型 - 基于此监控项原型创建触发器原型
- 触发器原型 - 点击查看此监控项原型中已配置触发器原型的链接列表
- 创建依赖监控项 - 为此监控项原型创建依赖监控项

如果希望一次更新多个监控项原型的属性，可以使用批量更新选项。

触发器原型

我们创建触发器原型的方式与监控项原型类似：

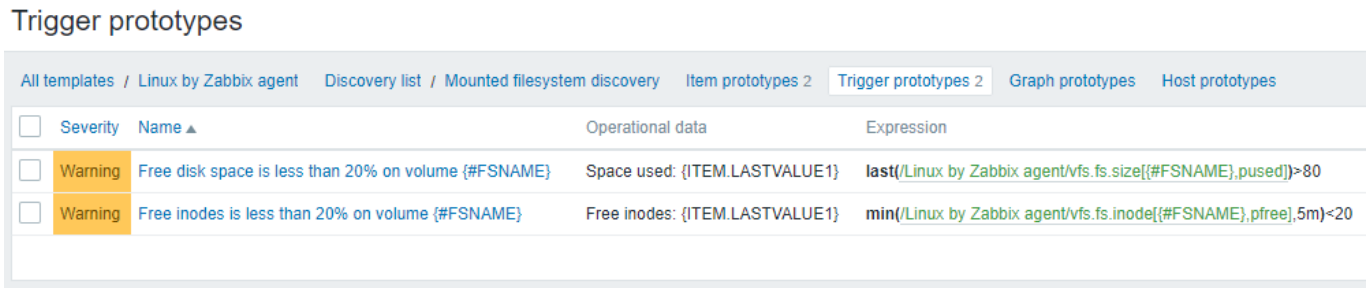


触发器原型特有的属性：

Parameter	Description
Create enabled	如果勾选，触发器将以启用状态添加。 如果未勾选，触发器仍会添加到已发现实体中，但处于禁用状态。
Discover	如果勾选（默认），触发器将添加到已发现实体中。 如果未勾选，除非在发现规则中对此设置进行了覆盖，否则不会将触发器添加到已发现实体中。

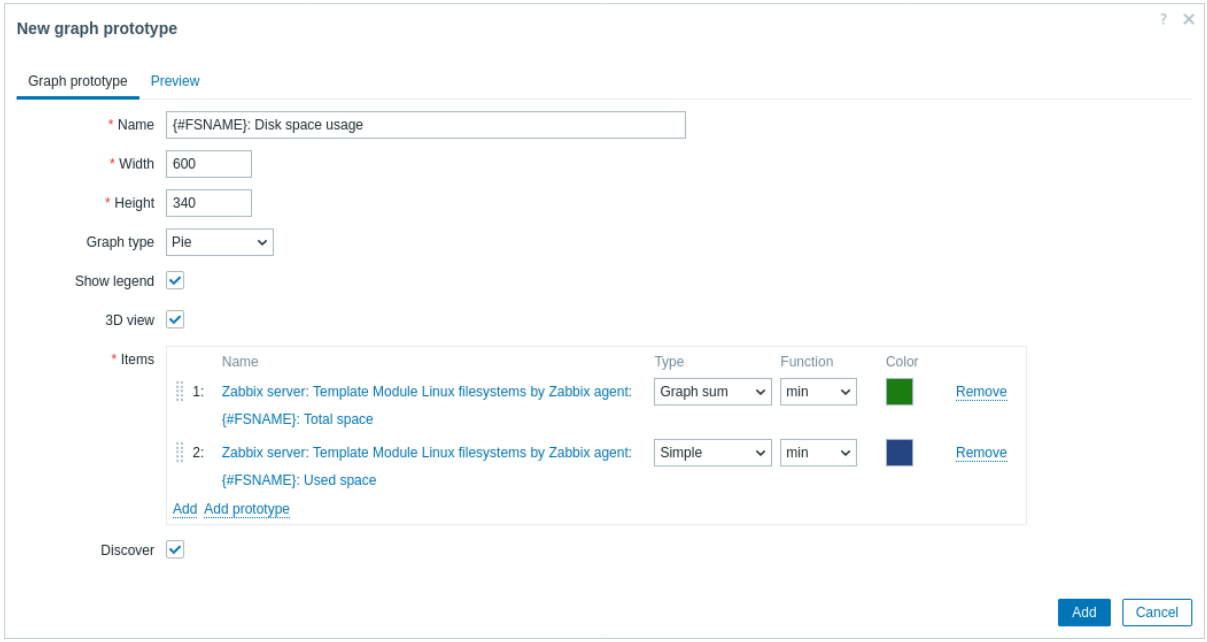
当根据原型创建实际触发器时，可能需要在表达式中用于比较的常量值（在我们的示例中为“20”）具有一定灵活性。请参阅带上下文的用户宏如何帮助实现这种灵活性。

您可以在触发器原型之间定义依赖关系。为此，请转到 *Dependencies* 选项卡。触发器原型可以依赖于同一低级别发现（LLD）规则中的另一个触发器原型，或依赖于普通触发器。触发器原型不能依赖于来自不同 LLD 规则的触发器原型，也不能依赖于由触发器原型创建的触发器。主机触发器原型不能依赖于模板中的触发器。



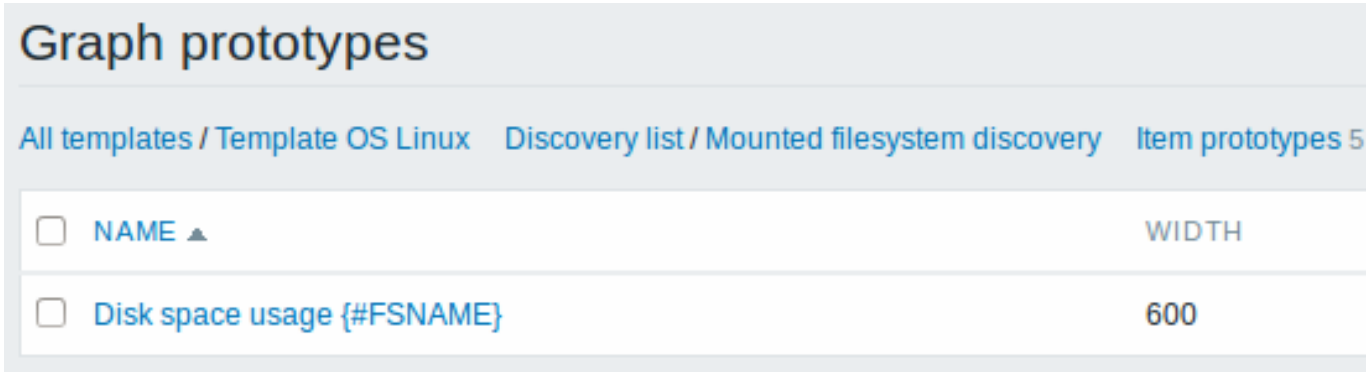
图形原型

我们也可以创建图形原型：

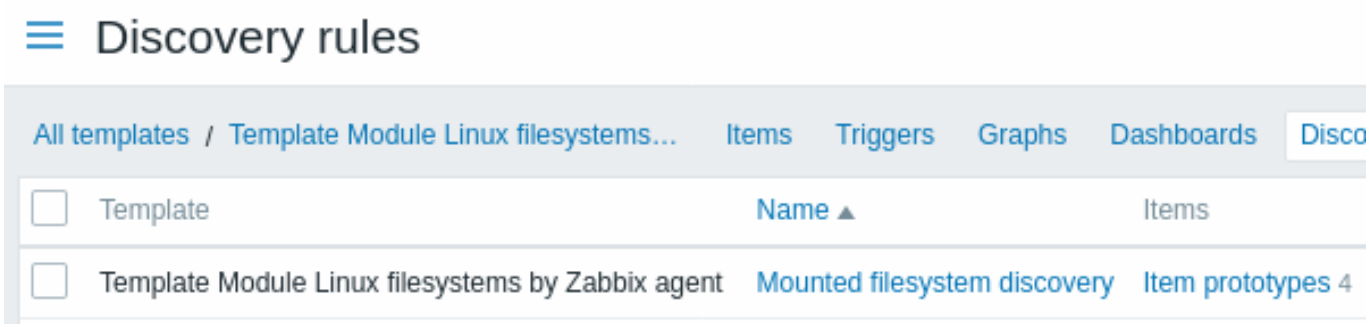


图形原型特有的属性：

Parameter	Description
Discover	如果勾选（默认），该图形将被添加到已发现的实体中。 如果未勾选，则该图形不会被添加到已发现的实体中，除非在发现规则中对此设置进行了覆盖。



最后，我们创建了一个如下所示的发现规则。它包含五个监控项原型、两个触发器原型和一个图形原型。



主机原型

主机原型是通过低级别自动发现规则 创建主机的蓝图。在被发现为主机之前，这些原型不能有监控项和触发器，除非是从模板链接的监控项和触发器。

配置

主机原型在低级别发现规则下进行配置。

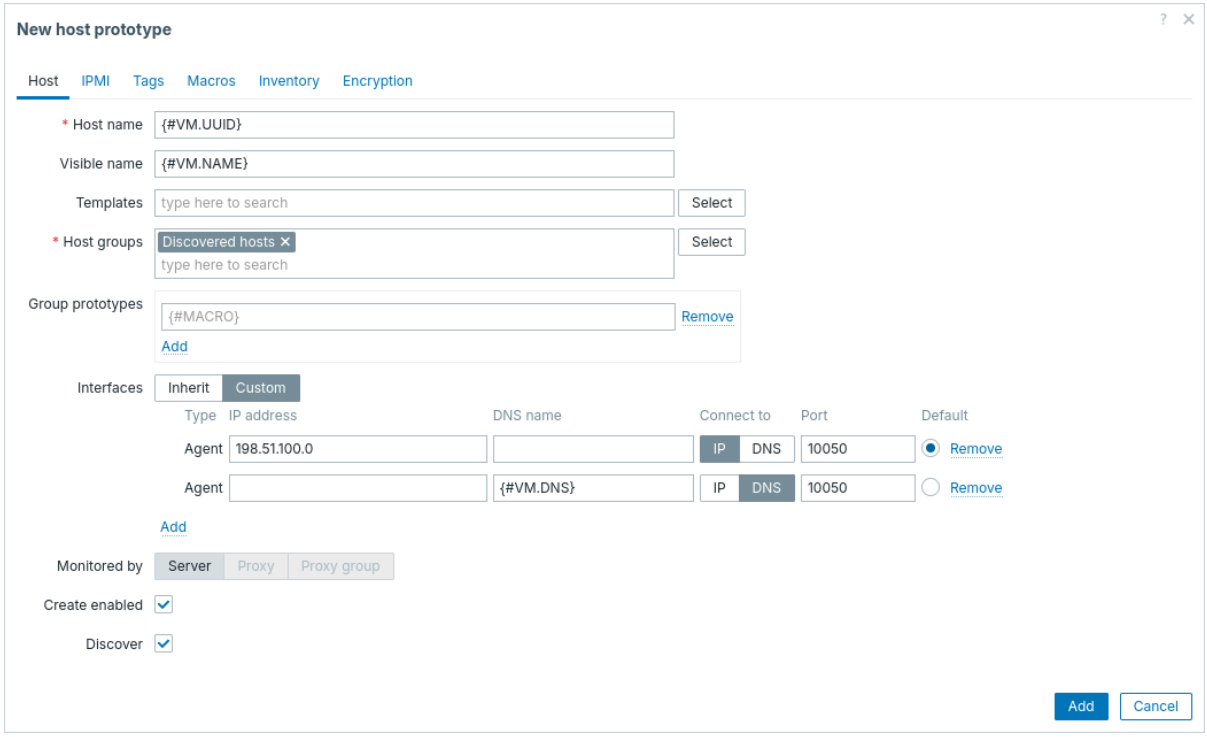
要创建主机原型：

1. 转到数据采集 > 主机。

2. 点击所需主机的 *Discovery*，以导航到为该主机配置的低级别发现规则列表。

3. 点击所需发现规则的 *Host prototypes*。

4. 点击右上角的 *Create host prototype* 按钮，并完成配置表单。



所有必填输入字段都用红色星号标记。

Parameter	Description
Host name	此参数必须至少包含一个低级别发现宏，以确保为创建的主机生成唯一的主机名。
Visible name	支持低级别发现宏。
Group prototypes	允许使用低级别发现宏指定主机组原型。 根据指定的组原型，将发现、创建并关联主机组到创建的主机；其他低级别发现规则已创建的已发现组也会关联到创建的主机。但是，与手动创建的主机组匹配的已发现主机组不会关联到创建的主机。
Interfaces	设置已发现主机是继承所属发现规则所在主机的 IP（默认），还是使用自定义接口。 支持低级别发现宏和用户宏。
Monitored by	此字段会与主机配置中的相应字段自动同步。
Create enabled	设置已发现主机的状态；如果未勾选，则主机将以禁用状态创建。
Discover	设置是否从主机原型创建主机；如果未勾选，则不会从主机原型创建主机（除非在低级别发现规则中覆盖此设置）。

Note:
低级别发现宏也支持标签值和主机原型用户宏值。
 Value maps 不支持主机原型。

有关如何配置主机原型的示例，请参见*Virtual machine monitoring*。

主机接口

要添加自定义接口，请将 *Interfaces* 选择器从 “继承” 切换为 “自定义”。点击 Add 并选择接口类型 - Zabbix agent、SNMP、JMX、IPMI。

Note:
如果选择了 自定义，但未设置任何接口，则将创建不带接口的主机。
 如果选择了 继承，并且主机原型属于某个模板，则所有发现的主机都将从链接该模板的主机继承主机接口。

如果指定了多个自定义接口，则可在 *Default* 列中设置主接口。

有关如何配置自定义主机接口的示例，请参见*VMware 监控设置示例*。

Warning:
只有当主机接口包含正确的数据时，才会创建主机。

已发现主机

在主机列表中，已发现主机会以前缀形式显示创建它们的发现规则名称。

已发现主机会从主机原型继承大多数参数，且这些参数为只读。只有以下已发现主机参数可以配置：

- 模板 - 链接新模板或取消链接手动添加的模板。无法取消链接从主机原型继承的模板。
- *Description* - 添加或编辑主机描述。
- *Status* - 启用或禁用主机。
- *Tags* - 添加新标签或移除手动添加的标签。无法移除从主机原型继承的标签。请注意，标签不能重复（名称和值都相同的标签）。如果向主机原型添加了一个新标签，并且它与某个已发现主机上手动添加的标签匹配，则在发现过程中，手动添加的标签将被继承的标签替换。
- *Macros* - 添加新的主机宏或移除手动添加的宏，以及从主机原型继承的宏；更改宏值和类型。

Note:
已发现主机还会从配置了发现规则的主机继承用户宏。这些继承的宏可以被移除，其值和类型也可以更改。

已发现主机也可以继续发现其他主机。例如，如果某个主机原型（用于发现 hypervisor）链接了一个模板，而该模板包含一条低级别发现规则及其自己的主机原型（用于发现虚拟机），那么 Zabbix 将会发现 hypervisor 及其虚拟机。若要进一步发现这些虚拟机上的容器，可以创建低级别发现规则，或为已发现的虚拟机链接一个带有主机原型的新模板，或者通过链接一个本身包含主机原型的模板来预先配置虚拟机主机原型。

已发现主机可以手动删除。但请注意，如果为其启用了发现，它们将会再次被发现。

不再被发现的主机可以：

- 自动禁用（基于发现规则的 *Disable lost resources* 值）
- 自动删除（基于发现规则的 *Delete lost resources* 值）。

发现原型

概述

发现原型是 “父级” 发现规则中的嵌套低级别发现规则，可用于对具有自身监控项、触发器等的对象进行多级发现。例如，您可能希望发现数据库服务器上的所有数据库实例，然后发现每个实例的表空间，最后发现每个表空间中的表。

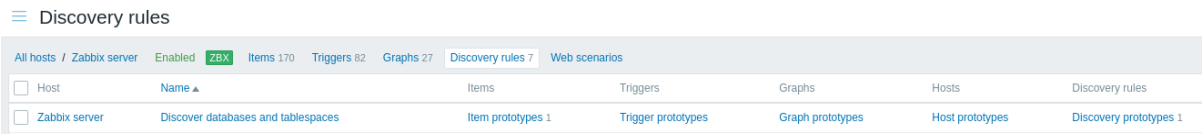
发现原型拥有自己的监控项、触发器、图形、主机和发现原型。如果指定 *Nested* 类型，嵌套发现原型将使用与父级规则相同的 JSON 值。

发现原型的嵌套层级不受限制。

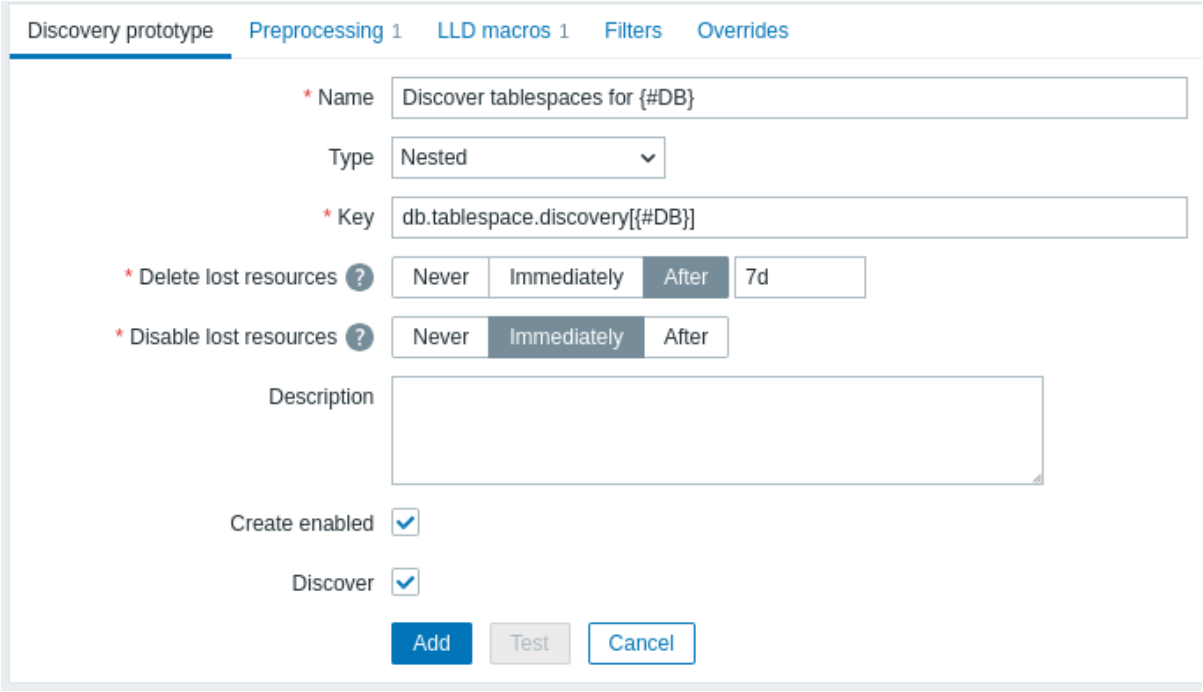
配置

要创建发现原型：

- 在现有发现规则所在行中单击 *Discovery prototypes*



- 单击 *Create discovery prototype*



此表单中的配置字段与常规的低级发现 共享。

如果你在打开的发现原型表单中将 *Type* 选择为"Nested"，则会基于与父发现规则相同的 JSON 值中的一个 JSON 对象生成发现规则（来自发现原型）。例如，如果原始 JSON 为 [<object A>, <object B>]，并且只有一个嵌套发现规则原型，那么将分别基于 object A 和 object B 的数据生成两个发现规则。

在这种情况下，发现原型会与父规则同时激活。因此，嵌套规则可以使用预处理，对父规则已获取的同一数据的另一个 "切片" 进行处理。

父 LLD 规则中的 LLD 宏可用于嵌套发现规则。

已发现主机上的嵌套 LLD 规则

嵌套低级别发现规则可用于分配给主机原型的主机模板。如果在已发现主机上存在 嵌套发现规则，则用于发现该主机的 JSON 对象也会发送到该主机上所有嵌套类型的 LLD 规则。更多详细信息，请参见示例。

创建该主机的发现规则中的 LLD 宏可用于嵌套发现规则。

示例

让我们基于接收到的以下多级 JSON 示例，说明发现原型的一种可能应用。

```
[
  {
    "database": "db1",
    "created_at": "2024-02-01T12:30:00Z",
    "encoding": "UTF8",
    "tablespaces": [
      { "name": "ts1", "max_size": "10GB" },
      { "name": "ts2", "max_size": "20GB" },
      { "name": "ts3", "max_size": "15GB" }
    ]
  },
  {
    "database": "db2",
    "created_at": "2023-11-15T08:45:00Z",
    "encoding": "UTF16",
    "tablespaces": [
      { "name": "ts1", "max_size": "5GB" },
      { "name": "ts2", "max_size": "25GB" },
      { "name": "ts3", "max_size": "30GB" }
    ]
  },
  {
    "database": "db3",
    "created_at": "2024-01-05T15:10:00Z",
```

```
    "encoding": "UTF8",
    "tablespaces": [
      { "name": "ts1", "max_size": "12GB" },
      { "name": "ts2", "max_size": "18GB" },
      { "name": "ts3", "max_size": "22GB" }
    ]
  }
]
```

案例 1

发现数据库服务器上的数据库实例，然后发现每个实例的表空间。

1. 您至少有一个与数据库服务器发现相关的主机。
2. 为该主机创建一个名为 发现数据库和表空间的LLD 规则。
3. 切换到该规则的 *LLD* 宏选项卡，添加宏 {#DB}=$.database。
4. 为该规则添加一个名为 *{#DB}* 的活动连接数的监控项原型（类型：Agent，键值：db.connections[{#DB}]）。
5. 发现与每个数据库相关的监控项：

```
Active connections to db1, Active connections to db2, Active connections to db3.
```

6. 为该规则创建一个名为 发现 *{#DB}* 的表空间的发现原型（类型：Nested，键值：db.tablespace.discovery[{#DB}]）。
7. 切换到该发现原型的 预处理选项卡，并添加步骤 JSONPath=$.tablespaces。
8. 切换到该发现原型的 *LLD* 宏选项卡，添加宏 {#TSNAME}=$.name。
9. 为该发现原型创建一个名为 *{#DB}* 的表空间 *{#TSNAME}* 的大小的监控项原型（类型：Agent，键值：db.ts.size[{#DB},{#TSNAME}]）。
10. 发现与每个数据库的每个表空间相关的监控项：

```
Size of tablespace ts1 for db1, Size of tablespace ts2 for db1, Size of tablespace ts3 for db1,
Size of tablespace ts1 for db2, Size of tablespace ts2 for db2, Size of tablespace ts3 for db2,
Size of tablespace ts1 for db3, Size of tablespace ts2 for db3, Size of tablespace ts3 for db3.
```

其键值为 db.ts.size[db1,ts1]、db.ts.size[db1,ts2]、... db.ts.size[db3,ts3]。

情况 2

通过将数据库服务器上的数据库实例表示为已发现的主机来发现这些实例，然后发现每个实例的表空间。

1. 您至少有一个与数据库服务器发现相关的主机（根主机）。
2. 创建一个模板，用于发现每个数据库的表空间。
3. 在此模板中创建一个名为 到 *{#DB}* 的活动连接数的监控项（类型：agent，键值：db.connections[{#DB}]）。
4. 为此模板创建一个名为 发现表空间的LLD 规则（类型：嵌套）。
5. 切换到此规则的 预处理选项卡，并添加步骤 JSONPath=$.tablespaces。
6. 切换到此规则的 *LLD* 宏选项卡，添加宏 {#TSNAME}=$.name。
7. 为此规则创建一个名为 *{#DB}* 的表空间 *{#TSNAME}* 的大小的监控项原型（类型：agent，键值：db.ts.size[{#DB},{#TSNAME}]）。
8. 回到根主机，为此主机创建一个名为 发现数据库和表空间的LLD 规则。
9. 切换到此规则的 *LLD* 宏选项卡，添加宏 {#DB}=$.database。
10. 为此规则添加一个名为 数据库 *{#DB}* 的主机的主机原型。
11. 切换到此主机原型的 宏选项卡，添加宏 {$DB}={#DB}（用于步骤 3 中监控项的名称和键值）。
12. 将步骤 2 中的模板链接到此主机原型。
13. 已发现的主机包含与每个数据库及其表空间相关的已发现监控项：

主机	监控项
数据库 *db1* 的主机	到 db1 的活动连接数
	db1 的表空间 ts1 的大小
	db1 的表空间 ts2 的大小
	db1 的表空间 ts3 的大小
数据库 *db2* 的主机	到 db2 的活动连接数
	db2 的表空间 ts1 的大小
	db2 的表空间 ts2 的大小
	db2 的表空间 ts3 的大小
数据库 *db3* 的主机	到 db3 的活动连接数
	db3 的表空间 ts1 的大小
	db3 的表空间 ts2 的大小
	db3 的表空间 ts3 的大小

低级别发现说明

在用户宏上下文中使用 LLD 宏

LLD 宏可在用户宏上下文中使用，例如，在触发器原型中。

同一监控项的多个 LLD 规则

可以为同一个发现监控项定义多个低级别发现规则。

为此，需要定义 Alias agent 参数，以便在不同的发现规则中使用经过修改的发现监控项键，例如 `vfs.fs.discovery[foo]`、`vfs.fs.discovery[bar]` 等。

返回值的数据限制

如果低级发现规则 JSON 数据由 Zabbix 服务器直接接收，则没有限制。这是因为返回值在处理时无需存储到数据库中。

自定义低级发现规则也没有限制。但是，如果使用用户参数获取自定义低级发现规则数据，则适用用户参数的返回值限制。

如果数据必须经过 Zabbix proxy，则它必须将这些数据存储到数据库中。在这种情况下，适用数据库限制。

发现规则示例

本节提供了为不同类型的实体配置发现规则的实际示例，例如文件系统、网络接口、服务、传感器、SNMP 对象、Java 应用程序和数据库。

每个示例都说明了发现规则可以如何构建，包括所使用的监控项键或查询、带有 LLD 宏的 JSON 输出，以及如何利用这些结果创建监控项、触发器和图形原型。

- 块设备的发现
- CPU 和 CPU 核心的发现
- Zabbix 中主机接口的发现
- IPMI 传感器的发现
- JMX 对象的发现
- 已挂载文件系统的发现
- 网络接口的发现
- SNMP OID 的发现
- SNMP OID 的发现（旧版）
- systemd 服务的发现
- Windows 性能计数器实例的发现
- Windows 服务的发现
- 使用 WMI 查询进行发现
- 使用 ODBC SQL 查询进行发现
- 使用 Prometheus 数据进行发现

自定义 **LLD** 规则

概述

也可以创建完全自定义的 LLD 规则，用于发现任何类型的实体——例如，数据库服务器上的数据库。

为此，应创建一个自定义监控项，使其返回一个 JSON 字符串，用于指定已发现的对象，以及可选的它们的一些属性。
每个实体的宏数量没有限制——内置发现规则通常返回一个或两个宏（例如，文件系统发现会返回两个宏），而自定义规则可以返回更多。

示例

所需的 JSON 字符串格式最好通过示例来说明。假设您正在运行一个旧版 Zabbix 1.8 agent（即不支持vfs.fs.discovery 键的版本），但您仍然需要发现文件系统。下面是一个适用于 Linux 的简单 Perl 脚本，用于发现已挂载的文件系统并以 JSON 格式输出，其中包含文件系统名称和类型。其一种使用方式是将其作为一个 UserParameter，键为 vfs.fs.discovery_perl：

```
####!/usr/bin/perl

$first = 1;

print "[\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"{#FSNAME}\":\"$fsname\",\n";
    print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

Attention:
LLD 宏名称允许使用的符号为 **0-9**、**A-Z**、_、.。名称中不支持小写字母。

其输出示例如下所示（为便于阅读，已重新格式化）。自定义发现检查的 JSON 必须遵循相同的格式。

```
[
    { "{#FSNAME}":"/",                          "{#FSTYPE}":"rootfs"   },
    { "{#FSNAME}":"/sys",                       "{#FSTYPE}":"sysfs"    },
    { "{#FSNAME}":"/proc",                      "{#FSTYPE}":"proc"     },
    { "{#FSNAME}":"/dev",                       "{#FSTYPE}":"devtmpfs" },
    { "{#FSNAME}":"/dev/pts",                   "{#FSTYPE}":"devpts"   },
    { "{#FSNAME}":"/lib/init/rw",               "{#FSTYPE}":"tmpfs"    },
    { "{#FSNAME}":"/dev/shm",                   "{#FSTYPE}":"tmpfs"    },
    { "{#FSNAME}":"/home",                      "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/tmp",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/usr",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/var",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/sys/fs/fuse/connections",   "{#FSTYPE}":"fusectl"  }
]
```

在前一个示例中，要求键名与原型中使用的 LLD 宏名称匹配；另一种方式是使用 JSONPath 提取 LLD 宏值，即 {#FSNAME} → $.fsname 和 {#FSTYPE} → $.fstype，从而可以使用如下脚本：

```
####!/usr/bin/perl

$first = 1;

print "[\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"fsname\":\"$fsname\",\n";
```

```
    print "\t\t\"fstype\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

其输出示例如下所示（为便于阅读，已重新格式化）。

```
[
    { "fsname":"/",                          "fstype":"rootfs"   },
    { "fsname":"/sys",                       "fstype":"sysfs"    },
    { "fsname":"/proc",                      "fstype":"proc"     },
    { "fsname":"/dev",                       "fstype":"devtmpfs" },
    { "fsname":"/dev/pts",                   "fstype":"devpts"   },
    { "fsname":"/lib/init/rw",               "fstype":"tmpfs"    },
    { "fsname":"/dev/shm",                   "fstype":"tmpfs"    },
    { "fsname":"/home",                      "fstype":"ext3"     },
    { "fsname":"/tmp",                       "fstype":"ext3"     },
    { "fsname":"/usr",                       "fstype":"ext3"     },
    { "fsname":"/var",                       "fstype":"ext3"     },
    { "fsname":"/sys/fs/fuse/connections",   "fstype":"fusectl"  }
]
```

然后，在发现规则的 *Filter* 字段中，您可以将 {#FSTYPE} 指定为宏，并将 rootfs|ext3 指定为正则表达式。

Note:
使用自定义 LLD 规则时，您不必使用像 FSNAME/FSTYPE 这样的宏名称，您可以自由使用任何您喜欢的名称。如果使用 JSONPath，则 LLD 行将是一个数组元素，它可以是一个对象，也可以是另一个数组或一个值。

请注意，如果使用用户参数，则返回值限制为 16 MB。更多详细信息，请参见LLD 返回值的数据限制。

虚拟机监控

概述　Zabbix 可以使用低级别发现规则自动发现 VMware 虚拟机监控程序和虚拟机，并基于预定义的主机原型创建用于监控它们的主机。

Zabbix 还包含用于监控 VMware vCenter 或 ESXi 虚拟机监控程序的开箱即用模板。

所需的最低 VMware vCenter 或 vSphere 版本为 5.1。

数据采集　虚拟机的监控包含两个步骤：

1. Zabbix *vmware collector*（vmware 采集器）进程完成对虚拟机的数据采集 - 这些进程通过 SOAP 协议收集包括必要的 VMware web 服务信息，对这些信息进行预处理并且存储于 Zabbix Server 的共享内存。
2. Zabbix *poller*（轮询）进程通过VMware monitoring item keys完成对数据的检索。

Zabbix 将采集的数据分为两种，分别是 VMware 配置数据和 VMware 性能计算数据。这两种类型的数据都由 *vmware collector* 独自完成采集。

下列所统计的参数都是可查看的 VMware 性能计算信息：

- Datastore（数据存储）
- Disk device（磁盘设备）
- CPU
- Power（电源）
- Network interface（网络接口）
- Custom performance counter items（自定义性能计算参数）

用户若对完整的 VMware 性能计算参数有查阅需求，请查看*VMware monitoring item keys*。

请注意，Vmware 事项的检索频率依赖于vmware.eventlog的轮询间隔，且该间隔不能小于 5 秒钟。

配置　如果 Zabbix 服务器是从sources 编译的，则必须使用 `--with-libcurl --with-libxml2` 配置选项进行编译，才能启用虚拟机监控。Zabbix 软件包在编译时已默认启用这些选项。

可通过修改以下 Zabbix 服务器配置文件参数来进行虚拟机监控：

- StartVMwareCollectors

Note:
建议启用的收集器数量多于受监控的 VMware 服务数量；否则，获取 VMware 性能计数器统计信息可能会因获取 VMware 配置数据而延迟（对于大型部署，这一过程需要一些时间）。

 通常，StartVMwareCollectors 的值不应低于 2，也不应超过受监控 VMware 服务数量的两倍：服务数量 < StartVMwareCollectors <（服务数量 * 2）。例如，监控一个 VMware 服务时，将 StartVMwareCollectors 设置为 2；监控三个服务时，将 StartVMwareCollectors 设置为 5。

 请注意，所需的收集器数量还取决于 VMware 环境的范围，以及 VMwareFrequency 和 VMwarePerfFrequency 配置参数。

- VMwareCacheSize
- VMwareFrequency
- VMwarePerfFrequency
- VMwareTimeout

Attention:
要支持数据存储容量指标，请确保 VMware 的 vpxd.stats.maxQueryMetrics 键的值至少设置为 64。有关更多信息，请参阅 *VMware Knowledge Base* article。

发现

Zabbix 可以使用低级别发现规则（例如，vmware.hv.discovery[{$VMWARE.URL}]）自动发现 VMware 虚拟机监控程序和虚拟机。此外，Zabbix 可以使用主机原型为已发现的实体自动生成实际主机。有关更多信息，请参见主机原型。

配置示例

有关如何使用 *VMware FQDN* 模板设置 Zabbix 以监控 VMware 的基本示例，请参见使用 *Zabbix* 监控 *VMware*。

有关如何创建主机、低级别发现规则以及用于监控 VMware 的主机原型的更详细示例，请参见设置示例。

扩展日志记录　可以使用调试级别 5 记录由 *vmware collector* 进程收集的数据，以便进行详细调试。可以在服务器和proxy配置文件中配置调试级别，也可以使用运行时控制选项 `-R log_level_increase="vmware collector,N"`，其中 N 为进程编号。

例如，要将所有 *vmware collector* 进程的调试级别从 4 提高到 5，请运行以下命令：

```
zabbix_server -R log_level_increase="vmware collector"
```

要将第二个 *vmware collector* 进程的调试级别从 4 提高到 5，请运行以下命令：

```
zabbix_server -R log_level_increase="vmware collector,2"
```

不再需要记录 VMware collector 数据的扩展日志时，建议运行 `-R log_level_decrease` 命令，将调试级别降低到默认值（3）。

故障排除

- 如果指标不可用，请确保它们没有在最近的 VMware vSphere 版本中被设为不可用或默认关闭，或者性能指标数据库查询没有设置某些限制。有关更多信息，请参见 ZBX-12094。
- 如果 config.vpxd.stats.maxQueryMetrics 无效，或出现超过允许的最大字符数的错误，请在 vCenter Server 设置中添加 config.vpxd.stats.maxQueryMetrics 参数。该参数的值应与 VMware 的 web.xml 文件中的 maxQuerysize 值相同。有关更多信息，请参见 *VMware Knowledge Base* 文章。
- 如果您怀疑 Zabbix 安装占用了过多内存，请参见使用 tcmalloc 分析过高的内存使用。

VMware 监控项键

概述　本手册页提供了有关可应用于监控VMware environments的简单检查内容。这些指标已通过监控的目标类型进行了分组。

支持的监控项键值　监控项键值以不带参数和附加信息的形式列出。单击监控项键值可查看完整详情。

监控项键值	描述	监控项组
vmware.eventlog	VMware 事件日志。	General
		service
vmware.fullname	VMware 服务全名。	
vmware.version	VMware 服务版本。	
vmware.cl.perfcounter	VMware 集群性能计数器指标。	Cluster
vmware.cluster.alarms.get	VMware 集群告警数据。	
vmware.cluster.discovery	VMware 集群发现。	
vmware.cluster.property	VMware 集群属性。	
vmware.cluster.status	VMware 集群状态。	
vmware.cluster.tags.get	VMware 集群标签数组。	
vmware.datastore.alarms.get	VMware 数据存储告警数据。	Datastore
vmware.datastore.discovery	VMware 数据存储发现。	
vmware.datastore.hv.list	数据存储虚拟机监控程序列表。	
vmware.datastore.perfcounter	VMware 数据存储性能计数器值。	
vmware.datastore.property	VMware 数据存储属性。	
vmware.datastore.read	从数据存储执行读取操作所需的时间。	
vmware.datastore.size	VMware 数据存储空间，单位为字节或占总量的百分比。	
vmware.datastore.tags.get	VMware 数据存储标签数组。	
vmware.datastore.write	向数据存储执行写入操作所需的时间。	
vmware.dc.alarms.get	VMware 数据中心告警数据。	Datacenter
vmware.dc.discovery	VMware 数据中心发现。	
vmware.dc.tags.get	VMware 数据中心标签数组。	
vmware.dvswitch.discovery	VMware vSphere 分布式交换机发现。	vSphere
		Distributed
		Switch
vmware.dvswitch.fetchportsbyid	VMware vSphere 分布式交换机端口数据。	
vmware.hv.alarms.get	VMware 虚拟机监控程序告警数据。	Hypervisor
vmware.hv.cluster.name	VMware 虚拟机监控程序集群名称。	
vmware.hv.connectionstate	VMware 虚拟机监控程序连接状态。	
vmware.hv.cpu.usage	VMware 虚拟机监控程序处理器使用率（Hz）。	
vmware.hv.cpu.usage.perf	间隔内 VMware 虚拟机监控程序处理器使用率百分比。	
vmware.hv.cpu.utilization	间隔内 VMware 虚拟机监控程序处理器使用率百分比，取决于电源管理或 HT。	
vmware.hv.datacenter.name	VMware 虚拟机监控程序数据中心名称。	
vmware.hv.datastore.discovery	VMware 虚拟机监控程序数据存储发现。	
vmware.hv.datastore.list	VMware 虚拟机监控程序数据存储列表。	
vmware.hv.datastore.multipath	可用的数据存储路径数量。	
vmware.hv.datastore.read	从数据存储执行读取操作的平均时间。	
vmware.hv.datastore.size	VMware 数据存储空间，单位为字节或占总量的百分比。	
vmware.hv.datastore.write	向数据存储执行写入操作的平均时间。	
vmware.hv.discovery	VMware 虚拟机监控程序发现。	
vmware.hv.diskinfo.get	VMware 虚拟机监控程序磁盘数据。	
vmware.hv.fullname	完整产品名称，包括版本信息。	
vmware.hv.hw.cpu.freq	VMware 虚拟机监控程序处理器频率。	
vmware.hv.hw.cpu.model	VMware 虚拟机监控程序处理器型号。	
vmware.hv.hw.cpu.num	VMware 虚拟机监控程序上的处理器核心数。	
vmware.hv.hw.cpu.threads	VMware 虚拟机监控程序上的处理器线程数。	
vmware.hv.hw.memory	VMware 虚拟机监控程序总内存大小。	
vmware.hv.hw.model	VMware 虚拟机监控程序型号。	
vmware.hv.hw.sensors.get	VMware 虚拟机监控程序硬件传感器值。	
vmware.hv.hw.serialnumber	VMware 虚拟机监控程序序列号。	
vmware.hv.hw.uuid	VMware 虚拟机监控程序 BIOS UUID。	
vmware.hv.hw.vendor	VMware 虚拟机监控程序厂商名称。	
vmware.hv.maintenance	VMware 虚拟机监控程序维护状态。	
vmware.hv.memory.size.ballooned	VMware 虚拟机监控程序气球内存大小。	
vmware.hv.memory.used	VMware 虚拟机监控程序已用内存大小。	
vmware.hv.net.if.discovery	VMware 虚拟机监控程序网络接口发现。	
vmware.hv.network.in	VMware 虚拟机监控程序网络输入统计。	
vmware.hv.network.linkspeed	VMware 虚拟机监控程序网络接口速度。	
vmware.hv.network.out	VMware 虚拟机监控程序网络输出统计。	
vmware.hv.perfcounter	VMware 虚拟机监控程序性能计数器值。	
vmware.hv.property	VMware 虚拟机监控程序属性。	

监控项键值	描述	监控项组
vmware.hv.power	VMware 虚拟机监控程序功耗。	
vmware.hv.sensor.health.state	VMware 虚拟机监控程序健康状态汇总传感器。	
vmware.hv.sensors.get	VMware 虚拟机监控程序硬件厂商状态传感器。	
vmware.hv.status	VMware 虚拟机监控程序状态。	
vmware.hv.tags.get	VMware 虚拟机监控程序标签数组。	
vmware.hv.uptime	VMware 虚拟机监控程序运行时间。	
vmware.hv.version	VMware 虚拟机监控程序版本。	
vmware.hv.vm.num	VMware 虚拟机监控程序上的虚拟机数量。	
vmware.rp.cpu.usage	VMware Resource Pool 中间隔内的 CPU 使用率（Hz）。	Resource pool
vmware.rp.memory	VMware 资源池的内存指标。	
vmware.alarms.get	VMware 虚拟中心告警数据。	Virtual center
vmware.vm.alarms.get	VMware 虚拟机告警数据。	Virtual machine
vmware.vm.attribute	VMware 虚拟机自定义属性值。	
vmware.vm.cluster.name	VMware 虚拟机名称。	
vmware.vm.consolidationneeded	VMware 虚拟机磁盘需要整合。	
vmware.vm.cpu.latency	由于争用物理 CPU 访问权，虚拟机无法运行的时间百分比。	
vmware.vm.cpu.num	VMware 虚拟机上的处理器数量。	
vmware.vm.cpu.readiness	虚拟机已就绪但无法被调度到物理 CPU 上运行的时间百分比。	
vmware.vm.cpu.ready	虚拟机已就绪但无法被调度到物理 CPU 上运行的时间。	
vmware.vm.cpu.swapwait	等待换入所花费的 CPU 时间百分比。	
vmware.vm.cpu.usage	VMware 虚拟机处理器使用率（Hz）。	
vmware.vm.cpu.usage.perf	间隔内 VMware 虚拟机处理器使用率百分比。	
vmware.vm.datacenter.name	VMware 虚拟机数据中心名称。	
vmware.vm.discovery	VMware 虚拟机发现。	
vmware.vm.guest.memory.swapped	已交换到交换空间的客户机物理内存量。	
vmware.vm.guest.osuptime	自上次操作系统启动以来经过的总时间。	
vmware.vm.hv.maintenance	VMware 虚拟机虚拟机监控程序维护状态。	
vmware.vm.hv.name	VMware 虚拟机虚拟机监控程序名称。	
vmware.vm.memory.size	VMware 虚拟机总内存大小。	
vmware.vm.memory.size.ballooned	VMware 虚拟机气球内存大小。	
vmware.vm.memory.size.compressed	VMware 虚拟机压缩内存大小。	
vmware.vm.memory.size.consumed	用于备份客户机物理内存页而消耗的主机物理内存量。	
vmware.vm.memory.size.private	VMware 虚拟机私有内存大小。	
vmware.vm.memory.size.shared	VMware 虚拟机共享内存大小。	
vmware.vm.memory.size.swapped	VMware 虚拟机交换内存大小。	
vmware.vm.memory.size.usage.guest	VMware 虚拟机客户机内存使用率。	
vmware.vm.memory.size.usage.host	VMware 虚拟机主机内存使用率。	
vmware.vm.memory.usage	已消耗的主机物理内存百分比。	
vmware.vm.net.if.discovery	VMware 虚拟机网络接口发现。	
vmware.vm.net.if.in	VMware 虚拟机网络接口输入统计。	
vmware.vm.net.if.out	VMware 虚拟机网络接口输出统计。	
vmware.vm.net.if.usage	间隔内 VMware 虚拟机网络利用率。	
vmware.vm.perfcounter	VMware 虚拟机性能计数器值。	
vmware.vm.powerstate	VMware 虚拟机电源状态。	
vmware.vm.property	VMware 虚拟机属性。	
vmware.vm.snapshot.get	VMware 虚拟机快照状态。	
vmware.vm.state	VMware 虚拟机状态。	
vmware.vm.storage.committed	VMware 虚拟机已提交存储空间。	
vmware.vm.storage.readoio	采集间隔内虚拟磁盘上未完成读取请求的平均数量。	
vmware.vm.storage.totalreadlatency	从虚拟磁盘读取所需的平均时间。	
vmware.vm.storage.totalwritelatency	向虚拟磁盘写入所需的平均时间。	
vmware.vm.storage.uncommitted	VMware 虚拟机未提交存储空间。	
vmware.vm.storage.unshared	VMware 虚拟机未共享存储空间。	
vmware.vm.storage.writeoio	采集间隔内虚拟磁盘上未完成写入请求的平均数量。	
vmware.vm.tags.get	VMware 虚拟机标签数组。	
vmware.vm.tools	VMware 虚拟机客户机工具状态或版本。	
vmware.vm.uptime	VMware 虚拟机运行时间。	
vmware.vm.vfs.dev.discovery	VMware 虚拟机磁盘设备发现。	

监控项键值	描述	监控项组
vmware.vm.vfs.dev.read	VMware 虚拟机磁盘设备读取统计。	
vmware.vm.vfs.dev.write	VMware 虚拟机磁盘设备写入统计。	
vmware.vm.vfs.fs.discovery	VMware 虚拟机文件系统发现。	
vmware.vm.vfs.fs.size	VMware 虚拟机文件系统统计。	

监控键详细信息　没有尖括号的参数是必要参数。标有尖括号 **< >** 的参数是可选参数。

vmware.eventlog[url,<mode>,<severity>]

 VMware 事件日志。
 返回值：*Log*。

参数：

- **url** - VMware 服务 URL；
- **mode** - *all*（默认）或 *skip* - 跳过对较旧数据的处理；
- **severity** - 按严重性过滤：*error*、*warning*、*info* 或 *user*。如果以逗号分隔的列表指定多个严重性，则必须将此参数用引号括起来（例如："`error,warning,info,user`"）。默认禁用。

注释：

- 每个 URL 只能有一个 `vmware.eventlog` 监控项键；
- 另请参见 VMware 事件日志记录的过滤示例。

vmware.fullnameurl

 VMware 服务全程。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL。

vmware.versionurl

 VMware 服务版本。
 返回值：*String*。

参数详情：

- **url** - VMware 服务的 URL。

vmware.cl.perfcounter[url,id,path,<instance>]

 VMware 集群性能计数器指标。
 返回值：*Integer*。

参数：

- **url** - VMware 服务 URL；
- **id** - VMware 集群 ID。`id` 可从 `vmware.cluster.discovery[]` 的 {#CLUSTER.ID} 获取。
- **path** - 性能计数器路径[1]；
- **instance** - 性能计数器实例。

vmware.cluster.alarms.get[url,id]

 VMware 集群告警数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **id** - VMware 集群 ID。

vmware.cluster.discoveryurl

 VMware 集群发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL。

vmware.cluster.property[url,id,prop]

 VMware 集群属性。
 返回值：*String*。

参数：

- **url** - VMware 服务 URL；
- **id** - VMware 集群 ID；

- **prop** - 属性路径，即 VMware SDK 中定义的 VM 对象属性名称。

示例：

```
vmware.vm.property[{$VMWARE.URL},{$VMWARE.VM.UUID},overallStatus]
vmware.vm.property[{$VMWARE.URL},{$VMWARE.VM.UUID},runtime.powerState]
```

vmware.cluster.status[url,name]

 VMware 集群状态。
 返回值：*0* - gray; *1* - green; *2* - yellow; *3* - red.

参数详情：

- **url** - VMware 服务的 URL；
- **name** - VMware 集群名称。

vmware.cluster.tags.get[url,id]

 VMware 集群标签数组。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **id** - VMware 集群 ID。

此监控项适用于 vSphere 6.5 及更新版本。

vmware.datastore.alarms.get[url,uuid]

 VMware 数据存储告警数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 数据存储的全局唯一标识符。

vmware.datastore.discovery[url,<filter_uuid>]

 发现 VMware 数据存储。
 返回值：JSON 对象。

参数：

- **url** - VMware 服务 URL；
- **filter_uuid** - 单个 HV 或 VM 唯一标识符（默认值：空）。如果指定，则仅发现已链接的数据存储。

vmware.datastore.hv.list[url,datastore]

 数据存储管理程序列表。
 返回值：*String*。

参数：

- **url** - VMware 服务 URL；
- **datastore** - 数据存储的 UUID 或名称。

输出示例：

```
esx7-01-host.zabbix.sandbox
esx7-02-host.zabbix.sandbox
```

vmware.datastore.perfcounter[url,uuid,path,<instance>]

 VMware 数据存储性能计数器值。
 返回值：*Integer* [2]。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 数据存储全局唯一标识符；
- **path** - 性能计数器路径[1]；
- **instance** - 性能计数器实例。对聚合值使用空实例（默认）。`instance` 可从 `vmware.datastore.discovery[]` 的 {#DATASTORE.EXTENT} 数组部分获取。

vmware.datastore.property[url,uuid,prop]

 VMware 数据存储属性。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 数据存储全局唯一标识符；

- **prop** - 属性路径。

vmware.datastore.read[url,datastore,<mode>]

 从数据存储执行读取操作所花费的时间（毫秒）。
 返回值：*Integer* [2]。

参数：

- **url** - VMware 服务 URL；
- **datastore** - 数据存储 UUID 或名称；
- **mode** - *latency*（平均值，默认）或 *maxlatency*（最大值）。

vmware.datastore.size[url,datastore,<mode>]

 VMware 数据存储空间，以字节或占总量的百分比表示。
 返回值：*Integer* - 用于字节；*Float* - 用于百分比。

参数：

- **url** - VMware 服务 URL；
- **datastore** - 数据存储 UUID 或名称；
- **mode** - 可选值：*total*（默认）、*free*、*pfree*（空闲百分比）、*uncommitted*。

vmware.datastore.tags.get[url,uuid]

 VMware 数据存储标签数组。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 数据存储全局唯一标识符。

此监控项适用于 vSphere 6.5 及更新版本。

vmware.datastore.write[url,datastore,<mode>]

 向数据存储执行写入操作所花费的时间（毫秒）。
 返回值：*Integer* [2]。

参数：

- **url** - VMware 服务 URL；
- **datastore** - 数据存储 UUID 或名称；
- **mode** - *latency*（平均值，默认）或 *maxlatency*（最大值）。

vmware.dc.alarms.get[url,id]

 VMware 数据中心告警数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **id** - VMware 数据中心 ID。

vmware.dc.discoveryurl

 VMware 数据中心发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL。

vmware.dc.tags.get[url,id]

 VMware 数据中心标签数组。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **id** - VMware 数据中心 ID。

此监控项适用于 vSphere 6.5 及更新版本。

vmware.dvswitch.discoveryurl

 VMware vSphere 分布式交换机发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL。

vmware.dvswitch.fetchports.get[url,uuid,<filter>,<mode>]

 VMware vSphere 分布式交换机端口数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware DVSwitch 全局唯一标识符。
- **filter** - 一个单独的字符串，使用逗号分隔的条件来选择端口；
- **mode** - *state*（默认，所有不包含"config" XML 节点的 XML）或 *full*。

filter 参数支持 VMware 数据对象 DistributedVirtualSwitchPortCriteria 中可用的 criteria。

示例：

```
vmware.dvswitch.fetchports.get[{$VMWARE.URL},{$VMWARE.DVS.UUID},"connected:true,active:true,uplinkPort:fal
```

vmware.hv.alarms.get[url,uuid]

 VMware 虚拟机监控程序告警数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序的全局唯一标识符。

vmware.hv.cluster.name[url,uuid]

 VMware hypervisor 集群名称。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.connectionstate[url,uuid]

 VMware hypervisor 连接状态。
 返回值：*String*: *connected*, *disconnected*, or *notResponding*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.cpu.usage[url,uuid]

 VMware hypervisor 处理器使用率（Hz）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.cpu.usage.perf[url,uuid]

 VMware hypervisor 间隔期间处理器使用率的百分比。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.cpu.utilization[url,uuid]

 VMware hypervisor 处理器使用率在间隔期间的百分比取决于电源管理或 HT。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.datacenter.name[url,uuid]

 VMware hypervisor 数据中心名称。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.datastore.discovery[url,uuid]

 VMware hypervisor 数据存储发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.datastore.list[url,uuid]

 VMware hypervisor 数据存储列表。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

输出举例：

```
SSD-RAID1-VAULT1
SSD-RAID1-VAULT2
SSD-RAID10
```

vmware.hv.datastore.multipath[url,uuid,<datastore>,<partitionid>]

 可用数据存储路径的数量。
 返回值：整数。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序全局唯一标识符；
- **datastore** - 数据存储 UUID 或名称；
- **partitionid** - 来自 `vmware.hv.datastore.discovery` 的物理设备内部 ID。

vmware.hv.datastore.read[url,uuid,datastore,<mode>]

 从数据存储读取操作的平均耗时（毫秒）。
 返回值：*Integer* [2]。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序的全局唯一标识符；
- **datastore** - 数据存储 UUID 或名称；
- **mode** - *latency*（默认）。

vmware.hv.datastore.size[url,uuid,datastore,<mode>]

 VMware 数据存储空间大小，以字节或占总量的百分比表示。
 返回值：*Integer* - 字节；*Float* - 百分比。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序的全局唯一标识符；
- **datastore** - 数据存储的 UUID 或名称；
- **mode** - 可选值：*total*（默认）、*free*、*pfree*（空闲百分比）、*uncommitted*。

vmware.hv.datastore.write[url,uuid,datastore,<mode>]

 写入 datastore 操作的平均耗时（毫秒）。
 返回值：*Integer* [2]。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **datastore** - datastore UUID 或名称；
- **mode** - *latency*（默认）。

vmware.hv.discoveryurl

 VMware hypervisors 发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL。

vmware.hv.diskinfo.get[url,uuid]

 VMware hypervisor 磁盘数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.fullname[url,uuid]

 完整的产品名称，包括版本信息。
 返回值：*String*。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序的全局唯一标识符。

vmware.hv.hw.cpu.freq[url,uuid]

 VMware hypervisor 处理器频率（Hz）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.hw.cpu.model[url,uuid]

 VMware hypervisor 处理器型号。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.hw.cpu.num[url,uuid]

 VMware hypervisor 核心处理器数量。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.hw.cpu.threads[url,uuid]

 VMware hypervisor 处理器线程数量。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.hw.memory[url,uuid]

 VMware hypervisor 总内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.hw.model[url,uuid]

 VMware hypervisor 型号。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.hw.sensors.get[url,uuid]

 VMware 虚拟机监控程序硬件传感器值。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序全局唯一标识符。

vmware.hv.hw.serialnumber[url,uuid]

 VMware hypervisor 序列号。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

此项目适用于 vSphere 6.5 及更新版本。

vmware.hv.hw.uuid[url,uuid]

 VMware hypervisor BIOS UUID。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

此项目适用于 vSphere 6.5 及更新版本。

vmware.hv.hw.vendor[url,uuid]

 VMware hypervisor 制造商名称。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

此项目适用于 vSphere 6.5 及更新版本。

vmware.hv.maintenance[url,uuid]

 VMware hypervisor 维护状态。
 返回值：*0* - 不在维护中；*1* - 正在维护。

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.memory.size.ballooned[url,uuid]

 VMware hypervisor 内存大小激增（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.memory.used[url,uuid]

 VMware hypervisor 已占用内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.net.if.discovery[url,uuid]

 VMware hypervisor 网络接口发现。
 返回值：*JSON object*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.network.in[url,uuid,<mode>]

 VMware hypervisor 网络输入统计信息（字节/秒）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **mode** - *bps* （缺省默认），*packets*，*dropped*，*errors*，*broadcast*。

vmware.hv.network.linkspeed[url,uuid,ifname]

 VMware hypervisor 网络接口速率。
 返回值：*Integer*。若网络接口关闭，则返回 *0*，若网络接口正常则显示接口速率。

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **ifname** - 接口名称。

vmware.hv.network.out[url,uuid,<mode>]

 VMware hypervisor 网络输出统计信息（字节/秒）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **ifname** - 接口名称。

vmware.hv.perfcounter[url,uuid,path,<instance>]

 VMware hypervisor 性能计数器值。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **path** - 性能计数器路径 [1]；
- **instance** - 性能计数器实例。对聚合值使用空实例（默认值）。

vmware.hv.property[url,uuid,prop]

 VMware hypervisor 属性。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **prop** - 属性路径。

vmware.hv.power[url,uuid,<max>]

 VMware hypervisor 功率使用（W）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符；
- **max** - 允许使用的最大功率。

vmware.hv.sensor.health.state[url,uuid]

 VMware hypervisor 运行状况汇总传感器。
 返回值：*Integer*: *0* - 灰色；*1* - 绿色；*2* - 黄色；*3* - 红色。

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

请注意，该项目可能在 VMware vSphere 6.5 及更新版本中不起作用，因为 VMware 已弃用 *VMware Rollup Health State* 传感器。

vmware.hv.sensors.get[url,uuid]

 VMware 虚拟机监控程序硬件供应商状态传感器。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序全局唯一标识符。

vmware.hv.status[url,uuid]

 VMware hypervisor 状态。
 参数详情：*Integer*: *0* - 灰色；*1* - 绿色；*2* - 黄色；*3* - 红色。

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

该监控项使用主机系统总体状态属性。

vmware.hv.tags.get[url,uuid]

 VMware 虚拟机监控程序标签数组。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机监控程序全局唯一标识符。

此监控项适用于 vSphere 6.5 及更新版本。

vmware.hv.uptime[url,uuid]

 VMware hypervisor 启动时长（秒）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

该监控项使用主机系统总体状态属性。

vmware.hv.version[url,uuid]

 VMware hypervisor 版本。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.hv.vm.num[url,uuid]

 VMware hypervisor 所拥有的虚拟机数量。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware hypervisor 全局唯一标识符。

vmware.rp.cpu.usage[url,rpid]

 VMware 资源池在间隔期间的 CPU 使用率（赫兹）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **rpid** - VMware 资源池 ID。

vmware.rp.memory[url,rpid,<mode>]

 VMware 资源池的内存指标。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **rpid** - VMware 资源池 ID；
- **mode** - 可能的值：
consumed（缺省默认）- 备份访客物理内存页所消耗的主机物理内存量
 ballooned - 访客中膨胀驱动程序从虚拟机回收的访客物理内存量
overhead - ESXi 数据结构为运行虚拟机而消耗的主机物理内存

vmware.alarms.geturl

 VMware 虚拟中心告警数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL。

vmware.vm.alarms.get[url,uuid]

 VMware 虚拟机告警数据。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.attribute[url,uuid,name]

 VMware 虚拟机自定义属性名称。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **name** - 自定义属性名称。

vmware.vm.cluster.name[url,uuid]

 VMware 虚拟机名称。
 返回值：*String*。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.consolidationneeded[url,uuid]

 VMware 虚拟机硬盘整合需求。
 返回值：*String*: *true* - 具备整合需求；*false* - 不具备整合需求。

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.cpu.latency[url,uuid]

 虚拟机由于争夺对物理 CPU（s）的访问而无法运行的时间百分比。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.cpu.num[url,uuid]

 VMware 虚拟机上的处理器数量。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.cpu.readiness[url,uuid,<instance>]

 虚拟机已准备就绪但无法实现在物理 CPU 上运行的时间百分比。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - CPU 实例。

vmware.vm.cpu.ready[url,uuid]

 虚拟机准备就绪但无法实现在物理 CPU 上运行的时间（以毫秒为单位）。CPU 准备时间取决于主机上虚拟机的数量及其 CPU 负载（%）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.cpu.swapwait[url,uuid,<instance>]

 CPU 等待换入所花费时间的百分比。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - CPU 实例。

vmware.vm.cpu.usage[url,uuid]

 VMware 虚拟机处理器的使用率（Hz）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.cpu.usage.perf[url,uuid]

 间隔期间 VMware 虚拟机处理器使用率的百分比。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.datacenter.name[url,uuid]

 VMware 虚拟机数据中心名称
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.discoveryurl

 VMware 虚拟机发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL。

vmware.vm.guest.memory.size.swapped[url,uuid]

 交换到交换空间的访客物理内存量（KB）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.guest.osuptime[url,uuid]

 自上次操作系统启动以来经过的总时间（以秒为单位）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.hv.maintenance[url,uuid]

 VMware 虚拟机管理程序的维护状态。
 返回值：*0* - 不处于维护状态；*1* - 处于维护状态。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.hv.name[url,uuid]

 VMware 虚拟机管理程序名称。
 返回值：*String*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size[url,uuid]

 VMware 虚拟机的总内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.ballooned[url,uuid]

 VMware 虚拟机的内存激增大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.compressed[url,uuid]

 VMware 虚拟机压缩的内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.consumed[url,uuid]

 备份访客物理内存页所消耗的主机物理内存量（KB）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.private[url,uuid]

 VMware 虚拟机专用内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.shared[url,uuid]

 VMware 虚拟机共享内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.swapped[url,uuid]

 VMware 虚拟机交换内存大小（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.usage.guest[url,uuid]

 VMware 虚拟机访客内存使用情况（字节）
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.size.usage.host[url,uuid]

 VMware 虚拟机主机内存使用情况（字节）
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.memory.usage[url,uuid]

 已消耗的主机物理内存的百分比。
 返回值：*Float*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.net.if.discovery[url,uuid]

 VMware 虚拟机网络接口发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.net.if.in[url,uuid,instance,<mode>]

 VMware 虚拟机网络接口输入统计信息（字节/数据包/秒）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 网络接口实例；
- **mode** - *bps* （缺省默认）或者 *pps* - 字节或数据包每秒。

vmware.vm.net.if.out[url,uuid,instance,<mode>]

 VMware 虚拟机网络接口输出统计信息（字节/数据包/秒）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 网络接口实例；
- **mode** - *bps* （缺省默认）或者 *pps* - 字节或数据包每秒。

vmware.vm.net.if.usage[url,uuid,<instance>]

 间隔期间的 VMware 虚拟机网络利用率（传输速率和接收速率的组合）（KBps）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 网络接口实例。

vmware.vm.perfcounter[url,uuid,path,<instance>]

 VMware 虚拟机性能计数器值。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **path** - 性能计数器路径 [1]；
- **instance** - 性能计数器实例。对聚合值使用空实例（默认值）。

vmware.vm.powerstate[url,uuid]

 VMware 虚拟机的电源状态。
 返回值：*0* - 电源关闭；*1* - 电源开启 *2* - 暂停的。

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.property[url,uuid,prop]

 VMware 虚拟机属性。
 返回值：*String*。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **prop** - 属性路径。

vmware.vm.snapshot.get[url,uuid]

 VMware 虚拟机快照状态。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；

- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.state[url,uuid]

 VMware 虚拟机状态。
 返回值：*String*: *notRunning*, *resetting*, *running*, *shuttingDown*, *standby*, or *unknown*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.storage.committed[url,uuid]

 VMware 虚拟机承诺的存储空间（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.storage.readoio[url,uuid,instance]

 在收集间隔内对虚拟磁盘的未完成读取请求的平均数。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 磁盘设备实例。

vmware.vm.storage.totalreadlatency[url,uuid,instance]

 从虚拟磁盘读取的平均时间（毫秒）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 磁盘设备实例。

vmware.vm.storage.totalwritelatency[url,uuid,instance]

 写入虚拟磁盘的平均时间（毫秒）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 磁盘设备实例。

vmware.vm.storage.uncommitted[url,uuid]

 VMware 虚拟机未提交的存储空间（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.storage.unshared[url,uuid]

 VMware 虚拟机的非共享存储空间（字节）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.storage.writeoio[url,uuid,instance]

 在收集间隔内对虚拟磁盘未完成的写入请求的平均数。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 磁盘设备实例。

vmware.vm.tags.get[url,uuid]

 VMware 虚拟机标签数组。
 返回值: JSON object。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

此监控项适用于 vSphere 6.5 及更新版本。

vmware.vm.tools[url,uuid,<mode>]

 VMware 虚拟机客户机工具的状态。
 返回值：*String*。
 在 *status* 模式下：*guestToolsExecutingScripts* - VMware Tools 正在启动；*guestToolsNotRunning* - VMware Tools 未运行；*guestToolsRunning* - VMware Tools 正在运行。
 在 *version* 模式下：版本。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **mode** - *version*、*status*（默认）。

vmware.vm.uptime[url,uuid]

 VMware 虚拟机正常运行时间（秒）。
 返回值：*Integer*.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.vfs.dev.discovery[url,uuid]

 VMware 虚拟机磁盘设备发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符。

vmware.vm.vfs.dev.read[url,uuid,instance,<mode>]

 VMware 虚拟机磁盘设备读取统计信息（字节/操作/秒）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 磁盘设备实例；
- **mode** - *bps* （缺省默认）或者 *ops* - 字节或每秒操作数。

vmware.vm.vfs.dev.write[url,uuid,instance,<mode>]

 VMware 虚拟机磁盘设备写入统计信息（字节/操作/秒）。
 返回值：*Integer* [2].

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **instance** - 磁盘设备实例；
- **mode** - *bps* （缺省默认）或者 *ops* - 字节或每秒操作数。

vmware.vm.vfs.fs.discovery[url,uuid]

 VMware 虚拟机文件系统发现。
 返回值：JSON object.

参数详情：

- **url** - VMware 服务的 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；

必须在访客虚拟机上安装 VMware Tools，此项目才能正常工作。

vmware.vm.vfs.fs.size[url,uuid,fsname,<mode>]

 VMware 虚拟机文件系统统计信息（字节/百分比）。
 返回值：*Integer* - 用于字节；*Float* - 用于百分比。

参数：

- **url** - VMware 服务 URL；
- **uuid** - VMware 虚拟机全局唯一标识符；
- **fsname** - 文件系统名称；
- **mode** - *total*、*free*、*used*、*pfree* 或 *pused*。

要使此监控项正常工作，必须在客户虚拟机上安装 VMware Tools。

脚注

[1] 参见为 VMware 创建自定义性能计数器名称。

[2] 这些监控项的值来自 VMware 性能计数器，并使用 VMwarePerfFrequency 参数 来刷新它们在 Zabbix VMware 缓存中的数据：

- vmware.cl.perfcounter
- vmware.hv.datastore.read
- vmware.hv.datastore.write
- vmware.hv.network.in
- vmware.hv.network.out
- vmware.hv.perfcounter
- vmware.vm.cpu.ready
- vmware.vm.net.if.in
- vmware.vm.net.if.out
- vmware.vm.perfcounter
- vmware.vm.vfs.dev.read
- vmware.vm.vfs.dev.write

更多信息

如何配置 Zabbix 监控 VMware 环境可以参考Virtual machine monitoring 的详细信息。

虚拟机发现键字段

下表列出了虚拟机相关发现键返回的字段。

监控项键		
描述	字段	返回内容
vmware.cluster.discovery		
执行集群发现。	{#CLUSTER.ID}	集群标识符。
	{#CLUSTER.NAME}	集群名称。
	”resource_pool”	包含资源池数据的数组，包括资源组 ID、tags 数组、资源池路径、虚拟机数量。 数组结构： [{ "rpid":"resource group id", "tags":[{}], "rpath":"resource group path", "vm_count":0 }]
	”tags”	有关 "tags" 数组结构，请参见”tags” 字段。 包含标签的数组，标签包含标签名称、描述和类别。 数组结构： [{ "tag":"tag name", "tag_description":"tag description", "category":"tag category" }]
vmware.datastore.discovery		
执行数据存储发现。	{#DATASTORE}	数据存储名称。

监控项键		
	{#DATASTORE.EXTENT}	包含数据存储 extent 分区 ID 和实例名称的数组。 数组结构： [{ "partitionid":1, "instance":"name" }]
	{#DATASTORE.TYPE}	数据存储类型。 值示例：VMFS、NFS、vsan 等。
	{#DATASTORE.UUID}	数据存储标识符。
	"tags"	包含标签的数组，标签包含标签名称、描述和类别。 数组结构： [{ "tag":"tag name", "tag_description":"tag description", "category":"tag category" }]
vmware.dc.discovery		
执行数据中心发现。	{#DATACENTER}	数据中心名称。
	{#DATACENTERID}	数据中心标识符。
	"tags"	包含标签的数组，标签包含标签名称、描述和类别。 数组结构： [{ "tag":"tag name", "tag_description":"tag description", "category":"tag category" }]
vmware.dvswitch.discovery		
执行 vSphere 分布式交换机发现。	{#DVS.NAME}	交换机名称。
	{#DVS.UUID}	交换机标识符。
vmware.hv.discovery		
执行虚拟机监控程序发现。	{#HV.UUID}	虚拟机监控程序唯一标识符。
	{#HV.ID}	虚拟机监控程序标识符（HostSystem managed object 名称）。
	{#HV.NAME}	虚拟机监控程序名称。
	{#HV.NETNAME}	虚拟机监控程序网络主机名。
	{#HV.IP}	虚拟机监控程序 IP 地址，可能为空。 在具有多个网络接口的 HA 配置中，接口选择优先级如下： - 优先选择与 vCenter IP 处于同一 IP 子网的 IP； - 优先选择来自默认网关所在 IP 子网的 IP； - 优先选择来自 ID 最小接口的 IP。
	{#CLUSTER.NAME}	集群名称，可能为空。
	{#DATACENTER.NAME}	数据中心名称。
	{#PARENT.NAME}	存储虚拟机监控程序的容器名称。
	{#PARENT.TYPE}	存储虚拟机监控程序的容器类型。值可以是 Datacenter、Folder、ClusterComputeResource、VMware，其中"VMware" 表示未知的容器类型。

监控项键		
	”resource_pool”	包含资源池数据的数组，包括资源组 ID、tags 数组、资源池路径、虚拟机数量。 数组结构： [{ "rpid":"resource group id", "tags":[{}], "rpath":"resource group path", "vm_count":0 }]
	”tags”	有关 "tags" 数组结构，请参见”tags” 字段。 包含标签的数组，标签包含标签名称、描述和类别。 数组结构： [{ "tag":"tag name", "tag_description":"tag description", "category":"tag category" }]
vmware.hv.datastore.discovery		
执行虚拟机监控程序数据存储发现。请注意，多个虚拟机监控程序可以使用同一个数据存储。	{#DATASTORE}	数据存储名称。
	{#DATASTORE.TYPE}	数据存储类型。 值示例：VMFS、NFS、vsan 等。
	{#DATASTORE.UUID}	数据存储标识符。
	{#MULTIPATH.COUNT}	已注册的数据存储路径数量。
	{#MULTIPATH.PARTITION.COUNT}	可用磁盘分区数量。
	”datastore_extent”	包含数据存储 extent 实例名称和分区 ID 的数组。 数组结构： [{ "partitionid":1, "instance":"name" }]
	”tags”	包含标签的数组，标签包含标签名称、描述和类别。 数组结构： [{ "tag":"tag name", "tag_description":"tag description", "category":"tag category" }]
vmware.hv.net.if.discovery		
执行虚拟机监控程序网络接口发现。	{#IFNAME}	接口名称。
	{#IFDRIVER}	接口驱动程序。
	{#IFDUPLEX}	接口双工设置。
	{#IFSPEED}	接口速度。
	{#IFMAC}	接口 MAC 地址。
vmware.vm.discovery		
执行虚拟机发现。	{#VM.UUID}	虚拟机唯一标识符。
	{#VM.ID}	虚拟机标识符（VirtualMachine managed object 名称）。
	{#VM.NAME}	虚拟机名称。
	{#HV.NAME}	虚拟机监控程序名称。
	{#HV.UUID}	虚拟机监控程序唯一标识符。
	{#HV.ID}	虚拟机监控程序标识符（HostSystem managed object 名称）。
	{#CLUSTER.NAME}	集群名称，可能为空。

	{#DATACENTER.NAME}	数据中心名称。
	{#DATASTORE.NAME}	数据存储名称。
	{#DATASTORE.UUID}	数据存储标识符。
	{#VM.IP}	虚拟机 IP 地址，可能为空。
	{#VM.DNS}	虚拟机 DNS 名称，可能为空。
	{#VM.GUESTFAMILY}	来宾虚拟机操作系统系列，可能为空。
	{#VM.GUESTFULLNAME}	来宾虚拟机操作系统完整名称，可能为空。
	{#VM.FOLDER}	虚拟机父文件夹链，可用作嵌套组的值；文件夹名称以"/" 连接。可能为空。
	{#VM.TOOLS.STATUS}	VMware 虚拟机 tools 状态。
	{#VM.POWERSTATE}	VMware 虚拟机电源状态（`poweredOff`、`poweredOn` 或 `suspended`）。
	{#VM.RPOOL.ID}	资源池标识符。
	{#VM.RPOOL.PATH}	完整资源池路径，不包括根名称"Resources"。文件夹名称以"/" 连接。
	{#VM.SNAPSHOT.COUNT}	VM 快照数量。

	"tags"	包含标签的数组，标签包含标签名称、描述和类别。
		
		数组结构：
		`[{`
		`"tag":"tag name",`
		`"tag_description":"tag description",`
		`"category":"tag category"`
		`}]`

	"vm_customattribute"	虚拟机自定义属性数组（如果已定义）。
		
		数组结构：
		`[{`
		`"name":"custom field name",`
		`"value":"custom field value"`
		`}]`

	"net_if"	虚拟机网络接口数组。
		
		数组结构：
		`[{`
		`"ifname": "interface name",`
		`"ifdesc": "interface description",`
		`"ifmac": "00:00:00:00:00:00",`
		`"ifconnected": true,`
		`"iftype": "interface type",`
		`"ifbackingdevice": "interface backing device",`
		`"ifdvswitch_uuid": "interface switch uuid",`
		`"ifdvswitch_portgroup": "interface switch port group",`
		`"ifdvswitch_port": "interface switch port",`
		`"ifip": ["interface ip addresses"]`
		`}]`
		
		有关返回数据的说明，请参见"vmware.vm.net.if.discovery" 监控项键。

vmware.vm.net.if.discovery		
执行虚拟机网络接口发现。	{#IFNAME}	网络接口名称。
	{#IFDESC}	接口描述。
	{#IFMAC}	接口 MAC 地址。
	{#IFCONNECTED}	接口连接状态（`false` - 已断开；`true` - 已连接）。
	{#IFTYPE}	接口类型。
	{#IFBACKINGDEVICE}	后端设备名称。
	{#IFDVSWITCH.UUID}	vSphere 分布式交换机唯一标识符。
	{#IFDVSWITCH.PORTGROUP}	分布式端口组。
	{#IFDVSWITCH.PORT}	vSphere 分布式交换机端口。
	"ifip"	接口地址数组。
vmware.vm.vfs.dev.discovery		
执行虚拟机磁盘设备发现。	{#DISKNAME}	磁盘设备名称。

监控项键		
vmware.vm.vfs.fs.discovery 执行虚拟机文件系统发现。	{#FSNAME}	文件系统名称。

VMware 监控项的 JSON 示例

概述　本节提供有关各种 VMware 监控项 返回的 JSON 对象的补充信息。

vmware.*.alarms.get　监控项 **vmware.alarms.get[]**、**vmware.cluster.alarms.get[]**、**vmware.datastore.alarms.get[]**、**vmware.dc.alarms.get[]**、**vmware.hv.alarms.get[]**、**vmware.vm.alarms.get[]** 返回具有以下结构的 JSON 对象（值仅为示例）：

```
{
    "alarms": [
        {
            "name": "Host connection and power state",
            "system_name": "alarm.HostConnectionStateAlarm",
            "description": "Default alarm to monitor host connection and power state",
            "enabled": true,
            "key": "alarm-1.host-2013",
            "time": "2022-06-27T05:27:38.759976Z",
            "overall_status": "red",
            "acknowledged": false,
            "entity": {
                "id": "host-2013",
                "uuid": "4201b707-f737-7cf8-881f-49215d1aad2a",
                "type": "HostSystem",
                "name": "esx-riga-01.prod.local"
            }
        },
        {
            "name": "Host memory usage",
            "system_name": "alarm.HostMemoryUsageAlarm",
            "description": "Default alarm to monitor host memory usage",
            "enabled": true,
            "key": "alarm-4.host-1004",
            "time": "2022-05-16T13:32:42.47863Z",
            "overall_status": "yellow",
            "acknowledged": false,
            "entity": {
                "id": "host-1004",
                "uuid": "5315c818-g848-8dg9-992g-50326e2bbe3a",
                "type": "HostSystem",
                "name": "esx-riga-02.prod.local",
            }
        },
        {
            // other alarms
        }
    ]
}
```

vmware.*.tags.get　以下这些监控项 **vmware.cluster.tags.get[]**, **vmware.datastore.tags.get[]**, **vmware.dc.tags.get[]**, **vmware.hv.tags.get[]**, **vmware.vm.tags.get[]** 都以如下格式返回其 JSON 对象 (具体的数据可参考示例):

```
{
    "tags": [
        {
```

```
            "name": "Windows",
        "description": "tag for cat OS type",
        "category": "OS type"
    },
    {
            "name": "SQL Server",
        "description": "tag for cat application name",
        "category": "application name"
        },
    {
            // other tags
        }
    ]
}
```

vmware.hv.diskinfo.get 监控项 **vmware.hv.diskinfo.get[]** 都以如下格式返回其 JSON 对象 (具体的数据可参考示例):

```
[
  {
    "instance": "mpx.vmhba32:C0:T0:L0",
    "hv_uuid": "8002299e-d7b9-8728-d224-76004bbb6100",
    "datastore_uuid": "",
    "operational_state": [
      "ok"
    ],
    "lun_type": "disk",
    "queue_depth": 1,
    "model": "USB DISK",
    "vendor": "SMI Corp",
    "revision": "1100",
    "serial_number": "CCYYMMDDHHmmSS9S62CK",
    "vsan": {}
  },
  {
    // other instances
  }
]
```

vmware.dvswitch.fetchports.get 监控项 **vmware.dvswitch.fetchports.get[]** 都以如下格式返回其 JSON 对象 (具体的数据可参考示例):

```
{
    "FetchDVPortsResponse":
    {
        "returnval": [
            {
                "key": "0",
                "dvsUuid": "50 36 6a 24 25 c0 10 9e-05 4a f6 ea 4e 3d 09 88",
                "portgroupKey": "dvportgroup-2023",
                "proxyHost":
                {
                    "@type": "HostSystem",
                    "#text": "host-2021"
                },
                "connectee":
                {
                    "connectedEntity":
                    {
                        "@type": "HostSystem",
                        "#text": "host-2021"
                    },
                    "nicKey": "vmnic0",
                    "type": "pnic"
```

```
                },
                "conflict": "false",
                "state":
                {
                    "runtimeInfo":
                    {
                        "linkUp": "true",
                        "blocked": "false",
                        "vlanIds":
                        {
                            "start": "0",
                            "end": "4094"
                        },
                        "trunkingMode": "true",
                        "linkPeer": "vmnic0",
                        "macAddress": "00:00:00:00:00:00",
                        "statusDetail": null,
                        "vmDirectPathGen2Active": "false",
                        "vmDirectPathGen2InactiveReasonOther": "portNptIncompatibleConnectee"
                    },
                    "stats":
                    {
                        "packetsInMulticast": "2385470",
                        "packetsOutMulticast": "45",
                        "bytesInMulticast": "309250248",
                        "bytesOutMulticast": "5890",
                        "packetsInUnicast": "155601537",
                        "packetsOutUnicast": "113008658",
                        "bytesInUnicast": "121609489384",
                        "bytesOutUnicast": "47240279759",
                        "packetsInBroadcast": "1040420",
                        "packetsOutBroadcast": "7051",
                        "bytesInBroadcast": "77339771",
                        "bytesOutBroadcast": "430392",
                        "packetsInDropped": "0",
                        "packetsOutDropped": "0",
                        "packetsInException": "0",
                        "packetsOutException": "0"
                    }
                },
                    "connectionCookie": "1702765133",
                    "lastStatusChange": "2022-03-25T14:01:11Z",
                    "hostLocalPort": "false"
            },
            {
                //other keys
            }
        ]
    }
}
```


vmware.hv.hw.sensors.get 监控项 **vmware.hv.hw.sensors.get[]** 都以如下格式返回其 JSON 对象 (具体的数据可参考示例):


```
{
    "val":
        {
            "@type": "HostHardwareStatusInfo",
            "storageStatusInfo": [
                {
                    "name": "Intel Corporation HD Graphics 630 #2",
                "status":
                        {
                            "label": "Unknown",
```

```
                    "summary": "Cannot report on the current status of the physical element",
                    "key": "Unknown"
                }
            },
            {
                    "name": "Intel Corporation 200 Series/Z370 Chipset Family USB 3.0 xHCI Controller #20"
                "status":
                        {
                     "label": "Unknown",
                     "summary": "Cannot report on the current status of the physical element",
                     "key": "Unknown"
                 }
            },
            {
                    // other hv hw sensors
                }
        ]
        }
}
```

vmware.hv.sensors.get 监控项 **vmware.hv.sensors.get[]** 都以如下格式返回其 JSON 对象 (具体的数据可参考示例):

```
{
    "val":
        {
            "@type": "ArrayOfHostNumericSensorInfo", "HostNumericSensorInfo": [
                {
                    "@type": "HostNumericSensorInfo",
                    "name": "System Board 1 PwrMeter Output --- Normal",
                    "healthState":
                        {
                            "label": "Green",
                            "summary": "Sensor is operating under normal conditions",
                    "key": "green"
                },
                    "currentReading": "10500",
                "unitModifier": "-2",
                "baseUnits": "Watts",
                "sensorType": "other"
            },
                {
                    "@type": "HostNumericSensorInfo",
                    "name": "Power Supply 1 PS 1 Output --- Normal",
                    "healthState":
                        {
                            "label": "Green",
                    "summary": "Sensor is operating under normal conditions",
                    "key": "green"
                },
                    "currentReading": "10000",
            "unitModifier": "-2",
            "baseUnits": "Watts",
            "sensorType": "power"
        },
            {
                    // other hv sensors
                }
        ]
        }
}
```

vmware.vm.snapshot.get 如果有任何的快照存在，则监控项 **vmware.snapshot.get[]** 以如下格式返回其 JSON 对象 (具体的数据可参考示例):

```
{
  "snapshot": [
    {
      "name": "VM Snapshot 4%2f1%2f2022, 9:16:39 AM",
      "description": "Descr 1",
      "createtime": "2022-04-01T06:16:51.761Z",
      "size": 5755795171,
      "uniquesize": 5755795171
    },
    {
      "name": "VM Snapshot 4%2f1%2f2022, 9:18:21 AM",
      "description": "Descr 2",
      "createtime": "2022-04-01T06:18:29.164999Z",
      "size": 118650595,
      "uniquesize": 118650595
    },
    {
      "name": "VM Snapshot 4%2f1%2f2022, 9:37:29 AM",
      "description": "Descr 3",
      "createtime": "2022-04-01T06:37:53.534999Z",
      "size": 62935016,
      "uniquesize": 62935016
    }
  ],
  "count": 3,
  "latestdate": "2022-04-01T06:37:53.534999Z",
  "latestage": 22729203,
  "oldestdate": "2022-04-01T06:16:51.761Z",
  "oldestage": 22730465,
  "size": 5937380782,
  "uniquesize": 5937380782
}
```


如果没有任何快照存在，则监控项 **vmware.snapshot.get[]** 返回其 JSON 对象但不包含任何参数：


```
{
  "snapshot": [],
  "count": 0,
  "latestdate": null,
  "latestage": 0,
  "oldestdate": null,
  "oldestage": 0,
  "size": 0,
  "uniquesize": 0
}
```


VMware 监控设置示例

概述

以下示例描述了如何设置 Zabbix 以监视 VMware 虚拟机。这涉及到：

- 创建一个代表您的 VMware 环境的主机；
- 创建一个低级别自动发现规则，用于发现 VMware 环境中的虚拟机；
- 创建一个主机原型，Zabbix 将在此基础上为低级别自动发现规则发现的虚拟机生成真实主机。

前提条件

> **Note:**
> 此示例不涵盖 VMware 的配置。假定 VMware 已经完成配置。

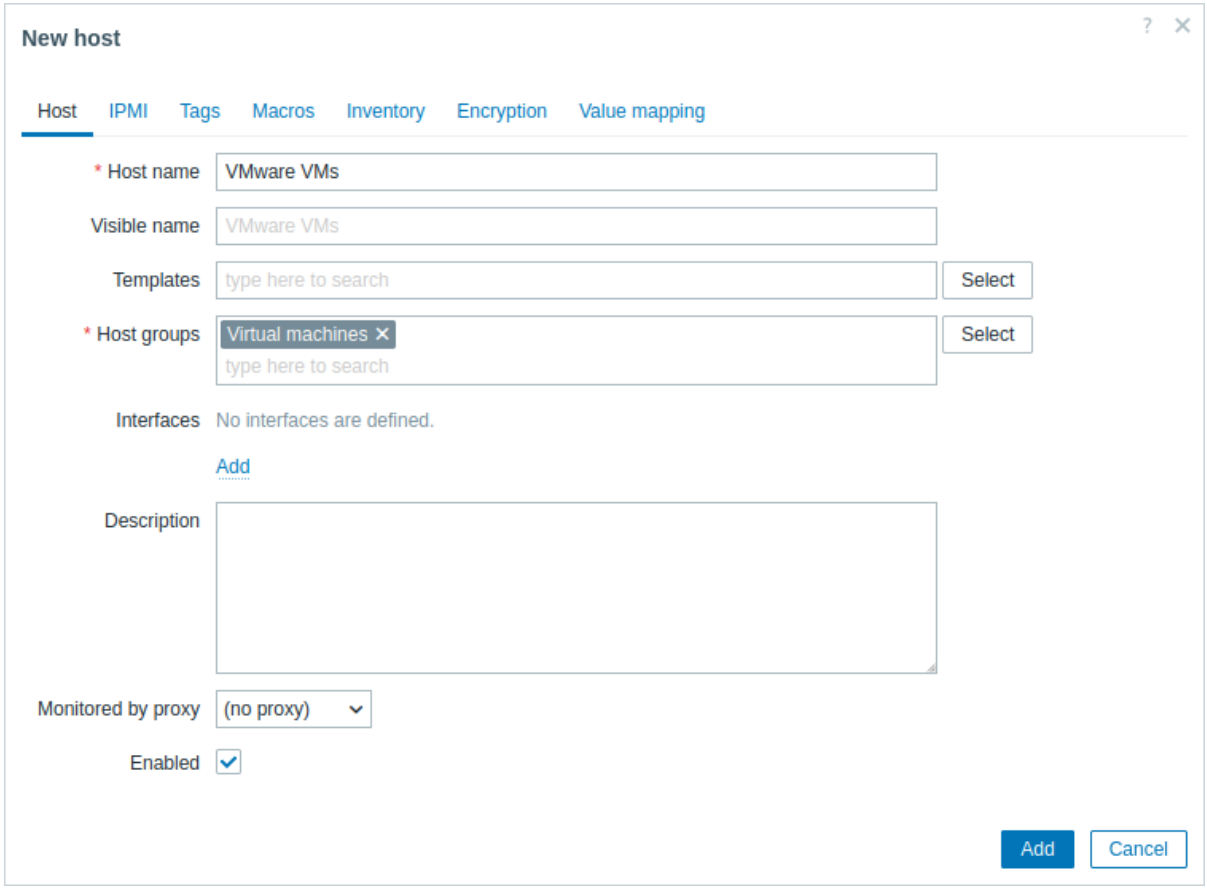
在继续之前，请将 Zabbix 服务器配置文件中的`StartVMwareCollectors` 参数设置为 2 或更大（默认值为 0）。

创建主机

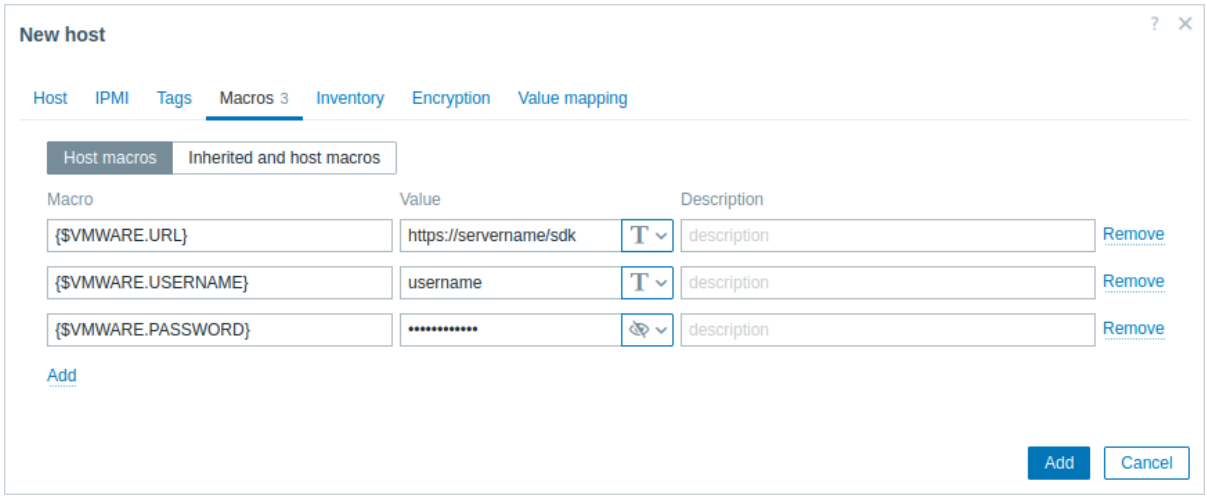
1. 转到 数据采集 > 主机。

2. 创建一个主机：

- 在 主机名称字段中，输入主机名称（例如，“VMware VMs”）。
- 在 主机群组字段中，输入或选择一个主机群组（例如，“Virtual machines”）。



- 在 宏选项卡中，设置以下主机宏：
  - {$VMWARE.URL} - VMware 服务（ESXi hypervisor）SDK URL (https://servername/sdk)
  - {$VMWARE.USERNAME} - VMware 服务用户名
  - {$VMWARE.PASSWORD} - VMware 服务 {$VMWARE.USERNAME} 用户密码



3. 点击 添加按钮创建主机。此主机将代表您的 VMware 环境。

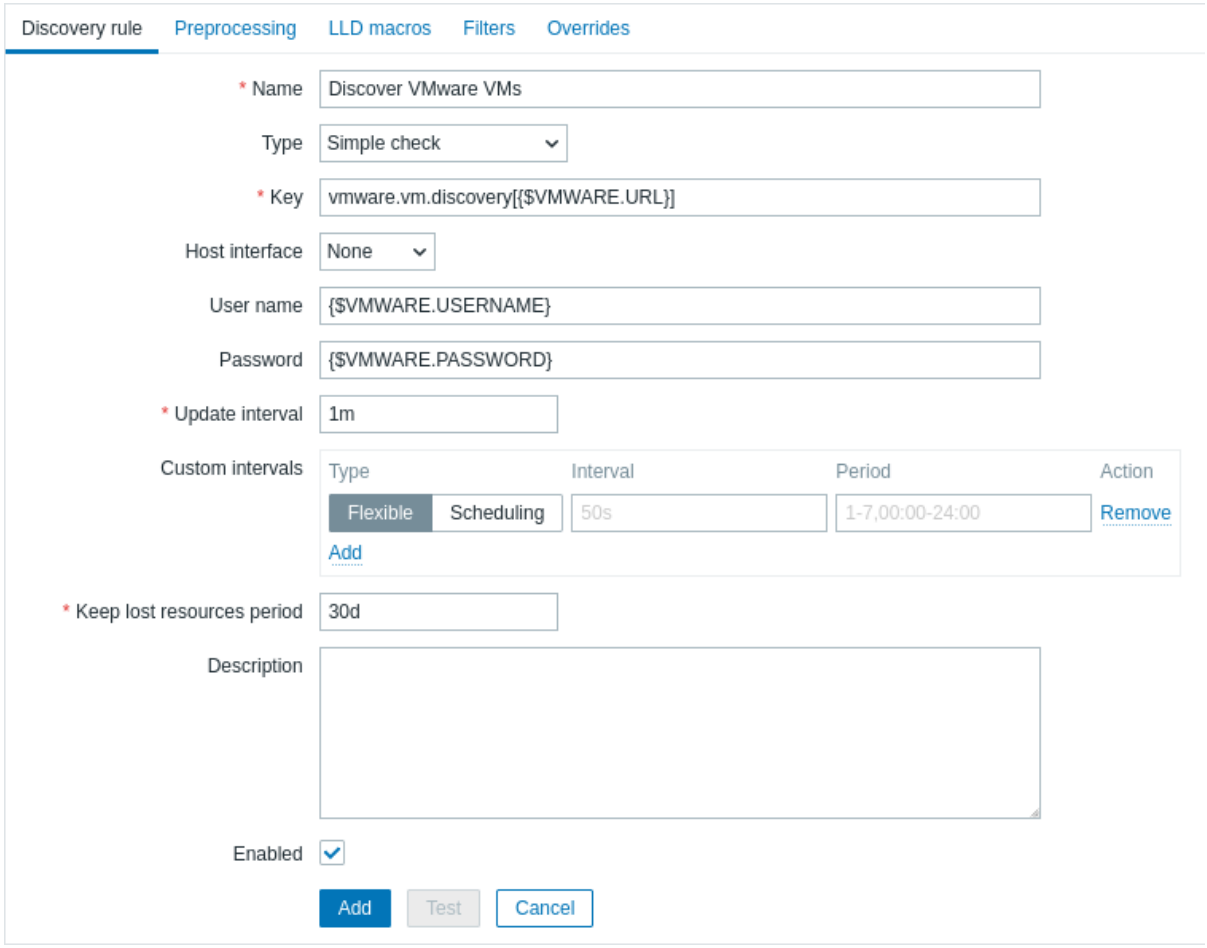
创建低级别发现规则

1. 单击已创建主机的 *Discovery*，进入该主机的低级别发现规则列表。

2. 创建 一个低级别发现规则：

- 在 *Name* 字段中，输入低级别发现规则名称（例如，“发现 VMware 虚拟机”）。

- 在 *Type* 字段中，选择 “Simple check”。
- 在 *Key* 字段中，输入用于发现 VMware 虚拟机的内置监控项键值：`vmware.vm.discovery[{$VMWARE.URL}]`
- 在 *User name* 和 *Password* 字段中，输入之前在主机上配置的相应宏。



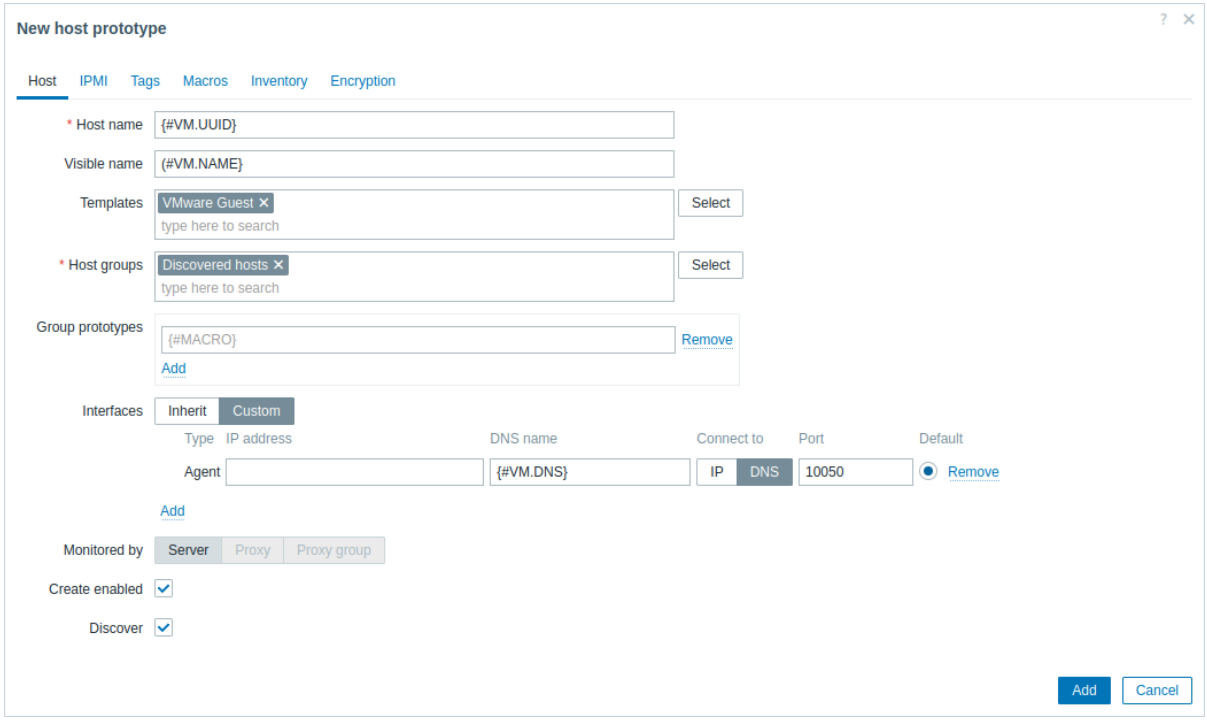
3. 单击 *Add* 按钮以创建低级别发现规则。
此发现规则将发现 VMware 环境中的虚拟机。

创建一个主机原型

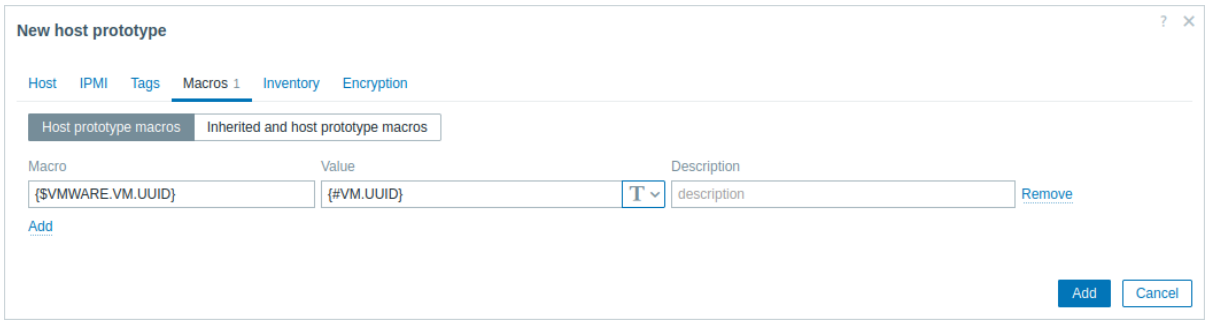
1. 在低级别发现规则列表中，为之前创建的低级别发现规则点击创建 *Host prototypes*（主机原型）。

2. 点击Create （创建）一个主机原型。由于主机原型是通过低级别自动发现规则创建主机的蓝图，因此大多数字段将包含low-level discovery macros（低级别自动发现宏）。这样可以确保主机的创建具有基于先前创建的低级别自动发现规则的content retrieved（检索到的内容）的属性。

- 在 *Host name*（主机名称）字段，输入宏 {#VM.UUID}。
- 在 *Visible name*（可见名称）字段，输入宏 {#VM.NAME}。
- 在 *Templates*（模板）字段，输入或者选择”VMware Guest”（VMware 访客）模板。该模板包括VMware items （VMware 监控项）和有关虚拟机的电源状态、CPU 利用率、内存利用率、网络设备等其它监控项的发现规则。
- 在 *Host groups*（主机组）字段，输入或者选择一个主机组 (例如，”Discovered hosts”)。
- 在 *Interfaces*（接口）字段，添加一个host interface（主机接口）。之后，输入一个宏 {#VM.DNS} 在 *DNS name* 字段或者输入一个宏 {#VM.IP} 在 *IP address* 字段。除此以外，如果用户 VMware 环境下的虚拟机存在多个接口，那么请进入*Advanced host interface configuration* （高级主机接口配置）拦。为了实现 *VMware Guest* 模板功能的正常运作，创建一个自动机主机接口是十分必要的。



- 在 *Macros*（宏）配置栏，设定 {$VMWARE.VM.UUID} 宏的值为 {#VM.UUID}。为了实现 *VMware Guest* 模板功能的正常运作使用该宏作为一个主机级别的用户宏 (例如，*vmware.vm.net.if.discovery[{$VMWARE.URL}, {$VMWARE.VM.UUID}]*).

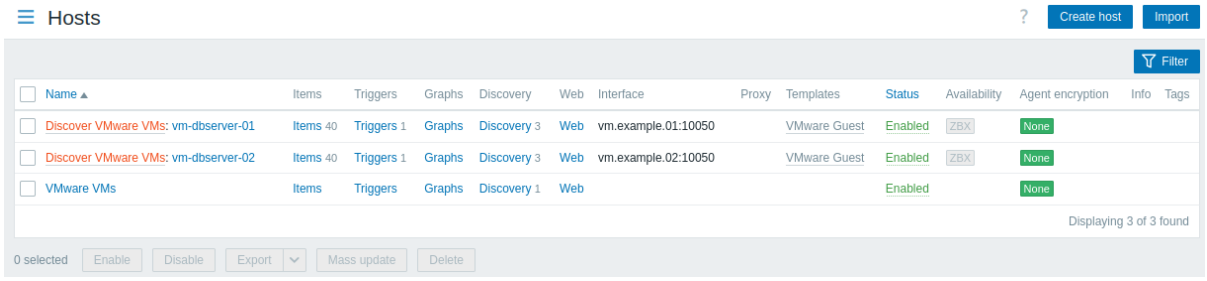


3. 点击 *Add*（添加）按钮创建一个用户原型。此主机原型将用于为先前创建的低级别自动发现规则发现的虚拟机创建主机。

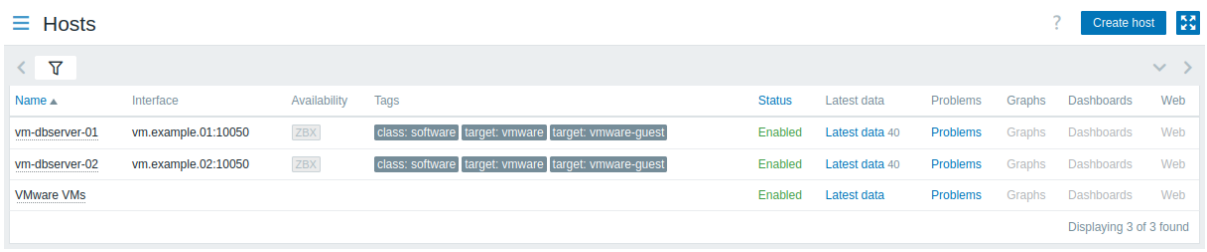
查看主机和监控项

在创建主机原型后，低级别发现规则将为已发现的 VMware 虚拟机创建主机，Zabbix 将开始监控它们。请注意，如有必要，也可以手动执行主机的发现和创建。

要查看已创建的主机，请导航到数据采集 > 主机 菜单部分。



要查看已收集的监控项，请导航到监测 > 主机 菜单部分，并为其中一个主机点击 最新数据。



高级主机接口配置

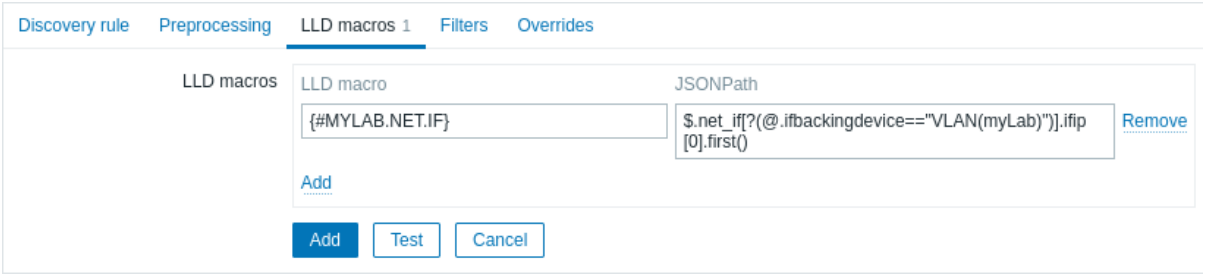
`vmware.vm.discovery[{$VMWARE.URL}]` 监控项键值，在*Create a low-level discovery rule* （创建低级别发现规则）配置栏处配置，返回的网络接口位于"net_if" field：

```
"net_if": [
      {
        "ifname": "5000",
        "ifdesc": "Network adapter 1",
        "ifmac": "00:11:22:33:44:55",
        "ifconnected": true,
        "iftype": "VirtualVmxnet3",
        "ifbackingdevice": "VLAN(myLab)",
        "ifdvswitch_uuid": "",
        "ifdvswitch_portgroup": "",
        "ifdvswitch_port": "",
        "ifip": [
          "127.0.0.1",
          "::1"
        ]
      },
      {
        "ifname": "5001",
        "ifdesc": "Network adapter 2",
        "ifmac": "00:11:22:33:44:55",
        "ifconnected": false,
        "iftype": "VirtualVmxnet3",
        "ifbackingdevice": "VLAN(myLab2)",
        "ifdvswitch_uuid": "",
        "ifdvswitch_portgroup": "",
        "ifdvswitch_port": "",
        "ifip": []
      }
    ]
```

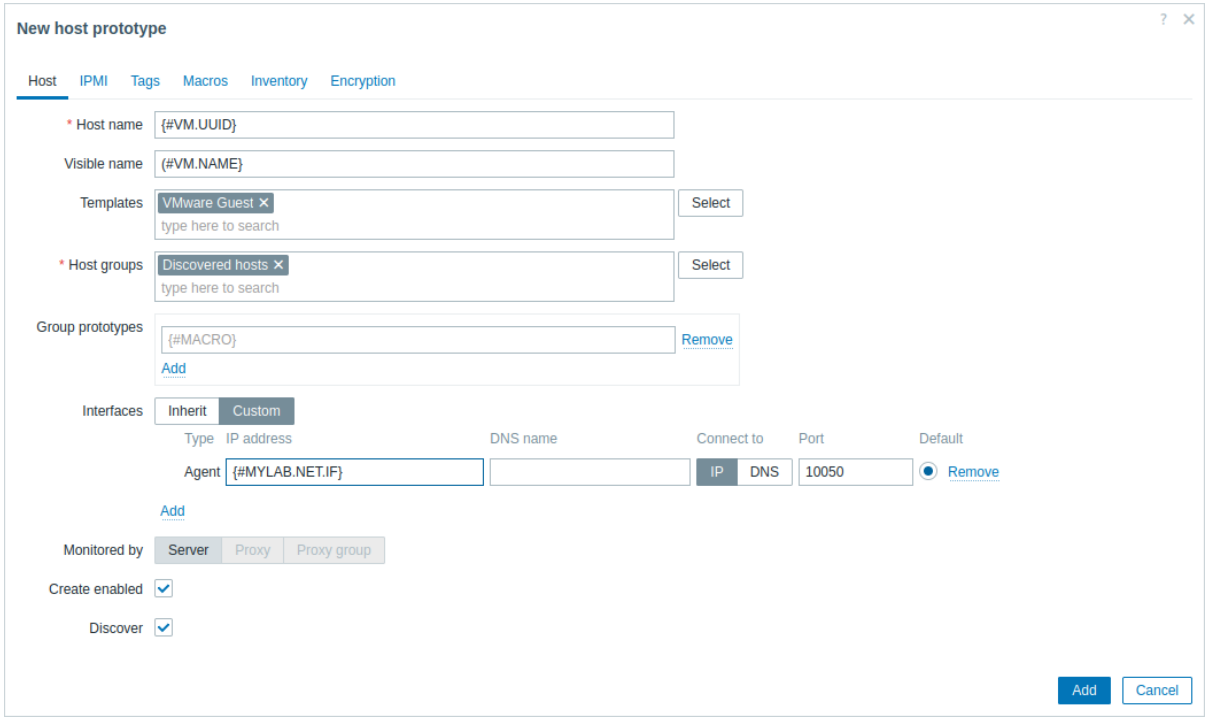
此数据可用于配置自定义主机接口。

1. 当creating a low-level discovery rule（创建一个低级别发现规则）时，另外配置一个low-level discovery macro（低级别发现宏）。在 *LLD macros* 配置选项中，创建一个具有JSONPath 值的自定义 LLD 宏。举例说明：

- `{#MYLAB.NET.IF} - $.net_if[?(@.ifbackingdevice=="VLAN(myLab)")].ifip[0].first()`



2. 当creating a host prototype（创建一个主机原型）时，添加一个自定义主机接口并输入 LLD 宏在 *DNS name* 或 *IP address* 字段。



Web 监控

概述　使用 Zabbix，您可以检查网站的多个可用性方面。

Attention:
要执行 web 监控，Zabbix 服务器必须先使用 cURL（libcurl）支持进行配置。

要启用 web 监控，您需要定义 web 场景。一个 web 场景由一个或多个 HTTP 请求或 “步骤” 组成。这些步骤由 Zabbix 服务器按预定义顺序定期执行。如果某个主机通过 proxy 监控，则这些步骤由 proxy 执行。

web 场景以与监控项、触发器等相同的方式关联到主机/模板。这意味着 web 场景也可以在模板级别创建，然后一次性应用到多个主机。

任何 web 场景都会收集以下信息：

- 整个场景所有步骤的平均每秒下载速度
- 失败的步骤编号
- 最后一个错误消息

任何 web 场景步骤都会收集以下信息：

- 每秒下载速度
- 响应时间
- 响应代码

更多详情，请参见web 监控监控项。

执行 web 场景时收集的数据会保存在数据库中。这些数据会自动用于图形、触发器和通知。

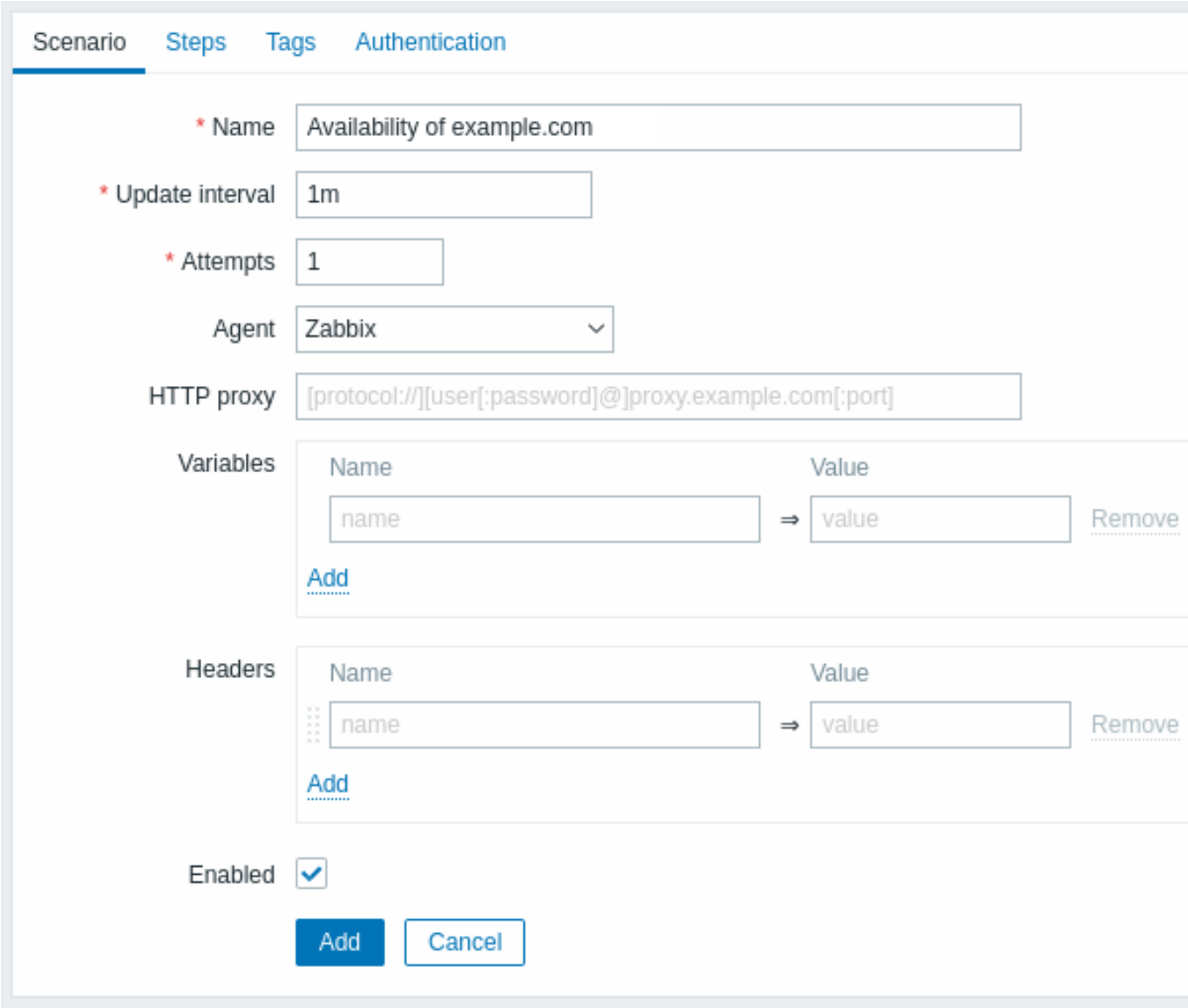
Zabbix 还可以检查检索到的 HTML 页面是否包含预定义字符串。它可以执行模拟登录，并沿着页面上的模拟鼠标点击路径进行操作。

Zabbix web 监控同时支持 HTTP 和 HTTPS。运行 web 场景时，Zabbix 可选择跟随重定向（参见下方选项 *Follow redirects*）。重定向的最大数量硬编码为 10（使用 cURL 选项 CURLOPT_MAXREDIRS）。在单个场景的执行过程中，所有 cookie 都会被保留。

配置 **Web** 场景　要配置 Web 场景：

1. 转到：数据采集 > 主机（或模板）。
2. 在主机/模板所在行中单击 *Web*。
3. 单击右侧的 创建 *Web* 场景（或单击场景名称以编辑现有场景）。
4. 在表单中输入场景参数。

场景选项卡用于配置 Web 场景的常规参数。



所有必填输入字段均以红色星号标记。

场景参数：

参数	描述
名称	唯一的场景名称。 支持用户宏。注意，如果使用用户宏，这些宏将在Web 监控监控项名称中保持未解析状态。
更新间隔	场景的执行频率。 支持时间后缀，例如 30s、1m、2h、1d。 支持用户宏。注意，如果使用用户宏且其值发生更改（例如，5m → 30s），下一次检查将按照之前的值执行（使用示例值时，执行时间会更晚）。 新建的 Web 场景将在创建后的 60 秒内进行检查。
尝试次数	执行 Web 场景步骤的尝试次数。如果出现网络问题（超时、无法连接等），Zabbix 可以多次重复执行某个步骤。设置的数值将同样应用于场景中的每个步骤。最多可指定 10 次尝试，默认值为 1。 注意：Zabbix 不会因为错误的响应代码或所需字符串不匹配而重复执行步骤。
Agent	选择客户端 agent。 Zabbix 将模拟所选浏览器。这在网站针对不同浏览器返回不同内容时非常有用。 此字段中可以使用用户宏。

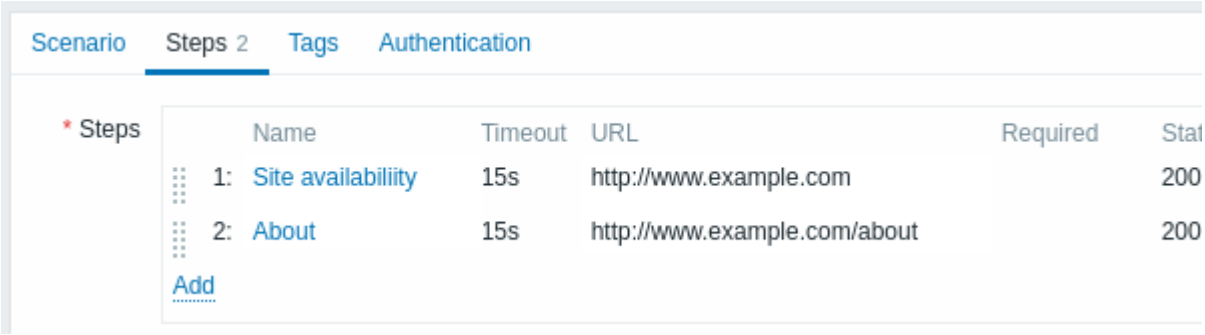
参数	描述
HTTP 代理	可以使用以下格式指定要使用的 HTTP 代理： `[protocol://][username[:password]@]proxy.example.com[:port]`。 这将设置 CURLOPT_PROXY cURL 选项。 可选的 `protocol://` 前缀可用于指定其他代理协议（cURL 7.21.7 开始支持协议前缀）。如果未指定协议，代理将被视为 HTTP 代理。 默认使用 1080 端口。 如果指定了代理，它将覆盖 http_proxy、HTTPS_PROXY 等代理相关环境变量。如果未指定代理，则不会覆盖代理相关环境变量。输入的值将按原样传递，不会进行有效性检查。 也可以输入 SOCKS 代理地址。如果指定了错误的协议，连接将失败，并且监控项将变为不支持。 注意，HTTP 代理仅支持简单身份验证。 此字段中可以使用用户宏。
变量	可在场景步骤中使用的变量（URL、post 变量）。 格式如下： **{macro1}**=value1 **{macro2}**=value2 **{macro3}**=regex:<regular expression> **{macro4}**=jsonpath:<jsonpath> **{macro5}**=xmlxpath:<xmlxpath> **{macro6}**={{macro}.function()}（请参阅宏函数） 例如： {username}=Alexei {password}=kj3h5kJ34bd {hostid}=regex:hostid is ([0-9]+) {url}=jsonpath:$.host_url {status}=xmlxpath://host/response/status {newvar}={{myvar}.btoa()} 之后，可以在步骤中引用这些宏，例如 {username}、{password}、{hostid} 等。Zabbix 将自动使用实际值替换它们。请注意，带有 `regex:` 的变量需要通过一个步骤获取其值，因此提取的值只能应用于后续步骤。 如果值部分以 `regex:` 开头，则其后的部分将被视为用于搜索网页的正则表达式；如果找到匹配项，则会将匹配结果存储在变量中。必须至少存在一个子组，以便提取匹配的值。 支持用户宏和 {HOST.*} 宏。 在查询字段或 post 变量的表单数据中使用变量时，变量会自动进行 URL 编码；但在原始 post 数据或直接在 URL 中使用变量时，必须手动进行 URL 编码。
标头	执行请求时使用 HTTP 标头。可以使用默认标头和自定义标头。 标头将根据场景级别下拉列表中所选的 Agent 类型，使用默认设置进行分配，并应用于所有步骤，除非在步骤级别对其进行了自定义定义。 请注意，在步骤级别定义标头会自动丢弃之前定义的所有标头，但场景级别下拉列表中选择**'User-Agent'** 所分配的默认标头除外。 但是，即使是'User-Agent' 默认标头，也可以通过在步骤级别指定它来覆盖。 要取消设置场景级别的标头，应在步骤级别指定该标头名称，并将其设置为空值。 标头应使用与其在 HTTP 协议中出现时相同的语法列出，也可以使用 CURLOPT_HTTPHEADER cURL 选项支持的其他功能。 例如： Accept-Charset=utf-8 Accept-Language=en-US Content-Type=application/xml; charset=utf-8 支持用户宏和 {HOST.*} 宏。
启用	选中此复选框时，场景处于活动状态；否则，场景将被禁用。

请注意，编辑现有场景时，表单中还会提供两个额外按钮：

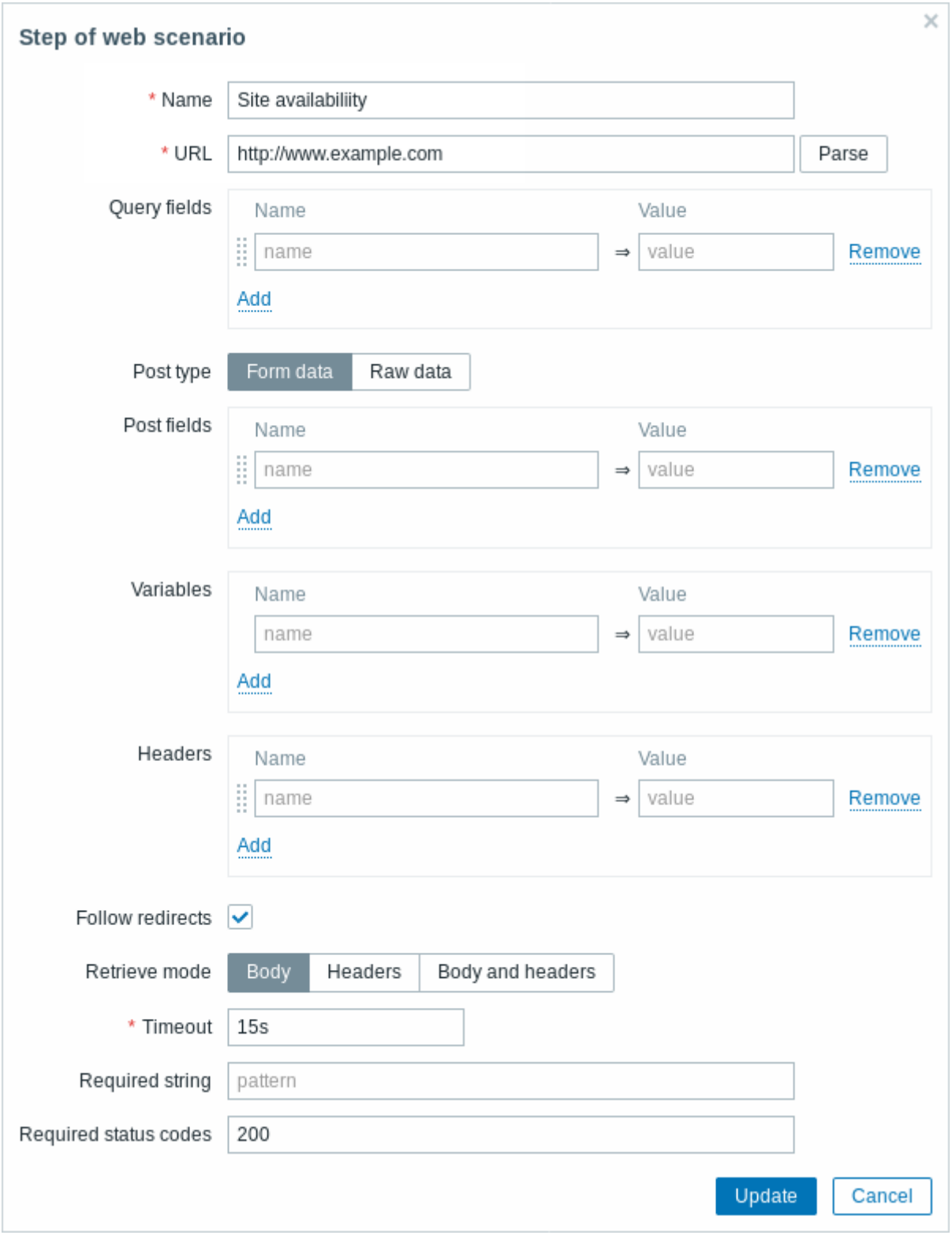
Clone	根据现有场景的属性创建另一个场景。
Clear history and trends	删除该场景的历史数据和趋势数据。删除数据后，服务器将立即执行该场景。

Note:
如果 *HTTP* 代理字段留空，还可以通过设置代理相关环境变量来使用 HTTP 代理。
对于 HTTP 检查 - 为 Zabbix 服务器用户设置 `http_proxy` 环境变量。例如：`http_proxy=http://proxy_ip:proxy_port`。
对于 HTTPS 检查 - 设置 `HTTPS_PROXY` 环境变量。例如：`HTTPS_PROXY=http://proxy_ip:proxy_port`。运行 shell 命令 `# man curl` 可获取更多详细信息。

步骤选项卡用于配置 Web 场景步骤。要添加 Web 场景步骤，请在步骤区域中单击 添加。



Note:
不得在 URL 中使用机密用户宏，因为它们将解析为”******”。



配置步骤

步骤参数：

参数	描述
名称	唯一的步骤名称。 支持用户宏。注意，如果使用用户宏，这些宏将在Web 监控监控项名称中保持未解析状态。

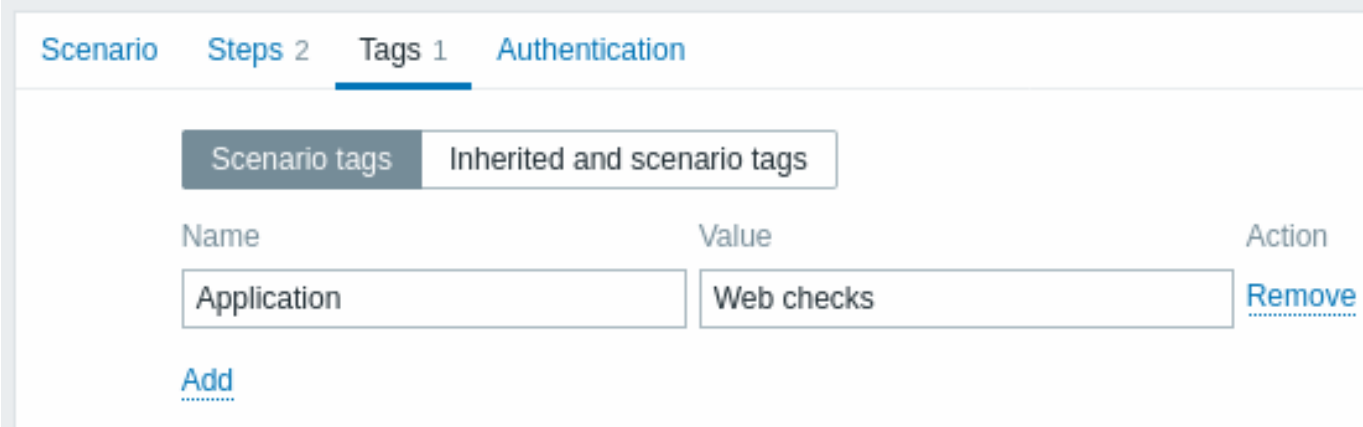
参数	描述
URL	要连接并检索数据的 URL。例如： https://www.example.com http://www.example.com/download 域名可以使用 Unicode 字符指定。在执行 Web 场景步骤时，域名会自动转换为 ASCII 格式的 punycode。 可以使用 解析按钮将可选查询字段（例如 ?key=value&limit=10）与 URL 分离，并将属性和值移至查询字段中，以便自动进行 URL 编码。 URL 中可以使用变量，语法为 {macro}。可以使用 {{macro}.urlencode()} 语法手动对变量进行 URL 编码。 支持用户宏和 {HOST.*} 宏。 出于安全原因，请勿在 URL 中包含实际的用户名或密码。请改用用户宏，以避免凭据泄露。 限制为 2048 个字符。
查询字段	URL 的 HTTP GET 变量。 以属性和值对的形式指定。 值会自动进行 URL 编码。场景变量、用户宏或 {HOST.*} 宏中的值会先解析，然后自动进行 URL 编码。使用 {{macro}.urlencode()} 语法会对其进行双重 URL 编码。 支持用户宏和 {HOST.*} 宏。
Post	HTTP POST 变量。 在表单数据模式下，以属性和值对的形式指定。 值会自动进行 URL 编码。场景变量、用户宏或 {HOST.*} 宏中的值会先解析，然后自动进行 URL 编码。 在原始数据模式下，属性和值会显示在同一行，并使用 **&** 符号连接。 可以使用 {{macro}.urlencode()} 或 {{macro}.urldecode()} 语法手动对原始值进行 URL 编码或解码。 例如：id=2345&userid={user} 如果将 {user} 定义为 Web 场景的变量，则在执行步骤时会将其替换为变量值。如果希望对变量进行 URL 编码，请将 {user} 替换为 {{user}.urlencode()}。 支持用户宏和 {HOST.*} 宏。
变量	可用于 GET 和 POST 函数的步骤级变量。 以属性和值对的形式指定。 步骤级变量会覆盖场景级变量或上一步骤中的变量。但是，步骤级变量的值只会影响后续步骤，不会影响当前步骤。 格式如下： **{macro}**=value **{macro}**=regex:< 正则表达式 > 有关更多信息，请参阅场景级别的变量描述。 变量用于查询字段或 POST 变量的表单数据时会自动进行 URL 编码，但用于原始 POST 数据或直接用于 URL 时，必须手动进行 URL 编码。
标头	执行请求时发送的自定义 HTTP 标头。 以属性和值对的形式指定。 在步骤级别定义的标头将用于相应步骤。 请注意，在步骤级别定义标头会自动丢弃之前定义的所有标头，但在场景级别从下拉列表中选择 **“User-Agent”** 所分配的默认标头除外。 但是，即使是 “User-Agent” 默认标头，也可以通过在步骤级别指定该标头来覆盖。 例如，为标头分配名称但不设置值，将取消场景级别的默认标头。 支持用户宏和 {HOST.*} 宏。 此设置对应于 cURL 的 CURLOPT_HTTPHEADER 选项。
跟随重定向	选中复选框以跟随 HTTP 重定向。 此设置对应于 cURL 的 CURLOPT_FOLLOWLOCATION 选项。
检索模式	选择检索模式： 正文 - 仅从 HTTP 响应中检索正文 标头 - 仅从 HTTP 响应中检索标头 正文和标头 - 从 HTTP 响应中检索正文和标头
超时	Zabbix 处理 URL 所用的时间不会超过设定值（从 1 秒到最长 1 小时）。实际上，此参数定义了与 URL 建立连接的最长时间，以及执行 HTTP 请求的最长时间。因此，Zabbix 在该步骤上花费的时间不会超过 **2 x Timeout** 秒。 支持时间后缀，例如 30s、1m、1h。支持用户宏。

参数	描述
必需字符串	必需的正则表达式模式。 除非检索到的内容（HTML）匹配所需模式，否则步骤将失败。如果为空，则不检查必需字符串。 例如： Homepage of Zabbix Welcome.*admin 注意：此字段不支持引用在 Zabbix 前端中创建的正则表达式。 支持用户宏和 {HOST.*} 宏。
必需状态码	预期 HTTP 状态码列表。如果 Zabbix 获取的状态码不在列表中，步骤将失败。 如果为空，则不检查状态码。 例如：200,201,210-299 支持用户宏。

Note:
只有在保存整个场景时，Web 场景步骤中的任何更改才会保存。

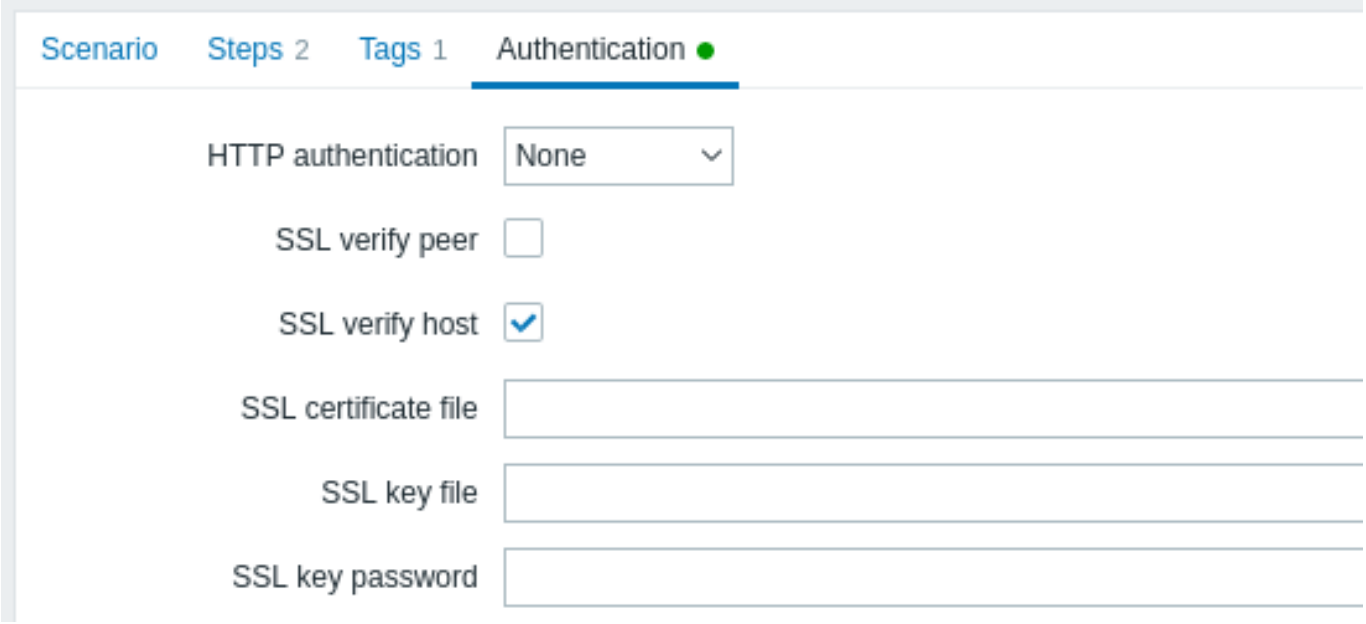
另请参阅实际示例，了解如何配置 Web 监控步骤。

配置标签　标签选项卡允许您定义场景级别的标签。



通过添加标签，可以筛选 Web 场景和Web 监控项。

配置身份验证　身份验证选项卡用于配置场景身份验证选项。选项卡名称旁的绿色圆点表示已启用某种 HTTP 身份验证。



身份验证参数：

参数	描述
HTTP authentication	选择身份验证选项： **None** - 不使用身份验证； **Basic** - 使用基本身份验证； **NTLM** - 使用 NTLM（Windows NT LAN Manager）身份验证； **Kerberos** - 使用 Kerberos 身份验证（另请参阅：使用 Zabbix 配置 Kerberos）； **Digest** - 使用摘要身份验证。
User	输入用户名（最多 255 个字符）。 当 *HTTP authentication* 设置为 Basic、NTLM、Kerberos 或 Digest 时，此字段可用。支持用户宏。
Password	输入用户密码（最多 255 个字符）。 当 *HTTP authentication* 设置为 Basic、NTLM、Kerberos 或 Digest 时，此字段可用。支持用户宏。
SSL verify peer	选中此复选框以验证 Web 服务器的 SSL 证书。 服务器证书将根据系统范围 CA 位置中的受信任证书颁发机构（CA）证书进行验证。可以使用 Zabbix 服务器或 proxy 配置参数SSLCALocation 覆盖 CA 文件的位置。 此选项设置 CURLOPT_SSL_VERIFYPEER cURL 选项。
SSL verify host	选中此复选框以验证 Web 服务器证书的 *Common Name* 字段或 *Subject Alternate Name* 字段是否匹配。 此选项设置 CURLOPT_SSL_VERIFYHOST cURL 选项。
SSL certificate file	用于客户端身份验证的 SSL 证书文件名称。证书文件必须采用 PEM[1] 格式。如果证书文件还包含私钥，请将 *SSL key file* 字段留空。如果密钥已加密，请在 *SSL key password* 字段中指定密码。包含此文件的目录由 Zabbix 服务器或 proxy 配置参数SSLCertLocation 指定。 此字段中可以使用 `HOST.*` 宏和用户宏。 此选项设置 CURLOPT_SSLCERT cURL 选项。
SSL key file	用于客户端身份验证的 SSL 私钥文件名称。私钥文件必须采用 PEM[1] 格式。包含此文件的目录由 Zabbix 服务器或 proxy 配置参数SSLKeyLocation 指定。 此字段中可以使用 `HOST.*` 宏和用户宏。 此选项设置 CURLOPT_SSLKEY cURL 选项。
SSL key password	SSL 私钥文件密码。 此字段中可以使用用户宏。 此选项设置 CURLOPT_KEYPASSWD cURL 选项。

Attention:
[1] Zabbix 仅支持 PEM 格式的证书和私钥文件。如果您的证书和私钥数据位于 PKCS #12 格式的文件中（通常扩展名为 *.p12 或 *.pfx），可以使用以下命令从中生成 PEM 文件：

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Note:
Zabbix 服务器无需重启即可获取证书变更。

Note:
如果客户端证书和私钥位于同一个文件中，只需在 “SSL certificate file” 字段中指定该文件，并将 “SSL key file” 字段留空。证书和密钥仍必须采用 PEM 格式。合并证书和密钥很简单：

```
cat client.crt client.key > client.pem
```

显示　要查看为主机配置的 Web 场景：

1. 转到 监测 > 主机。
2. 在列表中找到主机，然后单击最后一列中的 *Web* 超链接。
3. 单击场景名称以获取详细信息。



还可以通过 Web 监测小组件在 仪表板中显示 Web 场景概览。

Web 场景执行的最新结果可在 监测 > 最新数据部分查看。

扩展监控　有时需要记录接收到的 HTML 页面内容。
如果某个 web 场景步骤失败，这尤其有用。
调试级别 5（trace）用于实现这一目的。
该级别可以在服务器 和proxy 配置文件中设置，或者使用运行时控制 选项（`-R log_level_increase="http poller,N"`，其中 N 为进程编号）。
以下示例演示了在已设置调试级别 4 的情况下如何启动扩展监控：

```
# Increase log level of all http pollers:
zabbix_server -R log_level_increase="http poller"

# Increase log level of second http poller:
zabbix_server -R log_level_increase="http poller,2"
```

如果不需要扩展 web 监控，可以使用 `-R log_level_decrease` 选项停止它。

Web 监控项

概述

创建 Web 场景时会自动添加一些新的监控项以进行监控。

所有监控项都从 Web 场景继承标签。

场景监控项

场景一经创建，Zabbix 会自动添加以下监控项用于监控。

监控项	描述
场景 *<Scenario>* 的下载速度	此监控项将收集整个场景的下载速度信息（每秒字节数），即所有步骤的平均值。 监控项键值：web.test.in[Scenario,,bps] 类型：*Numeric(float)*
场景 *<Scenario>* 的失败步骤	此监控项将显示场景中失败步骤的编号。如果所有步骤都成功执行，则返回 0。 监控项键值：web.test.fail[Scenario] 类型：*Numeric(unsigned)*
场景 *<Scenario>* 的最后错误消息	此监控项返回场景的最后一条错误消息文本。仅当场景中存在失败步骤时才会存储新值。如果所有步骤都正常，则不会收集新值。 监控项键值：web.test.error[Scenario] 类型：*Character*

实际场景名称将替代 “Scenario”。

Note:
如果场景名称包含user macros，这些宏在 Web 监控项名称中将保持未解析状态。

 如果场景名称以双引号开头，或包含逗号或方括号，则会在监控项键值中被正确引用。在其他情况下，不会执行额外的引用。

Note:
Web 监控项添加后，历史数据保留期为 30 天，趋势数据保留期为 90 天。

这些监控项可用于创建触发器并定义通知条件。

示例 1

要创建”Web scenario failed（Web 场景失败）” 触发器，您可以定义触发器表达式：

```
last(/host/web.test.fail[Scenario])<>0
```

确保将’Scenario’ 替换为场景的真实名称。

示例 2

要在触发器名称中创建一个带有有用问题描述的”Web scenario failed（Web 场景失败）” 触发器，您可以使用名称定义触发器：

```
Web scenario "Scenario" failed: {ITEM.VALUE}
```

和触发表达式：

```
length(last(/host/web.test.error[Scenario]))>0 and last(/host/web.test.fail[Scenario])>0
```

确保将’Scenario’ 替换为场景的真实名称。

示例 3

要创建”Web application is slow（Web 应用程序慢）” 的触发器，可以定义以下触发器表达式：

```
last(/host/web.test.in[Scenario,,bps])<10000
```

确保将’Scenario’ 替换为场景的真实名称。

场景步骤监控项

一旦创建了一个步骤，Zabbix 就会自动添加以下监控项。

监控项	描述
Download speed for step <Step> of scenario <Scenario>	该监控项收集有关该步骤的下载速度（每秒字节数）的信息。 监控项键值：web.test.in[Scenario,Step,bps] 类型：*Numeric(float)*
Response time for step <Step> of scenario <Scenario>	该监控项收集有关步骤响应时间的信息（以秒为单位）。响应时间是从请求开始到所有信息传输完毕的时间。 监控项键值：web.test.time[Scenario,Step,resp] 类型：*Numeric(float)*
Response code for step <Step> of scenario <Scenario>	该监控项收集步骤的响应代码。 监控项键值：web.test.rspcode[Scenario,Step] 类型：*Numeric(unsigned)*

将分别使用实际场景和步骤名称代替”Scenario” 和”Step”。

Note:
Web 监控项添加了 30 天的历史记录和 90 天的趋势数据保留时长。

Note:
如果场景名称以双引号开头或包含逗号或方括号，它将在监控项键值中正确引用。在其他情况下，将不执行额外的引用。

这些监控项可用于创建触发器和定义通知条件。例如，要创建一个"Zabbix GUI login is too slow" 的触发器，可以定义一个触发器表达式：

```
last(/zabbix/web.test.time[ZABBIX GUI,Login,resp])>3
```

真实场景

概述

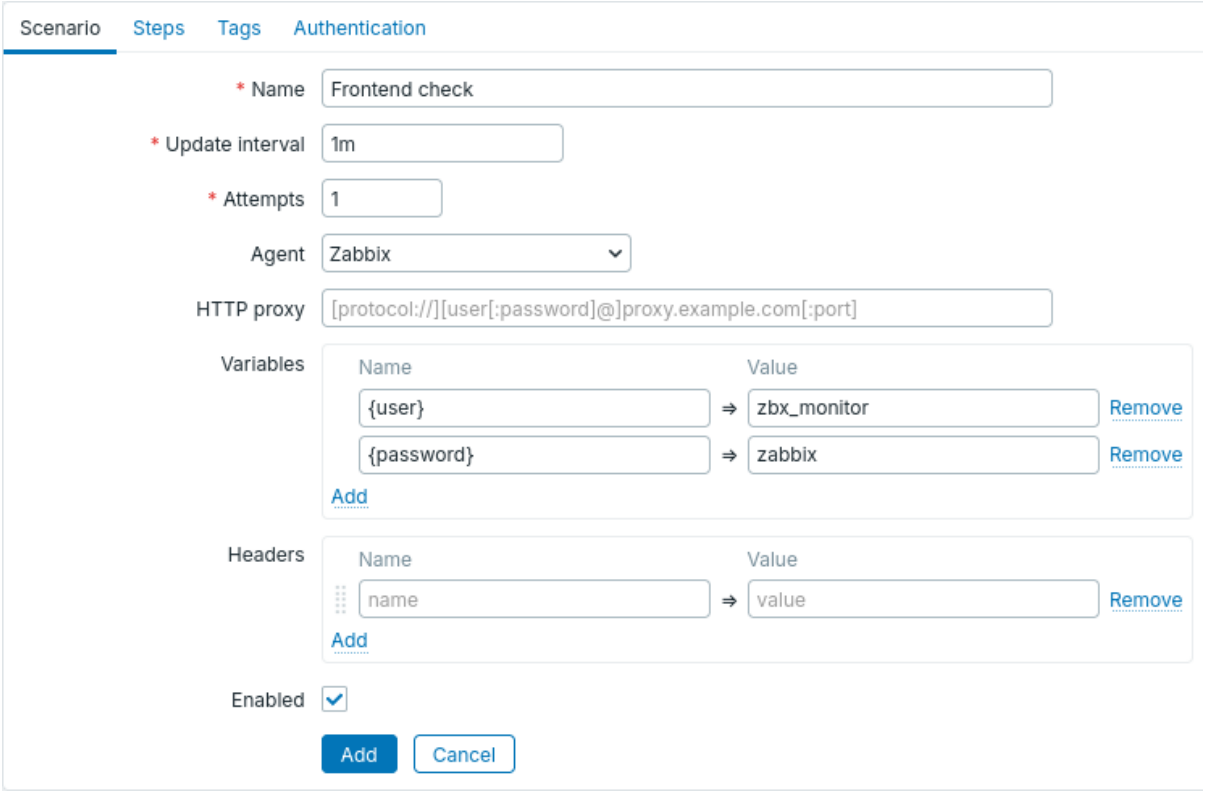
本节通过一个循序渐进的实际示例，介绍如何使用 web 监控。

将使用 Zabbix web 监控来监控 Zabbix 前端。目标是确定它是否可用、是否提供正确的内容，以及运行速度如何。为实现这一目标，需要执行多个步骤，包括检查第一页是否可访问、使用用户名和密码登录、验证登录是否成功、注销，以及确认已成功注销。

场景

添加新的 web 场景

1. 转到 *Data collection* > *Hosts*。
2. 选择一个主机，然后点击该主机所在行中的 *Web*。
3. 点击 *Create web scenario*。



4. 在新的场景表单中，打开 *Scenario* 选项卡，填写以下字段：
    - *Name*: Frontend check.
    - *Update interval*: 1m.
    - *Attempts*: 1.
    - *Agent*: Zabbix.
 在 *Variables* 部分，添加以下名称-值对：
    - `{user}`: 你现有的 Zabbix 用户凭据；
    - `{password}`: 你现有的 Zabbix 用户凭据。

Note:
出于安全考虑，建议创建一个权限最小的独立用户用于监控。

可选地，切换到 *Tags* 选项卡并添加 web 场景标签。你可以使用 web 场景标签快速识别相关的监控项和触发器，或搜索已收集的数据。例如，本教程适合使用的标签为 `component: web-scenario` 和/或 `target: frontend`。

配置 web 场景步骤

切换到 *Steps* 选项卡并为该场景定义步骤。
 配置每个步骤后，点击 *Add* 按钮。

Attention:
请注意，Zabbix 前端在登录时使用 JavaScript 重定向，因此必须先完成登录，且只有在后续步骤中才能检查已登录功能。此外，登录步骤必须使用指向 **index.php** 文件的完整 URL。

所有步骤的通用字段

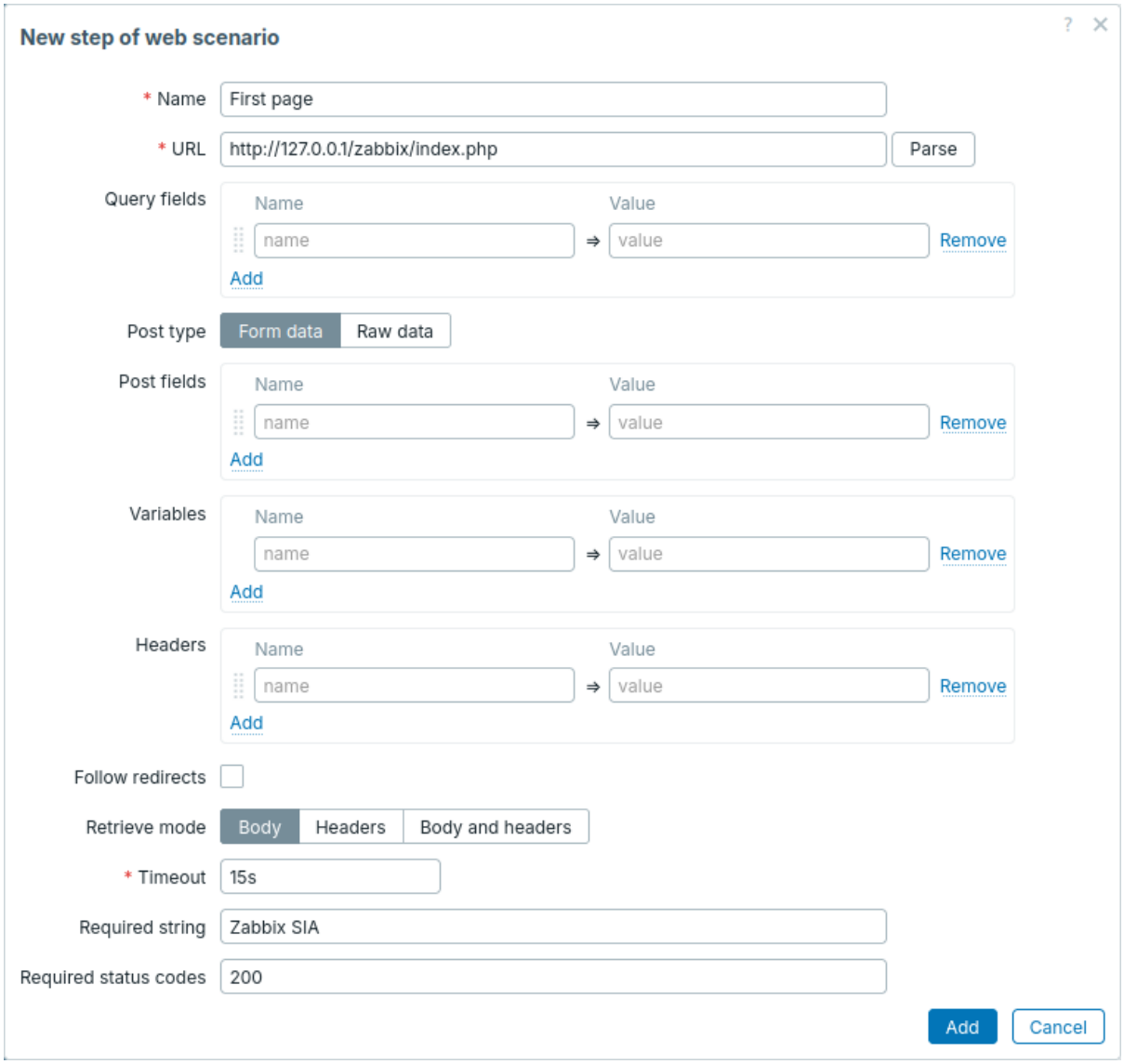
除了特定于步骤的字段外，还需为下面所述的每个步骤填写以下通用字段：

- *URL*：Zabbix 前端的 URL(`http://<server_ip_or_name>/zabbix/index.php`(适用于 Apache)或 `http://<server_ip_or_name`(适用于 NGINX))。对于 Web 场景步骤 3，请使用下面所述的其他 URL。
- *Timeout*：15s。
- *Required status codes*：200。

Web 场景步骤 1

检查第一页是否正确响应，返回 HTTP 状态码 200，并包含文本 *Zabbix SIA*。
 填写以下字段：

- 名称: First page.
- 超时: 15s.
- 必需字符串: Zabbix SIA.
- 所有步骤的通用字段。

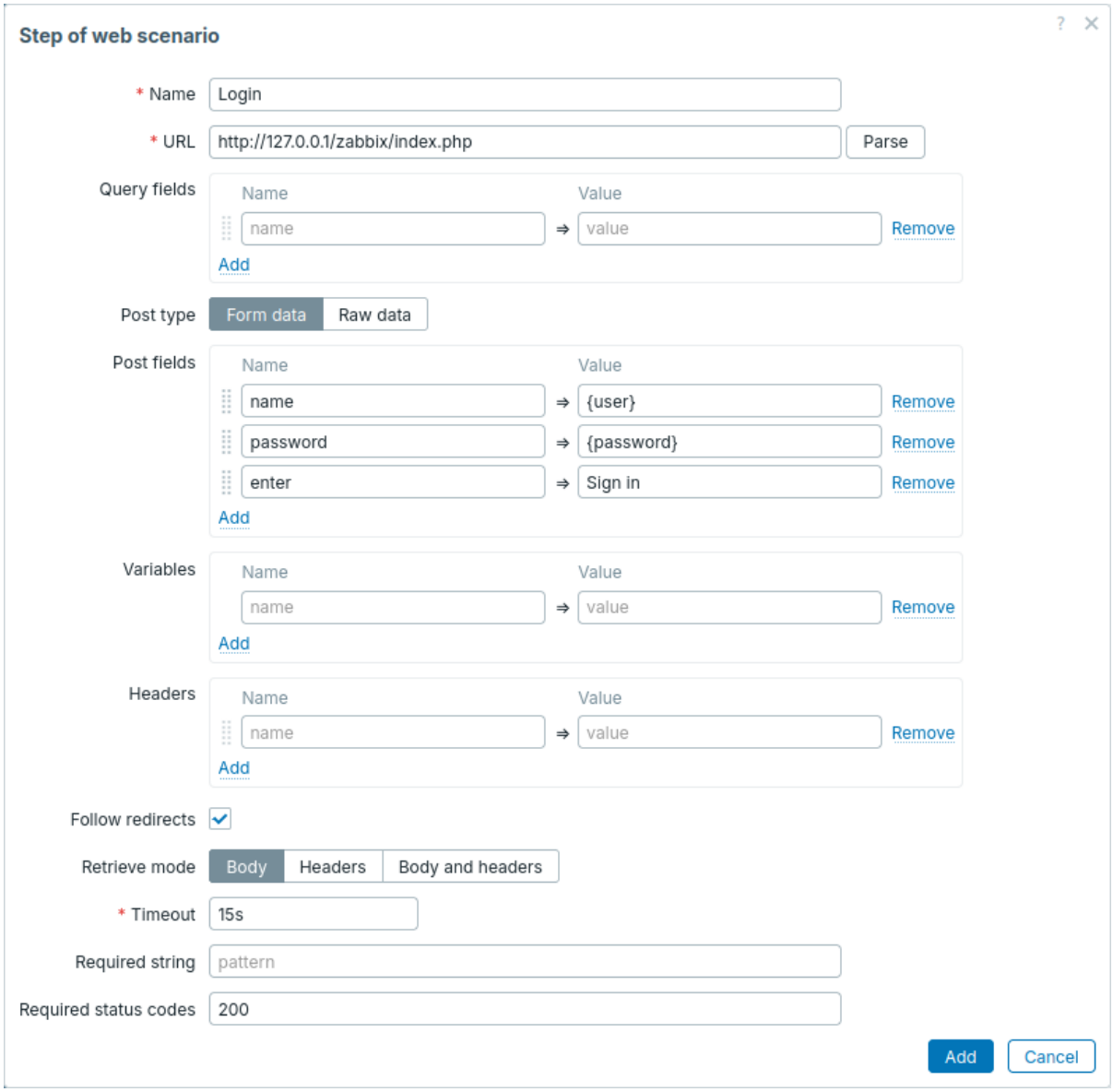


Web 场景步骤 2

使用 *Scenario* 选项卡中定义的宏（变量）*{user}* 和 *{password}* 登录 Zabbix 前端。
 填写以下字段：

- *Name*: Login.
- 在 *Post fields* 部分，添加以下名称-值对：
  - name: `{user}`;
  - password: `{password}`;
  - enter: Sign in.
- 所有步骤的通用字段。

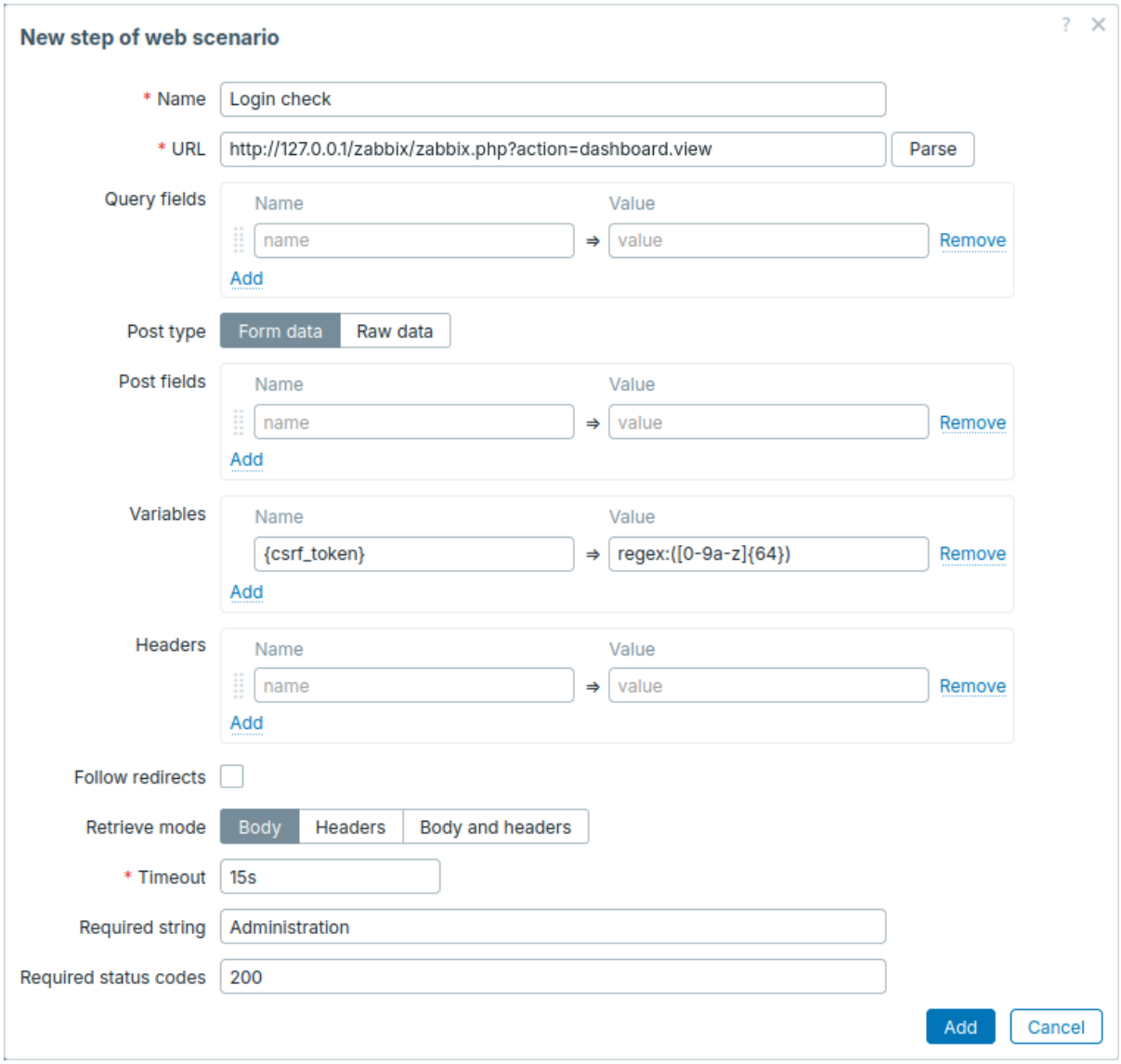
选中 *Follow redirects* 复选框。



Web 场景步骤 3

登录后，通过检查仅在登录后可见的字符串来验证是否成功，例如 *Administration*。
 填写以下字段：

- *Name*: Login check.
- *URL*: 用户登录后被重定向到的 URL（例如，http://127.0.0.1/zabbix/zabbix.php?action=dashboard.view）。
- 在 *Variables* 部分，添加名称-值对 `{csrf_token}`: `regex:([0-9a-z]{64})`。此变量将捕获分配的 CSRF 令牌值，以便在web 场景步骤 4 中复用。
- *Required string*: Administration.
- 所有步骤的通用字段。



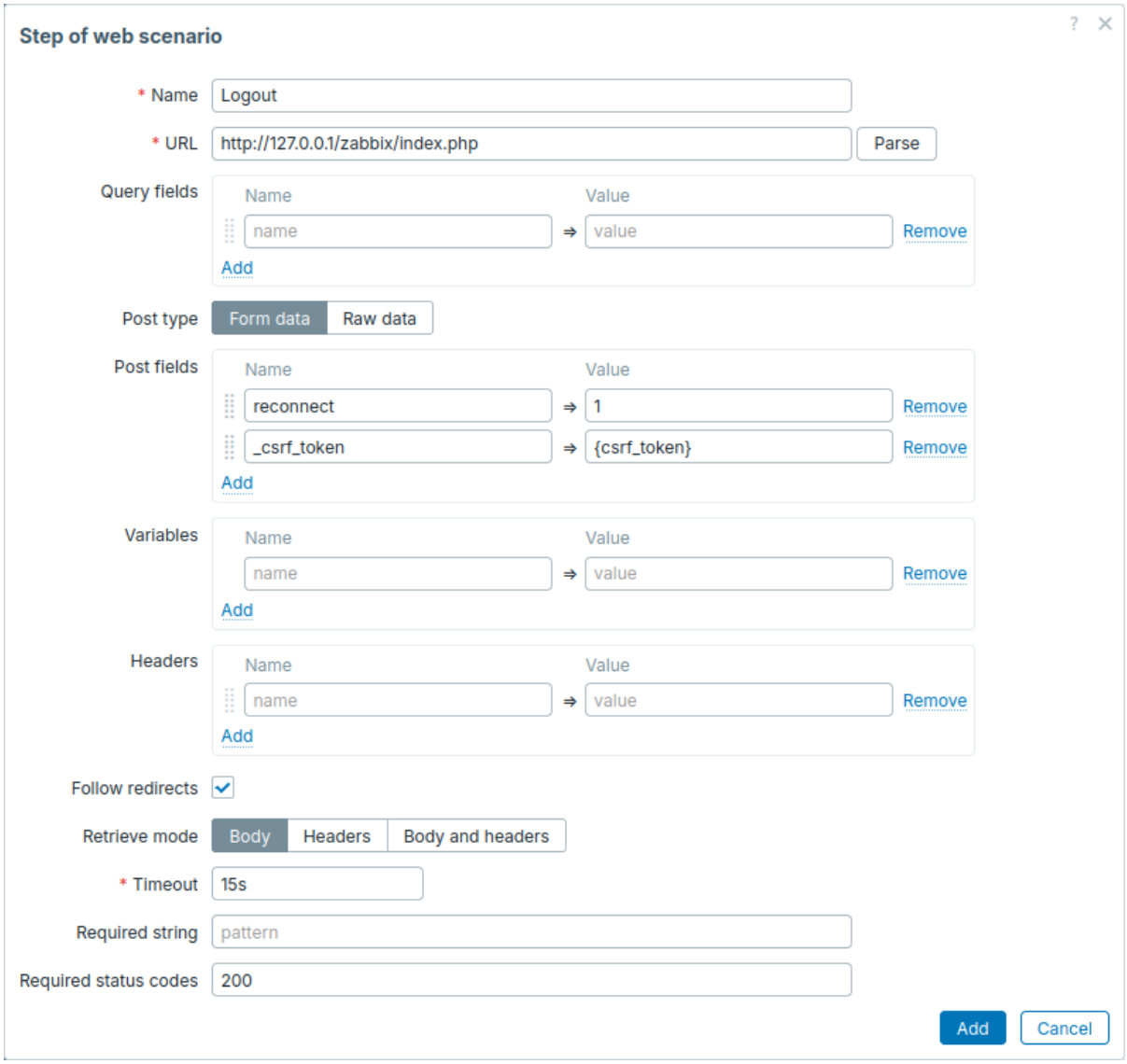
Web 场景步骤 4

确认前端可访问并且登录成功后，添加注销步骤；否则，Zabbix 数据库中将积累大量未关闭的会话记录。
 填写以下字段：

- 名称：注销。
- 在 *Post* 字段部分，添加以下名称-值对：
  - reconnect：1；
  - _csrf_token：`{csrf_token}`。
- 所有步骤的通用字段。

选中 跟随重定向复选框。

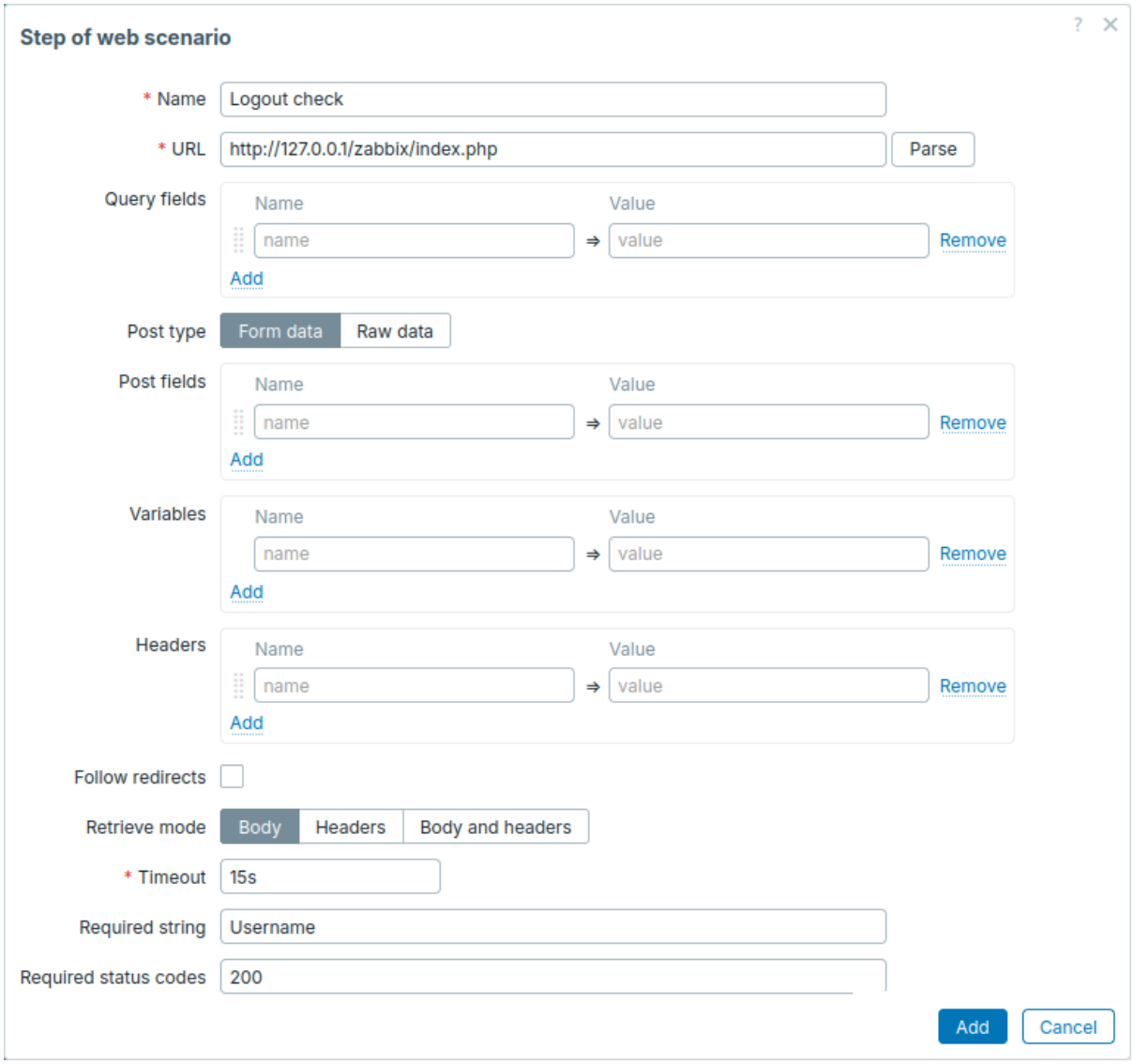
此步骤使用在Web 场景步骤 3 中获取的变量 `{csrf_token}`。



Web 场景步骤 5

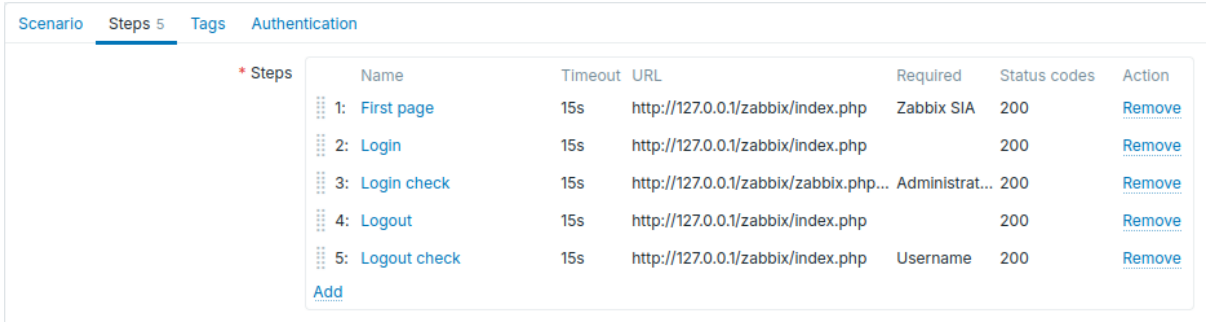
要确认注销，请检查 *Username* 字符串。
 填写以下字段：

- *Name*: Logout check.
- *Required string*: Username.
- 所有步骤的通用字段。



步骤的完整配置

web 场景步骤的完整配置应如下所示：

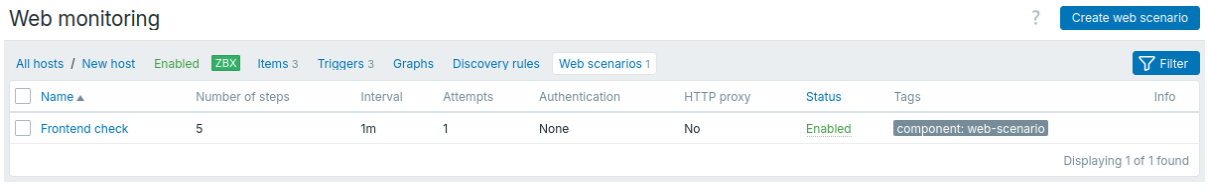


检查结果

保存已完成的 web 监控场景。

创建 web 场景后，会自动添加新的web 监控项 用于监控。

该场景将被添加到主机。要查看 web 场景信息，请转到 *Monitoring* > *Hosts*，在列表中找到该主机，然后单击最后一列中的 *Web* 超链接。



单击场景名称以查看详细统计信息：



服务监控

概述　服务监控是一种业务级监控，可用于概览整个 IT 基础设施服务树、识别基础设施中的薄弱环节、计算各种 IT 服务的 SLA，以及查看更高层级的其他信息。
服务监控关注服务的整体可用性，而不是磁盘空间不足、处理器负载过高等底层细节。
如果服务未按预期运行，服务监控还提供查找问题根本原因的功能。

服务监控提供了一种创建受监控数据层次结构的方法。

一个非常简单的服务结构可能如下所示：

```
服务
|
|-工作站
|  |
|  |-工作站1
|  |
|  |-工作站2
|
|-服务器
```

结构中的每个节点都有状态属性。
根据所选算法计算状态，并将其向上传播到上层。
各个节点的状态会受到已映射问题状态的影响。
问题映射通过标签完成。

如果检测到服务状态发生变化，Zabbix 可以发送通知，或在 Zabbix 服务器上自动执行脚本。您可以定义灵活的规则，以确定父服务是否应根据子服务的状态进入 “问题状态”。

随后，可以使用服务问题数据计算 SLA，并根据灵活的条件集发送 SLA 报告。

服务监控在“服务”菜单中配置，该菜单包含以下部分：

- 服务

“服务”部分允许您通过添加父服务，然后向父服务添加子服务，来构建受监控基础设施的层次结构。

除了配置服务树之外，此部分还提供整个基础设施的概览，并帮助您快速识别导致服务状态发生变化的问题。

- SLA

在此部分中，您可以定义服务级别协议，并为特定服务设置服务级别目标。

- SLA 报告

在此部分中，您可以查看 SLA 报告。

服务操作

您还可以配置服务操作。

服务操作是可选的，可用于：

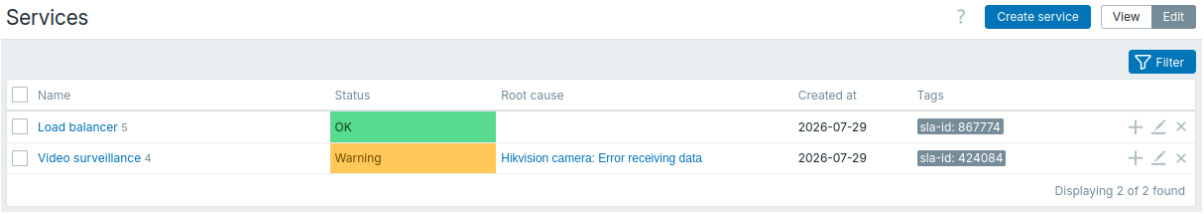
- 发送服务不可用通知
- 服务状态发生变化时，在 Zabbix 服务器上执行远程命令
- 服务恢复后发送恢复通知。

另请参阅：

- SLA 监控配置示例
- 关于从低于 6.0 的 Zabbix 版本升级服务的说明

服务树

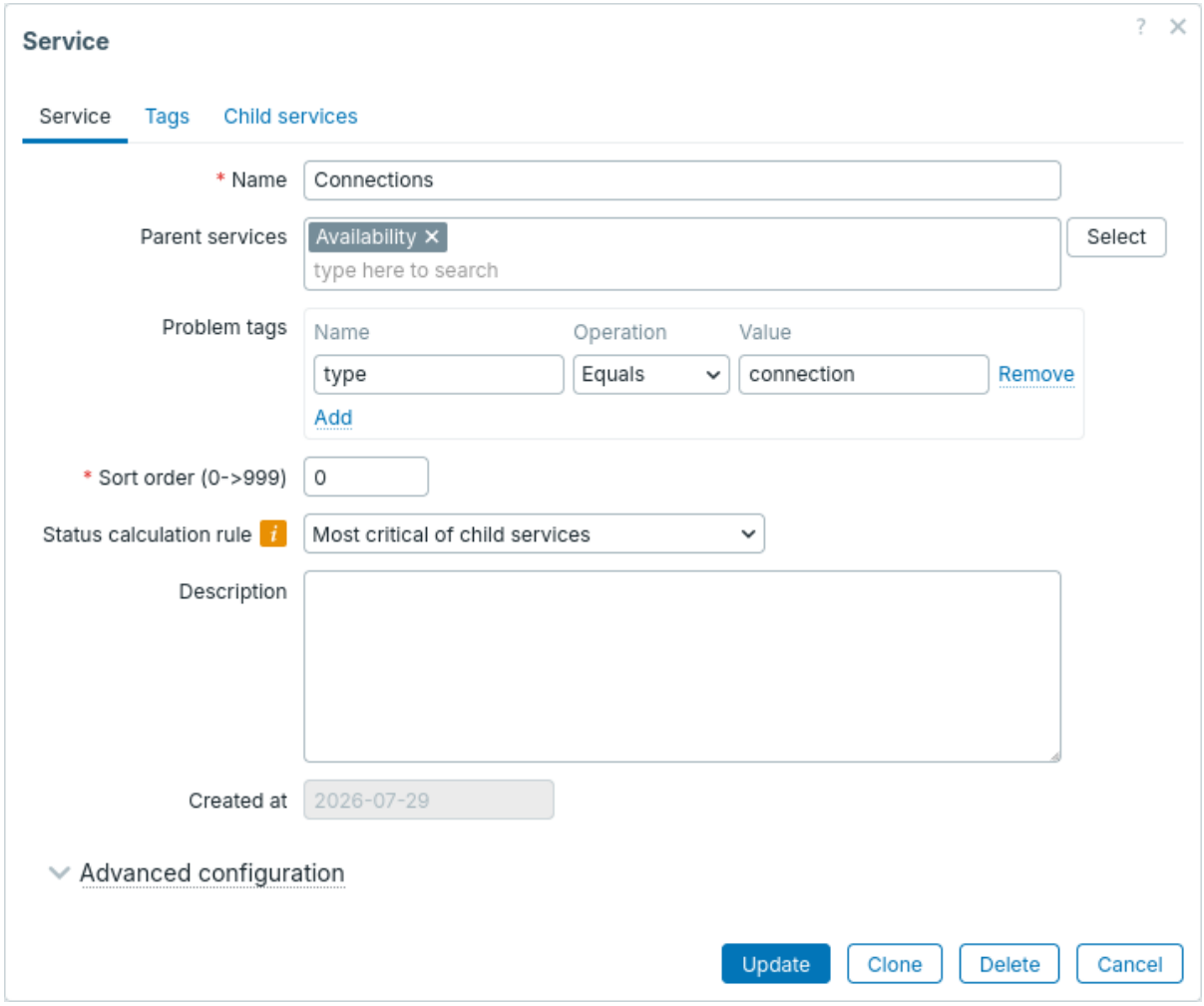
服务树在 *Services* > *Services* 菜单部分中进行配置。在右上角，将view 切换为编辑模式。



要配置新的服务，请点击右上角的 *Create service* 按钮。

如需快速添加子服务，也可以点击父服务旁边的加号图标。这将打开相同的服务配置表单，但 *Parent services* 参数会预先填充。

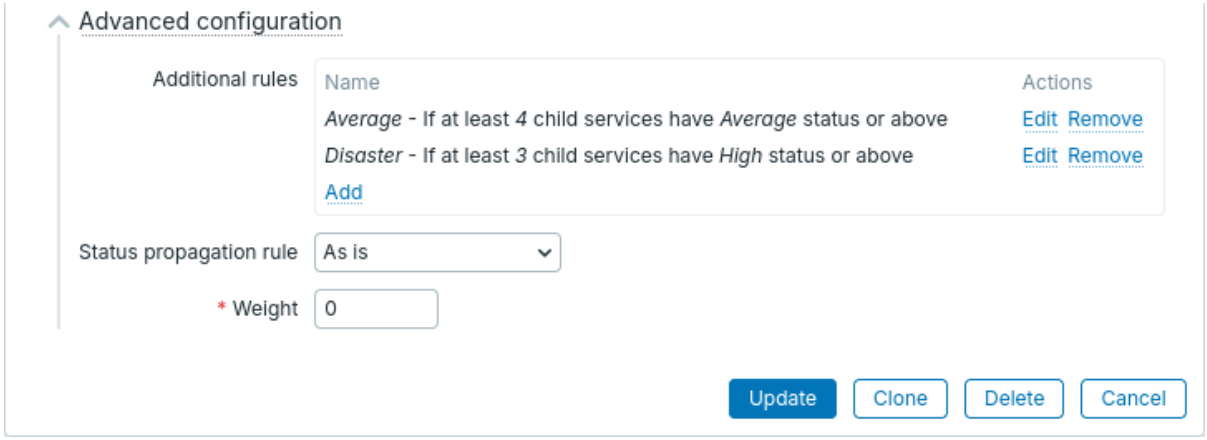
服务配置　在 **Service** 选项卡中，指定所需的服务参数：



所有必填输入字段都用红色星号标记。

Parameter	Description
Name	服务名称。
Parent services	该服务所属的父服务。 如果要添加最高级别的服务，请将此字段留空。 一个服务可以有多个父服务。在这种情况下，它会显示在服务树中每个父服务下。
Problem tags	指定用于将问题数据映射到该服务的标签。 可以设置多个条件。标签名称匹配始终区分大小写。 每个条件有两个可用运算符： **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，区分大小写）
Sort order	显示顺序，数值越小越靠前。
Status calculation rule	服务状态的计算规则： **Most critical if all children have problems** - 如果所有子项都有问题，则使用子服务中最严重的问题来为服务状态着色 **Most critical of child services** - 使用子服务中最严重的问题来为服务状态着色 **Set status to OK** - 不计算服务状态 可在高级配置选项中配置其他状态计算规则。
Description	服务描述。
Created at	服务创建时间；在编辑现有服务时显示。
Advanced configuration	单击 *Advanced configuration* 标题以显示高级配置选项。

高级配置



参数		说明
附加规则		单击 *Add* 以配置附加的状态计算规则。
	状态设置为	在条件匹配时，将服务状态设置为 *OK*（默认）、*Not classified*、*Information*、*Warning*、*Average*、*High* 或 *Disaster*。
	条件	为直接子服务选择条件： 如果至少有 **(N)** 个子服务的状态为 **(Status)** 或更高 如果至少有 **(N%)** 的子服务的状态为 **(Status)** 或更高 如果少于 **(N)** 个子服务的状态为 **(Status)** 或更低 如果少于 **(N%)** 的子服务的状态为 **(Status)** 或更低 如果状态为 **(Status)** 或更高的子服务权重至少为 **(W)** 如果状态为 **(Status)** 或更高的子服务权重至少为 **(N%)** 如果状态为 **(Status)** 或更低的子服务权重小于 **(W)** 如果状态为 **(Status)** 或更低的子服务权重小于 **(N%)** 如果指定了多个条件，且情况同时匹配多个条件，则将设置最高严重级别。
	N (W)	在条件中设置 N 或 W 的值（1-100000），或 N% 的值（1-100）。
	Status	在条件中选择 *Status* 的值：*OK*（默认）、*Not classified*、*Information*、*Warning*、*Average*、*High* 或 *Disaster*。
状态传播规则		将服务状态传播到父服务的规则： **As is** - 状态将不作更改地传播 **Increase by** - 可将传播的状态提高 1 到 5 个严重级别 **Decrease by** - 可将传播的状态降低 1 到 5 个严重级别 **Ignore this service** - 状态不会传播到父服务 **Fixed status** - 状态以静态方式传播，即始终保持相同状态
权重		服务的权重（整数，范围从 0（默认）到 1000000）。

Note:
附加状态计算规则只能用于将严重级别提高到基于主要 *Status calculation rule* 参数计算出的级别之上。如果根据附加规则状态应为 Warning，但根据 *Status calculation rule* 状态为 Disaster，则服务状态将为 Disaster。

Tags 选项卡包含服务级别标签。服务级别标签用于标识服务。此类标签不用于将问题映射到服务（如需执行此操作，请使用第一个选项卡中的问题标签）。

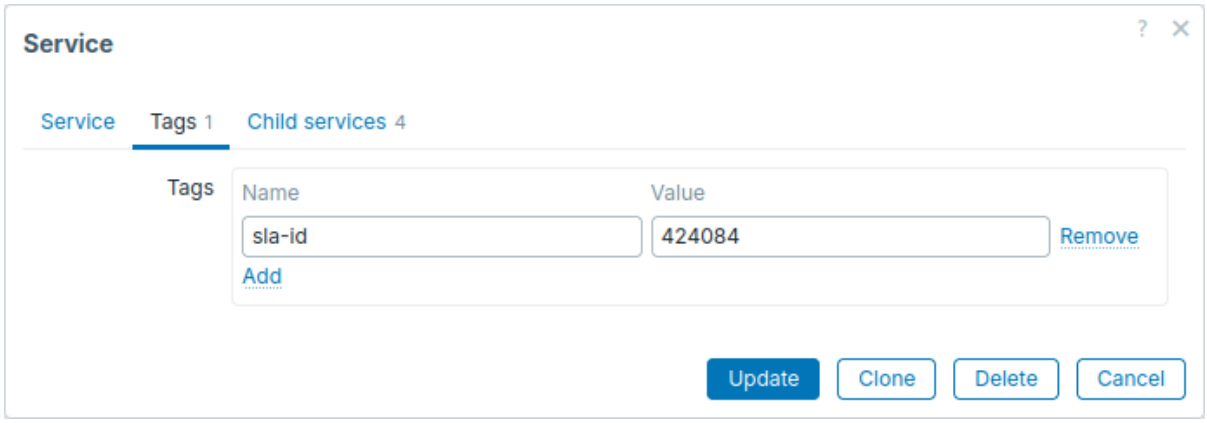
Child services 选项卡允许您指定依赖服务。点击 *Add*，从现有服务列表中添加服务。如果要添加新的子服务，请先保存此服务，然后点击刚创建的服务旁边的加号图标。

标签　服务中有两种不同类型的标签：

- 服务标签
- 问题标签

服务标签

服务标签用于将服务与服务操作和SLA进行匹配。这些标签在 *Tags* 服务配置选项卡中指定。对于 SLA 映射，使用 *OR* 逻辑：如果某个服务至少有一个匹配的标签，则会将其映射到该 SLA。在服务操作中，映射规则可配置，并且可以使用 *AND*、*OR* 或 *AND/OR* 逻辑。

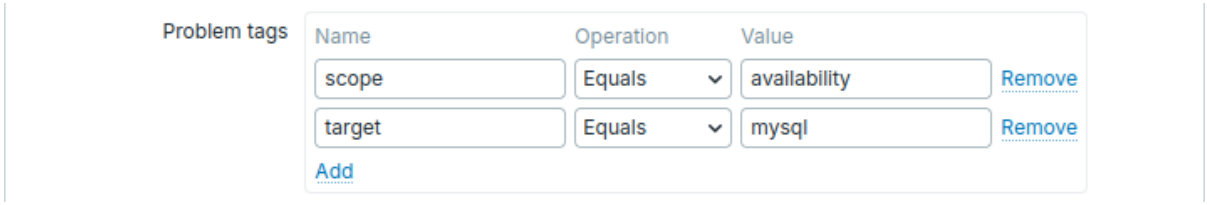


问题标签

问题标签用于匹配问题和服务。这些标签在主服务配置选项卡中指定。

只有最低层级的子服务可以定义问题标签，并直接与问题关联。如果问题标签匹配，服务状态将变为与问题相同的状态。若存在多个问题，服务将采用最严重问题的状态。随后，父服务的状态会根据子服务状态并按照状态计算规则进行计算。

如果指定了多个标签，则使用 *AND* 逻辑：问题必须具有在服务配置中指定的所有标签，才能映射到该服务。

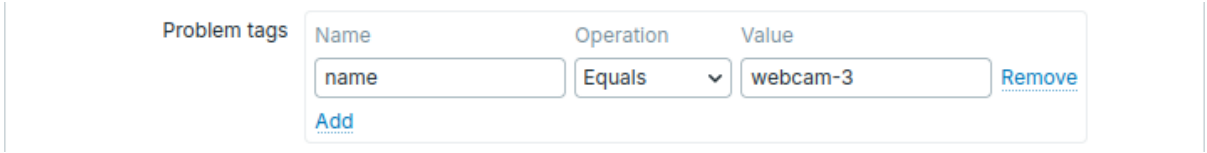


Note:
Zabbix 中的问题会继承来自整个模板、主机、监控项、web 场景和触发器链上的标签。这些标签中的任意一个都可用于将问题匹配到服务。

示例：

问题 *Web camera 3 is down* 的标签为 `type:video-surveillance`、`floor:1` 和 `name:webcam-3`，状态为 *Warning*

服务 **Web camera 3** 只有一个指定的问题标签：`name:webcam-3`



当检测到此问题时，服务状态将从 *OK* 变为 *Warning*。

如果服务 **Web camera 3** 的问题标签为 `name:webcam-3` 和 `floor:2`，那么在检测到该问题时，其状态不会改变，因为条件只满足了一部分。

修改现有服务

Note:
下文描述的按钮仅在 *Services* 部分处于编辑模式时可见。

要编辑现有服务，请单击服务旁边的铅笔图标。

要克隆现有服务，请单击铅笔图标打开其配置，然后单击 *Clone* 按钮。克隆服务时，会保留其父级链接，而子级链接不会保留。

要删除服务，请单击其旁边的 x 图标。删除父级服务时，其子服务不会被删除，而是会在服务树中上移一级（第 1 级子服务将与已删除的父级服务处于同一级别）。

服务列表下方的两个按钮提供一些批量编辑选项：

- *Mass update* - 批量更新服务属性
- *Delete* - 删除服务

要使用这些选项，请先勾选相应服务前面的复选框，然后单击所需按钮。

SLA

概述　创建服务后，您可以开始监控其性能是否达到服务级别协议（SLA）的要求。

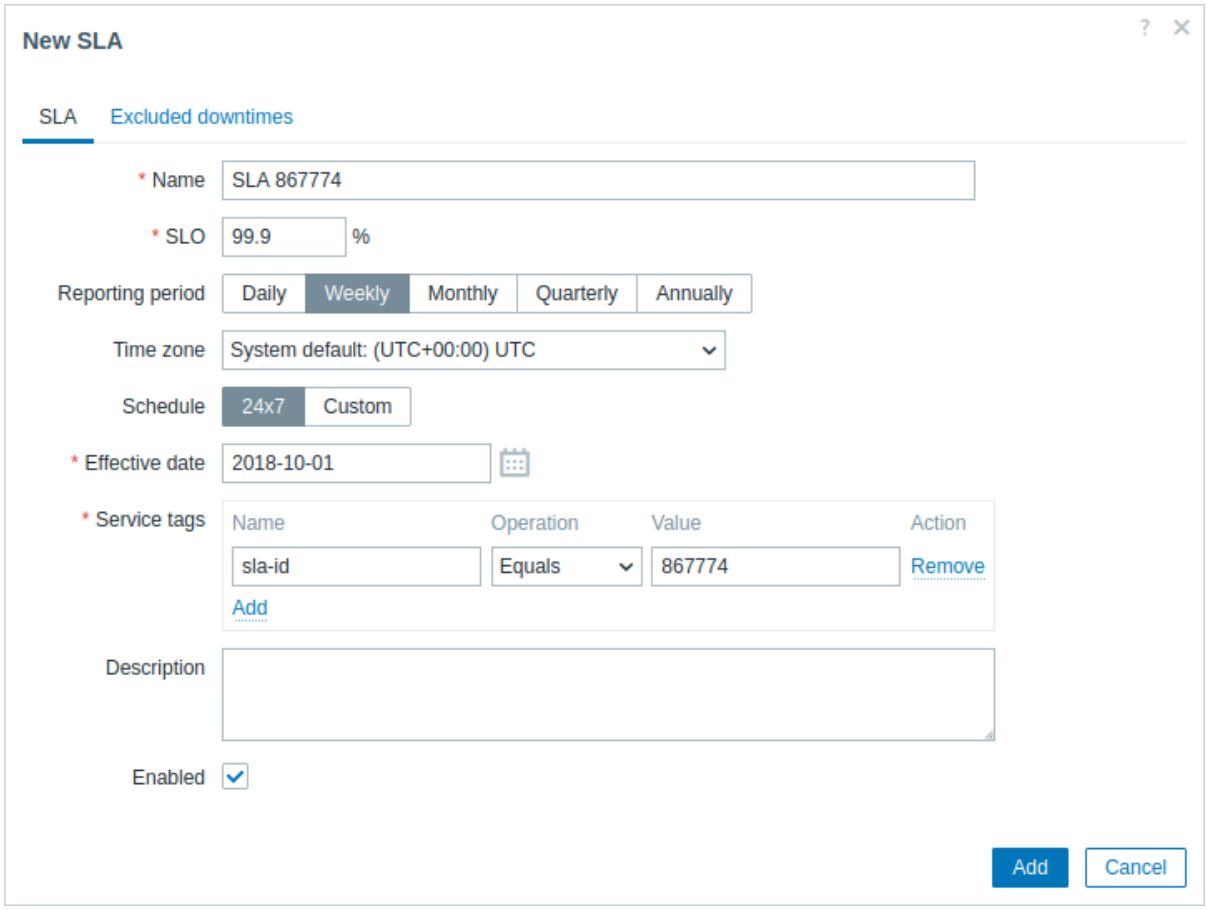
服务 > *SLA* 菜单部分用于为各种服务配置 SLA。Zabbix 中的 SLA 定义了服务级别目标（SLO）、预期的正常运行时间计划和计划内停机时间。

SLA 和服务通过服务标签进行匹配。同一个 SLA 可以应用于多个服务 - 系统会分别测量每个匹配服务的性能。单个服务可以分配多个 SLA - 每个 SLA 的数据将分别显示。

在 SLA 报告中，Zabbix 提供服务级别指标（SLI）数据，用于衡量实际的服务可用性。服务是否达到 SLA 目标，取决于将 SLO（预期可用性，以% 表示）与 SLI（实际可用性，以% 表示）进行比较。

配置　要创建新的 SLA，请点击 创建 *SLA* 按钮。

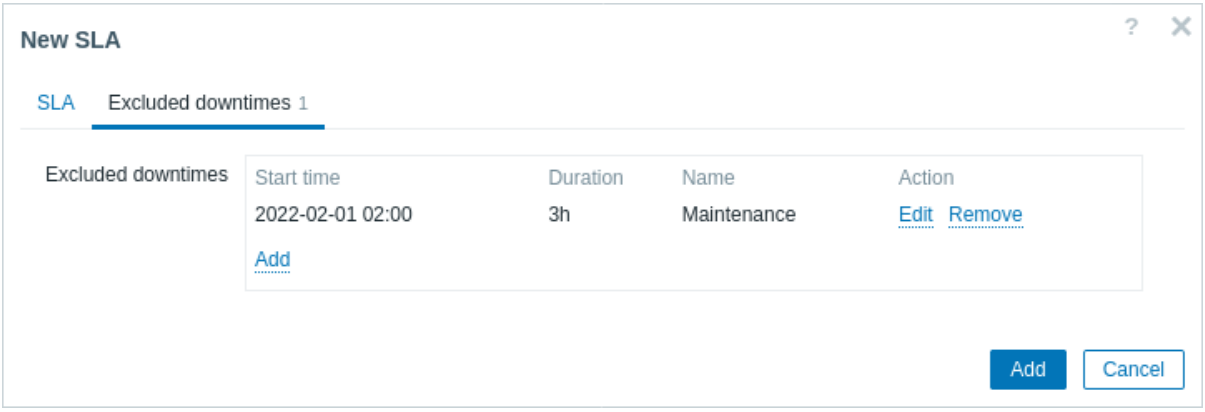
SLA 选项卡允许您指定常规 SLA 参数。



参数	描述
名称	输入 SLA 名称。
SLO	以百分比形式输入服务级别目标（SLO）。
报告周期	选择用于计算 SLI 数据的 SLI 报告周期，详见SLA 报告： 每日 - 每天，从 00:00:00 到 23:59:59。 每周 - 每周，从周日 00:00:00 到周六 23:59:59。 每月 - 每月，从第一天 00:00:00 到最后一天 23:59:59。 每季度 - 每个日历季度：Q1（1 月至 3 月）、Q2（4 月至 6 月）、Q3（7 月至 9 月）、Q4（10 月至 12 月）。 每年 - 每个日历年，从 1 月 1 日 00:00:00 到 12 月 31 日 23:59:59。
时区	选择 SLA 时区。
计划	选择 SLA 计划 - 24x7 或自定义。
生效日期	选择开始计算 SLA 的日期。

参数	描述
服务标签	添加服务标签，以标识应应用此 SLA 的服务。 名称 - 服务标签名称，必须完全匹配且区分大小写。 操作 - 如果标签值必须完全匹配（区分大小写），请选择 等于；如果标签值的一部分必须匹配（不区分大小写），请选择 包含。 值 - 根据所选操作搜索的服务标签值。 如果至少有一个服务标签匹配，则会将 SLA 应用于该服务。
描述	为 SLA 添加描述。
已启用	选中复选框以启用 SLA 计算。

排除的停机时间选项卡允许您指定从 SLA 计算中排除的停机时间。



点击 添加配置排除的停机时间，然后输入周期名称、开始日期和持续时间。

SLA 报告　服务相对于 SLA 的表现可在SLA 报告 中查看。SLA 报告可以通过以下方式查看：

- 在 *SLA* 部分中，点击 SLA 报告超链接；
- 在 *Services* 部分中，点击信息选项卡中的 SLA 名称；
- 在仪表板widget *SLA report* 中。

一旦配置了 SLA，服务部分中的 *Info* 选项卡也会显示一些有关服务性能的信息。

设置示例

概述　该章节向用户展示了如何简单地配置一个服务用来监控 Zabbix 的高可用集群。

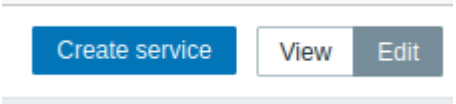
先决条件　在配置服务监控之前，您需要先配置好以下主机：

- *HA* 节点 *1*，至少包含一个触发器和一个标签（最好在触发器级别设置）`component:ha-node-1`
- *HA* 节点 *2*，至少包含一个触发器和一个标签（最好在触发器级别设置）`component:ha-node-2`

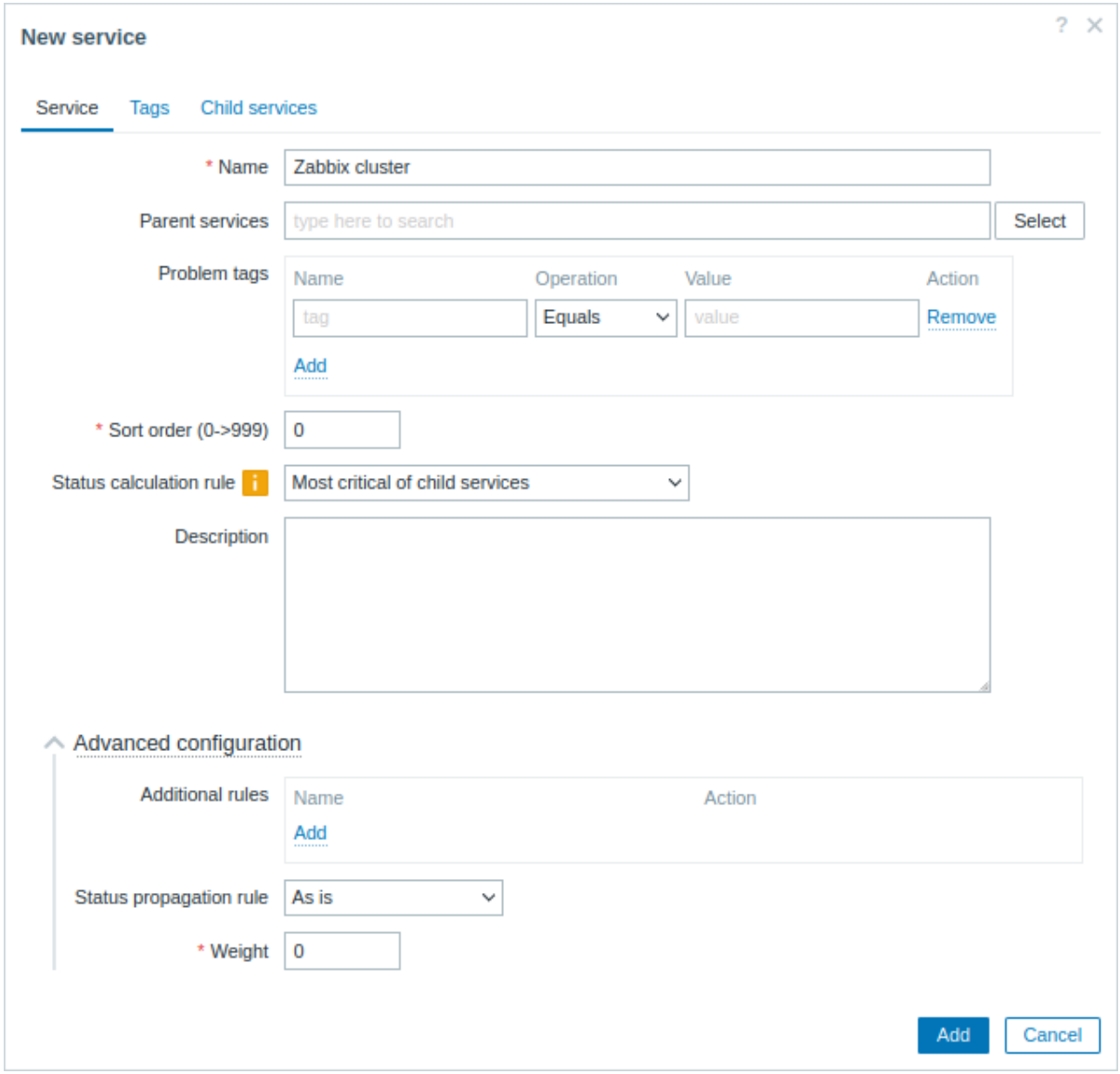
服务树　下一步是构建服务树。在此示例中，基础架构非常简单，由三个服务组成：*Zabbix cluster*（父服务）以及两个子服务 *Zabbix server node 1* 和 *Zabbix server node 2*。

```
Zabbix cluster
|
|- Zabbix server node 1
|- Zabbix server node 2
```

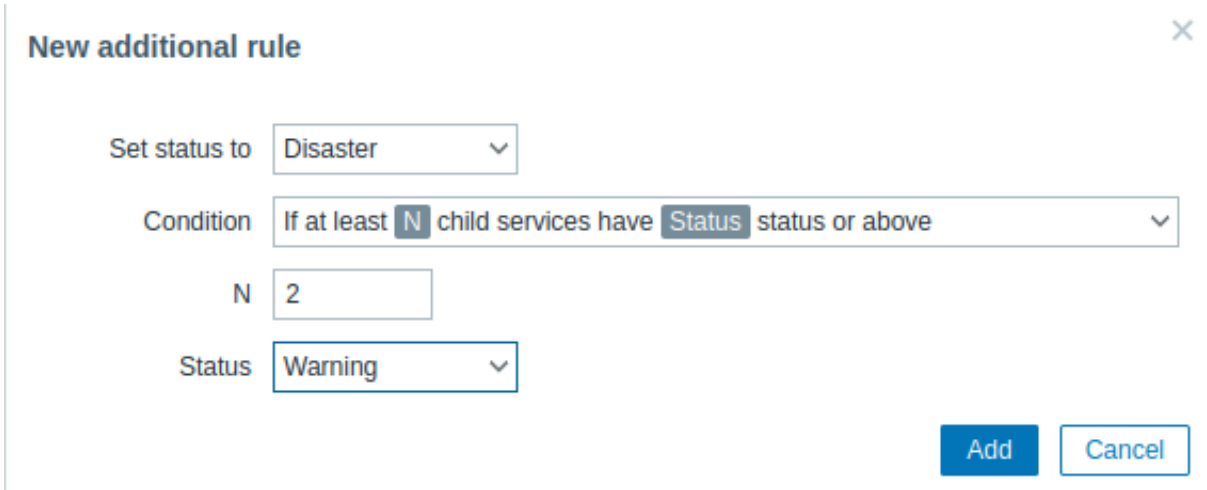
在“服务”页面，开启 *Edit* 模式并点击 *Create service*：



在服务配置窗口中，输入名称 *Zabbix cluster*，然后点击 *Advanced configuration* 标题以显示高级配置选项。



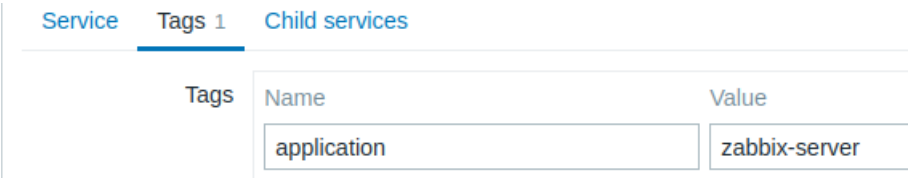
配置附加规则：



Zabbix cluster 将有两个子服务 - 每个 HA 节点对应一个。如果两个 HA 节点都出现至少为 *Warning* 状态的问题，则父服务状态应设置为 *Disaster*。为实现这一点，应将附加规则配置为：

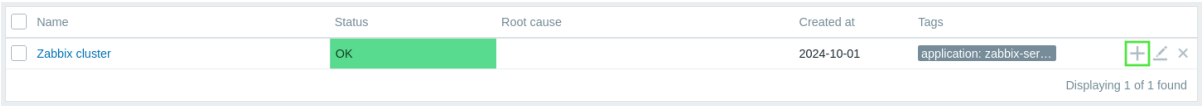
- Set status to: Disaster
- Condition: If at least `N` child services have `Status` status or above
- N: 2
- Status: Warning

切换到 *Tags* 选项卡，并添加标签 `application:zabbix-server`。此标签稍后将用于服务动作和 SLA 报告。



保存新服务。

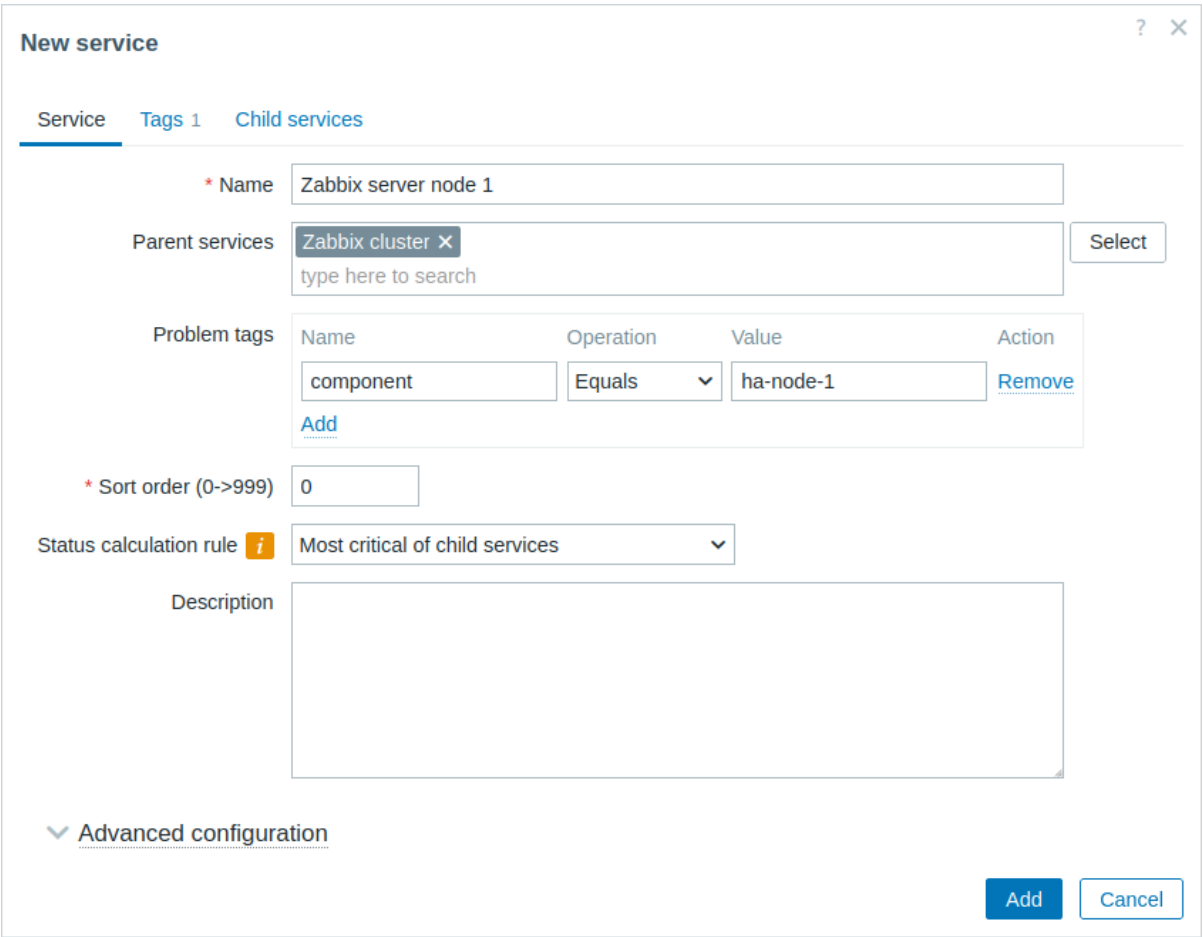
要添加子服务，请单击 Zabbix 集群服务旁边的加号图标（该图标仅在 编辑模式下可见）。



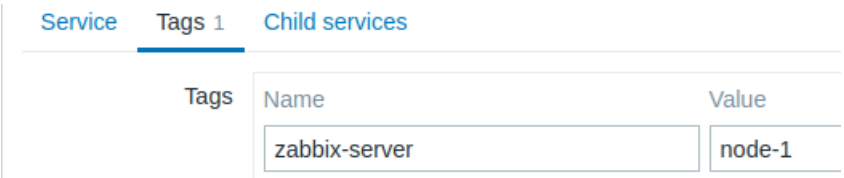
在服务配置窗口中，输入名称 *Zabbix server node 1*。请注意，Parent services 参数已预先填充为 *Zabbix cluster*。

此服务的可用性会受到主机 *HA node 1* 上问题的影响，该问题带有 `component:ha-node-1` 问题标签。在 Problem tags 参数中，输入：

- Name: component
- Operation: Equals
- Value: ha-node-1



切换到 *Tags* 选项卡，并添加一个服务标签：`zabbix-server:node-1`。此标签稍后将用于服务操作和 SLA 报告。



保存新服务。

创建 Zabbix 集群的另一个子服务，名称为 “Zabbix 服务器节点 2”。

将问题标签设置为：

- 名称：component
- 操作：等于
- 值：ha-node-2

切换到 标签选项卡并添加一个服务标签：`zabbix-server:node-2`。

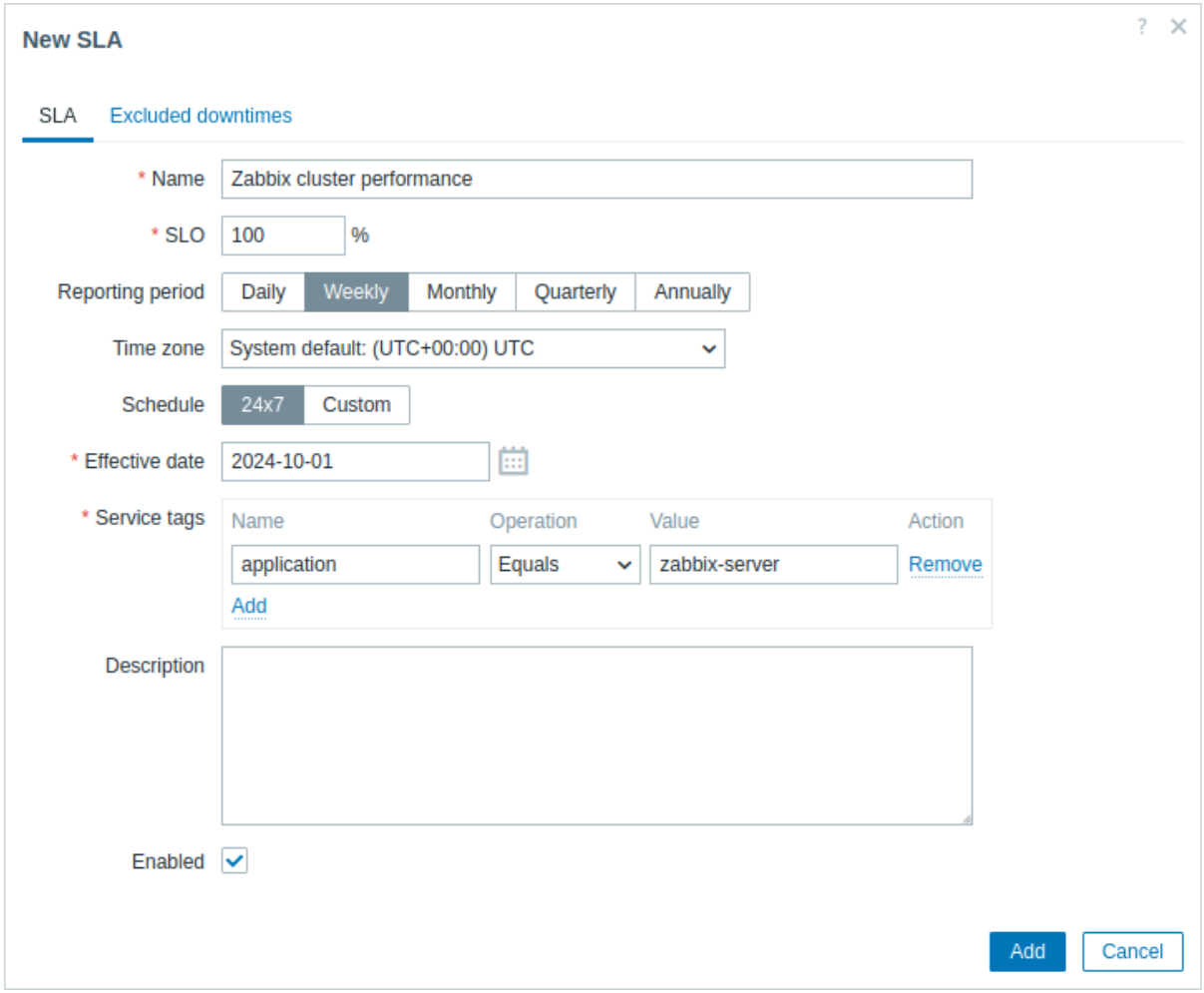
保存新服务。

SLA 在此示例中，预期的 Zabbix 集群性能为 100%，不包括每半年一次、持续一小时的维护时段。

首先，您需要添加新的服务级别协议：

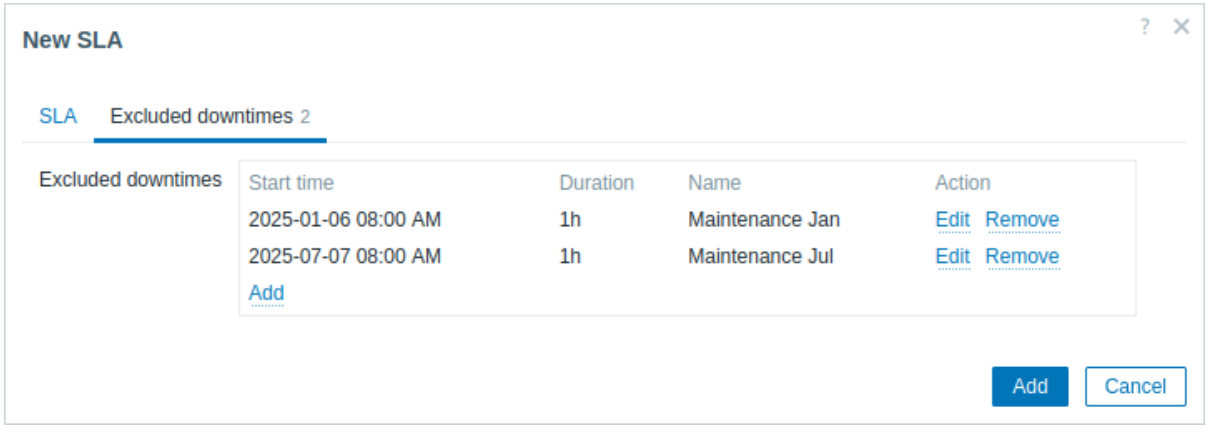
1. 转到 *Services* > *SLA*。
2. 单击 *Create SLA*。
3. 输入名称 *Zabbix cluster performance*，并将 SLO 设置为 100%。服务 Zabbix cluster 的服务标签为 `application:zabbix-server`。
4. 要使用此 SLA 衡量 Zabbix 集群的性能，请在 *Service tags* 参数中指定：

- Name: application
- Operation: Equals
- Value: zabbix-server



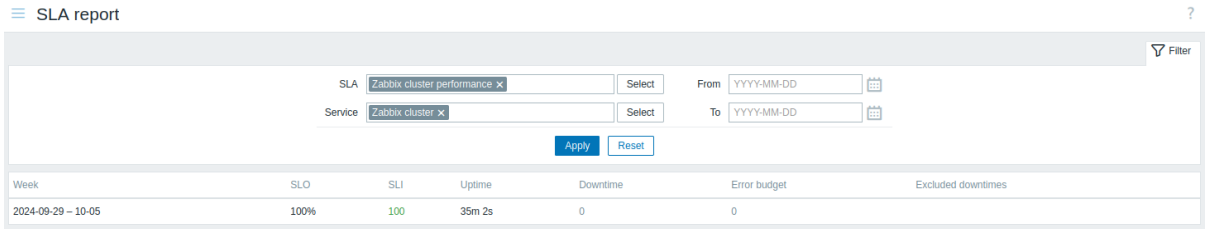
在实际环境中，您还可以更新所需的报告周期、时区和开始日期，或将计划从 24/7 更改为自定义计划。对于此示例，默认设置已足够。

5. 切换到 *Excluded downtimes* 选项卡，并添加计划维护时段的停机时间，以便在 SLA 计算中排除这些时段。
6. 在 *Excluded downtimes* 部分中，单击 *Add* 链接，输入停机时间名称、计划开始时间和持续时间。

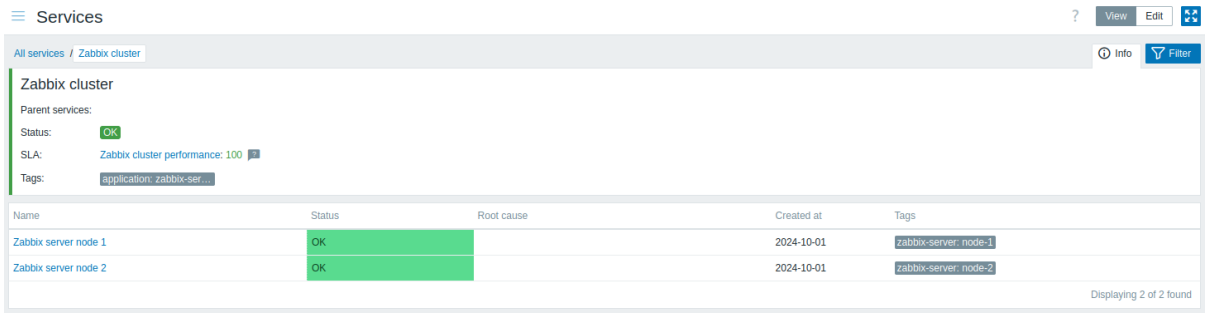


7. 单击 *Add* 以保存新的 SLA。

切换到 SLA 报告部分，查看 Zabbix 集群的 SLA 报告。



您也可以在 *Services* 部分查看 SLA 信息。



服务监控升级

概述　在 Zabbix 6.0 中，服务监控 功能经过了重大重构（有关变更列表，请参见 Zabbix 6.0.0 中的新功能）。

本页说明在升级到 Zabbix 6.0 或更新版本时，早期 Zabbix 版本中定义的服务和 SLA 会如何发生变化。

服务　在较早版本的 Zabbix 中，服务有两种依赖类型：软依赖和硬依赖。升级后，所有依赖将变为等同。

如果某个服务 “Child service” 之前通过硬依赖链接到 “Parent service 1”，并且还通过软依赖链接到 “Parent service 2”，那么升级后，“Child service” 将拥有两个父服务：“Parent service 1” 和 “Parent service 2”。

基于触发器的问题与服务之间的映射已被基于标签的映射所取代。在 Zabbix 6.0 及更新版本中，服务配置表单新增了参数 *Problem tags*，用于指定一个或多个标签名称和值对以进行问题匹配。已链接到某个服务的触发器将获得新的标签 `ServiceLink`：`<trigger ID>:<trigger name>`（标签值将被截断为 32 个字符）。已链接的服务将获得相同值的 `ServiceLink` 问题标签。

状态计算规则

“状态计算算法” 将按以下规则升级：

- 不计算 → 将状态设置为 OK
- 问题，如果至少有一个子项存在问题 → 子服务中最严重的状态
- 问题，如果所有子项都存在问题 → 如果所有子项都存在问题，则取最严重的状态

SLA 此前，必须为每个服务单独定义 SLA 目标。自 Zabbix 6.0 起，SLA 已成为一个独立实体，其中包含服务计划、预期服务级别目标（SLO）以及需要从计算中排除的停机时段信息。配置完成后，可通过服务标签将一个 SLA 分配给多个服务。

在升级期间：

- 为每个服务定义的相同 SLA 将被分组，并且每个组会创建一个 SLA。
- 每个受影响的服务都会获得一个特殊标签 `SLA:<ID>`，并且在相应 SLA 的 *Service tags* 参数中也会指定相同的标签。
- 对于现有服务，SLA 报告中的新指标 “服务创建时间” 将设置为 01/01/2000 00:00。

加密和安全

加密 Zabbix 支持 Zabbix 组件之间使用传输层安全（TLS）协议 v.1.2 和 1.3（取决于加密库）进行加密通信。支持基于证书和基于预共享密钥的加密。

可为以下连接配置加密：

- Zabbix 服务器、Zabbix proxy、Zabbix agent、Zabbix web service、zabbix_sender 和 zabbix_get 工具之间
- 到 Zabbix 数据库来自 Zabbix 前端和服务器/proxy
- Zabbix 前端和 Zabbix 服务器之间

加密是可选的，并且可针对各个组件单独配置：

- 某些 proxy 和 agent 可以配置为与服务器使用基于证书的加密，而其他 proxy 和 agent 可以使用基于预共享密钥的加密，还有一些则继续使用未加密通信（与之前相同）。
- 服务器（proxy）可以针对不同主机使用不同的加密配置。

Zabbix 守护进程程序对加密和未加密的入站连接使用同一个监听端口。启用加密不需要在防火墙上开放新的端口。

限制

- 私钥以明文形式存储在 Zabbix 组件在启动期间可读取的文件中。
- 预共享密钥在 Zabbix 前端中输入，并以明文形式存储在 Zabbix 数据库中。
- 内置加密不会保护运行 Zabbix 前端的 web 服务器与用户 web 浏览器之间的通信。
- 目前，每个加密连接都会通过完整的 TLS 握手建立，不支持会话缓存和票据。
- 启用加密会增加监控项检查和动作所需的时间，具体取决于网络延迟：
  - 例如，如果数据包延迟为 100ms，那么打开 TCP 连接并发送未加密请求大约需要 200ms。使用加密时，建立 TLS 连接大约会额外增加 1000 ms。
  - 可能需要增加超时时间，否则某些在 agent 上运行远程脚本的监控项和动作在未加密连接下可以正常工作，但在加密连接下会因超时而失败。
- 网络发现 不支持加密。网络发现执行的 Zabbix agent 检查将不会加密；如果 Zabbix agent 配置为拒绝未加密连接，这类检查将不会成功。

编译支持加密的 **Zabbix** 要支持加密，Zabbix 必须使用以下受支持的加密库之一进行编译和链接：

- GnuTLS - 3.1.18 及更高版本
- OpenSSL - 1.0.1、1.0.2、1.1.0、1.1.1、3.0.x - 3.5.x 版本
- LibreSSL - 已使用 2.7.4、2.8.2 版本进行测试：
  - 不支持 LibreSSL 2.6.x
  - LibreSSL 作为 OpenSSL 的兼容替代方案受到支持；不会使用新的 `tls_*()` LibreSSL 专用 API 函数。使用 LibreSSL 编译的 Zabbix 组件将无法使用 PSK，只能使用证书。

> **Note:**
> 有关为 Zabbix 前端设置 SSL 的更多信息，请参阅这些最佳实践。

通过向”configure” 脚本指定相应选项来选择加密库：

- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]`（LibreSSL 也使用此选项）

例如，要使用 *OpenSSL* 配置服务器和 agent 的源代码，可以使用类似以下的命令：

```
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --wit
```

不同的 Zabbix 组件可以使用不同的加密库进行编译（例如，服务器使用 *OpenSSL*，agent 使用 *GnuTLS*）。

Attention:
如果计划使用预共享密钥（PSK），请考虑在使用 PSK 的 Zabbix 组件中使用 *GnuTLS* 或 *OpenSSL 1.1.0*（或更高版本）库。*GnuTLS* 和 *OpenSSL 1.1.0* 库支持具有完全前向保密功能的 PSK 密码套件。较旧版本的 *OpenSSL* 库（1.0.1、1.0.2c）也支持 PSK，但可用的 PSK 密码套件不提供完全前向保密功能。

连接加密管理　Zabbix 中的连接可以使用：

- 不加密（默认）
- 基于 RSA 证书的加密
- 基于 PSK 的加密

用于指定 Zabbix 组件之间加密方式的两个重要参数：

- TLSConnect - 指定传出连接使用的加密方式（未加密、PSK 或证书）
- TLSAccept - 指定传入连接允许的连接类型（未加密、PSK 或证书）。可以指定一个或多个值。

`TLSConnect` 用于 Zabbix proxy 的配置文件（在主动模式下，仅指定与服务器的连接）以及 Zabbix agent 的配置文件（用于主动检查）。在 Zabbix 前端中，`TLSConnect` 对应于 数据采集 > 主机 > < 某个主机 > > 加密选项卡中的 连接到主机字段，以及 管理 > *Proxy* > < 某个 proxy> > 加密选项卡中的 连接到 *proxy* 字段。如果配置的连接加密类型失败，则不会尝试其他加密类型。

`TLSAccept` 用于 Zabbix proxy 的配置文件（在被动模式下，仅指定来自服务器的连接）以及 Zabbix agent 的配置文件（用于被动检查）。在 Zabbix 前端中，TLSAccept 对应于 数据采集 > 主机 > < 某个主机 > > 加密选项卡中的 来自主机的连接字段，以及 管理 > *Proxy* > < 某个 proxy> > 加密选项卡中的 来自 *proxy* 的连接字段。

通常，您只需为传入连接配置一种加密类型。但您可能希望切换加密类型，例如，在停机时间最短且可以回滚的情况下，从未加密切换到基于证书的加密。为此：

- 在 agent 配置文件中设置 `TLSAccept=unencrypted,cert`，然后重启 Zabbix agent。
- 使用带证书的 `zabbix_get` 测试与 agent 的连接。如果连接成功，您可以在 Zabbix 前端的 数据采集 > 主机 > < 某个主机 > > 加密选项卡中重新配置该 agent 的加密，将 连接到主机设置为 证书。
- 当服务器配置缓存更新后（如果该主机由 proxy 监控，则 proxy 配置也会更新），与该 agent 的连接将被加密。
- 如果一切按预期运行，您可以在 agent 配置文件中设置 `TLSAccept=cert`，然后重启 Zabbix agent。此时，agent 将只接受基于加密证书的连接。未加密连接和基于 PSK 的连接将被拒绝。

服务器和 proxy 的工作方式类似。如果在 Zabbix 前端的主机配置中将 来自主机的连接设置为 “证书”，则来自 agent（主动检查）和 zabbix_sender（trapper 监控项）的连接将只接受基于证书的加密连接。

通常，您会将传入和传出连接配置为使用相同的加密类型，或者完全不加密。但从技术上讲，也可以进行非对称配置，例如，对传入连接使用基于证书的加密，对传出连接使用基于 PSK 的加密。

每个主机的加密配置显示在 Zabbix 前端的 数据采集 > 主机中，位于 *Agent* 加密列。例如：

示例	连接到主机	允许来自主机的连接	拒绝来自主机的连接
NONE	未加密	未加密	加密、基于证书的加密和基于 PSK 的加密
CERT NONE PSK CERT	加密、基于证书	加密、基于证书	未加密和基于 PSK 的加密
PSK NONE PSK CERT	加密、基于 PSK	加密、基于 PSK	未加密和基于证书的加密
PSK NONE PSK CERT	加密、基于 PSK	未加密和基于 PSK 的加密	基于证书的加密
CERT NONE PSK CERT	加密、基于证书	未加密、基于 PSK 或基于证书的加密	-

Attention:
连接默认不加密。必须分别为每个主机和 proxy 配置加密。

zabbix_get 和 zabbix_sender 的加密使用　请参阅zabbix_get 和zabbix_sender 的手册页，了解如何在加密环境中使用它们。

密码套件　默认情况下，密码套件在 Zabbix 启动期间在内部进行配置。

GnuTLS 和 OpenSSL 也支持用户配置的密码套件。用户可以根据自己的安全策略配置密码套件。使用此功能是可选的（内置默认密码套件仍然可用）。

对于使用默认设置编译的加密库，Zabbix 内置规则通常会生成以下密码套件（按优先级从高到低排序）：

Library	Certificate ciphersuites	PSK ciphersuites
GnuTLS 3.1.18	TLS_ECDHE_RSA_AES_128_GCM_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA1 TLS_RSA_AES_128_GCM_SHA256 TLS_RSA_AES_128_CBC_SHA256 TLS_RSA_AES_128_CBC_SHA1	TLS_ECDHE_PSK_AES_128_CBC_SHA256 TLS_ECDHE_PSK_AES_128_CBC_SHA1 TLS_PSK_AES_128_GCM_SHA256 TLS_PSK_AES_128_CBC_SHA256 TLS_PSK_AES_128_CBC_SHA1
OpenSSL 1.0.2c	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-SHA256 AES128-SHA	PSK-AES128-CBC-SHA
OpenSSL 1.1.0	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA
OpenSSL 1.1.1d	TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA

用户配置的 **ciphersuite**　内置的 ciphersuite 选择条件可以通过用户配置的 ciphersuite 覆盖。

> **Attention:**
> 用户配置的 ciphersuite 是一项面向高级用户的功能，这些用户了解 TLS ciphersuite、其安全性以及错误可能带来的后果，并且熟悉 TLS 故障排查。

可以使用以下参数覆盖内置的 ciphersuite 选择条件：

覆盖范围	参数	值	说明
证书的 ciphersuite 选择	TLSCipherCert13	适用于 TLS 1.3 协议的有效 OpenSSL 1.1.1 cipher strings（其值会传递给 OpenSSL 函数 `SSL_CTX_set_ciphersuites()`）。	基于证书的 TLS 1.3 ciphersuite 选择条件 仅适用于 OpenSSL 1.1.1 或更新版本。
	TLSCipherCert	适用于 TLS 1.2 的有效 OpenSSL cipher strings 或有效的 GnuTLS priority strings。其值分别会传递给 `SSL_CTX_set_cipher_list()` 或 `gnutls_priority_init()` 函数。	基于证书的 TLS 1.2/1.3（GnuTLS）、TLS 1.2（OpenSSL）ciphersuite 选择条件
PSK 的 ciphersuite 选择	TLSCipherPSK13	适用于 TLS 1.3 协议的有效 OpenSSL 1.1.1 cipher strings（其值会传递给 OpenSSL 函数 `SSL_CTX_set_ciphersuites()`）。	基于 PSK 的 TLS 1.3 ciphersuite 选择条件 仅适用于 OpenSSL 1.1.1 或更新版本。
	TLSCipherPSK	适用于 TLS 1.2 的有效 OpenSSL cipher strings 或有效的 GnuTLS priority strings。其值分别会传递给 `SSL_CTX_set_cipher_list()` 或 `gnutls_priority_init()` 函数。	基于 PSK 的 TLS 1.2/1.3（GnuTLS）、TLS 1.2（OpenSSL）ciphersuite 选择条件

覆盖范围	参数	值	说明
证书和 PSK 的组合 ciphersuite 列表	TLSCipherAll13	适用于 TLS 1.3 协议的有效 OpenSSL 1.1.1 cipher strings（其值会传递给 OpenSSL 函数 `SSL_CTX_set_ciphersuites()`）。	TLS 1.3 的 ciphersuite 选择条件 仅适用于 OpenSSL 1.1.1 或更新版本。
	TLSCipherAll	适用于 TLS 1.2 的有效 OpenSSL cipher strings 或有效的 GnuTLS priority strings。其值分别会传递给 `SSL_CTX_set_cipher_list()` 或 `gnutls_priority_init()` 函数。	TLS 1.2/1.3（GnuTLS）、TLS 1.2（OpenSSL）的 ciphersuite 选择条件

要在zabbix_get 和zabbix_sender 工具中覆盖 ciphersuite 选择，请使用以下命令行参数：

- `--tls-cipher13`
- `--tls-cipher`

这些新参数是可选的。如果未指定某个参数，则使用内部默认值。如果定义了某个参数，则不能为空。

如果在加密库中设置某个 TLSCipher* 值失败，则服务器、proxy 或 agent 将不会启动，并且会记录错误。

重要的是要理解每个参数适用的场景。

出站连接

最简单的情况是出站连接：

- 对于使用证书的出站连接 - 使用 TLSCipherCert13 或 TLSCipherCert
- 对于使用 PSK 的出站连接 - 使用 TLSCipherPSK13 或 TLSCipherPSK
- 对于 zabbix_get 和 zabbix_sender 工具，可以使用命令行参数 `--tls-cipher13` 或 `--tls-cipher`（加密方式通过 `--tls-connect` 参数明确指定）

入站连接

对于入站连接，情况会稍微复杂一些，因为规则会因组件和配置而异。

对于 Zabbix **agent**：

Agent connection setup	Cipher configuration
TLSConnect=cert	TLSCipherCert, TLSCipherCert13
TLSConnect=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert	TLSCipherCert, TLSCipherCert13
TLSAccept=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert,psk	TLSCipherAll, TLSCipherAll13

对于 Zabbix 服务器和 **proxy**：

Connection setup	Cipher configuration
Outgoing connections using PSK	TLSCipherPSK, TLSCipherPSK13
Incoming connections using certificates	TLSCipherAll, TLSCipherAll13
Incoming connections using PSK if server has no certificate	TLSCipherPSK, TLSCipherPSK13
Incoming connections using PSK if server has certificate	TLSCipherAll, TLSCipherAll13

从上面的两个表中可以看出一些规律：

- 只有在使用证书和 PSK 基于的密码套件组合列表时，才可以指定 TLSCipherAll 和 TLSCipherAll13。出现这种情况有两种：服务器（proxy）已配置证书（如果加密库支持 PSK，则服务器和 proxy 始终会配置 PSK 密码套件；proxy 仅在加密库支持 PSK 时才会配置），agent 配置为同时接受基于证书和基于 PSK 的入站连接
- 在其他情况下，TLSCipherCert* 和/或 TLSCipherPSK* 就足够了

下表显示了 `TLSCipher*` 内置默认值。它们可以作为你自定义值的良好起点。

Parameter	GnuTLS 3.6.12
TLSCipherCert	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509
TLSCipherPSK	NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL
TLSCipherAll	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509

Parameter	OpenSSL 1.1.1d [1]
TLSCipherCert13	
TLSCipherCert	EECDH+aRSA+AES128:RSA+aRSA+AES128
TLSCipherPSK13	TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
TLSCipherPSK	kECDHEPSK+AES128:kPSK+AES128
TLSCipherAll13	
TLSCipherAll	EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128

[1] 默认值在较旧的 OpenSSL 版本（1.0.1、1.0.2、1.1.0）、LibreSSL，以及在编译 OpenSSL 时未启用 PSK 支持的情况下会有所不同。

用户配置的密码套件示例

请参见下面的用户配置密码套件示例：

- 测试密码字符串并仅允许 PFS 密码套件
- 从 AES128 切换到 AES256

测试密码套件字符串并仅允许 PFS 密码套件

要查看已选择的密码套件，需要在配置文件中设置 DebugLevel=4，或者为 zabbix_sender 使用 -vv 选项。

在获得所需的密码套件之前，可能需要对 TLSCipher* 参数进行一些尝试。仅为了调整 TLSCipher* 参数而多次重启 Zabbix 服务器、proxy 或 agent 很不方便。更方便的选项是使用 zabbix_sender 或 openssl 命令。下面分别进行说明。

1. 使用 zabbix_sender。

创建一个测试配置文件，例如 /home/zabbix/test.conf，其语法与 zabbix_agentd.conf 文件相同：

```
Hostname=nonexisting
ServerActive=nonexisting

TLSConnect=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agent.psk
```

本示例需要有效的 CA 和 agent 证书以及 PSK。请根据您的环境调整证书和 PSK 文件的路径及名称。

如果不使用证书，而只使用 PSK，可以创建一个更简单的测试文件：

```
Hostname=nonexisting
ServerActive=nonexisting

TLSConnect=psk
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agentd.psk
```

运行 zabbix_sender 即可查看已选择的密码套件（以下示例使用 OpenSSL 1.1.d 编译）：

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

这里显示的是默认选择的密码套件。这些默认值旨在确保与运行较旧 OpenSSL 版本（1.0.1 及更高版本）的 Zabbix agent 之间的互操作性。

在较新的系统上，可以通过仅允许少量密码套件来提高安全性，例如仅允许支持 PFS（Perfect Forward Secrecy，完全前向保密）的密码套件。下面配置 TLSCipher* 参数，使其仅允许支持 PFS 的密码套件。

> **Attention:**
> 如果使用 PSK，结果将无法与使用 OpenSSL 1.0.1 和 1.0.2 的系统互操作。基于证书的加密应该可以正常工作。

向 test.conf 配置文件中添加两行：

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
```

然后再次进行测试：

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

“certificate ciphersuites” 和 “PSK ciphersuites” 列表已发生变化——它们比之前更短，仅包含 TLS 1.3 密码套件和 TLS 1.2 ECDHE-* 密码套件，符合预期。

2. 无法使用 zabbix_sender 测试 TLSCipherAll 和 TLSCipherAll13；它们不会影响上例中显示的 “certificate and PSK ciphersuites” 值。要调整 TLSCipherAll 和 TLSCipherAll13，需要使用 agent、proxy 或服务器进行测试。

因此，如果要仅允许 PFS 密码套件，并且每个组件都配置了证书，同时 agent 还配置了 PSK，则可能需要将最多三个参数添加到 zabbix_agentd.conf、zabbix_proxy.conf 和 zabbix_server.conf：

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
TLSCipherAll=EECDH+aRSA+AES128:kECDHEPSK+AES128
```

如果您的 Zabbix 环境仅使用基于 PSK 的加密而不使用证书，则只需配置一个参数：

```
TLSCipherPSK=kECDHEPSK+AES128
```

现在您已经了解其工作方式，可以使用 openssl 命令在 Zabbix 之外测试密码套件选择。下面测试全部三个 TLSCipher* 参数值：

```
$ openssl ciphers EECDH+aRSA+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
$ openssl ciphers kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 E
$ openssl ciphers EECDH+aRSA+AES128:kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
```

您也可以使用带有 -V 选项的 openssl ciphers，以获得更详细的输出：

```
$ openssl ciphers -V EECDH+aRSA+AES128:kECDHEPSK+AES128
           0x13,0x02 - TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
           0x13,0x03 - TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256
           0x13,0x01 - TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
           0xC0,0x2F - ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(128) Mac=AEAD
           0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA256
           0xC0,0x13 - ECDHE-RSA-AES128-SHA    TLSv1 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA1
           0xC0,0x37 - ECDHE-PSK-AES128-CBC-SHA256 TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA256
           0xC0,0x35 - ECDHE-PSK-AES128-CBC-SHA TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA1
```

同样，也可以测试 GnuTLS 的优先级字符串：

```
$ gnutls-cli -l --priority=NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-A
Cipher suites for NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-ALL:+COMP-
TLS_ECDHE_RSA_AES_128_GCM_SHA256                0xc0, 0x2f      TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA256                0xc0, 0x27      TLS1.2

Protocols: VERS-TLS1.2
Ciphers: AES-128-GCM, AES-128-CBC
MACs: AEAD, SHA256
Key Exchange Algorithms: ECDHE-RSA
Groups: GROUP-SECP256R1, GROUP-SECP384R1, GROUP-SECP521R1, GROUP-X25519, GROUP-X448, GROUP-FFDHE2048, GR
PK-signatures: SIGN-RSA-SHA256, SIGN-RSA-PSS-SHA256, SIGN-RSA-PSS-RSAE-SHA256, SIGN-ECDSA-SHA256, SIGN-E
```

从 AES128 切换到 AES256

Zabbix 将 AES128 作为数据的内置默认值。假设您正在使用证书，并希望在 OpenSSL 1.1.1 上切换到 AES256。

这可以通过在 zabbix_server.conf 中添加相应参数来实现：

```
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/server.crt
TLSKeyFile=/home/zabbix/server.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
TLSCipherPSK13=TLS_CHACHA20_POLY1305_SHA256
TLSCipherPSK=kECDHEPSK+AES256:-SHA1
TLSCipherAll13=TLS_AES_256_GCM_SHA384
TLSCipherAll=EECDH+aRSA+AES256:-SHA1:-SHA384
```

Attention:
尽管实际只会使用与证书相关的密码套件，但也定义了 TLSCipherPSK* 参数，以避免使用其默认值，因为默认值包含安全性较低的密码套件，以便实现更广泛的互操作性。PSK 密码套件无法在服务器/proxy 上完全禁用。

以及在 zabbix_agentd.conf 中：

```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
```

加密问题排查　一般建议：

- 首先要弄清在问题场景中，哪个组件充当 TLS 客户端，哪个组件充当 TLS 服务器。
 Zabbix 服务器、proxy 和 agent 根据它们之间的交互方式，都可以充当 TLS 服务器和客户端。
 例如，Zabbix 服务器连接 agent 进行被动检查时，充当 TLS 客户端。agent 则充当 TLS 服务器。
 Zabbix agent 从 proxy 请求主动检查列表时，充当 TLS 客户端。proxy 则充当 TLS 服务器。
 zabbix_get 和 zabbix_sender 工具始终充当 TLS 客户端。
- Zabbix 使用双向认证。
 双方都会验证对端，并且可能拒绝连接。
 例如，Zabbix 服务器连接 agent 时，如果 agent 的证书无效，连接可能会立即关闭。反之亦然 -
 Zabbix agent 接受来自服务器的连接时，如果服务器不被 agent 信任，也可能会关闭连接。
- 检查 TLS 客户端和 TLS 服务器两端的日志文件。
 拒绝连接的一方通常会记录被拒绝的准确原因。另一方往往只会报告比较笼统的错误（例如，“Connection closed by peer”、“connection was non-properly terminated”）。
- 有时，配置错误的加密会导致令人困惑的错误消息，完全无法指向真正的原因。
 在下面的小节中，我们尝试提供一份（远非穷尽的）消息及其可能原因的集合，以帮助排查问题。
 请注意，不同的加密工具包（OpenSSL、GnuTLS）在相同问题场景下通常会产生不同的错误消息。
 有时，错误消息甚至取决于双方所使用的加密工具包的具体组合。

连接类型或权限问题排查　服务器已配置为使用 PSK 连接到 agent，但 agent 仅接受未加密连接

在服务器或 proxy 日志中（使用 *GnuTLS* 3.3.16）

```
从 agent 获取值失败：zbx_tls_connect(): gnutls_handshake() 失败： \
    -110 TLS 连接未正确终止。
```

在服务器或 proxy 日志中（使用 *OpenSSL* 1.0.2c）

```
从 agent 获取值失败：TCP 连接成功，无法在 [[127.0.0.1]:10050] 上建立 TLS： \
    连接被对端关闭。请检查允许的连接类型和访问权限
```

一侧使用证书连接，但另一侧只接受 PSK，或反之

在任何日志中（使用 *GnuTLS*）：

```
failed to accept an incoming connection: from 127.0.0.1: zbx_tls_accept(): gnutls_handshake() failed:\
    -21 Could not negotiate a supported cipher suite.
```

在任何日志中（使用 *OpenSSL* 1.0.2c）：

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake returned error code 1:\
    file .\ssl\s3_srvr.c line 1411: error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher:\
    TLS write fatal alert "handshake failure"
```

尝试使用启用了 **TLS** 支持编译的 **Zabbix sender** 向未启用 **TLS** 支持编译的 **Zabbix** 服务器/**proxy** 发送数据　在连接端日志中：

Linux：

```
...In zbx_tls_init_child()
...OpenSSL library (version OpenSSL 1.1.1  11 Sep 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...End of zbx_tls_connect():FAIL error:'connection closed by peer'
...send value error: TCP successful, cannot establish TLS to [[localhost]:10051]: connection closed by pee
```

Windows：

```
...OpenSSL library (version OpenSSL 1.1.1a  20 Nov 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...zbx_psk_client_cb() requested PSK identity "PSK test sender"
...End of zbx_tls_connect():FAIL error:'SSL_connect() I/O error: [0x00000000] The operation completed succ
...send value error: TCP successful, cannot establish TLS to [[192.168.1.2]:10051]: SSL_connect() I/O erro
```

在接收端日志中：

```
...failed to accept an incoming connection: from 127.0.0.1: support for TLS was not compiled in
```

一侧使用 **PSK** 连接，但另一侧使用 **LibreSSL** 或在编译时未启用加密支持　LibreSSL 不支持 PSK。

在连接端日志中：

```
...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() I/O error: [0] Success
```

在接受端日志中：

```
...failed to accept an incoming connection: from 192.168.1.2: support for PSK was not compiled in
```

在 Zabbix 前端中：

```
Get value from agent failed: TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect()
```

一侧使用 **PSK** 连接，但另一侧使用已禁用 **PSK** 支持的 **OpenSSL**　在连接方日志中：

```
...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() set result code to SSL_ERR
```

在接受方日志中：

```
...failed to accept an incoming connection: from 192.168.1.2: TLS handshake set result code to 1: file ssl
```

使用证书进行加密

概述　Zabbix 可以使用 PEM 格式的 RSA 证书，这些证书由公共证书颁发机构（CA）或内部证书颁发机构（CA）签名。

证书验证是基于预先配置的 CA 证书进行的。此外，还可以使用证书吊销列表（CRL）。

每个 Zabbix 组件只能配置一个证书。

有关设置和运行内部 CA、生成并签署证书请求以及吊销证书的更多信息，请参考诸如 OpenSSL PKI Tutorial v2.0 之类的教程。

请仔细考虑并测试您的证书扩展。有关更多详细信息，请参见使用 X.509 v3 证书扩展的限制。

证书配置参数　以下配置参数支持用于在 Zabbix 组件上设置证书。

Parameter	Mandatory	Description
TLSCAFile	yes	包含用于对等证书验证的顶级 CA 证书的文件完整路径名。 如果使用包含多个成员的证书链，请将较低级别的 CA 证书排在前面，然后是较高级别的 CA 证书。 来自多个 CA 的证书可以包含在同一个文件中。
TLSCRLFile	no	包含证书吊销列表（CRL）的文件完整路径名。
TLSCertFile	yes	包含证书的文件完整路径名。 如果使用包含多个成员的证书链，请将 server、proxy 或 agent 证书排在前面，然后是较低级别的 CA 证书，最后是较高级别的 CA 证书。

Parameter	Mandatory	Description
TLSKeyFile	yes	包含私钥的文件完整路径名。 请通过设置适当的访问权限，确保只有Zabbix 用户 可以读取此文件。
TLSServerCertIssuer	no	允许的 server 证书颁发者。
TLSServerCertSubject	no	允许的 server 证书主题。

配置示例　在设置好必要的证书后，配置 Zabbix 组件使用基于证书的加密。

以下是配置的详细步骤：

- Zabbix 服务器
- Zabbix proxy
- Zabbix agent

Zabbix 服务器

1. 准备 CA 证书文件。

为了验证对等方证书，Zabbix 服务器必须能够访问包含顶级自签名根 CA 证书的文件。例如，如果需要使用来自两个独立根 CA 的证书，请将它们放入 `/home/zabbix/zabbix_ca_file.crt` 文件中：

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
            ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            ...
-----BEGIN CERTIFICATE-----
MIID2jCCAsKgAwIBAgIBATANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
....
9wEzdN8uTrqoyU78gi12npLj08LegRKjb5hFTVmO
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
            ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ....
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            ....
-----BEGIN CERTIFICATE-----
MIID3DCCAsSgAwIBAgIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQB
...
```

```
vdGNYoSfvu41GQAR5Vj5FnRJRzv5XQOZ3B6894GY1zY=
-----END CERTIFICATE-----
```

2. 将 Zabbix 服务器证书/证书链放入文件中，例如 /home/zabbix/zabbix_server.crt。第一个证书是 Zabbix 服务器证书，后面是中间 CA 证书：

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix server
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints:
                CA:FALSE
            ...
-----BEGIN CERTIFICATE-----
MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixk
...
h02u1GHiy46GI+xfR3LsPwFKlkTaaLaL/6aaoQ==
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
        ...
-----BEGIN CERTIFICATE-----
MIID4TCCAsmgAwIBAgIBAjANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
...
dyCeWnvL7u5sd6ffo8iRny0QzbHKmQt/wUtcVIvWXdMIFJMOHw==
-----END CERTIFICATE-----
```

Note:
对于客户端和服务器证书，请仅使用上述提到的属性，以避免影响证书验证过程。例如，如果使用 *X509v3 Subject Alternative Name* 或 *Netscape Cert Type* 扩展，OpenSSL 可能无法建立加密连接。有关更多信息，请参阅使用 X.509 v3 证书扩展的限制。

3. 将 Zabbix 服务器私钥放入文件中，例如 /home/zabbix/zabbix_server.key：

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXDl
...
IJLkhbybBYEf47MLhffWa7XvZTY=
-----END PRIVATE KEY-----
```

4. 编辑 Zabbix 服务器配置文件中的 TLS 配置参数：

```
TLSCAFile=/home/zabbix/zabbix_ca_file.crt
TLSCertFile=/home/zabbix/zabbix_server.crt
TLSKeyFile=/home/zabbix/zabbix_server.key
```

Zabbix proxy

1. 按照Zabbix 服务器 部分中的说明，准备包含顶级 CA 证书、Zabbix proxy 证书/证书链以及私钥的文件。然后，在 Zabbix proxy 配置文件中相应地编辑 `TLSCAFile`、`TLSCertFile` 和 `TLSKeyFile` 参数。

2. 在 Zabbix proxy 配置文件中编辑其他 TLS 参数：

- 对于主动 proxy：`TLSConnect=cert`
- 对于被动 proxy：`TLSAccept=cert`

> **Note:**
> 为了提高 proxy 的安全性，还可以设置 `TLSServerCertIssuer` 和 `TLSServerCertSubject` 参数。有关更多信息，请参阅限制允许的证书颁发者和使用者。

最终 proxy 配置文件中的 TLS 参数可能如下所示：

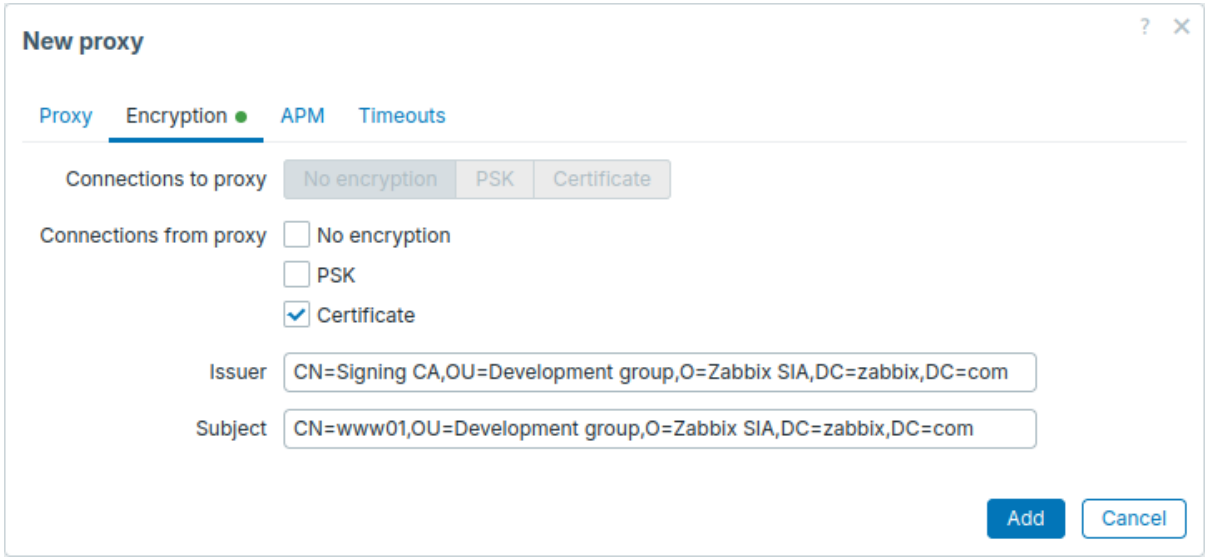
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file.crt
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_proxy.crt
TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

3. 在 Zabbix 前端中为此 proxy 配置加密：

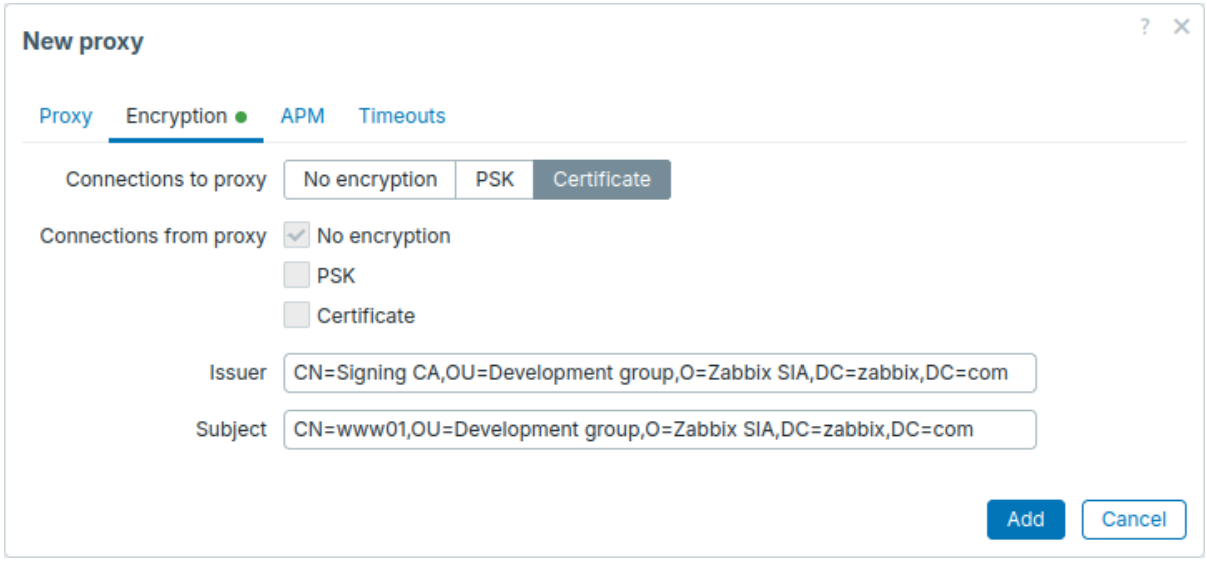
- 转到：*Administration* > *Proxies*。
- 选择该 proxy，然后单击 *Encryption* 选项卡。

在下面的示例中，*Issuer* 和 *Subject* 字段已填写。有关为何以及如何使用这些字段的更多信息，请参阅限制允许的证书颁发者和使用者。

对于主动 proxy：



对于被动 proxy：



Zabbix agent

1. 按照Zabbix 服务器 部分中的说明，准备包含顶级 CA 证书、Zabbix agent 证书/证书链以及私钥的文件。然后，在 Zabbix agent 配置文件中相应地编辑 `TLSCAFile`、`TLSCertFile` 和 `TLSKeyFile` 参数。

2. 在 Zabbix agent 配置文件中编辑其他 TLS 参数：

- 对于主动式 agent：`TLSConnect=cert`
- 对于被动式 agent：`TLSAccept=cert`

Note:
要提高 agent 的安全性，可以设置 `TLSServerCertIssuer` 和 `TLSServerCertSubject` 参数。有关更多信息，请参阅限制允许的证书颁发者和主题。

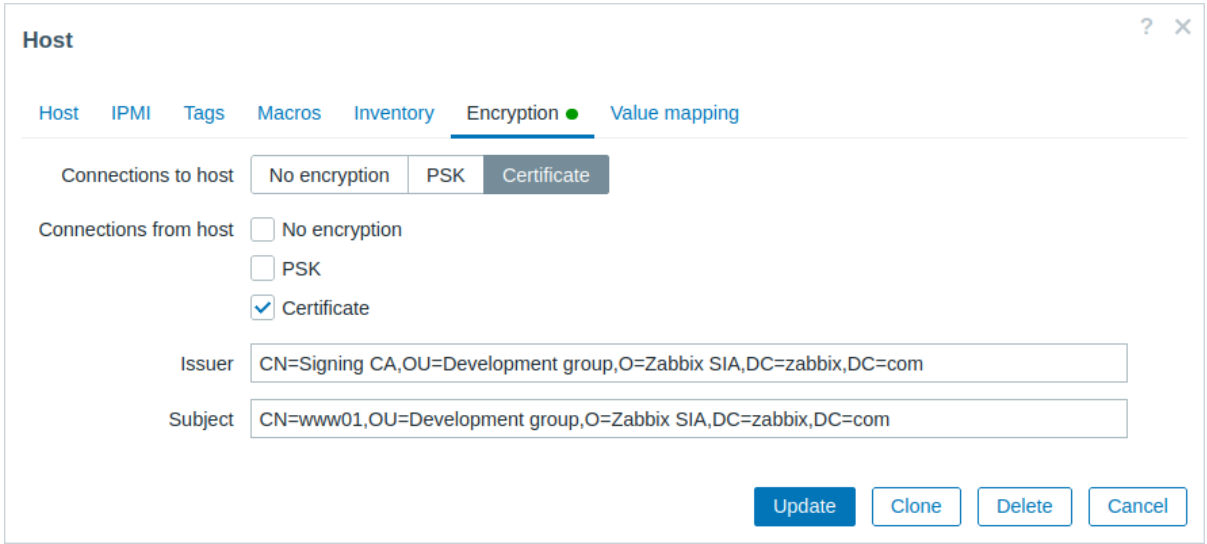
最终的 agent 配置文件中的 TLS 参数可能如下所示。请注意，该示例假设主机由 proxy 监控，因此将 proxy 指定为证书的 Subject：

```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file.crt
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_agentd.crt
TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

3. 在 Zabbix 前端中为由此 agent 监控的主机配置加密。

- 转到：数据采集 > 主机。
- 选择主机，然后单击加密选项卡。

在下面的示例中，已填写 *Issuer* 和 *Subject* 字段。有关使用这些字段的原因和方法的更多信息，请参阅限制允许的证书颁发者和主题。



Zabbix web service

1. 按照Zabbix 服务器部分中的说明，准备包含顶级 CA 证书、Zabbix web service 证书/证书链以及私钥的文件。然后，在 Zabbix web service 配置文件中相应地编辑 TLSCAFile、TLSCertFile 和 TLSKeyFile 参数。

2. 在 Zabbix web service 配置文件中编辑其他 TLS 参数：TLSAccept=cert

最终的 web service 配置文件中的 TLS 参数可能如下所示：

```
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file.crt
TLSCertFile=/home/zabbix/zabbix_web_service.crt
TLSKeyFile=/home/zabbix/zabbix_web_service.key
```

3. 编辑 Zabbix 服务器配置文件中的 WebServiceURL 参数，将 Zabbix 服务器配置为连接到已配置 TLS 的 Zabbix web service：

```
WebServiceURL=https://example.com:443/report
```

限制允许的证书颁发者和使用者　当两个 Zabbix 组件（例如服务器和 agent）建立 TLS 连接时，它们会验证对方的证书。如果对等方证书由受信任的 CA 签名（TLSCAFile 中预配置了顶级证书），证书有效且未过期，并通过其他检查，则组件之间可以继续通信。在这种最简单的情况下，不会验证证书颁发者和使用者。

但是，这会带来风险：任何拥有有效证书的人都可以冒充其他人（例如，可以使用主机证书冒充服务器）。在证书由专用的内部 CA 签名且冒充风险较低的小型环境中，这种方式可能是可以接受的；但在较大或安全性要求更高的环境中，这可能并不足够。

如果您的顶级 CA 会签发不应被 Zabbix 接受的证书，或者您希望降低冒充风险，则可以通过指定证书的颁发者和使用者来限制允许的证书。

例如，在 Zabbix proxy 配置文件中，可以指定：

```
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

使用这些设置时，主动 proxy 不会与证书具有不同颁发者或使用者的 Zabbix 服务器通信。同样，被动 proxy 不会接受来自此类服务器的请求。

Issuer 和 Subject 字符串的匹配规则

Issuer 和 Subject 字符串的匹配规则如下：

- Issuer 和 Subject 字符串分别独立检查。两者都是可选的。
- 未指定的字符串表示接受任意字符串。
- 字符串按原样比较，且必须完全匹配。
- 支持 UTF-8 字符。但是，不支持通配符（*）或正则表达式。
- 已实现以下 RFC 4514 要求 - 需要转义的字符（使用 \ 反斜杠，U+005C）：
  - 在字符串中的任意位置：" （U+0022）、+ （U+002B）、, （U+002C）、; （U+003B）、< （U+003C）、> （U+003E）、\\ （U+005C）；
  - 在字符串开头：空格（, U+0020）或数字符号（#，U+0023）；
  - 在字符串结尾：空格（, U+0020）。
- 不支持空字符（U+0000）。如果遇到空字符，匹配将失败。
- 不支持 RFC 4517 和 RFC 4518 标准。

例如，如果 Issuer 和 Subject 的组织（O）字符串包含尾随空格，且 Subject 的组织单位（OU）字符串包含双引号，则这些字符必须转义：

```
TLSServerCertIssuer=CN=Signing CA,OU=Development head,O=\ Example SIA\ ,DC=example,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group \"5\",O=\ Example SIA\ ,DC=example,DC=com
```

字段顺序和格式

Zabbix 遵循 RFC 4514 的建议，该规范为这些字段指定了“反向”顺序：从最低级别字段（CN）开始，依次到中间级别字段（OU、O），最后是最高级别字段（DC）。

```
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

相比之下，OpenSSL 默认会以从高层到低层的顺序显示 Issuer 和 Subject 字符串。在下面的示例中，Issuer 和 Subject 字段以最高级别（DC）开头，以最低级别（CN）结尾。空格和字段分隔符的格式也会因所使用的选项而异，因此不会与 Zabbix 所需的格式匹配。

```
$ openssl x509 -noout -in /home/zabbix/zabbix_proxy.crt -issuer -subject
issuer= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Signing CA
subject= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Zabbix proxy
```

```
$ openssl x509 -noout -text -in /home/zabbix/zabbix_proxy.crt
Certificate:
    ...
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix proxy
```

要为 Zabbix 正确格式化 *Issuer* 和 *Subject* 字符串，请使用以下选项调用 OpenSSL：

```
$ openssl x509 -noout -issuer -subject \
    -nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname\
    -in /home/zabbix/zabbix_proxy.crt
```

这样输出将按反向顺序排列，以逗号分隔，并可用于 Zabbix 配置文件和前端：

```
issuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
subject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

使用 **X.509 v3** 证书扩展的限制　在 Zabbix 中实现 X.509 v3 证书时，某些扩展可能无法得到完全支持，或者可能导致行为不一致。

Subject Alternative Name 扩展

Zabbix 不支持 *Subject Alternative Name* 扩展，该扩展用于指定其他 DNS 名称，例如 IP 地址或电子邮件地址。Zabbix 只能验证证书中 *Subject* 字段的值（请参见限制允许的证书颁发者和 Subject）。如果证书包含 `subjectAltName` 字段，证书验证结果可能会因用于编译 Zabbix 组件的具体加密工具包而异。因此，Zabbix 可能会根据这些组合接受或拒绝证书。

Extended Key Usage 扩展

Zabbix 支持 *Extended Key Usage* 扩展。不过，如果使用该扩展，通常要求同时指定 *clientAuth*（用于 TLS WWW 客户端认证）和 *serverAuth*（用于 TLS WWW 服务器认证）属性。例如：

- 在被动检查中，Zabbix agent 作为 TLS 服务器运行时，agent 证书中必须包含 *serverAuth* 属性。
- 在主动检查中，agent 作为 TLS 客户端运行时，agent 证书中必须包含 *clientAuth* 属性。

虽然 GnuTLS 可能会针对密钥用途违规发出警告，但通常仍会允许通信在这些警告下继续进行。

Name Constraints 扩展

对 *Name Constraints* 扩展的支持在不同加密工具包之间有所差异。请确保您所选用的工具包支持此扩展。根据所使用的具体工具包，如果该部分被标记为 critical，此扩展可能会限制 Zabbix 加载 CA 证书。

证书吊销列表（**CRL**）　如果证书已被泄露，证书颁发机构（CA）可以通过将该证书加入证书吊销列表（CRL）来吊销它。CRL 通过配置文件进行管理，并且可以在服务器、proxy 和 agent 配置文件中使用 `TLSCRLFile` 参数指定。例如：

```
TLSCRLFile=/home/zabbix/zabbix_crl_file.crt
```

在这种情况下，`zabbix_crl_file.crt` 可能包含来自多个 CA 的 CRL，内容可能如下所示：

```
-----BEGIN X509 CRL-----
MIIB/DCB5QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv
...
treZeUPjb7LSmZ3K2hpbZN7SoOZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t
...
CAEebS2CND3ShBedZ8YSil59O6JvaDP61lR5lNs=
-----END X509 CRL-----
```

CRL 文件仅在 Zabbix 启动时加载。要更新 CRL，请重启 Zabbix。

> **Attention:**
> 如果 Zabbix 组件使用 OpenSSL 编译，并且使用了 CRL，请确保证书链中的每个顶级 CA 和中间 CA 都在 `TLSCRLFile` 中包含对应的 CRL（即使该 CRL 为空）。

证书问题排查　OpenSSL 与 CRL 一起使用时，证书链中的某些 CA 的 CRL 未包含在 `TLSCRLFile` 中

在 *OpenSSL* 对端情况下，TLS 服务器日志中：

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
    file s3_srvr.c line 3251: error:14089086: SSL routines:ssl3_get_client_certificate:certificate verify
    TLS write fatal alert "unknown CA"
```

在 *GnuTLS* 对端情况下，TLS 服务器日志中：

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
    file rsa_pk1.c line 103: error:0407006A: rsa routines:RSA_padding_check_PKCS1_type_1:\
    block type is not 01 file rsa_eay.c line 705: error:04067072: rsa routines:RSA_EAY_PUBLIC_DECRYPT:padd
```

CRL 已过期或在服务器运行期间过期

OpenSSL，在服务器日志中：

- 过期前：

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
    SSL routines:ssl3_get_server_certificate:certificate verify failed:\
    TLS write fatal alert "certificate revoked"
```

- 过期后：

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
    SSL routines:ssl3_get_server_certificate:certificate verify failed:\
    TLS write fatal alert "certificate expired"
```

这里的关键点是：在 CRL 有效时，已吊销的证书会被报告为 “certificate revoked”。当 CRL 过期时，错误信息会变为 “certificate expired”，这相当误导。

GnuTLS，在服务器日志中：

- 过期前后相同：

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
      invalid peer certificate: The certificate is NOT trusted. The certificate chain is revoked.
```

自签名证书，未知 CA

OpenSSL，在日志中：

```
error:'self signed certificate: SSL_connect() set result code to SSL_ERROR_SSL: file ../ssl/statem/statem_
      line 1924: error:1416F086:SSL routines:tls_process_server_certificate:certificate verify failed:\
      TLS write fatal alert "unknown CA"'
```

当服务器证书由于误操作具有相同的 Issuer 和 Subject 字符串时，就会观察到这种情况，尽管它是由 CA 签名的。Issuer 和 Subject 在顶级 CA 证书中可以相同，但在服务器证书中不能相同。（这同样适用于 proxy 和 agent 证书。）

要检查证书是否包含相同的 Issuer 和 Subject 条目，请运行：

```
openssl x509 -in <yourcertificate.crt> -noout -text
```

根（顶级）证书的 Issuer 和 Subject 具有相同的值是可以接受的。

使用预共享密钥进行加密

概述　Zabbix 中的每个预共享密钥（PSK）实际上由以下两部分组成：

- 非机密的 PSK 标识字符串；
- 机密的 PSK 字符串值。

PSK 标识字符串是非空的 UTF-8 字符串。例如：”PSK ID 001 Zabbix agentd”。Zabbix 组件通过此唯一名称引用特定的 PSK。请勿在 PSK 标识字符串中放入敏感信息，因为该字符串会以未加密的形式通过网络传输。

PSK 值是由十六进制数字组成的、难以猜测的字符串，例如：e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e

> **Warning:**
> 切勿在生产环境中使用文档中提供的示例 PSK 值。请改为生成唯一且经过加密保护的 PSK。

大小限制　Zabbix 对 PSK identity 和 PSK value 有大小限制，但在某些情况下，加密库的限制可能更低：

组件	PSK identity 最大大小	PSK value 最小大小	PSK value 最大大小
Zabbix	128 个 UTF-8 字符	128 位（16 字节 PSK，输入为 32 个十六进制数字）	2048 位（256 字节 PSK，输入为 512 个十六进制数字）
GnuTLS	128 字节（可能包含 UTF-8 字符）	-	2048 位（256 字节 PSK，输入为 512 个十六进制数字）
OpenSSL 1.0.x, 1.1.0	127 字节（可能包含 UTF-8 字符）	-	2048 位（256 字节 PSK，输入为 512 个十六进制数字）
OpenSSL 1.1.1	127 字节（可能包含 UTF-8 字符）	-	512 位（64 字节 PSK，输入为 128 个十六进制数字）
OpenSSL 1.1.1a 及更高版本	127 字节（可能包含 UTF-8 字符）	-	2048 位（256 字节 PSK，输入为 512 个十六进制数字）

Attention:
Zabbix 前端允许配置长度最多为 128 个字符的 PSK identity 字符串，以及长度最多为 2048 位的 PSK，而不受所使用加密库的影响。
 如果某些 Zabbix 组件支持的限制更低，则用户有责任为 PSK identity 和 PSK value 配置这些组件允许的长度。
 超过长度限制会导致 Zabbix 组件之间通信失败。

在 Zabbix 服务器使用 PSK 连接到 agent 之前，服务器会在数据库中查找为该 agent 配置的 PSK identity 和 PSK value（实际上是在配置缓存中查找）。收到连接后，agent 会使用其配置文件中的 PSK identity 和 PSK value。如果双方具有相同的 PSK identity 字符串和 PSK value，则连接可能成功。

Attention:
每个 PSK identity 只能与一个 value 配对。用户有责任确保不存在具有相同 identity 字符串但 value 不同的两个 PSK。否则，使用此 PSK identity 字符串的 Zabbix 组件之间可能出现不可预测的错误或通信中断。

生成 **PSK**　例如，可以使用以下命令生成 256 位（32 字节）的 PSK：

- 使用 *OpenSSL*：

```
$ openssl rand -hex 32
af8ced32dfe8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
```

- 使用 *GnuTLS*：

```
$ psktool -u psk_identity -p database.psk -s 32
Generating a random key for user 'psk_identity'
Key stored to database.psk

$ cat database.psk
psk_identity:9b8eafedfaae00cece62e85d5f4792c7d9c9bcc851b23216a1d300311cc4f7cb
```

请注意，上面的"psktool" 会生成一个包含 PSK identity 及其关联 PSK 的数据库文件。Zabbix 只需要 PSK 文件中包含 PSK，因此应从文件中删除 identity 字符串和冒号（':'）。

配置用于服务器**-agent** 通信的 **PSK**(示例)　在 agent 主机上，将上面生成的 PSK 值写入文件，例如 `/home/zabbix/zabbix_agentd.psk`。该文件的第一行必须包含 PSK 值。

设置 PSK 文件的访问权限 - 该文件只能由 Zabbix 用户读取。

编辑 Zabbix agent 配置文件（`zabbix_agentd.conf`）中的 TLS 参数。例如，设置：

```
TLSConnect=psk
TLSAccept=psk
TLSPSKFile=/home/zabbix/zabbix_agentd.psk
TLSPSKIdentity=PSK 001
```

agent 将连接到服务器（主动检查），并且只接受来自服务器和 `zabbix_get` 的使用 PSK 的连接。PSK 标识将为 "PSK 001"。

重启 agent。现在，您可以使用 `zabbix_get` 测试连接，例如：

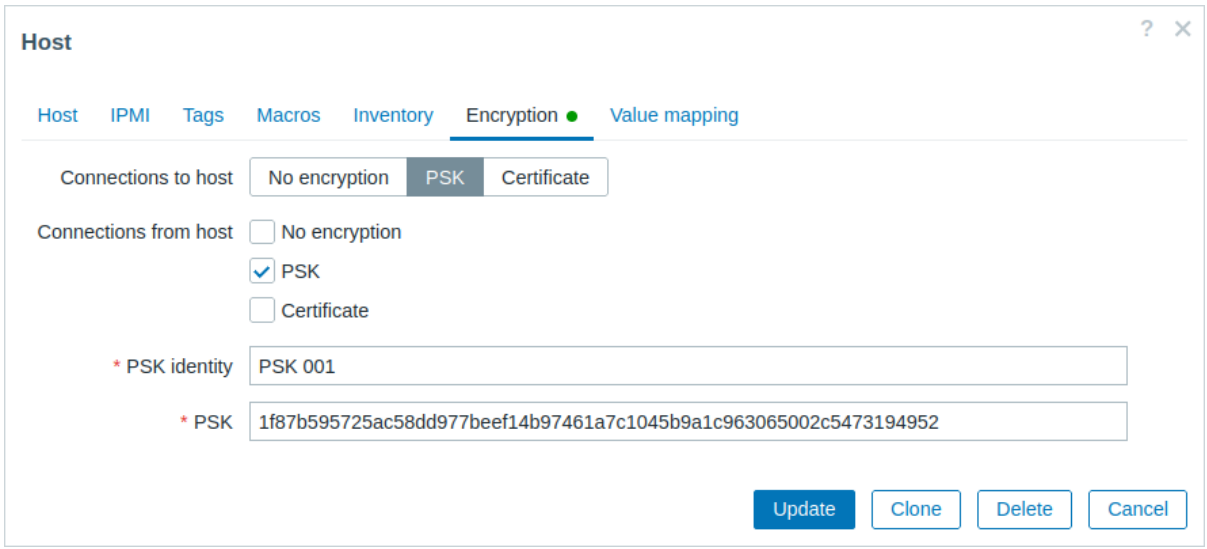
```
zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk --tls-psk-identity="PSK 001" --tl
```

(要最大限度地减少停机时间，请参阅如何在连接加密管理中更改连接类型。)

在 Zabbix 前端中为此 agent 配置 PSK 加密：

1. 转到：数据采集 > 主机。
2. 选择主机，然后单击加密选项卡。
3. 配置 PSK 加密详细信息。

示例：



所有必填输入字段均以红色星号标记。

当配置缓存与数据库同步后，新连接将使用 PSK。检查服务器和 agent 日志文件中的错误消息。

配置 **PSK** 用于服务器与主动 **proxy** 通信（示例） 在 proxy 上，将 PSK 值写入文件，例如：`/home/zabbix/zabbix_proxy.psk`。该文件的第一行文本必须包含 PSK，例如：

```
e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9
```

设置 PSK 文件的访问权限 - 该文件只能由 Zabbix 用户读取。

编辑 Zabbix proxy 配置文件（`zabbix_proxy.conf`）中的 TLS 参数。例如，设置：

```
TLSConnect=psk
TLSPSKFile=/home/zabbix/zabbix_proxy.psk
TLSPSKIdentity=PSK 002
```

proxy 将使用 PSK 连接到服务器。PSK 标识将为 “PSK 002”。

(如需最大限度减少停机时间，请参阅连接加密管理中有关如何更改连接类型的说明。)

在 Zabbix 前端中为此 proxy 配置 PSK。转到 管理 > *Proxy*，选择该 proxy，然后转到加密选项卡。在 “来自 proxy 的连接” 中勾选 PSK。在 “PSK 标识” 字段中粘贴 “PSK 002”，并在 “PSK” 字段中粘贴 “e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9”。单击 “更新”。

重启 proxy。它将开始使用基于 PSK 的加密连接连接到服务器。检查服务器和 proxy 的日志文件中是否有错误消息。

对于被动 proxy，操作过程非常相似。唯一的区别是：在 Zabbix proxy 配置文件中设置 `TLSAccept=psk`，并在 Zabbix 前端中将到 *proxy* 的连接设置为 PSK。

PSK 问题排查 PSK 包含奇数个十六进制数字

proxy 或 agent 未启动，proxy 或 agent 日志中的消息：

```
文件 "/home/zabbix/zabbix_proxy.psk" 中的 PSK 无效
```

长度超过 128 字节的 PSK 标识字符串被传递给 GnuTLS

在 TLS 客户端日志中：

```
gnutls_handshake() failed: -110 The TLS connection was non-properly terminated.
```

在 TLS 服务器端日志中：

```
gnutls_handshake() failed: -90 The SRP username supplied is illegal.
```

在 OpenSSL 1.1.1 中使用了过长的 PSK 值

在连接端日志中：

```
...OpenSSL library (version OpenSSL 1.1.1  11 Sep 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK 1"
...zbx_psk_client_cb() requested PSK identity "PSK 1"
...End of zbx_tls_connect():FAIL error:'SSL_connect() set result code to SSL_ERROR_SSL: file ssl\statem\ex
```

在接受端日志中：

```
...Message from 123.123.123.123 is missing header. Message ignored.
```

此问题通常出现在将 OpenSSL 从 1.0.x 或 1.1.0 升级到 1.1.1 时，并且 PSK 值长度超过 512 位（64 字节的 PSK，以 128 个十六进制数字输入）。

另请参见：值大小限制

到数据库的安全连接

概述

本节提供 Zabbix 的设置步骤和配置示例，用于在以下组件之间建立安全的 TLS 连接：

Database	Zabbix components
MySQL	Zabbix 前端, Zabbix 服务器, Zabbix proxy
PostgreSQL	Zabbix 前端, Zabbix 服务器, Zabbix proxy

要在 DBMS 内部设置连接加密，请参阅官方厂商文档了解详细信息：

- MySQL: 源和副本复制数据库服务器。
- MySQL: 组复制等数据库服务器。
- PostgreSQL 加密选项。

所有示例均基于通过官方仓库可获得的 MySQL CE (8.0) 和 PgSQL (13) 的 GA 版本，并使用 CentOS 8。

要求

设置加密需要满足以下条件：

- 开发者支持的操作系统，且 OpenSSL >=1.1.X 或替代方案。

> **Note:**
> 建议避免使用已进入生命周期终止状态的操作系统，尤其是在新安装的情况下

- 已从开发者提供的官方仓库安装并维护的数据库引擎（RDBMS）。操作系统通常会附带过时的数据库软件版本，而这些版本并未实现加密支持，例如基于 RHEL 7 的系统以及 PostgreSQL 9.2、MariaDB 5.5（不支持加密）。

术语

设置此选项会强制 Zabbix 服务器/proxy 和前端到数据库使用 TLS 连接：

- `required` - 使用 TLS 作为传输模式进行连接，不进行身份验证
- `verify_ca` - 使用 TLS 连接并验证证书
- `verify_full` - 使用 TLS 连接，验证证书，并验证 DBHost 指定的数据库标识（CN）与其证书匹配

Zabbix 配置

前端到数据库

可以在前端安装期间配置与数据库的安全连接：

- 在配置数据库连接步骤中勾选数据库 *TLS* 加密复选框，以启用传输加密。
- 勾选选中 *TLS* 加密字段后显示的验证数据库证书复选框，以启用基于证书的加密。

Note:
对于 MySQL，如果数据库主机设置为 localhost，则数据库 *TLS* 加密复选框处于禁用状态，因为使用套接字文件（在 Unix 上）或共享内存（在 Windows 上）的连接无法加密。
 对于 PostgreSQL，如果数据库主机字段的值以斜杠开头，或该字段为空，则 *TLS* 加密复选框处于禁用状态。

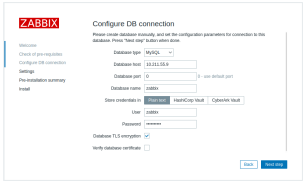
在基于证书的 TLS 加密模式下（如果两个复选框均已勾选），以下参数可用：

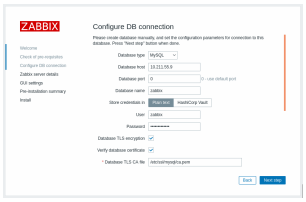
参数	描述
数据库 *TLS CA* 文件	指定有效 TLS 证书颁发机构（CA）文件的完整路径。
数据库 *TLS* 密钥文件	指定有效 TLS 密钥文件的完整路径。
数据库 *TLS* 证书文件	指定有效 TLS 证书文件的完整路径。
数据库主机验证	勾选此复选框以启用主机验证。 MySQL 不支持此选项，因为 PHP MySQL 库不允许跳过对等证书验证步骤。
数据库 *TLS* 加密套件列表	指定有效加密套件的自定义列表。加密套件列表的格式必须符合 OpenSSL 标准。 仅适用于 MySQL。

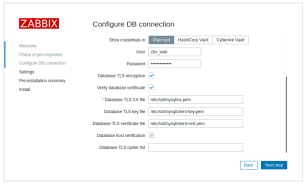
Attention:
TLS 参数必须指向有效文件。如果指向不存在或无效的文件，将导致授权错误。
 如果证书文件可写，前端会在系统信息报告中生成警告："TLS 证书文件必须为只读。"（仅当 PHP 用户是证书所有者时显示）。
不支持受密码保护的证书。

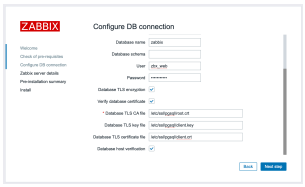
使用场景

Zabbix 前端使用 GUI 界面定义可用选项：`required`、`verify_ca`、`verify_full`。在安装向导的 配置数据库连接步骤中指定所需选项。这些选项按以下方式映射到前端配置文件（`zabbix.conf.php`）：

GUI 设置	配置文件	描述	结果
	... // 用于 TLS 连接。 $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = ''; $DB['CERT_FILE'] = ''; $DB['CA_FILE'] = ''; $DB['VERIFY_HOST'] = false; $DB['CIPHER_LIST'] = ''; ...	1. 勾选 数据库 *TLS* 加密 2. 不勾选 验证数据库证书	启用 `required` 模式。

GUI 设置	配置文件	描述	结果
	... $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = ''; $DB['CERT_FILE'] = ''; $DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem'; $DB['VERIFY_HOST'] = false; $DB['CIPHER_LIST'] = ''; ...	1. 勾选 数据库 *TLS* 加密和 验证数据库证书 2. 指定 数据库 *TLS CA* 文件的路径	启用 `verify_ca` 模式。

	...
	// 用于使用严格定义的密码套件列表进行 TLS 连接。
	$DB['ENCRYPTION'] = true;
	$DB['KEY_FILE'] = '<key_file_path>';
	$DB['CERT_FILE'] = '<key_file_path>';
	$DB['CA_FILE'] = '<key_file_path>';
	$DB['VERIFY_HOST'] = true;
	$DB['CIPHER_LIST'] = '<cipher_list>';
	...
	
	或者：
	
	...
	// 用于未定义密码套件列表的 TLS 连接 - 由 MySQL 服务器选择
	$DB['ENCRYPTION'] = true;
	$DB['KEY_FILE'] = '<key_file_path>';
	$DB['CERT_FILE'] = '<key_file_path>';
	$DB['CA_FILE'] = '<key_file_path>';
	$DB['VERIFY_HOST'] = true;
	$DB['CIPHER_LIST'] = '';
	...
		1. 勾选 数据库 *TLS* 加密和 验证数据库证书
		2. 指定 数据库 *TLS* 密钥文件的路径
		3. 指定 数据库 *TLS CA* 文件的路径
		4. 指定 数据库 *TLS* 证书文件的路径
		5. 指定 数据库 *TLS* 密码套件列表（可选）
			为 MySQL 启用 `verify_full` 模式。

GUI 设置	配置文件	描述	结果
	... $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = '<key_file_path>'; $DB['CERT_FILE'] = '<key_file_path>'; $DB['CA_FILE'] = '<key_file_path>'; $DB['VERIFY_HOST'] = true; $DB['CIPHER_LIST'] = ' '; ...	1. 勾选 数据库 *TLS* 加密和 验证数据库证书 2. 指定 数据库 *TLS* 密钥文件的路径 3. 指定 数据库 *TLS CA* 文件的路径 4. 指定 数据库 *TLS* 证书文件的路径 5. 勾选 数据库主机验证	为 PostgreSQL 启用 `verify_full` 模式。

另请参阅： MySQL 加密配置示例、PostgreSQL 加密配置示例。

Zabbix 服务器/proxy 配置

可以在 Zabbix 服务器和/或proxy 配置文件中，通过相应参数配置与数据库的安全连接。

配置	结果
无	与数据库建立未加密的连接。
设置 `DBTLSConnect=required`	服务器/proxy 与数据库建立 TLS 连接。不允许使用未加密的连接。
1. 设置 `DBTLSConnect=verify_ca` 2. 设置 `DBTLSCAFile` - 指定 TLS 证书颁发机构文件	服务器/proxy 在验证数据库证书后，与数据库建立 TLS 连接。
1. 设置 `DBTLSConnect=verify_full` 2. 设置 `DBTLSCAFile` - 指定 TLS 证书颁发机构文件	服务器/proxy 在验证数据库证书和数据库主机身份后，与数据库建立 TLS 连接。
1. 设置 `DBTLSCAFile` - 指定 TLS 证书颁发机构文件 2. 设置 `DBTLSCertFile` - 指定客户端公钥证书文件 3. 设置 `DBTLSKeyFile` - 指定客户端私钥文件	服务器/proxy 在连接数据库时提供客户端证书。
设置 `DBTLSCipher` - 客户端允许用于 TLS 1.2 及更低版本 TLS 协议连接的加密密码套件列表 或设置 `DBTLSCipher13` - 客户端允许用于 TLS 1.3 协议连接的加密密码套件列表	(MySQL) TLS 连接将使用所提供列表中的密码套件建立。 (PostgreSQL) 设置此选项将被视为错误。

MySQL 加密配置

概述

本节提供了 CentOS 8.2 和 MySQL 8.4.0 的若干加密配置示例，可作为加密数据库连接的快速入门指南。

Attention:
如果 MySQL 主机设置为 localhost，则加密选项将不可用。在这种情况下，Zabbix 前端与数据库之间的连接使用套接字文件（在 Unix 上）或共享内存（在 Windows 上），因此无法加密。

Note:
加密组合的列表并不局限于本页所列内容。还可以使用更多组合。

前提条件

从官方仓库安装 MySQL 数据库。

有关如何使用 MySQL 仓库的详细信息，请参见MySQL 文档。

MySQL 服务器已准备好使用自签名证书接受安全连接。

要查看哪些用户正在使用加密连接，请运行以下查询（Performance Schema 应处于 ON 状态）：

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

仅传输加密

MySQL 配置

现代版本的数据库开箱即用，已准备好使用 `required` 加密模式。完成初始设置并启动后，将创建服务器端证书。

为主要组件创建用户和角色（将 <strong_password> 替换为强度高且唯一的密码）：

```
mysql> CREATE USER
 'zbx_srv'@'%' IDENTIFIED WITH caching_sha2_password BY '<strong_password>',
 'zbx_web'@'%' IDENTIFIED WITH caching_sha2_password BY '<strong_password>'
 REQUIRE SSL
 PASSWORD HISTORY 5;

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

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

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

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

请注意，这里不会使用 X.509 协议检查身份，而是将用户配置为仅使用加密连接。有关配置用户的更多详细信息，请参阅 MySQL 文档。

运行以下命令检查连接（不能使用套接字连接测试安全连接）：

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

检查当前状态和可用的密码套件：

```
mysql> status
--------------
mysql Ver 8.4.0 for Linux on x86_64 (MySQL Community Server - GPL)

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


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

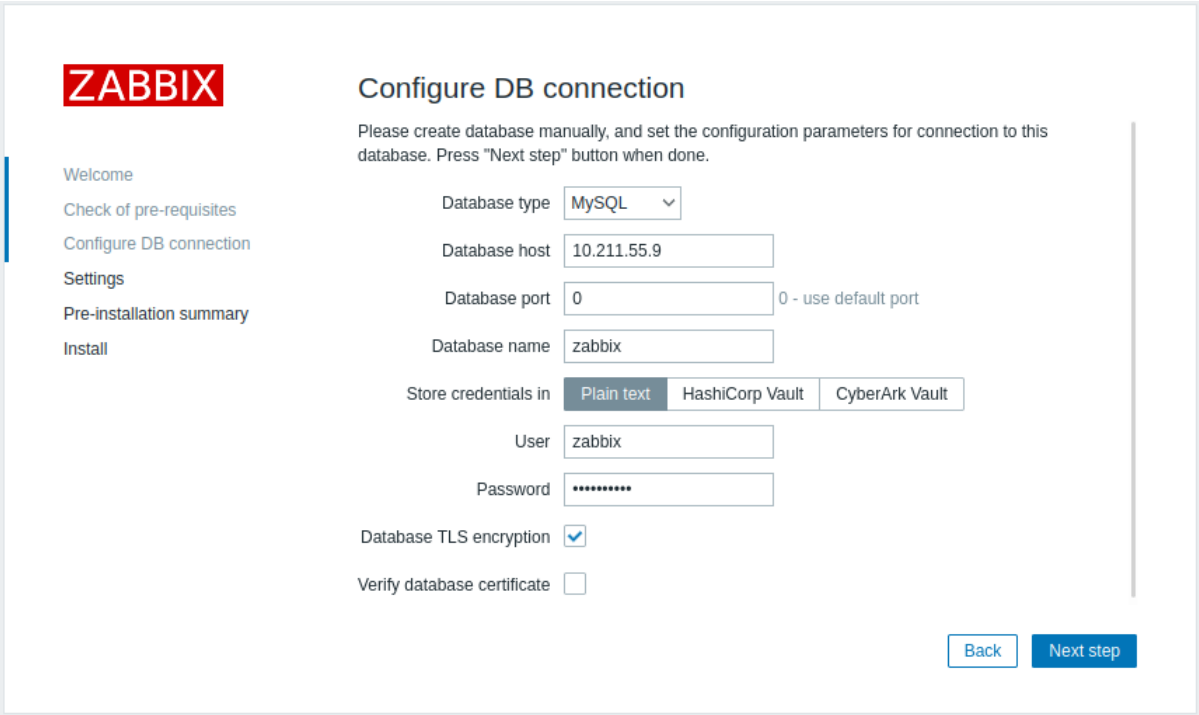
ERROR:
No query specified
```

前端

要为 Zabbix 前端与数据库之间的连接启用仅传输加密：

- 选中 *Database TLS encryption*

- 保持 *Verify database certificate* 未勾选



服务器

要为服务器与数据库之间的连接启用仅传输加密，请编辑服务器配置文件：

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

使用证书颁发机构验证进行加密

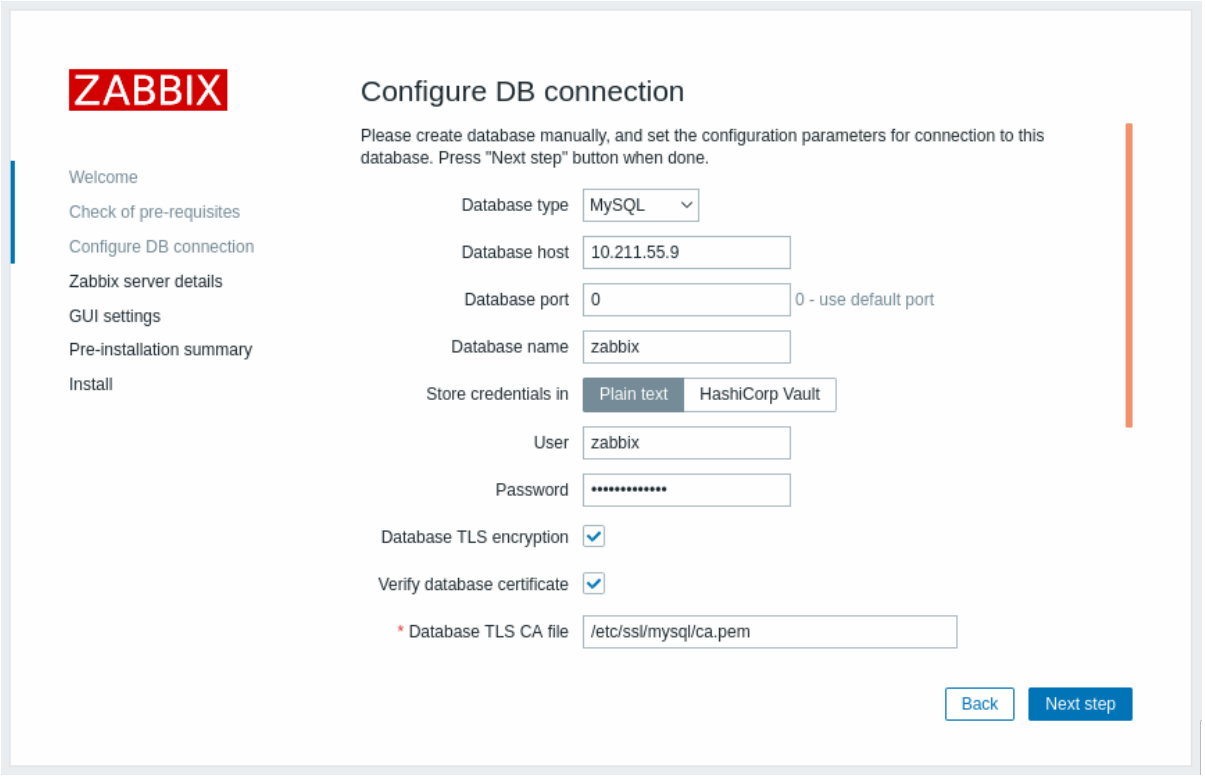
将所需的 MySQL CA 复制到 Zabbix 前端服务器，并分配适当的权限，以允许 web 服务器读取此文件。

> **Note:**
> 由于 MySQL 库版本较旧，此模式在 RHEL 7 上无法工作。

前端

要为 Zabbix 前端与数据库之间的连接启用带证书验证的加密：

- 勾选 数据库 *TLS* 加密和 验证数据库证书
- 指定数据库 TLS CA 文件的路径



或者，也可以在前端配置文件（zabbix.conf.php）中进行设置：

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

使用命令行工具检查所需用户是否可以建立连接，以排查问题：

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

服务器

要为 Zabbix 服务器与数据库之间的连接启用带证书验证的加密，请编辑服务器配置文件：

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

使用完整验证的加密

MySQL 配置

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

```
[mysqld]
...
#### 在此示例中，密钥位于 MySQL CE datadir 目录中
ssl_ca=ca.pem
ssl_cert=server-cert.pem
ssl_key=server-key.pem

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

MySQL CE 服务器和客户端（Zabbix 前端）的密钥应根据 MySQL CE 文档手动创建：使用 MySQL 创建 SSL 和 RSA 证书及密钥 或使用 openssl 创建 SSL 证书及密钥

> **Attention:**
> MySQL 服务器证书应包含 Common Name 字段，并将其设置为 FQDN 名称，因为 Zabbix 前端将使用 DNS 名称与数据库通信，或者使用数据库主机的 IP 地址。

创建 MySQL 用户：

```
mysql> CREATE USER
  'zbx_srv'@'%' IDENTIFIED WITH caching_sha2_password BY '<strong_password>',
  'zbx_web'@'%' IDENTIFIED WITH caching_sha2_password BY '<strong_password>'
  REQUIRE X509
  PASSWORD HISTORY 5;
```

检查是否可以使用该用户登录：

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

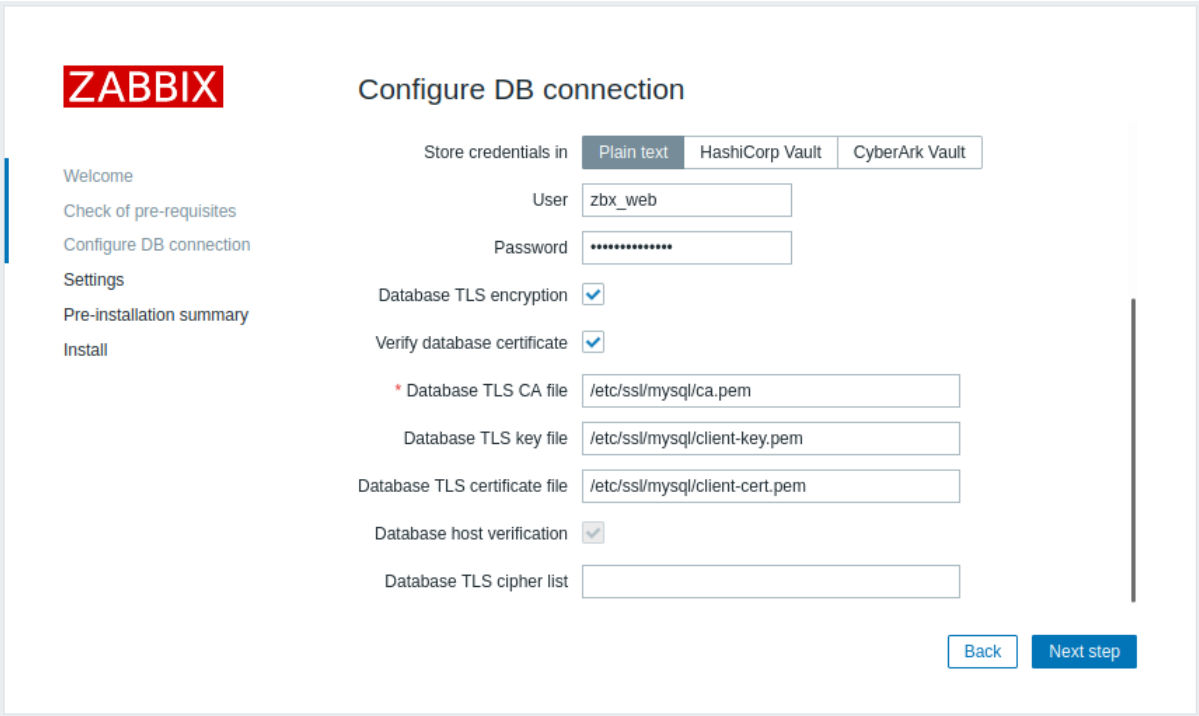
前端

要为 Zabbix 前端与数据库之间的连接启用完全验证的加密：

- 勾选 数据库 *TLS* 加密和 验证数据库证书
- 指定 数据库 *TLS* 密钥文件的路径
- 指定 数据库 *TLS CA* 文件的路径
- 指定 数据库 *TLS* 证书文件的路径

请注意，数据库主机验证已被勾选且显示为灰色 - MySQL 不允许跳过此步骤。

> **Warning:**
> 如果 数据库 *TLS* 加密套件列表字段留空，将启用前端（客户端）和服务器共同允许的加密套件。或者，也可以根据加密套件配置要求显式设置加密套件。



或者，也可以在前端配置文件（zabbix.conf.php）中进行设置：

```
...
// Used for TLS connection with strictly defined Cipher list.
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
```

```
$DB['CIPHER_LIST'] = 'TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_1
...
// or
...
// Used for TLS connection without Cipher list defined - selected by MySQL server
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';
...
```

服务器

要为 Zabbix 服务器与数据库之间的连接启用完全验证加密，请编辑服务器配置文件：

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/mysql/ca.pem
DBTLSCertFile=/etc/ssl/mysql/client-cert.pem
DBTLSKeyFile=/etc/ssl/mysql/client-key.pem
...
```

PostgreSQL 加密配置

概述

本节提供了几个适用于 CentOS 8.2 和 PostgreSQL 13 的加密配置示例。

> **Note:**
> 如果 *Database host* 字段的值以斜杠开头，或者该字段为空，则 Zabbix 前端与 PostgreSQL 之间的连接无法加密（GUI 中的参数会被禁用）。

前提条件

使用 官方仓库 安装 PostgreSQL 数据库。

PostgreSQL 默认未配置为接受 TLS 连接。请按照 PostgreSQL 文档中的说明，完成 使用 postgresql.conf 进行证书准备，以及通过 ph_hba.conf 进行 用户访问控制。

默认情况下，PostgreSQL socket 绑定到 localhost；对于网络远程连接，需要允许其在真实的网络接口上监听。

所有模式 的 PostgreSQL 设置可以如下所示：

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

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

对于访问控制，请调整 /var/lib/pgsql/13/data/pg_hba.conf：

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

### verify CA
```

```
hostssl all all 0.0.0.0/0 md5 clientcert=verify-ca

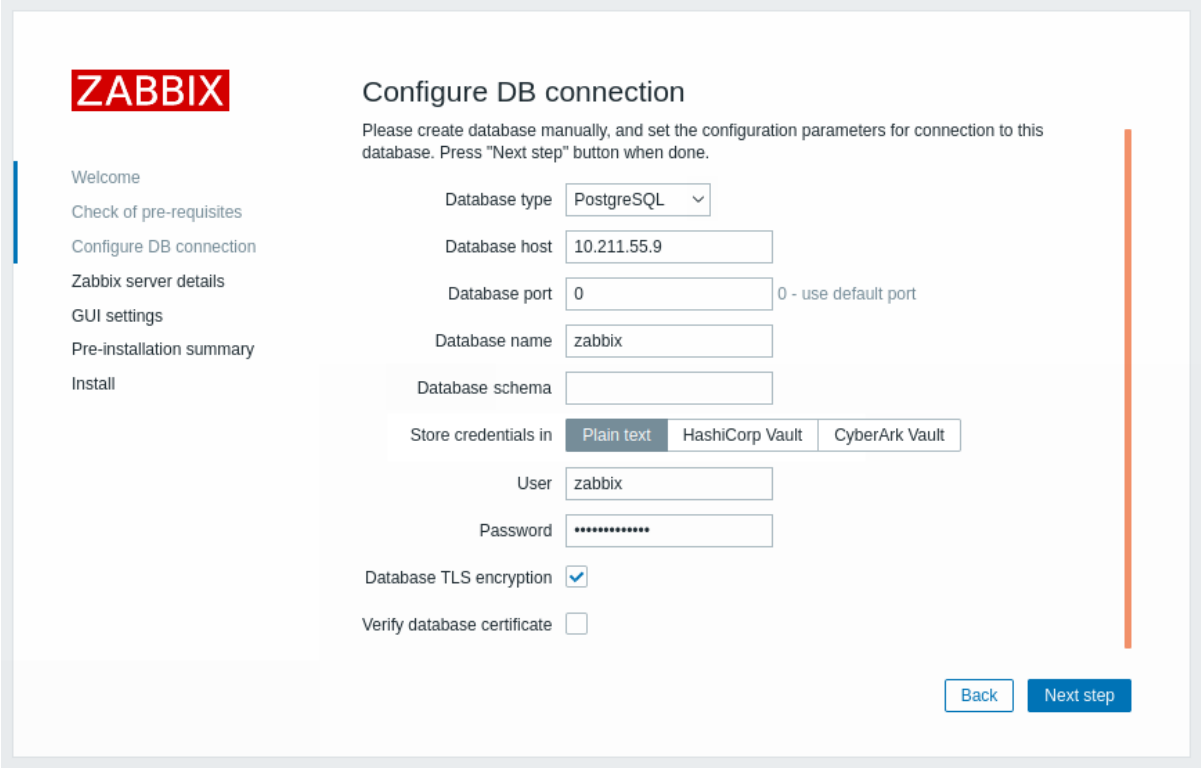
### verify full
hostssl all all 0.0.0.0/0 md5 clientcert=verify-full
...
```

仅传输加密

前端

要为 Zabbix 前端与数据库之间的连接启用仅传输加密：

- 选中 *Database TLS encryption*
- 保持 *Verify database certificate* 未勾选



服务器

要为服务器与数据库之间的连接启用仅传输加密，请编辑服务器配置文件：

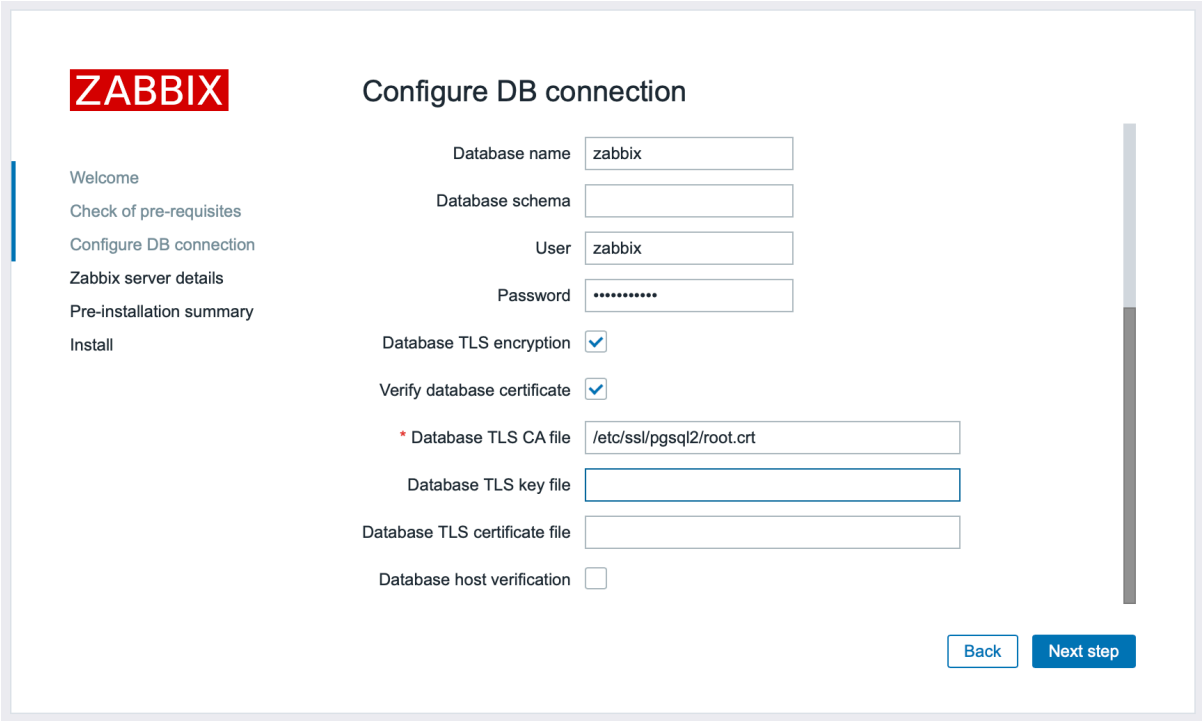
```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=required
...
```

使用证书颁发机构验证进行加密

前端

要为 Zabbix 前端与数据库之间的连接启用带证书颁发机构验证的加密：

- 勾选 数据库 *TLS* 加密和 验证数据库证书
- 指定 数据库 *TLS CA* 文件的路径



或者，也可以在前端配置文件（zabbix.conf.php）中进行设置：

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

服务器

要为 Zabbix 服务器与数据库之间的连接启用带证书验证的加密，请编辑服务器配置文件（将 <strong_password> 替换为强度高且唯一的密码）：

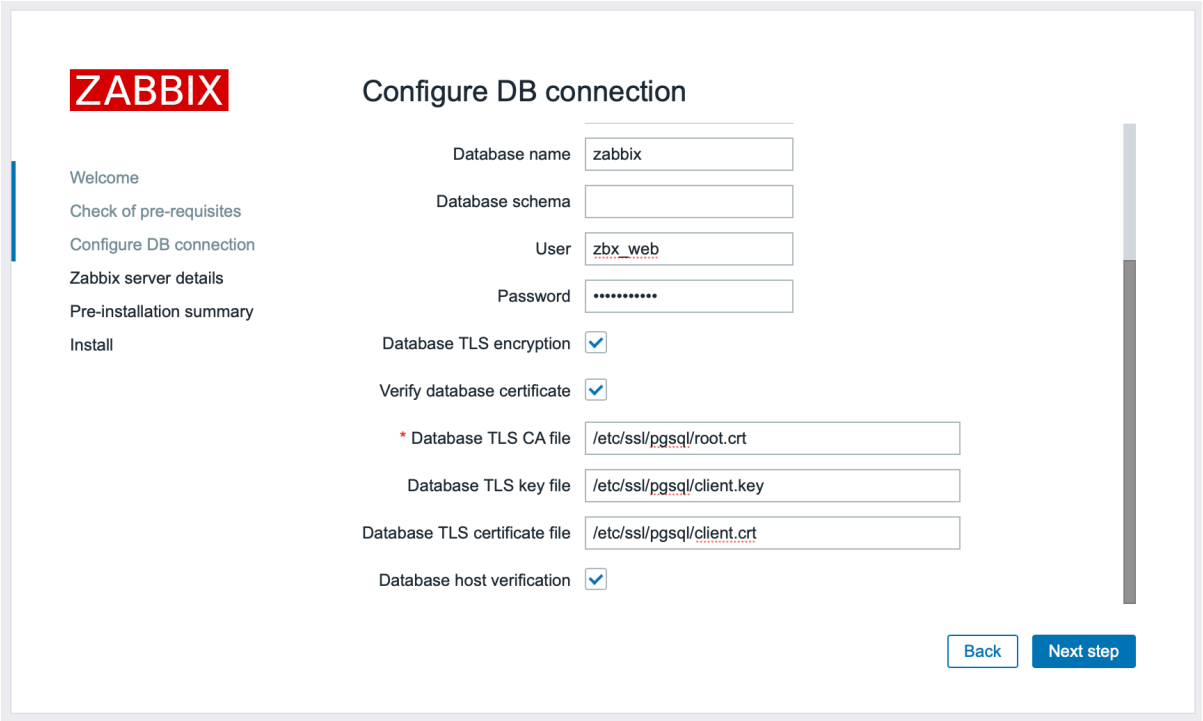
```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/pgsql/root.crt
...
```

使用完整验证的加密

前端

要为 Zabbix 前端与数据库之间的连接启用基于证书的加密和数据库主机身份验证：

- 勾选 数据库 *TLS* 加密和 验证数据库证书
- 指定 数据库 *TLS* 密钥文件的路径
- 指定 数据库 *TLS CA* 文件的路径
- 指定 数据库 *TLS* 证书文件的路径
- 勾选 数据库主机验证



或者，也可以在前端配置文件（zabbix.conf.php）中进行设置：

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

服务器

要为 Zabbix 服务器与数据库之间的连接启用加密，并验证数据库主机身份，请编辑服务器配置文件：

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/pgsql/root.crt
DBTLSCertFile=/etc/ssl/pgsql/client.crt
DBTLSKeyFile=/etc/ssl/pgsql/client.key
...
```

前端与服务器之间的安全连接

概述

本节提供 Zabbix 前端与 Zabbix 服务器之间建立安全 TLS 连接的设置步骤和配置示例。

配置

默认情况下，Zabbix 前端与 Zabbix 服务器之间的通信未加密。为提高安全性，请在两端启用 TLS。下面介绍一种最简单的配置方法。

1. 生成证书和密钥。

创建工作目录：

```
sudo mkdir -p /etc/zabbix/ssl && cd /etc/zabbix/ssl
```

创建 CA 证书（调整 MyZabbixCA 值，使其与实际的通用名称匹配）：

```
sudo openssl genrsa -out ca.key 4096
sudo openssl req -new -x509 -days 3650 -key ca.key -out ca.crt -subj "/CN=MyZabbixCA/"
```

为 Zabbix 服务器生成私钥和证书（调整 zabbix-server.example.com 值，使其与实际的通用名称匹配）：

```
sudo openssl genrsa -out server.key 2048
sudo openssl req -new -key server.key -out server.csr -subj "/CN=zabbix-server.example.com/"
sudo openssl x509 -req -days 365 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -sha256 -out serv
```

为 Zabbix 前端生成私钥和证书（调整 zabbix-frontend-node 值，使其与实际的通用名称匹配）：

```
sudo openssl genrsa -out frontend.key 2048
sudo openssl req -new -key frontend.key -out frontend.csr -subj "/CN=zabbix-frontend-node/"
sudo openssl x509 -req -days 365 -in frontend.csr -CA ca.crt -CAkey ca.key -CAcreateserial -sha256 -out fr
```

2. 设置适当的权限。

对于 Zabbix 服务器（根据您的发行版中 Zabbix 服务器守护进程用户调整所有者和组）：

```
sudo chown root:zabbix /etc/zabbix/ssl/server.{crt,key} /etc/zabbix/ssl/ca.crt
sudo chmod 640 /etc/zabbix/ssl/server.key
sudo chmod 644 /etc/zabbix/ssl/server.crt /etc/zabbix/ssl/ca.crt
```

对于前端（根据您的发行版中 Web 服务器用户调整所有者和组）：

```
sudo chown root:www-data /etc/zabbix/ssl/frontend.{crt,key}
sudo chmod 640 /etc/zabbix/ssl/frontend.key
sudo chmod 644 /etc/zabbix/ssl/frontend.crt
```

3. 配置 Zabbix 服务器。

在服务器配置文件（zabbix_server.conf）中添加：

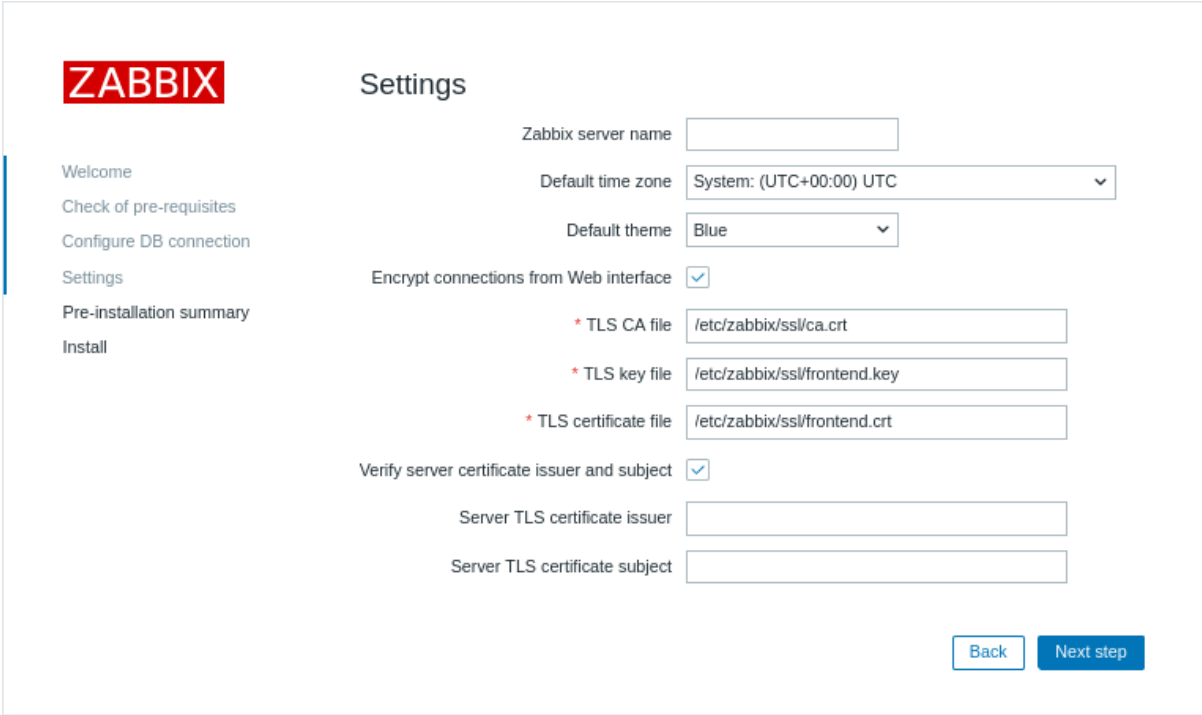
```
TLSFrontendAccept=cert
TLSCertFile=/etc/zabbix/ssl/server.crt
TLSKeyFile=/etc/zabbix/ssl/server.key
TLSCAFile=/etc/zabbix/ssl/ca.crt
### 可选：
### TLSFrontendCertIssuer=CN=MyZabbixCA
### TLSFrontendCertSubject=CN=zabbix-frontend-node
```

然后重启服务器：

```
sudo systemctl restart zabbix-server
```

4. 配置 Zabbix 前端。

在Web 界面安装期间，启用 *Encrypt connections from Web interface* 选项（如有必要，同时启用 *Verify server certificate issuer and subject* 选项），并填写 *TLS CA file*、*TLS key file*、*TLS certificate file* 字段（如有必要，同时填写 *Server TLS certificate issuer* 和 *Server TLS certificate subject* 字段）：



Parameter	Description
TLS CA file	指定用于验证服务器证书的证书颁发机构（CA）证书文件的完整路径。
TLS key file	指定与客户端证书对应的客户端私钥文件的完整路径。
TLS certificate file	如果需要双向 TLS 身份验证，请指定客户端证书文件的完整路径。
Server TLS certificate issuer	指定要与服务器证书匹配的颁发者可分辨名称（DN）。
Server TLS certificate subject	指定要与服务器证书匹配的使用者可分辨名称（DN）。

对于现有安装，请编辑前端配置文件中的以下参数：

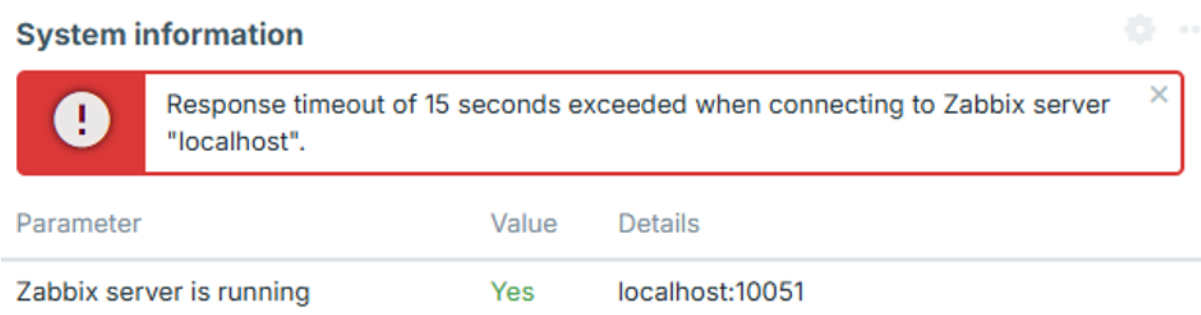
```
$ZBX_SERVER_TLS['ACTIVE'] = 'true';
$ZBX_SERVER_TLS['CA_FILE'] = '/etc/zabbix/ssl/ca.crt';
$ZBX_SERVER_TLS['KEY_FILE'] = '/etc/zabbix/ssl/frontend.key';
$ZBX_SERVER_TLS['CERT_FILE'] = '/etc/zabbix/ssl/frontend.crt';
// 可选：
// $ZBX_SERVER_TLS['CERTIFICATE_ISSUER']  = 'CN=MyZabbixCA';
// $ZBX_SERVER_TLS['CERTIFICATE_SUBJECT'] = 'CN=zabbix-server.example.com';
```

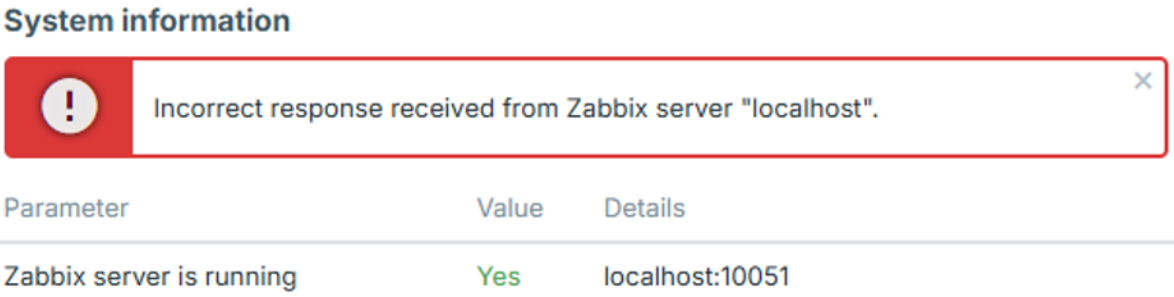
5. 通过确认 Zabbix 前端或 Zabbix 服务器日志文件中没有错误消息来验证加密：

```
tail -f /var/log/zabbix/zabbix_server.log
```

故障排除

不正确或不完整的 TLS 配置可能会导致 Zabbix 前端出现错误，或表明 Zabbix 服务器正在运行，但连接未能成功建立：





要确定确切的配置问题，请将 Zabbix 服务器日志级别设置为 4，在前端中重现该错误，然后检查 Zabbix 服务器日志中对应的错误消息。

以下情况介绍了常见错误及其可能原因。

不允许前端 IP 地址

Zabbix 服务器日志中包含类似以下错误：

```
54283:20260819:105926.400 frontend connection from "127.0.0.1" is not allowed by FrontendAllowedIP
```

由于前端 IP 地址未获授权，连接会在 TCP 握手后被丢弃。

将前端 IP 地址添加到 `zabbix_server.conf` 中的 `FrontendAllowedIP` 参数，然后重启 Zabbix 服务器。

不允许未加密连接

Zabbix 服务器日志中包含类似以下错误：

```
56884:20260819:111154.285 frontend connection of type "unencrypted" is not allowed by TLSFrontendAccept
```

Zabbix 服务器配置为仅接受 TLS 连接，例如设置了 `TLSFrontendAccept=cert`，但前端尝试在未加密的情况下进行连接。

启用前端加密，并在 `zabbix.conf.php` 中配置所需的 TLS 文件，具体说明请参阅配置中的第 4 步。

前端尝试使用 TLS，而服务器要求未加密连接

Zabbix 服务器日志中包含类似以下错误：

```
61091:20260819:115916.019 End of zbx_tls_accept():FAIL error:'TLS handshake set result code to 1: file ../
```

具体错误消息取决于 OpenSSL 版本。

Zabbix 服务器配置为仅接受未加密连接，例如设置了 `TLSFrontendAccept=unencrypted`，而前端尝试建立 TLS 连接。

通过设置 `TLSFrontendAccept=cert`，并在 `zabbix_server.conf` 中定义 `TLSCertFile` 和 `TLSKeyFile`，将 Zabbix 服务器配置为接受 TLS 连接。
或者，在 `zabbix.conf.php` 中禁用前端加密。

前端证书不受信任

Zabbix 服务器日志中包含类似以下错误：

```
62248:20260819:124203.812 End of zbx_tls_accept():FAIL error:'self-signed certificate: TLS handshake set r
```

前端提供了证书，但 Zabbix 服务器不信任签署该证书的证书颁发机构（CA）。

同样的问题也可能显示在前端的系统信息中：

```
fread(): SSL operation failed with code 1. OpenSSL Error messages:

error:0A000418:SSL routines::tlsv1 alert unknown ca [zabbix.php:17 → require_once() → ZBase->run() → ZBase
```

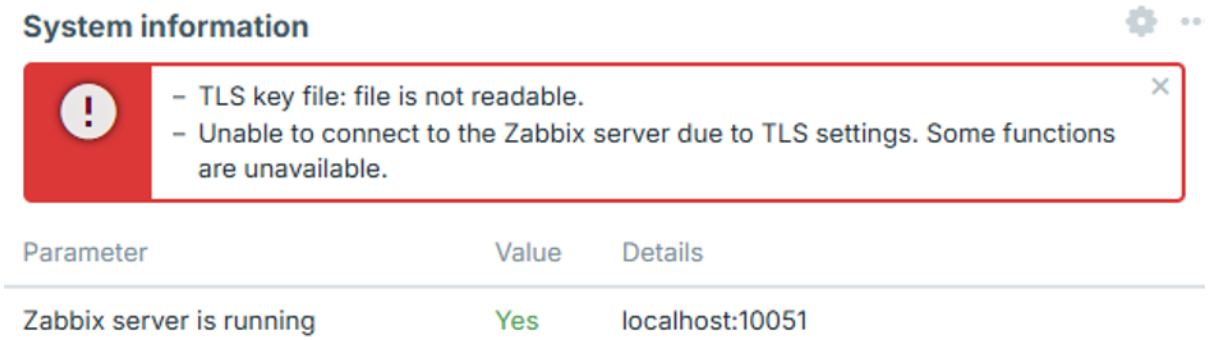
前端还可能显示 “从 Zabbix 服务器”localhost” 收到的响应不正确”。

请确认前端证书由 `zabbix_server.conf` 中 `TLSCAFile` 指定的 CA 签署。

TLS 密钥文件不可读

前端可能会显示 “TLS 密钥文件：文件不可读。”

也可能显示 “由于 TLS 设置，无法连接到 Zabbix 服务器。部分功能不可用。”

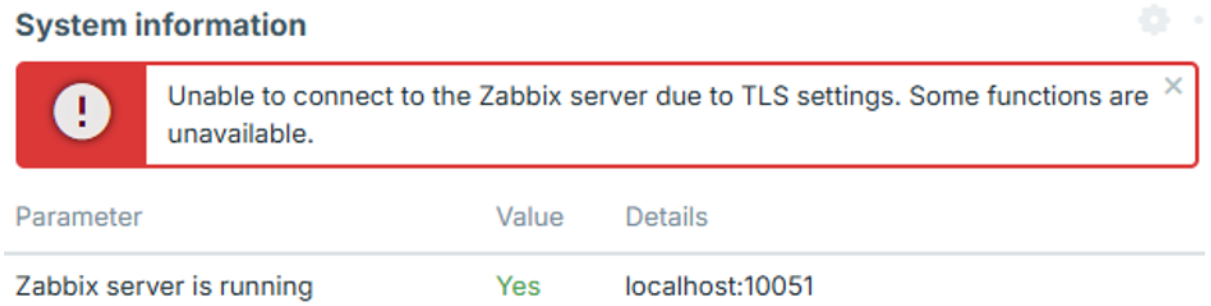


Web 服务器用户没有足够的权限读取 `zabbix.conf.php` 中配置的一个或多个证书、密钥或 CA 文件。如果证书文件存储在受限目录中（例如 /root/ 或用户的主目录），就可能发生这种情况。

将证书文件移动到 Web 服务器可访问的目录，并按照配置第 2 步中的说明设置适当的权限。

TLS 文件无效或已损坏

前端可能会显示 “由于 TLS 设置，无法连接到 Zabbix 服务器。部分功能不可用。”。



如果已配置的证书或密钥文件存在且可读取，但其内容为空、已损坏或以其他方式无效，则可能会发生此问题。在这种情况下，前端不会与 Zabbix 服务器建立连接，并且 Zabbix 服务器日志中不会出现相应的连接尝试记录。

请验证证书和密钥文件包含有效且格式正确的数据，并确认私钥与证书相匹配。

访问控制

概述

本节包含以安全方式设置访问控制的最佳实践。

最小权限原则

用户账户在任何时候都应以尽可能少的权限运行。这意味着，Zabbix 前端中的用户账户、数据库用户，或用于 Zabbix 服务器/proxy/agent 进程的用户，只应拥有执行预期功能所必需的权限。

Attention:
为’zabbix’ 用户授予额外权限将允许其访问配置文件并执行可能危及基础设施安全的操作。

在配置用户账户权限时，应考虑 Zabbix 前端用户类型。请注意，尽管 *Admin* 用户类型的权限少于 *Super Admin* 用户类型，但它仍然可以管理配置并执行自定义脚本。

Note:
即使对于非特权用户，也可以获取某些信息。例如，虽然 *Alerts* → *Scripts* 仅对 *Super Admin* 用户可用，但脚本也可以通过 Zabbix API 获取。在这种情况下，限制脚本权限并从脚本中排除敏感信息（例如访问凭据），有助于避免在全局脚本中暴露敏感信息。

为 Zabbix agent 配置安全用户

默认情况下，Zabbix 服务器、proxy 和 agent（或 agent 2）进程共用一个 `zabbix` 用户。为防止 Zabbix agent/agent 2（运行在与服务器/proxy 相同的机器上）访问服务器/proxy 配置中的敏感信息（例如数据库凭据），应使用不同的用户运行 agent：

对于 Zabbix agent：

1. 创建一个安全的组和用户（例如 zabbix-agent）。
2. 在 agent 配置文件中将此用户设置到User 参数。
3. 重启 agent，使其降权到新用户。

对于 Zabbix agent 2，必须在 service 级别应用配置，因为agent 2 配置文件不支持 User 参数。示例请参见 ZBX-26442。

撤销对 SSL 配置的写入权限（Windows）

如果你在 Windows 上编译了 Zabbix agent，并且 OpenSSL 位于一个未受保护的目录中（例如 C:\zabbix、c:\openssl-64bit、C:\OpenSSL-Win64-111-static 或 C:\dev\openssl），请确保撤销非管理员用户对此目录的写入权限。
否则，agent 会从一个可被非特权用户修改的路径加载 SSL 设置，从而可能导致安全漏洞。

加固 Zabbix 组件的安全性

可以关闭某些功能，以加固 Zabbix 组件的安全性：

- 可以在服务器配置中将 EnableGlobalScripts=0 设置为禁用 Zabbix 服务器上的全局脚本执行。
- Zabbix proxy 默认禁用全局脚本执行（可以在proxy 配置中将 EnableRemoteCommands=1 设置为启用）。
- Zabbix agent 默认禁用全局脚本执行(可以在agent 配置中为每个允许执行的命令添加 AllowKey=system.run[<command>,*] 或 AllowKeyRegexp=^system\.run\[.*\]$ 参数来启用)。
- 默认启用访问用户HTTP 身份验证配置的功能；可以在前端配置中将 $ZBX_FEATURE_FLAGS['http_auth_enabled'] = false 设置为禁用。
- 默认启用访问模块配置的功能；可以在前端配置中将 $ZBX_FEATURE_FLAGS['modules_config_enabled'] = false 设置为禁用。
- 可以在前端配置中设置 $ZBX_FEATURE_FLAGS['media_type_denylist'] = [<value1>, <value2>, ...]，以限制对特定媒介类型的编辑。可用值：'email'、'script'、'sms'、'webhook' 和 'push'。可以指定一个或多个值。

如果 zabbix.conf.php 配置文件存在，则通过运行 setup.php 重新安装前端会受到限制。

Zabbix agent 在 Windows 上访问 UNC 路径

Windows 上的 Zabbix agent 会在 vfs.file.*、vfs.dir.*、modbus.get 和 perf_counter* 等监控项中访问 UNC 路径（如 \\server\share\file.txt）。在某些情况下，这可能会带来安全风险。

当 Windows 被要求访问 UNC 路径时，它会尝试在该服务器上进行身份验证。这意味着，恶意请求 Zabbix agent 可能会将 NTLM 哈希暴露给请求方服务器。用户可以根据需要配置AllowKey、AllowKeyRegexp、DenyKey 和DenyKeyRegexp 配置参数，以降低此风险。

保护 **MySQL/MariaDB**

概述

本节包含用于保护 MySQL/MariaDB 数据库的最佳实践。

> **Note:**
> 对于基本设置，请参阅默认的MySQL/MariaDB 数据库创建 说明，其中包括创建对 Zabbix 数据库拥有全部权限的'zabbix' 用户。该用户是数据库所有者，在升级 Zabbix 时，也拥有修改数据库结构所需的必要权限。

为提高安全性，建议创建额外的数据库角色和用户，并为其分配最小权限。这些角色和用户应根据最小权限原则 进行配置，也就是说，它们只应拥有执行预期功能所必需的权限。

创建用户角色

创建以下角色，并赋予相应权限：

- **zbx_srv** - 用于运行 Zabbix 服务器和 proxy 的角色：

```
CREATE ROLE 'zbx_srv';
GRANT DELETE, INSERT, SELECT, UPDATE ON zabbix.* TO 'zbx_srv';
FLUSH PRIVILEGES;
```

- **zbx_web** - 用于运行 Zabbix 前端和 API 的角色：

```
CREATE ROLE 'zbx_web';
GRANT DELETE, INSERT, SELECT, UPDATE ON zabbix.* TO 'zbx_web';
FLUSH PRIVILEGES;
```

- **zbx_bckp** - 用于表备份的角色：

```
CREATE ROLE 'zbx_bckp';
GRANT LOCK TABLES, TRIGGER, SELECT ON zabbix.* TO 'zbx_bckp';
```

```
GRANT process ON *.* TO 'zbx_bckp';
FLUSH PRIVILEGES;
```

Note:
表恢复和升级应由数据库所有者执行。

- **zbx_part** - 具有较少权限集的数据库分区角色；请注意，此角色只能在数据库创建后创建，因为它会对特定数据库表授予权限：

```
CREATE ROLE 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.history TO 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.history_uint TO 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.history_str TO 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.history_text TO 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.history_log TO 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.trends TO 'zbx_part';
GRANT SELECT, ALTER, DROP ON zabbix.trends_uint TO 'zbx_part';
-- For MariaDB: skip the next line (GRANT session_variables_admin ON *.* TO 'zbx_part';)
GRANT session_variables_admin ON *.* TO 'zbx_part';
GRANT SELECT ON zabbix.dbversion TO 'zbx_part';
GRANT SELECT, DELETE ON zabbix.housekeeper TO 'zbx_part';
FLUSH PRIVILEGES;
```

创建角色后，即可将其分配给用户。

分配用户角色

要分配已创建的用户角色，请创建用户并为其分配相关角色。将 <user>、<host>、<role> 和 <password> 替换为相应的值。确保 <password> 足够强且具有唯一性。

```
CREATE USER '<user>'@'<host>' IDENTIFIED BY '<password>';
GRANT '<role>' TO '<user>'@'<host>';
SET DEFAULT ROLE '<role>' TO '<user>'@'<host>';
-- 对于 MariaDB：SET DEFAULT ROLE '<role>' FOR '<user>'@'<host>'
FLUSH PRIVILEGES;
```

例如，要创建并分配用于运行 Zabbix 服务器和 proxy 的角色：

```
CREATE USER 'usr_srv'@'localhost' IDENTIFIED BY 'strong-password';
GRANT 'zbx_srv' TO 'usr_srv'@'localhost';
SET DEFAULT ROLE ALL TO 'usr_srv'@'localhost';
FLUSH PRIVILEGES;
```

保护 **PostgreSQL/TimescaleDB**

概述

本节包含用于保护 PostgreSQL 数据库的最佳实践。

Note:
对于基本设置，请参阅默认的PostgreSQL 数据库创建 说明，其中包括创建对 Zabbix 数据库具有完全权限的'zabbix' 用户。该用户是数据库所有者，在升级 Zabbix 时，也拥有修改数据库结构所需的必要权限。

为提高安全性，建议配置安全的 schema 使用模式，并创建具有最小权限的额外数据库角色和用户。这些角色和用户应根据最小权限原则进行配置，也就是说，它们应仅拥有执行预期功能所必需的权限。

数据库设置

创建将作为数据库所有者的用户，并创建 Zabbix 数据库；数据库所有者是数据库创建时指定的用户：

```
createuser -U postgres -h localhost --pwprompt usr_owner
createdb -U postgres -h localhost -O usr_owner -E Unicode -T template0 zabbix
```

Note:
数据库的全新安装或升级必须由数据库所有者执行。这是因为删除数据库对象或修改其定义的权限，是数据库所有者固有的特权，不能被授予或撤销。

Attention:
本页中的以下命令必须在与 PostgreSQL 的连接明确指向 zabbix 数据库时执行。

创建 zabbix schema，并将数据库所有者（usr_owner）设为该 schema 的所有者：

```
CREATE SCHEMA zabbix AUTHORIZATION usr_owner;
```

配置安全的 schema 使用模式：

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON DATABASE zabbix FROM PUBLIC;
-- 注意：search_path 应指向 "zabbix" schema：
ALTER ROLE ALL IN DATABASE zabbix SET search_path = "zabbix";
```

完成数据库设置后，继续创建用户角色。

创建用户角色

创建以下角色及其对应权限：

- **zbx_srv** - 用于运行 Zabbix 服务器和 proxy 的角色：

```
CREATE ROLE zbx_srv;
GRANT CONNECT ON DATABASE zabbix TO zbx_srv;
GRANT USAGE ON SCHEMA zabbix TO zbx_srv;
ALTER DEFAULT PRIVILEGES FOR ROLE usr_owner IN SCHEMA zabbix GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE
ALTER DEFAULT PRIVILEGES FOR ROLE usr_owner IN SCHEMA zabbix GRANT SELECT, UPDATE, USAGE ON sequences TO z
```

- **zbx_web** - 用于运行 Zabbix 前端和 API 的角色：

```
CREATE ROLE zbx_web;
GRANT CONNECT ON DATABASE zabbix TO zbx_web;
GRANT USAGE ON SCHEMA zabbix TO zbx_web;
ALTER DEFAULT PRIVILEGES FOR ROLE usr_owner IN SCHEMA zabbix GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE
ALTER DEFAULT PRIVILEGES FOR ROLE usr_owner IN SCHEMA zabbix GRANT SELECT, UPDATE, USAGE ON sequences TO z
```

- **zbx_bckp** - 用于表备份的角色：

```
CREATE ROLE zbx_bckp;
GRANT CONNECT ON DATABASE zabbix TO zbx_bckp;
GRANT USAGE ON SCHEMA zabbix TO zbx_bckp;
ALTER DEFAULT PRIVILEGES FOR ROLE usr_owner IN SCHEMA zabbix GRANT SELECT ON TABLES TO zbx_bckp;
ALTER DEFAULT PRIVILEGES FOR ROLE usr_owner IN SCHEMA zabbix GRANT SELECT, UPDATE, USAGE ON sequences TO z
```

Note:
只有数据库所有者才能恢复表。

创建角色后，可以将它们分配给用户。

分配用户角色

要分配已创建的用户角色，请创建用户并为其分配相关角色。将 <user>、<role> 和 <password> 替换为相应的值。确保 <password> 足够强且具有唯一性。

```
CREATE USER <user> WITH ENCRYPTED password '<password>';
GRANT <role> TO <user>;
```

例如，要创建并分配用于运行 Zabbix 服务器和 proxy 的角色：

```
CREATE USER usr_srv WITH ENCRYPTED password 'strong-password';
GRANT zbx_srv TO usr_srv;
```

使用 TimescaleDB 进行数据库分区

TimescaleDB 可用于实现数据库分区。要使用 TimescaleDB，Zabbix 服务器需要具备数据库所有者权限。

如果 PostgreSQL zabbix 模式已经在 zabbix 数据库中创建，则可以使用以下命令启用 TimescaleDB：

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

密码学

概述

本节包含以安全方式设置加密的最佳实践。

为 Zabbix 前端设置 SSL

在基于 RHEL 的系统上，安装 mod_ssl 包：

```
dnf install mod_ssl
```

为 SSL 密钥创建目录：

```
mkdir -p /etc/httpd/ssl/private
chmod 700 /etc/httpd/ssl/private
```

创建 SSL 证书：

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/httpd/ssl/private/apache-selfsigned.key -
```

按提示正确填写。最重要的是请求 Common Name 的那一行。您必须输入要与服务器关联的域名。如果没有域名，也可以输入公网 IP 地址。

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

编辑 Apache SSL 配置文件（/etc/httpd/conf.d/ssl.conf）：

```
DocumentRoot "/usr/share/zabbix"
ServerName example.com:443
SSLCertificateFile /etc/httpd/ssl/apache-selfsigned.crt
SSLCertificateKeyFile /etc/httpd/ssl/private/apache-selfsigned.key
```

重启 Apache 服务以应用更改：

```
systemctl restart httpd.service
```

保护 **web** 服务器

概述

本节包含以安全方式设置 web 服务器的最佳实践。

将根 URL 重定向到 Zabbix SSL

在基于 RHEL 的系统上，向 Apache 配置（/etc/httpd/conf/httpd.conf）中添加一个虚拟主机，并为文档根目录设置到 Zabbix SSL URL 的永久重定向。
请注意，example.com 应替换为服务器的实际名称。

```
### 添加以下行：

<VirtualHost *:*>
    ServerName example.com
    Redirect permanent / https://example.com
</VirtualHost>
```

重启 Apache 服务以应用更改：

```
systemctl restart httpd.service
```

在 web 服务器上启用 HTTP 严格传输安全（HSTS）

为保护 Zabbix 前端免受协议降级攻击，我们建议在 web 服务器上启用 HSTS 策略。

要在 Apache 配置中为您的 Zabbix 前端启用 HSTS 策略，请按以下步骤操作：

1. 找到虚拟主机的配置文件：

- `/etc/httpd/conf/httpd.conf`，适用于基于 RHEL 的系统
- `/etc/apache2/sites-available/000-default.conf`，适用于 Debian/Ubuntu

2. 将以下指令添加到虚拟主机的配置文件中：

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

3. 重启 Apache 服务以应用更改：

```
### 适用于基于 RHEL 的系统：
systemctl restart httpd.service

### 适用于 Debian/Ubuntu
systemctl restart apache2.service
```

在 Zabbix 中强制使用安全和 SameSite 会话 cookie

在配置 Zabbix 时，必须强制会话 cookie 使用 secure 和 SameSite 属性，以增强安全性并防止跨站请求伪造（CSRF）攻击。然而，强制使用 SameSite=Strict 在某些场景下可能会导致问题，例如：

- 仪表盘 URL 小部件在嵌入同域 iframe 时显示“用户未登录”。
- 用户通过 HTTP 而不是 HTTPS 访问仪表盘时，可能会遇到登录问题。
- 无法将 URL 分享到特定的 Zabbix 菜单部分或主机。

为缓解这些问题，用户应当能够调整 SameSite 策略。

1. 安全 cookie

设置 `secure` 标志可确保 cookie 仅通过 HTTPS 传输，从而防止在未加密连接上泄露。

要在 Zabbix 中启用安全 cookie，请在 web 服务器配置中添加或修改以下设置：

对于 Apache：

```
Header always edit Set-Cookie ^(.*)$ $1;Secure
```

对于 Nginx：

```
proxy_cookie_path / "/; Secure";
```

请确保通过 HTTPS 访问您的 Zabbix 前端；否则，带有 `Secure` 标志的 cookie 将不会被发送。

2. 配置 SameSite 属性

web 服务器设置也可以强制使用 SameSite 属性：

对于 Apache：

```
<IfModule mod_headers.c>
    Header onsuccess edit Set-Cookie (.*) "$1; SameSite=Strict"
</IfModule>
```

对于 Nginx（1.19.3+ 版本）：

```
proxy_cookie_flags ~ samesite=Strict; # 将 ~ 替换为 'zbx_session' 以提高特异性
```

在 web 服务器上启用内容安全策略（CSP）

为了保护 Zabbix 前端免受跨站脚本（XSS）、数据注入及类似攻击，建议在 web 服务器上启用内容安全策略（Content Security Policy）。为此，请配置 web 服务器返回 HTTP 标头。

> **Attention:**
> 以下 CSP 标头配置仅适用于默认的 Zabbix 前端安装，以及所有内容都来自站点域名（不包括子域名）的情况。如果您正在配置*URL* 小部件以显示来自站点子域名或外部域名的内容，或者从 *OpenStreetMap* 切换到其他地图引擎，或者添加外部 CSS 或小部件，则可能需要不同的 CSP 标头配置。如果您使用 Duo Universal Prompt 多因素认证 方法，请务必在虚拟主机的配置文件中的 CSP 指令里添加”duo.com”。

要在 Apache 配置中为您的 Zabbix 前端启用 CSP，请按以下步骤操作：

1. 找到虚拟主机的配置文件：

- RHEL 系统：/etc/httpd/conf/httpd.conf
- Debian/Ubuntu：/etc/apache2/sites-available/000-default.conf

2. 在虚拟主机的配置文件中添加以下指令：

```
<VirtualHost *:*>
    Header set Content-Security-Policy: "default-src 'self' *.openstreetmap.org; script-src 'self' 'unsafe
</VirtualHost>
```

3. 重启 Apache 服务以应用更改：

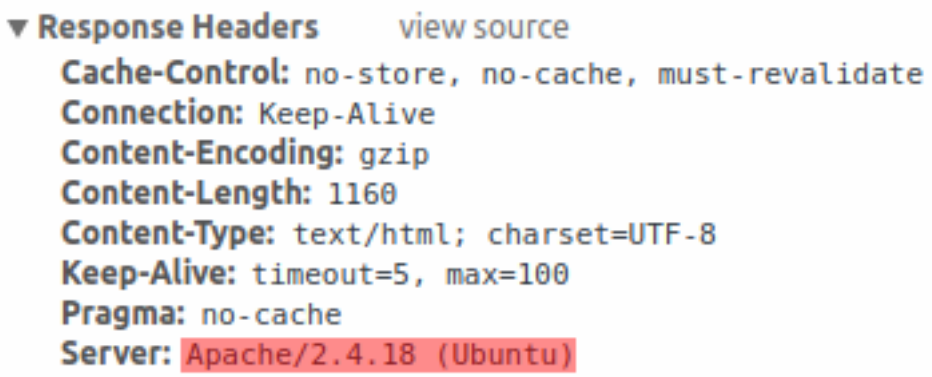
```
### 在 RHEL 系统上：
systemctl restart httpd.service

### 在 Debian/Ubuntu 上
systemctl restart apache2.service
```

禁用 Web 服务器信息暴露

为提高安全性，建议禁用所有 Web 服务器签名。

默认情况下，Web 服务器会暴露软件签名：



可以通过将以下参数添加到 Apache 配置文件中来禁用该签名：

```
ServerSignature Off
ServerTokens Prod
```

可以通过修改 php.ini 配置文件来禁用 PHP 签名（X-Powered-By HTTP 标头；默认情况下，该签名处于禁用状态）：

```
expose_php = Off
```

需要重启 Web 服务器，配置文件的更改才会生效。

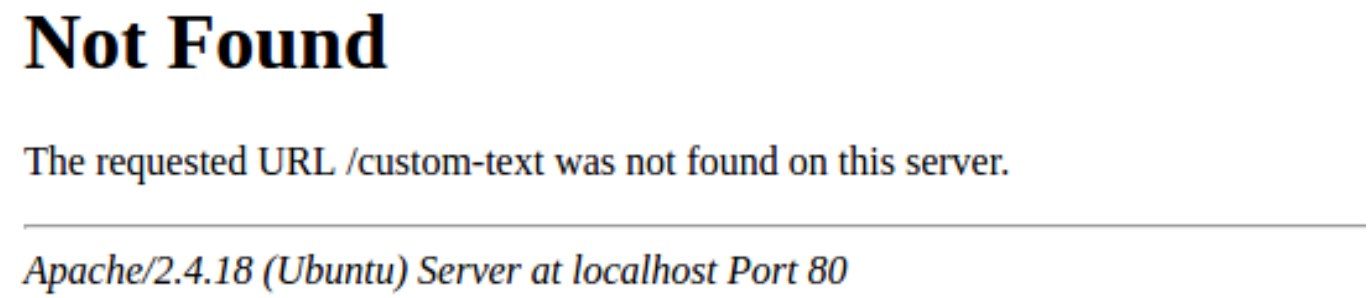
为进一步提高安全性，您可以在 Apache 中使用 *mod_security* 工具（软件包 *libapache2-mod-security2*）。该工具可以移除服务器签名，而不仅仅是移除服务器签名中的版本信息。安装 *mod_security* 后，通过将"SecServerSignature" 设置为所需的任意值，可以将服务器签名更改为任意内容。

请参阅 Web 服务器的文档，以了解如何移除或更改软件签名。

禁用 Web 服务器默认错误页面

为避免信息泄露，建议禁用默认错误页面。

默认情况下，Web 服务器使用内置错误页面：



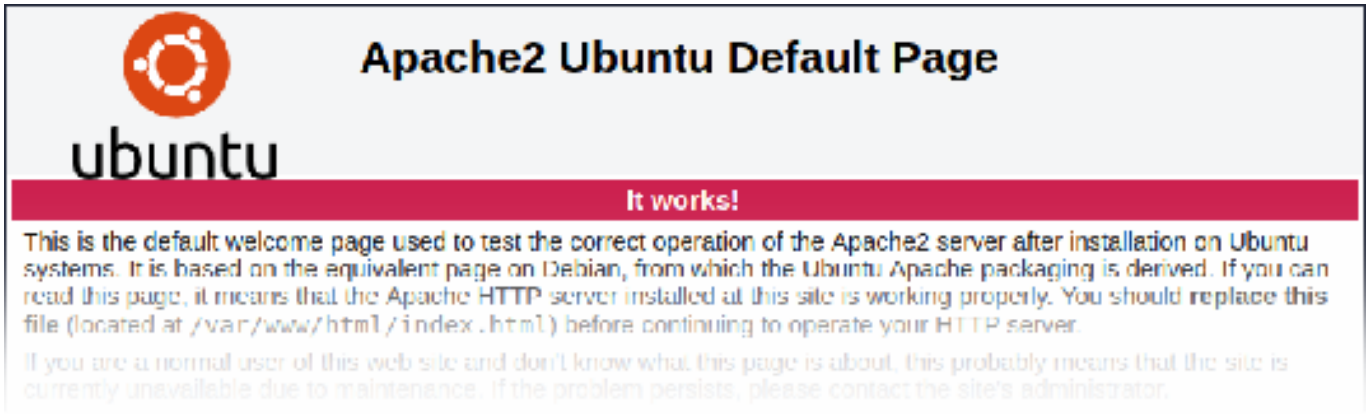
应替换或删除这些默认错误页面。例如，可以使用 “ErrorDocument” 指令为 Apache Web 服务器定义自定义错误页面或文本。

请参阅 Web 服务器的文档，以了解如何替换或删除默认错误页面。

移除 Web 服务器测试页面

为避免信息泄露，建议移除 Web 服务器测试页面。

默认情况下，Apache Web 服务器的 webroot 中包含 `index.html` 测试页面：



请参阅 Web 服务器的文档，以获取有关如何移除默认测试页面的帮助。

设置 X-Frame-Options HTTP 响应头

默认情况下，Zabbix 的 *Use X-Frame-Options HTTP header* 参数设置为 `SAMEORIGIN`。这意味着内容只能加载到与页面本身具有相同来源的框架中。

从外部 URL 拉取内容的 Zabbix 前端元素（即 URL dashboard widget）会在启用所有沙箱限制的 sandbox 中显示检索到的内容。

这些设置增强了 Zabbix 前端的安全性，并可防御 XSS 和点击劫持攻击。*Super admin* 用户可以根据需要修改 *Use iframe sandboxing* 和 *Use X-Frame-Options HTTP header* 参数。在更改默认设置之前，请仔细权衡风险和收益。不建议完全关闭 iframe sandboxing 或 X-Frame-Options HTTP header。

隐藏常见密码列表文件

为增加密码暴力破解攻击的复杂度，建议限制对 `ui/data/top_passwords.txt` 文件的访问。该文件包含最常见以及与上下文相关的密码列表，并可防止用户设置此类密码（如果在密码策略 中启用了 避免使用容易猜测的密码参数）。

要限制对 `top_passwords.txt` 文件的访问，请修改您的 web 服务器配置。

在 Apache 上，可以使用 `.htaccess` 文件限制文件访问：

```
<Files "top_passwords.txt">
    Order Allow,Deny
    Deny from all
</Files>
```

在 NGINX 上，可以使用 `location` 指令限制文件访问：

```
location = /data/top_passwords.txt {
    deny all;
    return 404;
}
```

概述　Zabbix 可以配置为从安全 vault 中检索敏感信息。支持以下密钥管理服务：HashiCorp Vault KV Secrets Engine - Version 2、CyberArk Vault CV12。

密钥可用于检索：

- 用户宏值
- 数据库访问凭据

Zabbix 提供对 vault 中密钥的只读访问，前提是这些密钥由其他人进行管理。

有关特定 vault provider 配置的信息，请参见：

- HashiCorp configuration
- CyberArk configuration

机密值的缓存　默认情况下，vault 机密宏值由 Zabbix 服务器在每次刷新配置数据时获取，然后存储在配置缓存中。Zabbix proxy 在每次配置同步时从 Zabbix 服务器接收 vault 机密宏的值，并将其存储在自身的配置缓存中。

Attention:
必须启用 Zabbix 服务器与 proxy 之间的加密；否则，系统会记录服务器警告消息。

也可以配置，使 Zabbix 服务器和 Zabbix proxy 分别获取宏值。

要手动触发从 vault 刷新缓存的机密值，请使用 `secrets_reload` 命令行选项。

Zabbix 前端默认禁用数据库凭据缓存，但可以在前端配置文件中将选项 `$DB['VAULT_CACHE'] = true` 设置为启用该功能。凭据将使用文件系统临时文件目录存储在本地缓存中。Web 服务器必须允许写入私有临时文件夹（例如，对于 Apache，必须设置配置选项 `PrivateTmp=True`）。要控制数据缓存刷新/失效的频率，请使用 ZBX_DATA_CACHE_TTL常量。

TLS 配置　要为 Zabbix 组件与 vault 之间的通信配置 TLS，请将由证书颁发机构（CA）签名的证书添加到系统范围的默认 CA 存储中。如需使用其他位置，请在 Zabbix 服务器/proxy 配置参数 SSLCALocation 中指定目录，
将证书文件放入该目录，然后运行 CLI 命令：

```
c_rehash .
```

CyberArk 配置

本节说明如何配置 Zabbix，以便从 CyberArk Vault CV12 检索密钥。

应按照官方 CyberArk 文档 中的说明安装并配置该 vault。

有关在 Zabbix 中配置 TLS 的信息，请参见密钥存储。

数据库凭据

需要为每个 Zabbix 组件单独配置对包含数据库凭据的机密的访问权限。

服务器和 proxy

要从保管库中获取 Zabbix 服务器或proxy 的数据库凭据，请在配置文件中指定以下配置参数：

- `Vault` - 应使用哪个保管库提供程序；
- `VaultURL` - 保管库服务器的 HTTP(S) URL；
- `VaultDBPath` - 包含数据库凭据的保管库机密的查询路径，系统将通过键”Content” 和”UserName” 获取凭据（仅当未指定 DBUser 和 DBPassword 时，才能使用此选项）；
- `VaultTLSCertFile`、`VaultTLSKeyFile` - SSL 证书和密钥文件名；设置这些选项不是强制性的，但强烈建议进行设置；
- `VaultPrefix` - 根据保管库的不同，用于保管库路径或查询的自定义前缀；如果未指定，将使用最合适的默认值。

Attention:
`Vault`、`VaultURL`、`VaultTLSCertFile`、`VaultTLSKeyFile` 和 `VaultPrefix` 配置参数还用于 Zabbix 服务器处理保管库机密宏时的保管库身份验证（如果已配置，Zabbix proxy 也会使用这些参数）。Zabbix 服务器和 proxy 不会打开包含来自 VaultDBPath 的数据库凭据的保管库机密宏。

Zabbix 服务器和 Zabbix proxy 会在启动时从 `zabbix_server.conf` 和 `zabbix_proxy.conf` 文件中读取与保管库相关的配置参数。

示例

1. 在 zabbix_server.conf 中，指定以下参数：

```
Vault=CyberArk
VaultURL=https://127.0.0.1:1858
VaultDBPath=AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_database
VaultTLSCertFile=cert.pem
VaultTLSKeyFile=key.pem
VaultPrefix=/AIMWebService/api/Accounts?
```

2. Zabbix 将向 vault 发送以下 API 请求：

```
curl \
--header "Content-Type: application/json" \
--cert cert.pem \
--key key.pem \
https://127.0.0.1:1858/AIMWebService/api/Accounts?AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbi
```

3. vault 响应将包含键”Content” 和”UserName”：

```
{
    "Content": <password>,
    "UserName": <username>,
    "Address": <address>,
    "Database": <Database>,
    "PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

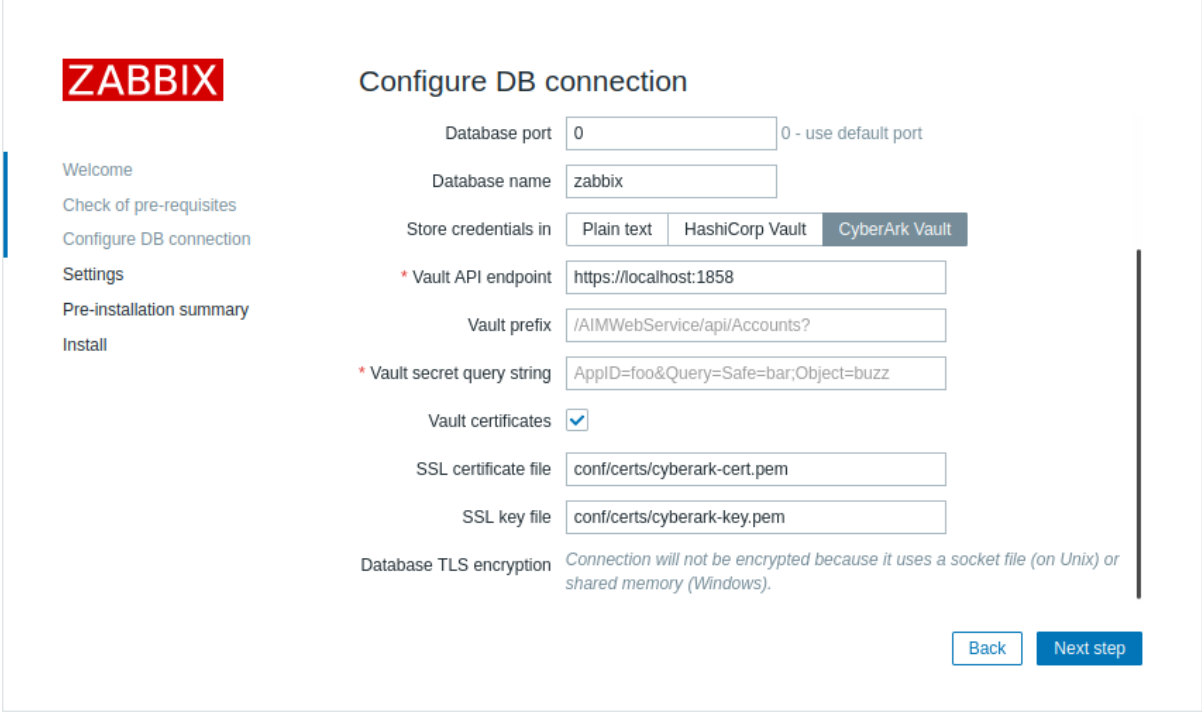
4. 因此，Zabbix 将使用以下凭据进行数据库身份验证：

- 用户名: <username>
- 密码: <password>

前端

要从 vault 获取 Zabbix 前端的数据库凭据，请在前端安装期间指定以下参数。

1. 在 配置数据库连接步骤中，将 凭据存储于参数设置为 *CyberArk Vault*。



2. 然后，填写其他参数：

参数	必填	默认值	描述
Vault API 端点	是	https://localhost:1858	指定用于连接 vault 的 URL，格式为 `scheme://host:port`
Vault 前缀	否	/AIMWebService/api/Accounts	为 vault 路径或查询提供自定义前缀。如果未指定，则使用默认值。
Vault 密钥查询字符串	是		指定应从何处检索数据库凭据的查询。 示例： `AppID=foo&Query=Safe=bar;Object=buzz`
Vault 证书	否		选中复选框后，将显示其他参数，允许您配置客户端身份验证。虽然此参数是可选的，但强烈建议启用它，以便与 CyberArk Vault 通信。
SSL 证书文件	否	conf/certs/cyberark-cert.pem	SSL 证书文件的路径。该文件必须采用 PEM 格式。 如果证书文件还包含私钥，请将 SSL 密钥文件参数留空。
SSL 密钥文件	否	conf/certs/cyberark-key.pem	用于客户端身份验证的 SSL 私钥文件名称。该文件必须采用 PEM 格式。

用户宏值

要使用 CyberArk Vault 存储 *Vault secret* 用户宏值，请确保：

- Zabbix 服务器已配置为可与 CyberArk Vault 协同工作；
- *Administration > General > Other* 中的 *Vault provider* 参数设置为"CyberArk Vault"。



Note:
Zabbix 服务器（以及 Zabbix proxy，如果已配置）需要能够从 vault 访问 *Vault secret* 宏值。Zabbix 前端不需要此类访问权限。

宏值应包含一个查询（格式为 `query:key`）。

有关 Zabbix 如何处理宏值的详细信息，请参见*Vault secret macros*。

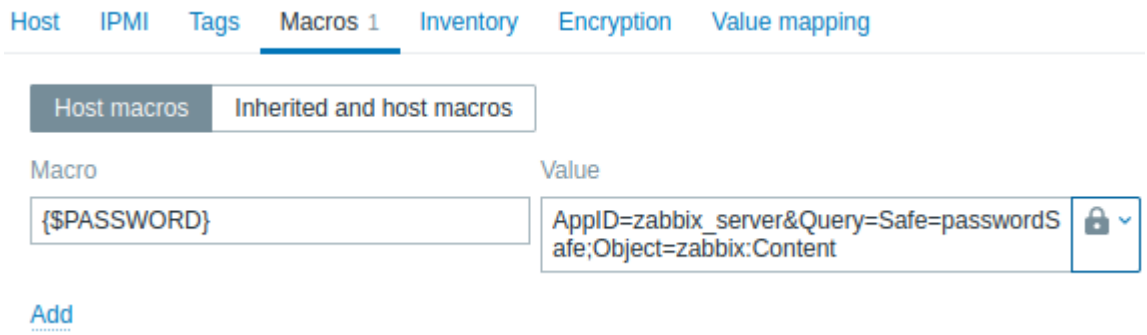
查询语法

冒号符号（`:`）保留用于分隔查询和键。

如果查询本身包含正斜杠或冒号，则应对这些符号进行 URL 编码（`/` 编码为 `%2F`，`:` 编码为 `%3A`）。

示例

1. 在 Zabbix 中，添加一个类型为 *Vault secret* 的用户宏 `{$PASSWORD}`，并将其值设置为 `AppID=zabbix_server&Query=Safe=passwor`



2. Zabbix 将向 vault 发送以下 API 请求：

```
curl \
--header "Content-Type: application/json" \
--cert cert.pem \
--key key.pem \
https://127.0.0.1:1858/AIMWebService/api/Accounts?AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbi
```

3. vault 响应将包含键 `Content`：

```
{
    "Content": <password>,
    "UserName": <username>,
    "Address": <address>,
    "Database" :<Database>,
    "PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

4. 因此，Zabbix 会将宏 {$PASSWORD} 解析为值 - <password>

更新现有配置

要更新用于从 CyberArk Vault 获取机密的现有配置：

1. 按照数据库凭据部分中的说明，更新 Zabbix 服务器或 proxy 配置文件参数。
2. 更新 Zabbix 前端配置文件参数：

```
$DB['VAULT']                    = 'CyberArk';
$DB['VAULT_URL']                = 'https://127.0.0.1:1858';
$DB['VAULT_DB_PATH']            = 'AppID=foo&Query=Safe=bar;Object=buzz';
$DB['VAULT_TOKEN']              = '';
$DB['VAULT_CERT_FILE']          = 'conf/certs/cyberark-cert.pem';
$DB['VAULT_KEY_FILE']           = 'conf/certs/cyberark-key.pem';
$DB['VAULT_PREFIX']             = '';
```

3. 如有必要，按照用户宏值部分中的说明配置用户宏。

要更新用于从 HashiCorp Vault 获取机密的现有配置，请参阅*HashiCorp 配置*。

HashiCorp 配置

概述

本节说明如何配置 Zabbix，以便从 HashiCorp Vault KV Secrets Engine - Version 2 检索密钥。

应按照官方 HashiCorp 文档 中的说明部署并配置 vault。

要了解如何在 Zabbix 中配置 TLS，请参见密钥存储。

获取数据库凭据

要成功获取包含数据库凭据的机密，必须同时配置以下两项：

- Zabbix 服务器/proxy
- Zabbix 前端

服务器/proxy

要配置 Zabbix 服务器或proxy，请在配置文件中指定以下配置参数：

- `Vault` - 应使用的 vault 提供程序。
- `VaultToken` - vault 身份验证令牌（详情请参见 Zabbix 服务器/proxy 配置文件）；不能与 `VaultAppRoleID` 一起使用。
- `VaultAppRoleID` 和 `VaultAppSecretID` - vault 身份验证角色 ID 和机密 ID（详情请参见 Zabbix 服务器/proxy 配置文件）；不能与 `VaultToken` 一起使用。
- `VaultURL` - vault 服务器 HTTP(S) URL。
- `VaultDBPath` - 包含数据库凭据的 vault 机密路径（仅当未指定 DBUser 和 DBPassword 时才能使用此选项）。Zabbix 服务器或 proxy 将通过键”password” 和”username” 获取凭据。
- `VaultPrefix` - vault 路径或查询所使用的自定义前缀，具体取决于 vault；如果未指定，将使用最合适的默认值。请注意，前缀必须以 / 开头和结尾。

> **Attention:**
> `Vault`、`VaultToken`（或 `VaultAppRoleID`/`VaultAppSecretID`）、`VaultURL` 和 `VaultPrefix` 配置参数还用于 Zabbix 服务器在处理 vault 机密宏时进行 vault 身份验证（如果已配置，Zabbix proxy 也会使用这些参数）。Zabbix 服务器和 proxy 不会打开包含来自 VaultDBPath 的数据库凭据的 vault 机密宏。

 强烈建议为不同的 proxy 使用不同的令牌。

Zabbix 服务器和 Zabbix proxy 在启动时从 `zabbix_server.conf` 和 `zabbix_proxy.conf` 读取与 vault 相关的配置参数。

此外，Zabbix 服务器和 Zabbix proxy 会在启动期间读取一次 `VAULT_TOKEN` 环境变量，并将其取消设置，以使其无法通过派生脚本访问；如果 `VaultToken` 和 `VAULT_TOKEN` 参数同时包含值，则会报错。

示例

1. 在 zabbix_server.conf 中，指定以下参数。

使用 VaultToken：

```
Vault=HashiCorp
VaultToken=hvs.CAESIIG_PILmULFYOsEyWHxkZ2mF2a8VPKNLE8eHqd4autYGGh4KHGh2cy5aeTYONFNSaUp3ZnpWbDF1RUNjUkNTZEg
VaultURL=https://127.0.0.1:8200
VaultDBPath=database
VaultPrefix=/v1/secret/data/zabbix/
```

或者，使用 VaultAppRoleID：

```
Vault=HashiCorp
VaultAppRoleID=6f1567f1-538d-7d70-e8e4-5d3e59471b2e
VaultAppSecretID=89dac8b1-f7d6-96a7-2ecf-f94d2de3d6f1
VaultURL=https://127.0.0.1:8200
VaultDBPath=database
VaultPrefix=/v1/secret/data/zabbix/
```

2. 运行以下 CLI 命令，在 vault 中创建所需的 secret：

```
#### 如果尚未启用，请启用 "secret/" 挂载点；注意必须使用 "kv-v2"。
vault secrets enable -path=secret/ kv-v2

#### 在挂载点 "secret/" 和路径 "zabbix/database" 下，使用键 username 和 password 写入新的 secret。
vault kv put -mount=secret zabbix/database username=zabbix password=<password>

#### 测试 secret 是否已成功添加。
vault kv get secret/zabbix/database

#### 最后使用 Curl 进行测试；注意需要在挂载点后手动添加 "data"，并在挂载点前添加 "/v1"，另请参见 --capath 参数。
curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix/database
```

3. 结果是，Zabbix 服务器将检索以下用于数据库认证的凭据：

- 用户名: zabbix
- 密码: <password>

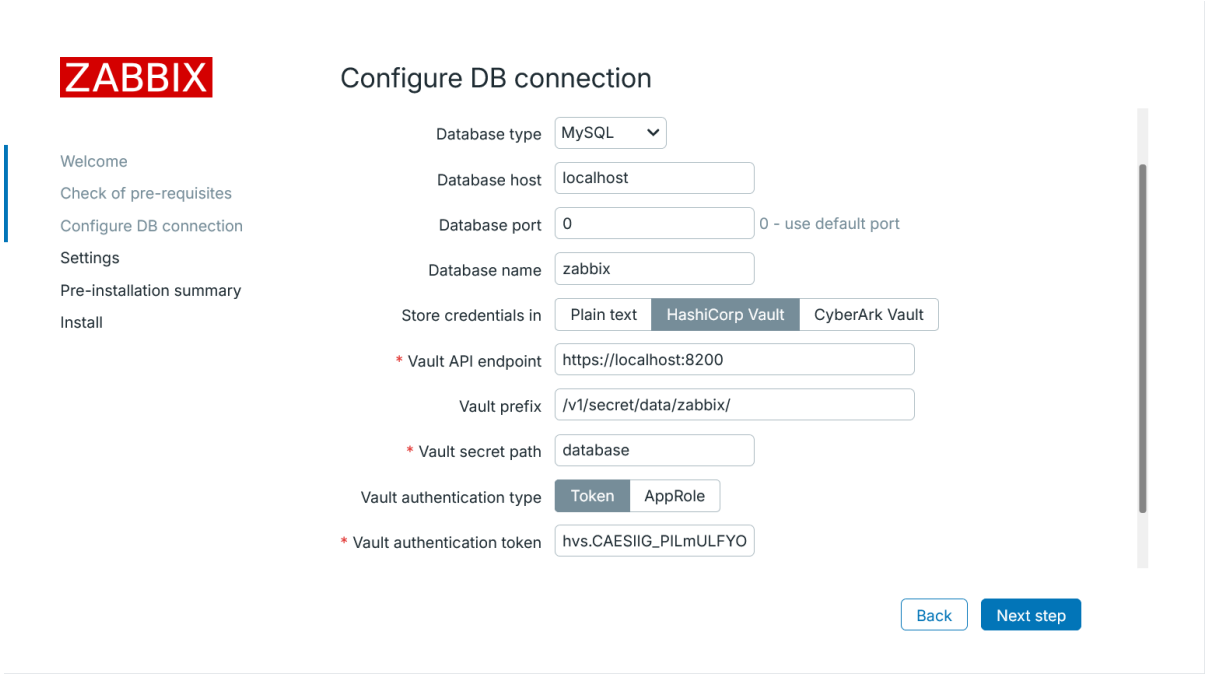
前端

Zabbix 前端可以配置为在安装前端期间，或通过更新前端配置文件（zabbix.conf.php），从 vault 获取数据库凭据。

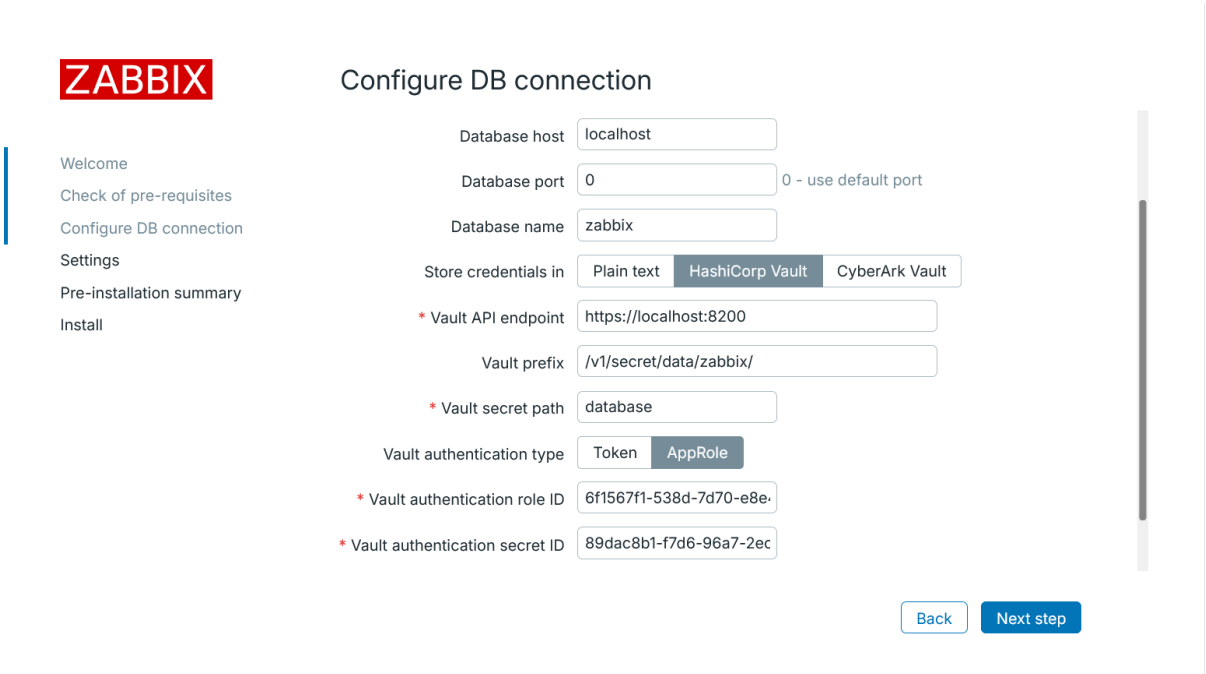
> **Attention:**
> 如果 vault 凭据自上次前端安装后发生了更改，请更新现有配置。

在前端安装期间，必须在配置数据库连接步骤中指定连接参数。

使用令牌身份验证的配置示例：



使用 AppRole 身份验证的配置示例：



参数	必填	默认值	描述
凭据存储于	否		选择 **HashiCorp Vault**。
Vault API 端点	是	https://localhost:8200	指定用于连接 vault 的 URL，格式为 `scheme://host:port`
Vault 前缀	否	/v1/secret/data/	为 vault 路径或查询提供自定义前缀。如果未指定，则使用默认值。 示例：`/v1/secret/data/zabbix/`
Vault 密钥路径	是		存储密钥的路径，系统将通过键”password” 和”username” 从中获取数据库凭据。 示例：`database`
Vault 身份验证类型	否		选择 vault 身份验证类型： **Token** - 使用令牌身份验证； **AppRole** - 使用 AppRole 身份验证方法。
Vault 身份验证令牌	是		提供对密钥路径具有只读访问权限的身份验证令牌。 有关创建令牌和 vault 策略的信息，请参阅 HashiCorp 文档。 仅当选择 *Token* 作为 vault 身份验证类型时，此字段才可用。
Vault 身份验证角色 *ID*	是		提供 AppRole 的角色 ID。 有关 AppRole 的信息，请参阅 HashiCorp 文档。 仅当选择 *AppRole* 作为 vault 身份验证类型时，此字段才可用。

参数	必填	默认值	描述
Vault 身份验证密钥 *ID*	是		提供针对 AppRole 签发的密钥 ID。 有关 AppRole 的信息，请参阅 HashiCorp 文档。 仅当选择 *AppRole* 作为 vault 身份验证类型时，此字段才可用。

获取用户宏值

要使用 HashiCorp Vault 存储 *Vault secret* 用户宏值，请确保：

- Zabbix 服务器/proxy 已配置为可与 HashiCorp Vault 配合工作。
- *Administration > General > Other* 中的 *Vault provider* 参数已设置为”HashiCorp Vault”（默认）。



Note:
Zabbix 服务器（以及 Zabbix proxy，如果已配置）需要能够从 vault 访问 *Vault secret* 宏值。Zabbix 前端不需要此类访问权限。

宏值应包含一个引用路径（格式为 `path:key`，例如 `macros:password`）。在 Zabbix 服务器/proxy 配置期间指定的认证令牌（通过 `VaultToken` 参数）必须对该路径提供只读访问权限。

有关 Zabbix 如何处理宏值的详细信息，请参见*Vault secret macros*。

路径语法

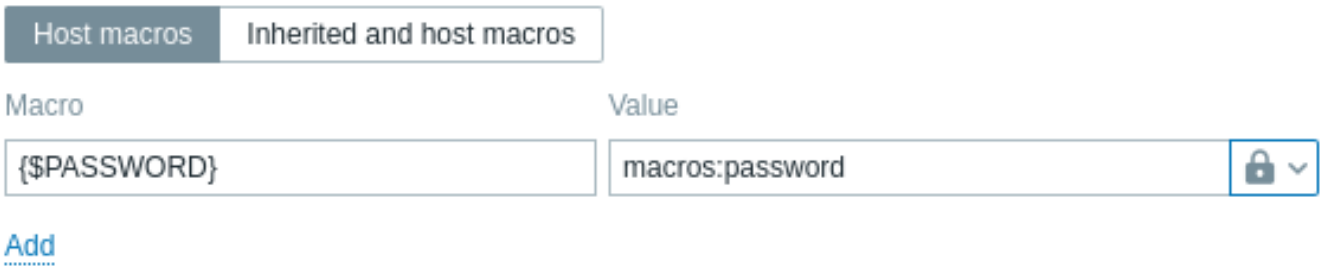
正斜杠（”/”）和冒号（”:”）是保留字符。

正斜杠只能用于分隔挂载点和路径（例如，*secret/zabbix*，其中挂载点为”secret”，路径为”zabbix”）。对于 Vault 宏，冒号只能用于分隔路径/查询和键。

如果需要创建一个名称中包含正斜杠分隔符的挂载点（例如，*foo/bar/zabbix*，其中挂载点为”foo/bar”，路径为”zabbix”，可以编码为”foo%2Fbar/zabbix”），或者挂载点名称或路径中需要包含冒号，则可以对正斜杠和冒号进行 URL 编码。

示例

1. 在 Zabbix 中，添加一个类型为”Vault secret” 的用户宏 `{$PASSWORD}`，其值为 `macros:password`



2. 运行以下 CLI 命令，在 vault 中创建所需的 secret：

```
#### 如果尚未启用，请启用 "secret/" 挂载点；注意必须使用 "kv-v2"。
vault secrets enable -path=secret/ kv-v2

#### 在挂载点 "secret/" 和路径 "zabbix/macros" 下，使用键 "password" 放入新的 secret。
vault kv put -mount=secret zabbix/macros password=<password>

#### 测试 secret 是否已成功添加。
vault kv get secret/zabbix/macros

#### 最后使用 Curl 测试；注意需要在挂载点后手动添加 "data"，并在挂载点前添加 "/v1"，另请参见 --capath 参数。
curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix/macros
```

3. 结果是，Zabbix 将把宏 {$PASSWORD} 解析为以下值：<password>

更新现有配置

要更新用于从 HashiCorp Vault 获取机密的现有配置：

1. 按照数据库凭据部分中的说明，更新 Zabbix 服务器或 proxy 配置文件参数。
2. 更新 Zabbix 前端配置文件参数：

使用 Vault 令牌：

```
$DB['VAULT']                    = 'HashiCorp';
$DB['VAULT_URL']                = 'https://localhost:8200';
$DB['VAULT_DB_PATH']            = 'database';
$DB['VAULT_TOKEN']              = 'hvs.CAESIIG_PILmULFYOsEyWHxkZ2mF2a8VPKNLE8eHqd4autYGGh4KHGh2cy5aeTY0NFN
$DB['VAULT_PREFIX']             = '/v1/secret/data/zabbix/';
```

或者，使用 Vault AppRole 方法：

```
$DB['VAULT']                    = 'HashiCorp';
$DB['VAULT_URL']                = 'https://localhost:8200';
$DB['VAULT_DB_PATH']            = 'database';
$DB['VAULT_APP_ROLE_ID']        = '6f1567f1-538d-7d70-e8e4-5d3e59471b2e';
$DB['VAULT_APP_SECRET_ID']      = '89dac8b1-f7d6-96a7-2ecf-f94d2de3d6f1';
$DB['VAULT_PREFIX']             = '/v1/secret/data/zabbix/';
```

3. 如有必要，按照用户宏值部分中的说明配置用户宏。

要更新用于从 CyberArk Vault 获取机密的现有配置，请参阅*CyberArk* 配置。

故障排除

1. 在 Zabbix 服务器/proxy 日志文件中：

```
237839:20260729:130916.425 Vault token renew started to fail: cannot login into HashiCorp vault with AppRo
```

可能原因：HashiCorp Vault 已经（重新）启动，但尚未解封（在默认配置中需要解封；如果配置了 Auto Unseal，则不需要）。

2. 在 Zabbix 服务器/proxy 日志文件中：

```
235545:20260729:125617.944 Vault token renew started to fail: cannot login into HashiCorp vault with AppRo
```

或

```
235545:20260729:125718.537 Vault token renew failed: cannot login into HashiCorp vault with AppRole method
```

可能原因：VaultAppSecretID 已在 HashiCorp vault 中过期，或已达到其使用次数上限。

Google Chrome TLS 证书信任

概述

本页提供了使用 Google Chrome 访问 Zabbix 前端或受自签名证书或私有证书颁发机构保护的 zabbix-web-service 端点时的 Zabbix 安装步骤和配置示例。

这些说明假定目标 web 服务器已配置为 HTTPS。有关为 Zabbix 前端配置 TLS 的信息，请参见Secure connection to the frontend。

配置

Linux 上的 Google Chrome 使用按用户划分的 NSS 证书数据库来存储受信任的证书。
要让 Chrome 信任自签名证书或私有证书颁发机构，请将所需证书添加到运行 Chrome 的用户账户的 NSS 数据库中。

1. 安装所需软件包。

对于 Debian/Ubuntu：

```
sudo apt install ca-certificates libnss3-tools
```

对于基于 RHEL 的系统：

```
sudo dnf install ca-certificates nss-tools
```

或者：

```
sudo yum install ca-certificates nss-tools
```

2. 准备证书文件。

对于每个所需证书：为自签名证书创建一个基于 PEM 的 .crt 文件，或者分别为根证书颁发机构证书以及（如需要）每个中间证书创建单独的基于 PEM 的 .crt 文件。

3. 创建 NSS 数据库目录。

使用运行 Chrome 的用户的 home 目录。

例如：

```
sudo mkdir -p /var/lib/zabbix/.pki/nssdb
sudo -u zabbix certutil -N -d sql:/var/lib/zabbix/.pki/nssdb
sudo chown -R zabbix:zabbix /var/lib/zabbix/.pki/nssdb
```

如果目标系统上的 Chrome 使用不同的 NSS 数据库位置，请改用该目录。

请确保证书文件对运行 Google Chrome 的用户可读。
如有必要，请将其复制到该用户可访问的位置。

4. 导入证书。

如果使用自签名证书：

```
sudo -u zabbix certutil -d sql:/var/lib/zabbix/.pki/nssdb \
  -A -t "P,," \
  -n "Zabbix self-signed certificate" \
  -i /path/to/self-signed.crt
```

如果使用私有证书颁发机构，请将根证书颁发机构证书导入数据库：

```
sudo -u zabbix certutil -d sql:/var/lib/zabbix/.pki/nssdb \
  -A -t "C,," \
  -n "Zabbix root certificate authority" \
  -i /path/to/root-ca.crt
```

如果使用了任何中间证书，请分别导入每个证书：

```
sudo -u zabbix certutil -d sql:/var/lib/zabbix/.pki/nssdb \
  -A -t ",," \
  -n "Zabbix intermediate certificate" \
  -i /path/to/intermediate-ca.crt
```

5. 验证证书数据库。

```
sudo -u zabbix certutil -d sql:/var/lib/zabbix/.pki/nssdb -L
```

6. 测试 Chrome。

Google Chrome 使用启动它的用户账户的证书数据库。
如果该用户账户没有可写的 home 目录，请在运行测试前配置可写的 home 目录和 XDG 目录。

示例（如果使用 Chromium，请将 google-chrome 替换为 chromium）：

```
sudo -u zabbix env \
  HOME=/var/lib/zabbix-home \
  XDG_CONFIG_HOME=/var/lib/zabbix-home/.config \
  XDG_DATA_HOME=/var/lib/zabbix-home/.local/share \
  XDG_CACHE_HOME=/var/lib/zabbix-home/.cache \
  google-chrome --headless --disable-gpu \
  --user-data-dir=/var/lib/zabbix-home/chrome-profile \
  --dump-dom https://FQDN_OF_ZABBIX_SERVER
```

如果配置正确，该命令会返回所请求页面的 HTML。

如果 Chrome 不信任该证书，该命令会返回一个证书错误页面，其中包含诸如 NET::ERR_CERT_AUTHORITY_INVALID 之类的消息。

故障排除

错误	可能原因
NET::ERR_CERT_AUTHORITY_INVALID	证书被导入到了错误的用户配置文件中，缺少根证书颁发机构证书，或者缺少中间证书。
NET::ERR_CERT_COMMON_NAME_INVALID	证书与浏览器 URL 中使用的主机名不匹配。
NET::ERR_CERT_DATE_INVALID	证书已过期、尚未生效，或者系统时钟不正确。
导入后 Chrome 仍显示证书警告	Chrome 正在为当前用户使用不同的 NSS 数据库，或者证书信任标志不正确。

安全最佳实践

概述　本节介绍以安全方式设置 Zabbix 的最佳实践。

本节中的实践并非 Zabbix 正常运行所必需，但建议采用，以提高系统安全性。

有关确保 Zabbix 移动应用和 Zabbix bridge adapter 安全的最佳实践，请参阅安全的移动访问。

UTF-8 编码　Zabbix 仅支持 UTF-8 编码。已知它可以正常工作，且不存在任何安全缺陷。用户应注意，如果使用其他某些编码，已知会存在安全问题。

Windows 安装程序路径　在使用 Windows 安装程序时，建议使用安装程序提供的默认路径。如果在没有适当权限的情况下使用自定义路径，可能会危及安装的安全性。

用户自定义全局脚本中的宏　为增强安全性，建议在用户自定义全局脚本中使用宏函数而不是普通宏，因为宏不会自动转义。

Zabbix 安全公告和 **CVE** 数据库　请参阅 *Zabbix* 安全公告和 *CVE* 数据库。

HTML 电子邮件模板　创建或编辑用于 HTML 电子邮件的消息模板时，始终使用 `htmlencode()` 宏函数包装每个宏。例如：

```
<b>Problem started</b> at {{EVENT.TIME}.htmlencode()} on {{EVENT.DATE}.htmlencode()}<br><b>Problem name:</
{{EVENT.NAME}.htmlencode()}<br><b>Host:</b> {{HOST.NAME}.htmlencode()}
<br><b>Severity:</b>
{{EVENT.SEVERITY}.htmlencode()}<br><b>Operational
data:</b> {{EVENT.OPDATA}.htmlencode()}<br><b>Original problem ID:</b> {{EVENT.ID}.htmlencode()}<br>{{TRIG
```

使用 `htmlencode()` 可确保宏值中的任何 HTML 字符都经过转义，并防止将 HTML 注入通知中（例如，攻击者在通知中插入恶意链接或钓鱼链接）。

注意：Zabbix 提供的默认 HTML 电子邮件消息已经对宏应用了 `htmlencode()`。在编辑现有模板或创建新模板时，都应遵循此指导原则——在使用模板发送 HTML 电子邮件之前，请确认宏已进行编码。

Web 界面

概述　为了能够从任何位置、任何平台轻松访问 Zabbix，Zabbix 提供了基于 Web 的前端。

> **Note:**
> 如果使用多个前端实例，请确保所有前端都以相同方式安装和配置了语言环境及库（LDAP、SAML 等）。

前端帮助　Zabbix 前端各部分和表单的右上角提供了帮助链接，可直接链接到文档中的相应部分。

对于带有帮助图标的表单字段，将鼠标悬停在图标上即可显示相应提示。

信息亭模式　对于带有信息亭模式按钮的前端部分（该按钮位于右上角），点击该按钮可将此部分切换到信息亭模式。在信息亭模式下，仅显示此部分的内容。

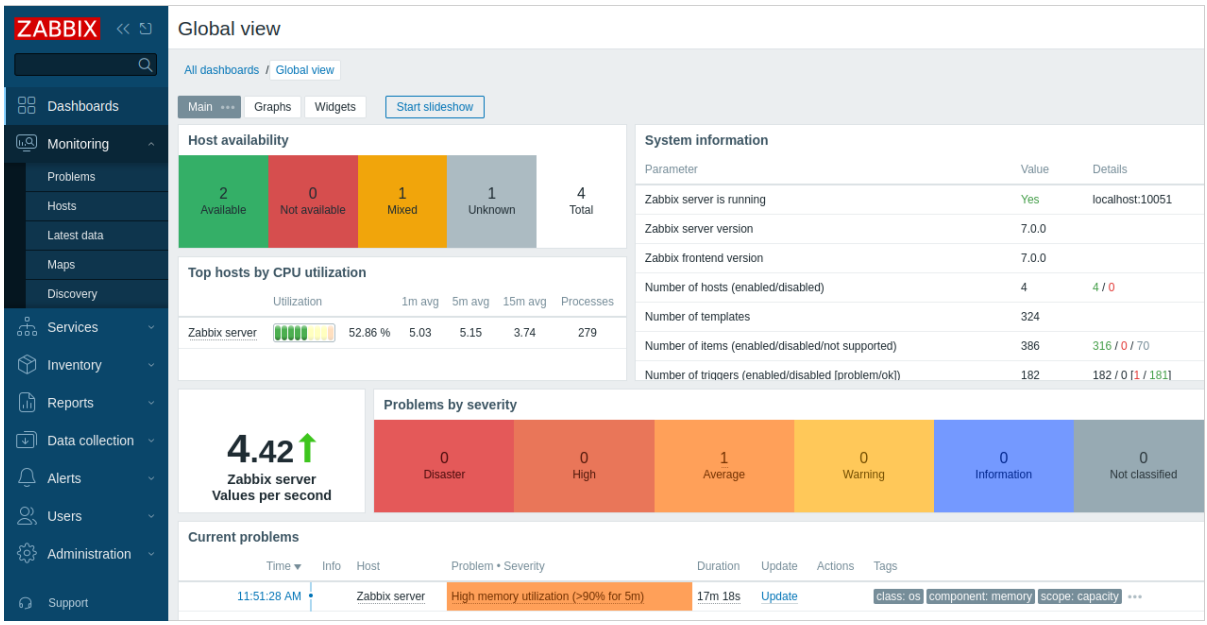
要退出信息亭模式，将鼠标悬停在内容上，然后点击右上角出现的普通视图按钮。

菜单

概述

侧边栏中的垂直菜单可用于访问 Zabbix 前端的各个部分。

在默认主题中，该菜单为深蓝色。

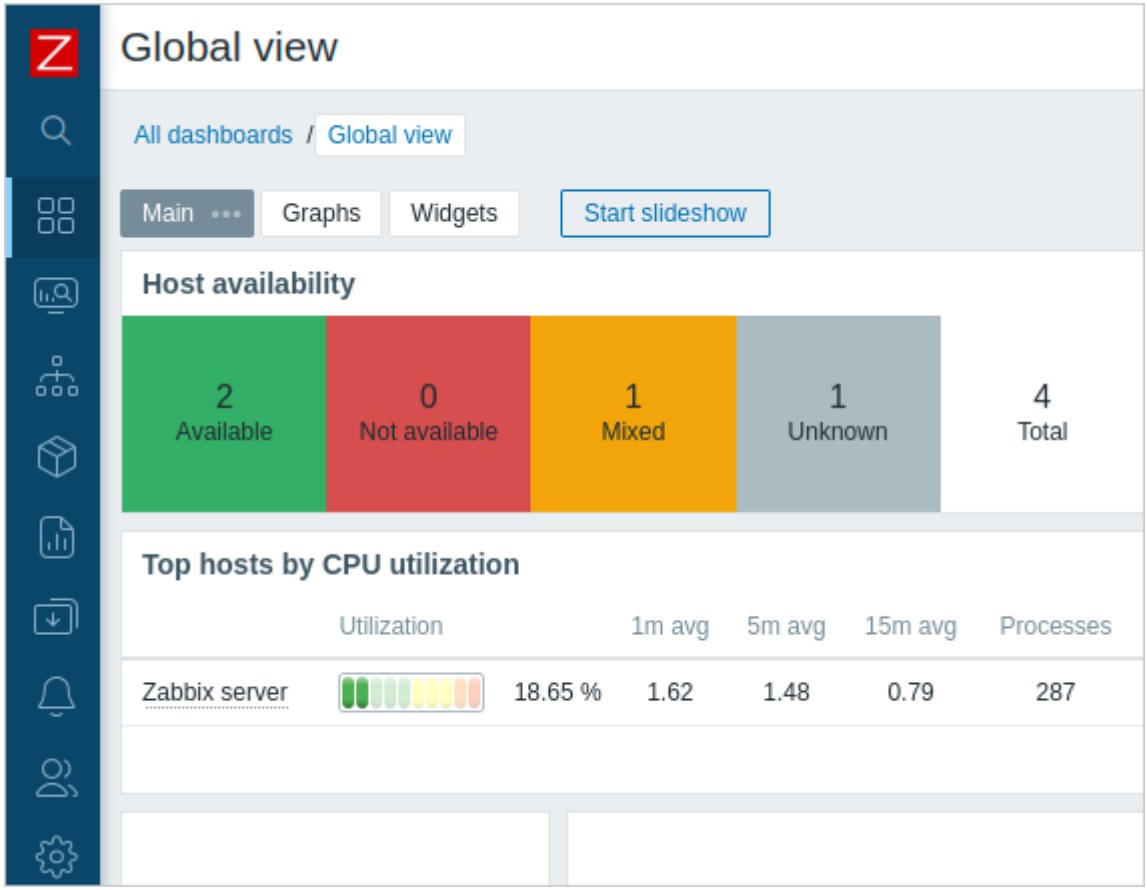


使用菜单

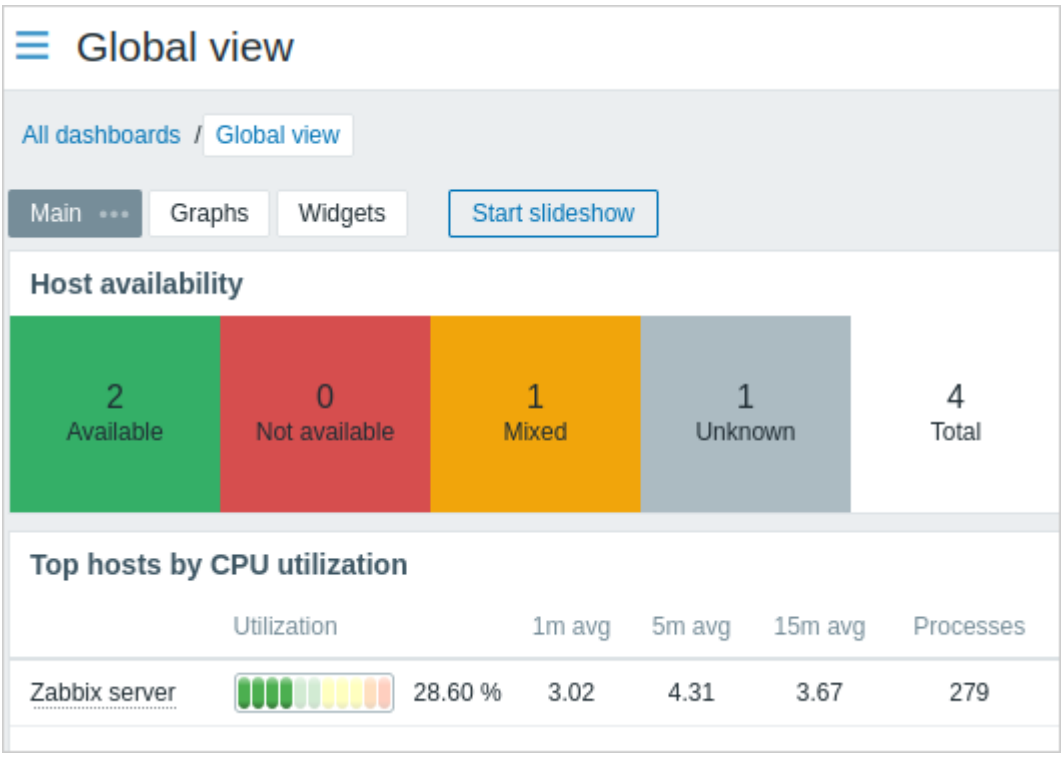
全局搜索框位于 Zabbix 标志下方。

菜单可以折叠或完全隐藏：

- 要折叠，请单击  Zabbix 标志旁边的按钮。在折叠后的菜单中，仅显示图标。



- 要隐藏，请单击  Zabbix 标志旁边的按钮。在隐藏后的菜单中，所有内容都会被隐藏。



折叠菜单

当菜单折叠为仅显示图标时，只要将鼠标光标移到菜单上，完整菜单就会重新显示。请注意，重新显示的菜单会覆盖页面内容；如需将页面内容向右移动，必须点击展开按钮。如果鼠标光标再次移出完整菜单，菜单将在两秒后再次折叠。

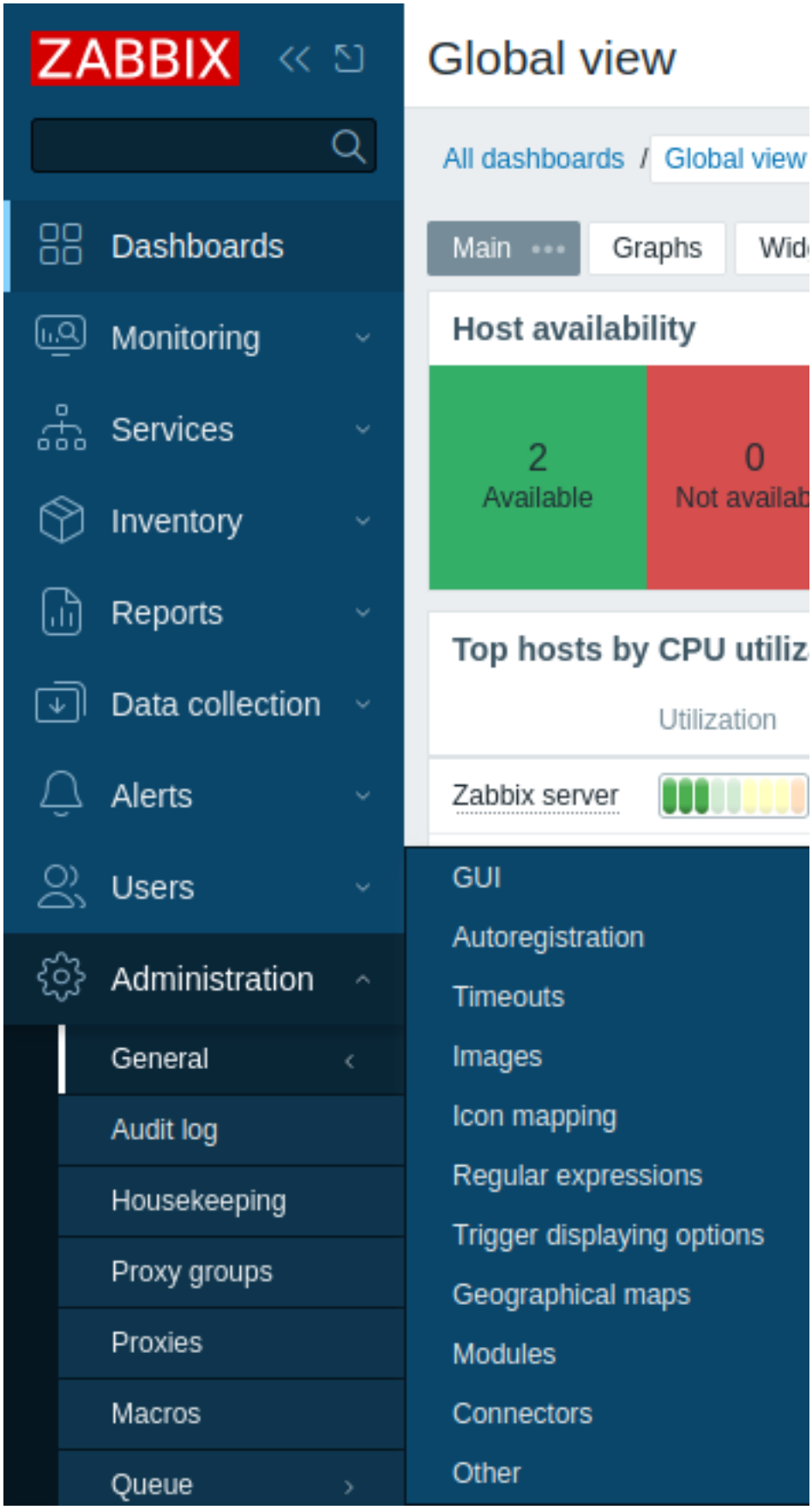
您也可以按 Tab 键，使折叠菜单完全重新显示。反复按 Tab 键，可以将焦点移至下一个菜单元素。

隐藏菜单

即使菜单完全隐藏，只需单击鼠标一次即可打开完整菜单，方法是单击汉堡图标 。请注意，它会重新显示在页面内容之上；若要将页面内容向右移动，必须通过单击显示侧边栏按钮来取消隐藏菜单。

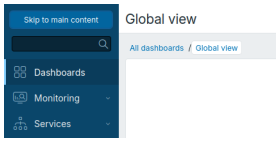
菜单等级

菜单中最多有三个级别。



可访问性：跳过导航

跳转到主要内容链接允许键盘和屏幕阅读器用户跳过重复的侧边栏导航，直接进入主要内容。



在正常使用时，它保持隐藏状态，仅在获得焦点时显示，例如在页面加载后按下 Tab 键，或通过屏幕阅读器访问时。

上下文菜单

除了主菜单之外，Zabbix 还为主机、监控项 和事件 提供上下文菜单，以便快速访问常用实体，例如最新值、简单图形、配置表单、相关脚本或外部链接。

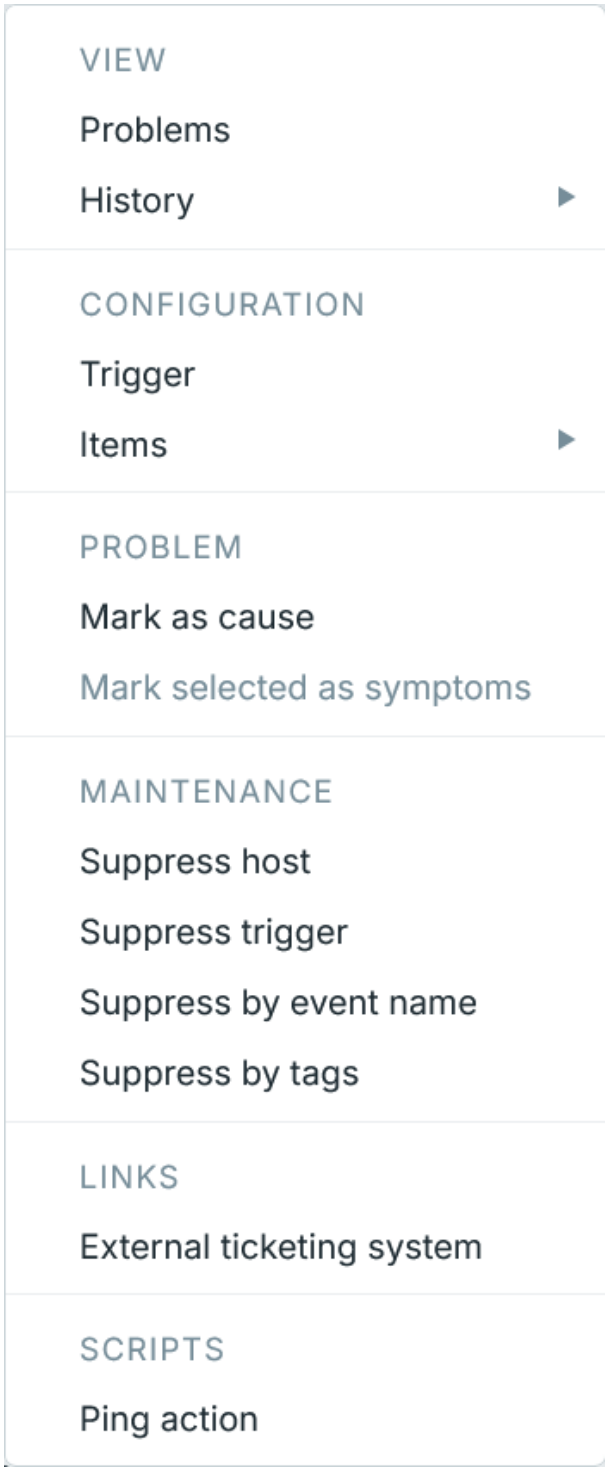
在受支持的位置单击主机、监控项或问题/触发器名称，即可访问上下文菜单。

事件菜单

概述

事件菜单包含指向事件所需操作或前端部分的快捷方式。

单击事件名称即可打开事件菜单。



内容

事件上下文菜单包含以下几个部分：查看、操作、配置、问题、维护、链接和 脚本。对于尚未配置的实体，链接处于禁用状态，并以灰色显示。如果已配置相应实体，则会显示 脚本和 链接部分。

查看部分包含以下链接：

- 问题 - 打开底层触发器的未解决问题列表；
- 历史记录 - 打开底层监控项的 最新数据图表/监控项历史记录。如果触发器使用多个监控项，则每个监控项都会提供相应链接。

操作部分仅在 触发器概览小组件中可用。其中包含以下链接：

- 更新问题 - 打开问题更新页面。

配置部分包含以下配置链接：

- 触发问题的触发器；
- 触发器表达式中使用的监控项。

Note:
请注意，配置部分仅对 Admin 和 Super admin 用户可用。

问题部分包含以下选项：

- 标记为原因 - 将该问题标记为原因；
- 将所选问题标记为症状 - 将选中的问题标记为该问题的症状。

维护部分包含用于启动临时维护的选项：

- 抑制主机 - 为主机启动维护期；
- 抑制触发器 - 为触发器启动维护期；
- 按事件名称抑制 - 为该事件名称启动维护期；
- 按标签抑制 - 为具有相应标签的事件启动维护期。

选择相应选项后，将打开维护期配置表单。

系统会预先选择从当前时间开始、持续一小时的一次性维护期（可在用户设置和个人资料中配置）。相应的主机/触发器/事件名称/标签也会被预先选中。维护名称将按 `Ad-hoc: <event name>` 格式预先填充。

链接部分包含以下链接：

- 访问已配置的触发器 URL；
- 访问在全局脚本中配置的自定义链接（范围为 “手动事件操作”，类型为 “URL”）；
- 访问为该问题配置的外部工单（配置webhook 时，请参阅 包含事件菜单项选项）。

脚本部分包含用于执行全局脚本的链接（范围为 手动事件操作）。此功能可用于运行在外部系统中管理问题工单的脚本。

支持的位置

通过在前端各个部分中单击问题或事件名称，可以访问事件上下文菜单，例如：

- 仪表板小部件，例如 *Problems* 小部件、*Trigger overview* 小部件等。
- *Monitoring* > *Problems*
- *Monitoring* > *Problems* > 事件详情
- *Reports* > *Top 100 triggers*（此位置不支持全局脚本和访问外部工单）

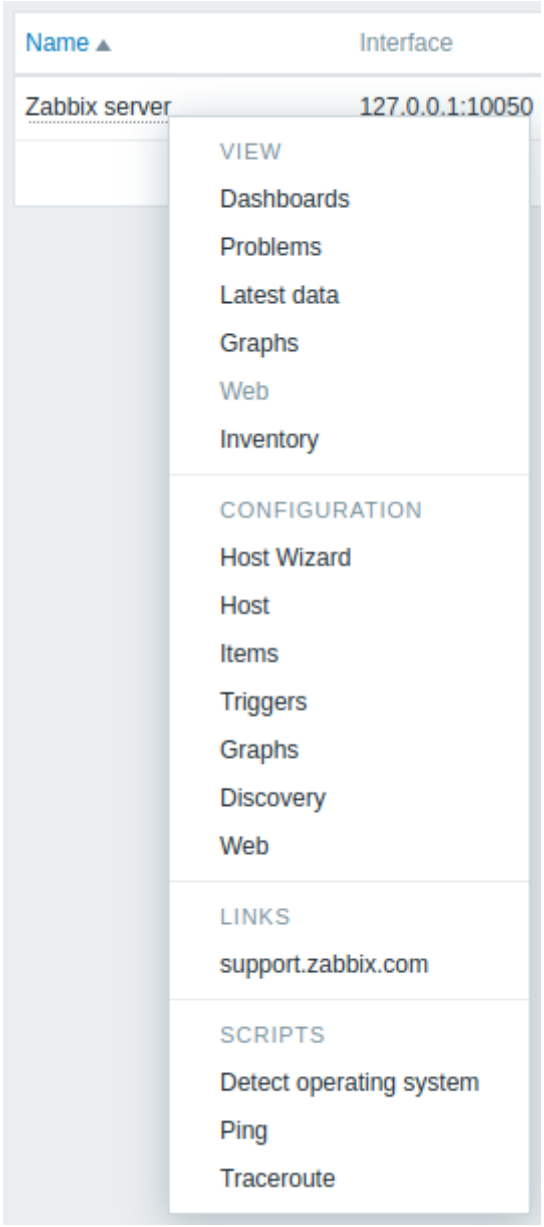
主机菜单

概述

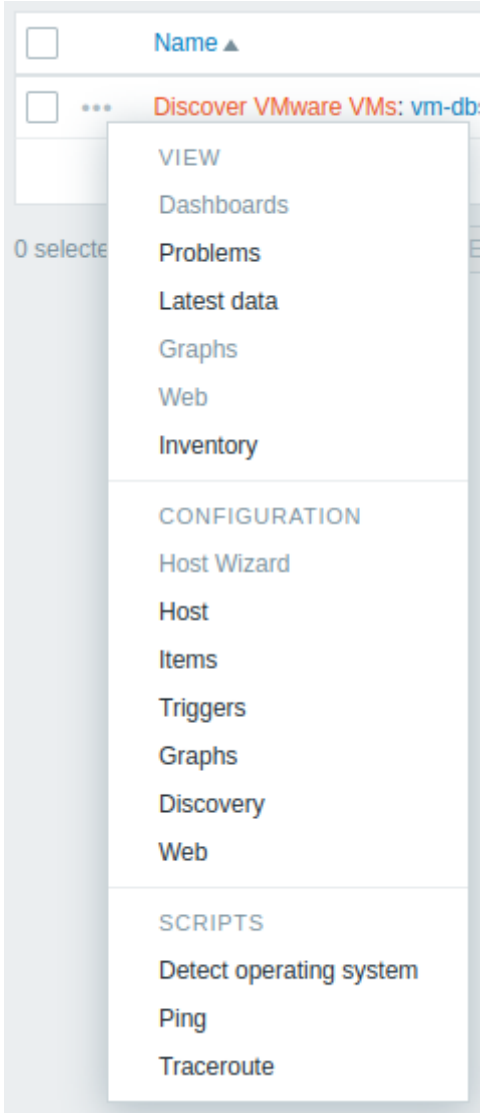
主机菜单包含指向主机经常需要执行的操作或前端部分的快捷方式。

主机菜单可以通过单击主机名称或三点图标打开，具体取决于前端部分，例如：

Monitoring > *Latest data*



Data collection > *Hosts*



内容

主机上下文菜单包含四个部分：查看、配置、链接和脚本。对于未配置的实体，链接将被禁用并显示为灰色。如果已配置相应实体，则会显示脚本和链接部分。

查看部分包含以下链接：

- 仪表板 - 打开小组件和图形。
- 问题 - 打开问题部分，其中列出底层触发器尚未解决的问题。
- 最新数据 - 打开最新数据部分，其中列出当前主机的所有最新数据。
- 图形 - 打开当前主机的简单图形。
- **Web** - 打开指向已配置 Web 场景的链接。
- 资产清单 - 打开指向当前主机资产清单的链接。

配置部分包含以下链接：

- 主机向导 - 为当前主机打开主机向导（对于自动发现的主机禁用）。
- 主机 - 当前主机的配置表单。
- 监控项 - 当前主机监控项的列表。
- 触发器 - 当前主机触发器的列表。
- 图形 - 当前主机的简单图形。
- 发现 - 当前主机低级别发现规则的列表。
- **Web** - 当前主机 Web 场景的列表。

Note:
请注意，配置部分仅对 Admin 和 Super admin 用户可用。

链接部分包含以下链接：

- 访问已配置的触发器 URL。
- 访问在全局脚本中配置的自定义链接（作用域为手动主机操作且类型为'URL'）。

脚本部分允许执行为当前主机配置的全局脚本。这些脚本必须将作用域定义为手动主机操作，才能在主机菜单中使用。

支持的位置

通过在前端的各个部分中单击主机名称，可以访问主机菜单，例如：

- 仪表板小部件，例如 *Problems*、*Top items*、*Trigger overview* 等。
- *Monitoring* > *Problems*
- *Monitoring* > *Problems* > 事件详情
- *Monitoring* > *Hosts*
- *Monitoring* > *Hosts* > *Web Monitoring*
- *Monitoring* > *Latest data*
- *Monitoring* > *Maps*
- *Inventory* > *Hosts*
- *Reports* > *Top 100 triggers*

在 Data collection > Hosts 中单击三个点图标，也可以访问主机菜单。

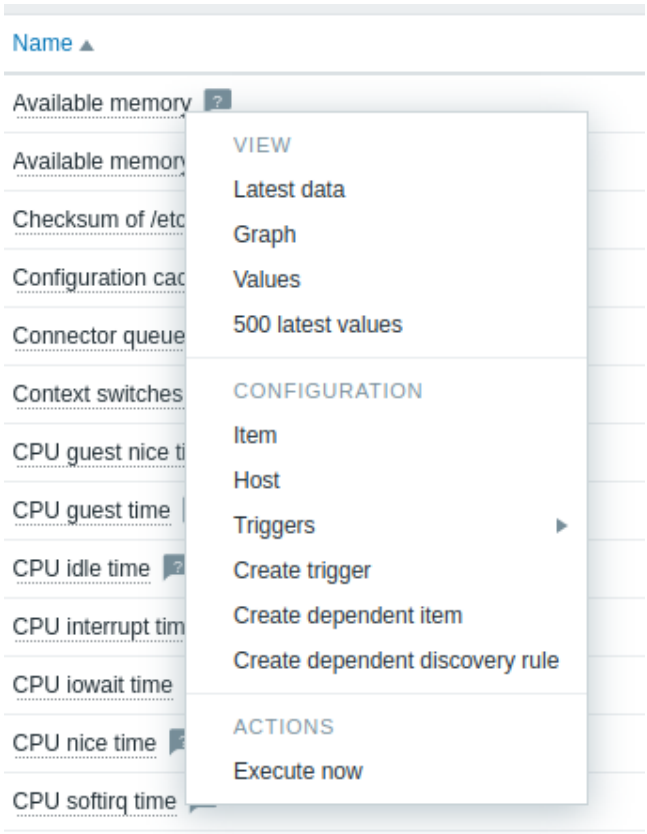
监控项菜单

概述

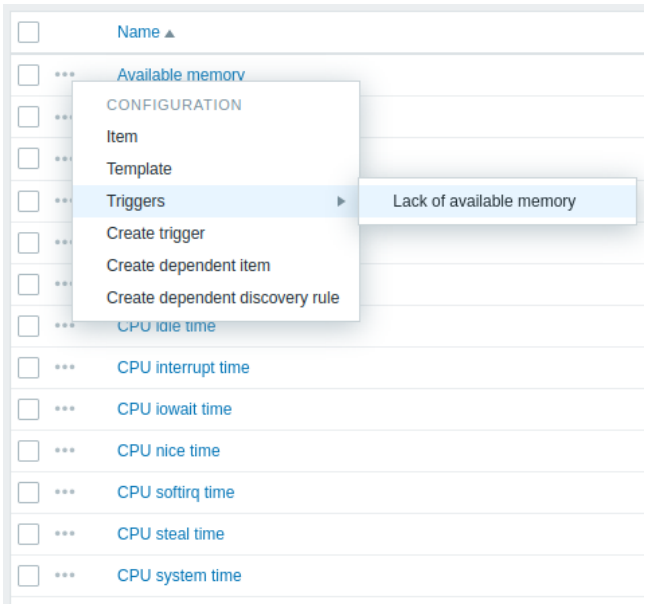
监控项菜单包含一些快捷方式，可快速访问监控项经常需要的操作或前端部分。

根据前端部分的不同，可以通过单击监控项名称或三点图标来打开监控项菜单，例如：

Monitoring > *Latest data*



Data Collection > *Templates* > *Items*



内容

监控项菜单包含三个部分：查看、配置和操作。

查看部分包含以下选项：

- 最新数据 - 打开按当前主机和监控项筛选的最新数据部分；
- 图形 - 打开当前监控项的简单图形；
- 值 - 打开过去 60 分钟内接收的当前监控项的所有值列表；
- 最新 **500** 个值 - 打开当前监控项的最新 500 个值列表。

配置部分（仅适用于管理员和超级管理员类型的用户）包含以下选项：

- 监控项 - 打开当前监控项的监控项配置表单；
- 模板 - 打开该监控项所属模板的模板配置表单（仅当从数据采集 > 模板 > 监控项访问监控项菜单时可用）；
- 主机 - 打开该监控项所属主机的主机配置表单；
- 触发器 - 鼠标悬停时打开监控项触发器列表（如果有）；单击触发器可打开其触发器配置表单；
- 创建触发器 - 打开触发器配置表单，为此监控项创建触发器；
- 创建依赖监控项 - 打开监控项配置表单，创建依赖监控项，并将当前监控项作为主监控项；
- 创建依赖发现规则 - 打开发现规则配置表单，创建依赖发现规则，并将当前监控项作为主监控项。

操作部分包含以下选项：

- 立即执行 - 立即为新的监控项值执行检查。

支持的位置

可以通过在前端的各个部分中单击监控项名称来访问监控项菜单，例如：

- 监控 > 最新数据
- 数据采集 > 主机 > 监控项
- 数据采集 > 主机 > 发现规则 > 监控项原型

还可以通过在*Top items* 仪表板小部件中单击监控项值来访问监控项菜单。

前端部分

菜单结构　Zabbix 前端菜单结构如下：

- 仪表板
- 监控
  - 问题
  - 主机
  - 最新数据
  - 拓扑图
  - 自动发现
- APM
  - 追踪
  - 指标
  - 日志
- 服务
  - 服务
  - SLA
  - SLA 报告
- 清单
  - 概览
  - 主机
- 报表
  - 系统信息
  - 计划报表
  - 可用性报告
  - 触发器 Top 100
  - 审计日志
  - 操作日志
  - 通知
- 数据采集
  - 模板组
  - 主机组

    - 模板
    - 主机
    - 维护
    - 事件关联
    - 自动发现
- 报警
    - 操作
        * 触发器操作
        * 服务操作
        * 自动发现操作
        * 自动注册操作
        * 内部操作
    - 媒介类型
    - 脚本
- 用户
    - 用户组
    - 用户角色
    - 用户
    - API 令牌
    - 认证
    - 设备
- 管理
    - 常规
        * GUI
        * 自动注册
        * 超时
        * 图片
        * 图标映射
        * 正则表达式
        * 触发器显示选项
        * 地理地图
        * 模块
        * 连接器
        * 其他
    - 审计日志
    - 内务处理
    - proxy 组
    - proxy
    - 宏
    - 队列
        * 队列概览
        * 按 proxy 查看队列概览
        * 队列详细信息

仪表板

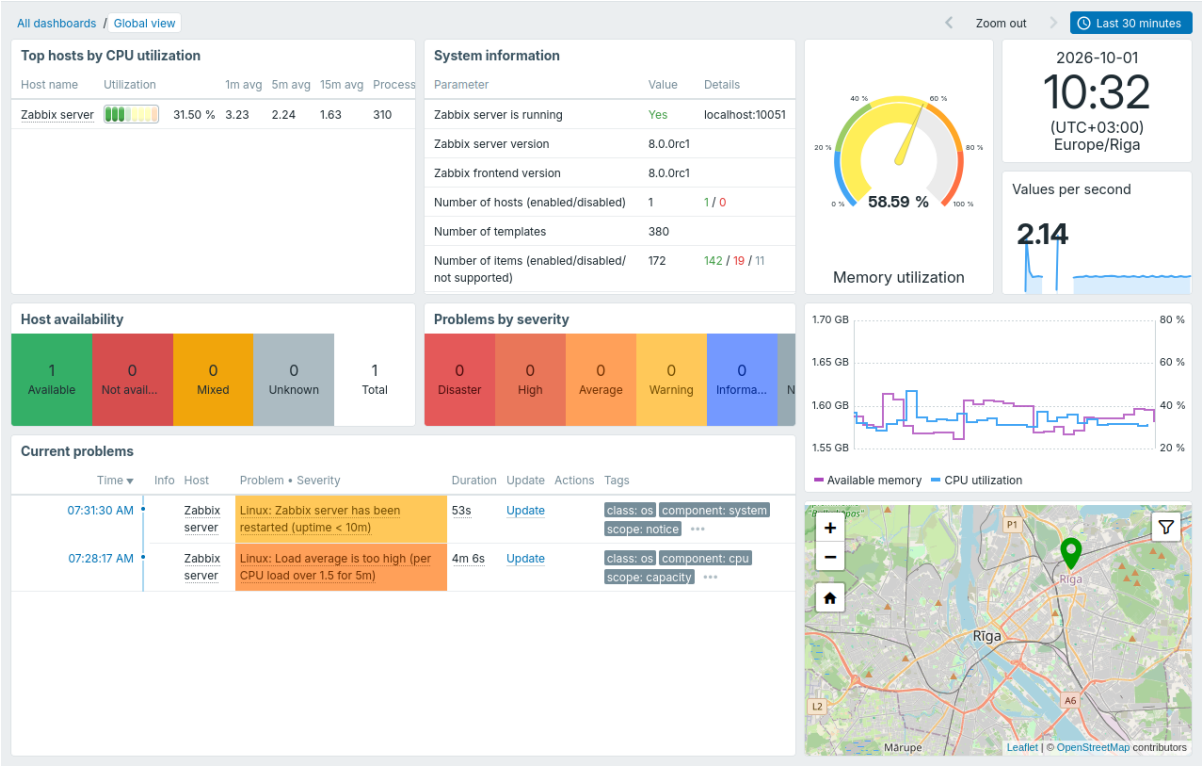
仪表板可让您概览受监控环境。您可以根据需要创建和自定义仪表板。

仪表板的构成元素是小组件：图形、地图、最近问题列表等，每个小组件都用于显示特定类型的信息。

小组件可以组织到多个仪表板页面中。这些页面可以作为幻灯片放映查看。您也可以单击页面选项卡导航到任意页面。

您可以创建多个仪表板，但一次只能显示一个仪表板。

Zabbix 附带了一些默认仪表板。其中一个是全局视图仪表板，用于显示受监控主机（包括 Zabbix 服务器）的运行状况和状态：

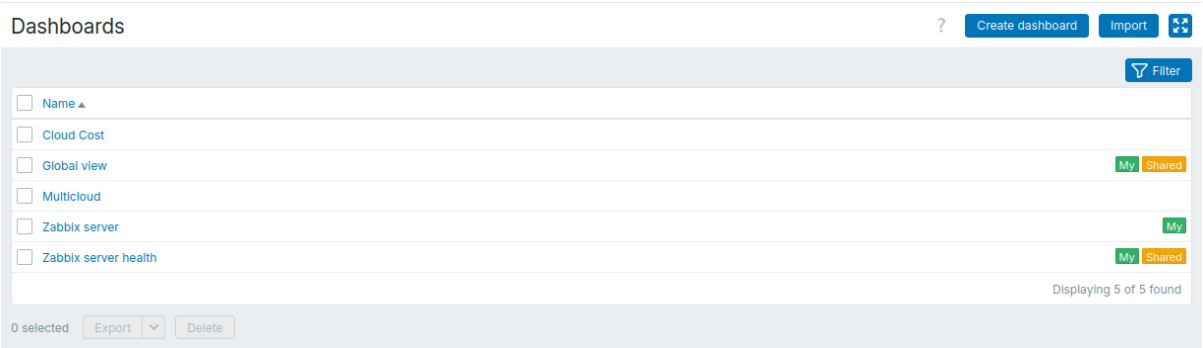


您可以在仪表板存储库中找到更多仪表板。这些仪表板旨在与现有的 Zabbix 模板配合使用，您可以将其下载并导入到您的 Zabbix 环境中。每个仪表板的详细说明（包括前提条件）都位于该仪表板的 README 文件中。

仪表板列表

要打开仪表板列表，可以：

- 在菜单中点击仪表板。
- 在查看仪表板时点击所有仪表板，打开仪表板列表。



是否可以访问仪表板菜单取决于用户角色以及 *UI* 元素访问权限下的仪表板权限。

在此处，您可以：- 创建仪表板。- 删除仪表板。选中仪表板对应的复选框，然后点击删除按钮即可删除仪表板。- 导入仪表板。- 导出仪表板。

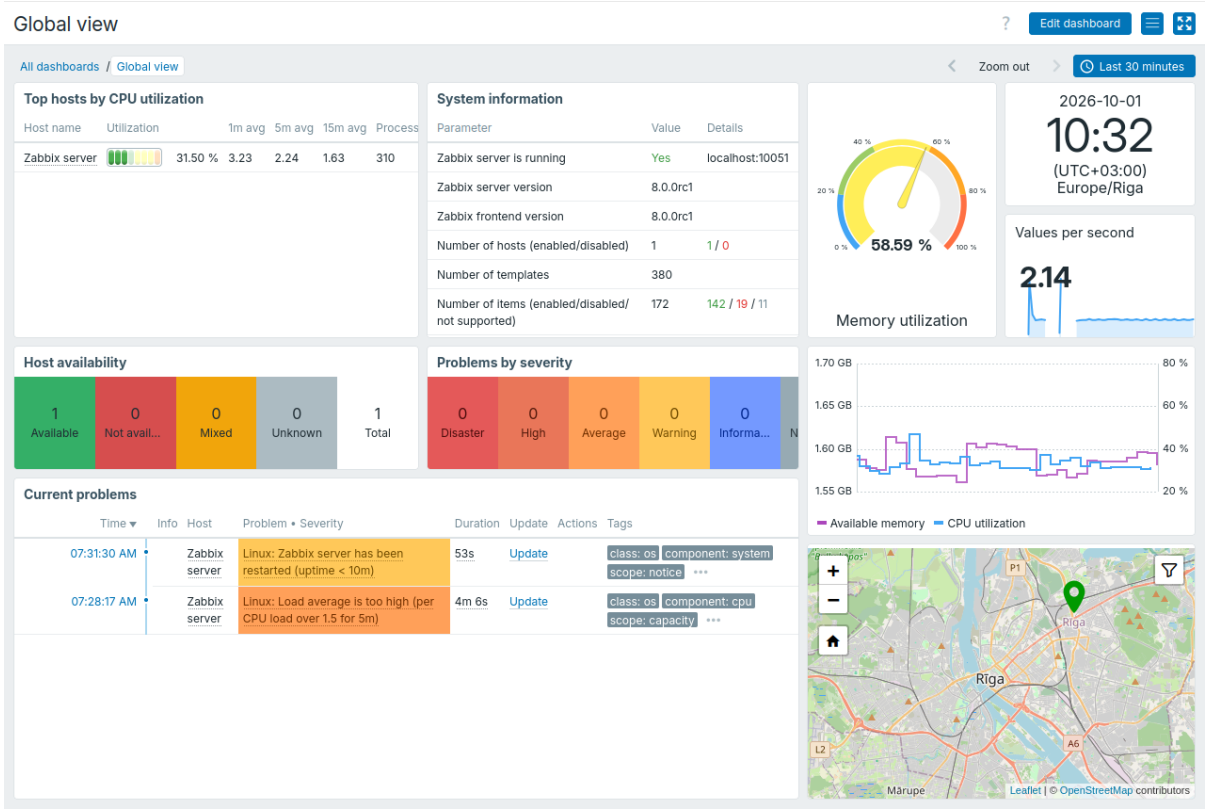
仪表板带有标签，用于指示其所有权和共享设置：

- 我的 - 仪表板归您所有，且仅对您可见。
- 我的 + 已共享 - 仪表板归您所有，但为公开仪表板，或已与特定用户或用户组共享。
- 没有标签的仪表板归其他用户所有，但已公开或与您共享。

您可以按仪表板名称和所有者筛选列表。

查看仪表板

在仪表板列表中点击仪表板名称，即可查看仪表板：



Zabbix 用户界面会记住上次打开的仪表板，以便在下次会话中打开。

Note:
您可以在仪表板上看到的数据取决于您对主机的访问权限。

对于显示图形的部件，上方的时间段选择器用于设置时间范围。它以标签形式显示当前时间范围。点击该标签可打开或关闭选择器。

您可以在此执行以下仪表板管理操作：

- 编辑仪表板。
- 从仪表板的操作菜单中执行以下操作：
  - 共享 - 编辑仪表板的共享设置。
  - 新建 - 创建新的仪表板。
  - 克隆 - 通过克隆当前仪表板创建新的仪表板。保存前可以编辑仪表板属性。
  - 删除 - 删除仪表板。
  - 导出 - 导出仪表板。
  - 创建新报告 - 创建计划报告。如果您没有管理计划报告的权限，此选项将被禁用。
  - 查看相关报告 - 查看当前仪表板的现有报告列表。如果没有相关报告，或者您没有管理计划报告的权限，此选项将被禁用。
- 切换到大屏模式。
- 启动或停止仪表板幻灯片放映。

在仪表板查看模式下，部件提供以下选项：

- 编辑部件。
- 从部件的操作菜单 ... 中执行以下操作：
  - 复制 - 复制部件。
  - 下载图像 - 将部件下载为 PNG 图像。此选项适用于支持此功能的部件（请参阅各个部件页面）。
  - 刷新间隔 - 选择刷新部件内容的频率。

对于包含多个页面的仪表板，在仪表板查看模式下，可以从页面的操作菜单  中执行以下操作：

- 复制 - 复制页面。
- 属性 - 自定义页面属性，包括名称和幻灯片放映间隔。

幻灯片放映

如果在仪表板属性中选中 开始幻灯片放映复选框，则当仪表板包含多个页面时，幻灯片放映会自动开始。否则，您可以点击 开始幻灯片放映按钮来开始幻灯片放映。



页面会按照仪表板属性和单个页面属性中指定的时间间隔进行切换。如果为单个页面定义了 幻灯片放映间隔属性，则应用该页面指定的时间间隔。否则，应用仪表板的 默认幻灯片放映间隔。

要停止幻灯片放映，请点击 停止幻灯片放映。

在信息亭模式下，使用以下按钮：

-  开始幻灯片放映；
-  停止幻灯片放映；
-  导航到上一页；
-  导航到下一页。

共享仪表板

仪表板可以是公开的，也可以是私有的。

公开仪表板对所有用户可见。私有仪表板仅对其所有者以及与其共享的用户或用户组可见。

要更改仪表板的共享设置，请在查看仪表板时，点击仪表板操作菜单中的共享：

参数	描述
类型	选择仪表板类型： 私有 - 仪表板仅对选定的用户组和用户可见。 公开 - 仪表板对所有用户可见。
用户组共享列表	选择可以访问仪表板的用户组。 您可以授予只读或读写访问权限。
用户共享列表	选择可以访问仪表板的用户。 您可以授予只读或读写访问权限。

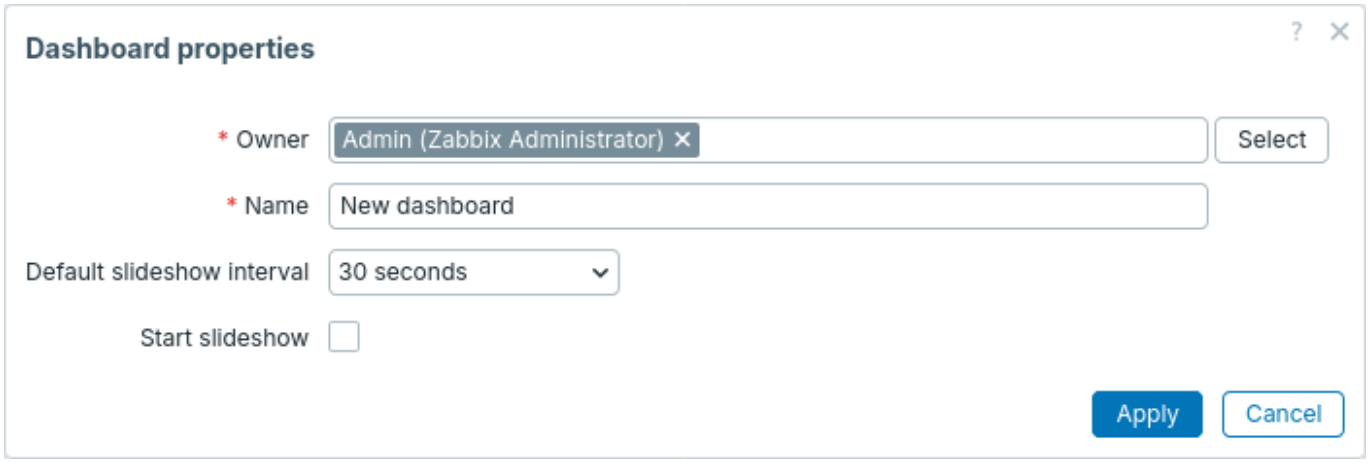
请参阅仪表板权限，了解共享选项如何限制用户可执行的操作。

创建仪表板

您可以通过以下两种方式创建仪表板：

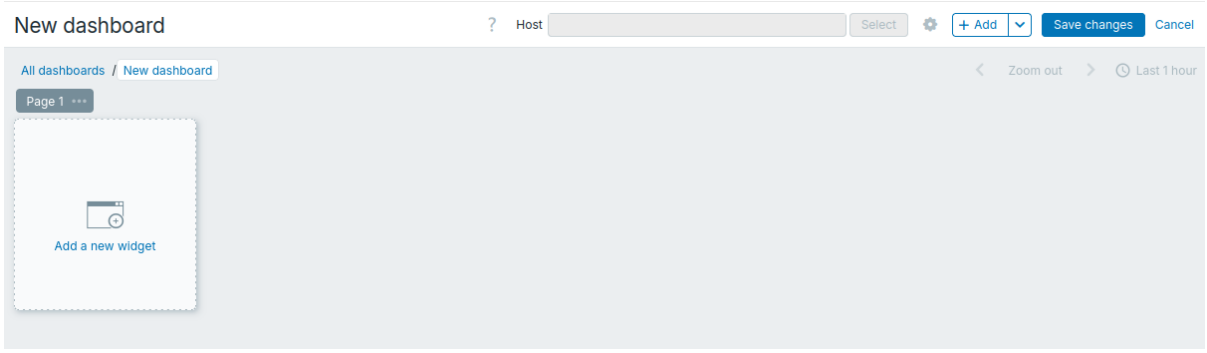
- 在仪表板列表中，单击创建仪表板按钮。
- 在仪表板查看模式下，单击仪表板操作菜单中的新建。

然后，指定仪表板属性：



参数	描述
所有者	选择拥有该仪表板的用户。
名称	输入仪表板名称。
默认幻灯片放映间隔	选择每个仪表板页面在幻灯片放映切换到下一页之前显示的时长。
开始幻灯片放映	选中此复选框，以便在仪表板包含多个页面后自动开始幻灯片放映。

然后，单击应用。新仪表板将在编辑模式下打开：



在此处，您可以添加仪表板小组件和页面。

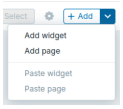
请注意，仪表板的最小宽度为 1200 像素；在较窄的窗口中，会显示水平滚动条。较宽的仪表板始终会填充浏览器窗口；其小组件无法超出窗口边缘。

单击保存更改按钮保存仪表板，或单击取消返回上一屏幕。

添加小组件

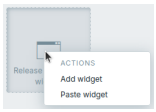
在仪表板编辑模式下，可以通过以下两种方式添加小组件：

- 点击 添加按钮。或者，点击 添加按钮上的下拉箭头，然后选择 添加小组件。



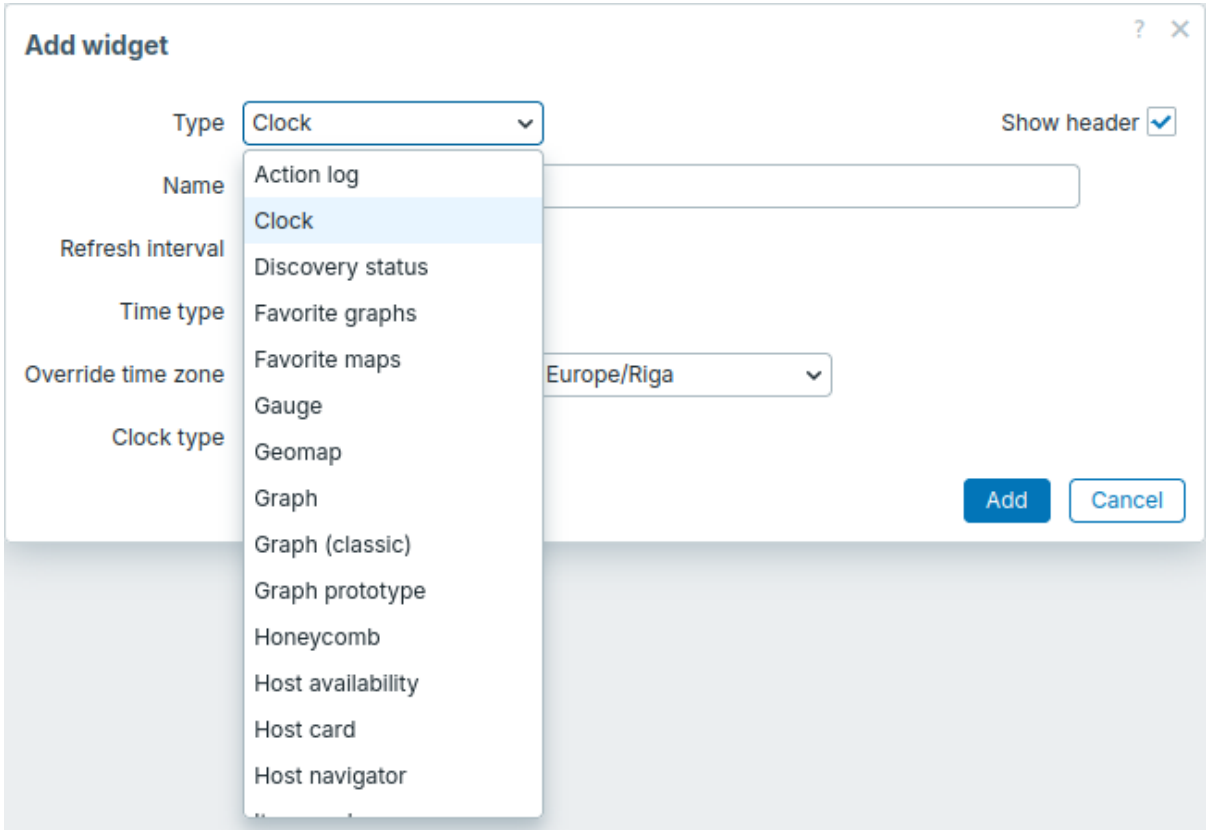
小组件占位符将以默认大小添加到仪表板上第一个可用的空白区域，并打开小组件配置表单。

- 将鼠标指针移到仪表板的空白区域。然后点击小组件占位符，打开小组件配置表单。或者，点击并拖动占位符调整其大小，然后释放鼠标以打开小组件配置表单；如果某个小组件已复制到剪贴板，系统会首先提示您选择 添加小组件或 粘贴小组件。



然后，执行以下步骤：

1. 在小组件配置表单中，选择小组件类型。
2. 配置小组件参数。
3. 点击 添加，将小组件添加到仪表板。



小组件将以默认大小添加，或调整大小以适应可用空间。您可以按照编辑小组件部分中的说明调整小组件大小和位置。

您还可以复制和粘贴小组件，以使用相同的配置创建新的小组件，无论是在同一仪表板中，还是在不同浏览器标签页中打开的仪表板之间。

复制/粘贴小组件

要复制和粘贴小组件：

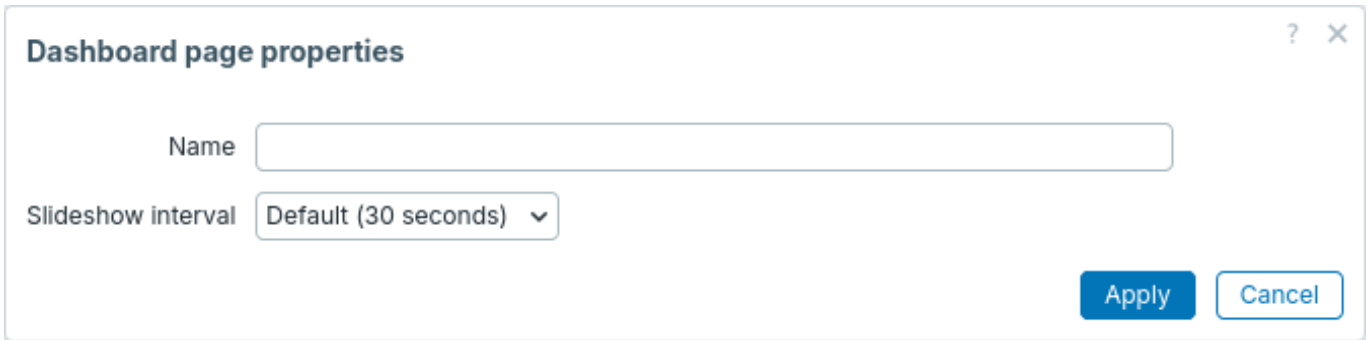
1. 在小组件的 操作菜单 … 中，选择 复制。
2. 在仪表板编辑模式下，单击 添加按钮上的下拉箭头，然后选择 粘贴小组件。也可以单击仪表板中的空白区域，然后选择 粘贴小组件。

您还可以使用现有小组件 操作菜单中的 粘贴选项，将复制的小组件粘贴到现有小组件上。

添加页面

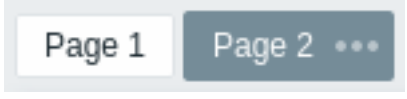
要向仪表板添加新页面：

1. 在仪表板编辑模式下，单击 添加按钮上的下拉箭头，然后选择 添加页面。
2. 填写页面属性：
   - 名称 - 输入页面名称。如果将名称留空，页面将以 页面 *N* 的名称添加，其中 “N” 是递增的页面编号。
   - 幻灯片放映间隔 - 选择页面在幻灯片放映切换到下一页之前显示的时长。



3. 单击 应用。

新页面会添加一个新选项卡（例如 页面 *2*）。



您还可以复制和粘贴页面，以使用相同的属性创建新页面；此操作既可以在同一仪表板内进行，也可以在不同选项卡中打开的仪表板之间进行。

复制/粘贴页面

要复制和粘贴页面：

1. 在页面的 操作菜单 中，选择 复制。
2. 在仪表板编辑模式下，单击 添加按钮上的下拉箭头，然后选择 粘贴页面。

编辑仪表板

要编辑仪表板，请点击右上角的 编辑仪表板按钮。或者，点击监控项的 编辑按钮。仪表板将以编辑模式打开。

在此处，您可以执行以下仪表板管理操作：

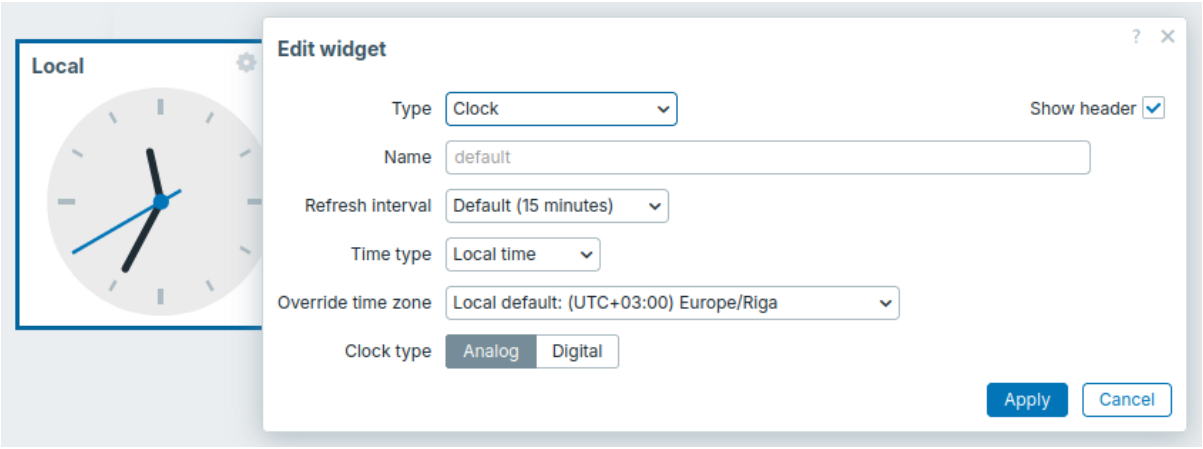
- 点击齿轮按钮，编辑仪表板属性。
- 编辑仪表板监控项。
- 点击 添加按钮上的下拉箭头，访问以下仪表板操作：
  - 添加监控项 - 向仪表板添加监控项。
  - 添加页面 - 向仪表板添加页面。
  - 粘贴监控项 - 粘贴一个已复制的监控项。如果尚未复制任何监控项，此选项将显示为灰色。一次只能复制一个监控项。
  - 粘贴页面 - 粘贴已复制的页面。如果尚未复制任何页面，此选项将显示为灰色。一次只能复制一个页面。

编辑仪表板后，点击 保存更改按钮保存仪表板，或点击 取消返回上一屏幕。

编辑小组件

在仪表板编辑模式下，点击小组件的 编辑按钮，即可编辑小组件。
或者，在查看仪表板时，将鼠标悬停在小组件上，然后点击 编辑按钮。

编辑小组件时，小组件会高亮显示，并实时预览更改。



您可以拖动小组件及其配置表单的标题栏来重新定位它们，也可以拖动小组件的调整大小控件来调整小组件大小。



编辑小组件时，可以从小组件的 操作菜单 ... 中执行以下操作：

- 复制 - 复制小组件。
- 粘贴 - 将已复制的小组件粘贴到此小组件上。如果尚未复制任何小组件，此选项将显示为灰色。
- 删除 - 删除小组件。

如果未配置小组件的必填参数，小组件将进入 小组件未完全配置状态。

编辑完成后，点击小组件配置表单中的 应用，然后点击仪表板右上角的 保存更改，以应用您的更改。

编辑页面

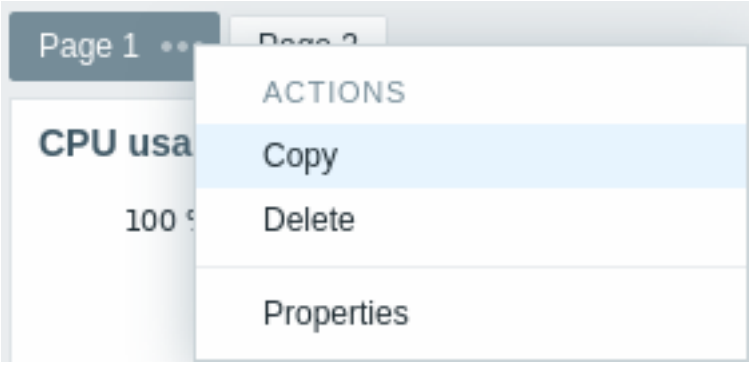
可以在仪表板编辑模式下编辑页面。

通过拖放页面选项卡可以重新排列页面。重新排列不会重命名页面，但具有自动分配名称（*Page N*）的页面除外，这些页面会重新编号。

编辑页面时，可以从页面的操作菜单 中执行以下操作：

- 复制 - 复制页面。
- 删除 - 删除页面。

- 属性 - 自定义页面属性。



仪表板权限

对于普通用户和 *Admin* 类型的用户，如果其分配的用户角色已启用仪表板访问权限以及 创建和编辑仪表板选项，则其仪表板权限受以下限制：

- 即使仪表板的共享设置中未添加任何用户或用户组，他们也可以查看和克隆公共仪表板。
- 如果他们在共享设置中至少拥有该仪表板的 读取权限，则可以查看和克隆私有仪表板。
- 只有在共享设置中拥有该仪表板的 读写权限时，他们才能编辑和删除仪表板。
- 他们无法更改仪表板所有者。

仪表板小部件

概述

Zabbix 提供了多种仪表板小部件，用于可视化和汇总监控数据，帮助您清晰了解系统性能和可用性的整体情况。

以下是所有小部件的概览，并按其主要用途进行分组。

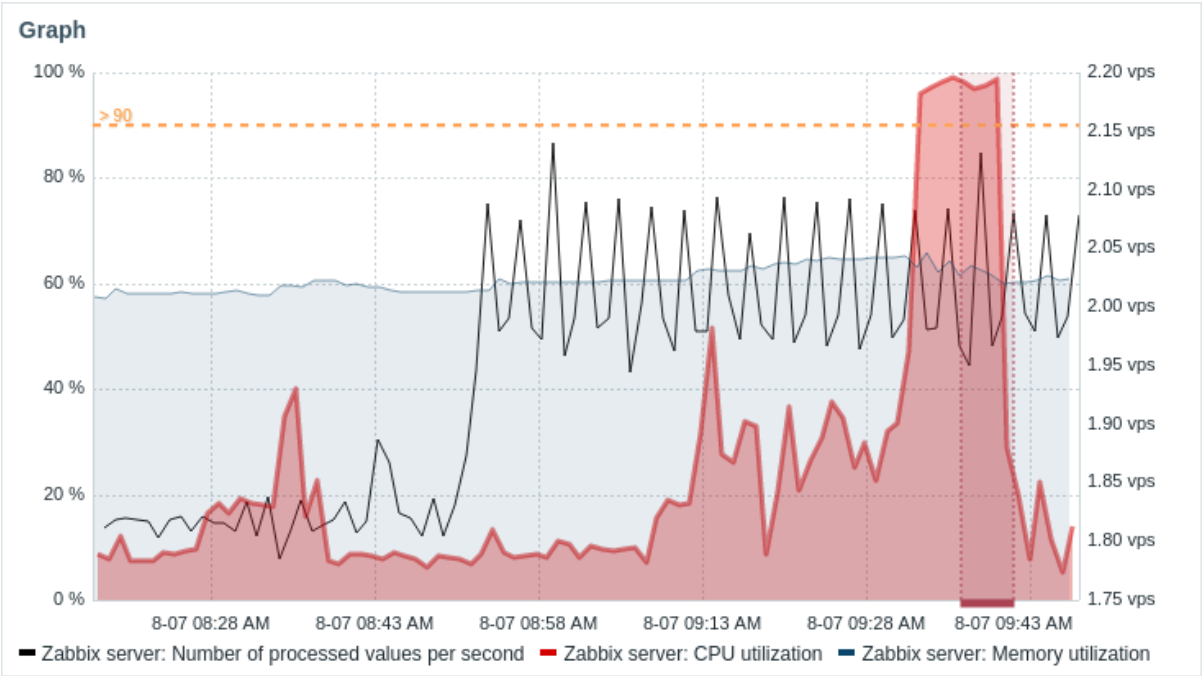
有关小部件配置的详细信息，请参见仪表板小部件参数。

数据可视化

这些小部件以各种可视化格式呈现数据，帮助您以不同方式监控数据和问题。

图形

Graph 小组件以矢量图形式显示数值型监控项数据。它可帮助您跟踪指标、发现问题，并比较一段时间内以及不同主机之间的值。



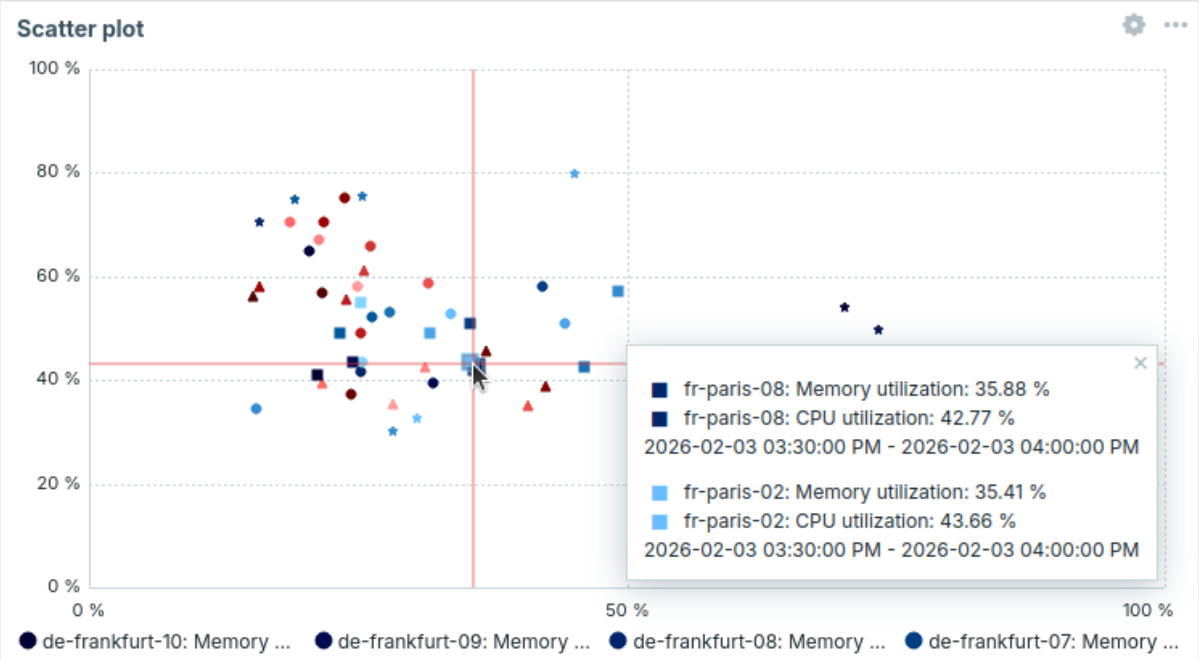
另请参见：

- Graph (classic) - 传统的 Graph 小组件，以图像形式显示数值型监控项数据。
- Graph prototype - 显示基于低级别发现规则图形原型或监控项原型自动创建的自定义图形网格。

- Favorite graphs - 按字母顺序显示当前用户标记为收藏的图形快捷方式列表。

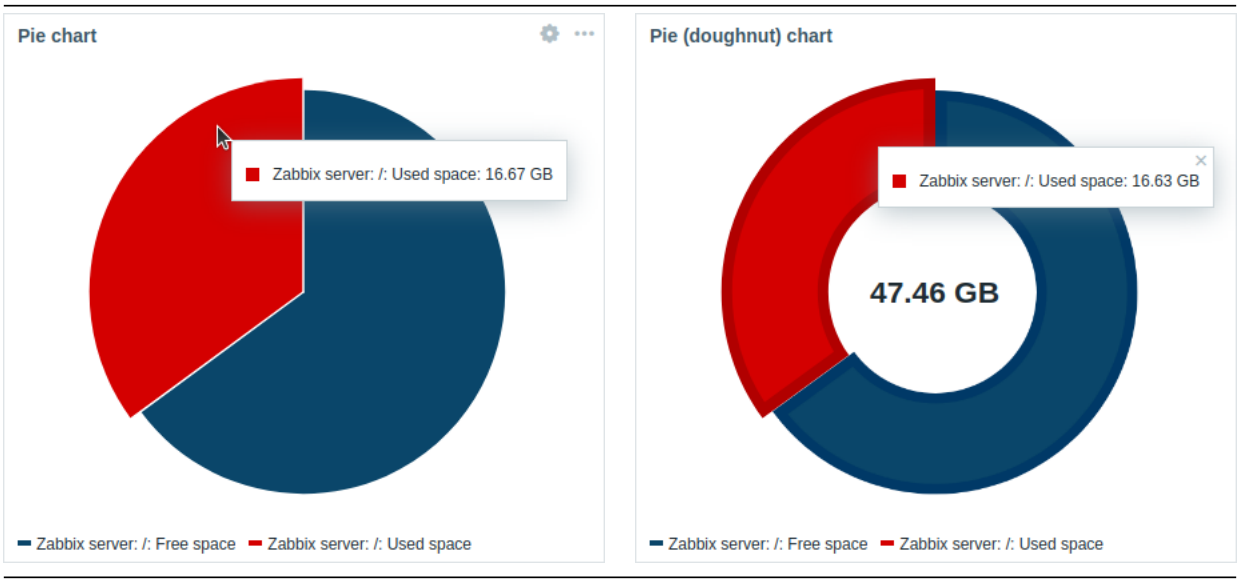
散点图

散点图 小组件通过沿 X 轴和 Y 轴绘制单个数据点来显示两个指标之间的关系。
这有助于揭示数据集中的模式、聚类、相关性和离群值。



饼图

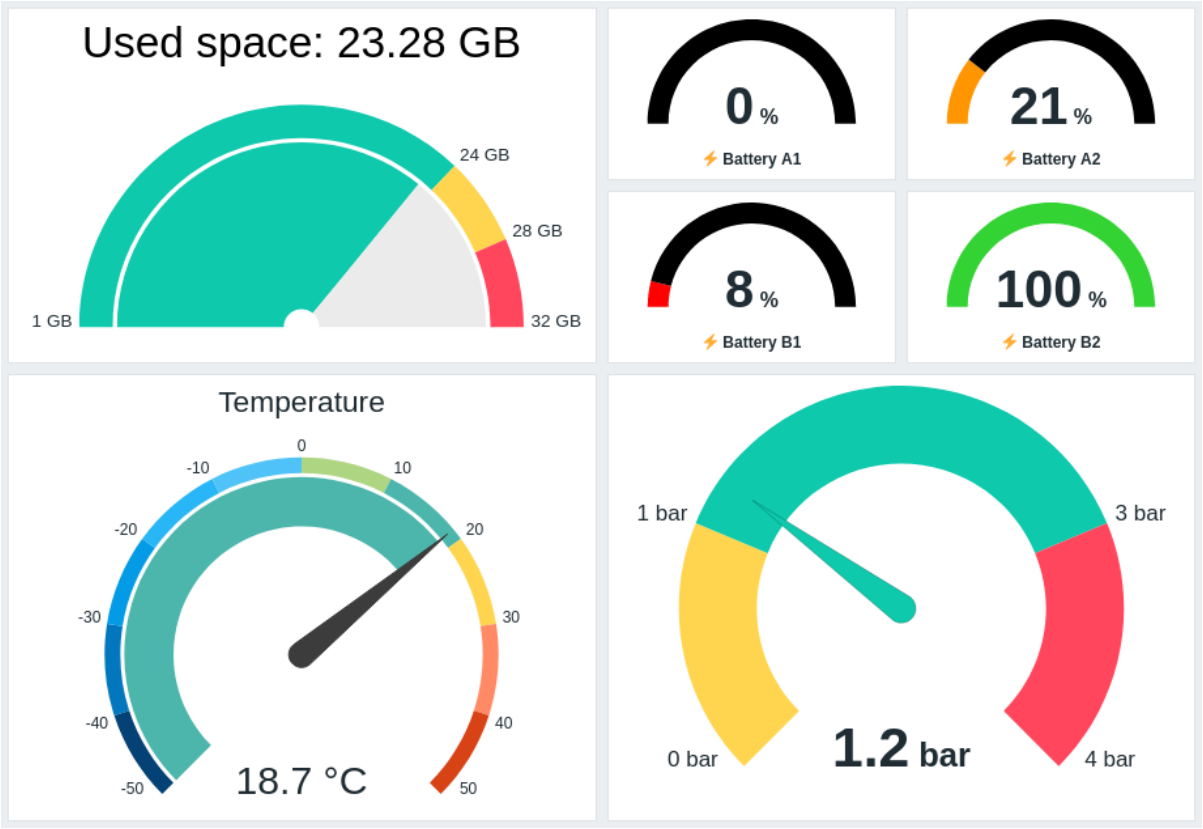
Pie chart 小部件将数值型监控项数据以基于矢量的饼图或环形图形式显示。该小部件可帮助您直观了解各监控项或主机对整体数据集的贡献。



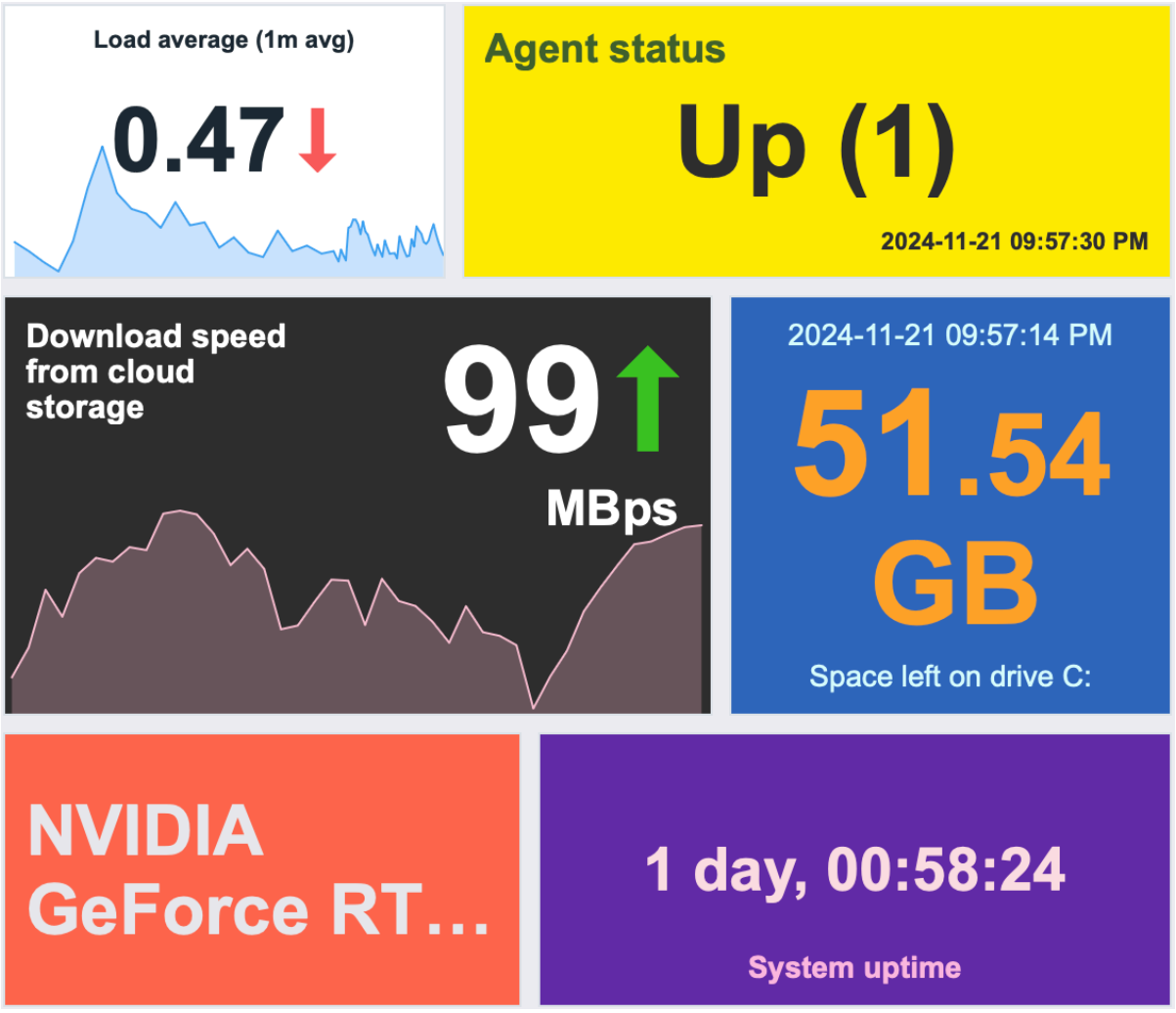
仪表盘/监控项值

仪表盘/监控项值 小部件显示单个监控项的值，可用于关注重要指标、可视化阈值以及检测数据中的突发变化。

仪表盘小部件：



监控项值小部件：



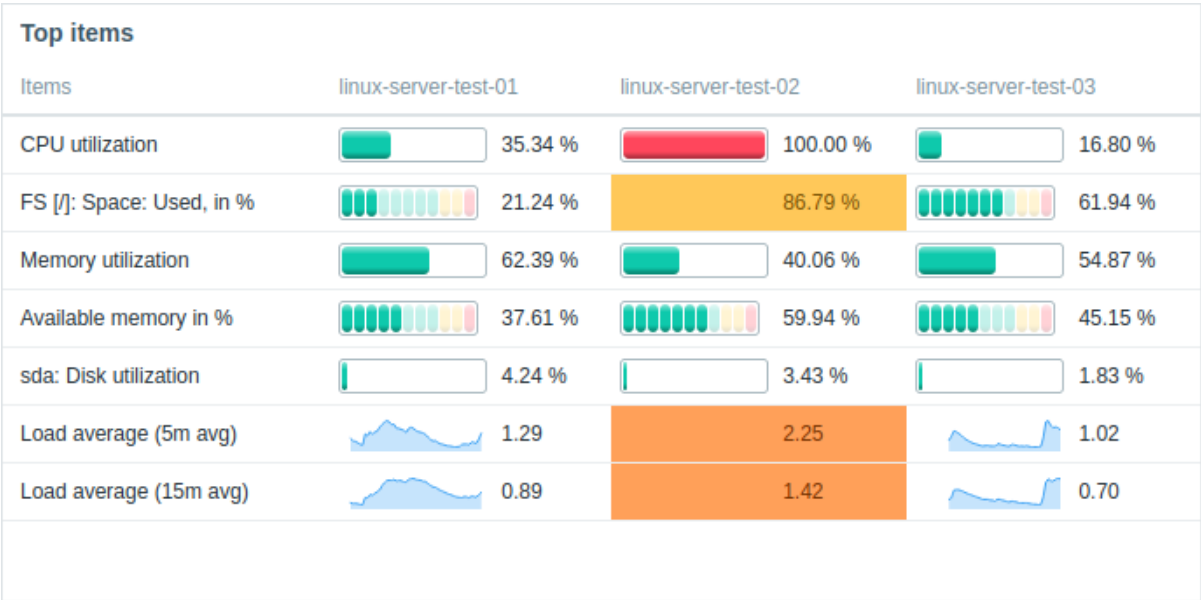
监控项历史

监控项历史 小部件以表格格式显示所有监控项类型（数值、文本等）的最新数据。
它可以显示进度条和图像（对浏览器监控项很有用），还可以高亮显示值（对日志文件监控很有用）。



热门监控项

Top items 小部件显示所选监控项的最高/最低值，便于快速概览其性能。

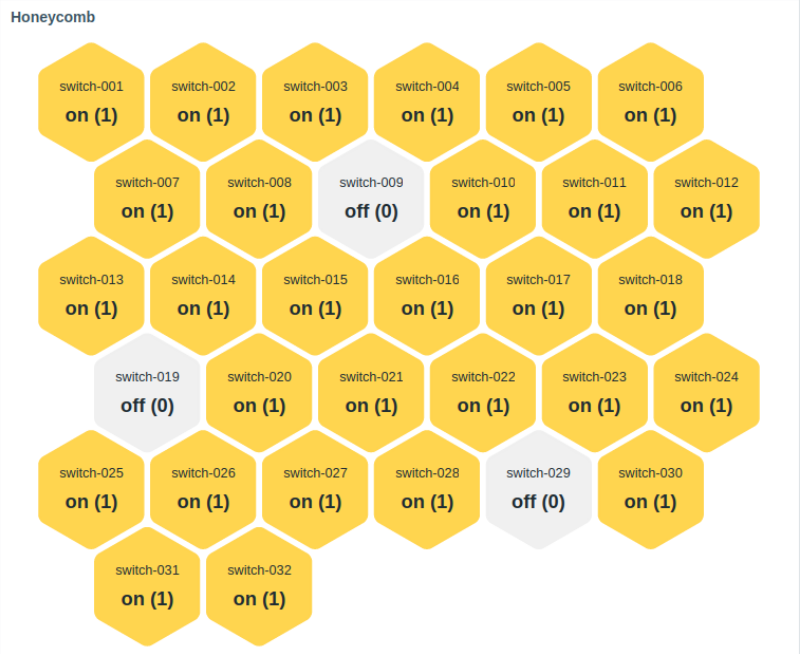


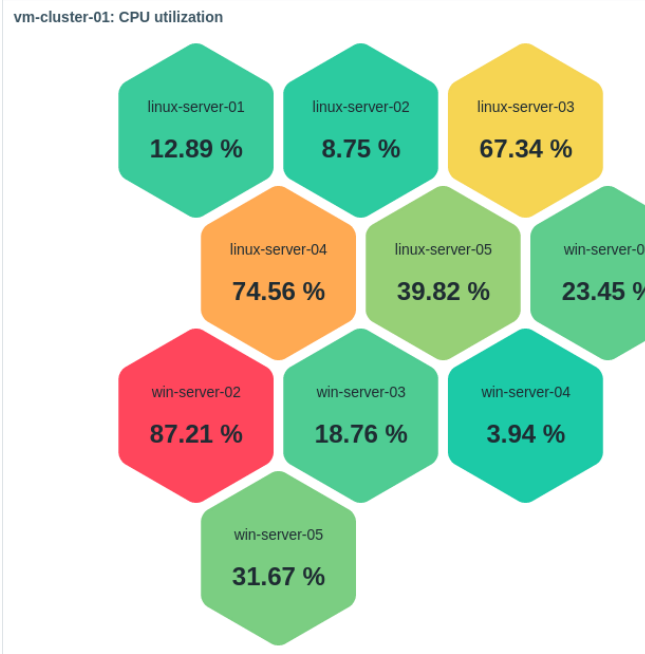
另请参见：

- Top hosts - 显示来自多个主机的所选监控项的最高/最低值。
- Top triggers - 显示问题数量最多的触发器。

蜂窝图

Honeycomb 小部件以网格中的六边形单元显示主机或监控项。这种布局便于检测有问题的主机/监控项、发现问题集群、快速比较各组情况等。



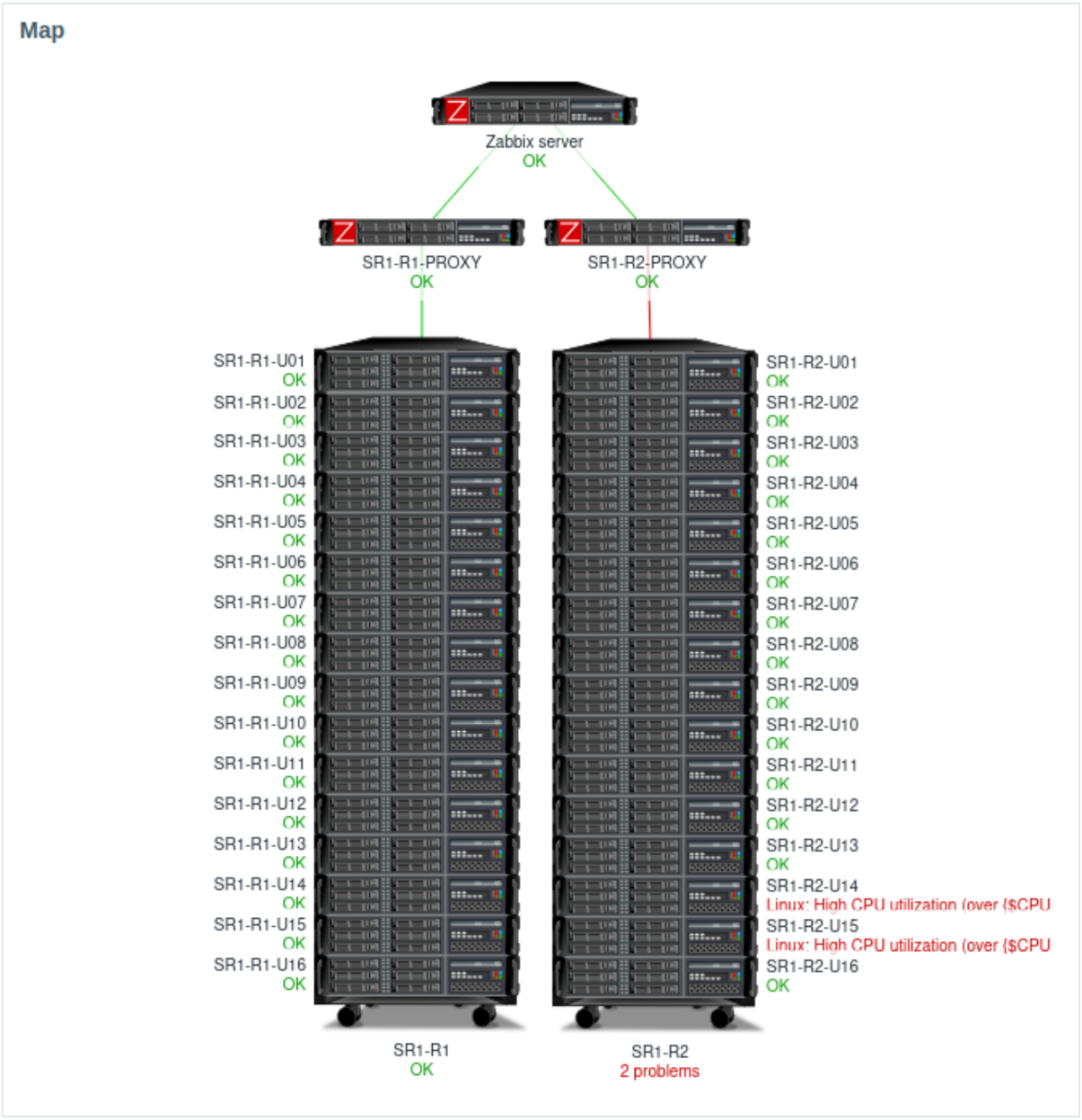


拓扑图

拓扑图小组件可将网络拓扑或主机位置可视化，帮助评估整体健康状况以及特定位置的健康状况。

地图

地图 小部件可显示网络地图或类似的可视化内容，为您提供网络的动态概览。

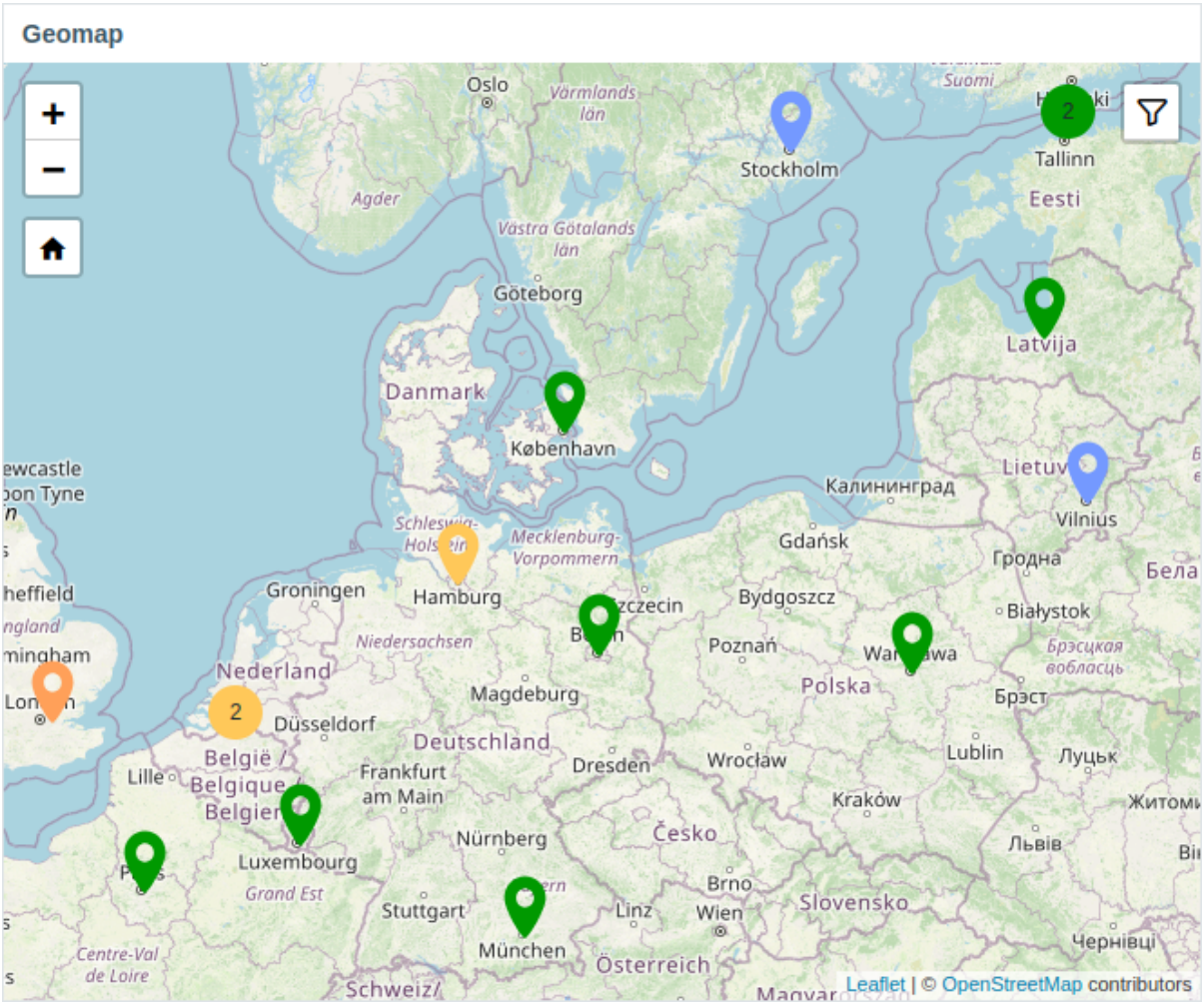


另请参见：

- 地图导航树 - 显示现有地图的层级结构，以及每个地图和地图组的问题数量。
- 收藏地图 - 显示当前用户标记为收藏的地图快捷方式列表，并按字母顺序排序。

地理地图

Geomap 小部件在基于 Leaflet 的交互式地图上将主机显示为标记，标记颜色表示每个主机最严重的问题。

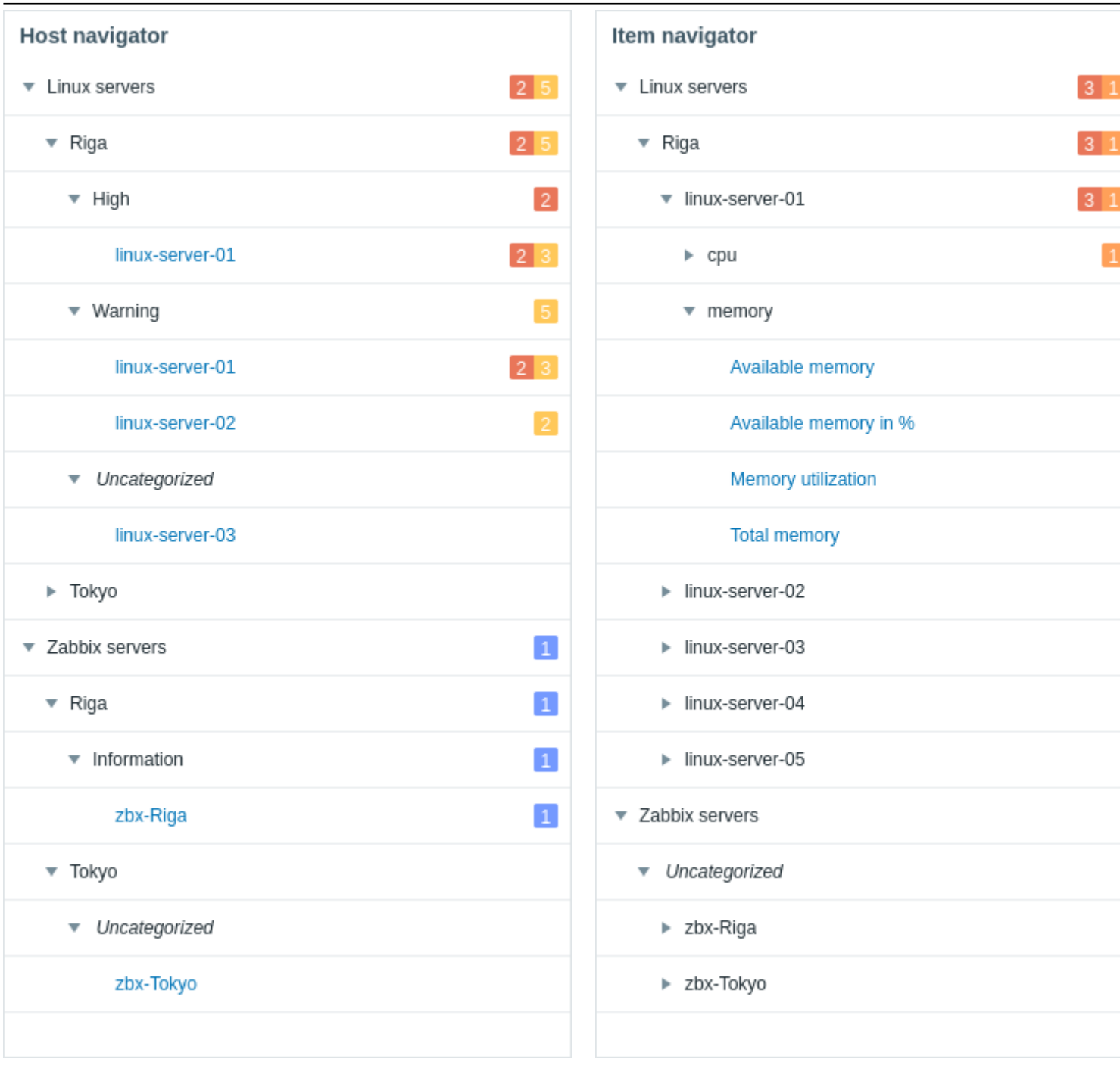


主机/监控项详情

这些小部件提供主机或监控项的列表和详细视图。

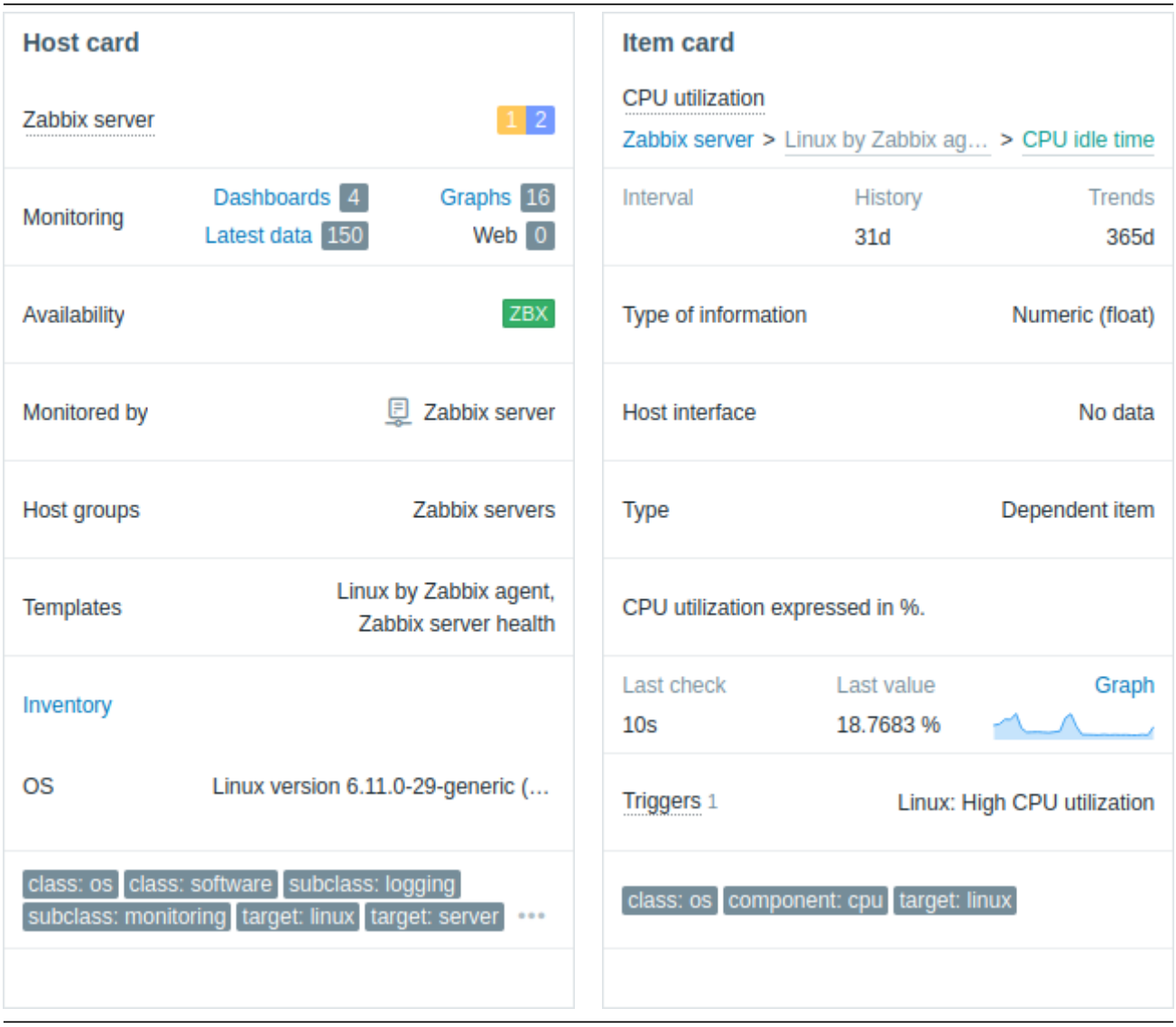
主机/监控项导航器

主机导航器/监控项导航器 小部件根据各种过滤和分组选项显示主机/监控项列表。
这些小部件最适合用于控制其他小部件根据所选主机/监控项显示的内容。



主机/监控项卡片

主机卡片/监控项卡片 小部件显示单个主机/监控项的详细信息，使您能够快速评估主机或监控项的状态和配置（主机可用性、问题、资产记录、监控项最新数据、错误、描述等）。



状态和报告

这些小部件显示状态数据（例如，可用性、触发器、问题）以及报告。

主机可用性

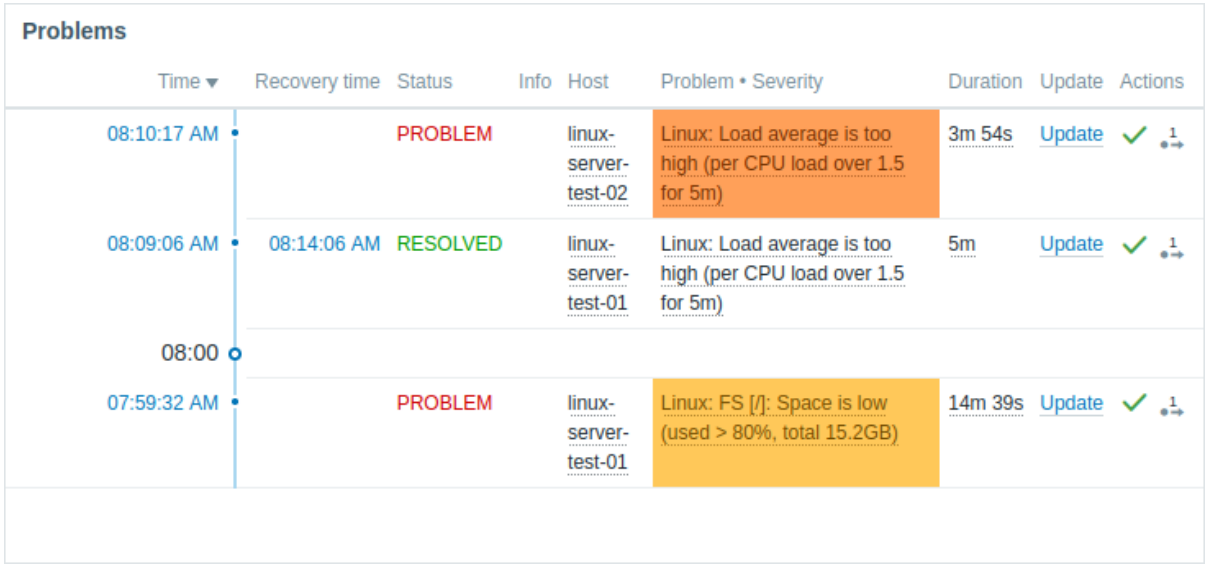
主机可用性 小部件显示所选主机组中的主机可用性，让您能够监控处于在线或离线状态的主机数量。

Host availability

	Total Hosts	Agent (active)	Agent (passive)	SNMP	JMX	IPMI
Available	10	3	8	2	0	0
Not available	2	0	3	0	0	0
Mixed	1	-	0	0	0	0
Unknown	4	0	4	0	0	0
Total	17	3	15	2	0	0

问题

问题 小组件显示按不同参数（主机群组、主机、问题名称等）过滤后的当前问题，让您清楚了解哪些内容需要关注。

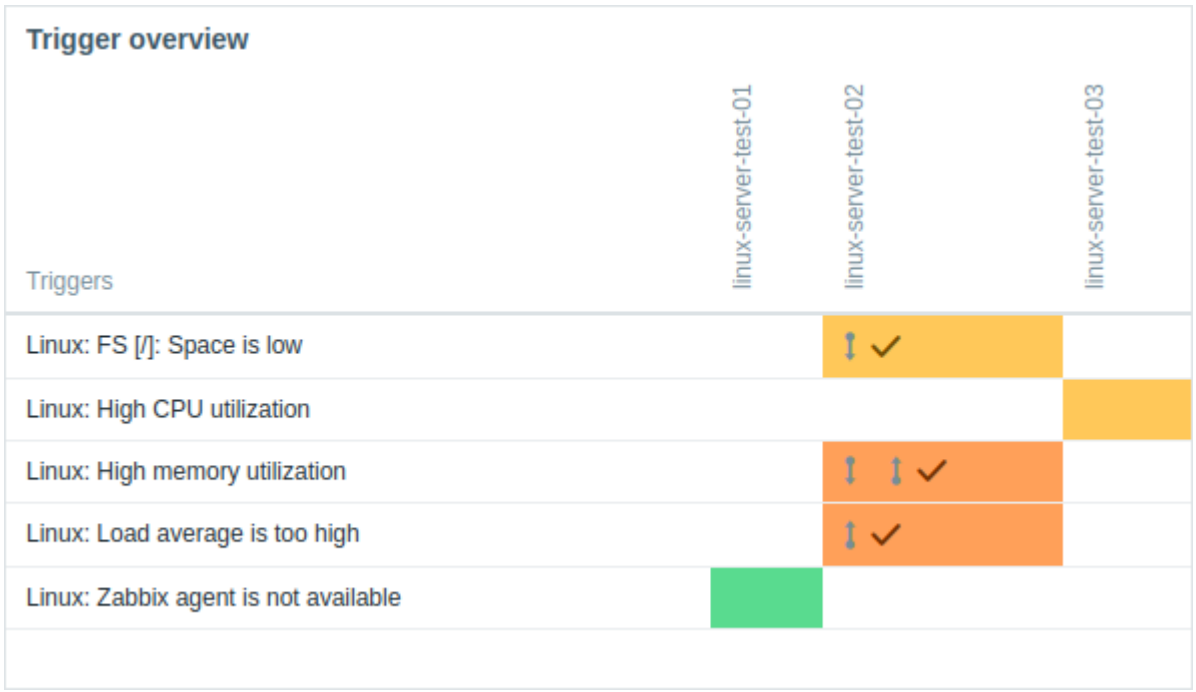


另请参见：

- 按严重性显示的问题 - 显示按严重性分组的根因问题数量。
- 问题主机 - 显示每个主机群组中的根因问题数量，并显示每个群组中的最高问题严重性。

触发器概览

触发器概览 小组件以彩色块表格的形式显示主机触发器的当前状态。
它可快速直观地展示不同主机上触发器的健康状态和活动情况。



操作日志

操作日志 小组件显示在一个动作中执行的操作（通知、远程命令）的详细信息。详细日志有助于审计、故障排查以及监控每个动作的执行情况。

Action log

Time ▾	Action	Media type	Recipient	Message	Status	Info
2025-08-15 07:58:30 AM	Report problems to Zabbix administrators	Email	Admin (Zabbix Administrator) *admin@example.com*	**Resolved in 1m 59s: Linux: Load average is too high (per CPU load over 1.5 for 5m)** Problem has been resolved at 10:58:30 on 2025.08.15 Problem name: Linux: Load average is too high (per CPU load over 1.5 for 5m) Problem duration: 1m 59s Host: Linux server Severity: Average Original problem ID: 45	Sent	
2025-08-15 07:56:30 AM	Report problems to Zabbix administrators	Email	Admin (Zabbix Administrator) *admin@example.com*	**Problem: Linux: Load average is too high (per CPU load over 1.5 for 5m)** Problem started at 10:56:30 on 2025.08.15 Problem name: Linux: Load average is too high (per CPU load over 1.5 for 5m) Host: Linux server Severity: Average Operational data: Load averages(1m 5m 15m): (6.256836 4.808105 3.398438), # of CPUs: 2 Original problem ID: 45	Sent	
2025-08-15 07:42:30 AM	Report problems to Zabbix administrators		Admin (Zabbix Administrator)		Failed	i

SLA 报告

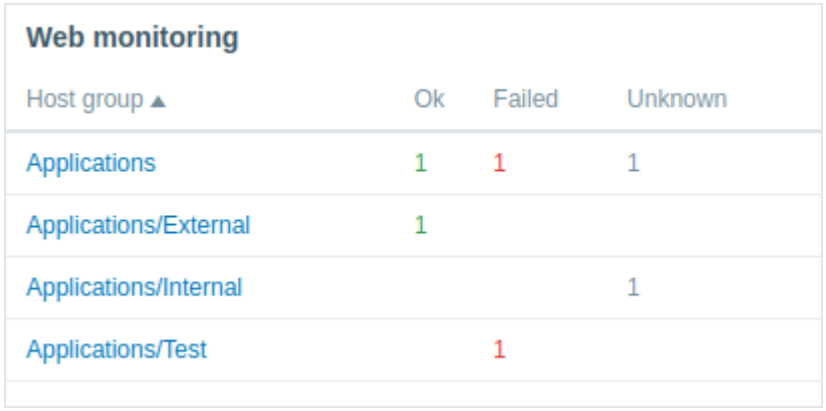
SLA 报告 小部件显示 SLA 报告，帮助您根据已定义的目标监控服务性能。

SLA report

Day	SLO	SLI	Uptime	Downtime	Error budget	Excluded downtimes
2025-09-04	99.9%	99.8437	5h 40m 50s	28s	-8s	
2025-09-03	99.9%	99.9479	23h 59m 15s	45s	41s	
2025-09-02	99.9%	99.9190	23h 58m 50s	1m 10s	16s	
2025-09-01	99.9%	99.9251	22h 58m 58s	1h 1m 2s	21s	2025-09-01 01:00 AM Maintenance: 1h
2025-08-31	99.9%	99.6863	23h 55m 29s	4m 31s	-3m 5s	
2025-08-30	99.9%	99.9190	23h 58m 50s	1m 10s	16s	
2025-08-29	99.9%	99.7095	23h 55m 49s	4m 11s	-2m 45s	
2025-08-28	99.9%	99.9421	23h 59m 10s	50s	36s	
2025-08-27	99.9%	99.7188	23h 55m 57s	3m 15s	-2m 37s	
2025-08-26	99.9%	99.7755	23h 56m 46s	3m 14s	-1m 48s	
2025-08-25	99.9%	99.9094	22h 58m 45s	1h 1m 15s	8s	2025-08-25 01:00 AM Maintenance: 1h

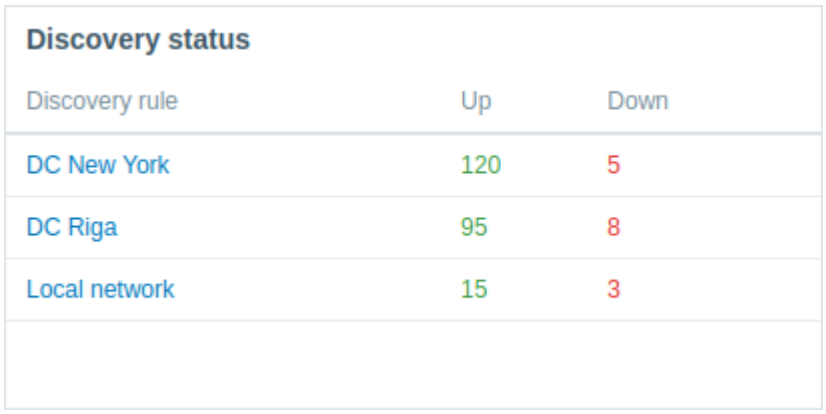
Web 监控

Web 监控 小组件显示活动 Web 场景的状态。



发现状态

Discovery status 小部件显示由已启用的网络发现规则发现的设备状态。
它可帮助您监控网络发现是否正常工作，以及处于在线或离线状态的设备数量。



仪表板实用工具和系统信息

这些小组件提供补充信息和工具，以增强您的仪表板。

系统信息

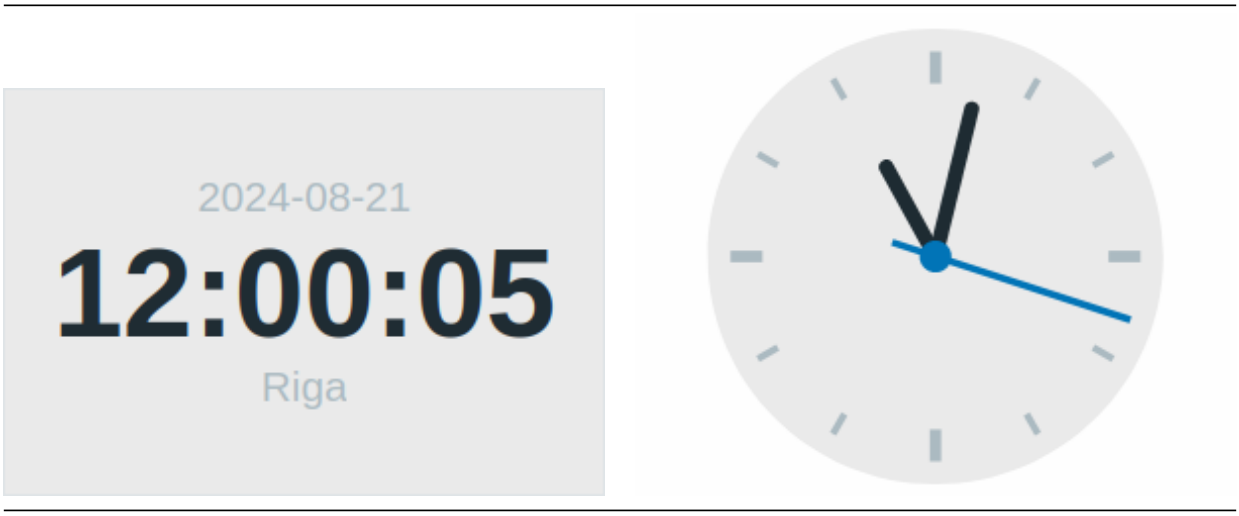
系统信息 小部件显示关键 Zabbix 服务器和系统数据的摘要，或高可用性节点的详细信息。

System information

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Zabbix server version	8.0.0	Up to date
Zabbix frontend version	8.0.0	Up to date
Software update last checked	2025-08-30	
Latest release	8.0.0	Release notes
Number of hosts (enabled/disabled)	2	2 / 0
Number of templates	312	
Number of items (enabled/disabled/not supported)	176	165 / 0 / 11
Number of triggers (enabled/disabled [problem/ok])	89	89 / 0 [4 / 85]
Number of users (online)	2	1
Required server performance, new values per second	2.68	
Global scripts on Zabbix server	Disabled	
High availability cluster	Disabled	

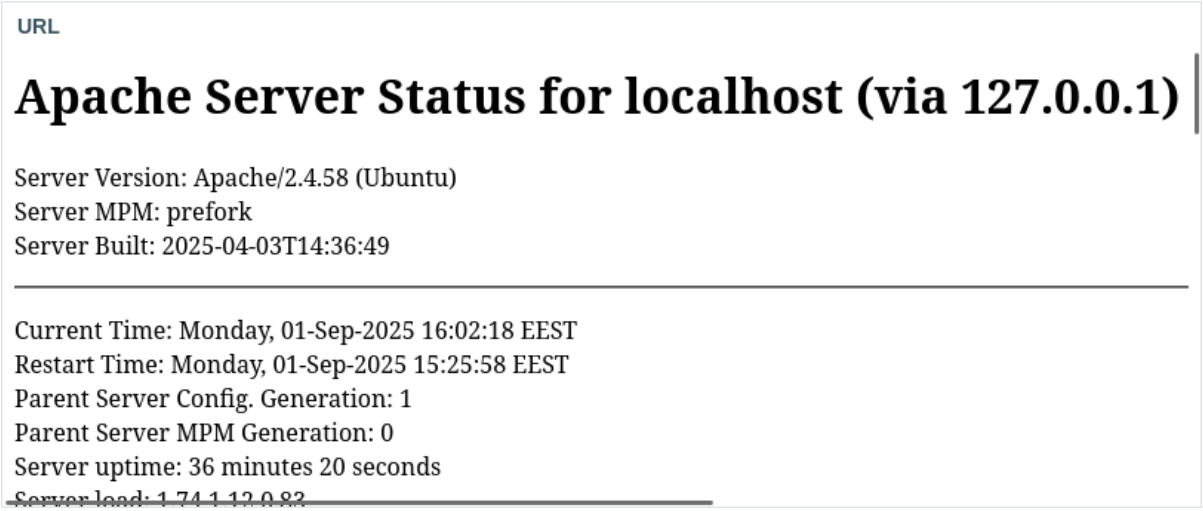
时钟

Clock 小组件以模拟或数字格式显示本地时间、服务器时间或主机时间。



URL

URL 小组件显示从 URL 检索到的内容。



仪表板小部件参数

概述

本页介绍所有仪表板小部件通用的参数，以及允许多个小部件彼此之间或与仪表板交换配置数据的动态参数。

要查看每个小部件的特定参数，请使用侧边栏导航到相应的小部件页面。

通用参数

以下参数适用于每个小组件：

名称	输入小组件名称。
刷新间隔	配置默认刷新间隔。 根据小组件类型的不同，小组件的默认刷新间隔范围为不刷新到 *15* 分钟。例如： - *URL* 小组件为不刷新； - 操作日志小组件为 *1* 分钟； - 时钟小组件为 *15* 分钟。 可以为所有用户设置默认刷新间隔。将仪表板切换到编辑模式，点击编辑小组件按钮，然后从下拉列表中选择所需的刷新间隔。 每个用户也可以设置自己的小组件刷新间隔。在仪表板的查看模式下，点击小组件上的三个点 按钮，然后从下拉列表中选择所需的刷新间隔。请注意，用户自定义的刷新间隔优先于小组件设置，并且即使修改了小组件设置，该刷新间隔仍会保留。
显示标题	选中复选框可永久显示小组件标题。 取消选中后，标题将被隐藏以节省空间，仅当鼠标悬停在小组件上时才会显示（查看模式和编辑模式下均如此）。将小组件拖动到新位置时，标题也会半透明显示。

动态参数

多个小部件参数（例如 主机、覆盖主机、时间段）支持动态数据源。
您无需配置静态值，而是可以将这些参数链接到其他小部件或仪表板本身。
这样，小部件就会根据仪表板其他部分的选择或广播自动更新。

小组件兼容性

某些小组件可以将配置数据广播给其他小组件，某些小组件可以侦听数据，还有一些小组件两者都可以。例如：

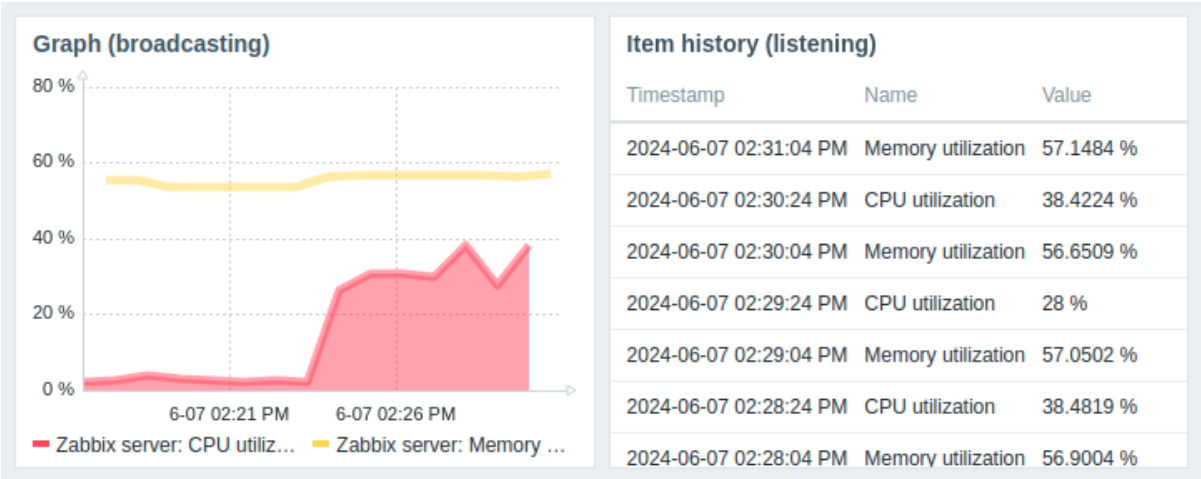
- 操作日志小组件只能从 图形、图形（经典）和 图形原型小组件获取时间段数据。
- 地理地图小组件可以将主机数据广播给侦听该数据的小组件（蜂窝图、热门监控项等），也可以侦听广播该数据的小组件发送的主机组和主机数据（蜂窝图、问题主机等）。
- 时钟小组件既不能广播任何数据，也不能侦听任何数据。

下表概述了各个小组件的广播和侦听能力。

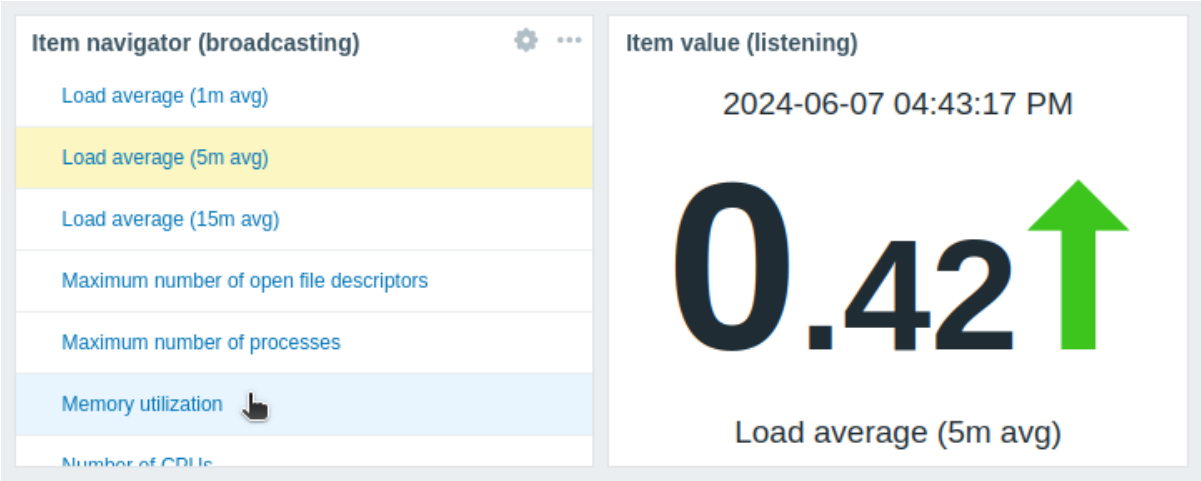
小组件	广播	侦听
操作日志	-	时间段
时钟	-	-
发现状态	-	-
收藏图形	-	-
收藏地图	-	-
仪表盘	-	主机，监控项
地理地图	主机	主机组，主机
图形	时间段，监控项	主机，监控项，时间段
图形（经典）	时间段	主机，监控项，图形，时间段
图形原型	时间段	主机，时间段
蜂窝图	主机，监控项	主机组，主机
主机可用性	-	主机组
主机卡片	-	主机
主机导航器	主机	主机组
监控项卡片	-	主机，监控项，时间段
监控项历史	监控项	主机，时间段
监控项导航器	监控项	主机组，主机
监控项值	-	主机，监控项，时间段
地图	主机组，主机	地图
地图导航树	地图	-
饼图	-	监控项，时间段
问题主机	主机组	主机组，主机
问题	事件	主机组，主机
按严重性分类的问题	主机组	主机组，主机
散点图	时间段，监控项	主机，监控项，时间段
SLA 报告	-	-
系统信息	-	-
热门主机	主机	主机组，主机
热门监控项	-	主机组，主机
热门触发器	-	时间段
触发器概览	-	主机组，主机
URL	-	主机
Web 监测	主机组	主机组，主机

各个小组件在向其他小组件广播数据的方式上有所不同。

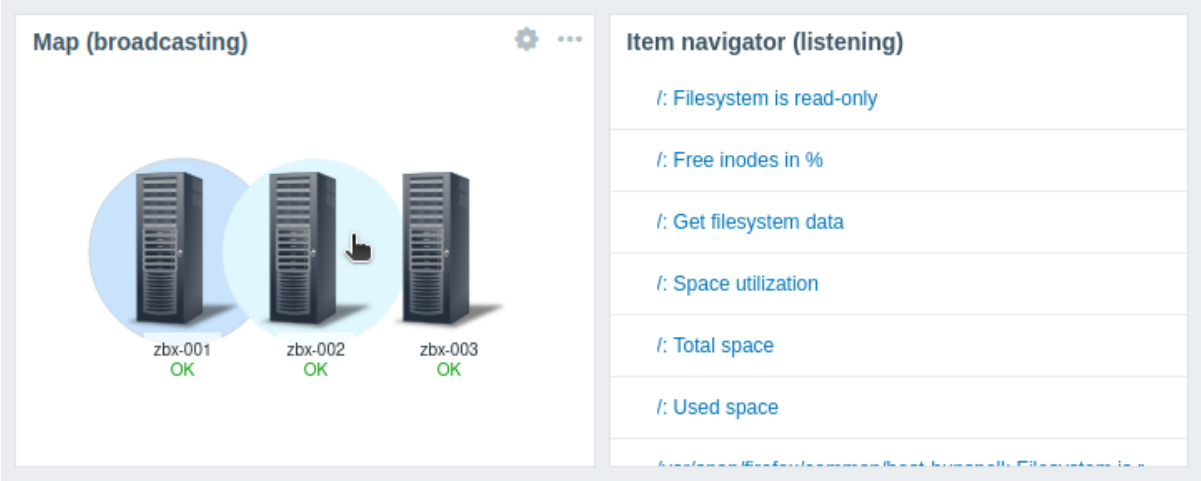
支持广播的小组件在创建后会自动开始广播。例如，图形小组件会立即向正在侦听的小组件广播时间段数据。



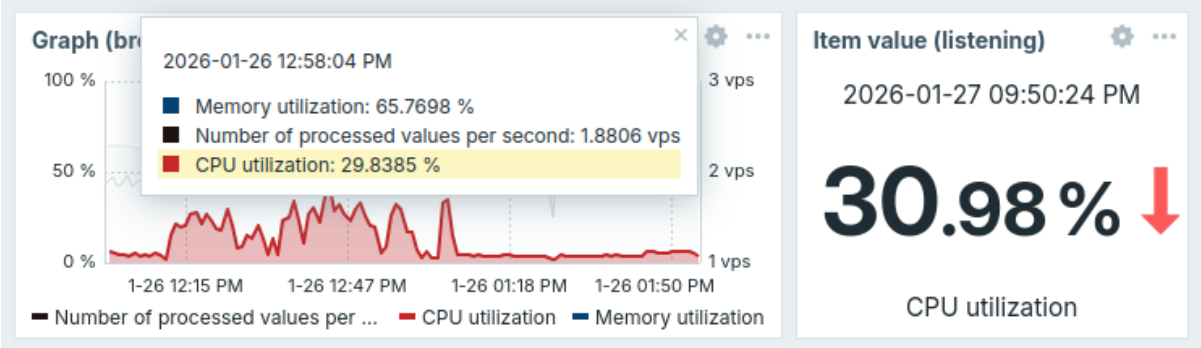
支持实体选择的小组件会自动广播第一个可用实体的数据。例如，监控项导航器小组件会广播其监控项列表中第一个监控项的数据。当选择其他监控项时，它会广播该监控项的数据。鼠标悬停时，监控项会以浅蓝色高亮显示；选中时，则会以黄色高亮显示。



地图小组件的行为类似，会广播最靠近小组件可见区域左上角的第一个元素的数据（对于 地理地图小组件，则是最靠近中心的元素）。当选择其他元素时，它会广播该元素的数据。鼠标悬停时，元素会以浅蓝色高亮显示；选中时，则会以深蓝色高亮显示。

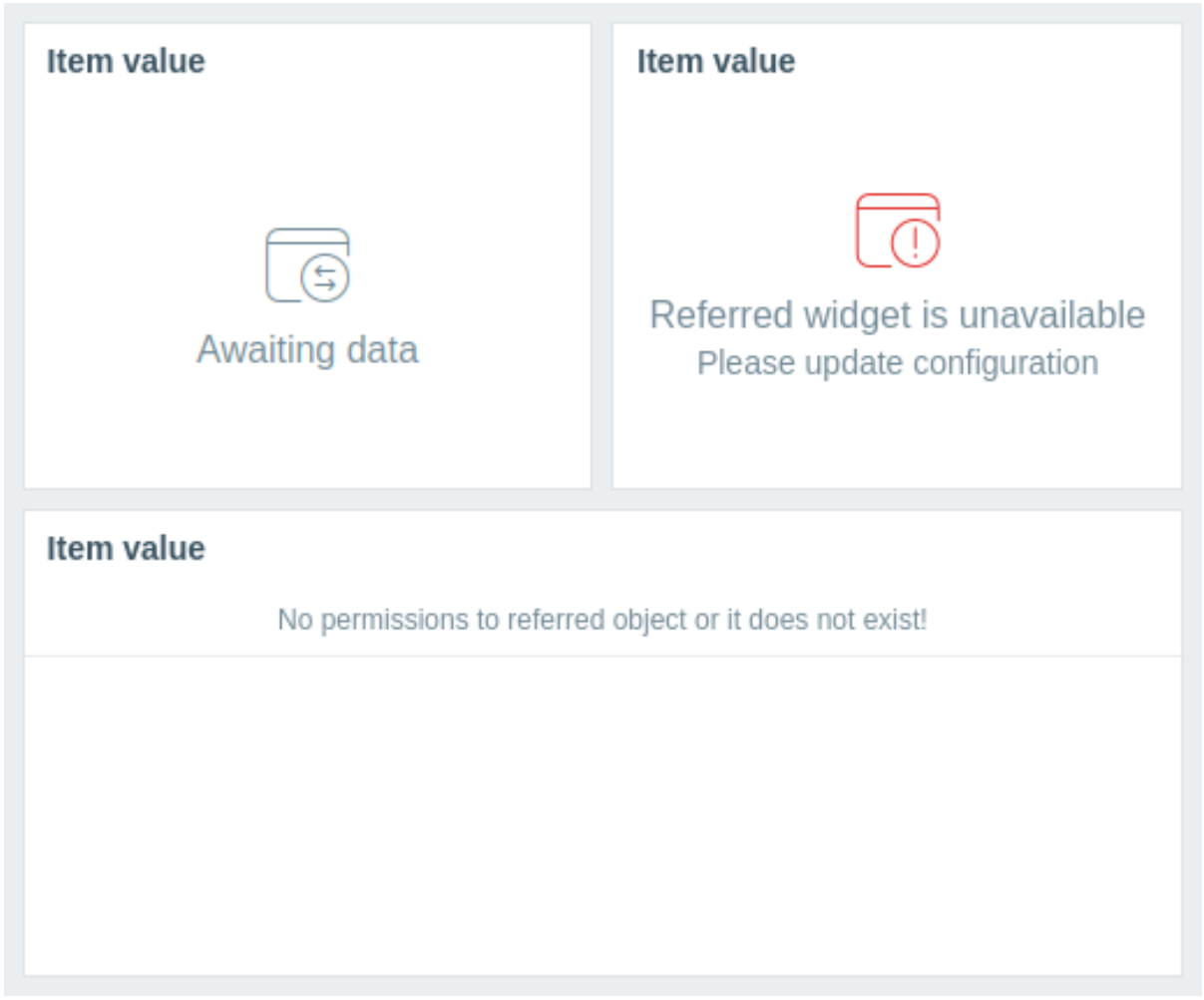


像 图形和 散点图这样的小组件，在其工具提示中选择某个监控项时，也可以广播该监控项的数据：



当小组件侦听来自其他小组件的数据时，也有特定的行为：

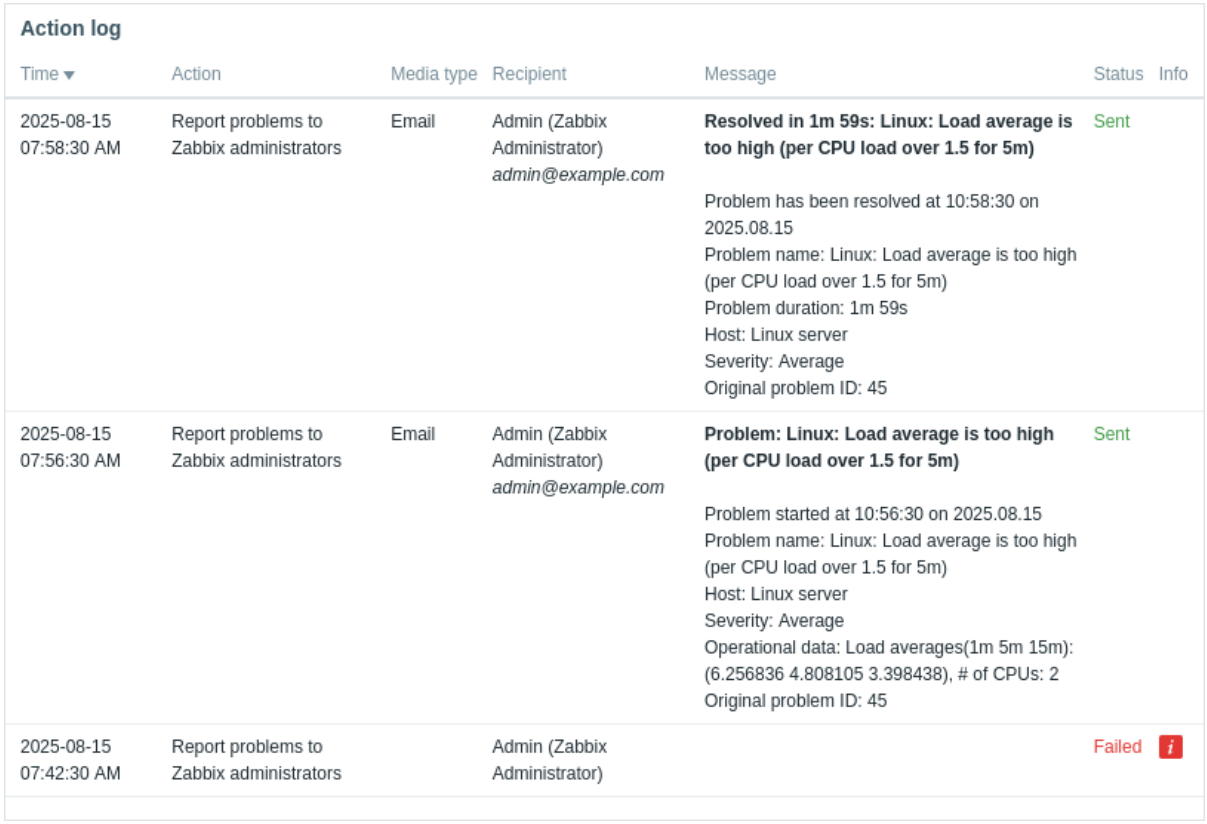
- 如果数据源小组件未广播数据，侦听小组件将进入 等待数据状态。
- 如果数据源小组件已被删除、被不兼容的小组件替换，或被移动到另一个仪表板页面，侦听小组件将进入 引用的小组件不可用状态。
- 如果数据源（小组件或仪表板）中指定的主机缺少侦听小组件中配置的实体（监控项、图形、地图等），或者用户没有访问该主机的权限，则侦听小组件会显示以下消息：*"No permissions to referred object or it does not exist!"*



操作日志

概述

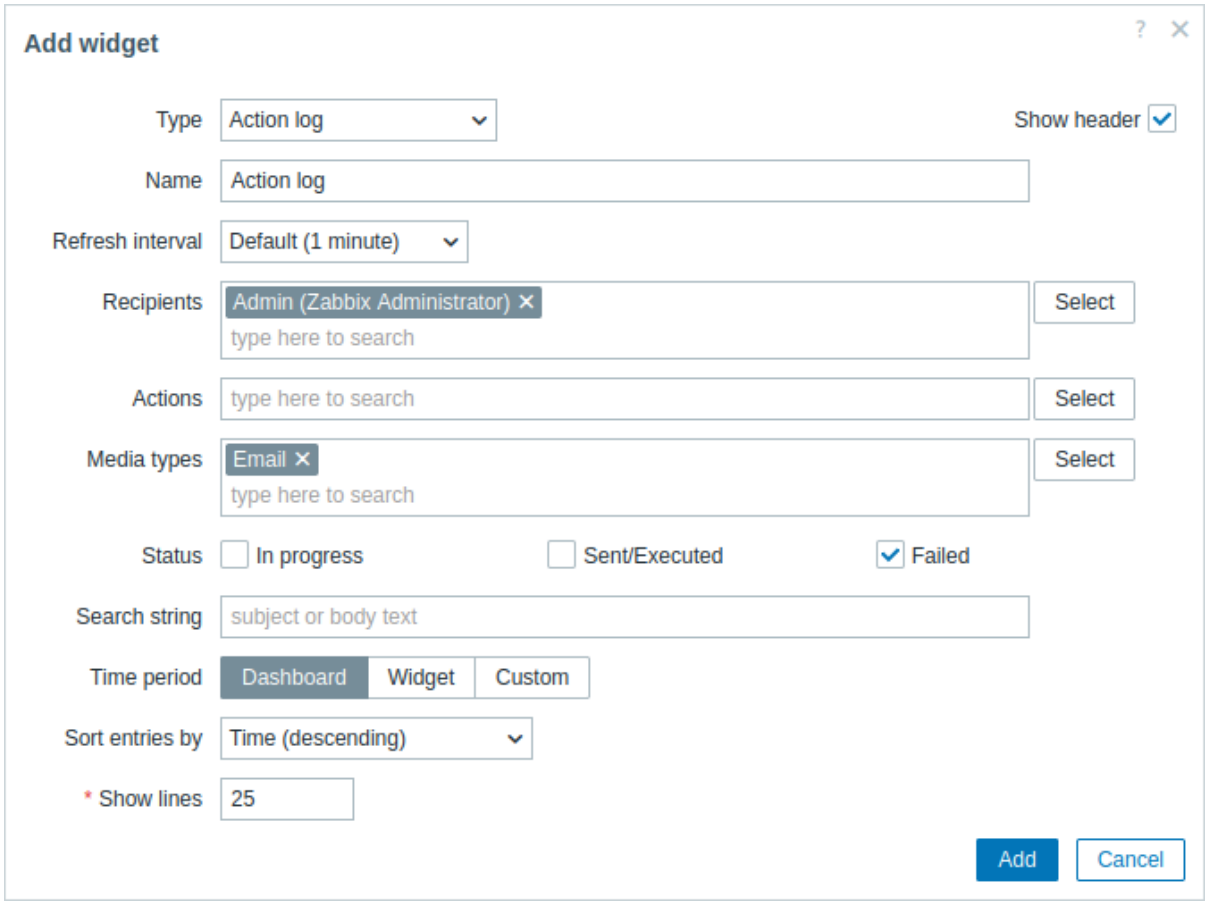
Action log 小组件显示在一个动作中执行的操作（通知、远程命令）的详细信息。详细日志有助于审计、故障排查以及监控每个动作的执行情况。



它显示与 *Reports* > *Action log* 相同的数据，最多可显示 1000 条记录。

配置

要进行配置，请选择 *Action log* 作为类型：



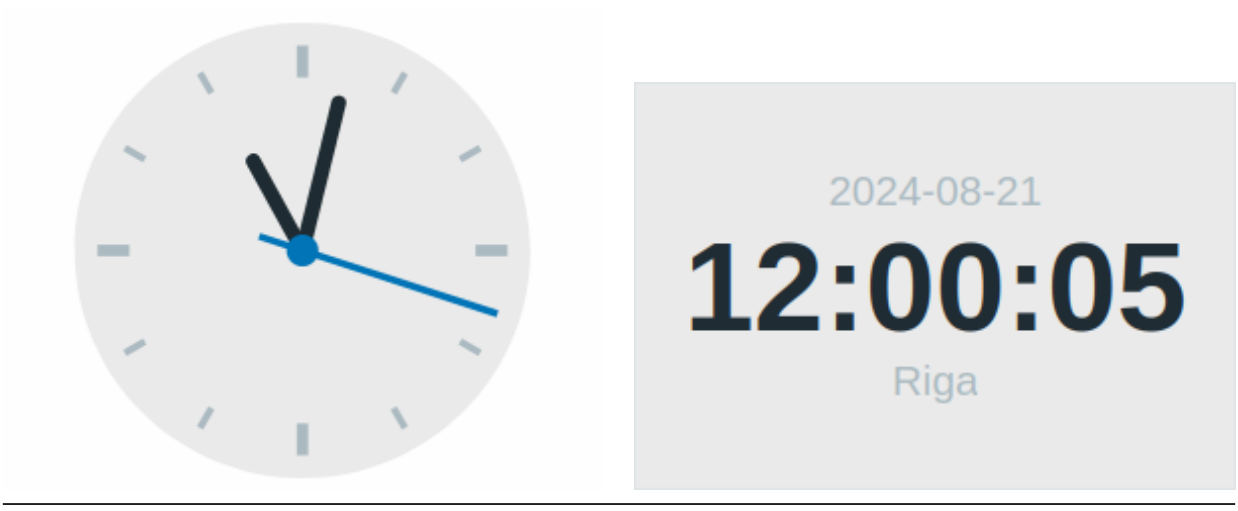
除了所有小组件通用的常见参数外，还可以设置以下特定选项：

Recipients	按收件人筛选条目。此字段支持自动补全，因此开始输入收件人名称时，将提供匹配收件人的下拉列表。 如果未选择任何收件人，则将显示所有收件人的动作操作详细信息。
Actions	按动作筛选条目。此字段支持自动补全，因此开始输入动作名称时，将提供匹配动作的下拉列表。如果未选择任何动作，则将显示所有动作的动作操作详细信息。
Media types	按介质类型筛选条目。此字段支持自动补全，因此开始输入介质类型名称时，将提供匹配介质类型的下拉列表。如果未选择任何介质类型，则将显示所有介质类型的动作操作详细信息。
Status	勾选复选框可按相应状态筛选条目： **In progress** - 显示正在进行中的动作操作； **Sent/Executed** - 显示已发送通知或已执行的动作操作； **Failed** - 显示失败的动作操作。
Search string	按消息/远程命令的内容筛选条目。如果在此处输入字符串，则仅显示消息/远程命令中包含该字符串的动作操作。宏不会被解析。
Time period	按时间段筛选条目。为时间段选择数据源： **Dashboard** - 使用仪表板时间段选择器； **Widget** - 使用兼容的小组件（在 *Widget* 参数中设置）； **Custom** - 使用在 *From* 和 *To* 参数中设置的自定义时间段；如果已设置，则会在小组件右上角显示一个时钟图标，鼠标悬停时会显示所设时间。 请注意，无论小组件的 *Time period* 配置如何，兼容的小组件仍可将其用作时间段的数据源。
Widget	输入或选择一个兼容的小组件，作为时间段的数据源。 如果 *Time period* 设置为 “Widget”，则此参数可用。
From	输入或选择时间段的开始时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 如果 *Time period* 设置为 “Custom”，则此参数可用。
To	输入或选择时间段的结束时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 如果 *Time period* 设置为 “Custom”，则此参数可用。
Sort entries by	按以下方式对条目排序： **Time**（降序或升序）； **Type**（降序或升序）； **Status**（降序或升序）； **Recipient**（降序或升序）。
Show lines	设置小组件中显示的动作日志行数。

时钟

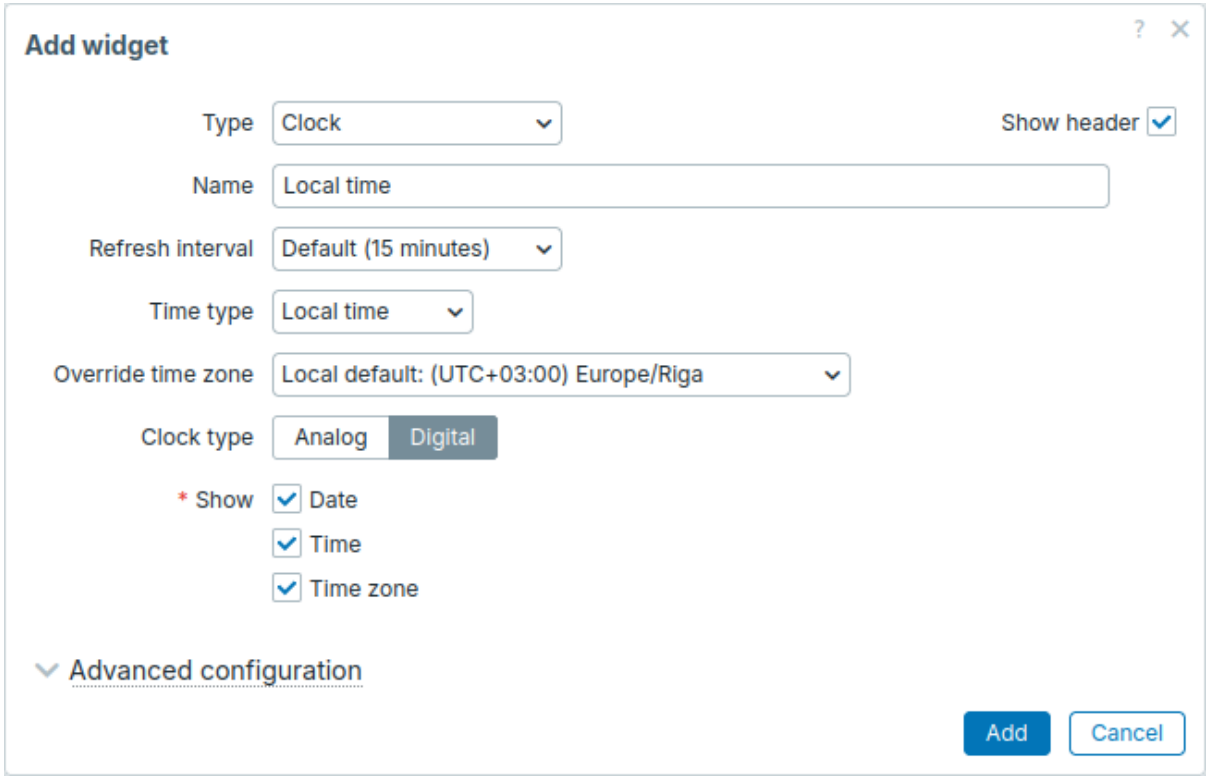
概述

Clock 小组件以模拟或数字格式显示本地时间、服务器时间或主机时间。



配置

要进行配置，请选择 *Clock* 作为类型：



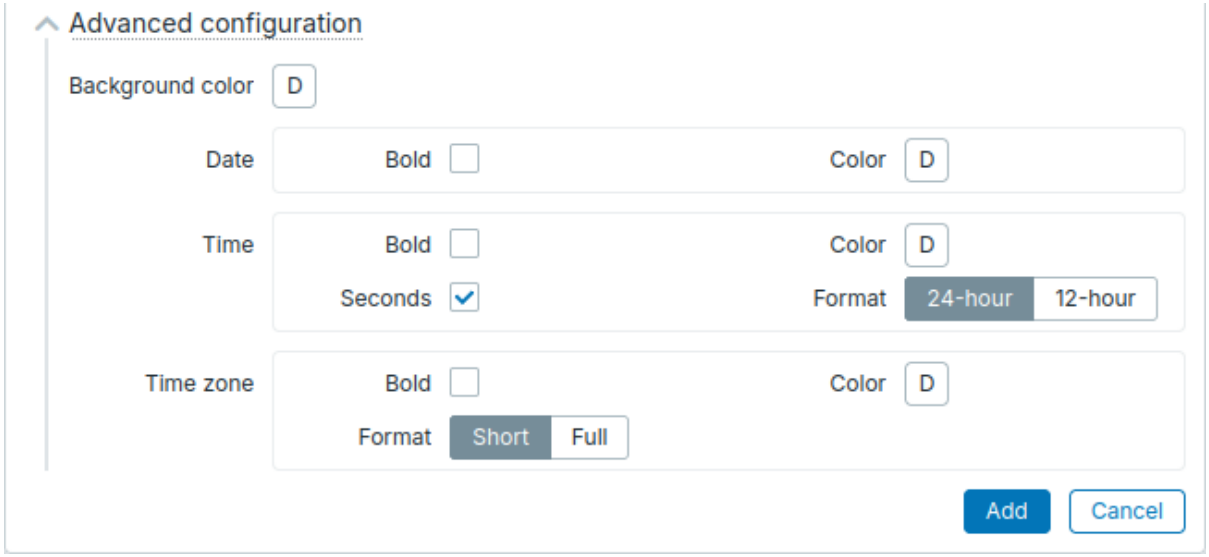
除了适用于所有小组件的通用参数外，还可以设置以下特定选项：

Time type	选择时间来源： **Local time** - 运行浏览器的设备的系统时间； **Server time** - 为 Zabbix 用户全局设置或单独设置的时区； **Host time** - 从主机监控项中获取的时间。
Item	选择包含`system.localtime[local]`监控项键值的监控项，以显示主机时间。 当 *Time type* 设置为 “Host time” 时，此参数可用。
Clock type	选择时钟类型（模拟或数字）。
Override time zone	选择时区。 对于 *Local time* 和 *Server time* 时间类型，此参数可用。
Show	选择要在数字时钟中显示的一个或多个信息单位（日期、时间、时区）。 单位大小会根据显示的单位、小组件大小和显示缩放比例自动调整。 当 *Clock type* 设置为 “Digital” 时，此参数可用。
Advanced configuration	点击 *Advanced configuration* 标题，以显示数字时钟的高级配置选项。 当 *Clock type* 设置为 “Digital” 时，此部分可用。

高级配置

高级配置选项位于可折叠的高级配置部分中，并且仅适用于在显示字段中选中的元素（见上文）。

此外，高级配置还允许您更改整个小组件的背景颜色。

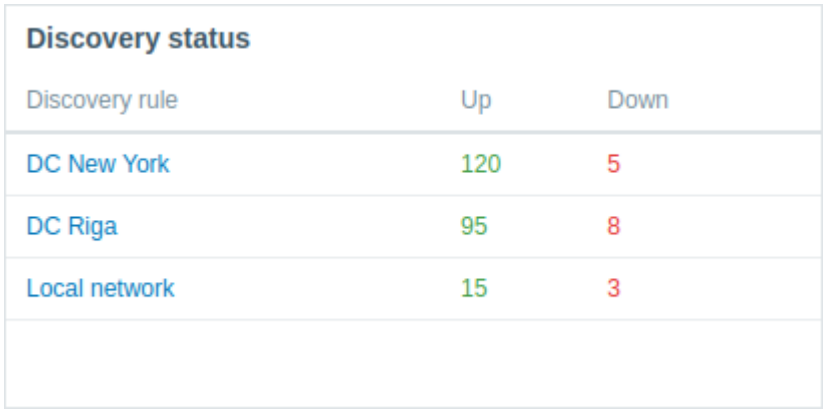


背景颜色		从颜色选择器中选择背景颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中点击使用默认值按钮。
日期		当在显示参数中选中日期复选框时，此部分可用。
	粗体	选中此复选框可将日期以粗体显示。
	颜色	从颜色选择器中选择日期颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中点击使用默认值按钮。
时间		当在显示参数中选中时间复选框时，此部分可用。
	粗体	选中此复选框可将时间以粗体显示。
	颜色	从颜色选择器中选择时间颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中点击使用默认值按钮。
	秒	选中此复选框可显示秒。否则，仅显示小时和分钟。
	格式	选择显示 24 小时制或 12 小时制时间。
时区		当在显示参数中选中时区复选框时，此部分可用。
	粗体	选中此复选框可将时区以粗体显示。
	颜色	从颜色选择器中选择时区颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中点击使用默认值按钮。
	格式	选择以简短格式（例如 New York）或完整格式（例如 (UTC-04:00) America/New York）显示时区。

发现状态

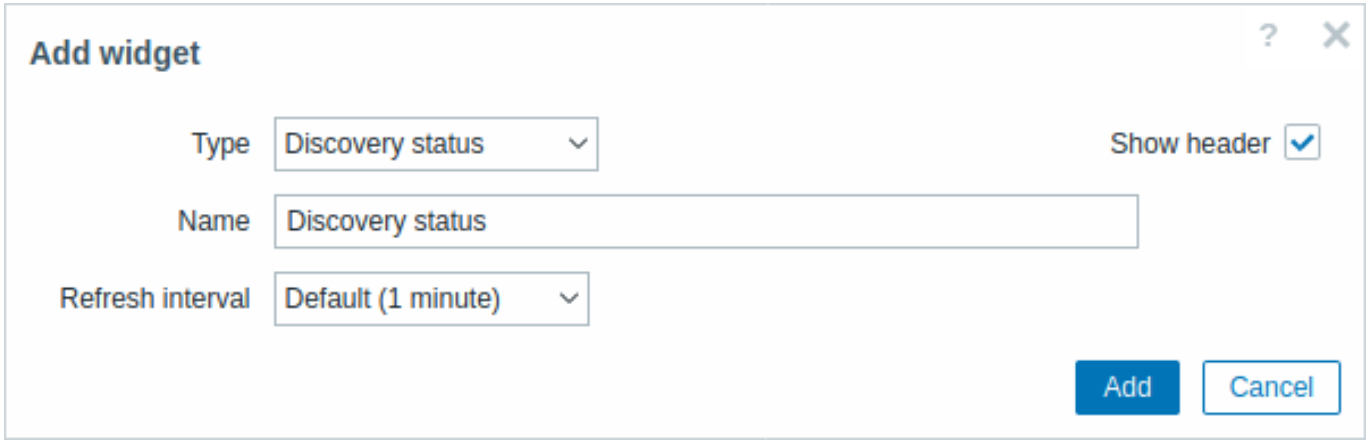
概述

Discovery status 部件显示由已启用的网络发现规则发现的设备状态。
它有助于监控网络发现是否正常工作，以及处于在线或离线状态的设备数量。



配置

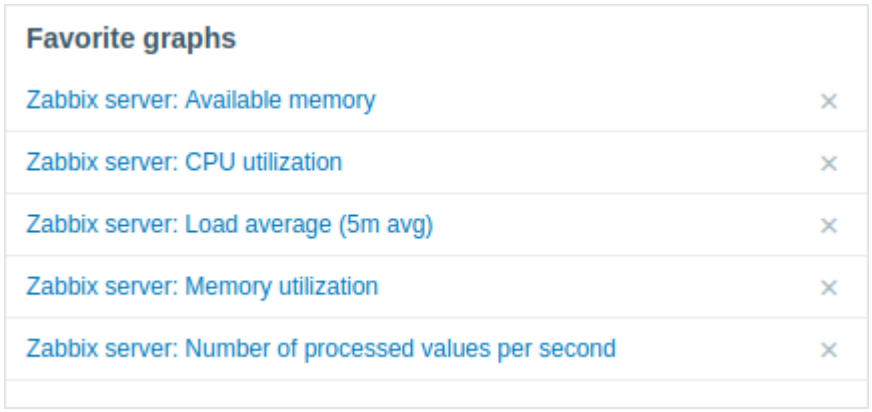
此小组件的所有配置参数与所有小组件的通用参数相同。



收藏图形

概述

收藏图形小组件会按字母顺序显示当前用户标记为收藏的图形快捷列表。



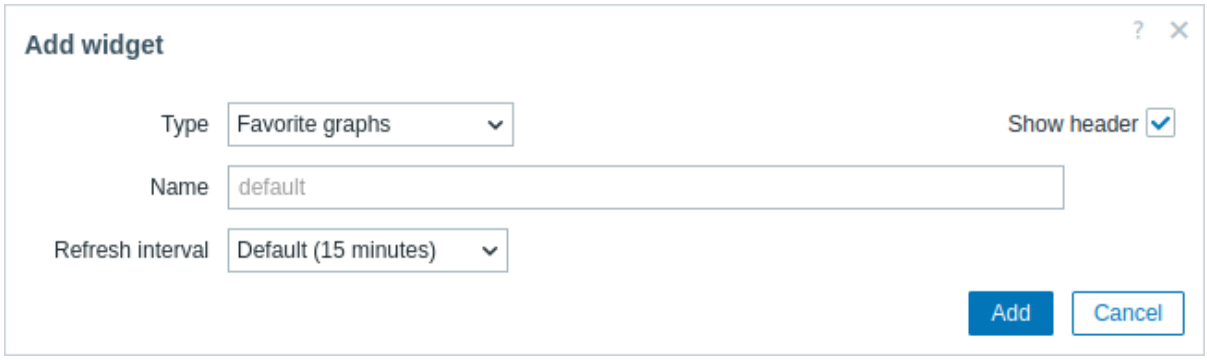
点击 x 会将该图形从列表和收藏中移除。

在 监控 > 最新数据 中查看监控项图形时，点击  添加到收藏按钮，即可将图形标记为收藏。



配置

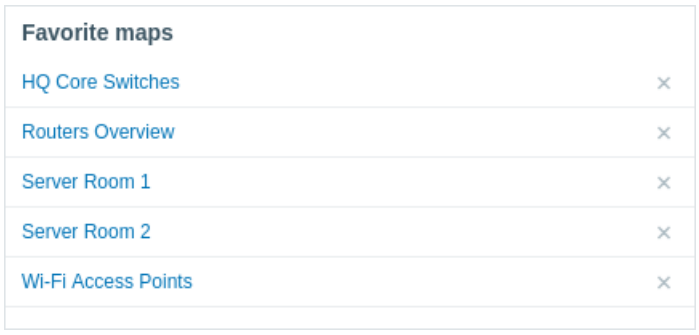
此小部件的所有配置参数均为所有小部件通用的通用参数。



收藏地图

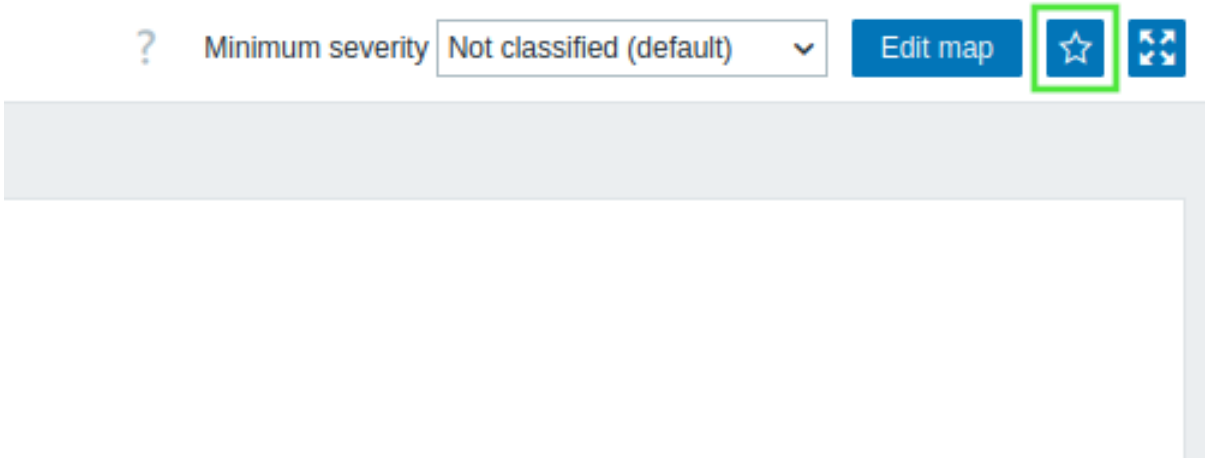
概述

收藏的地图小部件显示当前用户标记为收藏的地图快捷方式列表，并按字母顺序排序。



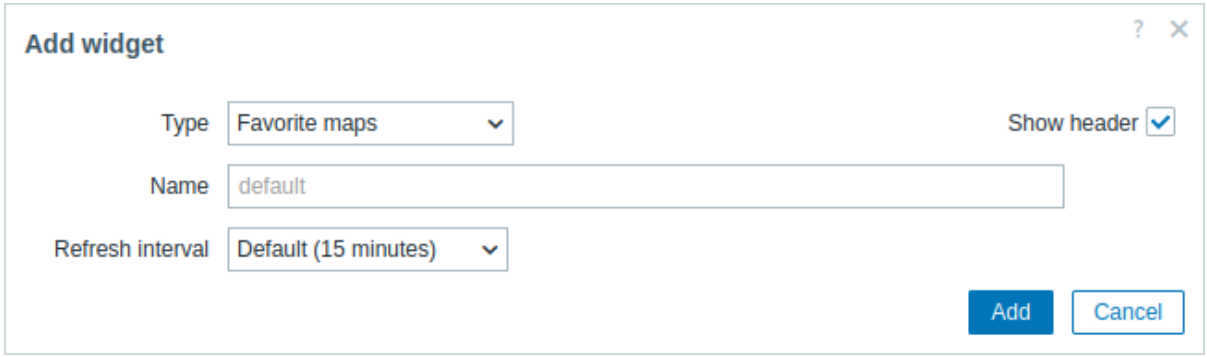
点击 x 可将该地图从收藏中移除。

在 监控 > 地图 中查看地图时，点击  添加到收藏按钮，即可将地图标记为收藏。



配置

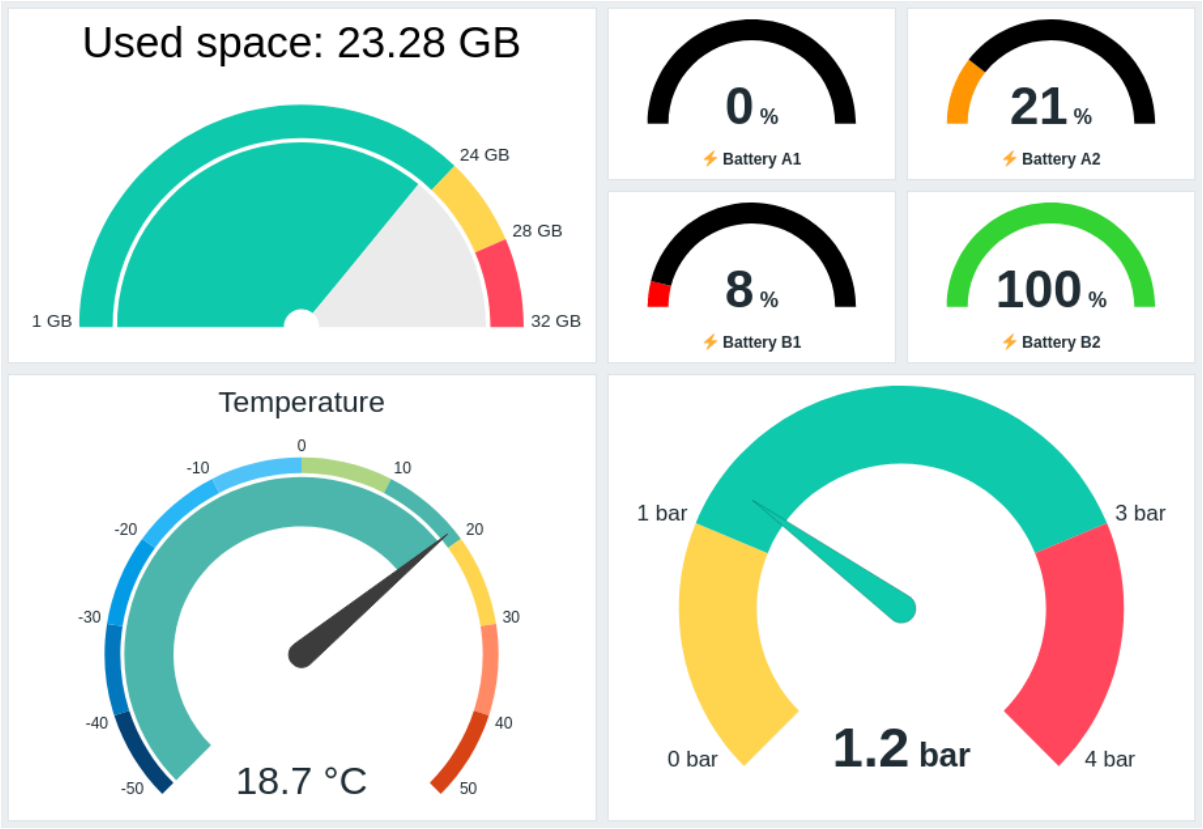
此小部件的所有配置参数与所有小部件的通用参数相同。



仪表盘

概述

仪表盘小组件以仪表的形式显示单个数值型监控项的值。它适用于关注重要指标、可视化阈值以及检测突发变化。



您可以将小组件配置为显示以下内容：

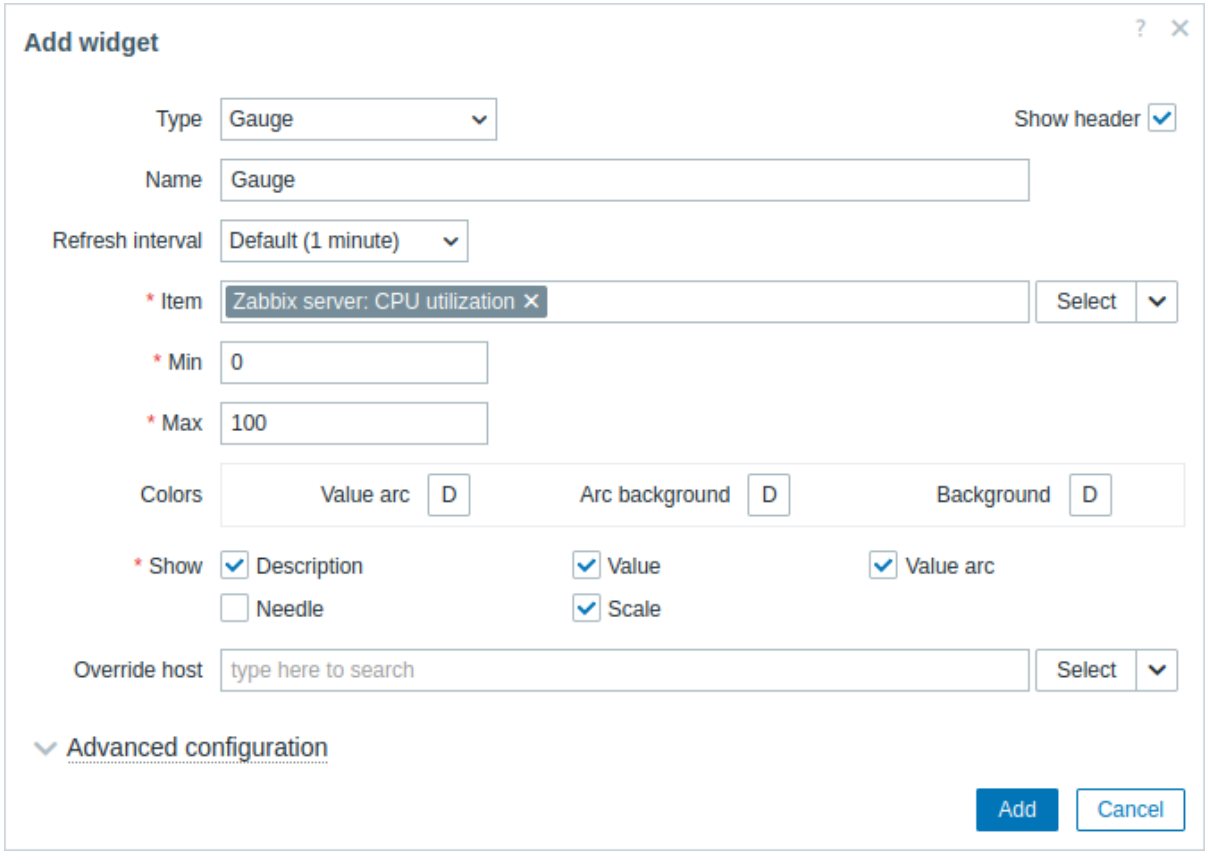
- 监控项描述（已用空间：{ITEM.LASTVALUE}、温度）
- 监控项值和单位（21 %、18.7 °C、1.2 bar）
- 仪表刻度（1GB/24GB/28GB/32GB、-50/-40/-30/等）
- 仪表弧线（仪表值弧线和仪表阈值弧线）
- 仪表指针

单击小组件可打开该监控项的简单图形。

将鼠标悬停在小组件上时，会显示小组件的操作菜单 ⋯。在该菜单中选择下载图像选项，即可将仪表盘小组件中显示的信息下载为 PNG 图像。根据浏览器设置，图像可能会自动保存到 `Downloads` 文件夹，或者打开保存对话框，以便您选择保存位置。

配置

要进行配置，请选择 *Gauge* 作为类型：

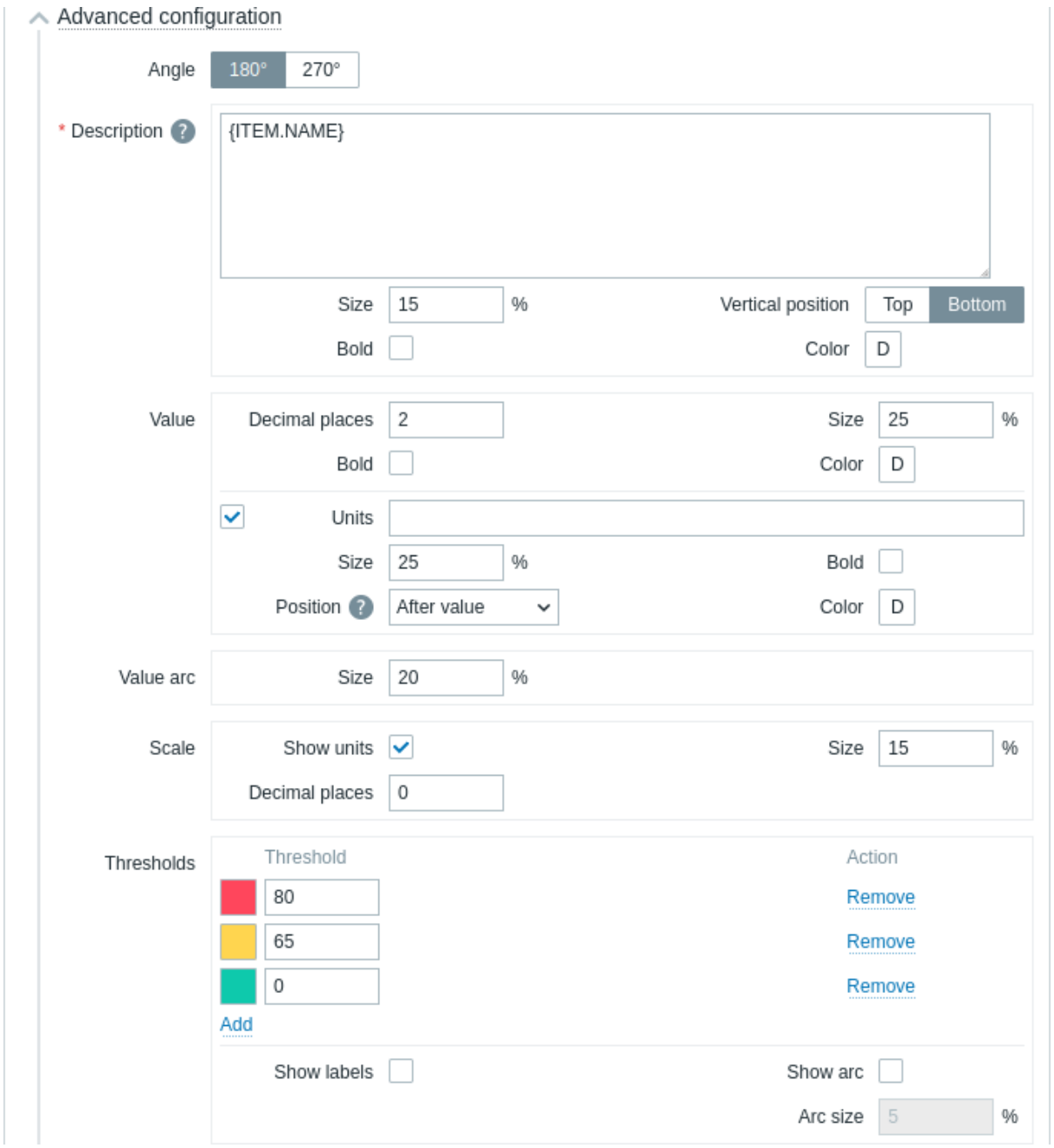


除了所有部件通用的常见参数外，您还可以设置以下特定选项：

Item	选择监控项。 或者，选择一个兼容的部件作为监控项的数据源。 此字段支持自动补全，因此开始输入监控项名称时，将显示包含匹配监控项的下拉列表。 请注意，您只能选择返回数值型数据的监控项（二进制数据除外）。
Min	输入仪表的最小值。 支持后缀（例如"1d"、"2w"、"4K"、"8G"）。支持值映射。
Max	输入仪表的最大值。 支持后缀（例如"1d"、"2w"、"4K"、"8G"）。支持值映射。
Colors	从颜色选择器中选择颜色： **Value arc** - 选择仪表值弧的颜色； **Arc background** - 选择仪表值弧和仪表阈值弧的背景颜色； **Background** - 选择部件背景颜色。 "D" 表示默认颜色，该颜色取决于前端主题。如果设置了*Thresholds*，则 **Value arc** 的默认颜色取决于阈值颜色。要恢复为默认颜色，请单击颜色选择器中的 *Use default* 按钮。
Show	勾选复选框以显示相应的仪表元素——描述、值、值弧、指针、刻度（仪表弧起始和结束位置的最小值和最大值）。取消勾选则隐藏。必须至少选择一个元素。 请注意，只有在显示仪表值弧或仪表阈值弧（请参见高级配置选项）时，才能显示仪表指针和刻度。另请注意，如果显示仪表指针，则值显示在指针下方；如果隐藏指针，则值与仪表弧底部对齐。
Override host	选择一个兼容的部件或仪表板主机选择器作为主机的数据源。 在模板仪表板上配置该部件时，此参数不可用。
Advanced configuration	单击 *Advanced configuration* 标题以显示高级配置选项。您还可以在此调整 *Show* 字段中选择的仪表元素。

高级配置

在可折叠的 高级配置中提供了高级配置选项：



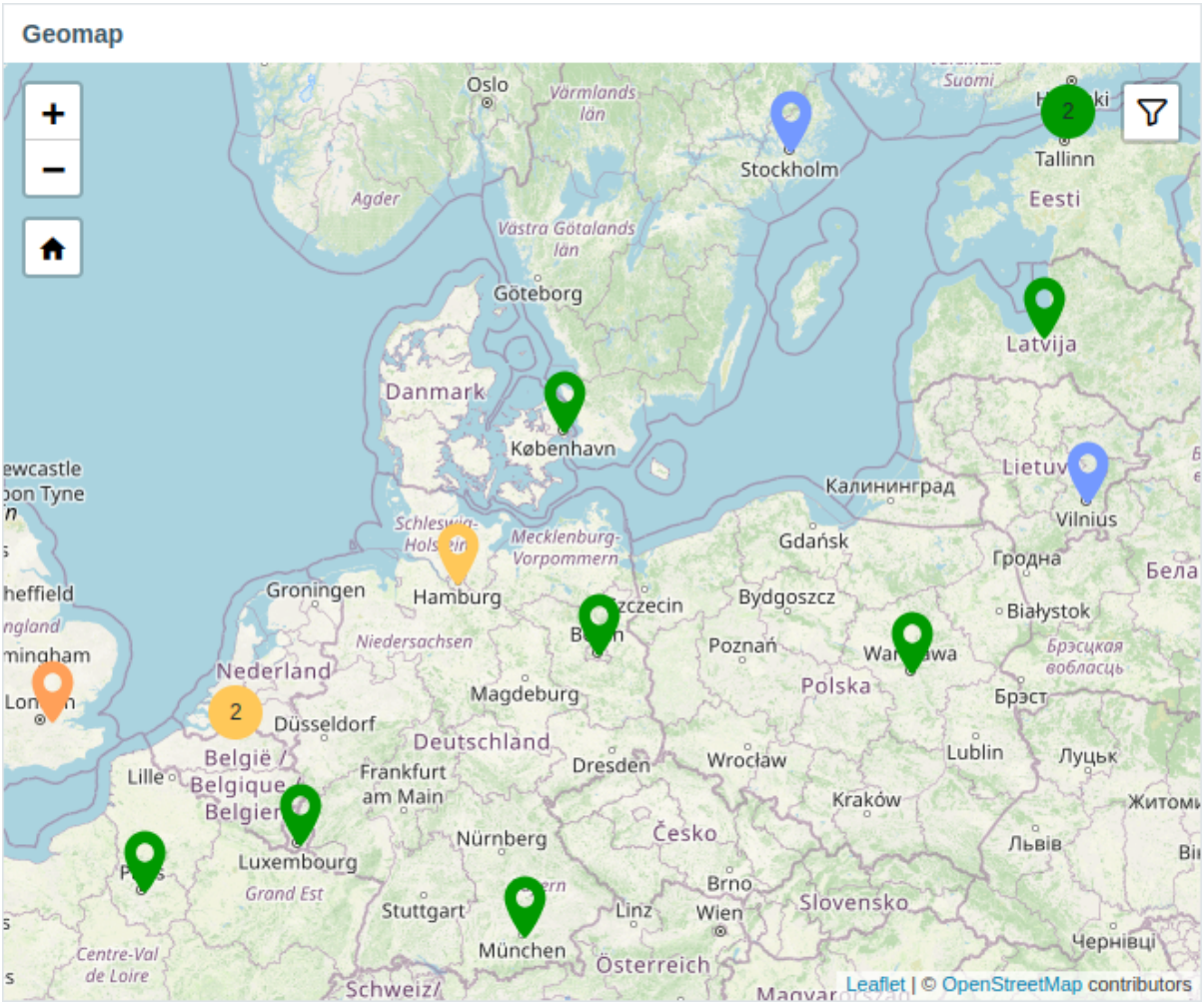
角度	选择仪表角度（180° 或 270°）。
说明	
说明	输入监控项说明。此说明可能会覆盖默认监控项名称。 支持多行说明。可以结合使用文本和支持的宏。 支持 {HOST.*}、{ITEM.*}、{INVENTORY.*} 和用户宏。
大小	输入监控项描述的字体大小高度（以百分比表示，相对于组件总高度）。
垂直位置	选择监控项描述的垂直位置（顶部或底部，相对于仪表弧）。
粗体	选中复选框以粗体显示监控项描述。
颜色	从颜色选择器中选择监控项描述颜色。 “D” 代表默认颜色，取决于前端主题。要返回默认颜色，请单击颜色选择器中的使用默认按钮。
值	
小数位数	输入要与值一起显示的小数位数。 此选项仅影响返回 [数字（浮点数）]（/manual/config/items/item#configuration）数据的监控项。
大小	输入值的字体大小高度（以百分比表示，相对于仪表弧高度）。
粗体	选中复选框以粗体显示值。
颜色	从颜色选择器中选择值颜色。 “D” 代表默认颜色，取决于前端主题。要返回默认颜色，请单击颜色选择器中的使用默认按钮。
单位	
单位	选中复选框以显示监控项值的单位。 如果输入单位名称，它将覆盖监控项配置中设置的单位。
大小	输入监控项单位的字体大小高度（以百分比表示，相对于仪表弧高度）。

粗体	选中复选框以粗体显示监控项单位。
位置	选择监控项单位的位置（相对于监控项值的上方、下方、前方或后方）。 对于以下时间相关单位，此选项将被忽略：UNIX 时间、运行时间、s。
颜色	从颜色选择器中选择监控项单位颜色。 “D” 代表默认颜色，取决于前端主题。要返回默认颜色，请单击颜色选择器中的使用默认按钮。
值弧	
弧大小	输入仪表值弧的大小高度（以百分比表示，相对于仪表弧半径）。
针	
颜色	从颜色选择器中选择仪表针颜色。 “D” 代表默认颜色，取决于前端主题。如果设置了阈值，则针的默认颜色取决于阈值颜色。要返回默认颜色，请单击颜色选择器中的使用默认按钮。
比例	
显示单位	选中复选框以显示仪表最小值和最大值的单位。
大小	输入仪表最小值和最大值的字体大小高度（以百分比表示，相对于仪表弧高度）。
小数位数	输入仪表最小值和最大值显示的小数位数。 此选项仅影响返回数字（浮点数） 数据的监控项。
阈值	
阈值	单击 “添加” 可添加阈值，从颜色选择器中选择阈值颜色，然后指定数值。 保存时，阈值列表将按升序排列。 请注意，配置为阈值的颜色仅对数字监控项才会正确显示。 支持后缀（例如，“1d”、“2w”、“4K”、“8G”）。支持值映射。
显示标签	勾选复选框以将阈值显示为仪表刻度上的标签。
显示圆弧	勾选复选框以显示仪表阈值圆弧。
圆弧大小	输入仪表阈值圆弧的大小高度（以百分比表示，相对于仪表圆弧半径）。

地理地图

概述

Geomap 小部件在基于 Leaflet 的交互式地图上将主机显示为标记点，标记颜色表示每个主机最严重的问题。

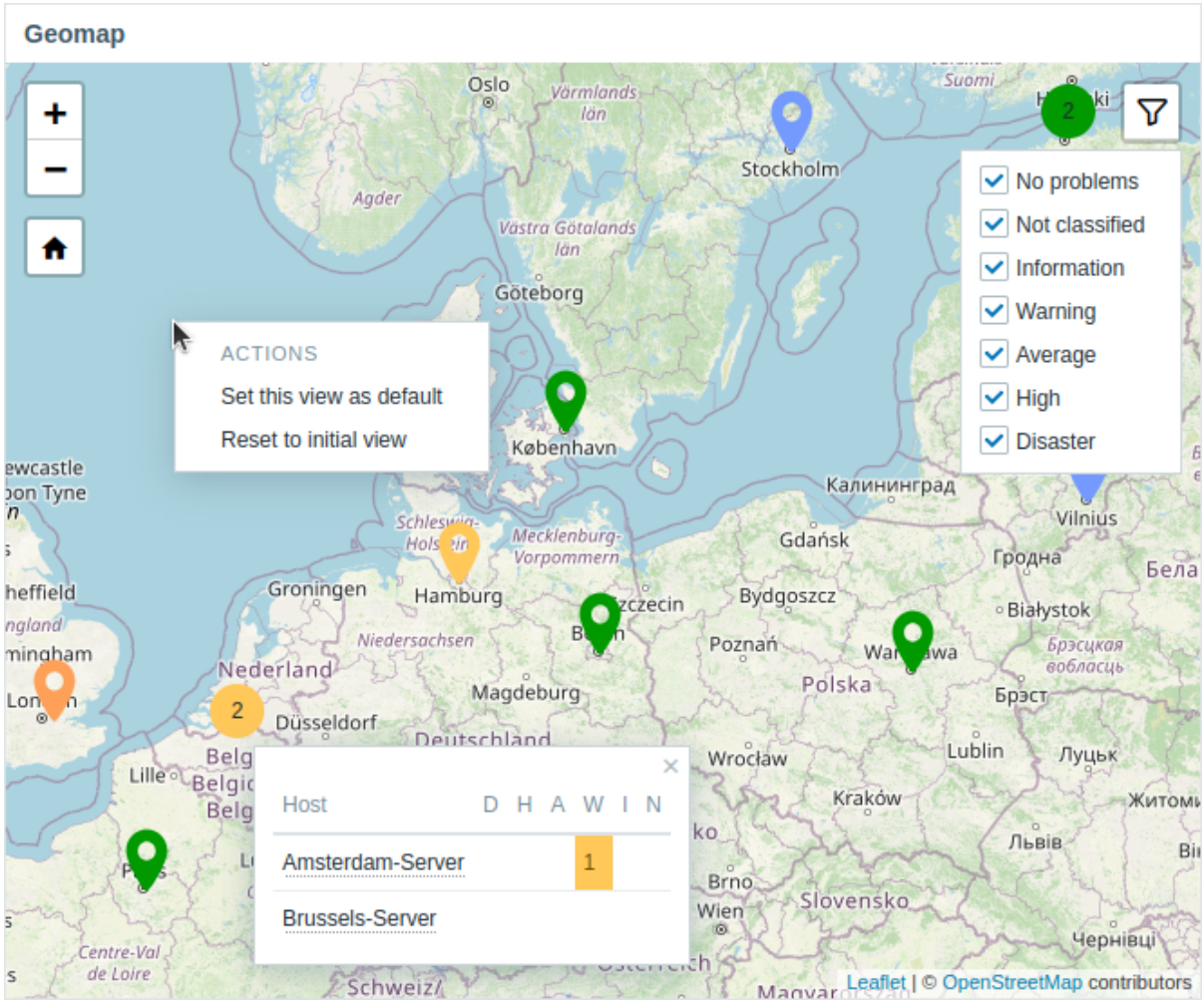


Note:
Zabbix 提供多个预定义的地图瓦片提供商，并允许您在 管理 > 常规 > 地理地图 中添加自定义提供商。

默认情况下，该小部件会显示其inventory中具有有效坐标（纬度 -90 到 90，经度 -180 到 180）的所有已启用主机。

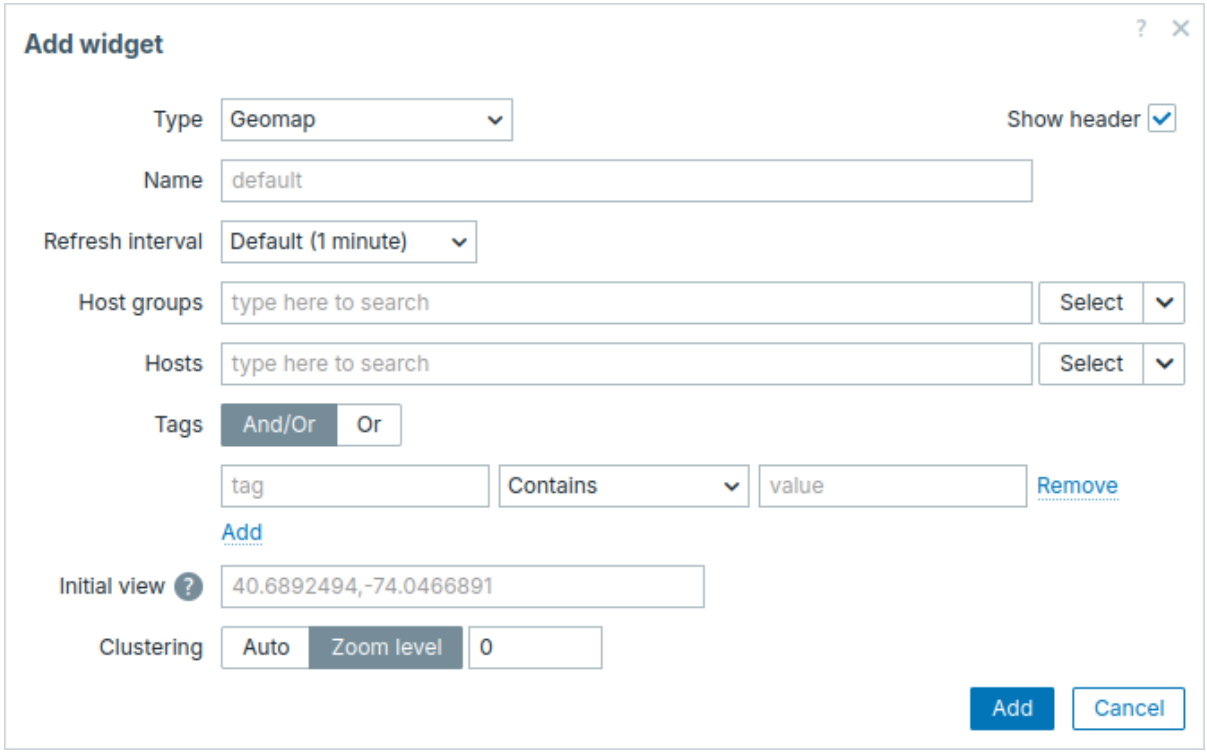
使用以下选项与该小部件交互：

- 单击标记点会显示主机名称和未解决的问题；单击主机名称会打开主机菜单。
- 使用右上角的筛选图标按问题严重性筛选主机。
- 右键单击地图可设置或重置默认视图；使用主页图标可返回默认视图。



配置

要添加该小组件，请选择 *Geomap* 作为类型。



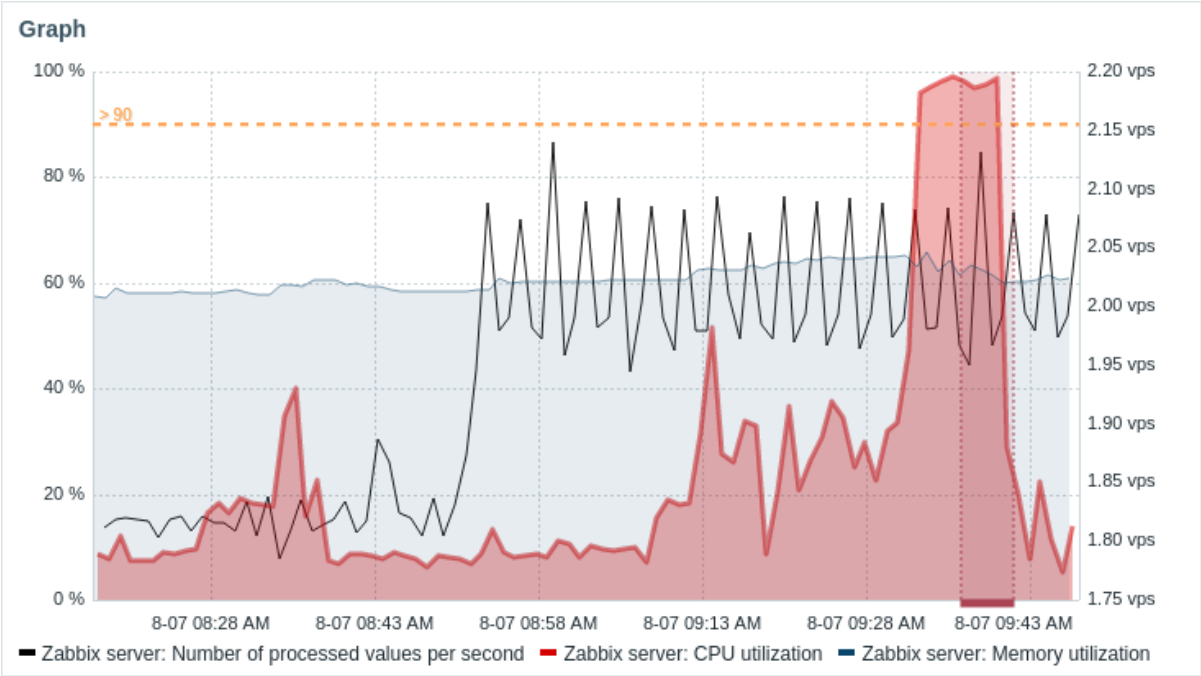
除了所有部件通用的常见参数外，您还可以设置以下特定选项：

主机组	选择要在地图上显示的主机组。 或者，选择一个兼容的部件作为主机组的数据源。 此字段支持自动补全，因此开始输入组名时，将提供匹配组的下拉列表。 如果在主机组和主机字段中都未选择任何内容，则将显示所有具有有效坐标的主机。 在模板仪表板上配置该部件时，此参数不可用。
主机	选择要在地图上显示的主机。 或者，选择一个兼容的部件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名时，将提供匹配主机的下拉列表。 如果在主机组和主机字段中都未选择任何内容，则将显示所有具有有效坐标的主机。 在模板仪表板上配置该部件时，此参数不可用。
标签	指定标签以限制部件中显示的主机数量。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下几种运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且其标签值包含输入的字符串（子串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且其标签值包含输入的字符串（子串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将按或条件分组； 或 - 只需满足一个条件即可。 在模板仪表板上配置该部件时，此参数不可用。
初始视图	指定部件初次加载时显示的中心坐标以及可选的缩放级别，格式为 `<latitude>,<longitude>,<zoom>`，各值以逗号分隔。 如果指定了初始缩放级别，则 Geomap 部件将按给定的缩放级别加载。否则，初始缩放级别将计算为特定瓦片提供方的最大缩放级别的一半。 如果设置了默认视图，则将忽略初始视图（见下文）。 示例： 40.6892494,-74.0466891 40.6892494,-122.0466891
聚类	指定如何将邻近的主机标记合并为一个带计数的单一标记： 自动 - 自动合并标记； 缩放级别 - 根据指定的缩放级别合并标记（0 - 最大缩小，30 - 最大放大）。例如，如果该参数设置为 0，则在任何缩放级别下都不会合并标记；如果设置为 15，则从最大缩小开始直到缩放级别 15 都会合并标记；如果设置为 30，则在所有缩放级别下都会合并标记。 请注意，最大放大级别由地图瓦片提供方决定。

图表

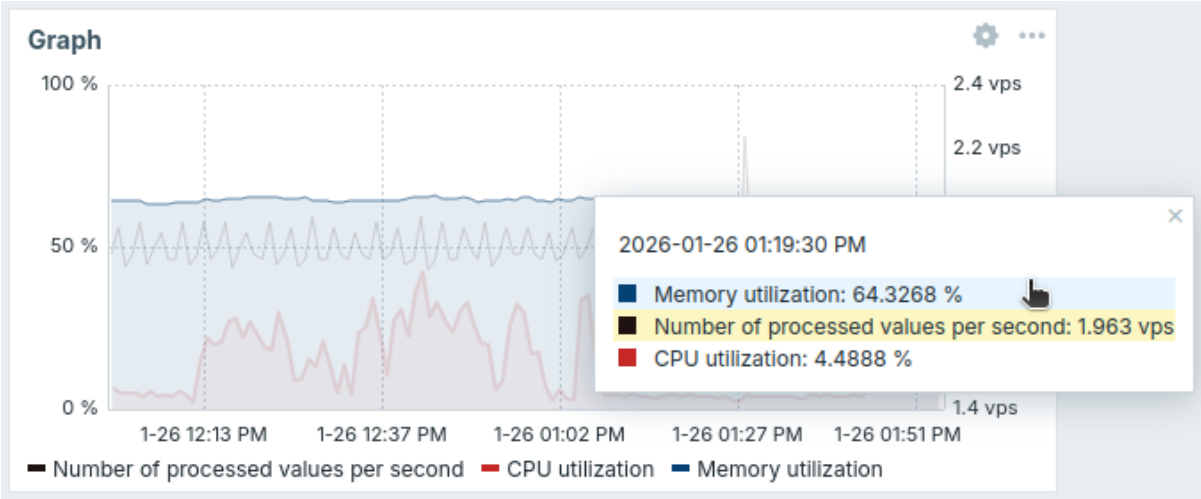
概述

Graph 小组件以基于矢量的图形显示数值型监控项数据。它可以帮助您跟踪趋势、发现问题，以及比较不同时间和主机之间的值。

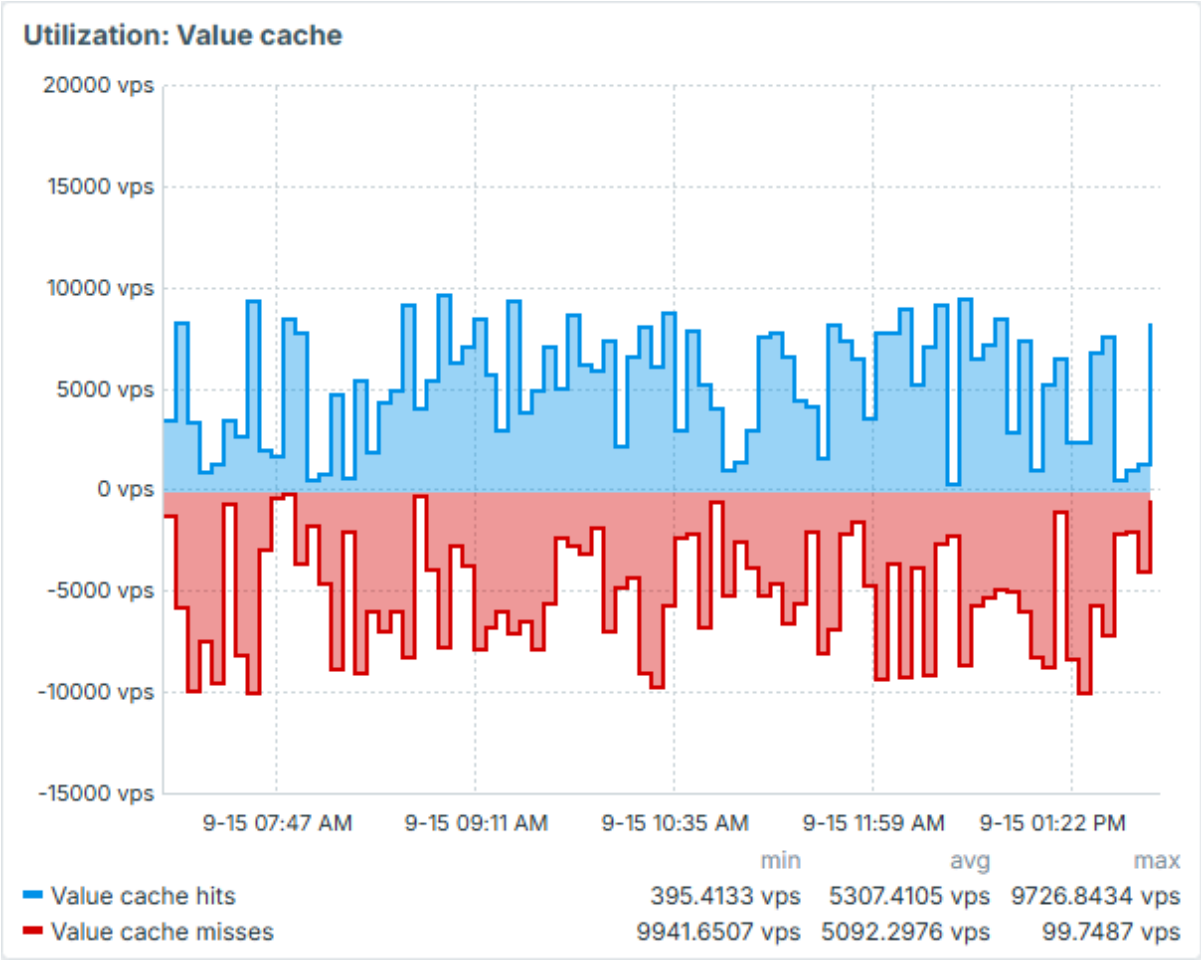


将鼠标悬停在图形上时，会显示一个工具提示，其中包含按值降序排列的监控项列表。

将鼠标悬停在工具提示中的某个监控项上时，该监控项对应的图形会突出显示（其他图形会变暗）；选择该监控项后，其数据会广播到其他小组件。



图形可以使用反转的 Y 轴值显示；这只会影响视觉呈现，不会更改原始值。

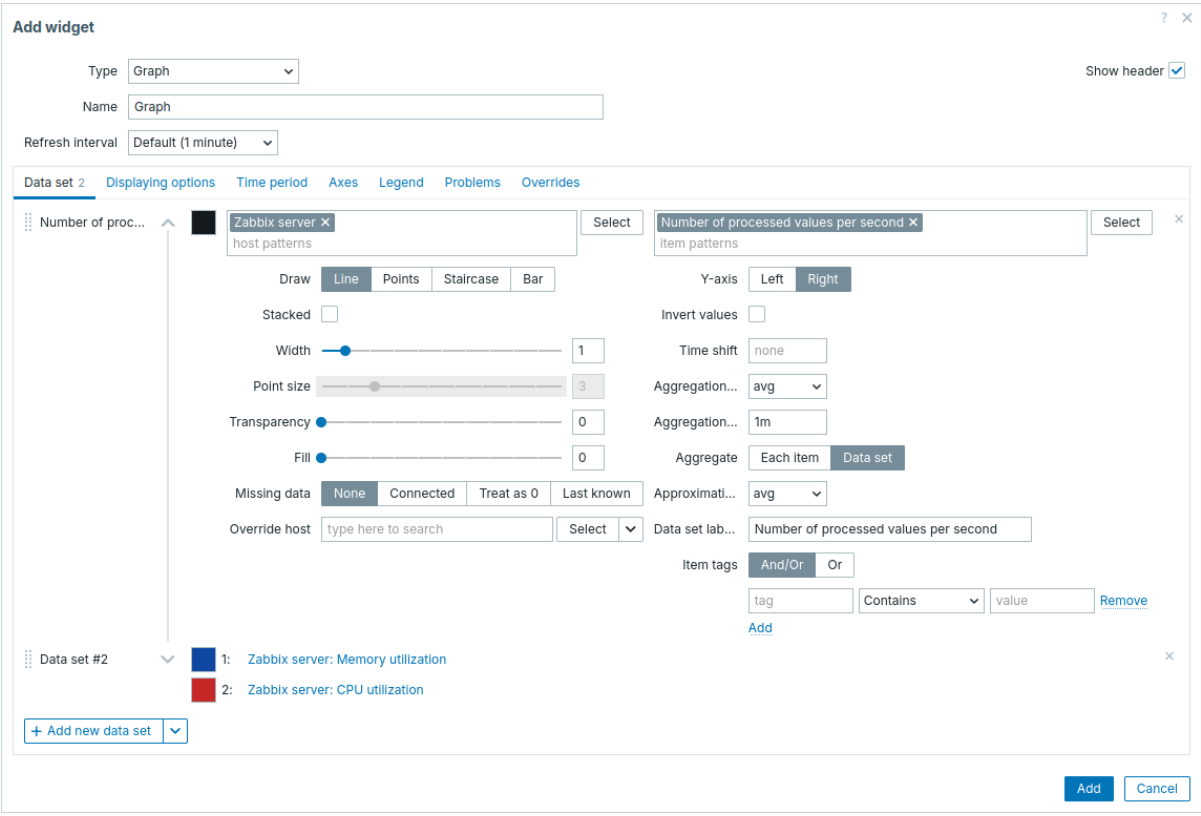


此外，还提供旧版Graph (classic) 小组件。

在小组件上悬停时，会显示小组件的操作菜单 ⋮。选择其中的下载图像选项，可以将 *Graph* 小组件中显示的信息下载为 PNG 图像。根据浏览器设置，图像可能会自动保存到 `Downloads` 文件夹，或者打开保存对话框，以便您选择保存位置。

配置

要配置，请选择 图表作为类型：



数据集

数据集选项卡允许通过添加数据集来选择图形中的数据。可以添加两种类型的数据集：

- 监控项模式 - 显示匹配监控项的数据。可以选择单一基础颜色，或选择调色板中的一行，为每个匹配的监控项分配不同的颜色。
- 监控项列表 - 显示所选监控项的数据。可以在颜色选择器中分别选择每个监控项的颜色。

默认情况下会添加一个监控项模式数据集。

数据集	对于监控项模式数据集： 选择或输入主机和监控项模式；图形中将显示与这些模式匹配的监控项数据；最多可显示 50 个监控项。 选择时可以使用通配符模式（例如，* 将返回匹配零个或多个字符的结果）。 要指定通配符模式，请手动输入字符串并按 *Enter*。 通配符始终会被解释，因此如果存在其他匹配的监控项（例如，*item2*、*item3*），则无法单独添加名为 *item** 的监控项。 对于 “监控项模式” 数据集，必须指定主机和监控项模式。 另请参阅：数据集配置详细信息。 对于监控项列表数据集： 点击添加监控项按钮，为图形选择监控项。 也可以点击添加小组件按钮，选择兼容的小组件作为监控项的数据源。 对于 “监控项列表” 数据集，必须指定监控项或小组件。 另请参阅：数据集配置详细信息。 请注意，仅允许使用数值型监控项类型。 在模板仪表板上配置小组件时，用于指定主机模式的参数不可用，并且用于指定监控项列表的参数仅允许选择在模板上配置的监控项。
绘制	选择指标的绘制类型。 可用的绘制类型：线条（默认设置）、点、阶梯和柱状图。 请注意，如果线条/阶梯图中只有一个数据点，则无论绘制类型如何，该数据点都会以点的形式绘制。点的大小根据线宽计算，但即使线宽小于 3 像素，点的大小也不会小于 3 像素。
堆叠	选中复选框，以堆叠方式显示数据（显示填充区域）。 选择点绘制类型时，此选项不可用。
宽度	设置线宽。 选择线条或阶梯绘制类型时，此选项可用。
点大小	设置点的大小。 选择点绘制类型时，此选项可用。
透明度	设置透明度级别。
填充	设置填充级别。 选择线条或阶梯绘制类型时，此选项可用。
缺失数据	选择显示缺失数据的方式： 无 - 保持空白间隔； 连接 - 连接两个边界值； 视为 **0** - 将缺失数据显示为 0 值； 最后已知值 - 使用与最后一个已知值相同的值显示缺失数据；不适用于点和柱状图绘制类型。
覆盖主机	选择兼容的小组件或仪表板主机选择器，作为主机的数据源。 在模板仪表板上配置小组件时，此参数不可用。
Y 轴	选择图形中显示 Y 轴的一侧。
反转值	选中复选框，在图形中显示之前将所有 Y 轴值乘以 -1。 此操作仅影响视觉表示；原始数据保持不变。 此选项不会影响工具提示、图例或百分位线。如果为坐标轴选择了简单触发器选项，触发器阈值将以反转后的形式显示。
时间偏移	如有需要，指定时间偏移。 此字段中可以使用时间后缀。允许使用负值。

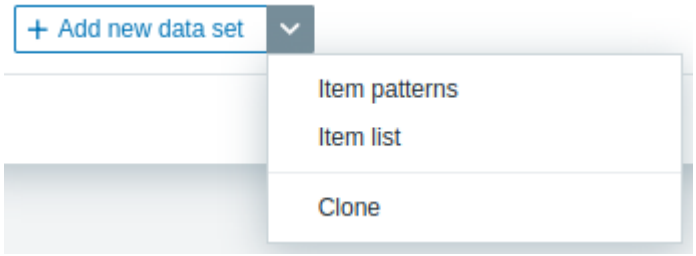
聚合函数	指定在所选聚合间隔内对每个监控项或整个数据集使用的聚合函数： **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值； **sum** - 显示值的总和； **count** - 显示值的数量； **first** - 显示第一个值； **last** - 显示最后一个值。 如果选择不使用，则显示所有值（不进行聚合）。 *min*、*max*、*avg* 和 *sum* 只能显示数值数据。对于 *count*，非数值数据将转换为数值。 另请参阅：图形中的聚合。
聚合间隔	指定聚合值的间隔。 此字段中可以使用时间后缀。不带后缀的数值将被视为秒。 请注意，如果小组件配置为根据趋势显示历史数据（历史数据选择设置为趋势或自动），建议使用 1 小时的倍数作为聚合间隔（例如 3600、60m、1h、3h 等）。趋势存储按小时聚合的值，因此使用非 1 小时倍数的聚合间隔（例如 100s、7min、15min、90min 等）可能会导致难以解释的结果。
聚合对象	指定是否进行聚合： 每个监控项 - 分别聚合并显示数据集中的每个监控项； 数据集 - 聚合数据集中的所有监控项，并将其显示为一个值。
近似方式	当每个垂直图形像素对应多个值时，指定要显示的值： **all** - 显示最小值、最大值和平均值； **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值。 当以较短的更新间隔显示较长时间段的图形时，此设置非常有用（例如，显示一年内每 10 分钟收集的值）。
数据集标签	指定显示在图形数据集配置和图形图例中的数据集标签（适用于聚合数据集）。 所有数据集都会编号，包括已指定数据集标签的数据集。如果未指定标签，系统将根据数据集编号自动添加标签（例如 “数据集 #2”、“数据集 #3” 等）。重新排序/拖动数据集后，将重新计算数据集编号。 过长的数据集标签将被缩短，以适应其显示位置（例如 “进程数...”）。
监控项标签	指定用于筛选小组件中显示的监控项的标签。 可以包含或排除特定标签及标签值。可以设置多个条件。 标签名称匹配始终区分大小写。 每个条件有多个可用运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将通过 “或” 条件进行分组； 或 - 满足一个条件即可。

数据集配置详细信息

现有数据集将以列表形式显示。您可以：

- 单击 移动图标，并将数据集拖动到列表中的新位置。
- 单击 展开图标，以展开数据集详细信息。展开后，此图标将变为 折叠图标。
- 单击 颜色图标以打开颜色选择器。您可以输入十六进制代码、选择纯色样本，或切换到 调色板选项卡并选择预定义颜色行。对于 监控项列表数据集，所选颜色会直接应用；对于 监控项模式，所选颜色将作为生成不同色调的基础颜色。使用 Tab 键在对话框控件之间移动，使用箭头键浏览颜色样本或调色板行，按 Enter 键进行选择，按 Esc 键取消。
- 单击 添加新数据集按钮，以添加一个空数据集，从而选择主机和监控项模式。如果单击 添加新数据集按钮旁边的向下箭头图标，将显示一个下拉菜单，允许您添加新的 监控项模式或 监控项列表数据集，或者 克隆当前打开的数据集。如果所有数据集都处于折叠

状态，则 克隆选项不可用。

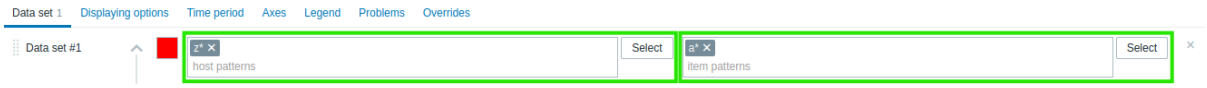


监控项模式数据集包含 主机模式和 监控项模式字段，这两个字段都可识别完整名称或包含通配符（*）的模式。此功能允许您选择所有包含所选模式的主机名和监控项名称。在 监控项模式字段中输入监控项名称或监控项模式时，下拉列表中仅显示属于所选主机名的监控项。

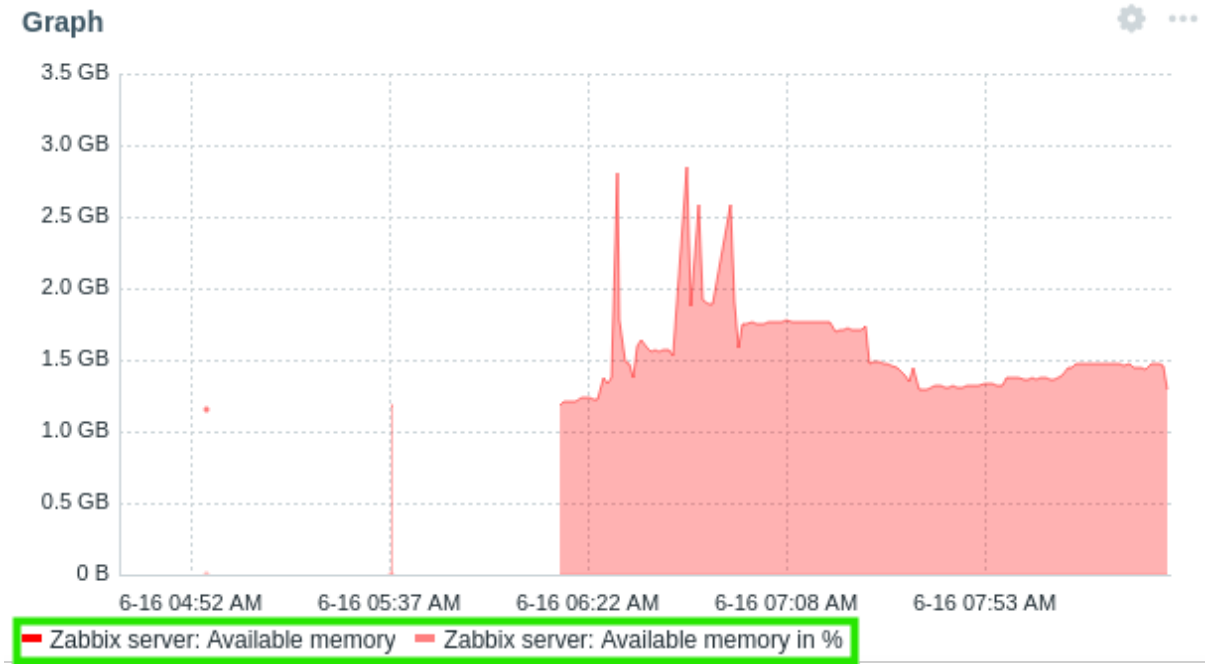
例如，在 主机模式字段中输入模式 **z*** 后，下拉列表会显示所有包含该模式的主机名：**z***、**Zabbix server**、**Zabbix proxy**。按下 *Enter* 后，该模式会被接受并显示为 **z***。类似地，在 监控项模式字段中输入模式 **a*** 后，下拉列表会显示所有包含该模式的监控项名称：**a***、**Available memory**、**Available memory in %**。



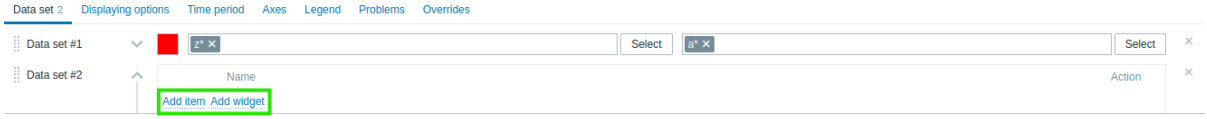
按下 *Enter* 后，该模式会被接受并显示为 **a***。



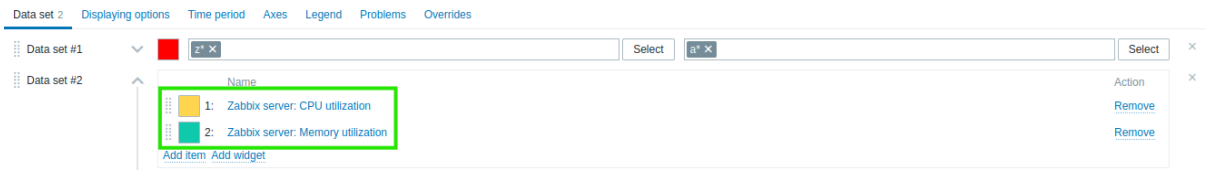
随后，图形将显示属于所选主机名的所有监控项。



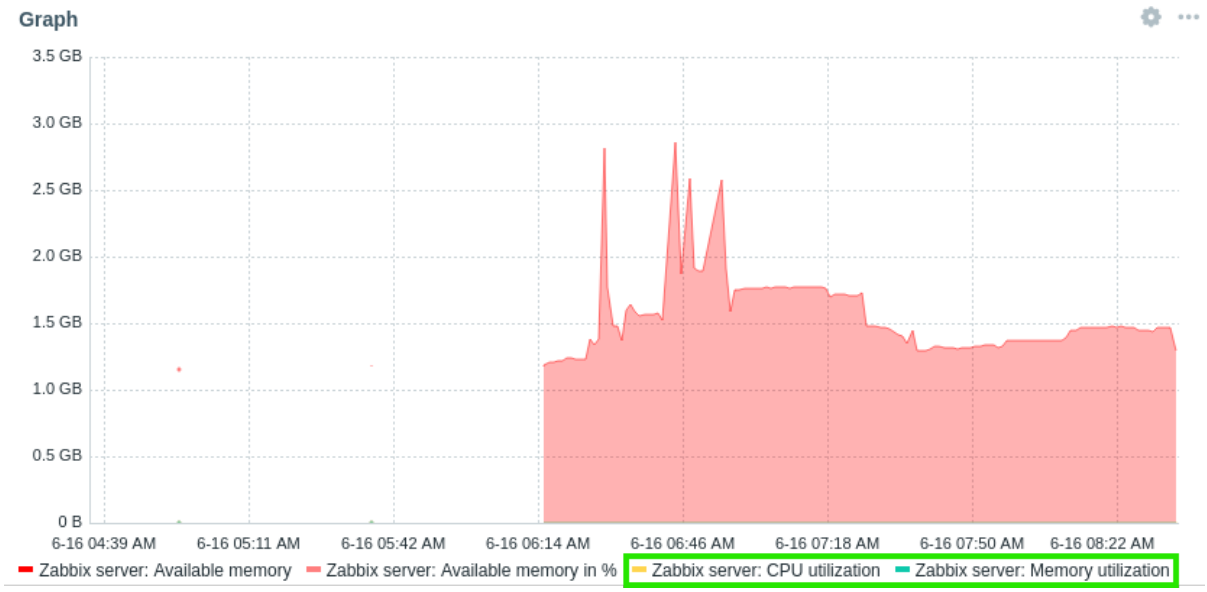
监控项列表数据集包含 添加监控项按钮，可用于添加要显示在图形上的监控项。
您还可以通过点击 添加部件按钮，将兼容的部件添加为监控项的数据源。



例如，点击 添加监控项按钮会打开一个包含 主机参数的弹出窗口。
选择主机后，该主机上所有可供选择的监控项都会显示在列表中。



选择一个或多个监控项后，它们将显示在数据集监控项列表和图形中。

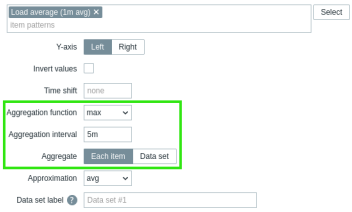


图表中的聚合

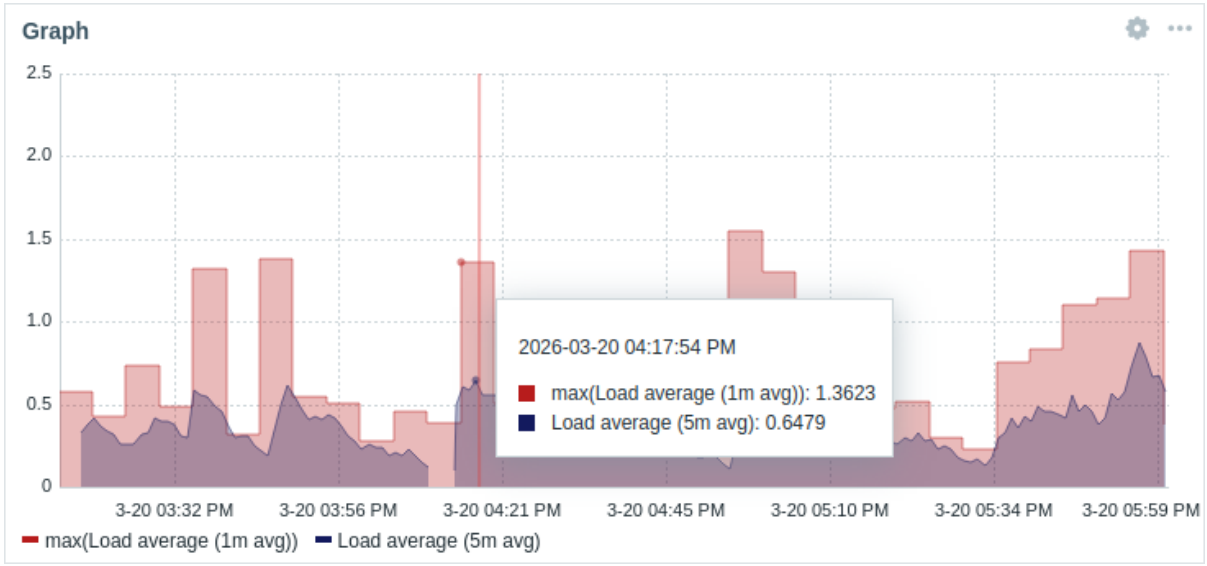
聚合允许显示指定时间间隔（5 分钟、1 小时、1 天等）内的聚合值（最小值、最大值、平均值等），而不是显示所有单独的值。在一个数据集中，聚合既可以分别应用于每个监控项，也可以应用于整个数据集。

聚合功能的一些使用场景如下：- 每天向 NGINX 服务器发出的平均请求数。- 每周各集群中的最低磁盘空间。

在以下示例中，”max” 聚合函数应用于一个监控项，聚合间隔为 5 分钟：



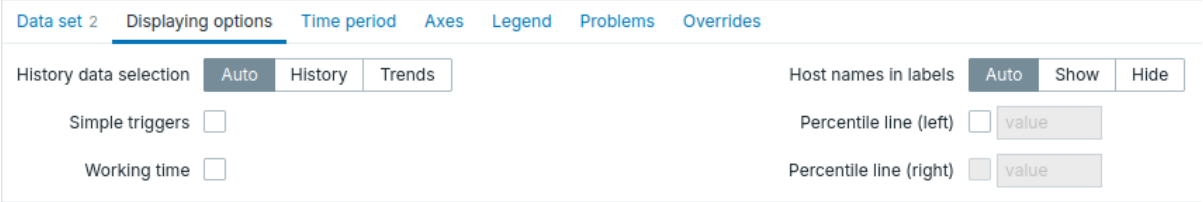
该示例展示了 5 分钟内的最大 CPU 负载（由每分钟的平均值聚合而成），并将其与 5 分钟内的平均 CPU 负载进行比较。



为图例选择 *Show aggregation function* 后，监控项会显示在括号中，并在图例和工具提示中以聚合函数作为前缀。如果为图表指定了 *Data set label*，该标签会显示在括号中，并以聚合函数作为前缀。

显示选项

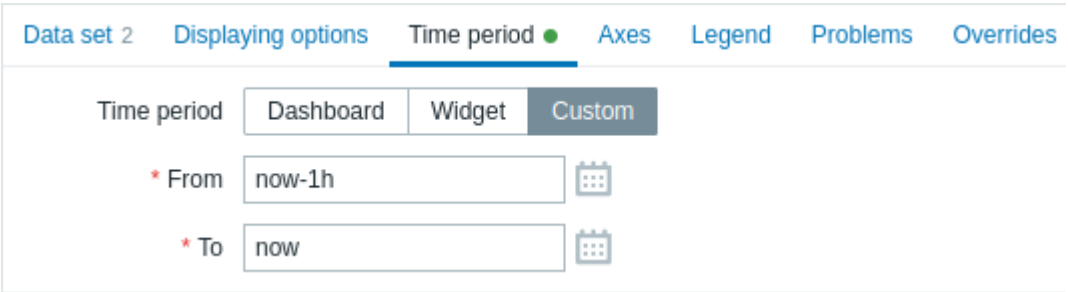
显示选项选项卡用于定义历史数据选择：



历史数据选择	设置图形数据源： 自动 - 根据经典图形算法选择数据源（默认）； 历史数据 - 使用历史数据； 趋势数据 - 使用趋势数据。
简单触发器	选中复选框以显示简单触发器的触发器阈值。阈值将使用触发器严重性颜色以虚线显示。 简单触发器是表达式中针对一个监控项且仅包含一个函数（last、max、min、avg）的触发器。 最多可以绘制三个触发器。请注意，触发器必须位于绘制范围内才能显示。
工作时间	选中复选框以在图形上显示工作时间。工作时间（工作日）在图形中显示为白色背景，而非工作时间显示为灰色（使用原始蓝色默认前端主题时）。
标签中的主机名称	选择是否在图形图例以及将鼠标悬停在图形数据上时显示的工具提示中显示主机名称： 自动 - 仅当所有数据集中的主机数量超过一个时显示主机名称（默认）。 显示 - 显示主机名称。 隐藏 - 隐藏主机名称。 在模板仪表板上配置小组件时，此参数不可用。
百分位线（左）	选中复选框并输入百分位值，以在图形左侧 Y 轴上以线条形式显示指定的百分位。 例如，如果设置为 95% 百分位，则百分位线将位于 95% 的值低于该水平的位置。
百分位线（右）	选中复选框并输入百分位值，以在图形右侧 Y 轴上以线条形式显示指定的百分位。 例如，如果设置为 95% 百分位，则百分位线将位于 95% 的值低于该水平的位置。

时间段

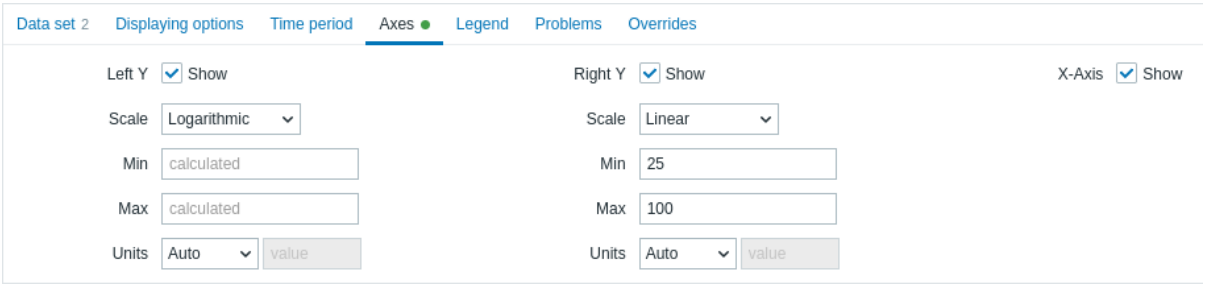
时间段选项卡允许您设置图表中显示数据的时间段：



时间段	选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。
小组件	输入或选择兼容的小组件作为时间段的数据源。 当时间段设置为“小组件”时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。

坐标轴

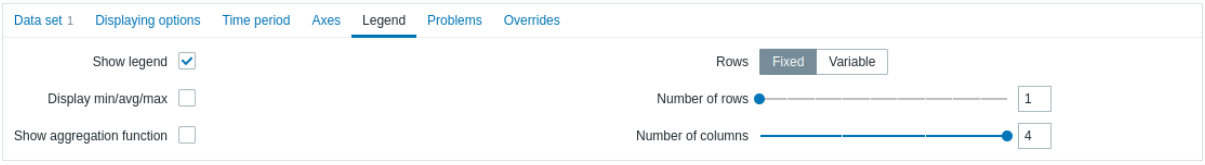
坐标轴选项卡允许您自定义坐标轴的显示方式：



左 *Y* 轴	选中此复选框可显示左 Y 轴。 如果在数据集或覆盖选项卡中取消选择，可能会禁用此复选框。
右 *Y* 轴	选中此复选框可显示右 Y 轴。 如果在数据集或覆盖选项卡中取消选择，可能会禁用此复选框。
X 轴	取消选中此复选框可隐藏 X 轴（默认选中）。
刻度	从下拉列表中选择图表坐标轴值的刻度： 线性 - 坐标轴值按固定增量递增（例如 10、20、30），适用于变化平稳或范围较小到中等的数据； 对数 - 坐标轴值按指数递增（例如 10、100、1000），适用于变化迅速或范围较大的数据。
最小值	设置相应坐标轴的最小值。 指定 Y 轴的可见范围最小值。
最大值	设置相应坐标轴的最大值。 指定 Y 轴的可见范围最大值。
单位	从下拉列表中选择图表坐标轴值的单位： 自动 - 使用数据集中第一个监控项的单位显示坐标轴值； 静态 - 使用值输入字段中指定的单位显示坐标轴值；如果该字段留空，则仅显示数值。

图例

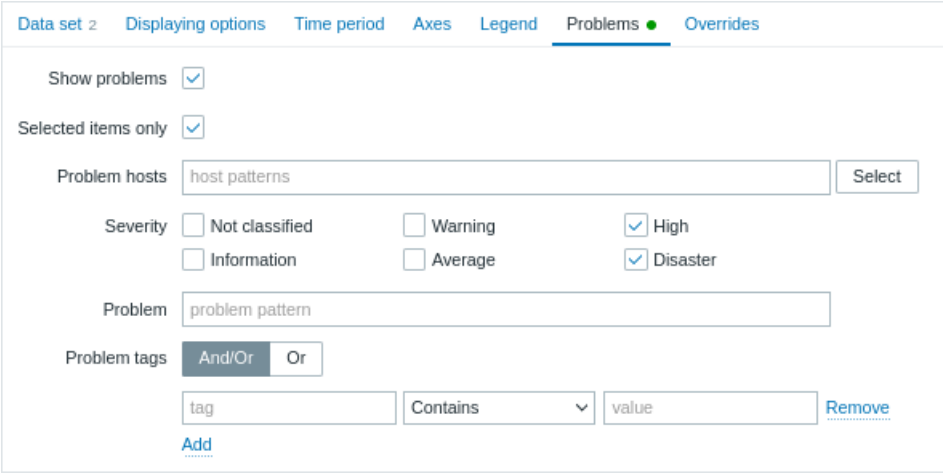
图例选项卡允许您自定义图形图例：



显示图例	取消选中此复选框可隐藏图形上的图例（默认选中）。
显示最小值/平均值/最大值	选中此复选框可在图例中显示监控项的最小值、平均值和最大值。
显示聚合函数	选中此复选框可在图例中显示聚合函数。
行	选择图例行的显示模式： 固定 - 显示的行数由行数参数值决定； 可变 - 显示的行数由已配置监控项的数量决定，但不会超过最大行数参数值。
行数/ 最大行数	如果行设置为“固定”，请设置要显示的图例行数（1-10）。 如果行设置为“可变”，请设置要显示的图例最大行数（1-10）。
列数	设置要显示的图例列数（1-4）。 当未选中显示最小值/平均值/最大值时，此参数可用。

问题

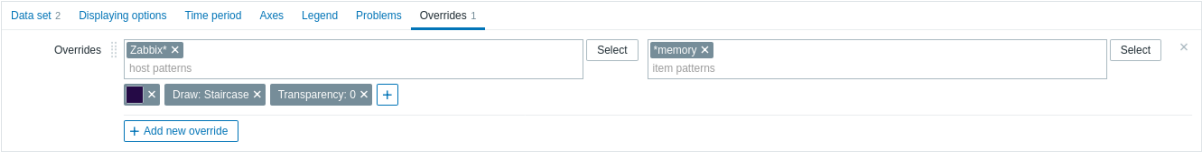
问题选项卡允许您自定义问题显示方式：



显示问题	勾选此复选框可启用在图形上显示问题（默认未勾选，即禁用）。
仅选定的监控项	勾选此复选框后，图形上仅显示选定监控项的问题。
问题主机	选择要在图形上显示其问题的主机。 可以使用通配符模式（例如，* 将返回匹配零个或多个字符的结果）。 要指定通配符模式，请手动输入字符串并按 *Enter*。 输入时，请注意下拉列表中会显示所有匹配的主机。 在模板仪表板上配置小组件时，此参数不可用。
严重性	勾选问题严重性，以筛选要在图形上显示的问题。 如果未勾选任何严重性，则会显示所有问题。
问题	指定要在图形上显示的问题名称。
问题标签	指定问题标签，以限制小组件中显示的问题数量。 可以包含或排除特定标签及标签值。可以设置多个条件。标签名称匹配始终区分大小写。 每个条件有以下几种可用运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含标签值中包含所输入字符串的指定标签名称（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除标签值中包含所输入字符串的指定标签名称（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件；具有相同标签名称的条件将通过 *Or* 条件进行分组； **Or** - 满足一个条件即可。

覆盖

覆盖选项卡允许您为数据集添加自定义覆盖：



当使用 * 通配符为数据集选择多个监控项，并且您希望更改这些监控项的默认显示方式（例如，默认基色或其他属性）时，覆盖功能非常有用。

现有覆盖（如果有）将以列表形式显示。要添加新的覆盖：

- 单击  按钮

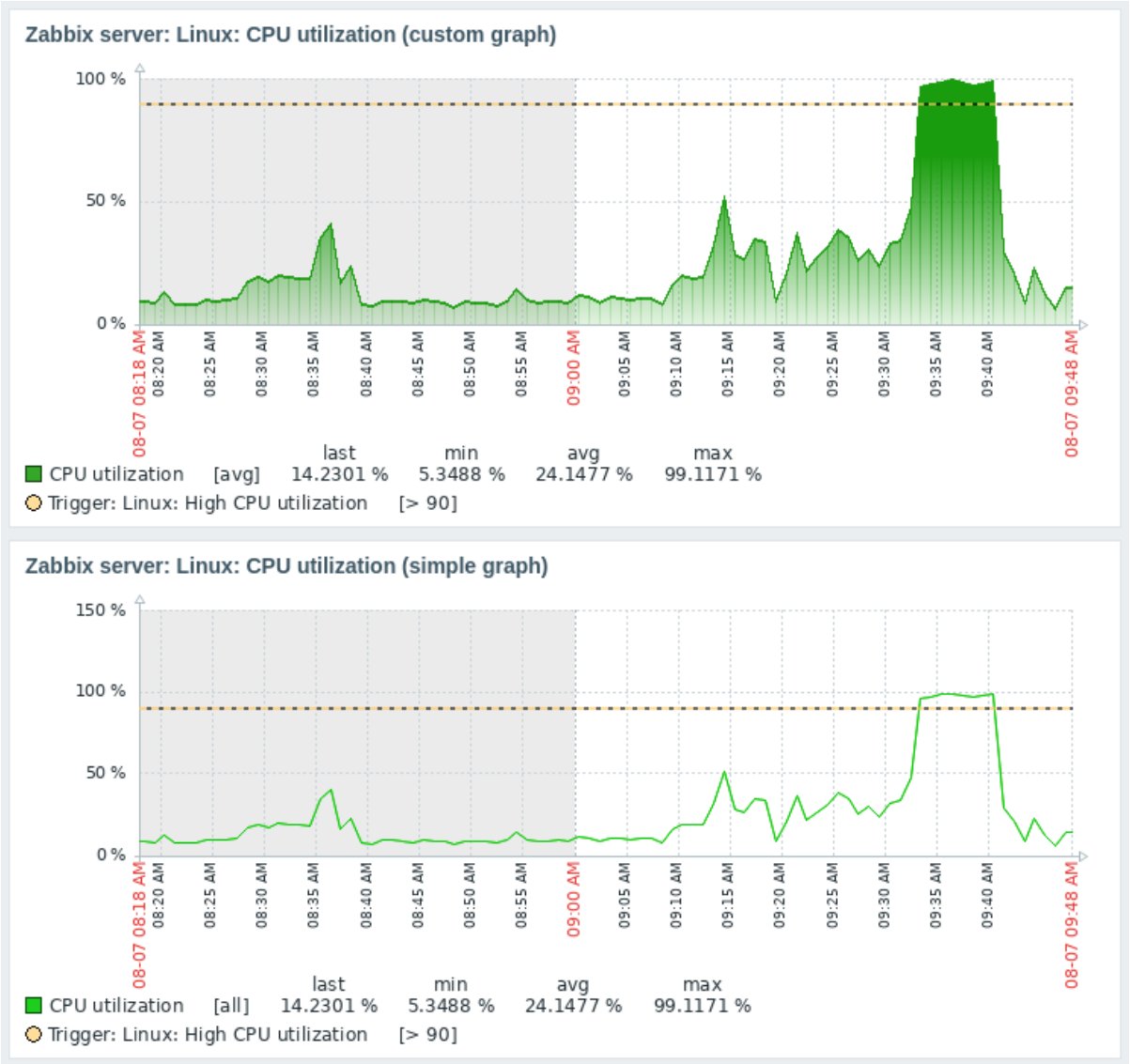
- 为覆盖选择主机和监控项。或者，您也可以输入主机和监控项模式。可以使用通配符模式（例如，* 将返回匹配零个或多个字符的结果）。要指定通配符模式，只需手动输入字符串，然后按 *Enter*。输入时，请注意下拉列表中会显示所有匹配的主机。通配符始终会被解释，因此，如果存在其他匹配的监控项（例如 item2、item3），则无法单独添加名为 “item*” 的监控项。主机模式和监控项模式参数为必填项。在模板仪表板上配置小组件时，用于指定主机模式的参数不可用。在模板仪表板上配置小组件时，用于指定监控项列表的参数仅允许选择在模板上配置的监控项。

- 单击 ，选择覆盖参数。至少应选择一个覆盖参数。有关参数说明，请参阅上面的数据集选项卡。

图表（经典）

概述

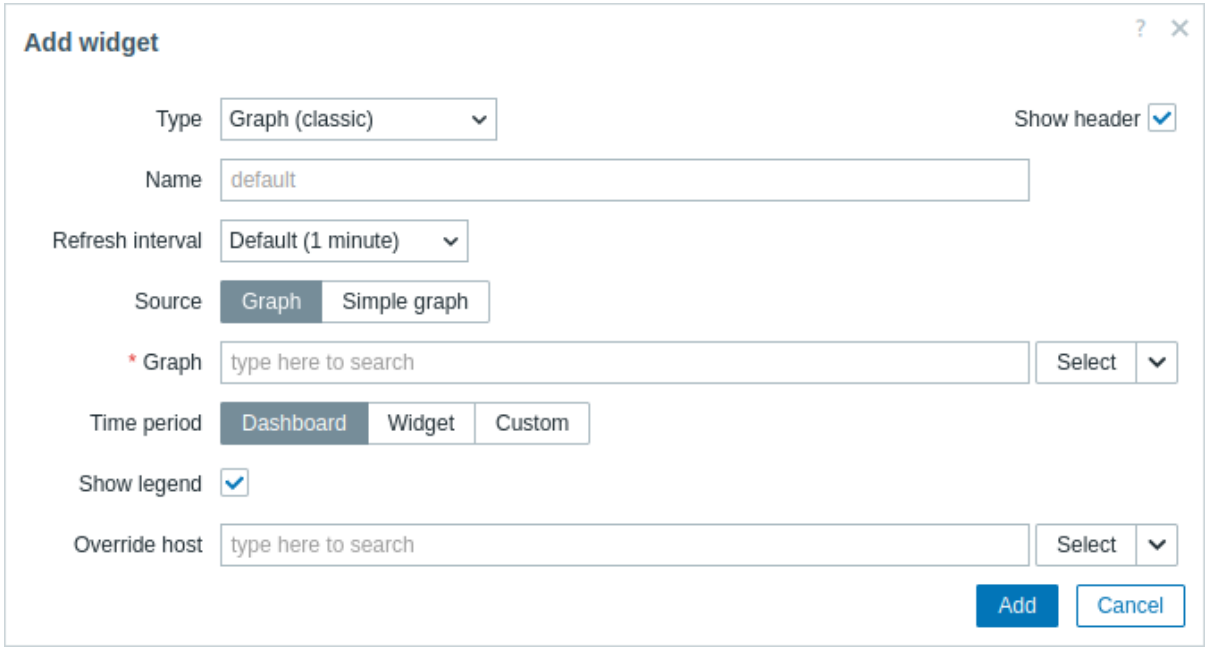
图形（经典）小组件以图像形式显示监控项的数值数据，可以是自定义图形或简单图形。它支持选择时间段、显示触发器线以及可选的图例。



图形（经典）小组件中显示的信息可以下载为 PNG 图像：将鼠标悬停在小组件上时，会显示小组件的操作菜单 ...，选择其中的下载图像选项即可。根据浏览器设置，图像可能会自动保存到 Downloads 文件夹，或者打开保存对话框，以便选择保存位置。

配置

要进行配置，请选择 *Graph (classic)* 作为类型：



除了所有部件通用的常用参数外，您还可以设置以下特定选项：

Source	选择图形类型： **Graph** - 自定义图形； **Simple graph** - 简单图形。
Graph	选择要显示的自定义图形。 或者，选择一个兼容的部件作为图形的数据源。 当 *Source* 设置为 “Graph” 时，此参数可用。
Item	选择要在简单图形中显示的监控项。 或者，选择一个兼容的部件作为监控项的数据源。 当 *Source* 设置为 “Simple graph” 时，此参数可用。
Time period	设置图形中显示数据的时间段。为时间段选择数据源： **Dashboard** - 使用仪表板的时间段选择器； **Widget** - 使用兼容的部件（在 *Widget* 参数中设置）； **Custom** - 使用在 *From* 和 *To* 参数中设置的自定义时间段；如果已设置，部件右上角将显示一个时钟图标，鼠标悬停时会指示所设置的时间。 请注意，无论部件的 *Time period* 如何配置，兼容的部件仍可将其用作时间段的数据源。
Widget	输入或选择一个兼容的部件，作为时间段的数据源。 当 *Time period* 设置为 “Widget” 时，此参数可用。
From	输入或选择时间段的开始时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 当 *Time period* 设置为 “Custom” 时，此参数可用。
To	输入或选择时间段的结束时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 当 *Time period* 设置为 “Custom” 时，此参数可用。
Show legend	取消选中此复选框可隐藏图形上的图例（默认选中）。
Override host	选择一个兼容的部件或仪表板的主机选择器，作为主机的数据源。 在模板仪表板上配置该部件时，此参数不可用。

图形的图例由三个部分组成：

- 监控项及其聚合值
- 百分位数（如果已配置）
- 触发器（如果与显示的监控项相关联）

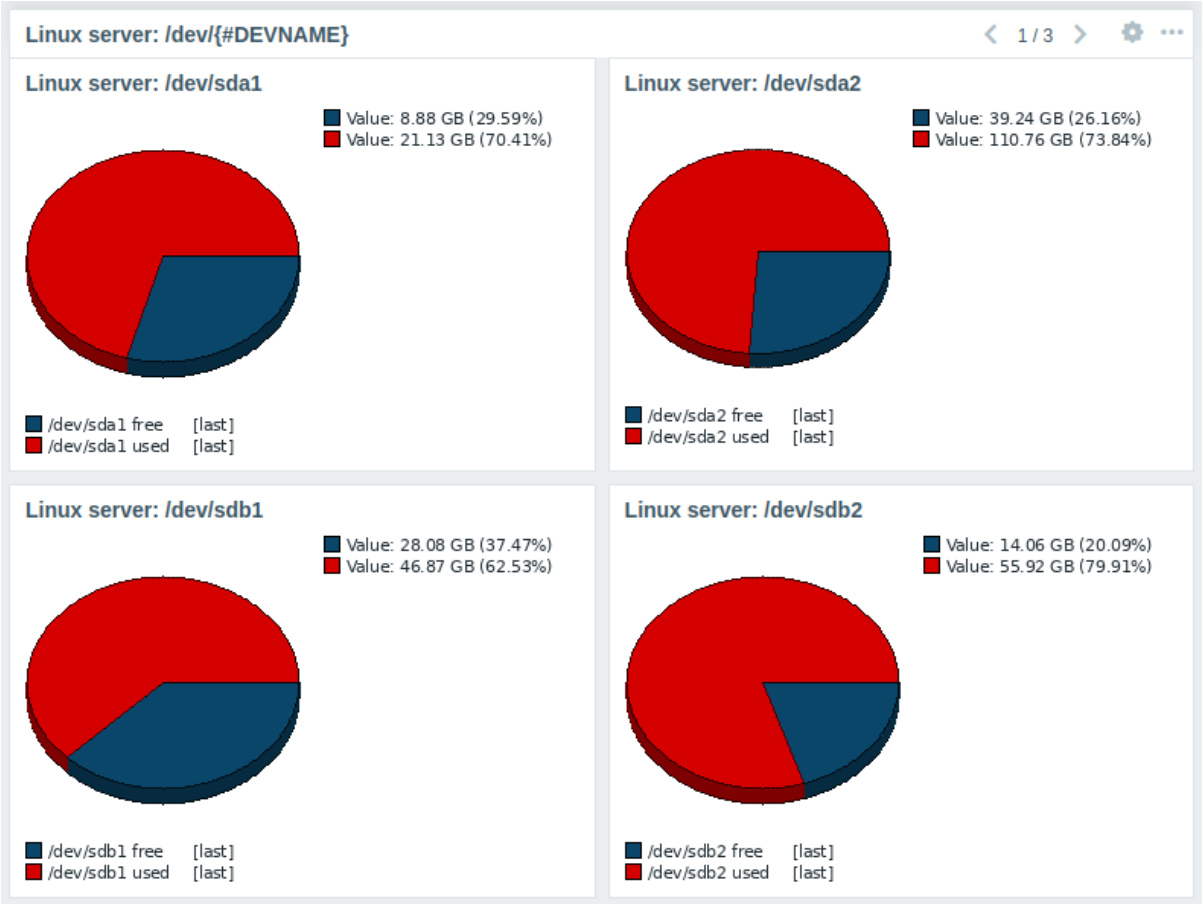
如果部件内图形的高度不足，图例可能不会显示，或者只会部分显示。
将首先隐藏触发器和百分位数，然后再隐藏监控项图例。要显示完整图例，请增加部件的垂直尺寸。

Note:
最多只能显示 3 条触发器线。
如果触发器数量更多，则会优先显示严重性较低的触发器。

图原型

概述

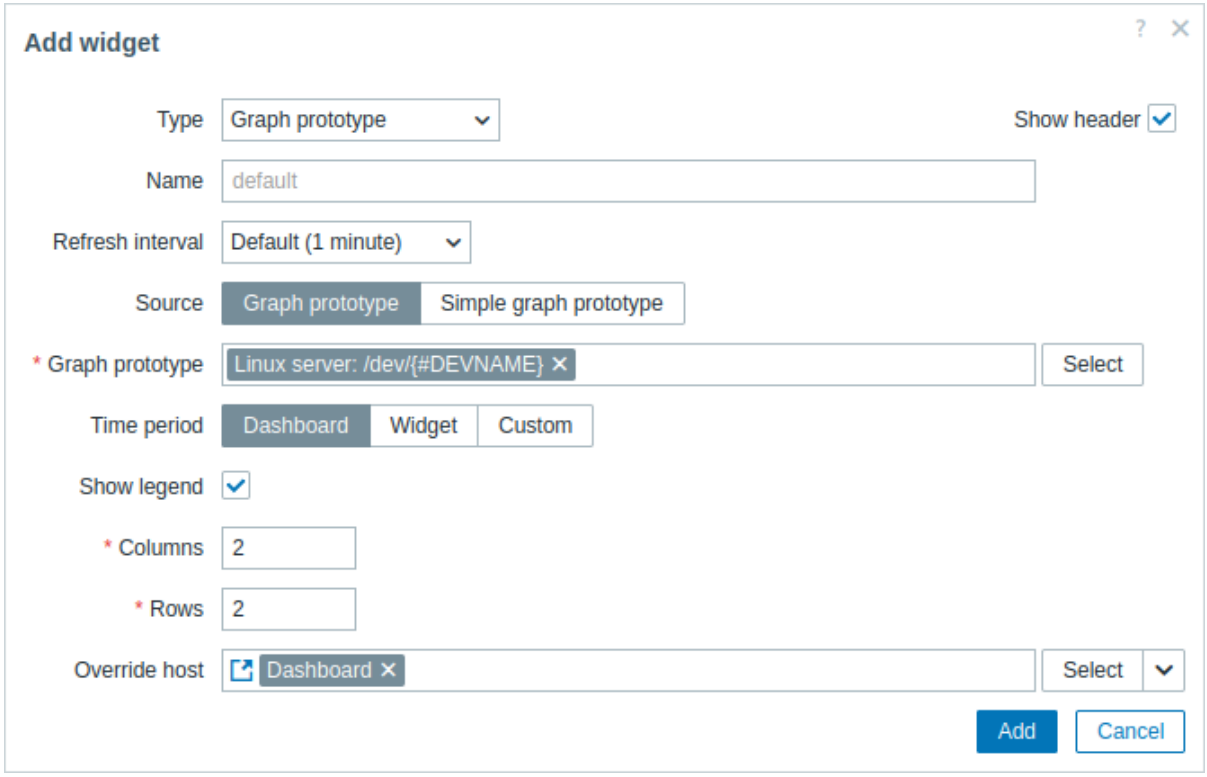
图形原型小部件会显示一个由自动创建的自定义图形组成的网格，这些图形基于低级别发现规则的图形原型或监控项原型。



图形会按可配置的网格布局排列；如果图形数量超出布局容量，则小部件中会启用分页。

配置

要进行配置，请选择 图形原型作为小组件类型：



除了适用于所有小组件的通用参数外，还可以设置以下特定选项：

来源	选择图形的来源：图形原型或简单图形原型。
图形原型	选择一个图形原型，以显示由该图形原型发现的图形。 当来源设置为“图形原型”时，此参数可用。
监控项原型	选择一个监控项原型，以显示由该监控项原型发现的监控项的简单图形。 当来源设置为“简单图形原型”时，此参数可用。
时间段	设置图形中显示数据的时间段。选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。
小组件	输入或选择一个兼容的小组件，作为时间段的数据源。 当时间段设置为“小组件”时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。
显示图例	取消选中此复选框可隐藏图形上的图例（默认选中）。
覆盖主机	选择一个兼容的小组件或仪表板的主机选择器，作为主机的数据源。 在模板仪表板上配置小组件时，此参数不可用。
列	输入图形原型小组件中要显示的图形列数。
行	输入图形原型小组件中要显示的图形行数。

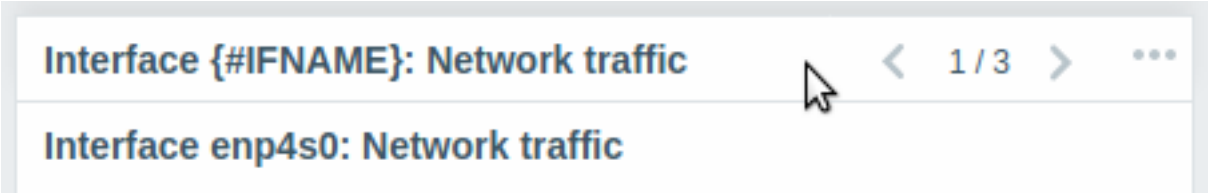
图形图例由三个部分组成：

- 监控项及其聚合值
- 百分位数（如果已配置）
- 触发器（如果显示的监控项关联了触发器）

如果小组件中图形的高度不足，图例可能不会显示，或仅显示部分内容。触发器和百分位数会首先被隐藏，随后是监控项图例。要显示完整图例，请增加小组件的垂直尺寸。

Note:
最多可以显示 3 条触发器线。如果触发器数量更多，则优先显示严重性较低的触发器。

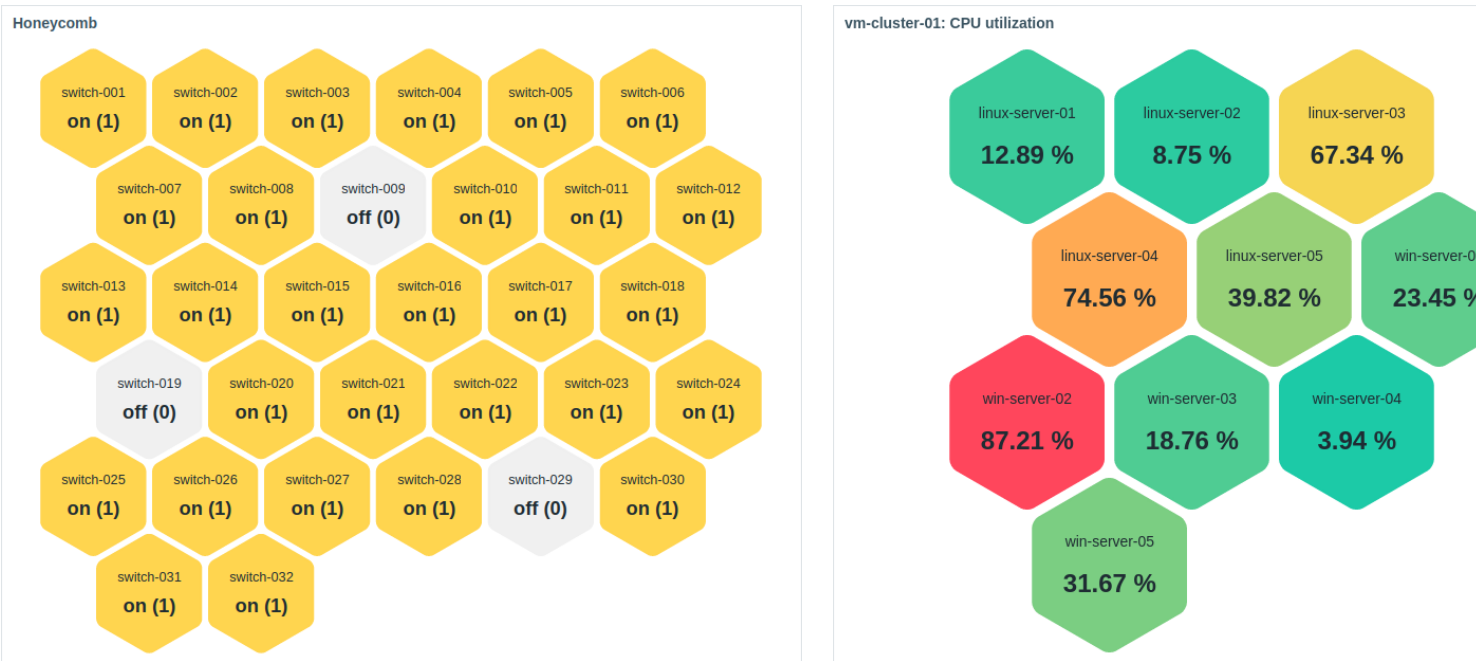
虽然列和行参数允许在小组件中容纳多个图形，但发现的图形数量仍可能多于小组件中的列数或行数。在这种情况下，小组件中将提供分页功能，并显示一个上滑标题栏，您可以使用左右箭头在页面之间切换：



蜂巢

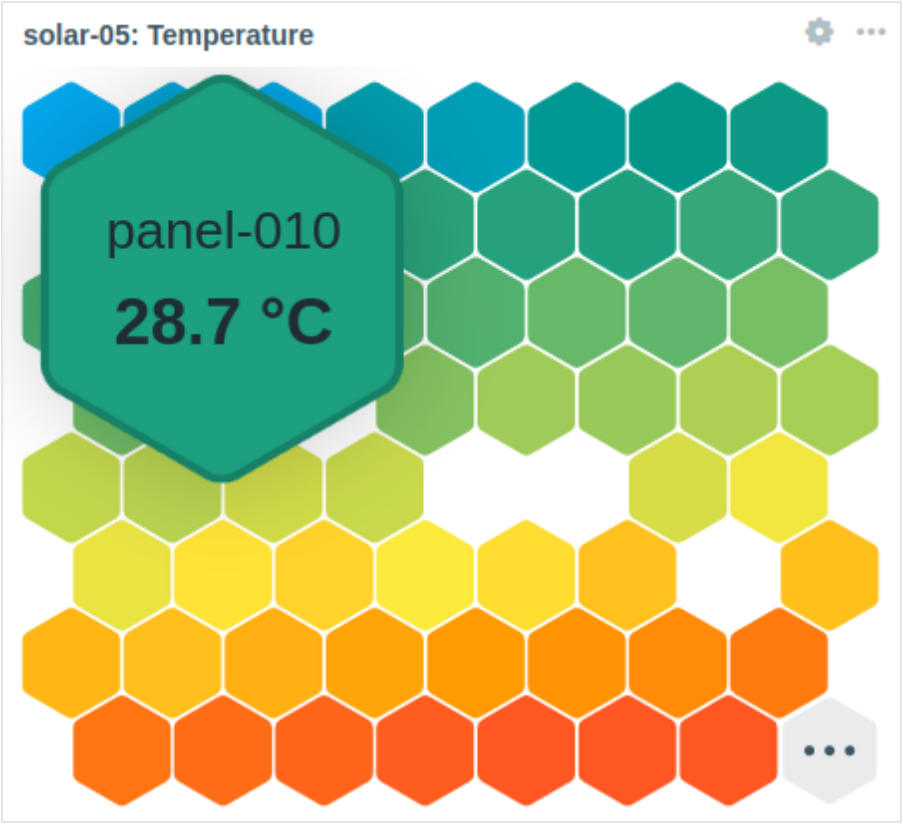
概述

蜂窝图小组件以网格中的六边形单元格显示主机或监控项。这种布局便于检测存在问题的主机/监控项、发现问题集群、快速比较各组信息等。



显示的单元格数量受小组件大小和最小单元格大小（32px）的限制。如果无法容纳所有单元格，最后一个单元格会显示省略号。小组件可以调整大小，单元格数量、大小和位置也会动态调整。

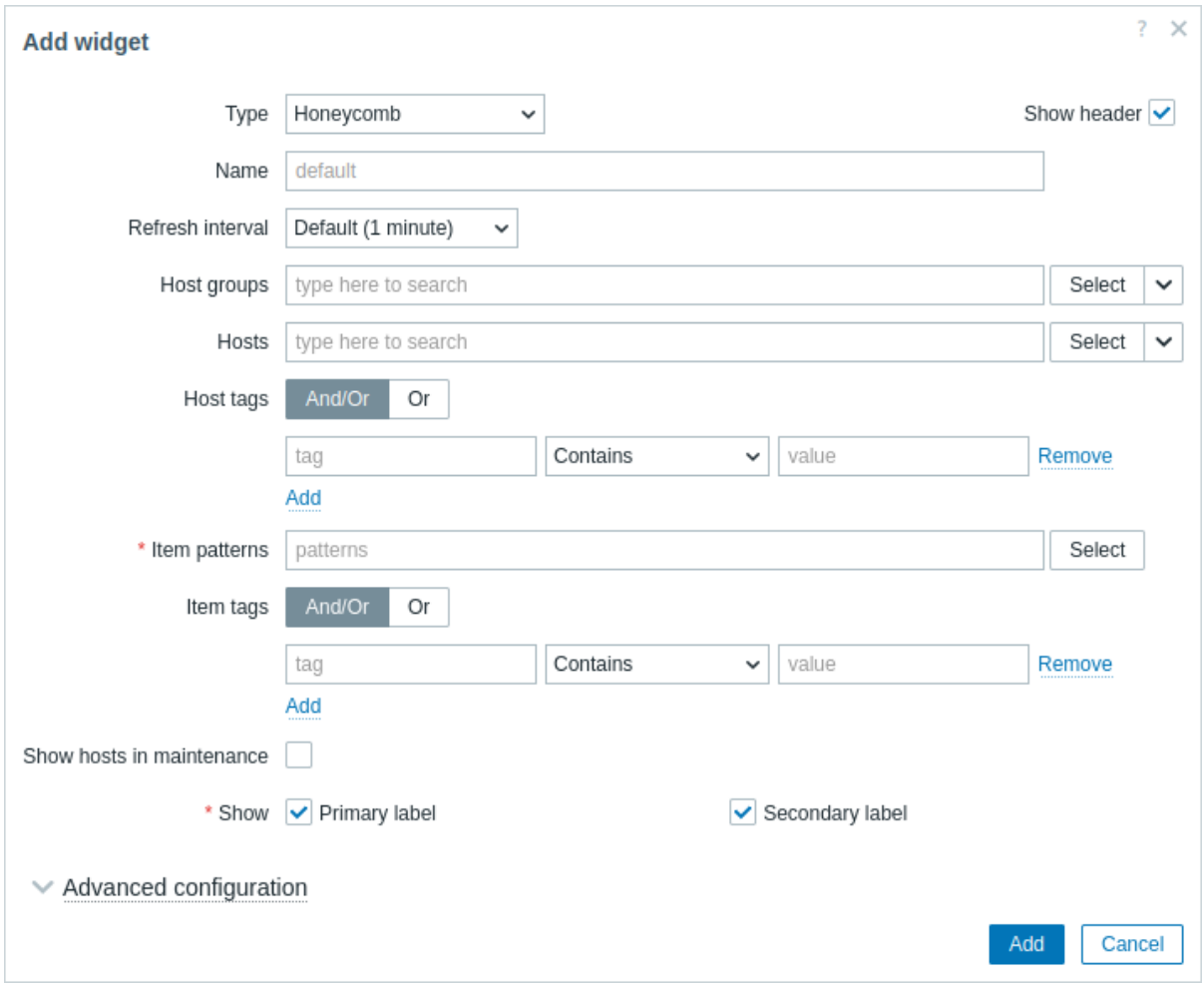
将鼠标悬停在单元格上时，当前聚焦的单元格会放大，以提高可见性。单击单元格后，其边框会突出显示，直到选择其他单元格。



在小组件上显示的信息可以下载为 PNG 图像：将鼠标悬停在小组件上，在出现的小组件操作菜单 ⋯ 中选择下载图像选项。根据浏览器设置，图像可能会自动保存到 `Downloads` 文件夹，或者打开保存对话框，以便选择保存位置。

配置

要进行配置，请选择 *Honeycomb* 作为类型：

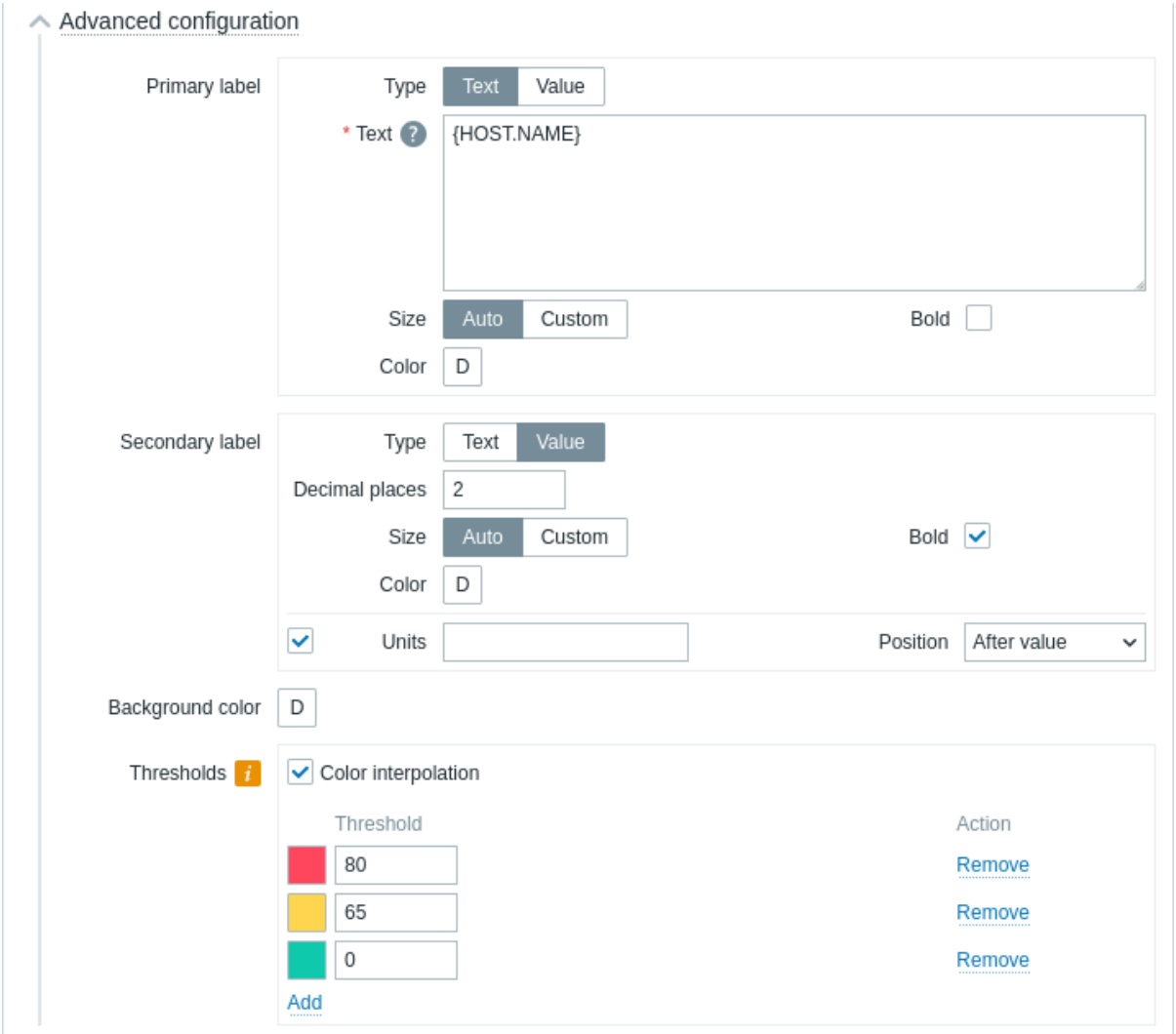


除了所有部件通用的常用参数外，还可以设置以下特定选项：

Host groups	选择主机组。 或者，选择一个兼容的部件作为主机组的数据源。 此字段支持自动补全，因此开始输入组名时，将显示匹配组的下拉列表。 选择父主机组会隐式选择其下所有嵌套主机组；如果未选择任何主机组，部件将显示所有包含主机且这些主机具有与所选 *Item pattern* 匹配监控项的主机组（见下文）。 在模板仪表板上配置该部件时，此参数不可用。
Hosts	选择主机。 或者，选择一个兼容的部件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名时，将显示匹配主机的下拉列表。 如果未选择任何主机，部件将显示所有具有与所选 *Item pattern* 匹配监控项的主机（见下文）。 在模板仪表板上配置该部件时，此参数不可用。
Host tags	指定标签以限制部件中显示的主机数量。 可以包含或排除特定标签及标签值。可设置多个条件。 标签名称匹配始终区分大小写。 每个条件可使用以下几种运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含指定的标签名称，且标签值包含输入的字符串（子串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除指定的标签名称，且标签值包含输入的字符串（子串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组； **Or** - 只需满足一个条件即可。 在模板仪表板上配置该部件时，此参数不可用。
Item patterns	输入监控项模式，或选择现有监控项作为监控项模式。与输入或所选模式匹配的监控项数据将显示在蜂窝图中。*Item patterns* 参数为必填项。 可使用通配符模式进行选择（例如，`*` 将返回匹配零个或多个字符的监控项；`Zabbix*` 将返回以“Zabbix”开头的监控项）。 要指定通配符模式，请手动输入字符串并按 *Enter*。开始输入时，将显示匹配监控项的下拉列表；如果已选择 *Hosts* 或 *Host groups* 中的主机组，则列表仅限于属于这些主机或主机组内主机的监控项。通配符始终会被解释，因此如果存在其他匹配监控项（例如 item2、item3），则无法单独添加名为 *item** 的监控项。 在模板仪表板上配置该部件时，此参数仅允许选择模板上配置的监控项。
Item tags	指定标签以限制部件中显示的监控项数量。更多信息请参见上文的 *Host tags*。
Show hosts in maintenance	选中此复选框以显示维护中的主机（在这种情况下，主机名旁将显示维护图标）。 在模板仪表板上配置该部件时，此参数标签为 *Show data in maintenance*。
Show	选中此复选框以显示相应的蜂窝单元元素——主标签、次标签。 必须至少选择一个元素。
Advanced configuration	单击 *Advanced configuration* 标题以显示高级配置选项。

高级配置

可折叠的高级配置部分中提供了高级配置选项，并且仅适用于在显示字段中选择的元素（见上文），以及蜂窝单元的背景颜色或阈值。



主/副标签

类型 选择标签类型：
文本 - 标签将显示为文本参数指定的文本；
值 - 标签将显示监控项值，小数位数由小数位数参数指定。

文本 输入标签文本。此文本可覆盖默认监控项名称。
支持多行文本。也可以组合使用文本和支持的宏。
支持 {HOST.*}、{ITEM.*}、{INVENTORY.*} 以及用户宏。
蜂窝单元按主机名的字母顺序排序，并且在每个主机内按监控项名称排序。
当类型设置为文本时，此参数可用。

小数位数 输入值显示时的小数位数。
当类型设置为值时，此参数可用，并且仅影响返回numeric (float)数据的监控项。

大小 选择标签大小：
自动 - 使用自动调整的标签大小；
自定义 - 输入自定义标签大小（百分比，相对于蜂窝单元大小）。
请注意，超出蜂窝单元大小的标签将被截断。

粗体 选中复选框以粗体显示标签。

颜色 从颜色选择器中选择标签颜色。
“D” 表示默认颜色，取决于前端主题。要恢复默认颜色，请在颜色选择器中单击使用默认值按钮。

单位

单位 选中复选框以在监控项值中显示单位。
如果输入单位名称，它将覆盖监控项配置中设置的单位。
当类型设置为值时，此参数可用。

位置 选择监控项单位的位置（在监控项值之前或之后）。
对于以下时间相关单位，将忽略此参数：unixtime、uptime、s。
当类型设置为值且已选择单位选项时，此参数可用。

背景颜色

背景颜色 从颜色选择器中选择蜂窝单元背景颜色。
“D” 表示默认颜色，取决于前端主题。要恢复默认颜色，请在颜色选择器中单击使用默认值按钮。

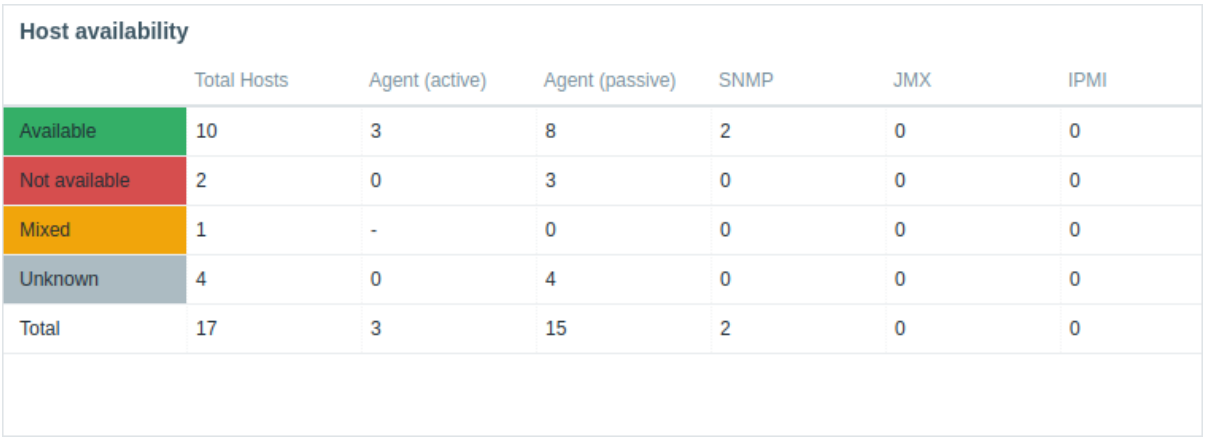
阈值

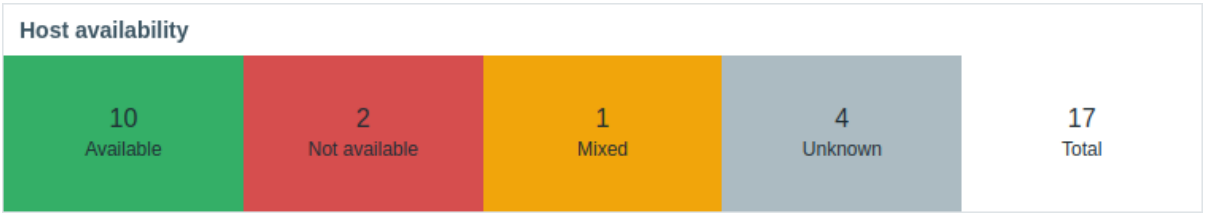
颜色插值	选中复选框以启用蜂窝单元阈值颜色之间的平滑过渡。 当设置了两个或更多阈值时，此参数可用。
阈值	单击添加以添加阈值，从颜色选择器中选择阈值颜色，并指定一个数值。 保存时，阈值列表将按升序排序。 请注意，配置为阈值的颜色仅对数值型监控项正确显示。 支持后缀（例如 “1d”、“2w”、“4K”、“8G”）。支持值映射。

主机可用性

概述

主机可用性小组件显示所选主机组中的主机可用性，使您能够监控处于运行或停止状态的主机数量。



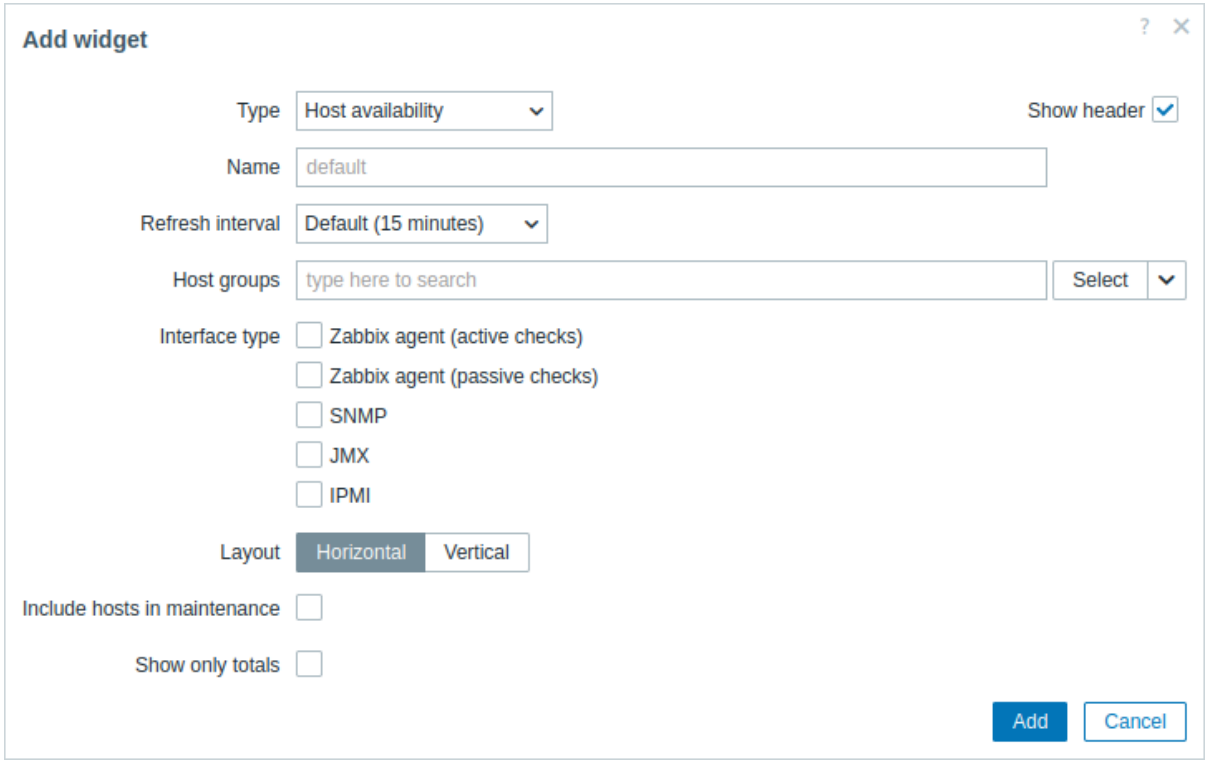


主机可用性的统计方式如下：

- 可用 - 所有接口均可用的主机
- 不可用 - 所有接口均不可用的主机
- 混合 - 至少有一个接口不可用，且至少有一个接口可用或未知的主机
- 未知 - 至少有一个接口状态未知，但没有任何接口不可用的主机
- 总计 - 所有主机总数

配置

要进行配置，请选择 主机可用性作为类型：



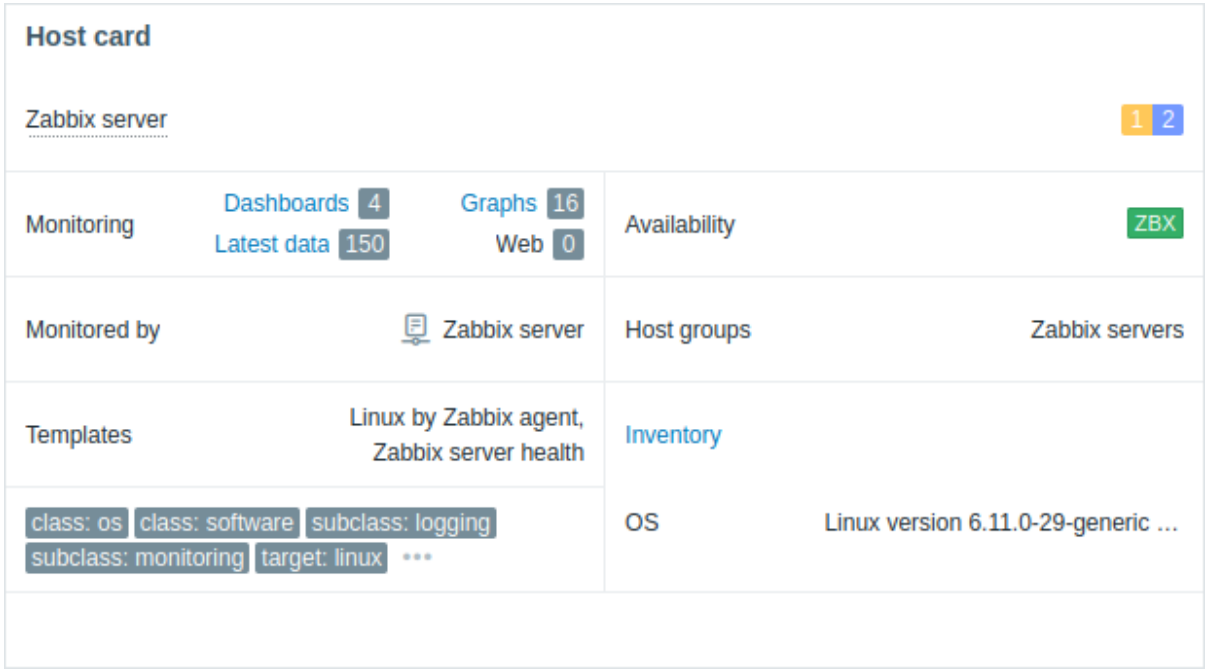
除了所有小部件都通用的常见参数外，您还可以设置以下特定选项：

主机群组	选择主机群组。 或者，选择一个兼容的小部件作为主机群组的数据源。 此字段支持自动补全，因此开始输入群组名称时，将显示包含匹配群组的下拉列表。 在模板仪表板上配置小部件时，此参数不可用。
接口类型	选择您希望查看哪些主机接口的可用性数据——*Zabbix agent*（主动检查）、*Zabbix agent*（被动检查）、*SNMP*、*JMX*、*IPMI*。 对于 *Zabbix agent*（主动检查），*Mixed* 单元格将始终为空，因为此类监控项不能有多个接口。 如果未选择任何项，则默认显示所有接口的可用性。
布局	选择水平显示（列）或垂直显示（行）。
包含维护中的主机	在统计中包含处于维护状态的主机。 在模板仪表板上配置小部件时，此参数标记为 在维护中显示数据。
仅显示总计	如果选中，则显示主机总数，而不按接口细分。如果仅选择了一个接口，则此选项将被禁用。

主机卡片

概述

主机卡片部件显示单个主机的详细信息，使您能够快速评估主机的状态和配置（可用性、问题、资产记录等）。

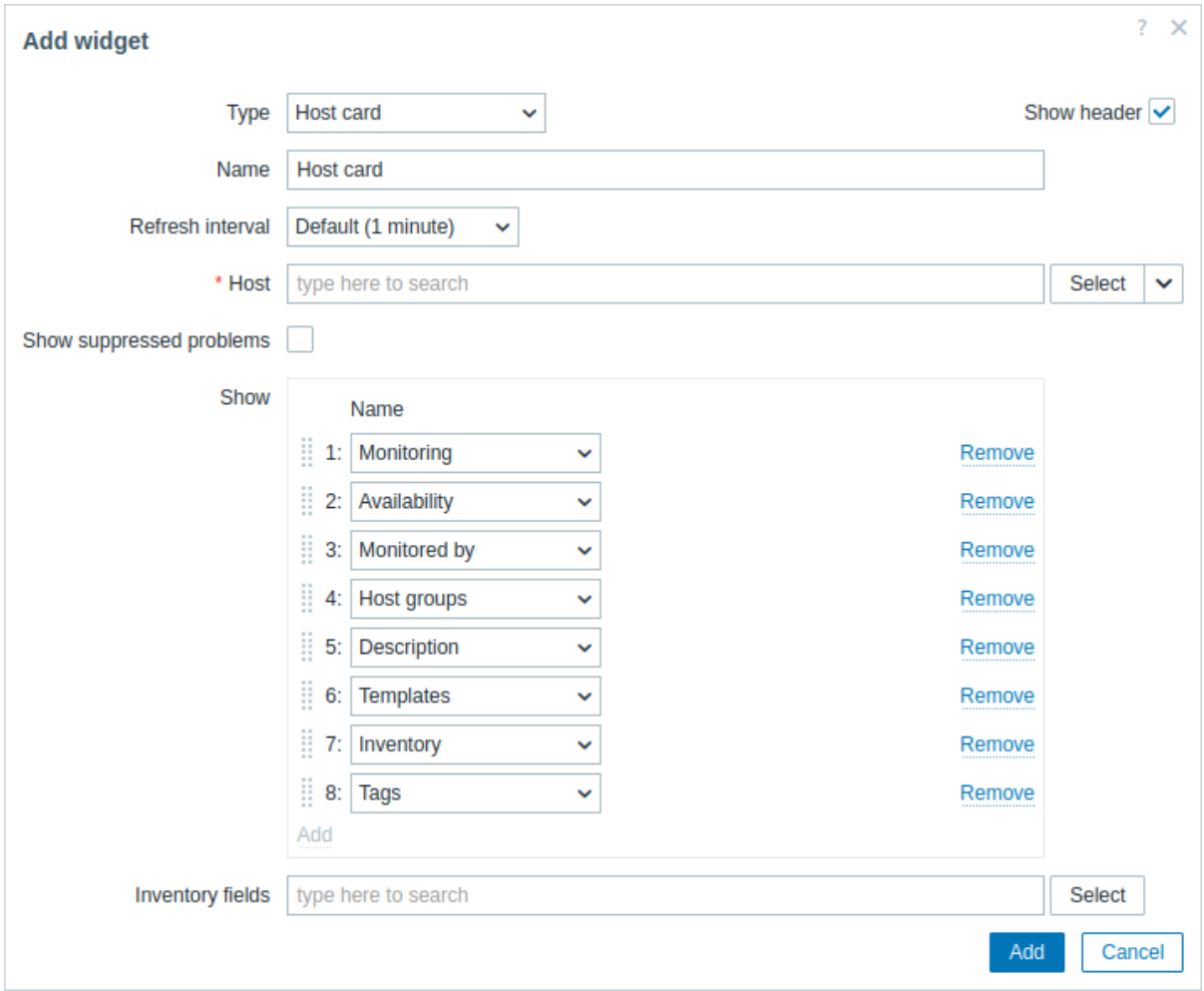


可以在部件配置中直接设置主机，也可以从兼容的部件或仪表板上的主机选择器中选择主机。

该部件可显示多个区块，每个区块展示不同的信息。
区块布局会根据部件宽度自动调整；当水平扩展时，区块会自动重新排列为多列。

配置

要进行配置，请选择 主机卡片作为类型：



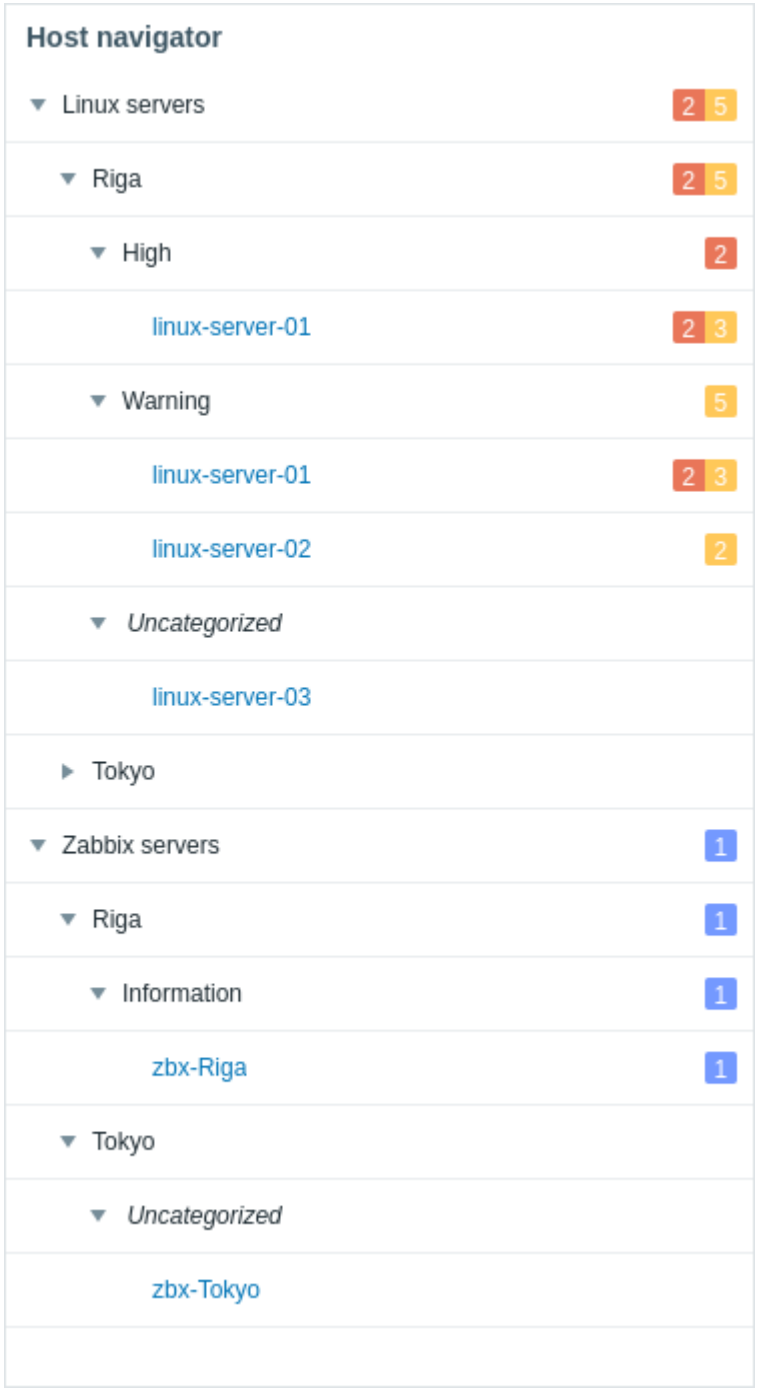
除了所有小部件都通用的常用参数外，还可以设置以下特定参数：

主机		选择主机。 或者，选择兼容的小部件或仪表板作为主机的数据源。 输入时会显示匹配建议。 小部件顶部区域始终显示主机名称（点击后会打开主机菜单）： - 如果主机已禁用，则会在主机名称旁显示红色的 *Disabled* 标签。 - 如果主机处于维护中，则会在主机名称旁显示维护图标。 小部件顶部区域还会显示按严重性分组的主机问题数量（点击后会打开主机的问题页面）。 在模板仪表板上配置小部件时，此参数不可用。
显示已抑制的问题		勾选此复选框可在问题计算中包含已抑制的问题。
显示		添加要在小部件中显示的区块。可通过拖放重新排列区块。
	主机群组	显示该主机所属的主机群组。
	描述	显示主机描述。
	监控	显示用于跳转到主机仪表板、最新数据、图形和Web 场景的快速链接。每个链接旁的数字表示相应实体的数量。
	可用性	按接口显示主机可用性。
	监控方式	显示该主机是由 Zabbix 服务器还是由特定的 proxy 或 proxy 组监控。
	模板	显示链接到该主机的模板。
	资产记录	显示主机资产记录字段。
	标签	显示主机标签。
资产记录字段		选择要显示的资产记录字段。 输入时会显示匹配建议。 如果未指定资产记录字段，则会显示所有已填充的资产记录字段。 仅当 显示包含 资产记录时，此参数才可用。

主机导航器

概述

主机导航器小部件根据各种过滤和分组选项显示主机列表。



用于组织主机的组可以展开或折叠。

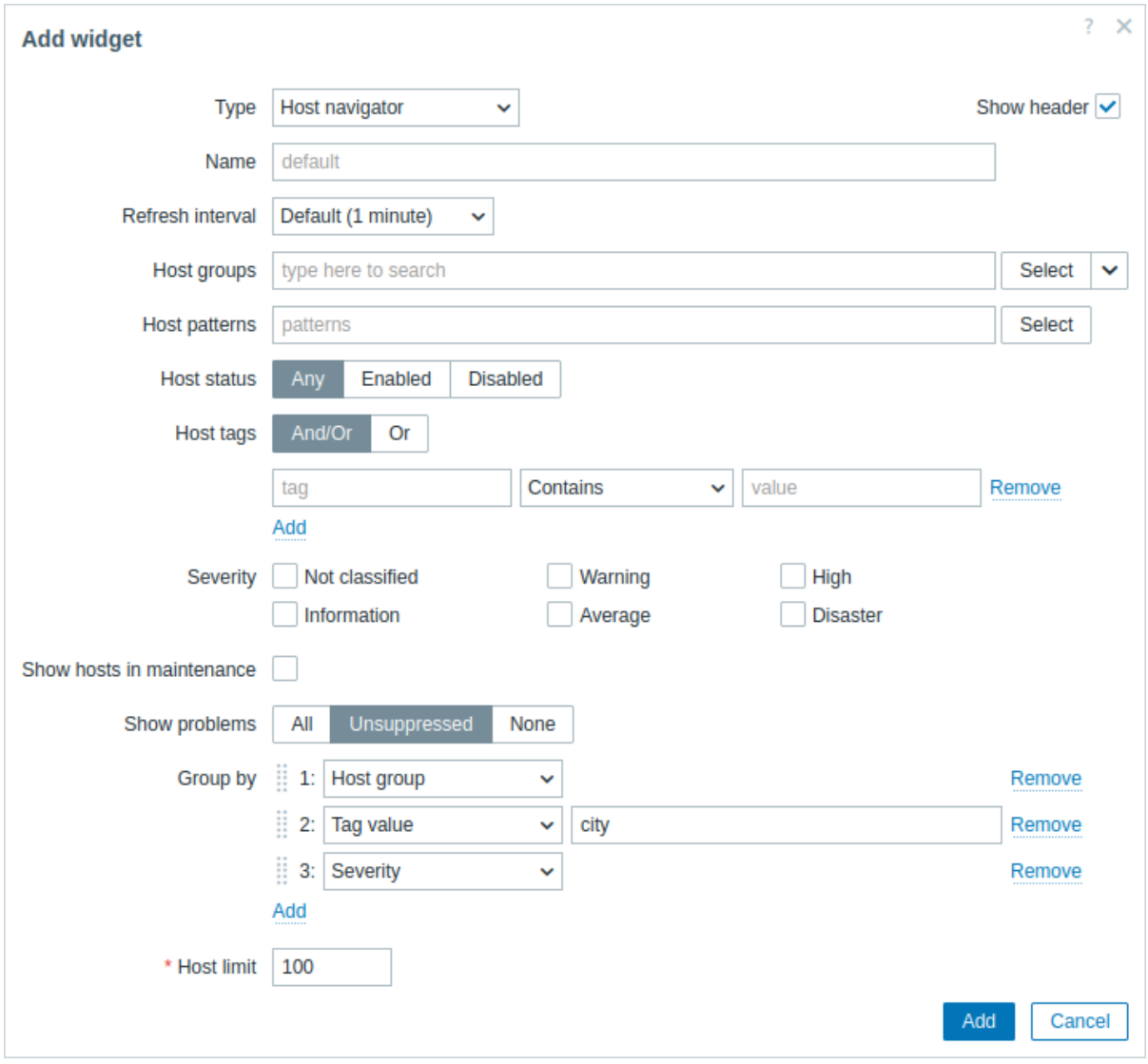
对于组、问题以及处于维护中的主机，可通过鼠标悬停提示查看附加详细信息。

该小部件最适合用于控制其他小部件根据所选主机显示的内容。



配置

要进行配置，请选择 *Host navigator* 作为类型：



除了适用于所有小组件的通用参数外，还可以设置以下特定选项：

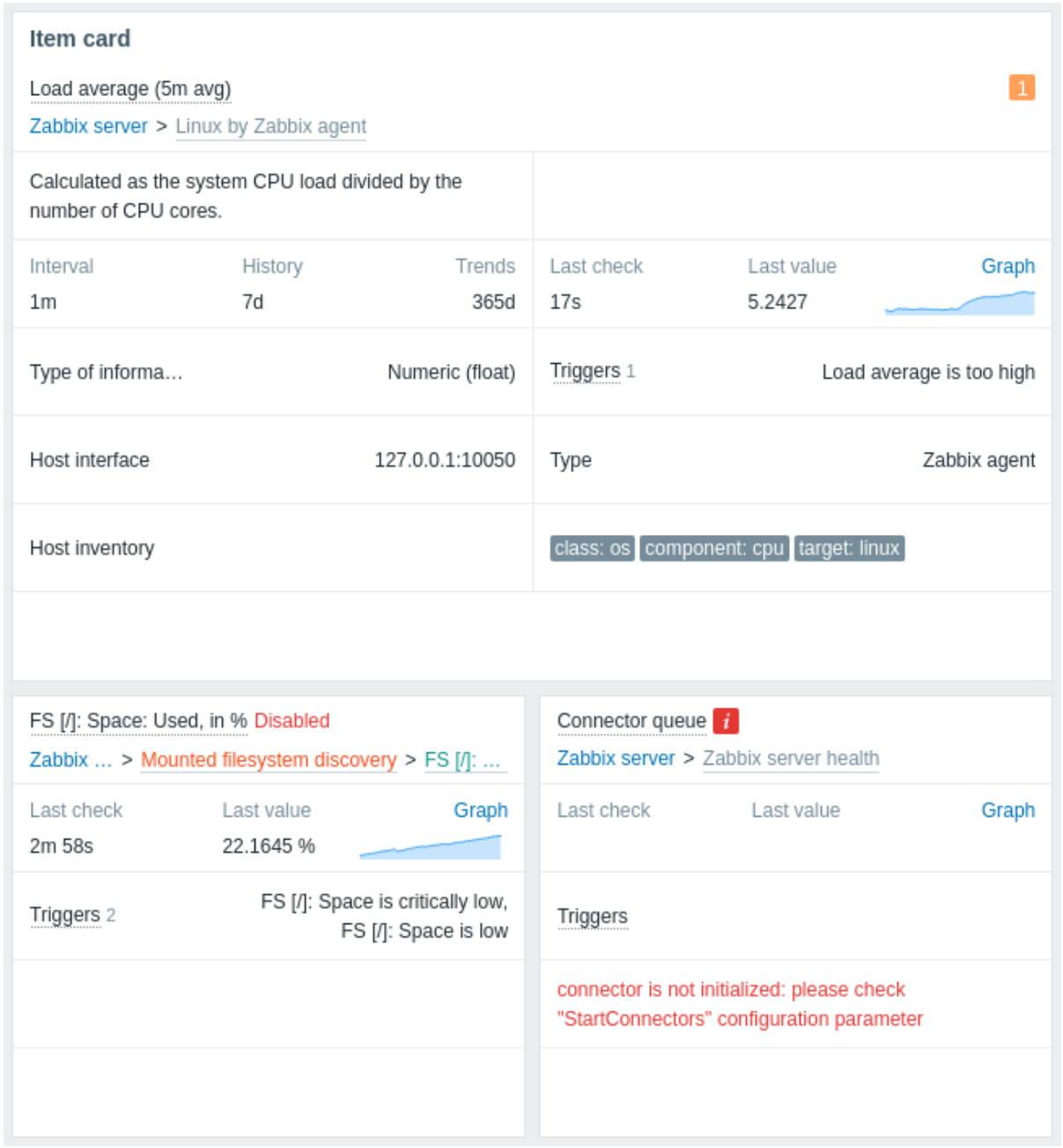
主机群组	选择主机群组。 或者，选择兼容的小组件作为主机群组的数据源。 此字段支持自动补全，因此开始输入群组名称时，将显示包含匹配群组的下拉列表。 选择父主机群组会隐式选择所有嵌套的主机群组；如果未选择主机群组，小组件将显示所有主机群组中的所有主机。 在模板仪表板上配置小组件时，此参数不可用。
主机模式	输入主机模式，或选择现有主机作为主机模式。符合指定模式的主机将显示在主机导航器中。 此字段支持自动补全，因此开始输入主机名称时，将显示包含匹配主机的下拉列表。 如果未选择主机，小组件将显示所有主机。 选择时可以使用通配符模式（例如，`*` 将返回匹配零个或多个字符的主机；`Zabbix*` 将返回以”Zabbix” 开头的主机）。 要指定通配符模式，请手动输入字符串并按 *Enter*。开始输入时，下拉列表将显示匹配的主机；如果选择了 主机群组，则仅显示属于所选主机群组中主机的匹配主机。 在模板仪表板上配置小组件时，此参数不可用。
主机状态	根据主机状态（任意、已启用、已禁用）筛选要显示的主机。 在模板仪表板上配置小组件时，此参数不可用。
主机标签	指定标签，以筛选小组件中显示的主机。 可以包含或排除特定标签及标签值。可以设置多个条件。 标签名称匹配始终区分大小写。 每个条件有以下运算符可用： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件；具有相同标签名称的条件将按 Or 条件分组； **Or** - 满足一个条件即可。 在模板仪表板上配置小组件时，此参数不可用。
严重性	标记问题严重性，以筛选要在小组件中显示的存在问题的主机。 如果未标记任何严重性，将显示存在所有问题的所有主机。
显示处于维护中的主机	选中此复选框可显示处于维护中的主机（此时，主机名称旁将显示维护图标）。 在模板仪表板上配置小组件时，此参数标记为 显示维护中的数据。
显示问题	根据问题状态（全部、未抑制、无）筛选要显示的问题。 问题以主机行末尾的彩色块表示。颜色基于问题严重性，可在 管理 > 常规 > 触发器显示选项中自定义。 更新问题时可以更改问题严重性。

分组依据	添加分组属性，以便根据该属性对所选主机进行分组： 主机群组 - 按主机所属的主机群组对主机分组； 标签值 - 输入标签名称，以便根据该标签的值对主机分组（例如，输入”city”，即可按”Riga”、”Tokyo”等值对主机分组）； 严重性 - 按问题严重性对主机分组。 如果将 显示问题配置为显示问题，则显示方式如下： - 对于每个严重性组，仅显示相应的问题数量； - 对于每台主机，显示其所有问题数量。 请注意，主机仅按 严重性参数中标记的严重性进行分组；如果未标记任何严重性，所有主机将按所有严重性进行分组。 可以通过拖动组名称前的控件上下移动来重新排列分组属性。请注意，分组属性的顺序决定了分组嵌套顺序。例如，指定多个标签名称（1：color，2：city）后，主机将先按颜色（red、blue 等）分组，然后按城市（Riga、Tokyo 等）分组。 根据配置的分组属性，一台主机可能显示在多个组中（例如，按主机群组分组时，主机属于多个主机群组）。单击此类主机，会在所有组中选中并突出显示该主机。 不符合配置的分组属性的主机将显示在 未分类组中。 最多可以指定 10 个分组属性，并且所有属性必须唯一。 如果未指定分组属性，主机将不会分组。
主机限制	输入要显示的主机最大数量。可输入的值范围为 1 至 9999。 当可显示的主机数量超过设置的限制时，显示的主机下方会显示相应消息（例如，” 显示了 100 个主机，共 100+ 个”）。 请注意，配置的主机限制也会影响已配置组的显示；例如，如果主机限制设置为 100，并按标签值分组（超过 200 个），小组件中将仅显示前 100 个标签值及其对应的主机。 此参数不受 管理 > 常规 > *GUI*中的 搜索和筛选结果限制参数影响。 在模板仪表板上配置小组件时，此参数不可用。

监控项卡片

概述

监控项卡片小组件显示单个监控项的详细信息，使您能够快速评估监控项的状态和配置（最新数据、错误、描述等）。

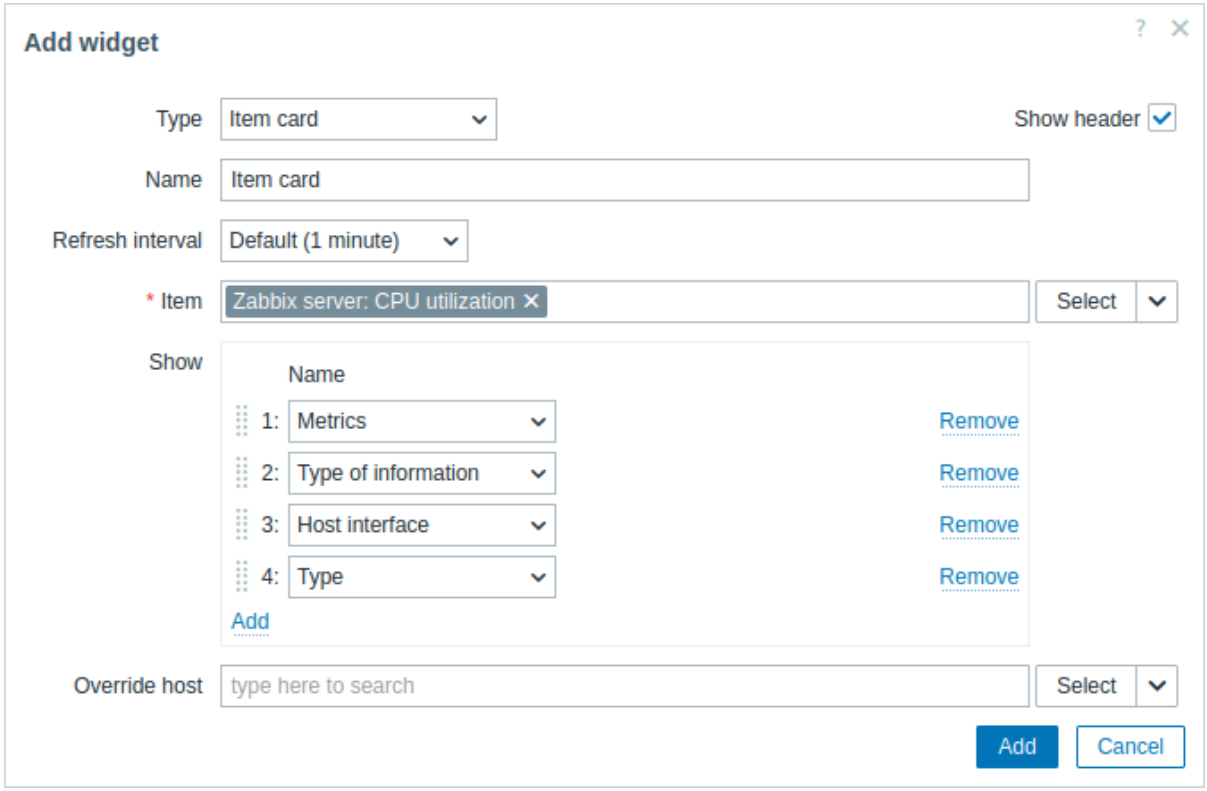


该监控项可以在小组件配置中直接设置，也可以从仪表板上的兼容小组件中选择。

该小组件可显示多个部分，每个部分展示不同的信息。部分布局会根据小组件宽度自动调整；当水平扩展时，各部分会自动重新排列为多列。

配置

要进行配置，请选择 *Item card* 作为类型：



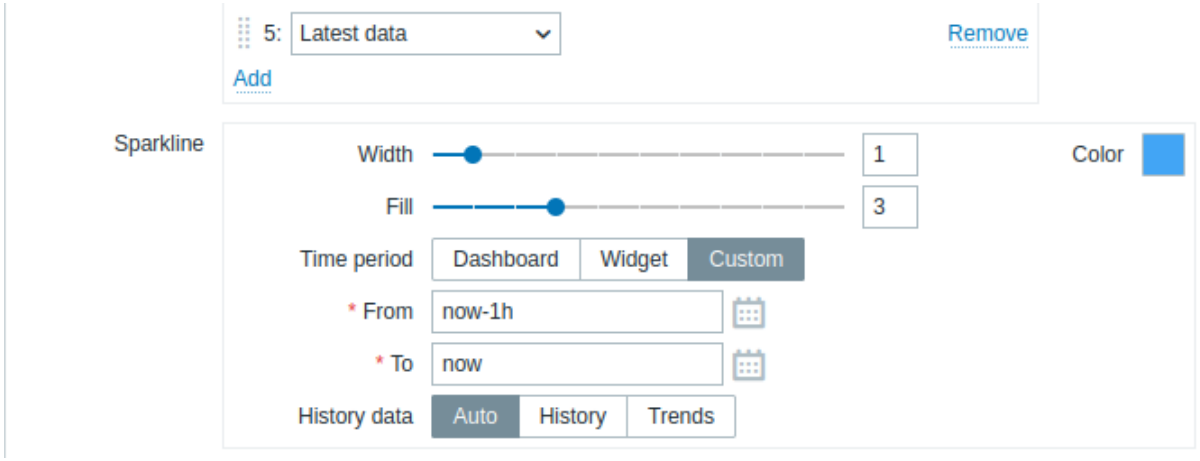
除了所有小部件通用的常见参数外，您还可以设置以下特定选项：

Item	选择监控项。 或者，选择一个兼容的小部件作为监控项的数据源。 输入时会显示匹配建议。 小部件顶部区域始终显示监控项名称（点击可打开监控项菜单）： - 如果监控项已禁用，则会在监控项名称旁显示红色的 *Disabled* 标签。 - 如果存在错误，则会显示带字母 “i” 的方形图标。如果存在问题，则会针对每个问题严重性显示带有问题数量的方形图标。将鼠标悬停在图标上会打开包含详细信息的弹出窗口。 - 在模板仪表板中，显示的是监控项的技术名称（而不是可见名称）。 小部件顶部区域还会显示监控项路径——其主机，以及在适用情况下的模板、低级别发现（LLD）规则或主监控项（点击某个实体会打开其配置）。 只有对相应实体具有权限的用户才能看到监控项路径。
Show	添加要在小部件中显示的区块。区块可通过拖放重新排列。
Description	显示监控项描述。宏会被解析。
Error text	显示监控项错误（如有）。
Metrics	显示以下信息： - 监控项更新间隔（trapper items除外）； - 监控项历史记录配置（如有）； - 监控项趋势配置（如有）。 宏会被解析。 如果监控项使用了自定义间隔，则会在监控项更新间隔旁显示一个帮助图标。将鼠标悬停在该图标上会打开包含自定义间隔详细信息的弹出窗口。
Latest data	显示以下信息： - 自上次检查监控项以来的时间； - 监控项最新值； - 指向监控项值的简单图形/历史记录的链接（如果存储了历史记录或趋势）； - 可配置的sparkline 图表（如果监控项为数值型）。 对于二进制类型的监控项，显示的是缩略图或 *Show* 选项，而不是最新值。将鼠标悬停在缩略图上会打开包含图像的弹出窗口。将鼠标悬停在 *Show* 上会打开包含监控项值（Base64 字符串）的弹出窗口。 在模板仪表板中，显示的是 *No data*，而不是监控项最新值。
Type of information	显示监控项的信息类型。

Triggers	显示以下信息： - 带有触发器数量的 *Triggers* 标签（将鼠标悬停在 *Triggers* 上会打开包含触发器详细信息的弹出窗口）； - 触发器名称列表（宏会被解析）。
Host interface	显示监控项使用的主机接口；如果监控项没有主机接口（例如，依赖监控项、简单检查），则显示 *No data*。 在模板仪表板中，显示 *No data*。
Type	显示监控项类型。
Host inventory	显示由监控项值填充的主机资产记录字段。
Tags	显示监控项标签。
Override	选择一个兼容的小部件或仪表板主机选择器作为主机的数据源。
host	在模板仪表板上配置该小部件时，此参数不可用。

迷你图

仅当将 最新数据部分添加到该小部件后，才可使用迷你图的图表配置选项。

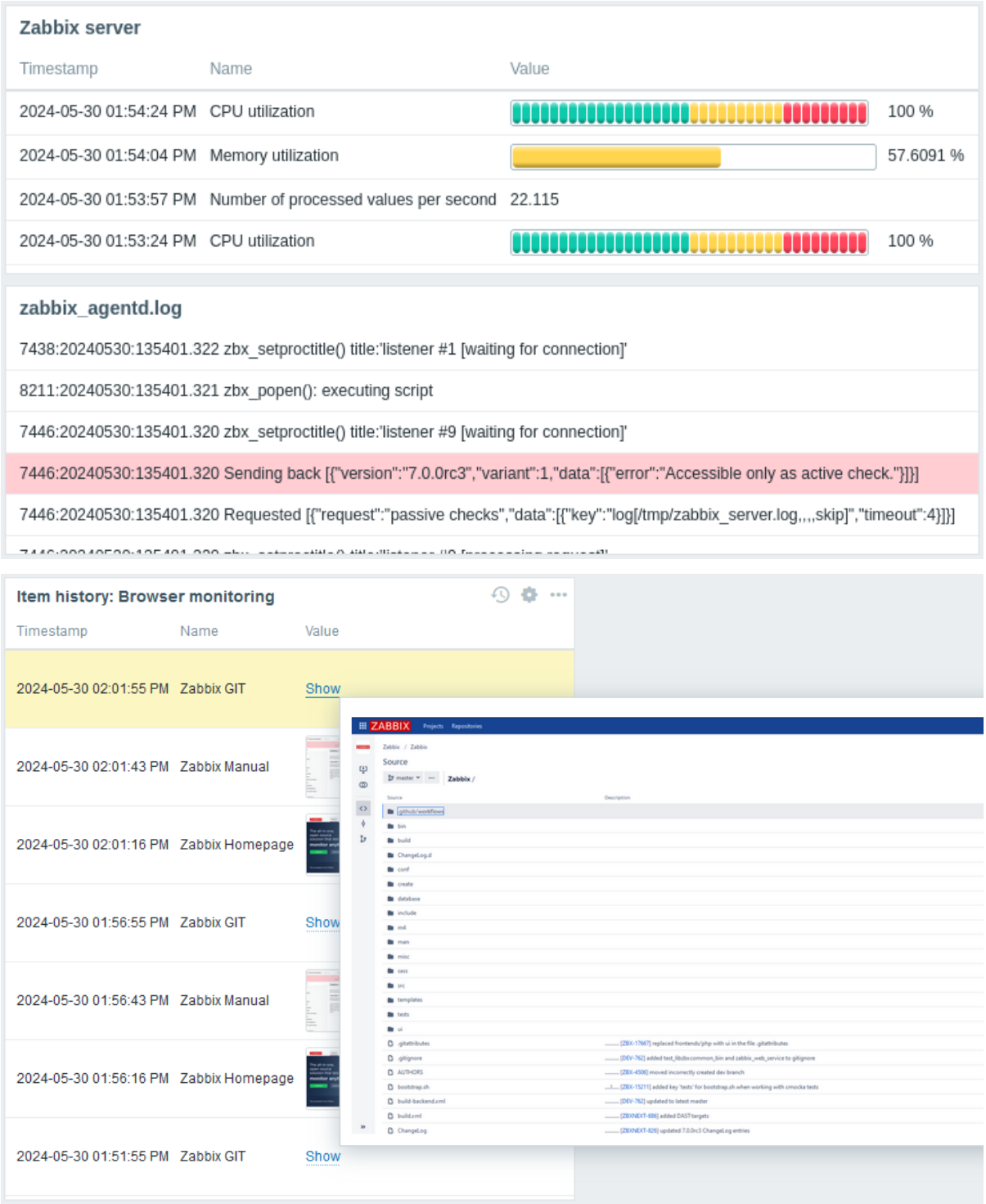


宽度	使用滑块或手动输入 0 到 10 范围内的值，以设置图形线条粗细。
颜色	选择线条和填充颜色。
填充	使用滑块或手动输入 0 到 10 范围内的值，以设置填充颜色的透明度级别。
时间周期	为时间周期选择数据源： 仪表板 - 使用仪表板的时间周期选择器； 小部件 - 使用兼容的小部件（在 小部件参数中设置）； 自定义 - 使用在 从和 到参数中设置的自定义时间周期；如果已设置，则会在小部件右上角显示一个时钟图标，鼠标悬停时会提示所设置的时间。 请注意，无论小部件的 时间周期如何配置，兼容的小部件仍可将其用作时间周期的数据源。
小部件	输入或选择一个兼容的小部件，作为时间周期的数据源。 仅当 时间周期设置为 “小部件” 时，此参数可用。
从	输入或选择时间周期的开始时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 仅当 时间周期设置为 “自定义” 时，此参数可用。
到	输入或选择时间周期的结束时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 仅当 时间周期设置为 “自定义” 时，此参数可用。
历史数据	从历史数据或趋势数据中取值： 自动 - 自动选择； 历史 - 使用历史数据； 趋势 - 使用趋势数据。 此参数仅适用于数值型数据。非数值型数据始终从历史数据中获取。

监控项历史

概述

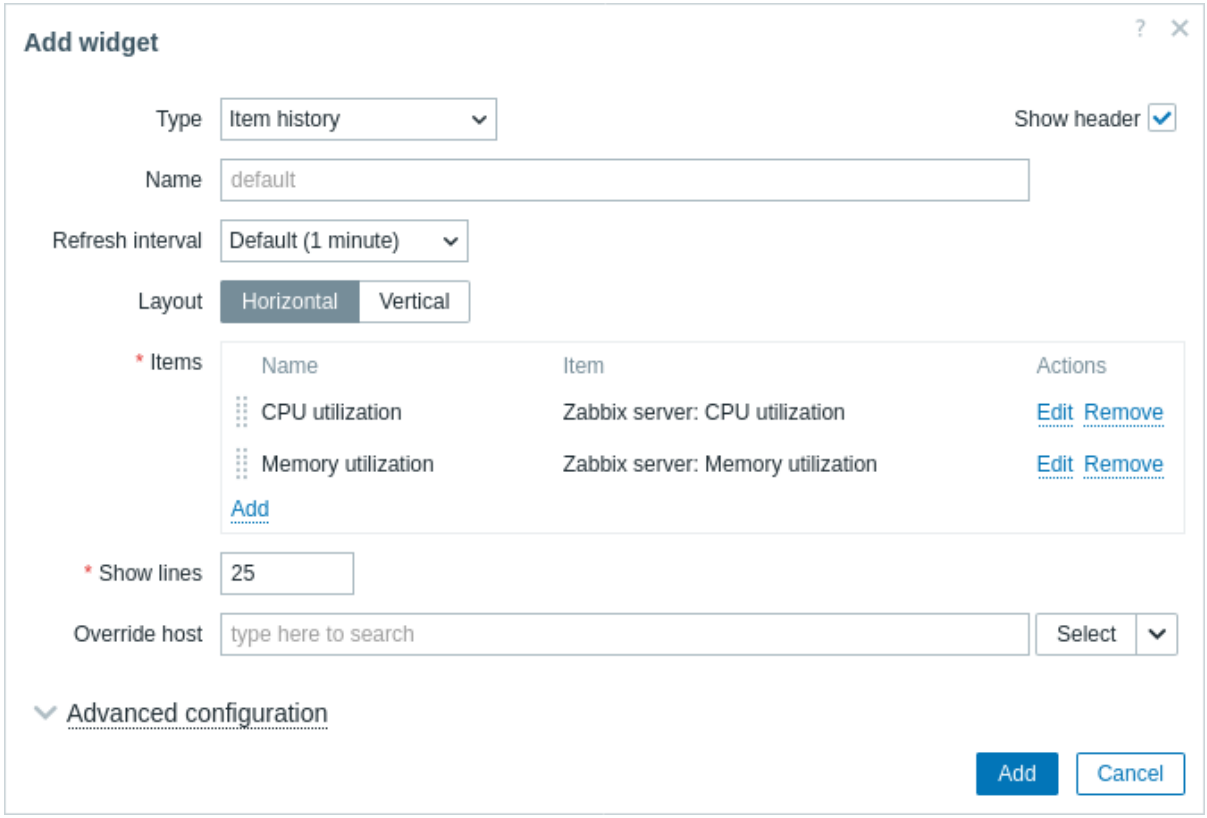
监控项历史小组件以表格形式显示所有监控项类型（数值、文本等）的最新数据。它可以显示进度条和图像（对browser items 很有用），并且可以高亮显示值（对log file monitoring 很有用）。



最多可显示 1000 条记录。

配置

要进行配置，请选择 *Item history* 作为类型：

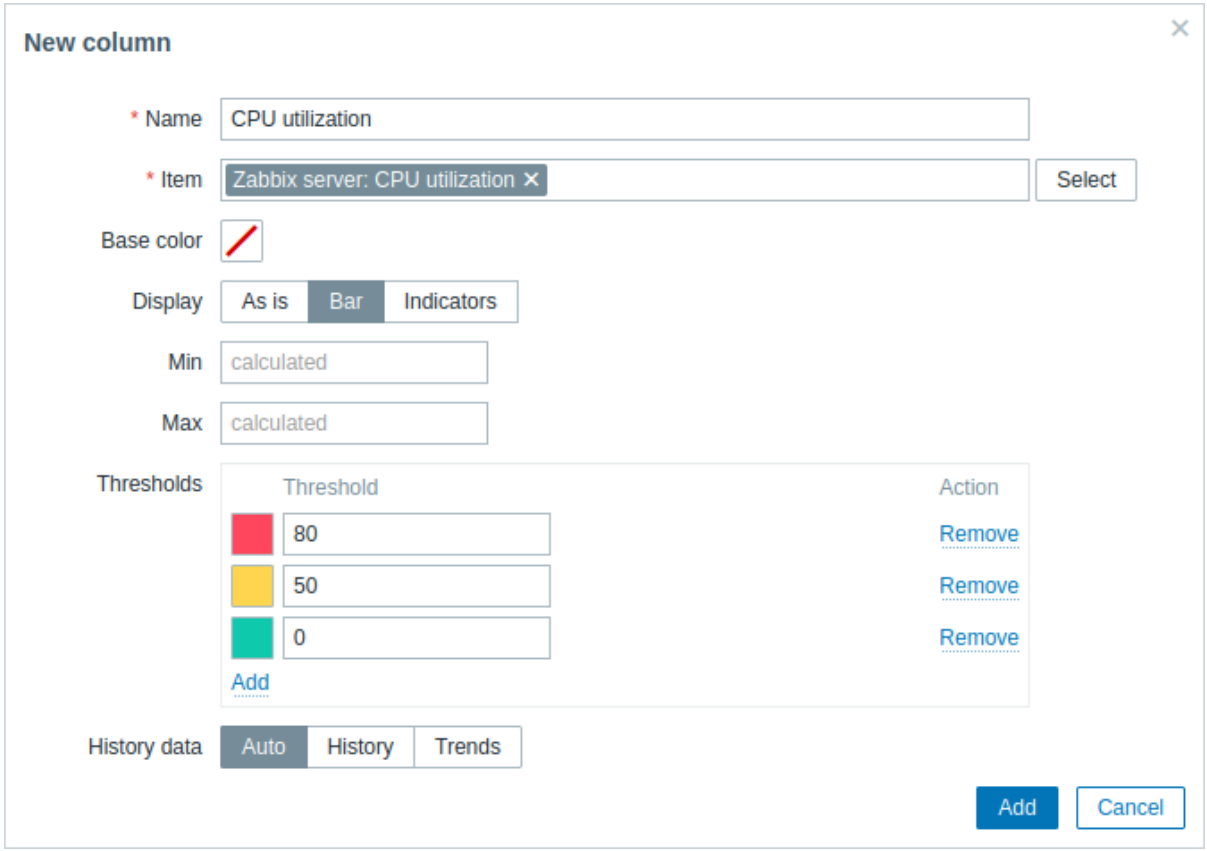


除了所有部件通用的常见参数外，还可以设置以下特定选项：

Layout	为监控项列选择布局选项： **Horizontal** - 监控项将水平显示，值将垂直显示； **Vertical** - 监控项将垂直显示，值将水平显示。
Items	添加要显示的监控项列。 监控项的顺序决定其显示顺序。 可以通过拖动监控项名称前的手柄上下移动来重新排序监控项。
Show lines	指定要显示的监控项值行数。
Override host	为主机选择兼容的部件或仪表板主机选择器作为数据源。 在模板仪表板上配置该部件时，此参数不可用。
Advanced configuration	点击 *Advanced configuration* 标签以显示高级配置选项。

列配置

要配置监控项列，请在 监控项参数中单击 添加：



常用列参数：

名称	输入列的名称。 如果留空，则使用监控项参数中的监控项名称。
监控项	选择监控项。 请注意，列配置参数根据所选监控项的信息类型而有所不同；有关更多信息，请参阅下面的各个参数。 在模板仪表板上配置组件时，只能选择在模板上配置的监控项。
基本颜色	如果显示设置为“条形图”或“指标”，请选择列的背景颜色或填充颜色。 请注意，阈值或突出显示颜色可以覆盖基本颜色。

特定于数字类型监控的列参数：

显示	选择监控值的显示方式： 原样 - 作为常规文本； 条形图 - 作为实心、颜色填充的条形图； 指标 - 作为分段、颜色填充的条形图。
最小值	输入条形图/指标的最小值。 如果留空，组件将使用监控的最小值。 此参数仅在显示设置为“条形图”或“指标”时可用。
最大值	输入条形图/指标的最大值。 如果留空，组件将使用监控的最大值。 此参数仅在显示设置为“条形图”或“指标”时可用。
阈值	单击“添加”可添加阈值，从颜色选择器中选择阈值颜色，并指定数值。 阈值列表在保存时将按升序排列。 支持后缀（例如，“1d”、“2w”、“4K”、“8G”）。支持值映射。
历史数据	选择从历史记录还是趋势中获取数据： 自动 - 自动选择； 历史 - 获取历史数据； 趋势 - 获取趋势数据。

字符、文本、日志和 JSON 类型监控项特有的列参数：

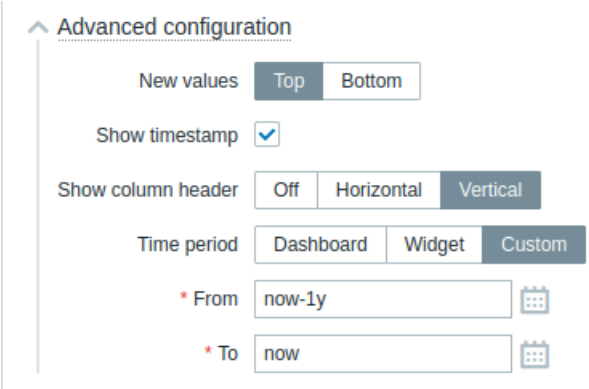
Highlights	单击 *Add* 添加高亮，使用颜色选择器选择高亮颜色，并指定正则表达式。 所选颜色将用作监控项值的背景色，当指定的正则表达式与文本匹配时生效。
Display	选择监控项值的显示方式： **As is** - 按接收时的原样显示。如果值包含换行符，则保留换行；如果 *Layout* 设置为"Vertical" 且配置了多于一列，则会应用自动换行； **HTML** - 作为 HTML 格式文本显示； **Single line** - 作为单行显示，并截断为指定长度（1-500 个字符）。 将鼠标悬停在截断值上或单击该值，会弹出窗口显示完整内容。
Use monospace font	勾选此复选框可使用等宽字体显示监控项值（默认未勾选）。
Display local time	勾选此复选框可在时间戳列中显示本地时间而不是时间戳。 请注意，*advanced configuration* 中的 *Show timestamp* 复选框也必须勾选。 此参数仅适用于日志类型监控项。

二进制类型监控项特有的列参数：

显示缩略图	选中此复选框可为图像二进制数据值显示缩略图，或为非图像二进制数据值显示 "显示" 选项。 取消选中此复选框可为所有二进制监控项值显示 "显示" 选项。 将鼠标悬停在 "显示" 选项上或单击该选项，会打开一个包含监控项值（图像或 Base64 字符串）的弹出窗口。 如果监控项值为空字符串，则会显示 "显示" 选项；将鼠标悬停在其上或单击它，会打开一个包含 "空字符串" 的弹出窗口。

高级配置

可折叠的高级配置部分中提供了高级配置选项：



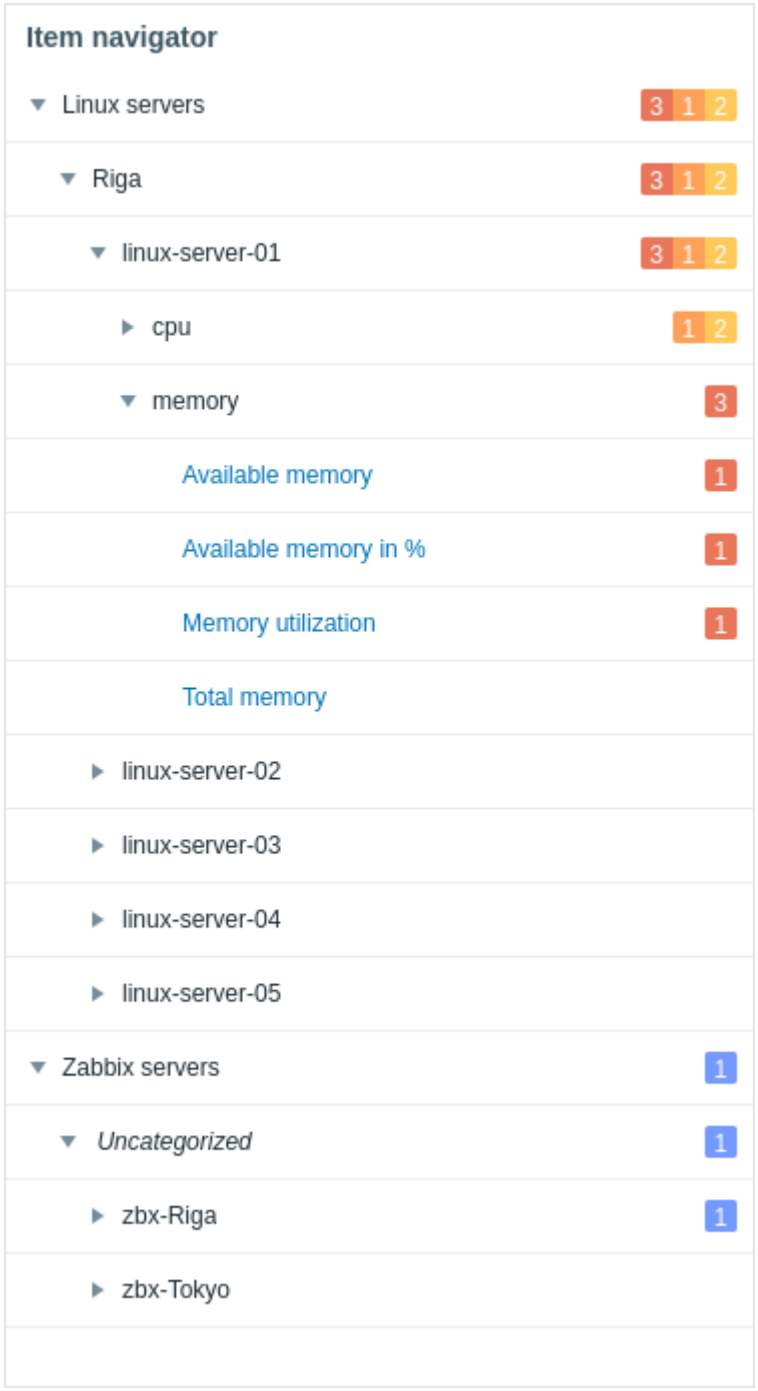
新值	选择新监控项值的添加位置： 顶部 - 在列顶部； 底部 - 在列底部。
显示时间戳	勾选此复选框以显示时间戳列（默认不勾选）。
显示列标题	选择列标题方向： 关闭 - 隐藏标题； 水平 - 水平显示标题； 垂直 - 垂直显示标题。
时间段	为时间段选择数据源： 仪表板 - 使用仪表板的时间段选择器； 部件 - 使用兼容的部件（在部件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，部件右上角将显示一个时钟图标，鼠标悬停时会提示所设置的时间。 请注意，无论部件的时间段如何配置，兼容的部件仍可将其用作时间段的数据源。
部件	输入或选择一个兼容的部件（图形、图形（经典）、图形原型）作为时间段的数据源。 如果时间段设置为 "部件"，则此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 如果时间段设置为 "自定义"，则此参数可用。

到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 如果时间段设置为“自定义”，则此参数可用。

监控项导航器

概述

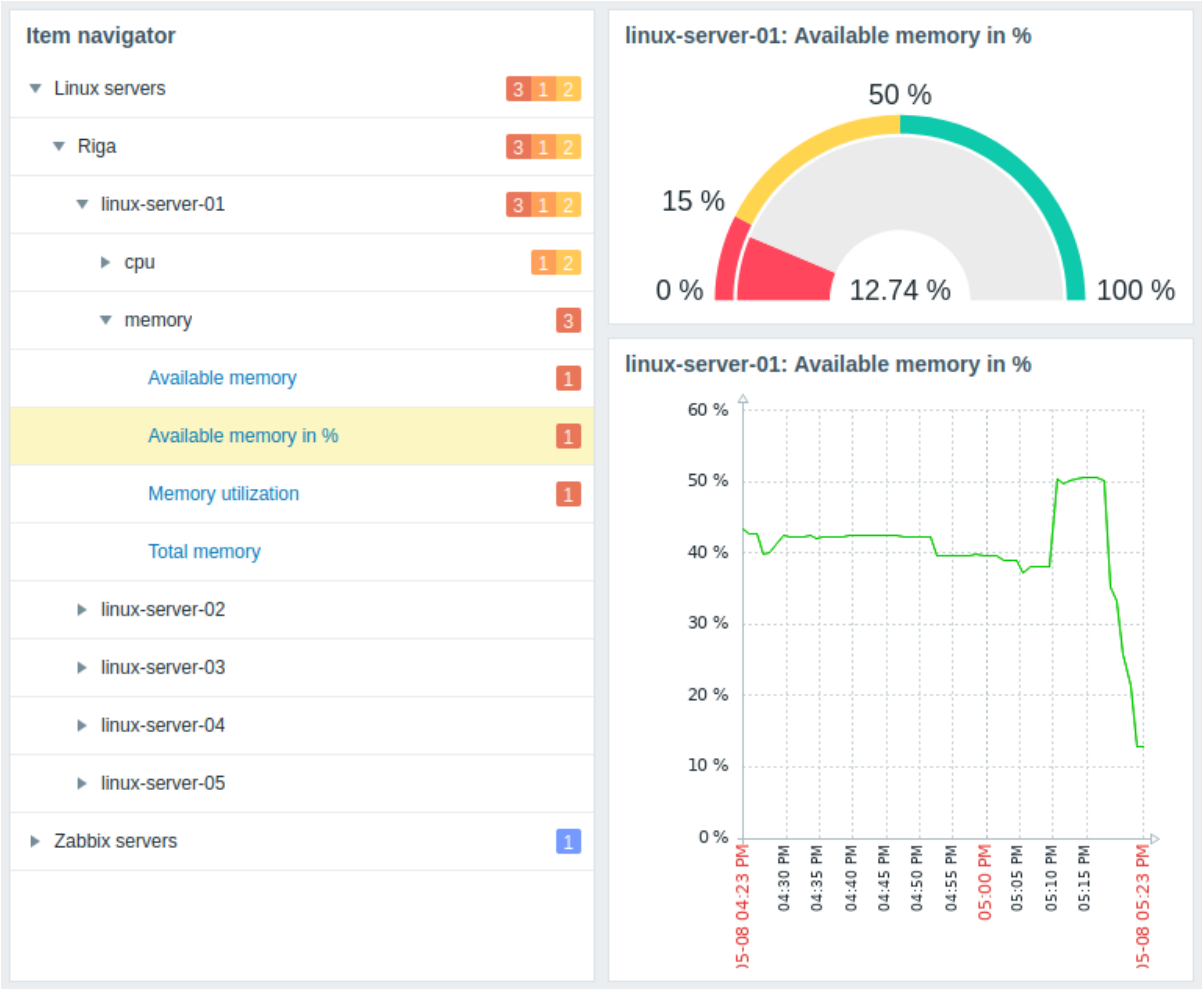
监控项导航器小组件根据各种筛选和分组选项显示监控项列表。



用于组织监控项的分组可以展开或折叠。

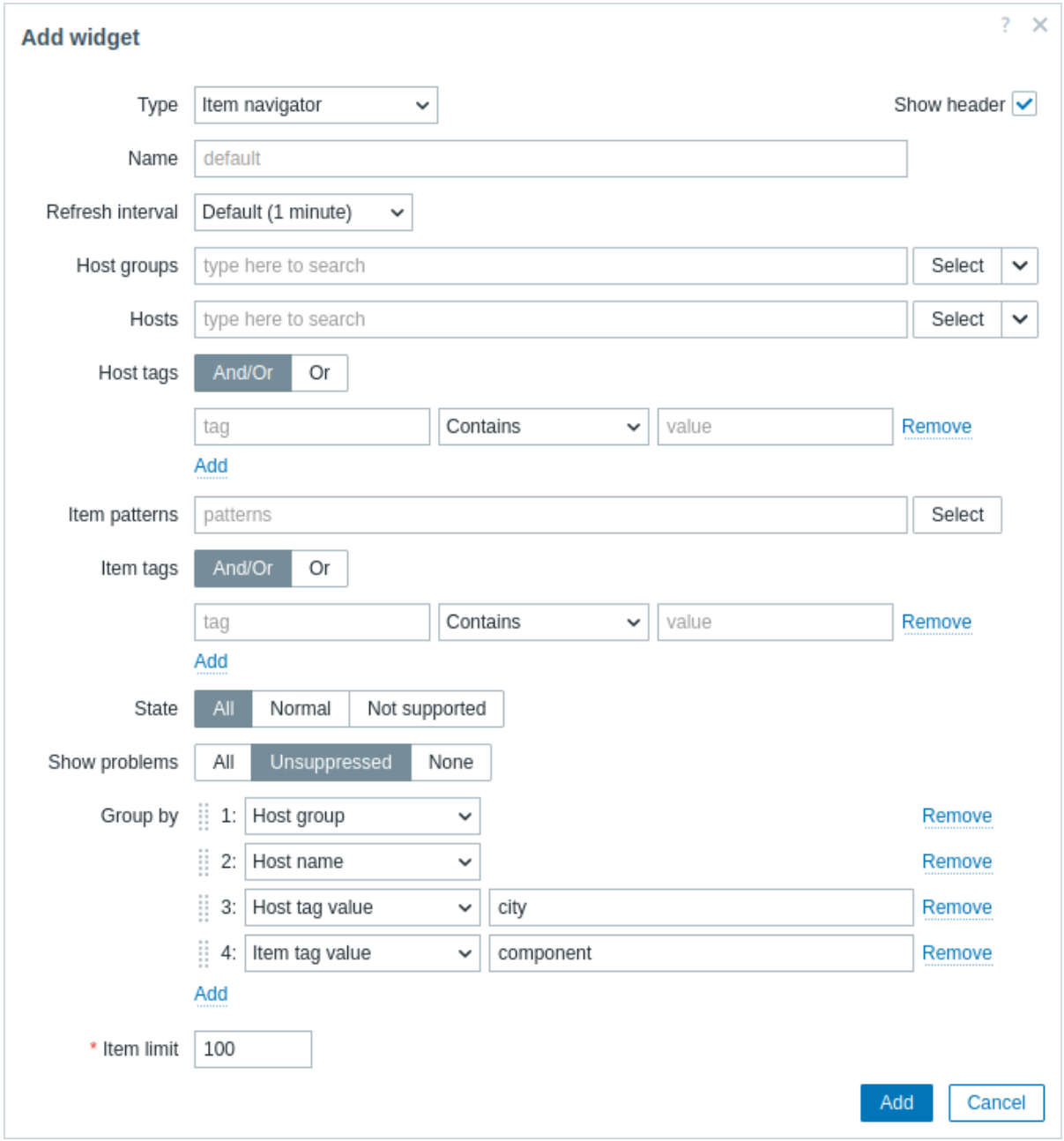
对于分组和问题，可通过鼠标悬停提示查看附加详细信息。

该小组件最适合用于控制其他小组件根据所选监控项显示的内容。



配置

要进行配置，请选择 监控项导航器作为类型：



除了适用于所有小组件的通用参数外，还可以设置以下特定选项：

主机组	选择主机组。 或者，选择兼容的小组件作为主机组的数据源。 此字段支持自动完成，因此开始输入主机组名称时，将显示包含匹配主机组的下拉列表。 选择父主机组会隐式选择所有嵌套的主机组；如果未选择主机组，小组件将显示属于所有主机组中所有主机的监控项。 在模板仪表板上配置小组件时，此参数不可用。
主机	选择主机。 或者，选择兼容的小组件或仪表板作为主机的数据源。 此字段支持自动完成，因此开始输入主机名称时，将显示包含匹配主机的下拉列表。 如果未选择主机，小组件将显示属于所有主机的监控项。 在模板仪表板上配置小组件时，此参数不可用。

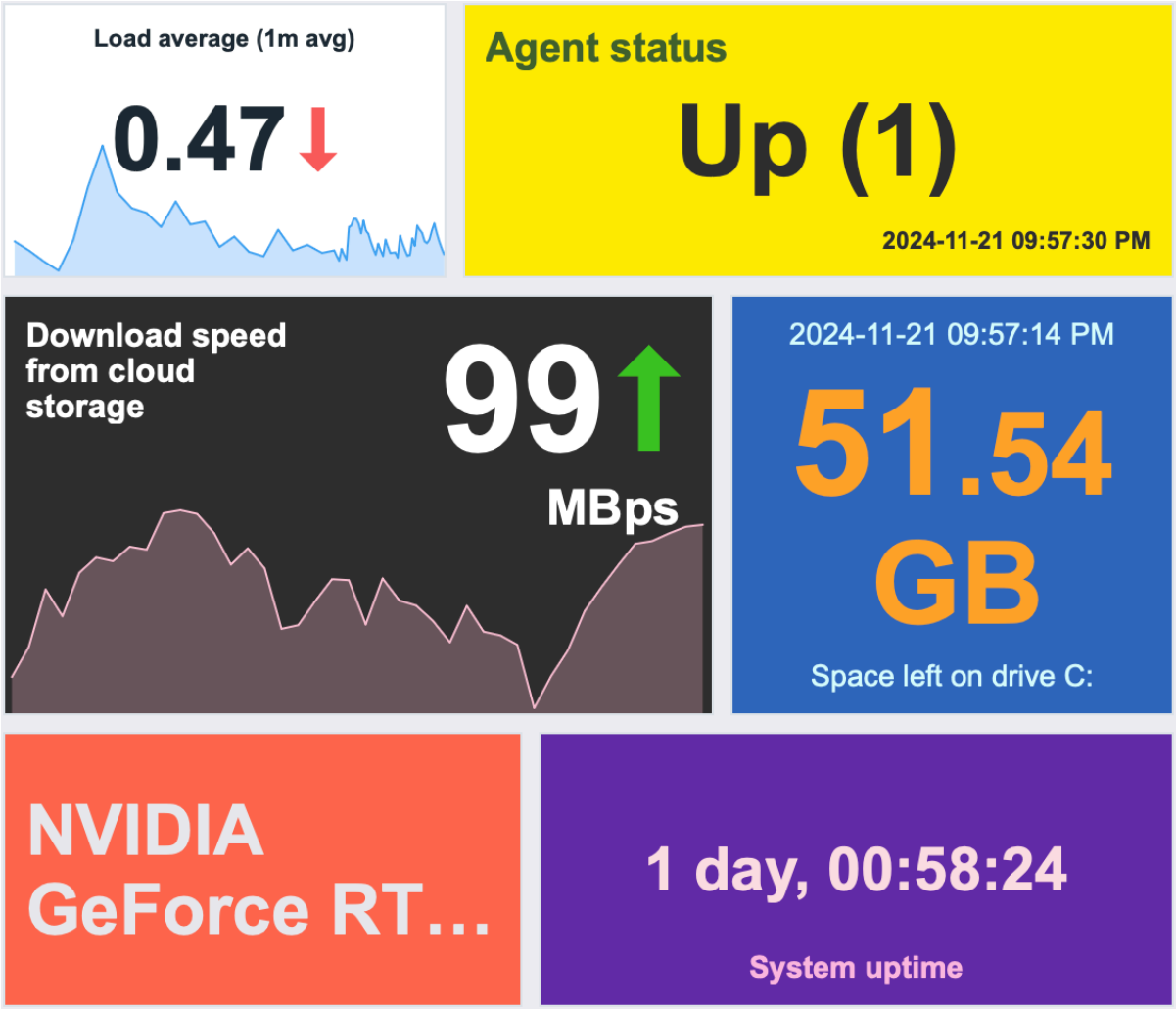
主机标签	指定主机标签，以筛选小组件中显示的监控项。 可以包含或排除特定标签及标签值。可以设置多个条件。 标签名称匹配始终区分大小写。 每个条件有以下运算符可用： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件；具有相同标签名称的条件将通过 “或” 条件进行分组； 或 - 满足一个条件即可。 在模板仪表板上配置小组件时，此参数不可用。
监控项模式	输入监控项模式，或选择现有监控项作为监控项模式。匹配指定模式的监控项将显示在监控项导航器中。 选择时可以使用通配符模式（例如，* 将返回匹配零个或多个字符的监控项；Zabbix* 将返回以 “Zabbix” 开头的监控项）。 要指定通配符模式，请手动输入字符串并按 *Enter*。开始输入时，下拉列表将显示匹配的监控项；如果已选择 主机或 主机组，则仅显示属于所选主机或所选主机组中主机的监控项。通配符始终会被解释，因此，如果存在其他匹配的监控项（例如 item2、item3），则无法单独添加名为 *item** 的监控项。 在模板仪表板上配置小组件时，此参数仅允许选择在模板上配置的监控项。
监控项标签	指定监控项标签，以筛选小组件中显示的监控项。 可以包含或排除特定标签及标签值。可以设置多个条件。 标签名称匹配始终区分大小写。 每个条件有以下运算符可用： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件；具有相同标签名称的条件将通过 “或” 条件进行分组； 或 - 满足一个条件即可。
状态	根据监控项的状态（全部、正常、不支持）筛选要显示的监控项。
显示问题	根据问题状态（全部、未抑制、无）筛选要显示的问题。 问题会以彩色块显示在监控项行末尾。颜色基于问题严重性，可在 管理 > 常规 > 触发器显示选项中自定义。更新问题时可以更改问题严重性。

分组依据	添加用于对监控项进行分组的分组属性： 主机组 - 按主机所属的主机组对监控项分组； 主机名称 - 按主机对监控项分组； 主机标签值 - 输入标签名称，按该主机标签的值对监控项分组（例如，输入 “city”，按 “Riga”、“Tokyo” 等值对监控项分组）； 监控项标签值 - 输入标签名称，按该监控项标签的值对监控项分组（例如，输入 “component”，按 “cpu”、“memory” 等值对监控项分组）。 可以通过拖动组名称前的控件上下移动来重新排列分组属性。请注意，分组属性的顺序决定了组的嵌套顺序。例如，指定多个主机标签名称（1：color，2：city）后，监控项将先按 color（red、blue 等）分组，再按 city（Riga、Tokyo 等）分组。 根据配置的分组属性，一个监控项可能显示在多个组中（例如，按主机组分组时，监控项所属的主机属于多个主机组）。单击此类监控项会在所有组中选中并突出显示该监控项。 不匹配所配置分组属性的监控项将显示在 未分类组中。 如果将 显示问题配置为显示问题，则问题显示方式如下： - 对于每个组，显示所有下属监控项的问题数量； - 对于每个监控项，仅显示该监控项的问题数量。 最多可以指定 10 个分组属性，并且所有属性必须唯一。 如果未指定分组属性，则不会对监控项进行分组。
监控项限制	输入要显示的监控项最大数量。可输入的值范围为 1 至 9999。 当可显示的监控项数量超过设置的限制时，显示的监控项下方会显示相应消息（例如，“显示了 100 个监控项，共 100+ 个”）。 请注意，配置的监控项限制也会影响已配置组的显示；例如，如果监控项限制设置为 100，并且按主机对监控项分组（每个主机包含 200 个监控项），则小组件中只会显示第一个主机及其 100 个监控项。 此参数不受 管理 > 常规 > *GUI*中的 搜索和筛选结果限制参数影响。

监控项值

概述

监控项值小组件显示单个数值型或字符串型监控项的值。它适用于持续关注重要指标、可视化阈值以及检测数据中的突然变化。



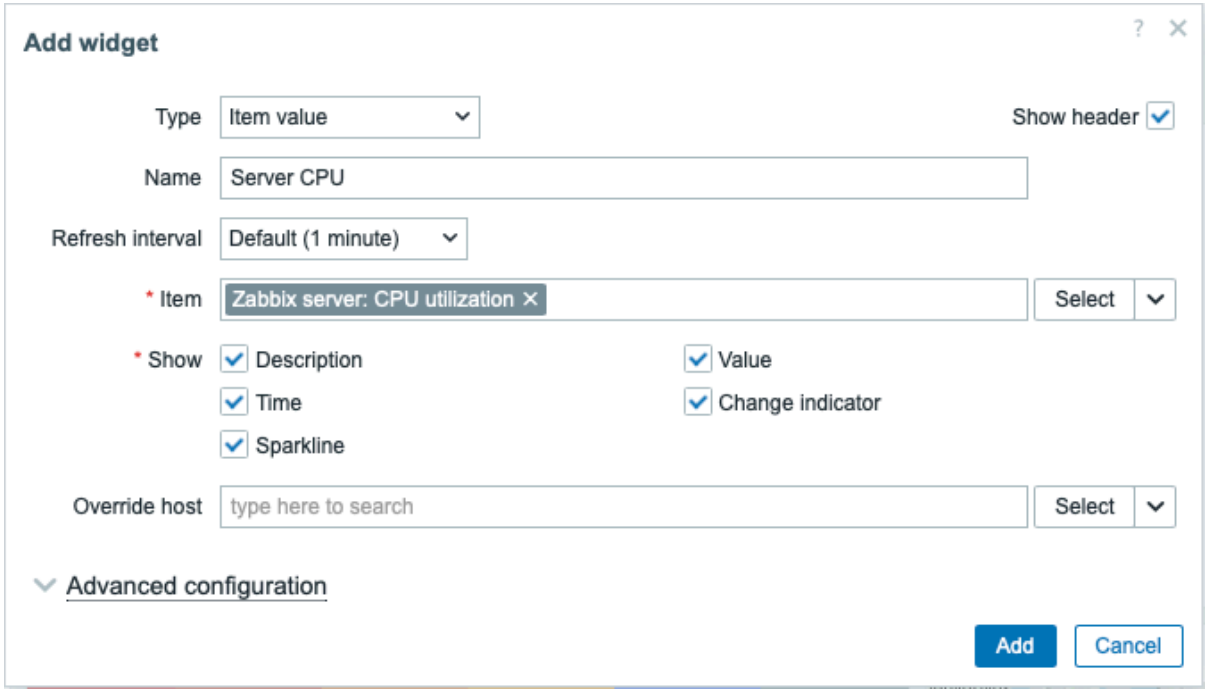
您可以将该小组件配置为显示以下内容：

- 指标时间戳 (2024-11-21 09:57:30)
- 监控项描述 (Load average (1m), Agent status)
- 监控项值、变化指示器和单位 (0.47 ↓, 99 ↑ Mbps)
- 背景颜色
- 指定时间段内数值的迷你趋势图

单击该小组件会为数值型监控项打开一个临时图形，或为字符串型监控项打开最新数据。

配置

要进行配置，请选择 监控项值作为小部件类型：



除了所有小部件通用的常用参数外，您还可以设置以下特定选项：

监控项	选择监控项。 或者，选择一个兼容的小部件作为该监控项的数据源。 不支持返回二进制数据的监控项。
显示	勾选复选框以显示相应元素：描述、值、时间、变化指示器、迷你图。取消勾选则隐藏。必须至少选择一个元素。
覆盖主机	选择一个兼容的小部件或仪表板的主机选择器作为主机的数据源。 在模板仪表板上配置该小部件时，此参数不可用。
高级配置	单击 高级配置标题以显示高级配置选项。

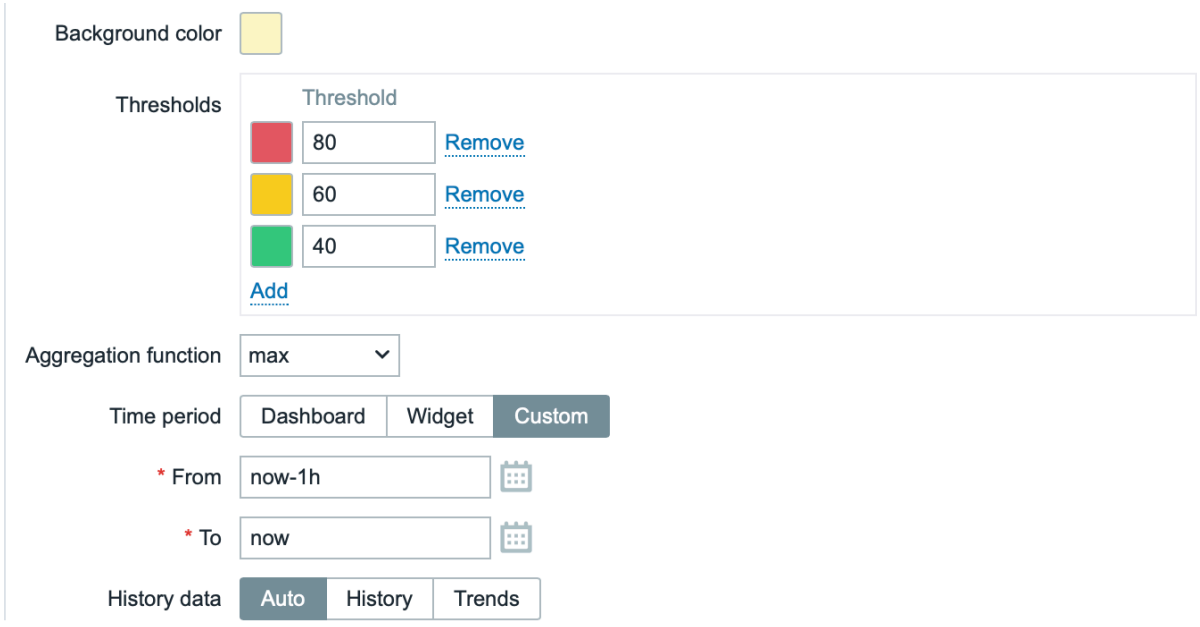
高级配置

可折叠的高级配置部分中提供了高级配置选项，并且仅适用于在显示字段中选定的那些元素（见上文）。

请注意，多个元素不能占用同一空间；如果它们被放置在同一空间中，将显示错误消息。

通用小组件参数

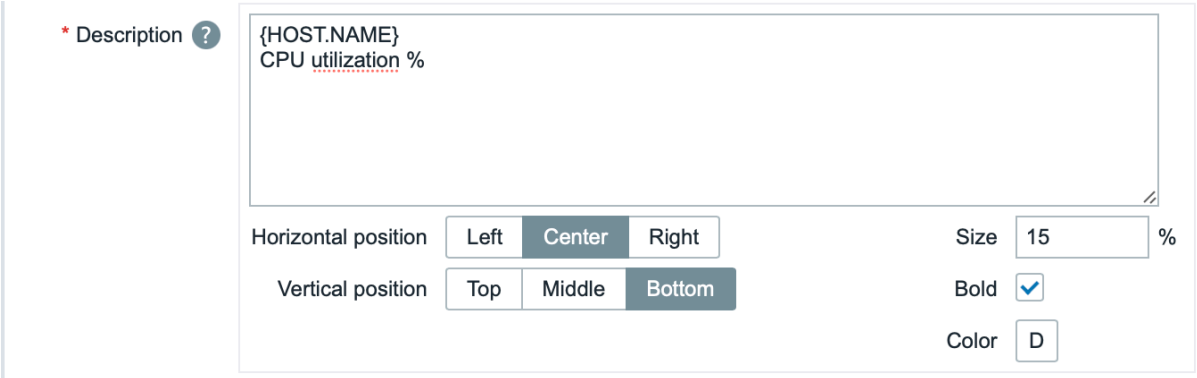
这些参数用于确定整个小组件的背景颜色（静态或动态），以及用于显示值的聚合函数。



背景颜色	从颜色选择器中选择整个小组件的背景颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中点击 使用默认值按钮。
阈值	配置整个小组件的动态背景颜色。点击 添加添加阈值，从颜色选择器中选择背景颜色，并指定数值。当监控项值等于或大于阈值时，背景颜色将发生变化。 保存时，列表将按升序排序。 请注意，只有数值型监控项才能正确显示动态背景颜色。
聚合函数	指定在选定的 时间段内使用的聚合函数： **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值； **count** - 显示值的数量； **sum** - 显示值的总和； **first** - 显示第一个值； **last** - 显示最后一个值。 如果选择 不使用，则显示最新值（不进行聚合）。 聚合允许显示指定时间间隔（5 分钟、1 小时、1 天）的聚合值，而不是最新值。 *min*、*max*、*avg* 和 *sum* 只能显示数值型数据。对于 *count*，非数值型数据将转换为数值。
时间段	用于聚合值的时间段。选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在 小组件参数中设置）； 自定义 - 使用在 从和 到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的 时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。 当 聚合函数设置为 “不使用” 时，此参数将隐藏。
小组件	输入或选择兼容的小组件，作为时间段的数据源。 当 时间段设置为 “小组件” 时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当 时间段设置为 “自定义” 时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当 时间段设置为 “自定义” 时，此参数可用。
历史数据	从历史数据或趋势数据中获取数据： 自动 - 自动选择； 历史 - 获取历史数据； 趋势 - 获取趋势数据。 此设置仅适用于数值型数据。非数值型数据始终从历史数据中获取。

描述

这些参数决定了监控项描述的显示方式。

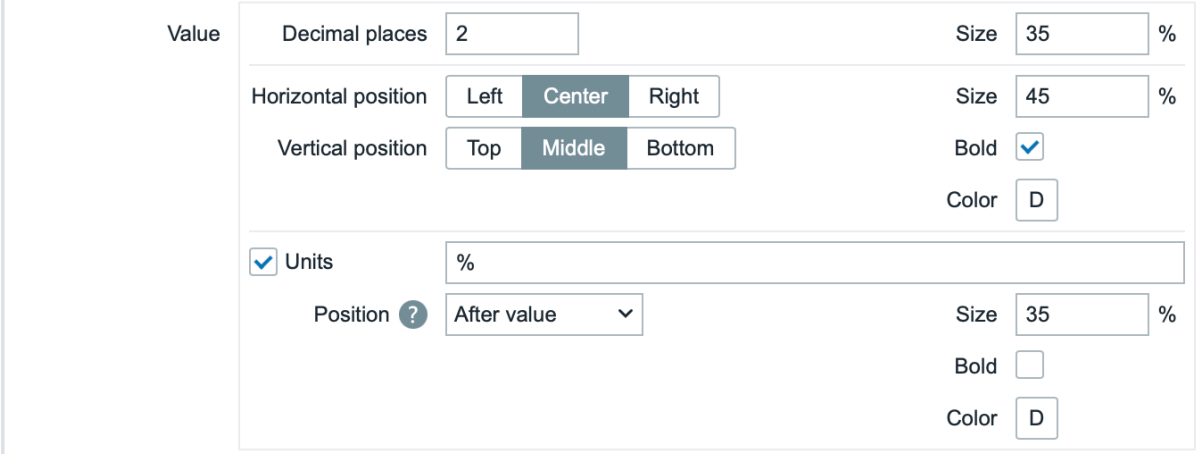


描述	输入监控项描述。此描述可覆盖默认的监控项名称。支持多行描述。可以组合使用文本和受支持的宏。 支持 {HOST.*}、{ITEM.*}、{INVENTORY.*} 以及用户宏。
水平位置	选择监控项描述的水平位置——左、右或居中。
垂直位置	选择监控项描述的垂直位置——上、下或中间。
大小	输入监控项描述的字体高度（以相对于整个小组件高度的百分比表示）。

粗体	勾选复选框以粗体显示监控项描述。
颜色	从颜色选择器中选择监控项描述的颜色。 D 表示默认颜色（取决于前端主题）。要恢复为默认值，请在颜色选择器中单击 *Use default* 按钮。

值

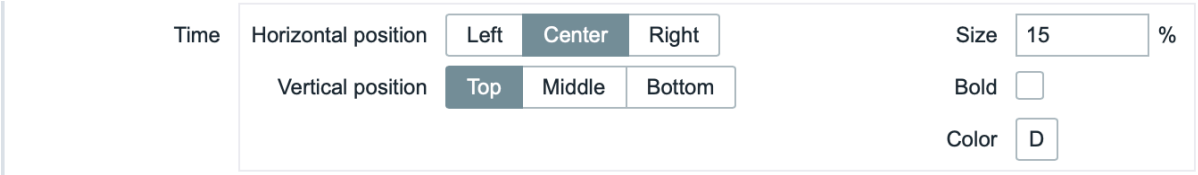
这些参数用于确定如何显示监控项值。



小数位数	选择值显示时保留的小数位数。此值仅影响浮点型监控项。对于单位设置为 “s” 的监控项，如果此值大于 0（默认值为 2），部件会对最高有效时间单位进行四舍五入，并显示数值（例如 “10.43m”），而不是将该值转换为完整的时间字符串。设置为 0 时，秒数将转换为人类可读的时间字符串（例如 “4h 56m 30s”）。
大小	输入小数位数字体高度（相对于部件总高度的百分比）。
水平位置	选择监控项值的水平位置——左、右或居中。
垂直位置	选择监控项值的垂直位置——上、下或中间。
大小	输入监控项值的字体高度（相对于部件总高度的百分比）。 请注意，监控项值的大小具有更高优先级；其他元素必须为该值让出空间。不过对于变化指示器，如果该值过大，则会被截断以显示变化指示器。
粗体	勾选复选框以粗体显示监控项值。
颜色	从颜色选择器中选择监控项值的颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中单击 使用默认值按钮。
单位	
单位	勾选复选框以在监控项值旁显示单位。如果输入单位名称，它将覆盖监控项配置中的单位。
位置	选择监控项单位的位置——位于值的上方、下方、前面或后面。
大小	输入监控项单位的字体高度（相对于部件总高度的百分比）。
粗体	勾选复选框以粗体显示监控项单位。
颜色	从颜色选择器中选择监控项单位的颜色。 D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中单击 使用默认值按钮。

时间

这些参数用于确定时间（来自监控项历史记录的时钟值）的显示方式。



Horizontal position	选择时间的水平位置——左、右或居中。
Vertical position	选择时间的垂直位置——上、下或中间。
Size	输入时间的字体高度（以相对于 widget 总高度的百分比表示）。
Bold	勾选复选框以粗体显示时间。
Color	从颜色选择器中选择时间颜色。 D 表示默认颜色（取决于前端主题）。要恢复为默认值，请在颜色选择器中单击 *Use default* 按钮。

变化指示器

此部分允许您通过颜色选择器选择变化指示器的颜色。



变化指示器如下：

- ↑ - 监控项值上升（适用于数值型监控项）
- ↓ - 监控项值下降（适用于数值型监控项）
- ↕ - 监控项值发生变化（适用于字符串型监控项以及使用值映射的监控项）

注意：

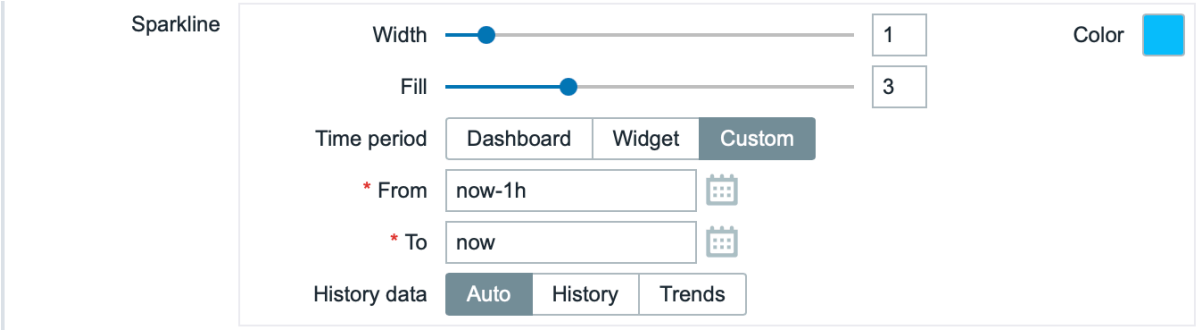
- 如果只有一个值，则不会显示上升（↑）和下降（↓）指示器。
- 值始终在相同范围内进行比较；例如：
  - 将最新值与前一个值进行比较。
  - 将月度值与上一个月进行比较。
  - 对于聚合数据，前一个周期的时长与所选周期相同，并且在所选周期开始前立即结束。

D 表示默认颜色（取决于前端主题）。要恢复默认值，请在颜色选择器中点击 使用默认值按钮。

变化指示器的垂直大小等于值的大小（对于数值型监控项，则等于值的整数部分）。

迷你图

这些参数用于确定迷你图表的显示方式。

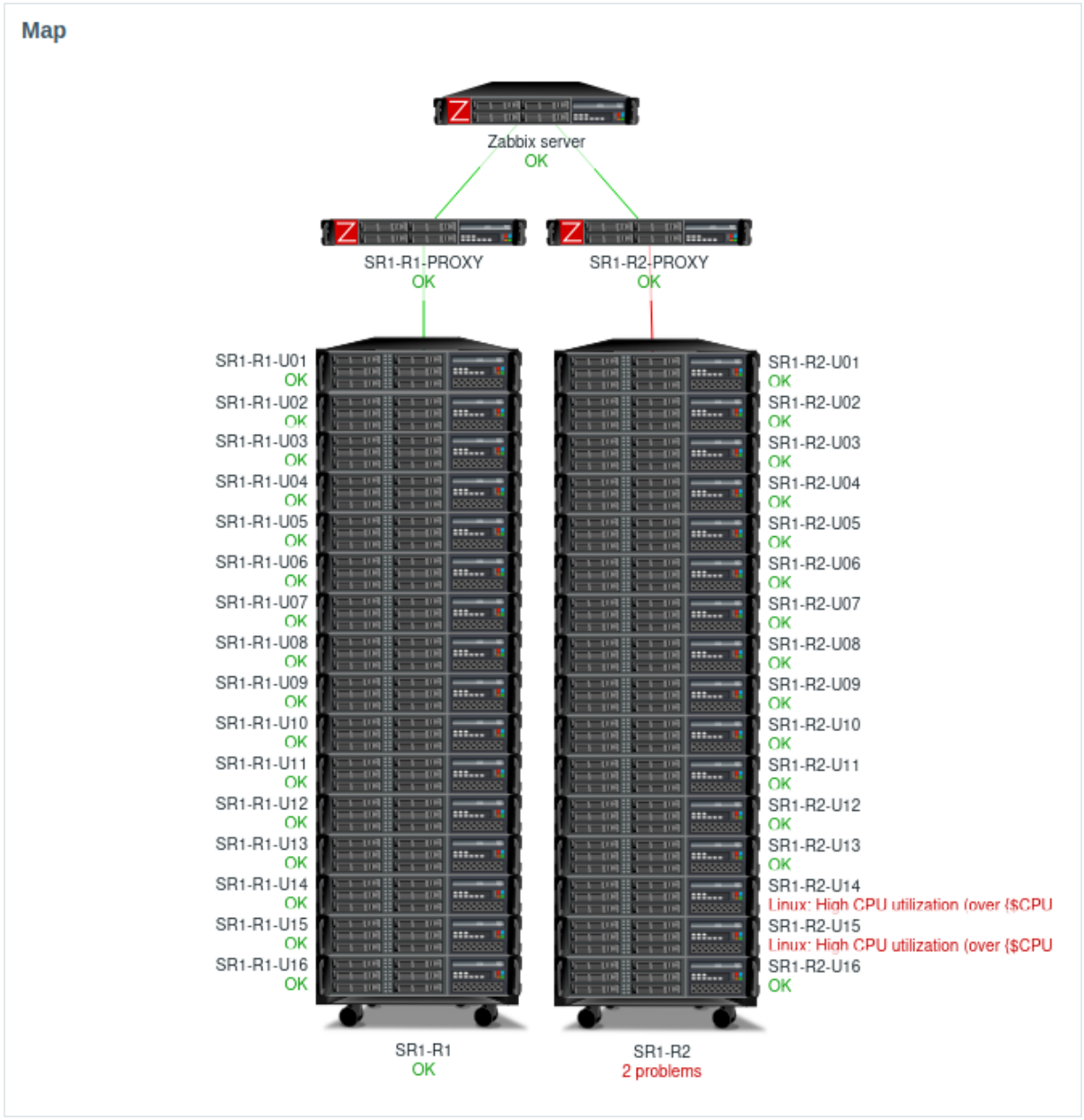


宽度	使用滑块或手动输入 0 到 10 范围内的值来设置图形线条粗细。
颜色	选择线条和填充颜色。
填充	使用滑块或手动输入 0 到 10 范围内的值来设置填充颜色透明度级别。
时间段	指定要包含在迷你图中的值的时间段。为该时间段选择数据源： 仪表板 - 使用仪表板时间段选择器； 部件 - 使用兼容的部件（在 部件参数中设置）； 自定义 - 使用在 从和 到参数中设置的自定义时间段；如果已设置，部件右上角将显示一个时钟图标，鼠标悬停时会指示所设置的时间。 请注意，无论部件的 时间段配置如何，兼容部件仍可将其用作时间段的数据源。 如果 聚合函数设置为“未使用”，则此参数不可用。
部件	输入或选择一个兼容的部件作为时间段的数据源。 如果 时间段设置为“部件”，则此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 如果 时间段设置为“自定义”，则此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 如果 时间段设置为“自定义”，则此参数可用。
历史数据	从历史数据或趋势数据中取值： 自动 - 自动选择； 历史 - 使用历史数据； 趋势 - 使用趋势数据。

地图

概述

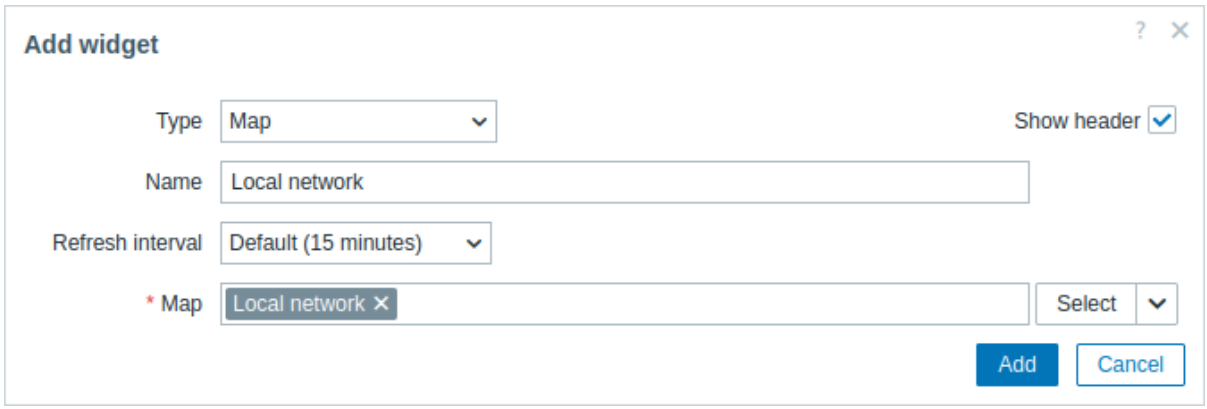
地图小组件可显示网络地图或类似的可视化内容，为您提供网络的动态概览。



您可以将其与地图导航树小组件配合使用，以动态显示在树中选择的地图。

配置

要配置，请选择 拓扑图作为类型：



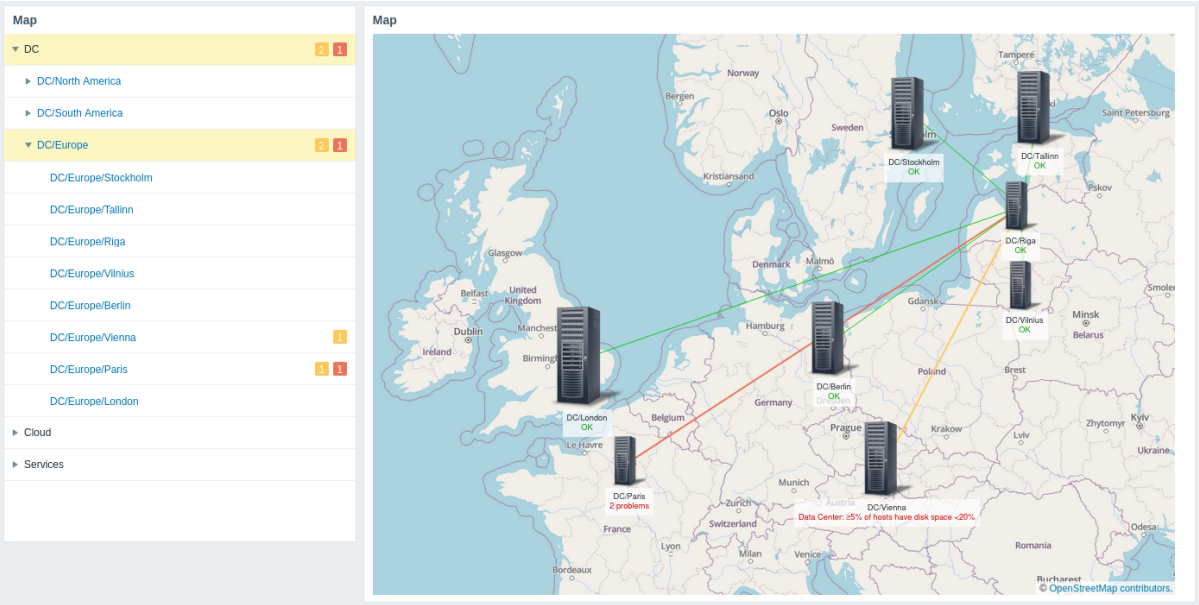
除了所有组件的通用 参数外，您还可以设置以下特定选项：

拓扑图	设置要显示的拓扑图。 或者，选择兼容的组件作为要显示的拓扑图的数据源。 此字段是自动完成的，因此开始输入拓扑图或组件的名称将提供匹配的拓扑图或组件的下拉列表。

地图导航树

概述

地图导航树小组件显示现有地图的层级结构，以及每个地图和地图组的问题数量。

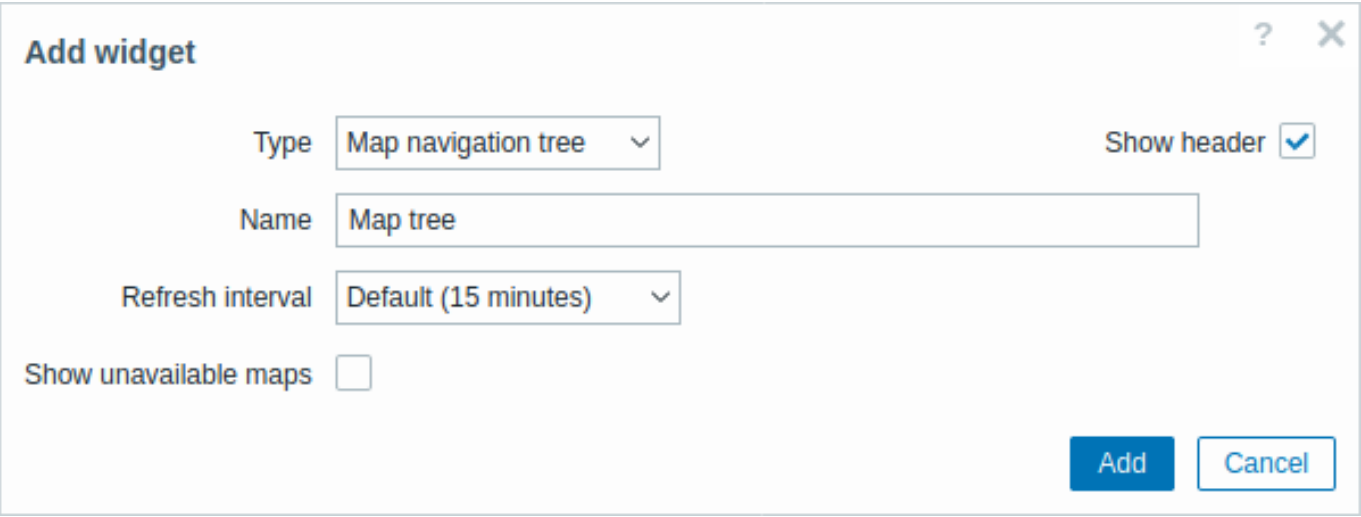


您可以将其与地图 小组件配合使用，以动态显示在树中选定的地图。

层级结构中顶级地图的问题数量包括所有子地图中的问题以及其自身的问题。

配置

要配置，请选择 地图导航树作为类型：

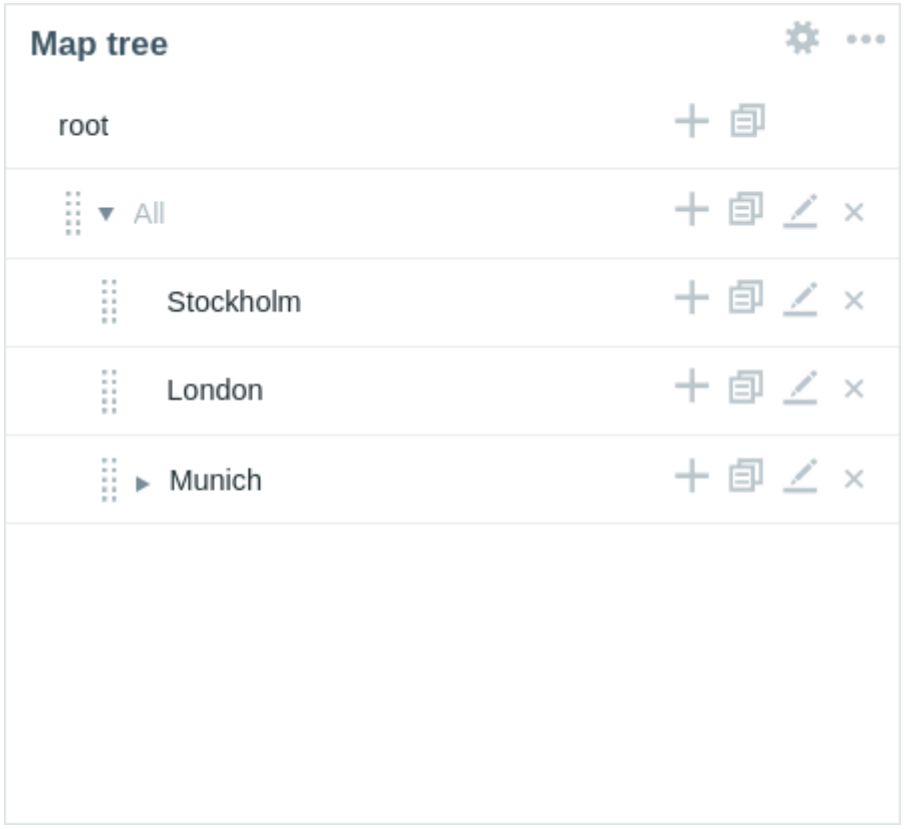


除了所有组件的通用 参数外，您还可以设置以下特定选项：

显示不可用的拓扑图	选中此复选框可显示用户无权读取的拓扑图。 导航树中不可用的拓扑图将以灰色图标显示。 请注意，如果选中此复选框，即使父级拓扑图不可用，也会显示可用的子拓扑图。如果未标记，则根本不会显示不可用父级拓扑图的可用子拓扑图。 问题计数是根据可用拓扑图和可用拓扑图元素计算的。

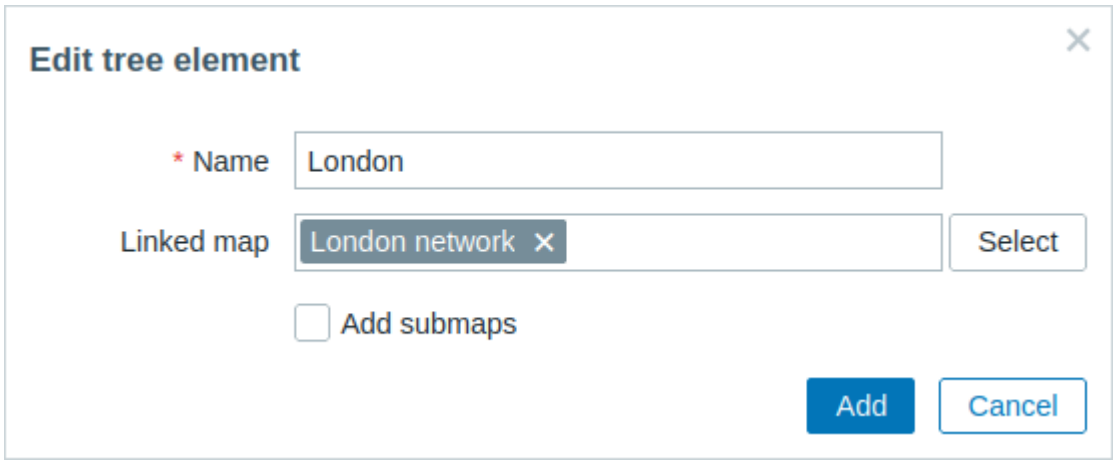
导航树元素显示在列表中。您可以：

- 将元素（包括其子元素）拖到列表中的新位置；
- 展开或折叠元素以显示或隐藏其子元素；
- 向元素添加子元素（带或不带链接拓扑图）；
- 向元素添加多个子元素（带链接拓扑图）；
- 编辑元素；
- 删除元素（包括其子元素）。



元素配置

要配置导航树元素，请添加新元素或编辑现有元素。



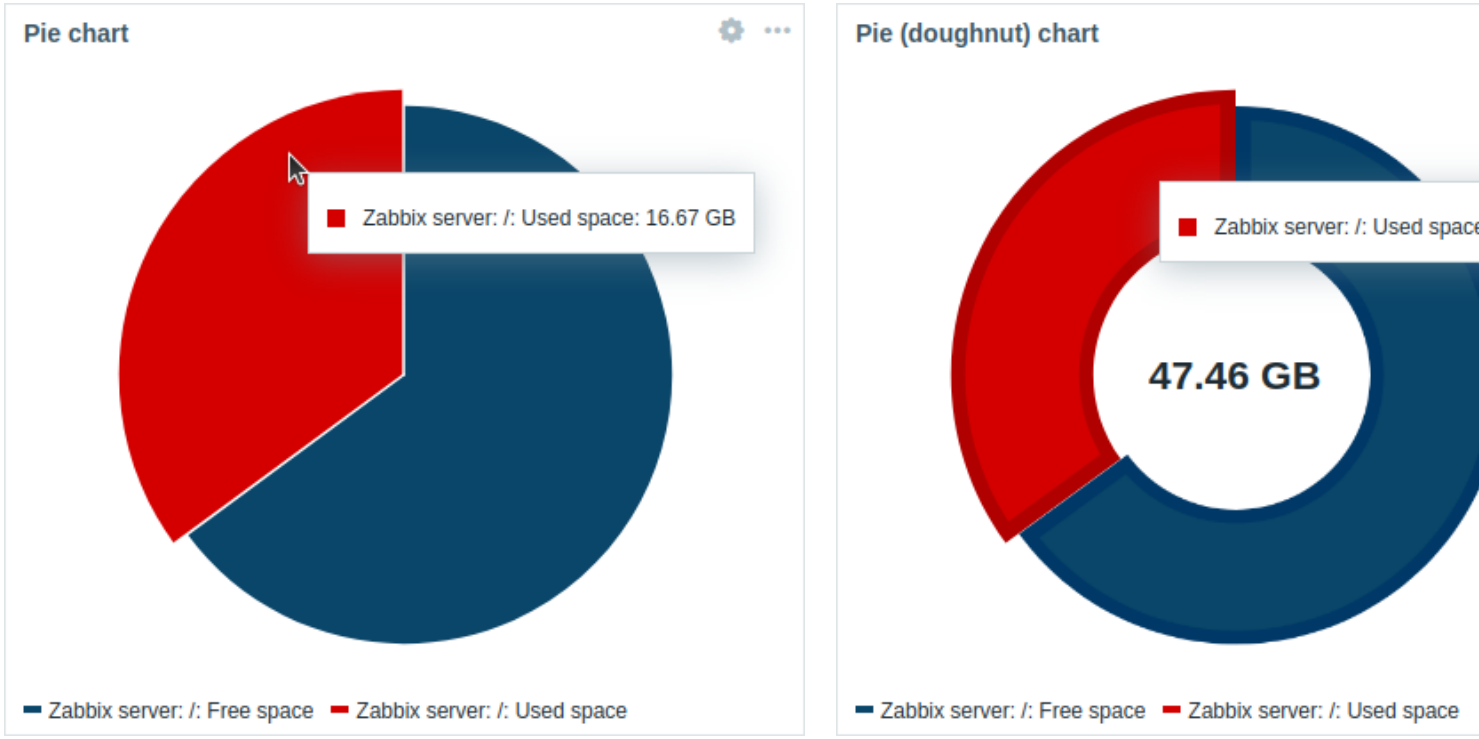
以下导航树元素配置参数可用：

名称	输入导航树元素名称。
链接拓扑图	选择要链接到导航树元素的拓扑图。 此字段是自动完成的，因此开始输入拓扑图名称将提供匹配拓扑图的下拉列表。
添加子拓扑图	选中此复选框可将链接拓扑图的子拓扑图作为子元素添加到导航树元素。

饼图

概述

饼图小组件将监控项的数值数据显示为基于矢量的饼图或环形图。此小组件可以帮助您直观地了解各监控项或主机对整体数据集的贡献。

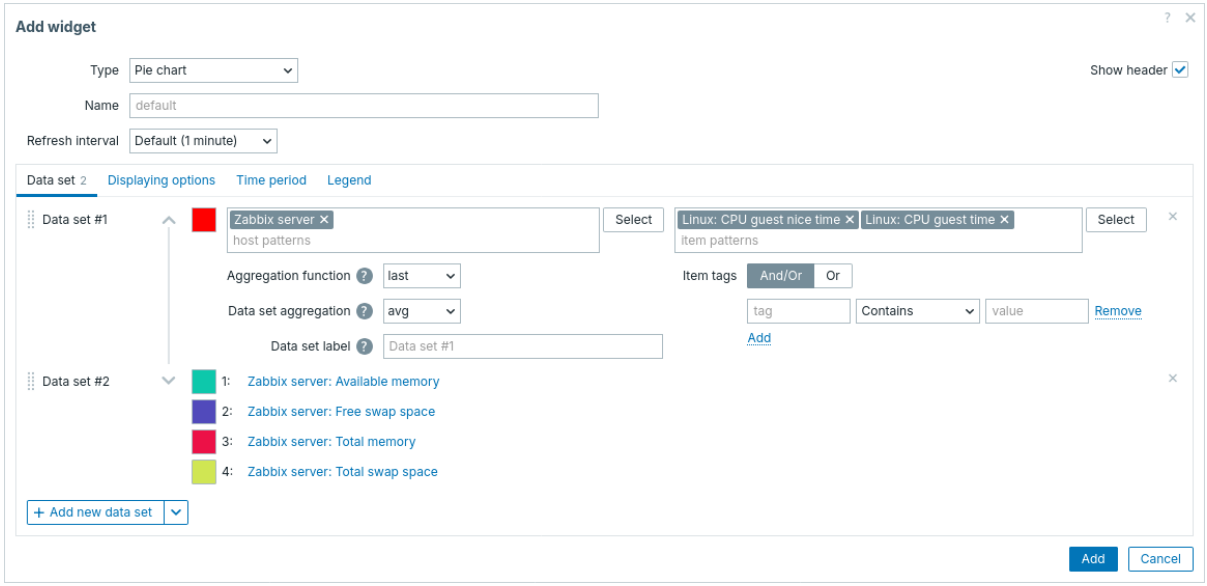


将鼠标悬停在某个扇区上时，该扇区会展开并显示其图例；单击该扇区后，它会保持展开状态，并持续显示图例，直到将其关闭。

在小组件上悬停时会显示小组件的操作菜单 ...，选择其中的下载图像选项，即可将饼图小组件中显示的信息下载为 PNG 图像。根据浏览器设置，图像可能会自动保存到 Downloads 文件夹，或者打开保存对话框，以便您选择保存位置。

配置

要配置，请选择 饼图作为类型：



数据集

数据集选项卡允许通过添加数据集来选择饼图的数据。可以添加两种类型的数据集：

- 监控项模式 - 显示匹配监控项的数据。可以选择单一基础颜色，或选择调色板行，为每个匹配的监控项分配不同的颜色。
- 监控项列表 - 显示所选监控项的数据。可以在颜色选择器中分别选择每个监控项的颜色。

默认情况下会添加一个监控项模式数据集。

数据集		对于监控项模式数据集： 选择或输入主机和监控项模式；饼图中将显示与输入模式匹配的监控项数据；最多可显示 50 个监控项。 选择时可以使用通配符模式（例如，* 将返回匹配零个或多个字符的结果）。 要指定通配符模式，请手动输入字符串并按 *Enter*。 通配符始终会被解释，因此如果存在其他匹配的监控项（例如 *item2*、*item3*），则无法单独添加名为 *item** 的监控项。 对于 “监控项模式” 数据集，必须指定主机和监控项模式。 另请参阅：数据集配置详细信息。 对于监控项列表数据集： 点击添加监控项按钮，为饼图选择监控项。 监控项名称后的类型下拉列表允许选择每个监控项的显示类型： 普通 - 监控项值按比例显示在饼图中（默认）； 总计 - 监控项值占据整个饼图。请注意，每个饼图中只能有一个 “总计” 监控项，并且该监控项将显示在饼图图例的第一位。如果某个监控项设置为 “总计”，则数据集聚合参数（见下文）将被禁用并设置为 “不使用”。 还可以点击添加小组件按钮，选择兼容的小组件作为监控项的数据源。小组件适用的选项与单个监控项相同。 对于 “监控项列表” 数据集，必须指定监控项或小组件。 另请参阅：数据集配置详细信息。 请注意，只允许使用数值型监控项类型。 在模板仪表板上配置小组件时，用于指定主机模式的参数不可用；用于指定监控项列表的参数仅允许选择模板上配置的监控项。
	聚合函数	指定在所选时间段内对数据集中的每个监控项使用的聚合函数： **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值； **sum** - 显示值的总和； **count** - 显示值的数量； **first** - 显示第一个值； **last** - 显示最后一个值（默认）。 聚合允许显示指定时间间隔（5 分钟、1 小时、1 天）的聚合值。 对于 *min*、*max*、*avg* 和 *sum*，只能显示数值数据。对于 *count*，非数值数据将转换为数值。
	数据集聚合	指定对整个数据集使用的聚合函数： 不使用 - 不进行聚合，分别显示监控项（默认）； **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值； **sum** - 显示值的总和； **count** - 显示值的数量。 聚合允许显示在时间段选项卡中选择的、或整个仪表板使用的指定时间间隔（5 分钟、1 小时、1 天）的聚合值。
	数据集标签	为数据集指定自定义标签。 该标签会显示在数据集配置和饼图图例中（对于聚合数据集）。 所有数据集都会进行编号，包括指定了数据集标签的数据集。如果未指定标签，系统将根据数据集编号自动添加标签（例如 “数据集 #2”、“数据集 #3” 等）。重新排序或拖动数据集后，数据集编号会重新计算。 过长的数据集标签将被缩短，以适应其显示位置（例如 “Number of proc...”）。

监控项标签	指定用于筛选小组件中显示的监控项的标签。 可以包含或排除特定标签及标签值。可以设置多个条件。 标签名称匹配始终区分大小写。 每个条件有以下运算符可用： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含标签值中包含所输入字符串的指定标签名称（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除标签值中包含所输入字符串的指定标签名称（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组； **Or** - 满足一个条件即可。

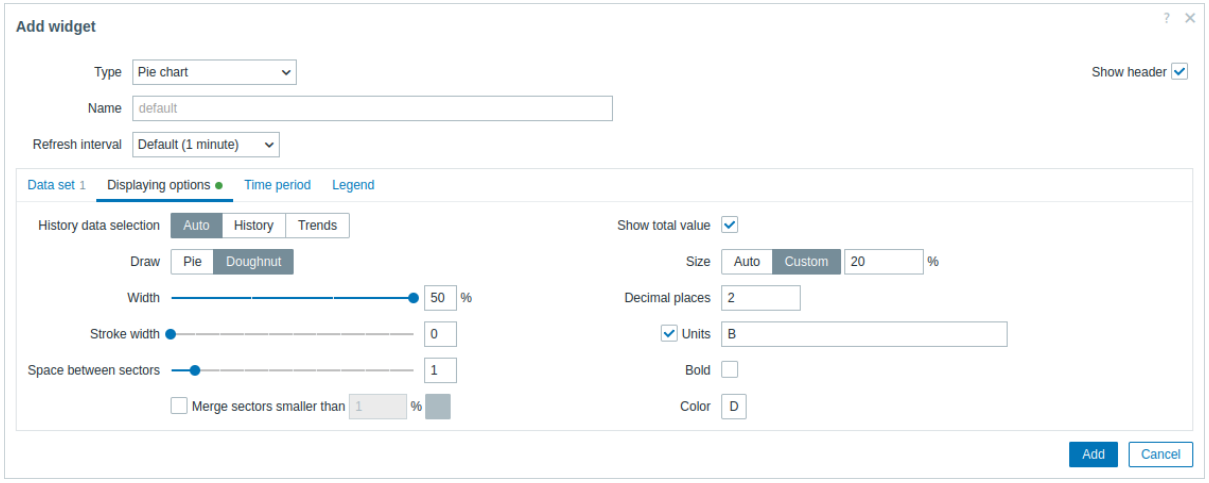
数据集配置详情

现有数据集会显示在列表中。您可以重新排列、展开/折叠、更改颜色以及克隆这些数据集。

更多信息，请参见*Graph* 部件中的数据集配置详情。这些详情也适用于 *Pie chart* 部件。

显示选项

显示选项选项卡允许您定义饼图的历史数据选择和可视化选项：

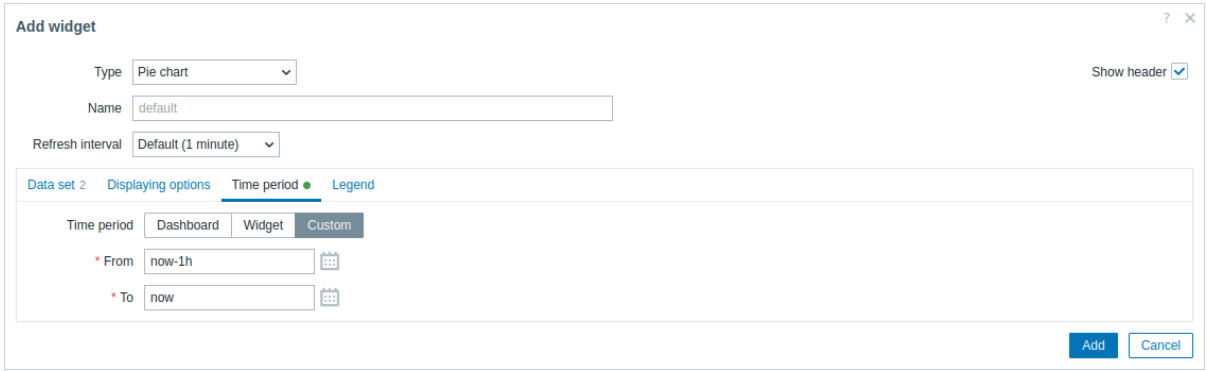


历史数据选择	选择数据源： 自动 - 按照经典算法获取数据（默认）； 历史数据 - 从历史数据中获取数据； 趋势数据 - 从趋势数据中获取数据。
绘制	选择饼图的可视化样式： 饼图 - 完整的饼图（扇区占据半径的 100%）； 圆环图 - 中间带有空心圆的饼图（扇区最多占据半径的 50%）。
扇区之间的间距	选择扇区之间的间距大小（单位为 0-10，默认为 “1”）。

合并小于*N%*的扇区绘制样式：圆环图		选中复选框以合并小于 N% 的扇区。 启用后，选择合并扇区的颜色以及合并小扇区的百分比阈值（N）。
	宽度	选择圆环宽度：半径的 20%、30%、40% 或 50%（默认）。
	描边宽度	选择圆环扇区边框的宽度（0-10）。
	显示总值	选中复选框以在圆环图中间显示总值。
	大小	选择总值的大小选项： 自动 - 自动调整文本大小，使其能够清晰地适应圆环中间区域； 自定义 - 将文本大小指定为小组件总高度的百分比。
	小数位数	指定总值的小数位数（0-6）。
	单位	指定总值的单位。
	粗体	选中复选框以粗体显示总值。
	颜色	选择总值的颜色。

时间段

时间段选项卡允许您为饼图的聚合设置自定义时间段：

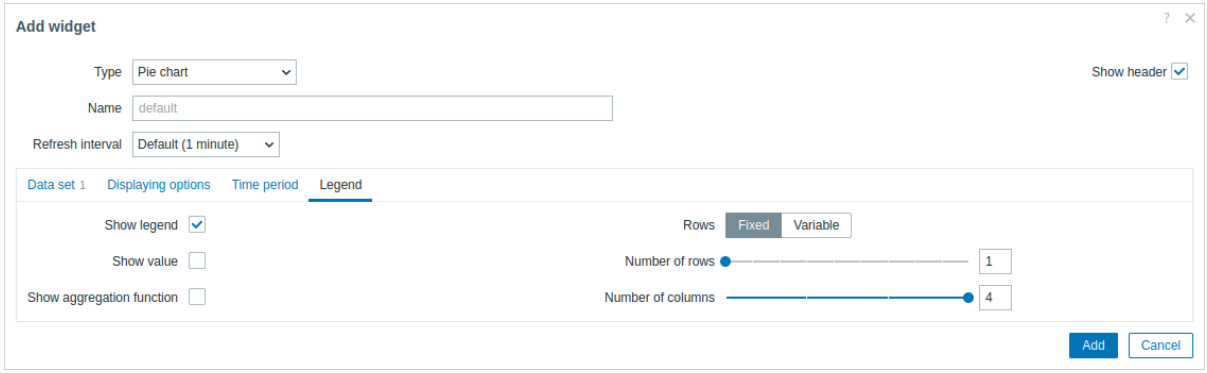


时间段	选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中指定的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。
小组件	输入或选择兼容的小组件（图形、图形（经典）、图形原型）作为时间段的数据源。 当时间段设置为“小组件”时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。

到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。

图例

图例选项卡允许您自定义饼图图例：

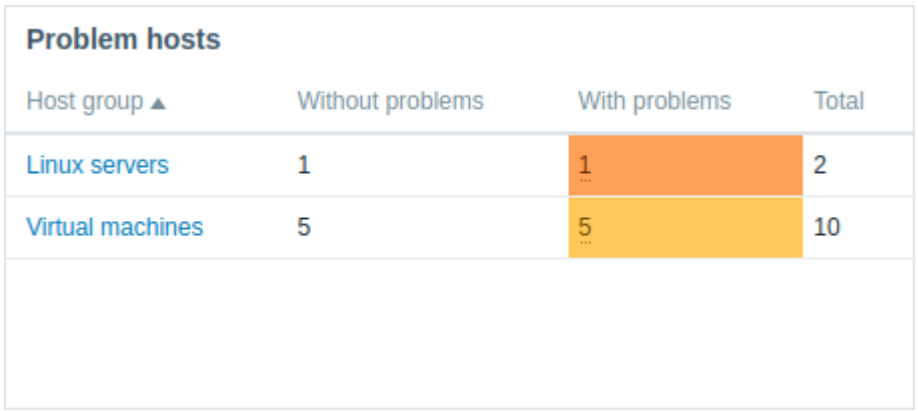


显示图例	取消选中此复选框可隐藏饼图上的图例（默认选中）。
显示值	选中此复选框可在图例中显示监控项的值。
显示聚合函数	选中此复选框可在图例中显示聚合函数。
行	选择图例行的显示模式： 固定 - 显示的行数由行数参数值决定； 可变 - 显示的行数由已配置监控项的数量决定，但不会超过最大行数参数值。
行数/ 最大行数	如果行设置为“固定”，设置要显示的图例行数（1-10）。 如果行设置为“可变”，设置要显示的图例最大行数（1-10）。
列数	设置要显示的图例列数（1-4）。 当未选中显示值时，此参数可用。

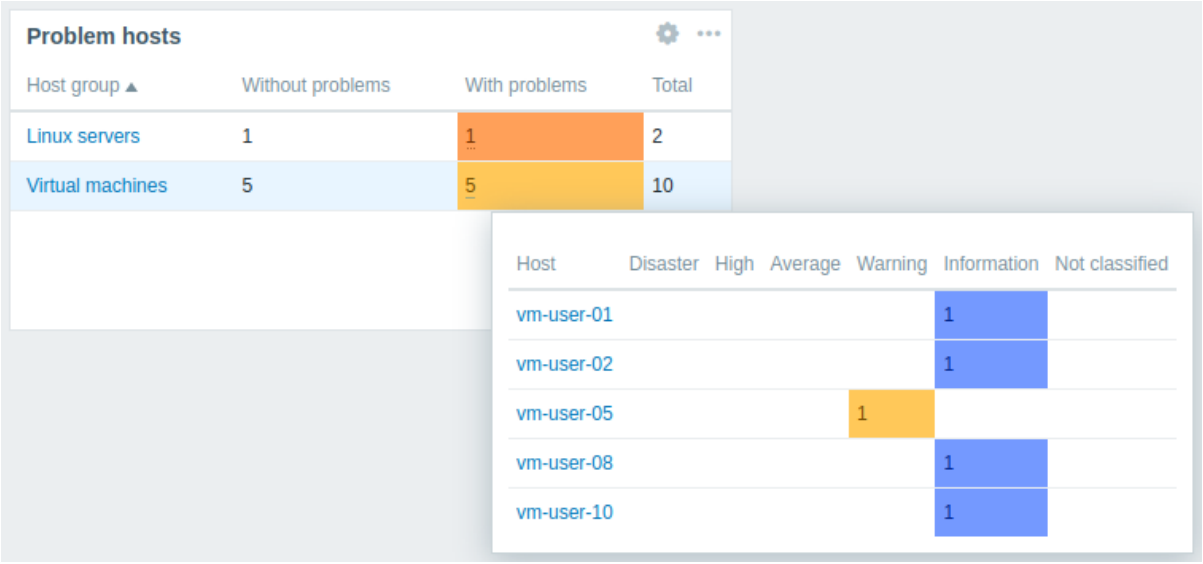
问题主机

概述

Problem hosts 小组件按主机组显示根因问题的数量，并显示每个组中最高的问题严重性。它允许按不同参数筛选问题（主机组、主机、问题名称等）。

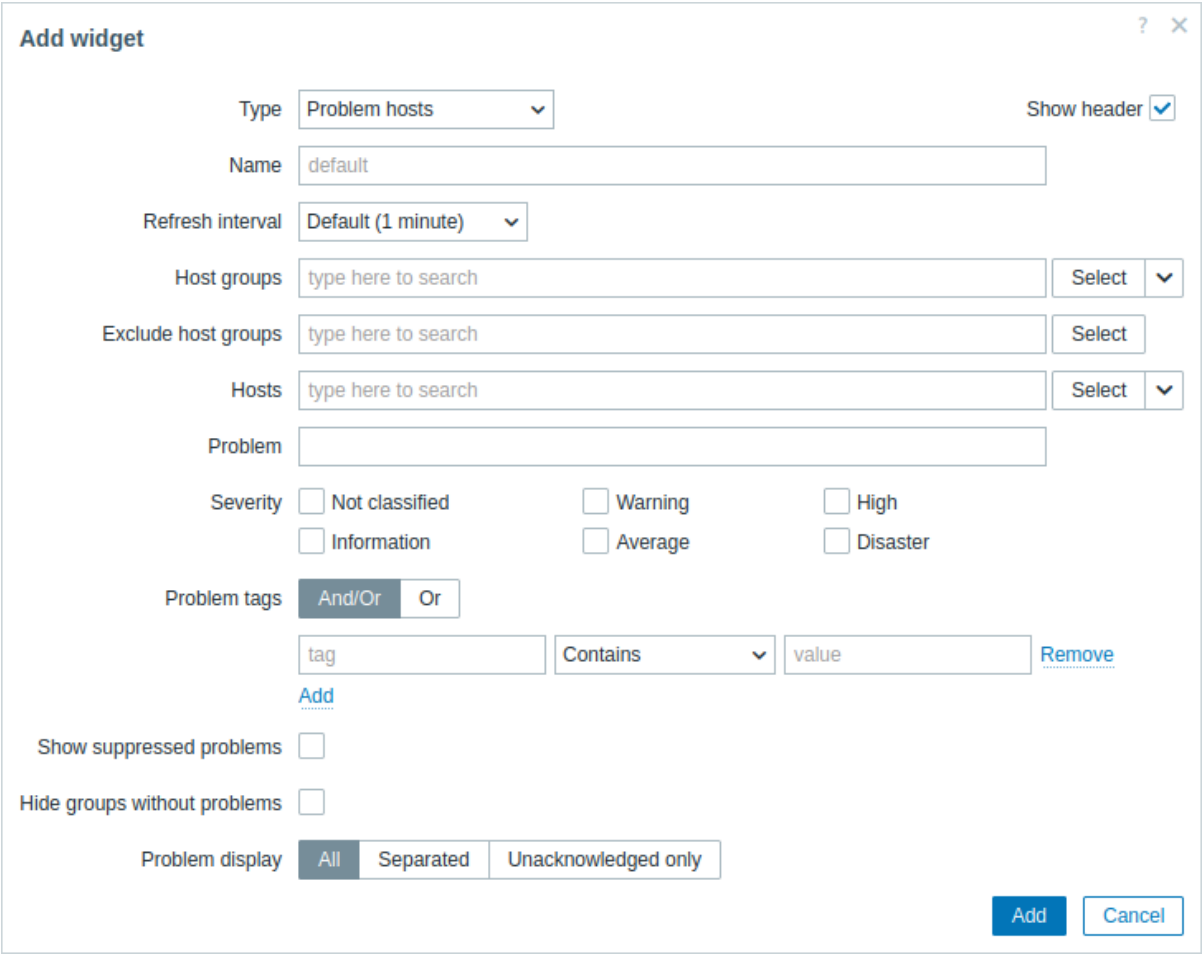


将鼠标悬停在问题计数上方时，会显示存在问题的主机列表：



配置

要配置，请选择 问题主机作为类型：



除了所有组件的通用 参数外，您还可以设置以下特定选项：

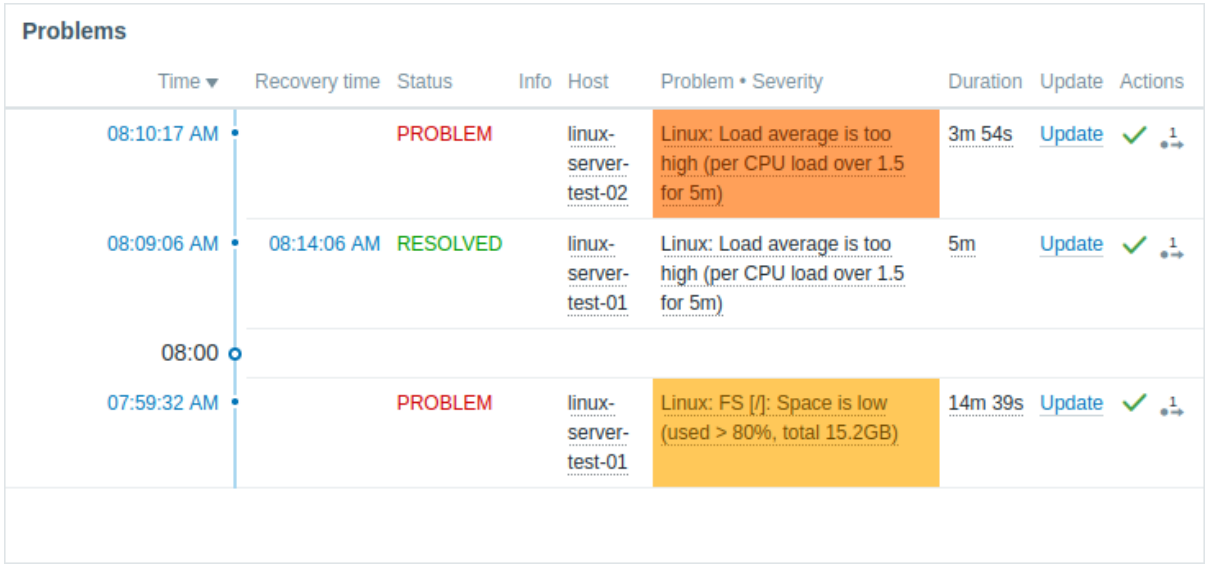
主机组	选择要在组件中显示的主机组。 或者，选择兼容的组件作为主机组的数据源。 此字段是自动完成的，因此开始输入组的名称将提供匹配组的下拉列表。 指定父主机组会隐式选择所有嵌套主机组。 这些主机组中的主机数据将显示在组件中；如果未输入任何主机组，则将显示所有主机组。 在模板仪表板上配置组件时，此参数不可用。

排除主机组	选择要从组件中隐藏的主机组。 此字段是自动完成的，因此开始输入组的名称将提供匹配组的下拉列表。 指定父主机组会隐式选择所有嵌套主机组。 这些主机组中的主机数据将不会显示在组件中。例如，主机 001、002、003 可能在 A 组中，而主机 002、003 也可能在 B 组中。如果我们同时选择显示 A 组和排除 B 组，则仪表板中只会显示来自主机 001 的数据。 在模板仪表板 上配置组件时，此参数不可用。
主机	选择要在组件中显示的主机。 或者，选择兼容的组件或仪表板作为主机的数据源。 此字段是自动完成的，因此开始输入主机名称将提供匹配主机的下拉列表。 如果未输入任何主机，则将显示所有主机。 在模板仪表板上配置组件时，此参数不可用。
问题	您可以通过问题名称来限制显示的问题主机数量。 如果您在此处输入字符串，则只会显示名称中包含输入字符串的有问题的主机。 宏不会展开。
严重性	标记问题严重性以过滤要在窗口组件中显示的问题。 如果未标记任何严重性，则将显示所有问题。
问题标签	指定问题标签以限制在窗口组件中显示的问题数量。 可以包含或排除特定标签和标签值。可以设置多个条件。标签名称匹配始终区分大小写。 每个条件都有多个运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将按或条件分组； 或 - 只要满足一个条件就足够了。
显示被抑制的问题	勾选复选框以显示由于主机维护而被抑制（不显示）的问题。
隐藏没有问题的组	勾选隐藏没有问题的组选项以在窗口组件中隐藏没有问题的主机组的数据。 在模板仪表板上配置窗口组件时，此参数不可用。
问题显示	将问题计数显示为： 全部 - 将显示完整的问题计数； 分隔 - 未确认的问题计数将以总问题计数的数字分隔显示； 仅未确认 - 仅显示未确认的问题计数。

问题

概述

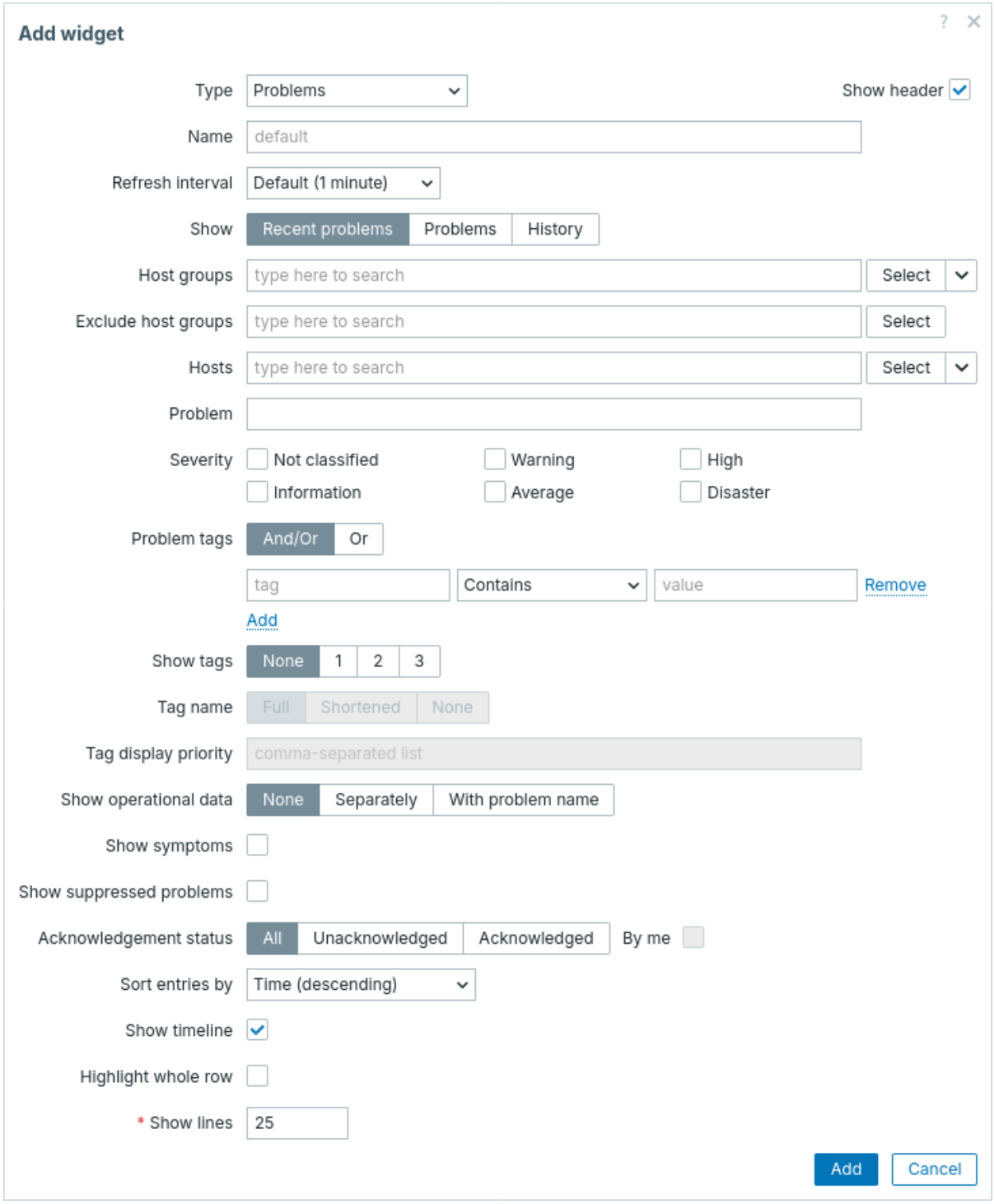
问题小组件会显示按不同参数（主机组、主机、问题名称等）过滤后的当前问题，让您清楚了解哪些内容需要关注。



它显示与 监控 > 问题 相同的数据，最多可显示 1000 条记录。

配置

要进行配置，请将类型选择为 *Problems*：



您可以通过多种方式限制小组件中显示的问题数量 - 按问题状态、问题名称、严重性、主机组、主机、事件标签、确认状态等。

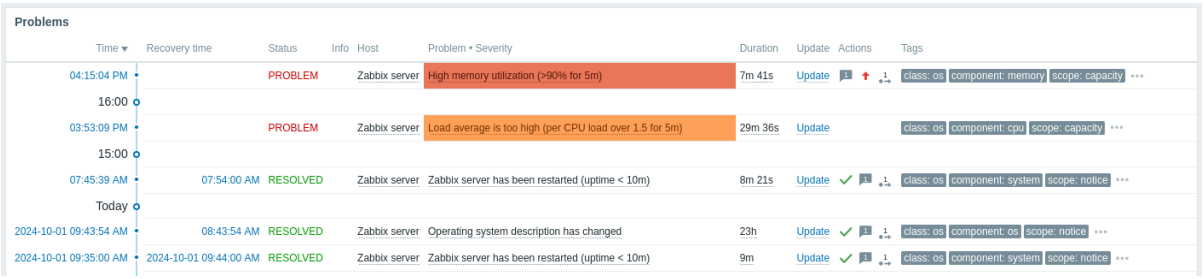
除了所有小组件通用的参数之外，您还可以设置以下特定选项：

Show	按问题状态筛选： **Recent problems** - 显示未解决且最近已解决的问题（默认）； **Problems** - 显示未解决的问题； **History** - 显示所有事件的历史记录。
Host groups	选择要在小组件中显示其问题的主机组。 或者，选择一个兼容的小组件作为主机组的数据源。 此字段支持自动补全，因此开始输入组名时会显示匹配组的下拉列表。 指定父主机组会隐式选择其所有嵌套主机组。 这些主机组中的问题将显示在小组件中；如果未输入任何主机组，则会显示所有主机组中的问题。 在模板仪表板上配置小组件时，此参数不可用。

Exclude host groups	选择要在小组件中隐藏其问题的主机组。 此字段支持自动补全，因此开始输入组名时会显示匹配组的下拉列表。 指定父主机组会隐式选择其所有嵌套主机组。 这些主机组中的问题将不会显示。例如，主机 001、002、003 可能位于 Group A 中，而主机 002、003 也位于 Group B 中。如果我们同时选择 *show* Group A 和 *exclude* Group B，则小组件中只会显示主机 001 的问题。 在模板仪表板上配置小组件时，此参数不可用。
Hosts	选择要在小组件中显示其问题的主机。 或者，选择一个兼容的小组件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名时会显示匹配主机的下拉列表。 如果未输入任何主机，则会显示所有主机的问题。 在模板仪表板上配置小组件时，此参数不可用。
Problem	您可以按问题名称限制显示的问题数量。 如果在此处输入字符串，则只会显示名称中包含所输入字符串的问题。 宏不会展开。
Severity	勾选问题严重性，以筛选小组件中要显示的问题。 如果未勾选任何严重性，则会显示所有问题。
Problem tags	指定问题标签，以限制小组件中显示的问题数量。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含标签值中含有所输入字符串的指定标签名称（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除标签值中含有所输入字符串的指定标签名称（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 *Or* 条件分组； **Or** - 满足任一条件即可。 筛选后，此处指定的标签将优先与问题一起显示，除非被下面的 *Tag display priority* 列表覆盖。
Show tags	选择显示的标签数量： **None** - 不显示 *Tags* 列； **1** - *Tags* 列包含一个标签； **2** - *Tags* 列包含两个标签； **3** - *Tags* 列包含三个标签。 要查看问题的所有标签，请将鼠标悬停在三个点图标上。
Tag name	选择标签名称的显示模式： **Full** - 完整显示标签名称和值； **Shortened** - 标签名称缩短为 3 个字符，但标签值完整显示； **None** - 仅显示标签值，不显示名称。
Tag display priority	以逗号分隔的标签列表形式输入问题的标签显示优先级。 只能使用标签名称，不要使用值。 示例：`Services,Applications,Application` 此列表中的标签将始终优先显示，覆盖按字母顺序的自然排序。
Show operational data	选择显示运行数据的模式： **None** - 不显示运行数据； **Separately** - 在单独的列中显示运行数据； **With problem name** - 将运行数据附加到问题名称后，并使用括号括起运行数据。 有关详细信息，请参见触发器配置中 *Operational data* 字段的说明。
Show symptoms	勾选此复选框，以在单独一行中显示被归类为症状的问题。
Show suppressed problems	勾选此复选框，以显示原本会因主机维护或单个问题抑制而被抑制（不显示）的问题。
Acknowledgement status	筛选显示所有问题、仅未确认的问题，或仅已确认的问题。勾选附加复选框可筛选掉那些曾经被您确认过的问题。

Sort entries by	按以下方式排序条目： **Time**（降序或升序）； **Severity**（降序或升序）； **Problem name**（降序或升序）； **Host**（降序或升序）。 在模板仪表板上配置小组件时，无法按 **Host**（降序或升序）排序条目。
Show timeline	勾选此复选框以显示可视化时间线。
Highlight whole row	勾选此复选框以高亮显示未解决问题的整行。高亮使用问题严重性颜色。 高对比度主题不支持 *Highlight whole row*。
Show lines	指定要显示的问题行数。

使用小组件

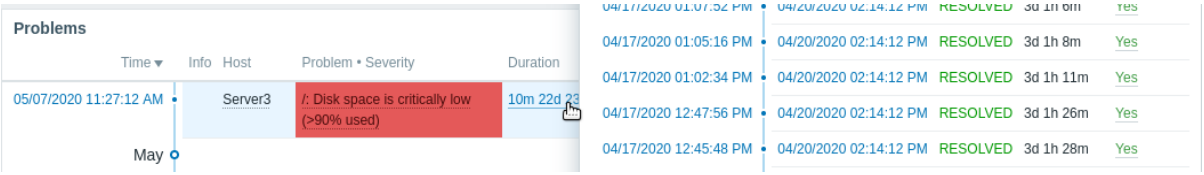


Problems 小组件可快速访问更多信息：

- 单击问题日期（在 Time 列中）或恢复日期（在 Recovery Time 列中）以查看事件详情。
- 如果 Info 字段不为空，将鼠标悬停在显示的图标上以查看更多详情。
- 单击主机名称以打开主机菜单。
- 单击问题名称以打开事件菜单。
- 将鼠标悬停在问题持续时间上，或单击它，以查看问题事件弹出窗口。
- 单击 *Update* 会打开更新问题 窗口。
- 将鼠标悬停在 Actions 列中的灰色箭头图标上，或单击该图标，以查看已执行操作列表。

问题事件弹出窗口

问题事件弹出窗口包含此触发器的问题事件列表，以及（如果已定义）触发器描述和一个可点击的 URL。



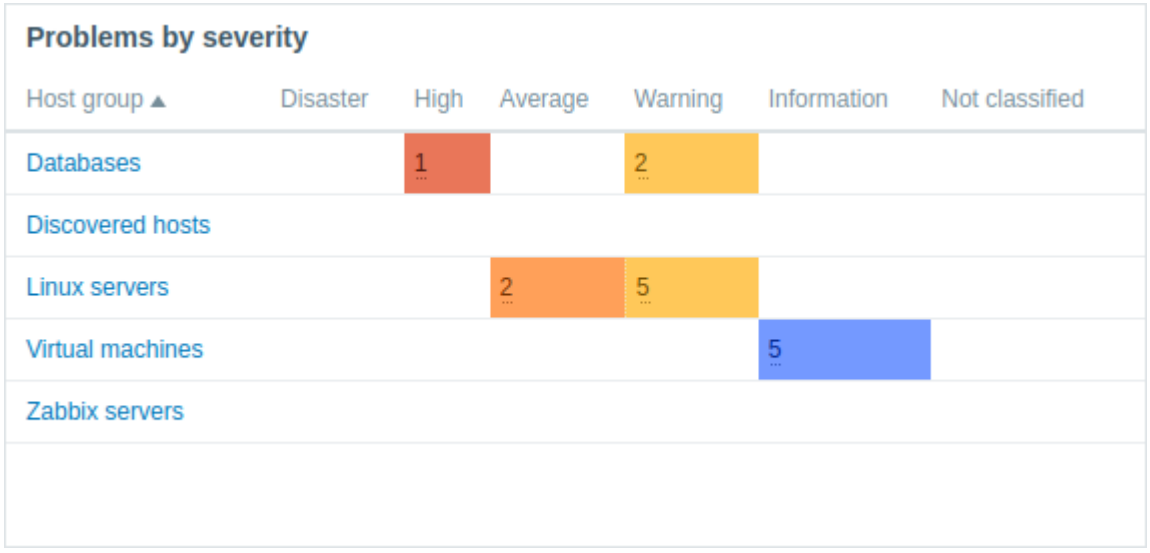
要打开问题事件弹出窗口：

- 将鼠标悬停在 *Problems* 小组件的 *Duration* 列中的问题持续时间上。当鼠标移开持续时间后，弹出窗口会消失。
- 单击 *Problems* 小组件的 *Duration* 列中的持续时间。只有再次单击持续时间时，弹出窗口才会消失。

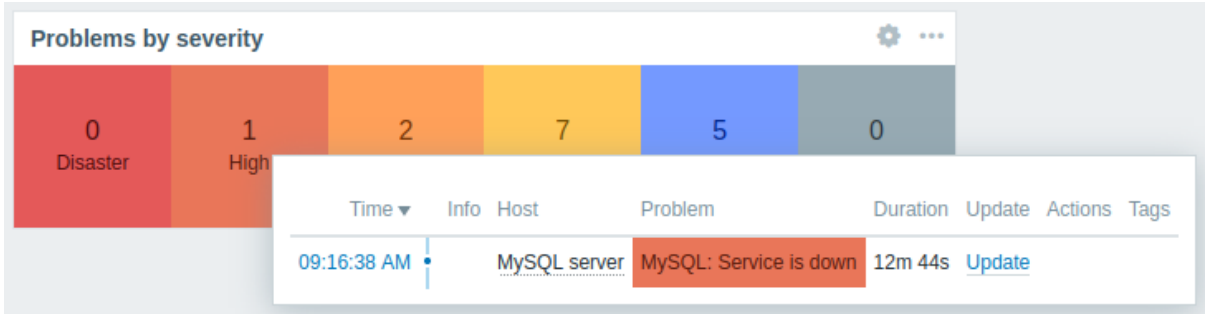
按严重性划分的问题

概述

按严重性显示问题小组件按严重性分组显示cause problems的数量。它允许按不同参数筛选问题（主机群组、主机、问题名称等）。

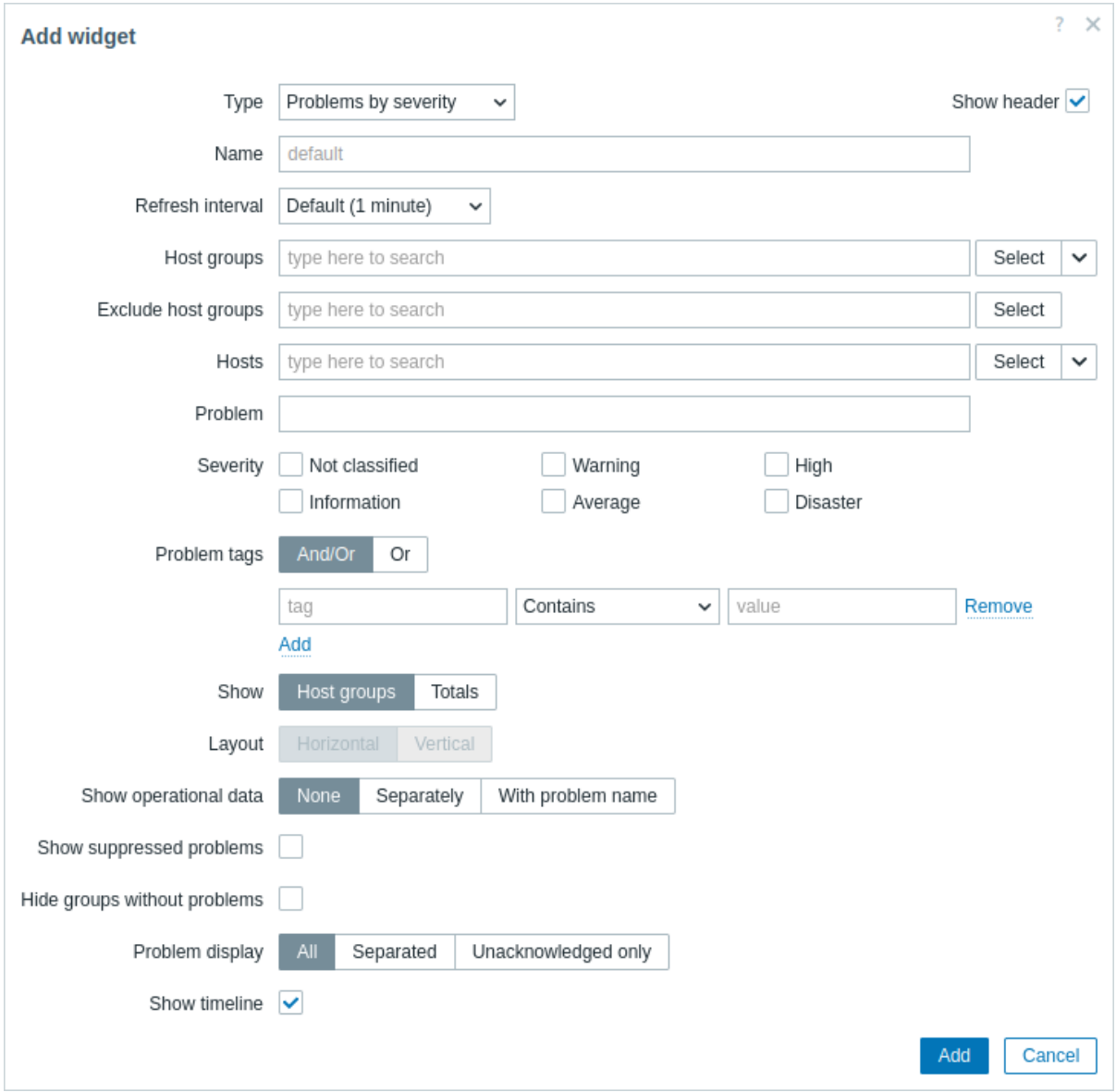


将鼠标悬停在问题数量上方时，会显示相关问题列表：



配置

要进行配置，请将类型选择为 *Problems by severity*：



除了所有小部件通用的参数外，您还可以设置以下特定选项：

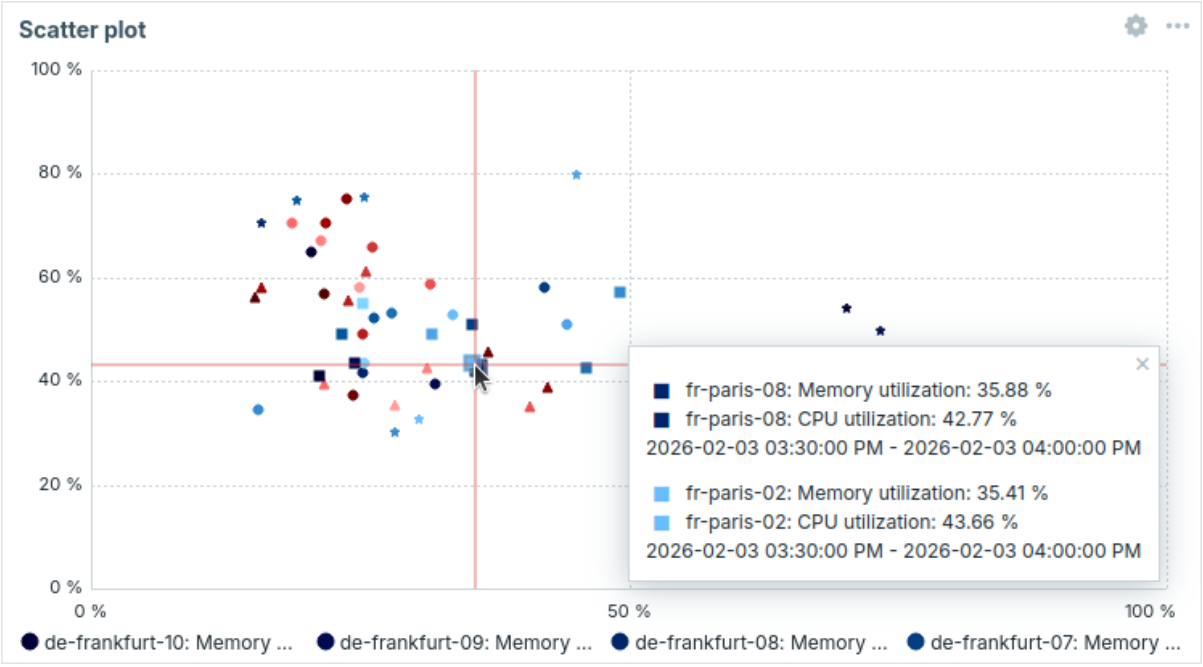
Host groups	选择要在小部件中显示的主机组。 或者，选择一个兼容的小部件作为主机组的数据源。 此字段支持自动补全，因此在开始输入组名时，会显示匹配组的下拉列表。 指定父主机组会隐式选择所有嵌套主机组。 这些主机组中的主机数据将显示在小部件中；如果未输入任何主机组，则将显示所有主机组。 在模板仪表板 上配置小部件时，此参数不可用。
Exclude host groups	选择要在小部件中隐藏的主机组。 此字段支持自动补全，因此在开始输入组名时，会显示匹配组的下拉列表。 指定父主机组会隐式选择所有嵌套主机组。 这些主机组中的主机数据将不会显示。例如，主机 001、002、003 可能都在 Group A 中，而主机 002、003 也在 Group B 中。如果我们同时选择 *show* Group A 并 *exclude* Group B，则仪表板中只会显示主机 001 的数据。 在模板仪表板 上配置小部件时，此参数不可用。
Hosts	选择要在小部件中显示的主机。 或者，选择一个兼容的小部件或仪表板作为主机的数据源。 此字段支持自动补全，因此在开始输入主机名时，会显示匹配主机的下拉列表。 如果未输入任何主机，则将显示所有主机。 在模板仪表板 上配置小部件时，此参数不可用。
Problem	您可以按问题名称限制显示的问题主机数量。 如果在此处输入字符串，则只会显示那些问题名称中包含所输入字符串的主机。 宏不会展开。
Severity	勾选问题严重性，以筛选要在小部件中显示的问题。 如果未勾选任何严重性，将显示所有问题。

Problem tags	指定问题标签，以限制在小部件中显示的问题数量。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含标签值中含有所输入字符串的指定标签名称（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除标签值中含有所输入字符串的指定标签名称（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 *Or* 条件分组； **Or** - 满足任一条件即可。
Show	选择显示选项： **Host groups** - 按主机组显示问题； **Totals** - 以与问题严重性对应的彩色块显示所有所选主机组的问题总数。 在模板仪表板 上配置小部件时，此参数不可用，并且只会显示问题总数。
Layout	选择布局选项： **Horizontal** - 总数的彩色块将水平显示； **Vertical** - 总数的彩色块将垂直显示。 当 *Show* 设置为”Totals” 时，此参数可用。
Show operational data	将运行数据显示为： **None** - 不显示运行数据 **Separately** - 运行数据将作为单独一行显示 **With problem name** - 运行数据将与问题名称一起显示。 有关详细信息，请参见触发器配置 中 *Operational data* 字段的说明。
Show suppressed problems	勾选此复选框以显示因主机维护而本应被抑制（不显示）的问题。
Hide groups without problems	勾选 *Hide groups without problems* 选项，以在小部件中隐藏没有问题的主机组数据。 在模板仪表板 上配置小部件时，此参数不可用。
Problem display	将问题计数显示为： **All** - 显示完整的问题计数； **Separated** - 未确认的问题计数将与总问题计数分开显示； **Unacknowledged only** - 仅显示未确认的问题计数。
Show timeline	勾选此复选框以显示可视化时间线。

散点图

概述

散点图小组件通过沿 X 轴和 Y 轴绘制单个数据点，显示两个指标之间的关系。这有助于发现数据集中的模式、聚类、相关性和异常值。



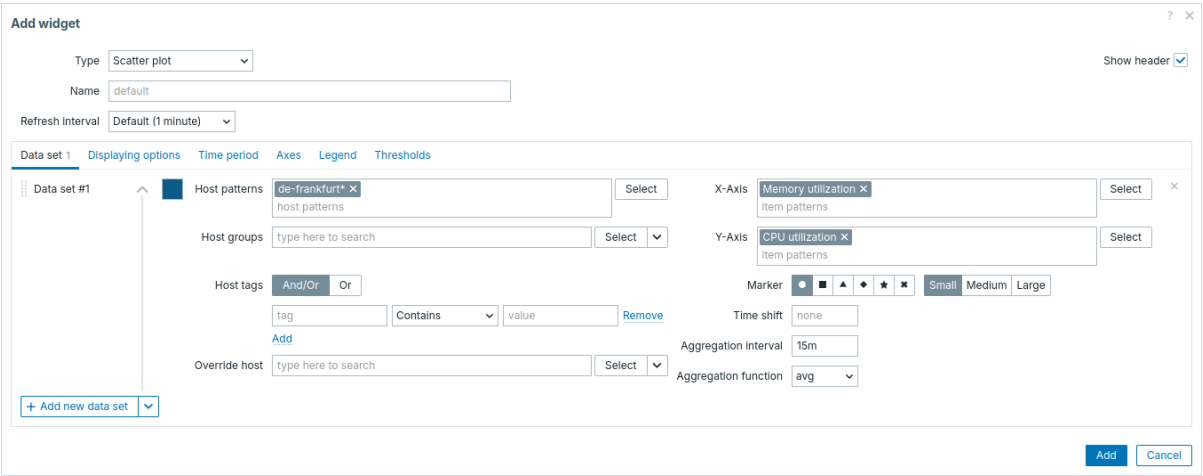
将鼠标悬停在数据点上时，会显示一个工具提示，其中列出来自该数据点以及 6px 半径范围内其他数据点的监控项；这些监控项首先按散点图 X 轴值升序排列，然后按 Y 轴值降序排列。

在工具提示中选择监控项后，其数据会广播到其他小组件。

在小组件上悬停时，选择小组件操作菜单 ... 中的下载图像选项，可以将散点图小组件中显示的信息下载为 PNG 图像。根据浏览器设置，图像可能会自动保存到 `Downloads` 文件夹，或者打开保存对话框，以便选择保存位置。

配置

要进行配置，请选择 *Scatter plot* 作为类型：



数据集

数据集选项卡允许通过添加数据集来选择散点图的数据。可以添加两种类型的数据集：

- 监控项模式 - 将匹配的监控项所在的主机显示为数据点。您可以选择单一基础颜色，或选择调色板行，为每个包含匹配监控项的主机分配不同的颜色。
- 监控项列表 - 将选定的监控项显示为数据点，其中每个点代表一对监控项（X 和 Y）。您可以在颜色选择器中分别选择每个数据集的颜色。

仅当主机/监控项对在两个坐标轴上都至少有一个值时，才会创建数据点。

默认情况下，会添加一个监控项模式数据集。

数据集		对于监控项模式数据集： 选择或输入主机和监控项模式；匹配这些模式的监控项数据将显示在散点图中。请注意，如果数据点数量超过 8k，应在 `php.ini` 配置文件中增大 `memory_limit`。对于 20k 个数据点，建议将内存限制设置为 512 MB。 选择时可以使用通配符模式（例如，* 将返回匹配零个或多个字符的结果）。 要指定通配符模式，请手动输入字符串，然后按 *Enter*。 通配符始终会被解释，因此如果存在其他匹配的监控项（例如 *item2*、*item3*），则无法单独添加名为 *item** 的监控项。 对于 “监控项模式” 数据集，必须指定主机和监控项模式。 另请参阅：数据集配置详细信息。 对于监控项列表数据集： 点击添加监控项按钮，为散点图选择监控项。 您还可以点击添加小组件按钮，选择兼容的小组件作为监控项的数据源。 对于 “监控项列表” 数据集，必须指定监控项或小组件。 另请参阅：数据集配置详细信息。 请注意，仅允许使用数值型监控项类型。 在模板仪表板上配置小组件时，用于指定主机模式的参数不可用；用于指定监控项列表的参数仅允许选择在模板上配置的监控项。
	主机群组	选择主机群组。 或者，选择兼容的小组件作为主机群组的数据源。 此字段支持自动补全，因此开始输入群组名称时，将显示包含匹配群组的下拉列表。 指定父主机群组会隐式选择所有嵌套的主机群组。 这些主机群组中的主机数据将显示在小组件中；如果未输入主机群组，则将显示所有主机群组。 此参数仅适用于监控项模式数据集，在模板仪表板上配置小组件时不可用。
	主机标签	指定标签，以筛选小组件中显示的主机。 可以包含或排除特定标签及标签值。可以设置多个条件。 标签名称匹配始终区分大小写。 每个条件有以下运算符可用： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件；具有相同标签名称的条件将按 Or 条件分组； **Or** - 满足一个条件即可。 此参数仅适用于监控项模式数据集，在模板仪表板上配置小组件时不可用。
	覆盖主机	选择兼容的小组件或仪表板的主机选择器，作为主机的数据源。 在模板仪表板上配置小组件时，此参数不可用。
	标记	选择数据集中主机数据点的标记类型和大小。
	时间偏移	如有需要，指定时间偏移。 此字段中可以使用时间后缀。允许使用负值。
	聚合函数	指定在选定的聚合间隔内对数据集中每个监控项使用的聚合函数： **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值； **sum** - 显示值的总和； **count** - 显示值的数量（如果两个坐标轴上都至少存在一个值）； **first** - 显示第一个值； **last** - 显示最后一个值。 聚合允许在指定的时间间隔（5 分钟、1 小时、1 天）内显示聚合值，而不是显示所有值。 *min*、*max*、*avg* 和 *sum* 只能显示数值数据。对于 *count*，非数值数据将转换为数值。

聚合间隔	指定聚合值的时间间隔。 此字段中可以使用时间后缀。不带后缀的数值将被视为秒。 请注意，如果小组件配置为基于趋势显示历史数据（历史数据选择设置为趋势或自动），建议使用 1 小时的倍数作为聚合间隔（例如 3600、60m、1h、3h 等）。趋势存储按小时聚合的值，因此使用非 1 小时倍数的聚合间隔（例如 100s、7min、15min、90min 等）可能导致难以解释的结果。 必须指定聚合间隔。 聚合间隔决定数据点的时间间隔和坐标；例如，如果 X 轴监控项有 4 个值，而 Y 轴监控项有 1 个值，则坐标反映了在指定间隔内对 X 轴的 4 个值和 Y 轴的 1 个值进行聚合。

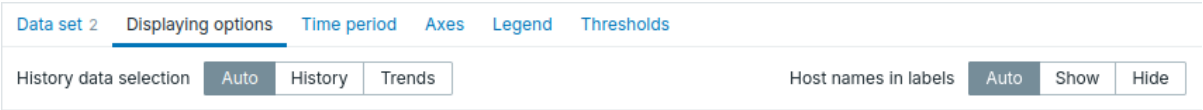
数据集配置详情

现有数据集会显示在列表中。您可以重新排列、展开/折叠、更改颜色以及克隆这些数据集。

更多信息请参见*Graph* 部件中的数据集配置详情。这些详情同样适用于 *Scatter plot* 部件。

显示选项

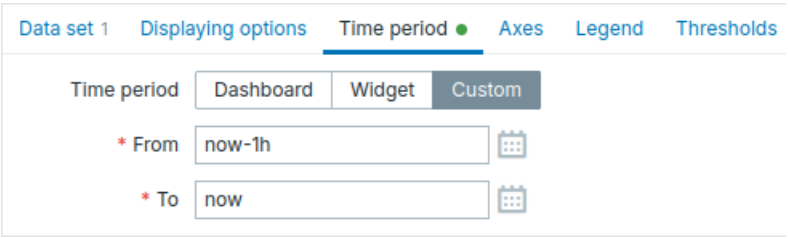
显示选项选项卡允许您定义历史数据选择方式：



历史数据选择	设置散点图数据的来源： 自动 - 根据经典图形算法获取数据（默认）。 历史数据 - 从历史数据中获取数据。 趋势数据 - 从趋势数据中获取数据。
标签中的主机名称	选择是否在图形图例以及将鼠标悬停在图形数据上时显示的工具提示中显示主机名称： 自动 - 仅当所有数据集中存在多个主机时显示主机名称（默认）。 显示 - 显示主机名称。 隐藏 - 隐藏主机名称。 在模板仪表板上配置小组件时，此参数不可用。

时间段

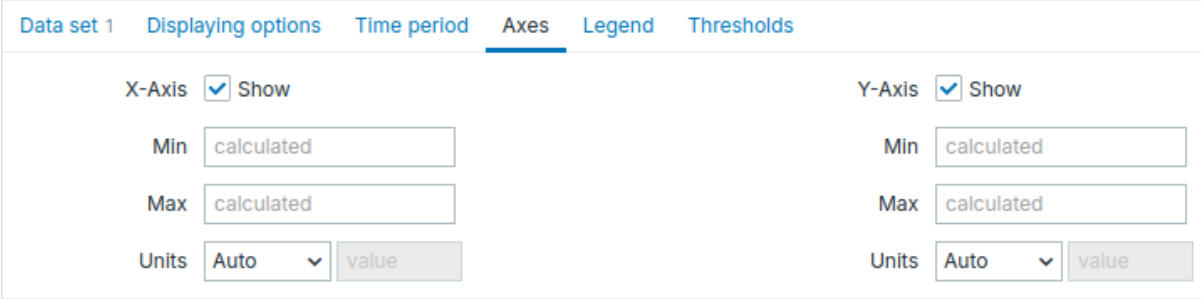
时间段选项卡允许您设置散点图中显示数据的时间段：



时间段	选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。
小组件	输入或选择兼容的小组件作为时间段的数据源。 当时间段设置为“小组件”时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。

坐标轴

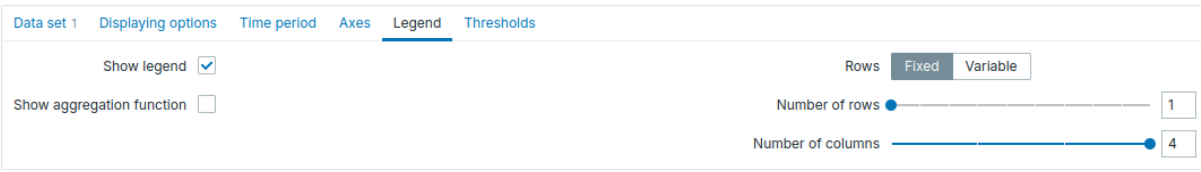
坐标轴选项卡用于自定义坐标轴的显示方式：



X 轴/*Y* 轴	取消选中此复选框可隐藏 X 轴/Y 轴（默认选中）。
最小值	设置相应坐标轴的最小值。
最大值	设置相应坐标轴的最大值。
单位	从下拉列表中选择散点图坐标轴值的单位： 自动 - 坐标轴值使用数据集中第一个监控项的单位显示； 静态 - 坐标轴值使用值输入字段中指定的单位显示；如果该字段留空，则仅显示数值。

图例

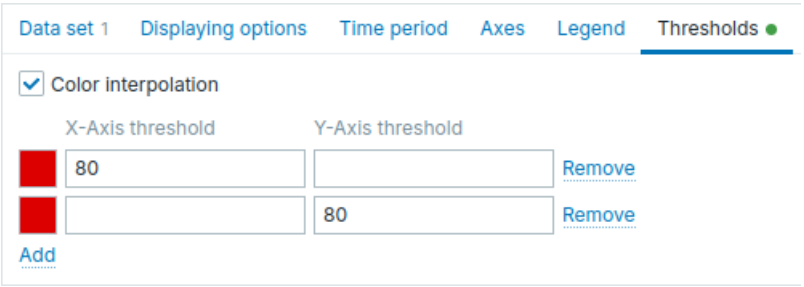
图例选项卡允许您自定义散点图图例：



显示图例	取消选中此复选框可隐藏散点图上的图例（默认选中）。
显示聚合函数	选中此复选框可在图例中显示聚合函数。
行	选择图例行的显示模式： 固定 - 显示的行数由行数参数值决定； 可变 - 显示的行数由已配置监控项的数量决定，但不会超过最大行数参数值。
行数/ 最大行数	如果行设置为“固定”，设置要显示的图例行数（1-10）。 如果行设置为“可变”，设置要显示的图例最大行数（1-10）。
列数	设置要显示的图例列数（1-4）。 当未选中显示最小值/平均值/最大值时，此参数可用。

阈值

阈值选项卡用于配置决定主机标记颜色的阈值：



颜色插值	选中复选框后，可在主机标记的阈值颜色之间启用平滑过渡。 如果为 X 轴、Y 轴中的任一轴或同时为两个轴设置了两个或更多阈值，则此参数适用。
X 轴/*Y* 轴阈值	点击添加以添加阈值，从颜色选择器中选择阈值颜色，并指定数值。 阈值可以应用于单个轴、分别应用于两个轴，或同时应用于两个轴。 例如，设置一个 X=80 且 Y=80 的阈值后，只有在两个条件均满足时，标记才会着色。 保存时，阈值列表将按升序排序。 支持后缀（例如“1d”、“2w”、“4K”、“8G”）。支持值映射。

SLA 报告

概述

SLA 报表小部件显示SLA 报表，帮助您根据已定义的目标监控服务性能。

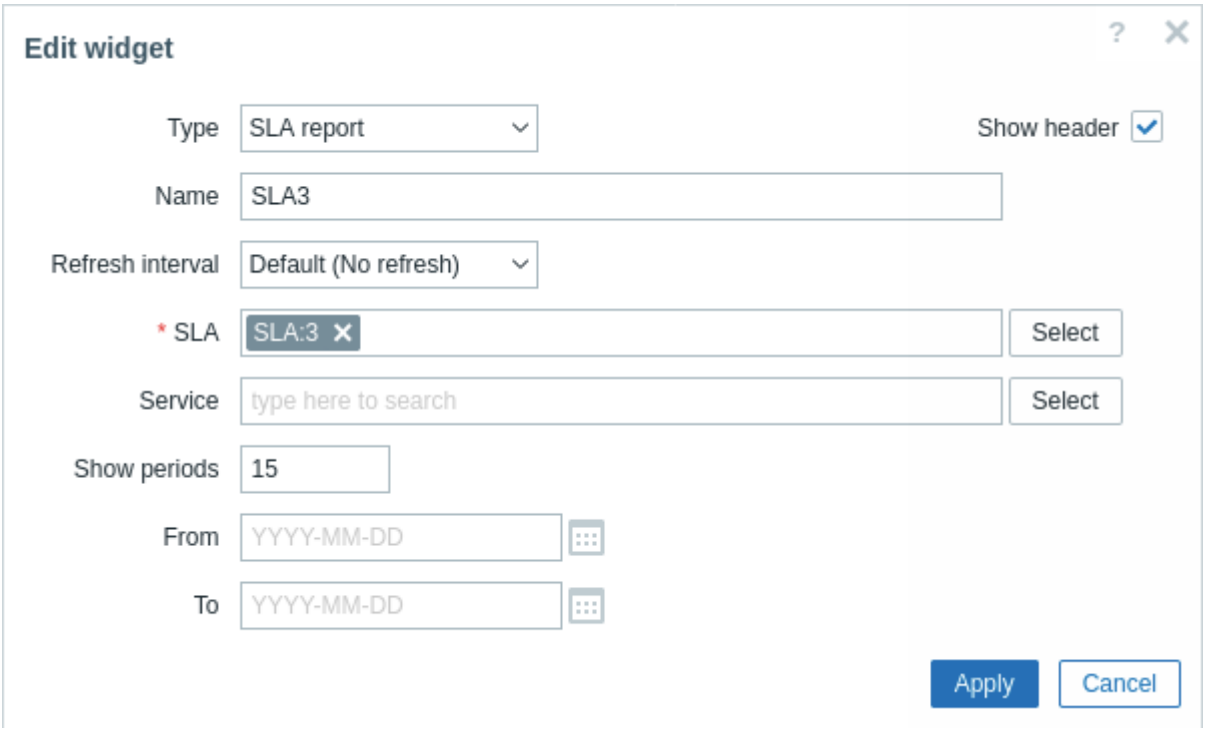
SLA report

Day	SLO	SLI	Uptime	Downtime	Error budget	Excluded downtimes
2025-09-04	99.9%	99.8437	5h 40m 50s	28s	-8s	
2025-09-03	99.9%	99.9479	23h 59m 15s	45s	41s	
2025-09-02	99.9%	99.9190	23h 58m 50s	1m 10s	16s	
2025-09-01	99.9%	99.9251	22h 58m 58s	1h 1m 2s	21s	2025-09-01 01:00 AM Maintenance: 1h
2025-08-31	99.9%	99.6863	23h 55m 29s	4m 31s	-3m 5s	
2025-08-30	99.9%	99.9190	23h 58m 50s	1m 10s	16s	
2025-08-29	99.9%	99.7095	23h 55m 49s	4m 11s	-2m 45s	
2025-08-28	99.9%	99.9421	23h 59m 10s	50s	36s	
2025-08-27	99.9%	99.7188	23h 55m 57s	3m 15s	-2m 37s	
2025-08-26	99.9%	99.7755	23h 56m 46s	3m 14s	-1m 48s	
2025-08-25	99.9%	99.9094	22h 58m 45s	1h 1m 15s	8s	2025-08-25 01:00 AM Maintenance: 1h

它显示与 服务 > *SLA* 报表 相同的数据。

配置

要进行配置，请选择 *SLA report* 作为类型：



除了所有部件通用的常见参数外，您还可以设置以下特定选项：

SLA	为报表选择 SLA。
Service	为报表选择服务。
Show periods	设置部件中显示的周期数（默认 20，最大 100）。
From	选择报表的开始日期。 支持相对日期：`now`、`now/d`、`now/w-1w` 等；支持的日期修饰符：`d`、`w`、`M`、`y`。

To	选择报表的结束日期。 支持相对日期：`now`、`now/d`、`now/w-1w` 等；支持的日期修饰符：`d`、`w`、`M`、`y`。

系统信息

概述

系统信息小部件显示 Zabbix 服务器和系统关键数据的摘要；当启用高可用性时，还会显示高可用性节点的详细信息。

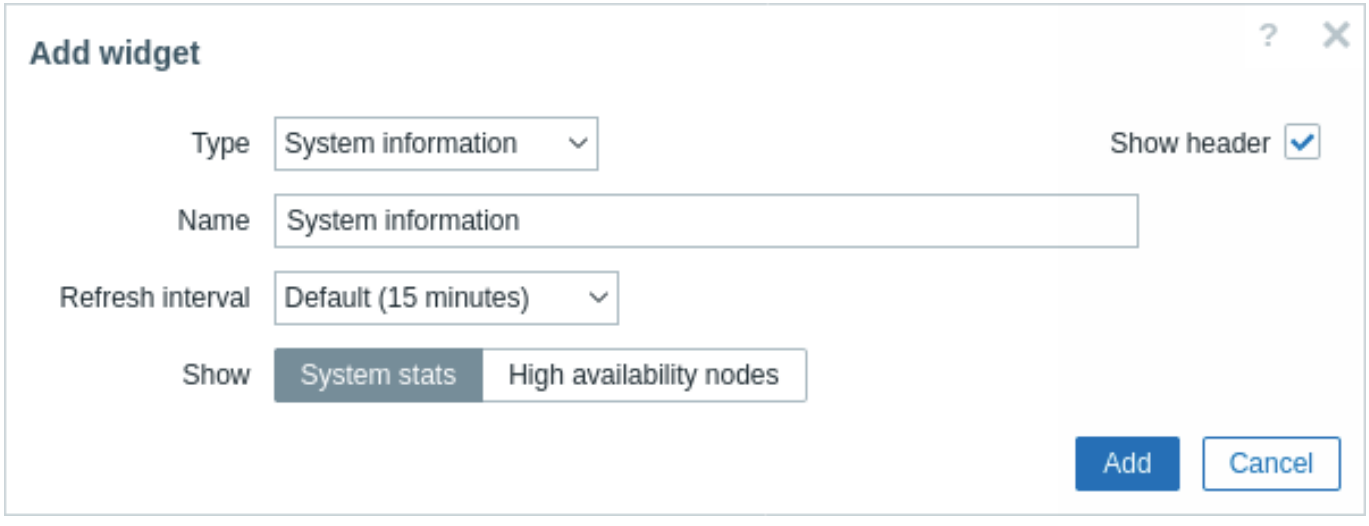
System information

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Zabbix server version	8.0.0	Up to date
Zabbix frontend version	8.0.0	Up to date
Software update last checked	2025-08-30	
Latest release	8.0.0	Release notes
Number of hosts (enabled/disabled)	2	2 / 0
Number of templates	312	
Number of items (enabled/disabled/not supported)	176	165 / 0 / 11
Number of triggers (enabled/disabled [problem/ok])	89	89 / 0 [4 / 85]
Number of users (online)	2	1
Required server performance, new values per second	2.68	
Global scripts on Zabbix server	Disabled	
High availability cluster	Disabled	

它显示的数据与 *Reports* > *System information* 相同，但不包括 *Zabbix server ID*。

配置

要进行配置，请选择 “系统信息” 作为类型：



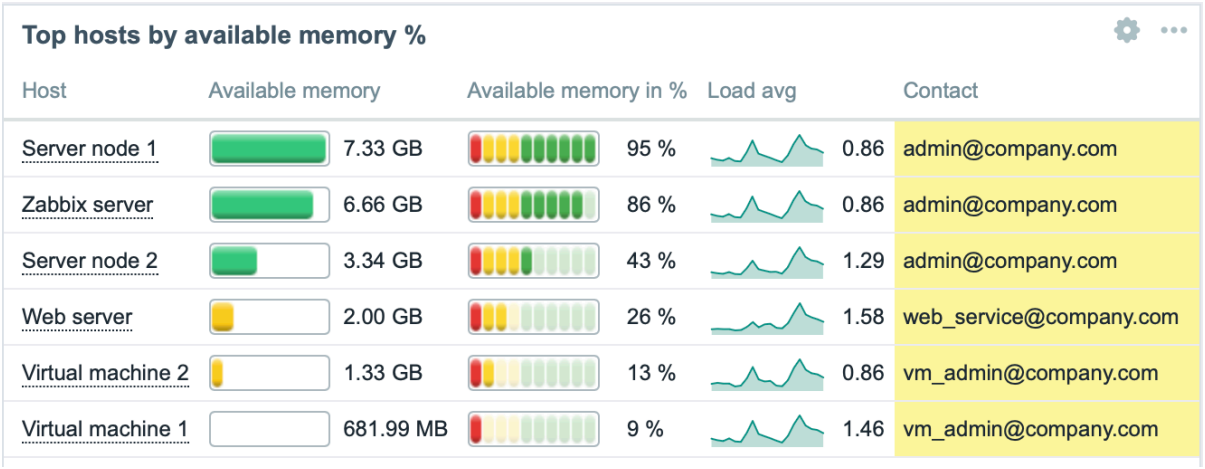
除了适用于所有小组件的通用参数外，还可以设置以下特定选项：

显示	选择要显示的内容： 系统统计信息 - 显示 Zabbix 服务器和系统关键数据的摘要； 高可用性节点 - 显示高可用性节点的状态（如果已启用高可用性集群）。
显示软件更新检查详细信息	选中此复选框可显示 Zabbix 软件更新检查的详细信息。 仅当在 Zabbix 服务器配置中启用了软件更新检查，并且在显示字段中选择了“系统统计信息”时，此选项才可用。

监控项最多的主机

概述

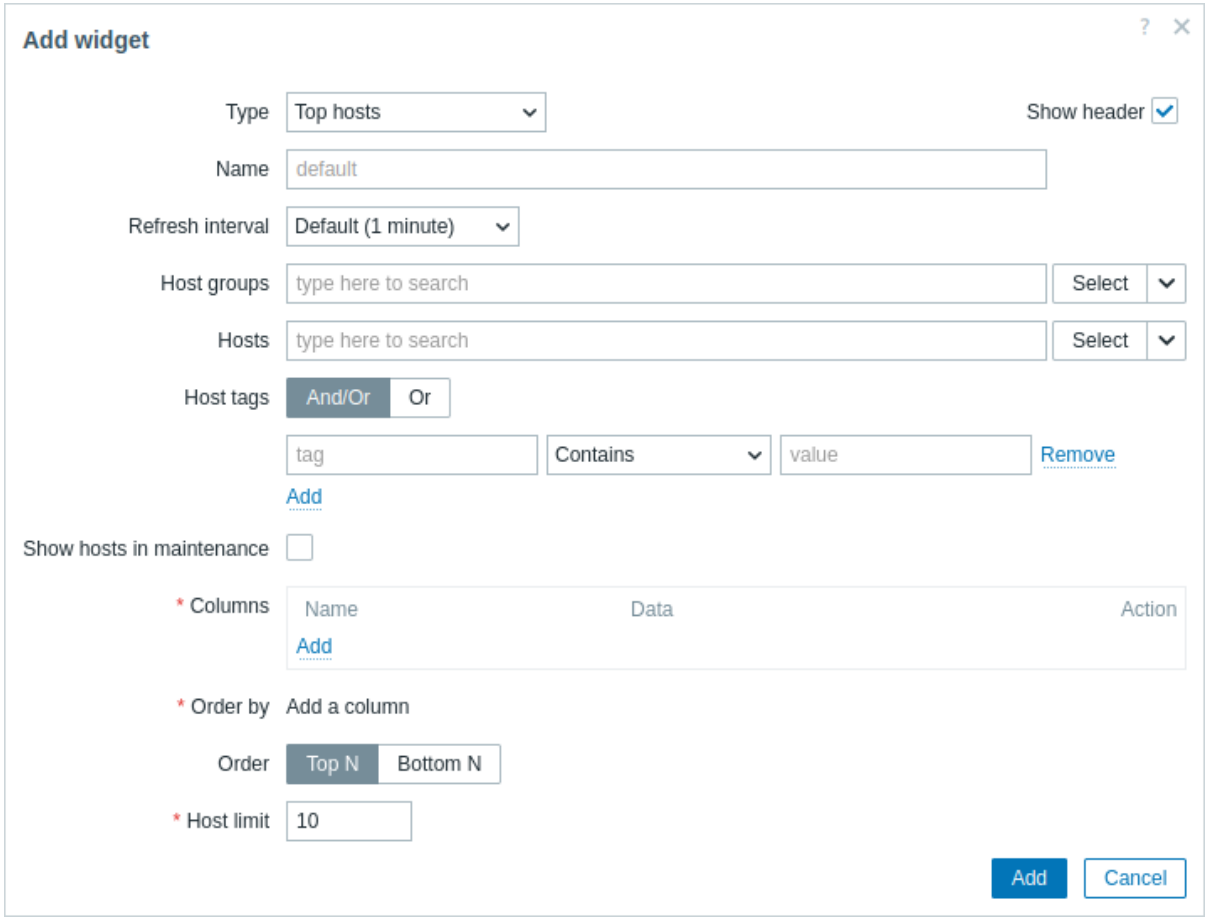
Top hosts 小组件显示来自多个主机的所选监控项的最高/最低值。



该小组件支持同时显示最多 1000 个主机。

配置

要配置，请选择 *Top* 主机作为类型：



除了所有部件通用的常见参数外，您还可以设置以下特定选项：

主机组	选择要在部件中显示的主机组。 或者，选择一个兼容的部件作为主机组的数据源。 此字段支持自动补全，因此开始输入组名时，将显示匹配组的下拉列表。 在模板仪表板上配置部件时，此参数不可用。
主机	选择要在部件中显示的主机。 或者，选择一个兼容的部件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名时，将显示匹配主机的下拉列表。 在模板仪表板上配置部件时，此参数不可用。
主机标签	指定标签以限制部件中显示的主机数量。 可以包含特定标签和标签值，也可以排除特定标签和标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件都有多个可用运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且其标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且其标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将按“或”条件分组； 或 - 只需满足一个条件即可。 在模板仪表板上配置部件时，此参数不可用。
显示维护中的主机	勾选此复选框后，也会显示处于维护中的主机（在这种情况下，主机名旁边将显示维护图标）。默认不勾选。
列	添加要显示的数据列。 列的顺序决定了它们从左到右的显示顺序。 可以通过拖动列名前方的手柄上下调整列顺序。
排序依据	从已定义的列列表中指定一列，用于 *Top N* 或 *Bottom N* 排序。

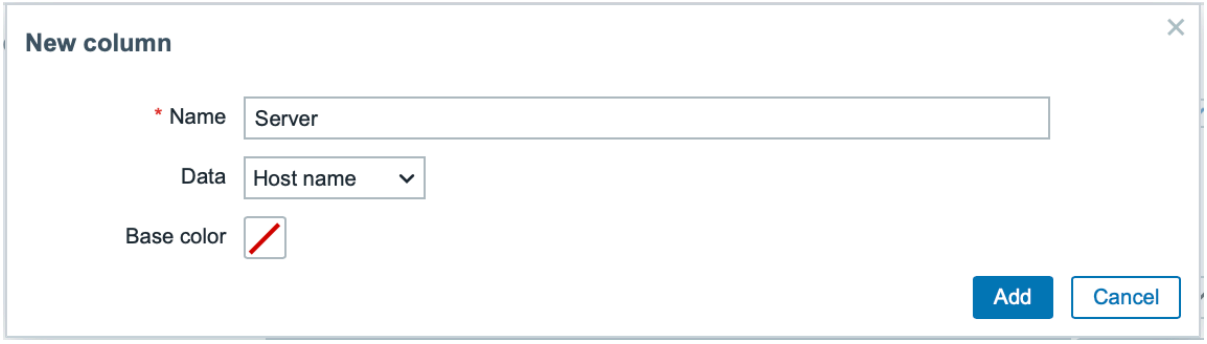
排序	指定行的排序方式： **Top N** - 根据排序依据的聚合值按降序排列； **Bottom N** - 根据排序依据的聚合值按升序排列。
主机限制	要显示的主机行数（1-1000）。 在模板仪表板上配置部件时，此参数不可用。

列配置

可以添加三种数据类型的列：主机名称、监控项值或文本。可用的列参数列表取决于列的数据类型；对于监控项值类型，还取决于值格式。

主机名称列

主机名称列用于显示主机的名称。



支持的参数：

Name	列的名称。
Data	要在列中显示的数据类型，选择 **Host name**。
Base color	列的背景颜色。

文本列

文本列用于显示任何指定的文本字符串。

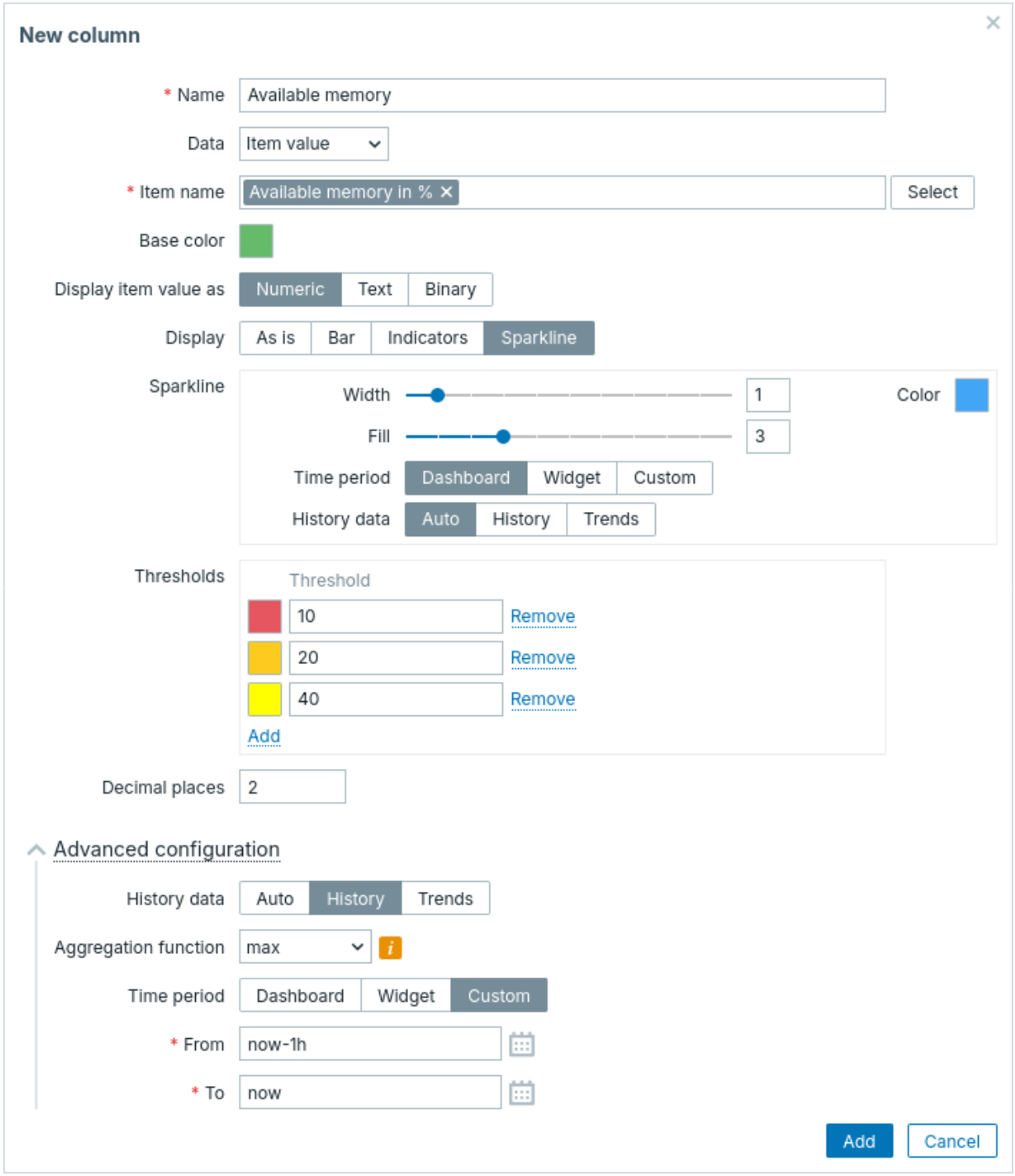


支持的参数：

名称	列的名称。
数据	要在列中显示的数据类型，选择 文本。
基础颜色	列的背景颜色。
文本	输入要显示的字符串。 可包含主机和资产清单inventory 宏。

监控项值列

监控项值列用于显示指定监控项的值；支持多种值显示格式和选项。



支持的参数：

名称	列的名称。
数据	列中显示的数据类型，选择 监控项值。
监控项名称	选择一个监控项；其名称用于匹配并显示所有选定主机上名称相同的监控项。如果某台主机有多个此类监控项，小组件将显示键按字母顺序排列最靠前的监控项（例如，显示键为 `proc.cpu.util` 的 *CPU* 利用率，而不是键为 `system.cpu.util` 的监控项）。 在模板仪表板上配置小组件时，只能选择在模板上配置的监控项。
基础颜色	列的背景颜色；当以条形图/指示器形式显示监控项值数据时，为填充颜色。如果监控项值超过指定阈值之一，可以使用自定义颜色覆盖默认颜色。
将监控项值显示为	监控项值的显示格式：数值、文本或二进制。 所选选项决定可用的其他参数。有关每种格式的参数，请参阅下面的参数列表。
高级配置	
历史数据	从历史数据或趋势数据中获取数据： 自动 - 自动选择； 历史数据 - 获取历史数据； 趋势数据 - 获取趋势数据。 此参数仅适用于数值监控项值。对于文本值和二进制值，数据始终从历史数据中获取。

聚合函数	指定在选定的时间段内使用的聚合函数： 最小值 - 显示最小值； 最大值 - 显示最大值； 平均值 - 显示平均值； 计数 - 显示值的数量； 总和 - 显示值的总和； 第一个 - 显示第一个值； 最后一个 - 显示最后一个值。 如果选择不使用，则显示最新值（不进行聚合）。 聚合允许针对指定时间间隔（5 分钟、一小时、一天）显示聚合值，而不是最新值。 对于最小值、最大值、平均值和总和，只能显示数值数据。对于计数，非数值数据将转换为数值。
时间段	指定用于聚合值的时间段。选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。 如果聚合函数设置为“不使用”，则此参数不可用。
小组件	输入或选择兼容的小组件，作为时间段的数据源。 当时间段设置为“小组件”时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。

数值监控项值参数：

显示	定义值的显示方式： 原样 - 以常规文本显示； 条形图 - 以纯色填充的实心条形图显示； 指示器 - 以分段的彩色条形图显示； 迷你趋势图 - 迷你折线图。
阈值	指定背景/填充颜色应发生变化时的阈值。 保存时，列表将按升序排序。 对于迷你趋势图，阈值仅应用于监控项的最后一个值。
小数位数	指定值显示的小数位数。
条形图/指示器	
最小值	最小值。
最大值	最大值。
迷你趋势图	
宽度	使用滑块设置图形线条粗细，或手动输入 0 到 10 范围内的值。
颜色	选择线条和填充颜色。
填充	使用滑块设置填充颜色的透明度，或手动输入 0 到 10 范围内的值。
时间段	指定要包含在迷你趋势图中的值的时间段。选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论小组件的时间段配置如何，兼容的小组件仍可将其用作时间段的数据源。
小组件	输入或选择兼容的小组件，作为时间段的数据源。 当时间段设置为“小组件”时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（now、now/d、now/w-1w 等）。 当时间段设置为“自定义”时，此参数可用。

历史数据	从历史数据或趋势数据中获取数据： 自动 - 自动选择； 历史数据 - 获取历史数据； 趋势数据 - 获取趋势数据。

文本监控项值参数：

高亮显示	指定正则表达式；匹配这些表达式时，背景/填充颜色将发生变化。

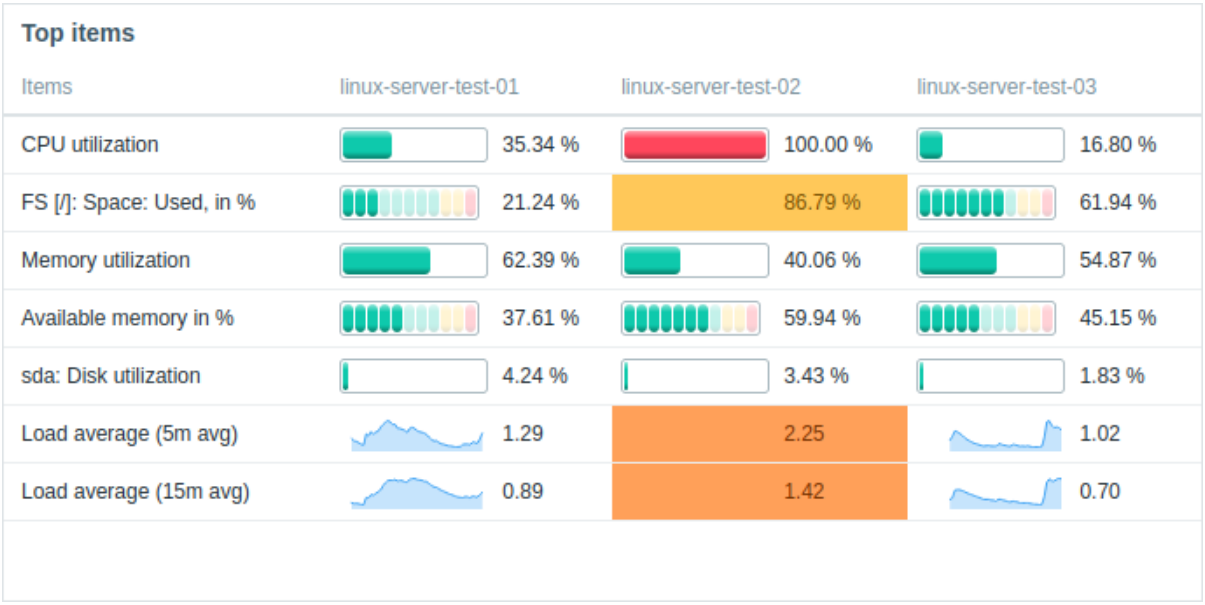
二进制监控项值参数：

显示缩略图	指定是否为包含二进制数据的图像创建并显示缩略图，或者在值列中显示指向完整尺寸图像的超链接显示。

监控项排行

概述

Top items 小部件显示所选监控项的最高/最低值，从而快速概览其性能。

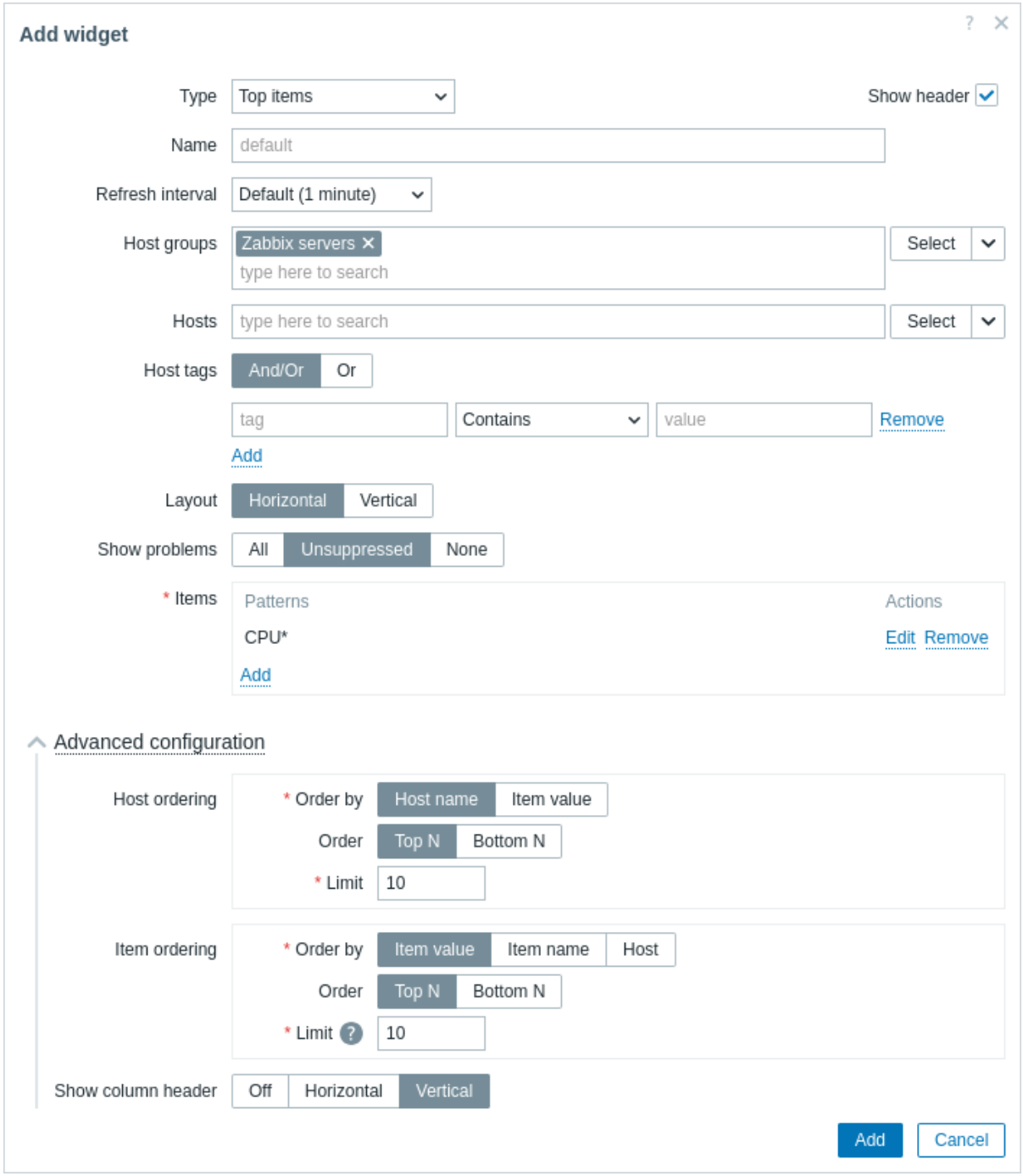


默认情况下，该小部件显示最近 24 小时内的值。
此时间段可在 *Administration* > *General* > *GUI*（*Max history display period*）中更改。

点击监控项值将打开监控项菜单。

配置

要进行配置，请将类型选择为 *Top items*：



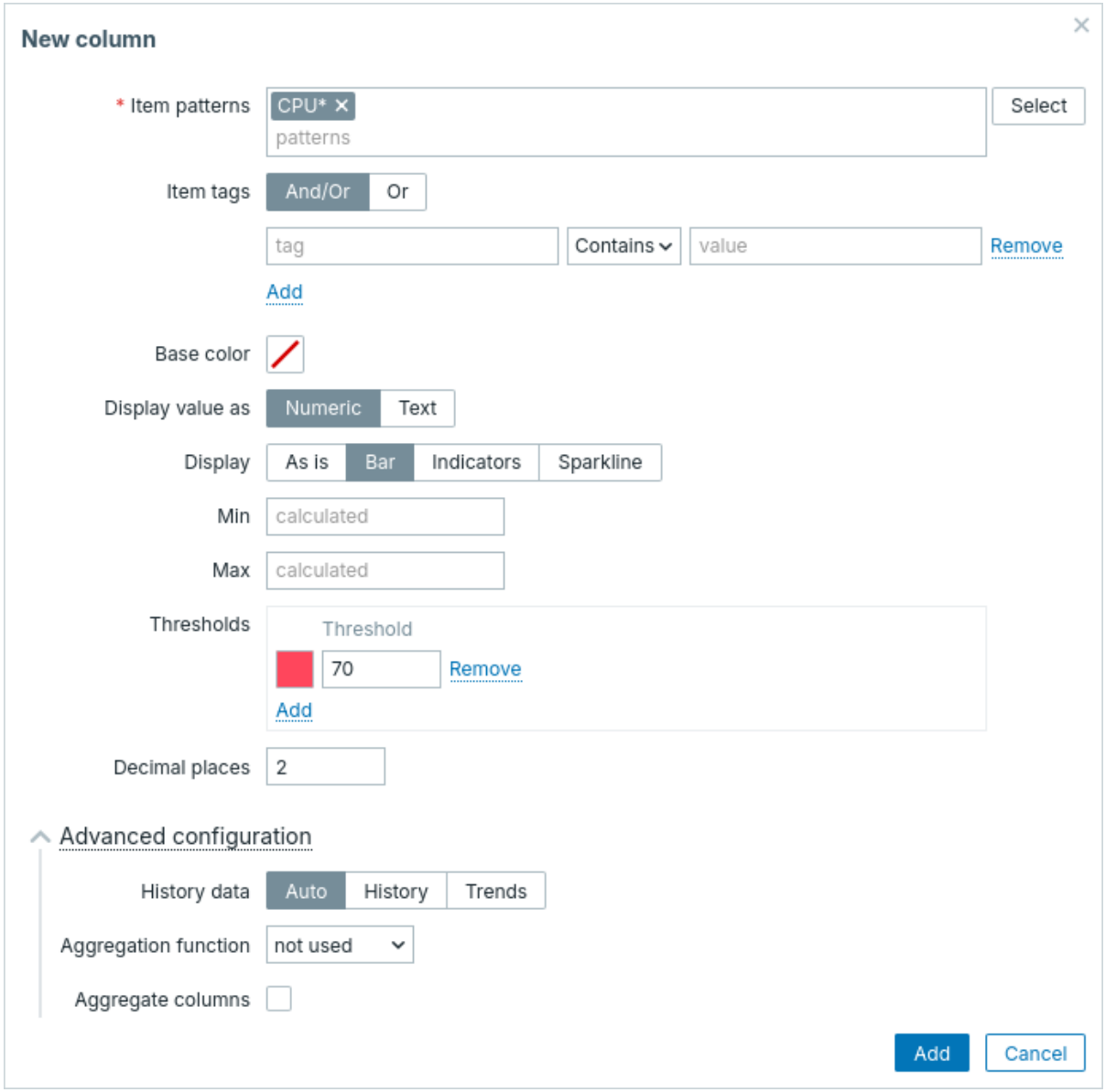
除了适用于所有小部件的通用参数外，您还可以设置以下特定选项：

Host groups	选择主机组。 或者，选择一个兼容的小部件作为主机组的数据源。 此字段支持自动补全，因此开始输入组名时会显示匹配组的下拉列表。 在模板仪表板上配置该小部件时，此参数不可用。
Hosts	选择主机。 或者，选择一个兼容的小部件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名时会显示匹配主机的下拉列表。 在模板仪表板上配置该小部件时，此参数不可用。

Host tags	指定标签以限制小部件中显示的主机数据数量。 可以包含或排除特定标签及标签值。可设置多个条件。标签名匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名； **Equals** - 包含指定的标签名和值（区分大小写）； **Contains** - 包含标签值中含有输入字符串的指定标签名（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名； **Does not equal** - 排除指定的标签名和值（区分大小写）； **Does not contain** - 排除标签值中含有输入字符串的指定标签名（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名的条件将按 *Or* 条件分组； **Or** - 满足任一条件即可。
Layout	选择布局选项： **Horizontal** - 主机名将显示在左侧； **Vertical** - 主机名将显示在顶部。
Show problems	根据问题状态筛选要显示的问题（全部、未抑制、无）。 问题会以彩色监控项值显示，并覆盖监控项列的显示设置。颜色基于问题严重性，可在 *Administration > General > Trigger displaying options* 中自定义。问题严重性可在更新问题时更改。
Items	添加监控项模式或特定监控项以供显示（参见列配置）。
Advanced configuration	
Host ordering	选择主机列/行的排序选项。
Order by	按以下方式对主机排序： **Host name** - 按主机名排序； **Item value** - 按所选监控项的值排序。
Order	选择显示最高值还是最低值： **TopN** - 前 N 个值； **BottomN** - 后 N 个值。 N 的值在 *Limit* 字段中选择。
Limit	输入可显示主机的限制值（范围 1-1000；默认值 10）。 此值将成为 *Order* 字段中的 N 值。
Item ordering	选择监控项列/行的排序选项。
Order by	按以下方式对监控项排序： **Item value** - 按监控项值排序； **Item name** - 按监控项名称排序； **Host** - 按所选主机模式排序。
Order	选择显示最高值还是最低值： **TopN** - 前 N 个值； **BottomN** - 后 N 个值。 N 的值在 *Limit* 字段中选择。
Limit	输入可显示监控项的限制值（范围 1-1000；默认值 10）。 此值将成为 *Order* 字段中的 N 值。
Show column header	选择列标题显示选项： **Off** - 不显示列标题； **Horizontal** - 在标题中水平显示文本； **Vertical** - 在标题中垂直显示文本。

列配置

要配置监控项列，请在 *Items* 参数中单击 *Add*：



通用列参数：

Item patterns	使用通配符指定一个或多个监控项模式。 或者，直接选择监控项。 在模板仪表板上配置该小部件时，只能选择模板上已配置的监控项。
Item tags	指定标签，以限制小部件中显示的监控项数据数量。 可以包含或排除特定的标签和标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下几种运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含指定的标签名称，且其标签值包含输入的字符串（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除指定的标签名称，且其标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 *Or* 条件分组； **Or** - 只需满足一个条件即可。
Base color	选择列的背景颜色；如果 *Display* 设置为 “Bar” 或 “Indicators”，则选择填充颜色。 请注意，基础颜色可被阈值颜色或高亮颜色覆盖。

Display value as	监控项值的显示格式：**Numeric** 或 **Text**。 所选选项决定了哪些附加参数可用。请参见下方各格式对应的参数列表。

数值的列参数：

显示	定义值的显示方式： 原样 - 以常规文本显示； 条形图 - 以实心、填充颜色的条形图显示； 指示器 - 以分段、填充颜色的条形图显示； 迷你图 - 迷你折线图。
最小值	条形图/指示器显示的最小值。
最大值	条形图/指示器显示的最大值。
宽度	使用滑块或手动输入 0 到 10 范围内的值来设置图形线条粗细。 此参数仅适用于迷你图显示。
填充	使用滑块或手动输入 0 到 10 范围内的值来设置填充颜色的透明度级别。 此参数仅适用于迷你图显示。
颜色	选择线条和填充颜色。 此参数仅适用于迷你图显示。
有关时间段和历史数据选择相关字段的说明，请参阅高级配置。	
阈值	指定背景色/填充色应发生变化的阈值。 保存时，列表将按升序排序。 对于迷你图，阈值仅应用于监控项的最新值。
小数位数	指定值显示的小数位数。
高级配置	
历史数据	从历史数据或趋势数据中获取数据： 自动 - 自动选择； 历史数据 - 获取历史数据； 趋势数据 - 获取趋势数据。 此参数仅适用于数值型监控项值。对于文本值，数据始终从历史数据中获取。
聚合函数	指定在选定的时间段内使用的聚合函数： **min** - 显示最小值； **max** - 显示最大值； **avg** - 显示平均值； **count** - 显示值的数量； **sum** - 显示值的总和； **first** - 显示第一个值； **last** - 显示最后一个值。 如果选择不使用，则显示最新值（不进行聚合）。 聚合允许针对指定的时间间隔（5 分钟、1 小时、1 天）显示聚合值，而不是最新值。 对于 *min*、*max*、*avg* 和 *sum*，只能显示数值数据。对于 *count*，非数值数据将转换为数值。
时间段	指定用于聚合值的时间段。选择时间段的数据源： 仪表板 - 使用仪表板的时间段选择器； 小组件 - 使用兼容的小组件（在小组件参数中设置）； 自定义 - 使用在从和到参数中设置的自定义时间段；如果已设置，鼠标悬停时，小组件右上角将显示时钟图标，指示所设置的时间。 请注意，无论兼容小组件的时间段配置如何，兼容小组件仍可将其用作时间段的数据源。 如果聚合函数设置为 “不使用”，则此参数不可用。
小组件	输入或选择兼容的小组件作为时间段的数据源。 当时间段设置为 “小组件” 时，此参数可用。
从	输入或选择时间段的开始时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 当时间段设置为 “自定义” 时，此参数可用。
到	输入或选择时间段的结束时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 当时间段设置为 “自定义” 时，此参数可用。

聚合列	选中后，将所有与该列监控项模式匹配的监控项分组到单个合并列/行中。匹配监控项的值将使用选定的列聚合函数进行聚合，并显示在合并列名称下。默认：关闭（未选中）。此参数仅在将值显示为设置为数值时显示。
列聚合函数	选择如何合并分组监控项模式中的值。可用选项：**min**、**max**、**avg**、**count**、**sum**（默认）。仅在选中聚合列时显示此参数。
合并列名称	输入聚合列的显示名称。选中聚合列后，此字段为必填项，并将作为合并值的列标题。

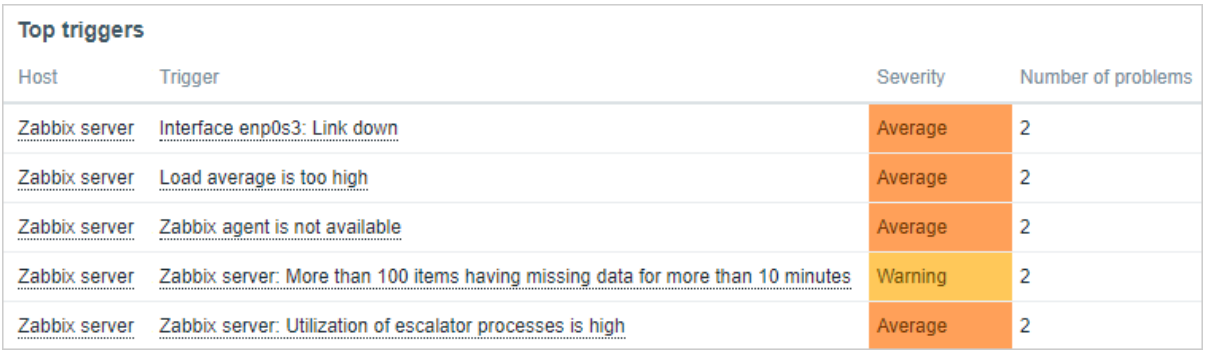
文本值的列参数：

高亮显示	指定正则表达式。匹配这些表达式时，背景色/填充色将发生变化。

顶级触发器

概述

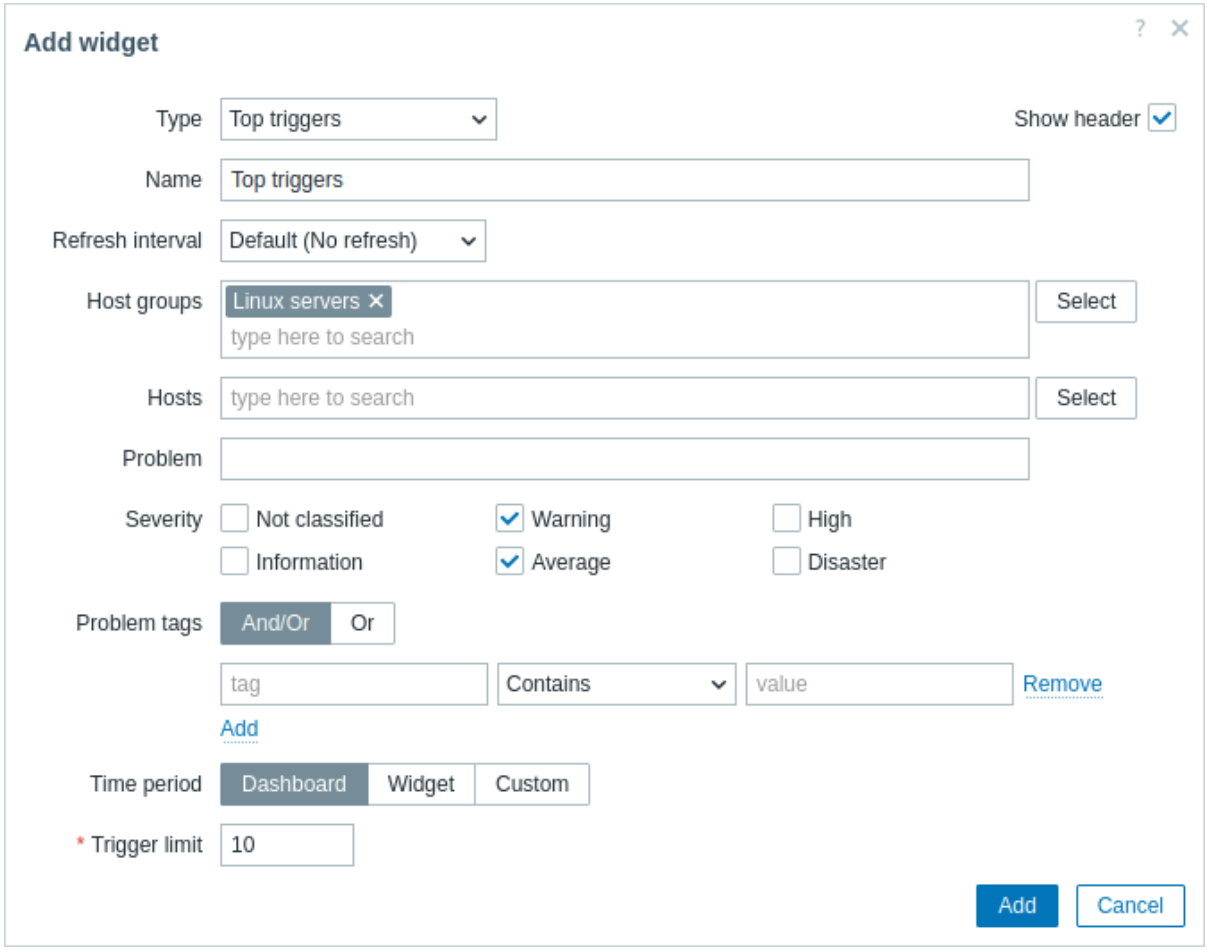
Top triggers 小部件显示问题数量最多的触发器。



它显示与 * 报告 > Top 100 triggers* 相同的数据，并且最多可显示 1000 条记录。

配置

要进行配置，请选择 *Top triggers* 作为部件类型：



除了所有部件通用的常见参数外，还可以设置以下特定选项：

Host groups	选择要在部件中显示其问题的主机组。 此字段支持自动补全，因此开始输入组名时，将提供匹配组的下拉列表。 指定父主机组时，将隐式选择其下所有嵌套主机组。 这些主机组中的问题将显示在部件中；如果未输入任何主机组，则将显示所有主机组中的问题。 在模板仪表板上配置部件时，此参数不可用。
Hosts	选择要在部件中显示其问题的主机。 此字段支持自动补全，因此开始输入主机名时，将提供匹配主机的下拉列表。 如果未输入任何主机，则将显示所有主机的问题。 在模板仪表板上配置部件时，此参数不可用。
Problem	您可以仅查看特定问题的触发器。为此，请输入要在问题名称中匹配的字符串。 宏不会被展开。
Severity	标记触发器严重性，以筛选要在部件中显示的触发器。 如果未标记任何严重性，则将显示所有触发器。
Problem tags	指定要在部件中显示的问题标签。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件都有多个可用运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 *Or* 条件分组； **Or** - 只需满足一个条件即可。

Time period	为时间段选择数据源： **Dashboard** - 使用仪表板的时间段选择器； **Widget** - 使用兼容的部件（在 *Widget* 参数中设置）； **Custom** - 使用在 *From* 和 *To* 参数中设置的自定义时间段；如果已设置，部件右上角将显示一个时钟图标，鼠标悬停时会指示所设置的时间。 请注意，无论部件的 *Time period* 如何配置，兼容部件仍可将其用作时间段的数据源。
Widget	输入或选择一个兼容部件，作为时间段的数据源。 如果 *Time period* 设置为"Widget"，则此参数可用。
From	输入或选择时间段的开始时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 如果 *Time period* 设置为"Custom"，则此参数可用。
To	输入或选择时间段的结束时间。 支持相对时间语法（`now`、`now/d`、`now/w-1w` 等）。 如果 *Time period* 设置为"Custom"，则此参数可用。
Trigger limit	设置要显示的触发器数量。可能的取值范围：1-1000。

触发器概览

概述

Trigger overview 小部件以彩色块表格的形式显示主机触发器的当前状态。它可为不同主机上的触发器健康状况和活动情况提供快速的可视化概览。



该小部件使用多个可视元素来传达触发器详情：

- 对于 PROBLEM 触发器，块颜色与问题严重性相匹配。
- 闪烁的块表示触发器状态最近发生了变化（在最近 2 分钟内）。
- 灰色箭头表示具有依赖关系的触发器（将鼠标悬停其上可查看详情）。
- 复选框图标表示已确认的问题；只有当该触发器的所有活动或已解决问题都已确认时，才会显示此图标。

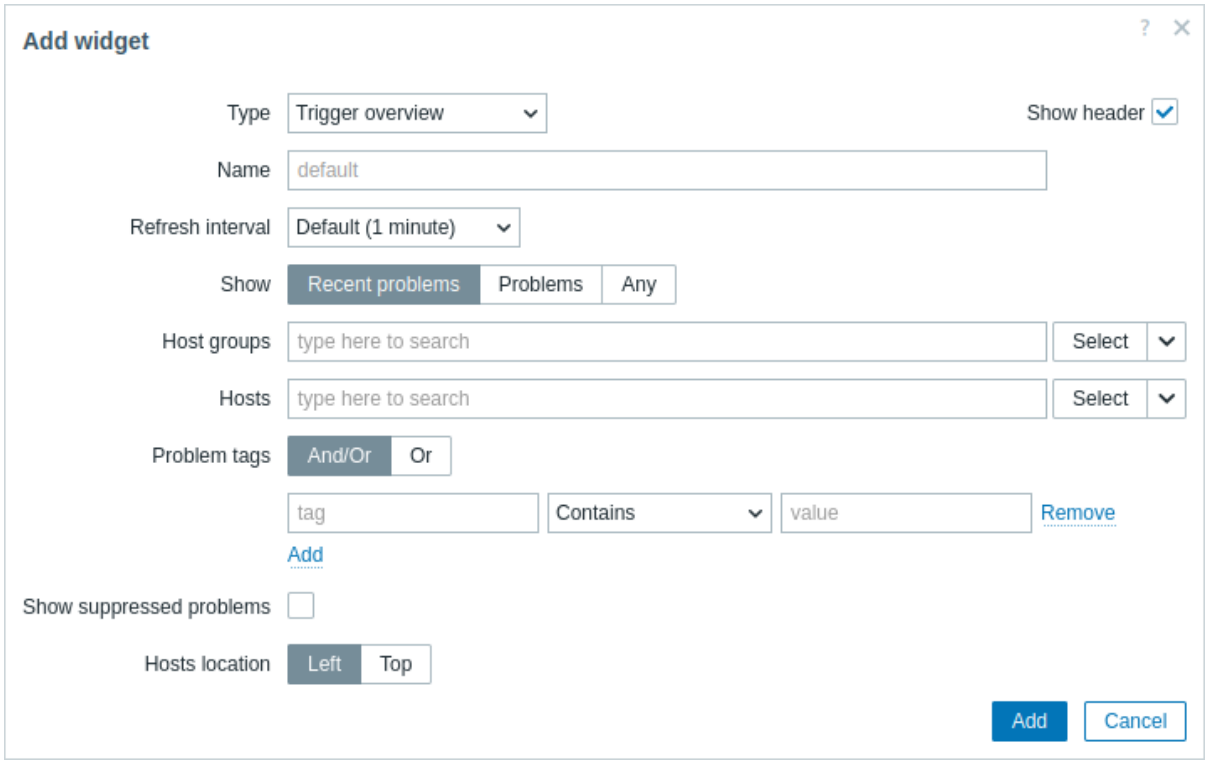
Note:
触发器严重性颜色可在 *Administration* > *General* 中调整。在更新问题 时可以修改触发器严重性。

单击某个块会提供与触发器相关的信息链接（问题、触发器配置等）。

默认情况下，最多显示 50 条记录（可在 *Administration* > *General* > *GUI* 下的 *Max number of columns and rows in overview tables* 中进行配置）。如果存在更多触发器，系统会提示您细化筛选条件。此限制会在任何附加筛选之前应用（例如按标签筛选）。

配置

要进行配置，请选择 触发器概览作为类型：



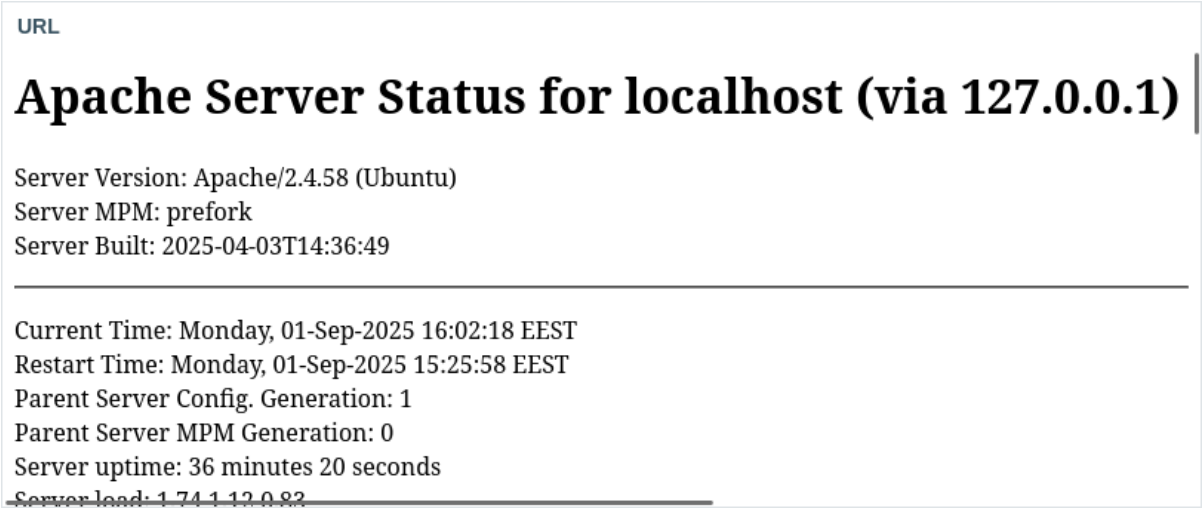
除了所有部件通用的常用参数外，您还可以设置以下特定选项：

显示	按触发器状态筛选触发器： 最近的问题 - *(默认)* 显示最近曾处于或当前仍处于 PROBLEM 状态的触发器（已恢复和未恢复）； 问题 - 显示处于 PROBLEM 状态的触发器（未恢复）； 任意 - 显示所有触发器。
主机群组	选择主机群组。 或者，选择一个兼容的部件作为主机群组的数据源。 此字段支持自动补全，因此开始输入群组名称时，将提供匹配群组的下拉列表。 在模板仪表板上配置该部件时，此参数不可用。
主机	选择主机。 或者，选择一个兼容的部件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名称时，将提供匹配主机的下拉列表。 在模板仪表板上配置该部件时，此参数不可用。
问题标签	指定标签以筛选部件中显示的触发器。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 注意：如果参数 显示设置为 “任意”，即使指定了标签，也会显示所有触发器。但是，虽然最近的触发器状态变化（显示为闪烁块）会针对所有触发器更新，触发器状态详细信息（问题严重性颜色以及问题是否已确认）则只会针对匹配指定标签的触发器更新。 每个条件可使用以下几种运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将按 或条件分组； 或 - 只需满足一个条件即可。
显示已抑制的问题	勾选此复选框以显示原本会因主机维护而被抑制（不显示）的问题。
布局	选择布局选项： 水平 - 主机名称显示在左侧； 垂直 - 主机名称显示在顶部。

URL

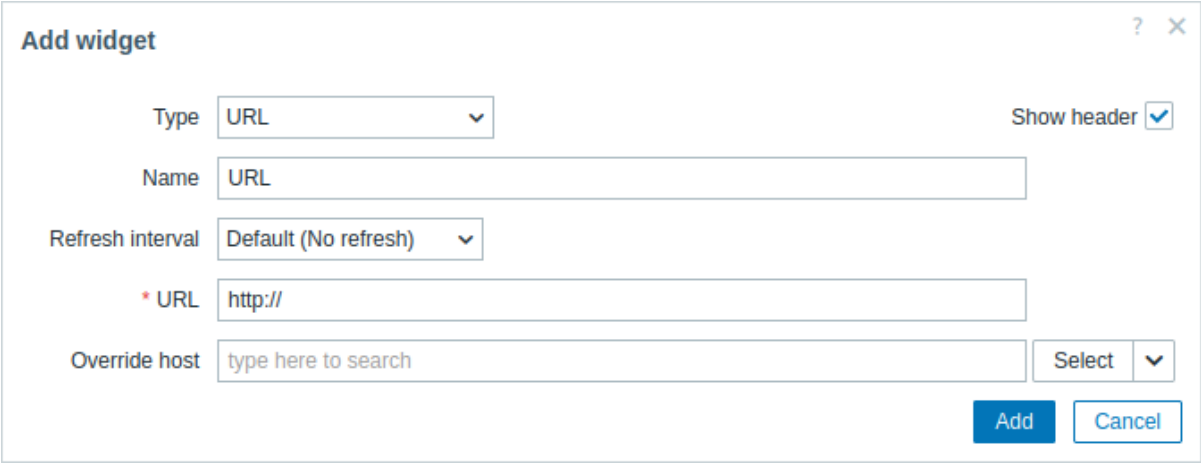
概述

URL 小部件显示从 URL 检索到的内容。



配置

要进行配置，请将类型选择为 *URL*：



除了所有小组件通用的公共参数外，您还可以设置以下特定选项：

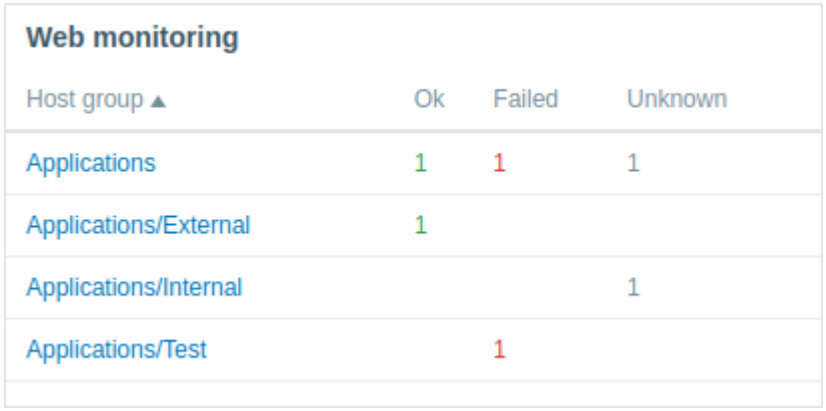
URL	输入要显示的 URL（最多 2048 个字符）。 外部 URL 必须以 `http://` 或 `https://` 开头。 内部 URL 支持相对路径（例如，`zabbix.php?action=report.status`）。 支持 `{HOST.*}` 宏。
Override host	选择一个兼容的小组件或仪表板的主机选择器作为主机的数据源。 在模板仪表板上配置该小组件时，此参数不可用。

Attention:
如果 Zabbix 前端通过 HTTPS 访问，浏览器可能不会加载在小组件中配置的 HTTP 页面。此外，您可能需要修改X-Frame-Options HTTP 响应头，以显示 Zabbix 前端之外的内容。

Web 监控

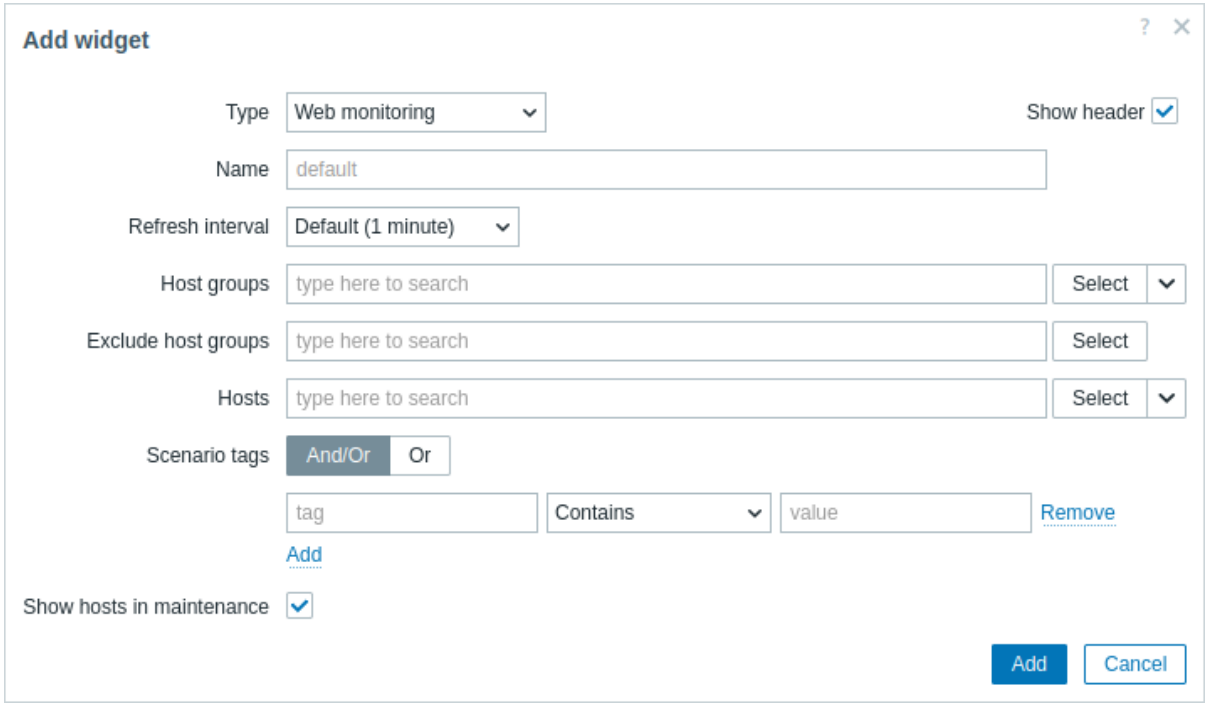
概述

Web 监控小组件显示活动 Web 场景的状态。



单击主机组名称将打开 监控 > 主机 > *Web* 场景，您可以在其中查看场景详细信息。

配置



除了所有部件都通用的常用参数外，您还可以设置以下特定选项：

主机群组	选择要在部件中显示的主机群组。 或者，选择一个兼容的部件作为主机群组的数据源。 此字段支持自动补全，因此开始输入群组名称时，将显示匹配群组的下拉列表。 指定父主机群组会隐式选择其下所有嵌套的主机群组。 这些主机群组中的主机数据将显示在部件中；如果未输入任何主机群组，则将显示所有主机群组。 在模板仪表板上配置该部件时，此参数不可用。
排除主机群组	选择要从部件中隐藏的主机群组。 此字段支持自动补全，因此开始输入群组名称时，将显示匹配群组的下拉列表。 指定父主机群组会隐式选择其下所有嵌套的主机群组。 这些主机群组中的主机数据将不会显示在部件中。例如，主机 001、002、003 可能属于群组 A，而主机 002、003 也属于群组 B。如果我们同时选择显示群组 A 并排除群组 B，则仪表板中将只显示主机 001 的数据。 在模板仪表板上配置该部件时，此参数不可用。
主机	选择要在部件中显示的主机。 或者，选择一个兼容的部件或仪表板作为主机的数据源。 此字段支持自动补全，因此开始输入主机名称时，将显示匹配主机的下拉列表。 如果未输入任何主机，则将显示所有主机。 在模板仪表板上配置该部件时，此参数不可用。

场景标签	指定标签，以限制在部件中显示的 Web 场景数量。 可以包含特定标签和标签值，也可以排除特定标签和标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下几种运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 *Or* 条件分组； **Or** - 只要满足一个条件即可。
显示维护中的主机	在统计中包含处于维护状态的主机。 在模板仪表板上配置该部件时，此参数标签显示为 在维护中显示数据。

监控

概述

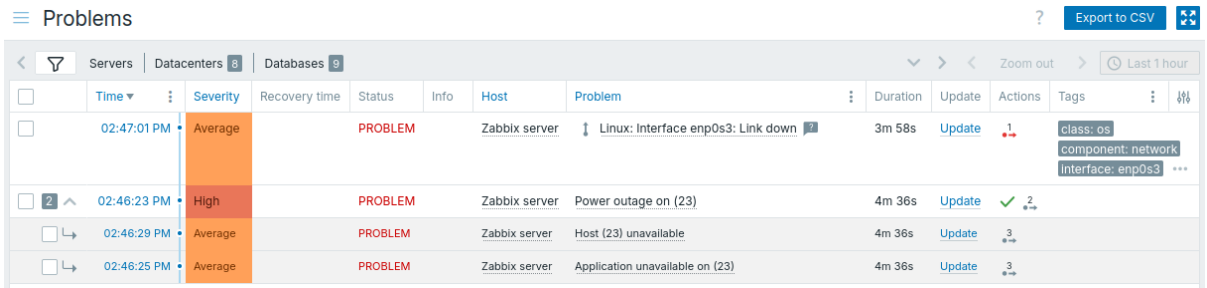
“监控” 菜单用于显示数据。无论 Zabbix 配置为收集、可视化和处理何种信息，这些信息都会显示在 “监控” 菜单的各个部分中。

问题

概述

在 *Monitoring* > *Problems* 中，你可以查看当前存在的问题。问题是处于 “Problem” 状态的那些触发器。

默认情况下，所有新问题都被归类为原因问题。可以手动将某些问题重新分类为原因问题的症状问题。有关更多详细信息，请参见原因和症状事件。



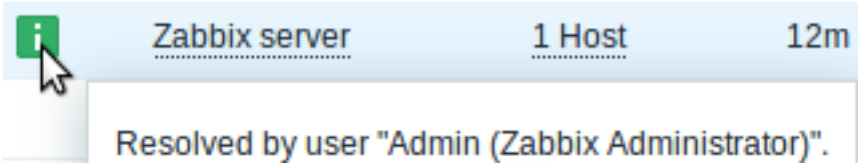
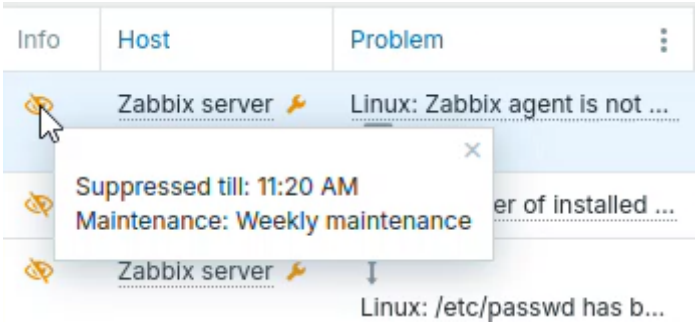
问题详情会显示在一个可自定义的表格中。表头中的 *Customize table* 按钮可让你显示、隐藏或重新排序列，并将布局重置为默认值。还提供以下选项：

- *Compact view* - 减少行高，以便在屏幕上显示更多问题。
- *Highlight full row* - 使用问题严重性颜色高亮显示未解决的问题（在高对比度主题中不可用）。

你还可以通过拖动每列右边框来调整列宽；双击边框可重置。所有自定义都会立即生效，并且始终会为当前用户在活动的filter 选项卡（默认或已保存）中记住。

可用的列如下：

Column	Description
Checkbox	显示用于选择问题的复选框。 复选框旁边的图标含义如下：  - 原因问题对应的症状事件数量；  - 展开以显示症状事件；  - 折叠以隐藏症状事件；  - 这是一个症状事件。

Column	Description
Time	显示问题开始时间。 点击列标题中的 *Options* 按钮，可使用 *Show timeline* 选项，用于切换可视化时间线和问题分组（启用 *Compact view* 时不可用）。
Severity	显示问题严重性。 问题严重性最初基于底层问题触发器的严重性，但在事件发生后，可以通过 *Update problem* screen 进行更新。问题严重性的颜色会在问题持续期间用作单元格背景色。
Recovery time	显示问题恢复时间。 当filter 中的 *Show* 设置为 *Problems* 时，此列不可用。
Status	显示问题状态： **Problem** - 未解决的问题 **Resolved** - 最近已解决的问题。你可以使用筛选器隐藏最近已解决的问题。 新问题和最近已解决的问题会闪烁 2 分钟。已解决的问题总共显示 5 分钟。这两个值都可以在 *Administration* > *General* > *Trigger displaying options* 中配置。 当filter 中的 *Show* 设置为 *Problems* 时，此列不可用。
Info	如果问题因全局关联或在更新问题时手动关闭，则会显示绿色信息图标。将鼠标悬停在图标上会显示更多详细信息：  如果显示的是被抑制的问题，则会显示以下图标：  。当事件因主机维护而被抑制时，图标会显示为橙色：  。如果同时适用维护和手动抑制，则图标颜色以手动抑制为优先。将鼠标悬停在图标上会显示更多详细信息： 
Host	显示问题主机。 点击主机名称会打开host menu。
Problem	显示问题名称。 问题名称基于底层问题触发器的名称。 触发器名称中的宏会在问题发生时解析，解析后的值之后不会再更新。 点击问题名称会打开event menu。 将鼠标悬停在问题名称后的  图标上会显示触发器描述（适用于包含该描述的问题）。 点击列标题中的 *Options* 按钮，可使用以下两个选项： - *Show operational data*：在问题名称后以括号形式显示运行数据（仅当trigger configuration 中的 *Operational data* 字段不为空时）。 - *Show trigger expression*：在问题名称下方显示该问题的触发器表达式（启用 *Compact view* 时不可用）。
Operational data	显示问题的动态实时数据。内容取决于trigger configuration 中的 *Operational data* 字段：如果定义了包含宏的字符串，则在此显示；如果该字段留空，则改为显示触发器表达式中所有监控项的最新值。
Duration	显示问题持续时间。 另请参见：负问题持续时间
Update	点击 *Update* 链接可进入problem update 界面，在那里可以对问题执行各种操作，包括添加评论和确认问题。

Column	Description
Actions	使用符号图标显示与问题相关的活动历史：  - 问题已被确认。此图标始终最先显示。  - 已添加评论。还会显示评论数量。  - 问题严重性已提高（例如，Information > Warning）。  - 问题严重性已降低（例如，Warning > Information）。  - 问题严重性已更改，但又恢复到原始级别（例如，Warning > Information > Warning）。  - 问题已被抑制。  - 问题已取消抑制。  - 问题因主机维护而被抑制。  - 问题在主机维护后已取消抑制。  - 已执行操作。还会显示操作数量。  - 已执行操作，且至少有一个正在进行中。还会显示操作数量。  - 已执行操作，且至少有一个失败。还会显示操作数量。 将鼠标移到这些图标上时，会显示包含活动详情的弹出窗口。有关弹出窗口中用于已执行操作的图标的更多信息，请参见查看详情。
Tags	显示问题tags。 此外，还可能显示来自外部工单系统的标签（请参见配置webhooks 时的 *Process tags* 选项）。 将鼠标悬停在已显示标签后的 ··· 三点图标上，或点击该图标，会打开一个包含所有标签的弹出窗口。 点击列标题中的 *Options* 按钮，可设置显示的标签数量（1-3）、选择标签名称的显示方式（完整、缩短或不显示）、设置标签显示优先级，或复制该列（复制后的列可重命名或删除）。
Tag value	显示问题tag 值。 点击列标题中的 *Options* 按钮，可重命名该列、选择要显示的标签值，或复制该列（复制后的列可重命名或删除）。
Custom text	显示自定义文本、宏值，或两者的组合。 点击列标题中的 *Options* 按钮，可重命名该列、输入自定义文本或宏，或复制该列（复制后的列可重命名或删除）。 支持的宏：`{HOST.*}`（除 `{HOST.METADATA}` 外）、`{INVENTORY.*}`、`{ITEM.*}`（除 `{ITEM.STATE.ERROR}` 外）、`{TRIGGER.ID}`、`{EVENT.ID}`，以及user macros。

负的问题持续时间

实际上，在一些常见情况下，问题持续时间可能为负值，即问题恢复时间早于问题创建时间。例如：

- 如果某个主机由 proxy 监控，并且发生网络错误，导致一段时间内未从 proxy 接收到数据，服务器将触发 nodata(/host/key) 触发器。当连接恢复后，服务器将从 proxy 接收到带有过去时间戳的监控项数据。随后，nodata(/host/key) 问题将被恢复，并且其问题持续时间将为负值；
- 当用于恢复问题事件的监控项数据由 Zabbix sender 发送，且其包含的时间戳早于问题创建时间时，也会显示负的问题持续时间。

Note:
负的问题持续时间不会以任何方式影响特定触发器的SLA 计算 或可用性报告；它既不会减少问题时间，也不会扩大问题时间。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 批量更新 - 通过进入问题更新 界面来更新所选问题

要使用此选项，请先勾选相应问题前面的复选框，然后单击 批量更新按钮。

按钮

右侧的按钮提供以下选项：

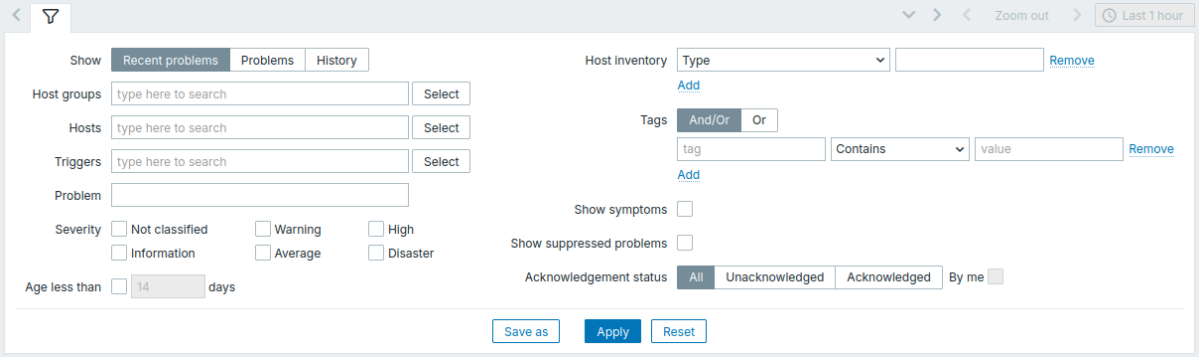
Export to CSV	将所有页面中的内容导出到 CSV 文件。

视图模式按钮是所有部分通用的，其说明请参见监控页面。

使用过滤器

您可以使用过滤器仅显示您感兴趣的问题。为了获得更好的搜索性能，数据搜索时不会解析宏。

过滤器位于表格上方。可以将常用的过滤器设置保存为选项卡，然后通过单击过滤器上方的选项卡 快速访问。

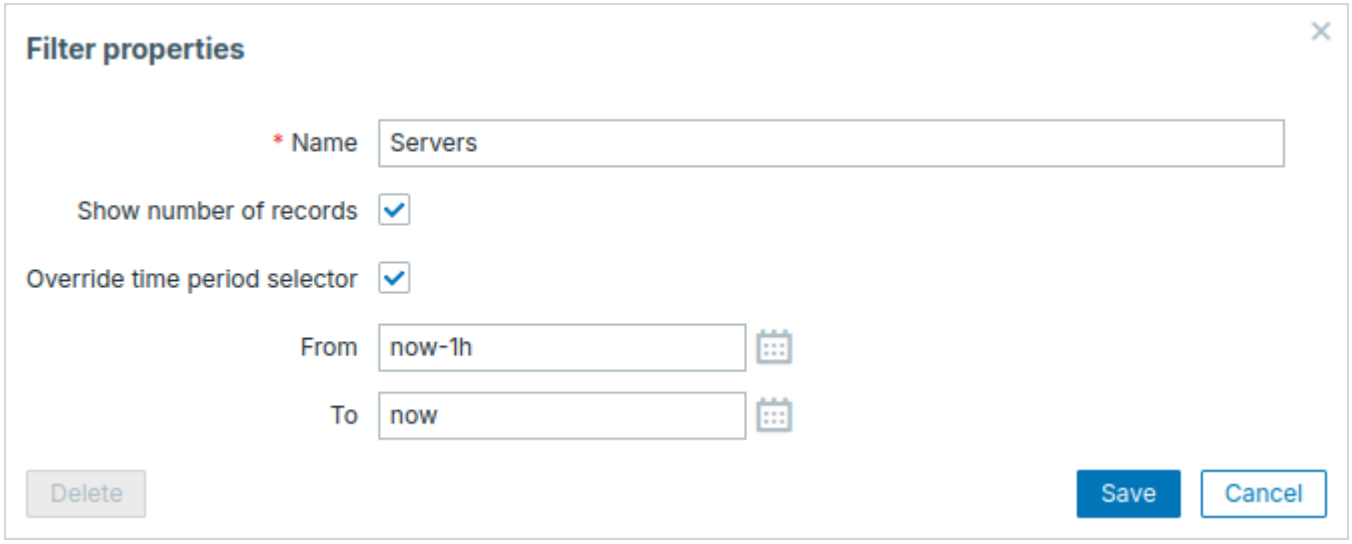


Parameter	Description
Show	按问题状态过滤： **Recent problems** - 显示未解决和最近已解决的问题（默认） **Problems** - 显示未解决的问题 **History** - 显示所有事件的历史记录
Host groups	按一个或多个主机组过滤。 指定父主机组会隐式选择所有嵌套主机组。
Hosts	按一个或多个主机过滤。
Triggers	按一个或多个触发器过滤。
Problem	按问题名称过滤。
Severity	按触发器（问题）严重性过滤。
Age less than	按问题存在时间过滤。
Host inventory	按清单类型和值过滤。
Tags	按事件标签 名称和值过滤。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含指定的标签名称，其中标签值包含输入的字符串（子串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除指定的标签名称，其中标签值包含输入的字符串（子串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足任一条件即可 过滤后，此处指定的标签将优先与问题一起显示，除非被下面的 *Tag display priority* 列表覆盖。
Show symptoms	勾选此复选框可在单独一行中显示被归类为症状的问题。
Show suppressed problems	勾选此复选框可显示原本会因主机维护而被抑制并隐藏的问题，或由用户手动抑制 的问题。
Acknowledgement status	过滤以显示所有问题、仅未确认的问题，或仅已确认的问题。勾选附加复选框可过滤掉那些曾经被您确认过的问题。

收藏过滤器的标签页

常用的过滤参数组合可以保存在标签页中。

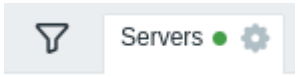
要保存一组新的过滤参数，请打开主标签页并配置过滤设置，然后点击 *Save as* 按钮。在新的弹出窗口中，定义 *Filter properties*。



Parameter	Description
Name	要在标签页列表中显示的过滤器名称。
Show number of records	如果希望在标签页名称旁显示问题数量，请勾选。
Override time period selector	勾选后可为此过滤器集设置特定的默认时间段。如果启用，只有通过更新过滤设置才能更改此标签页的时间段。对于没有自定义时间段的标签页，可以通过点击右上角的时间选择器按钮来更改时间范围（按钮名称取决于所选时间间隔：This week、Last 30 minutes、Yesterday 等）。 此选项仅适用于 *Monitoring* > *Problems* 中的过滤器。
From/To	时间段 的开始和结束时间，支持绝对时间语法（`Y-m-d H:i:s`）或相对时间语法（`now-1d`）。当勾选 *Set custom time period* 时可用。

保存后，过滤器会作为一个命名过滤器标签页创建，并立即激活。

要编辑现有过滤器的过滤器属性，请点击当前活动标签页名称旁边的齿轮图标。



注意：

- 要隐藏过滤区域，请点击当前标签页的名称。再次点击活动标签页名称可重新打开过滤区域。
- 支持键盘导航：使用方向键在标签页之间切换，按 *Enter* 打开。
- 过滤器上方的左右按钮可用于在已保存的过滤器之间切换。或者，向下箭头按钮会打开一个包含所有已保存过滤器的下拉菜单，您可以点击所需的过滤器。
- 如果已保存过滤器的设置已更改（但尚未保存），过滤器名称后会显示一个绿色圆点。要根据新设置更新过滤器，请点击 *Update* 按钮，该按钮会替代 *Save as* 按钮显示。
- 表格列布局按每个过滤器标签页分别保存。要自定义已保存过滤器的布局，请先打开该过滤器标签页，然后再自定义列。
- 每个标签页的过滤器标签页、过滤器设置以及表格列布局都会保存在用户配置文件中，并在用户再次打开该页面时恢复。

Note:
要共享过滤器，请复制活动过滤器的 URL 并发送给他人。打开此 URL 后，其他用户将能够把这组参数保存为其 Zabbix 账户中的永久过滤器。
 另请参见：页面参数。

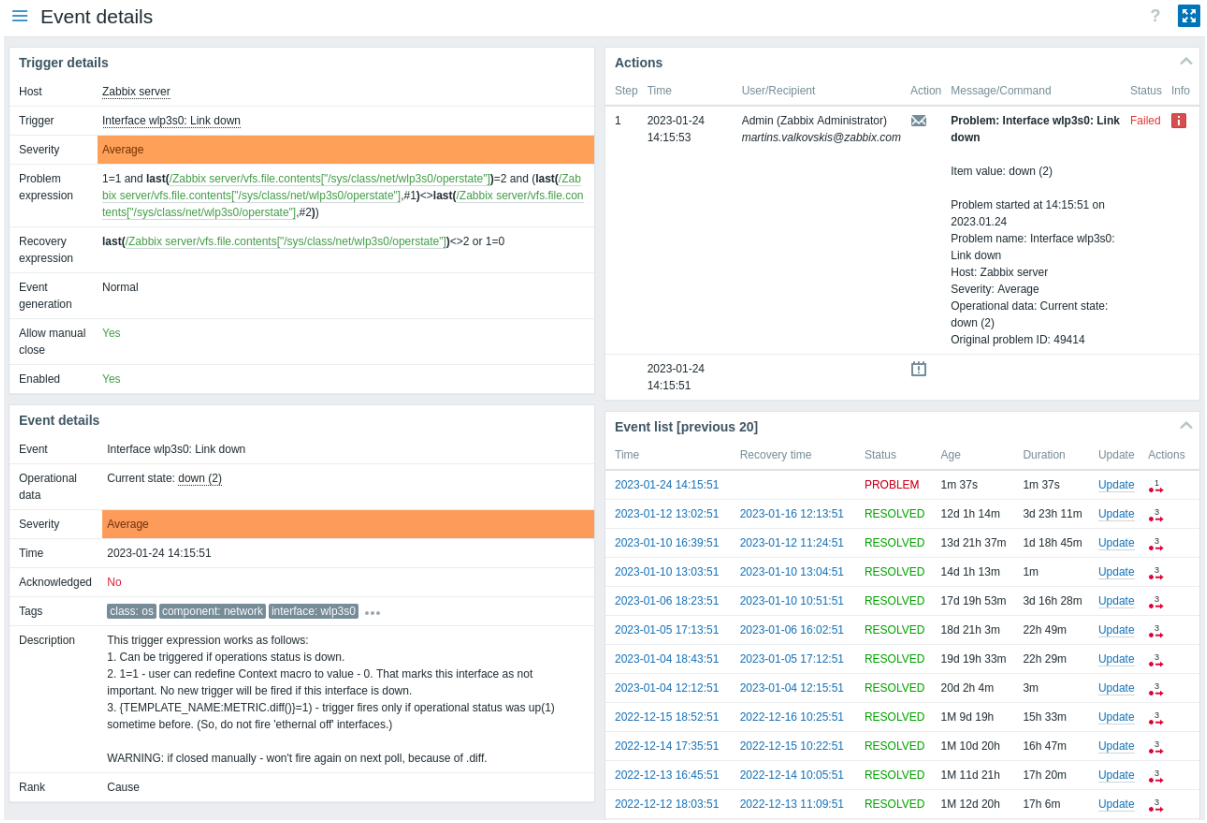
过滤器按钮

列	描述
Apply	应用指定的过滤器条件（不保存）。
Reset	重置当前过滤器并返回当前选项卡的已保存参数。在主选项卡上，这将清除过滤器。
Save as	在新选项卡中保存当前的过滤器参数。只在主选项卡上可用。

列	描述
Update	将选项卡参数替换为当前指定的参数。在主选项卡上不可用。

查看详情

Monitoring > *Problems* 中问题开始和恢复的时间是链接。点击它们会打开该事件的更多详情。



在事件详情中，你可以查看该事件是否因维护而被抑制，或者是否被用户手动抑制。

请注意，问题严重性在触发器和问题事件中可能不同 - 如果已使用 *Update problem* screen 对问题事件进行了更新。

在操作列表中，使用以下图标表示活动类型：

- - 已生成问题事件。
- - 已发送消息。
- - 问题事件已确认。
- - 问题事件未确认。
- - 已添加评论。
- - 问题严重性已提高（例如，Information > Warning）。
- - 问题严重性已降低（例如，Warning > Information）。
- - 问题严重性已更改，但已恢复到原始级别（例如，Warning > Information > Warning）。
- - 已执行远程命令。
- - 问题事件已恢复。
- - 问题已手动关闭。
- - 问题已被抑制。
- - 问题已取消抑制。
- - 问题因主机维护而被抑制。

- - 问题在主机维护后已取消抑制。
- - 问题已转换为症状问题。
- - 问题已转换为原因问题。

原因和症状问题

概述

默认情况下，所有新问题都被归类为根因问题。可以手动将某些问题重新分类为根因问题的症状问题。

例如，停电可能是某些主机无法访问或某些服务中断的实际根本原因。在这种情况下，“主机无法访问” 和 “服务中断” 问题必须被归类为 “停电”(根因问题) 的症状问题。

根因-症状层次结构仅支持两个级别。已经是症状的问题不能被指定为 “下级” 症状问题；任何被指定为症状问题的症状的问题都将成为同一个根因问题的症状。

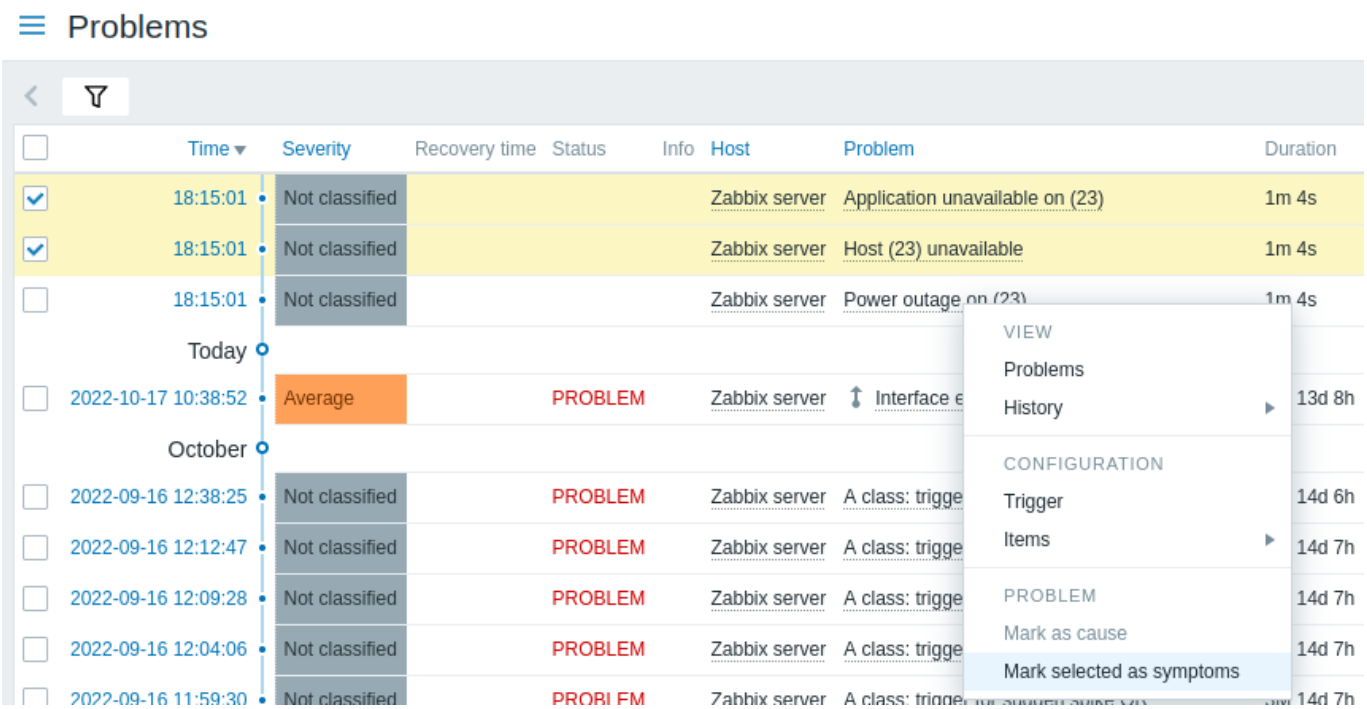
只有根因问题才会被计入地图、仪表盘小部件，分类可按照问题严重等级或问题主机中的问题总数划分。但是，问题排名不会影响服务。

一个症状问题只能与一个根因问题相关联。如果根因问题已解决或关闭，症状问题不会自动解决。

配置

要将问题重新分类为症状问题，首先在问题 列表中选中它。可以选择一个或多个问题。

然后转到原因问题，并在其上下文菜单中单击 *Mark selected as symptoms* 选项。



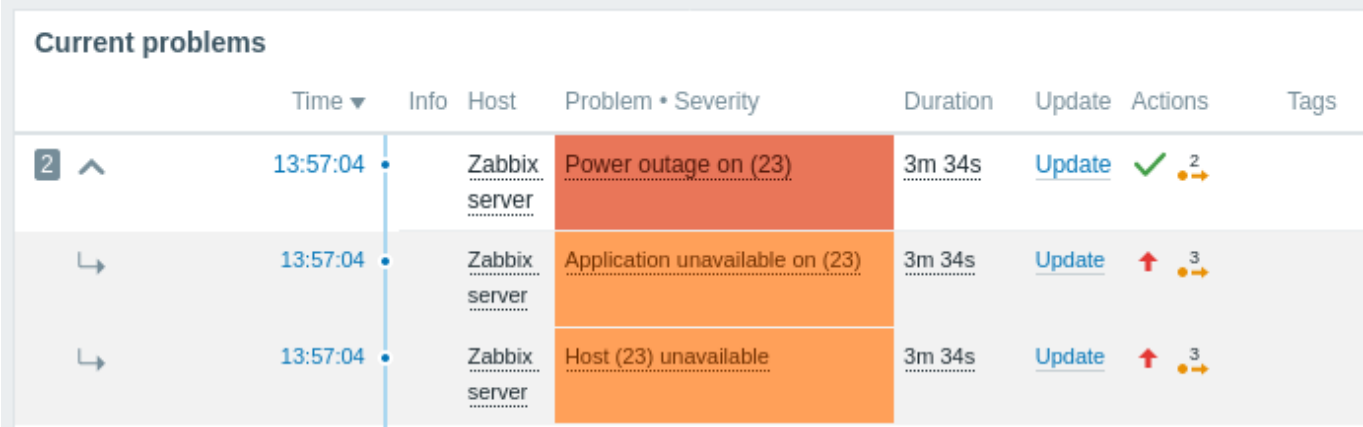
之后，所选问题将由服务器更新为该原因问题的症状问题。

在问题状态更新期间，它会以以下两种方式之一显示：

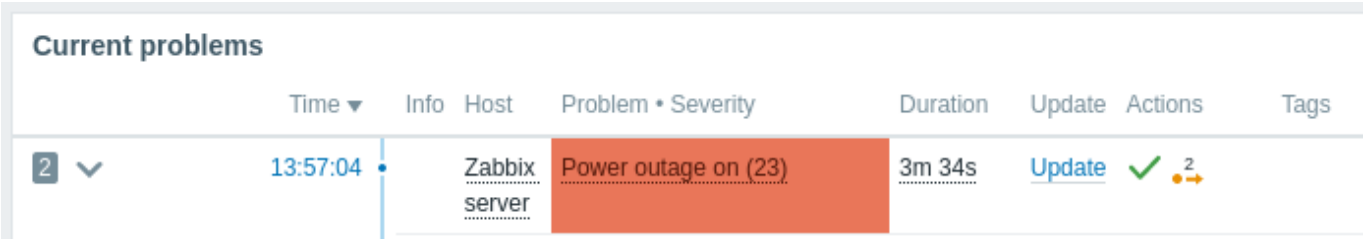
- 在 Status 列中显示闪烁的”UPDATING” 状态；
- 在 Info 列中显示闪烁的 或 图标（当筛选器中仅选择了 *Problems*，因此不显示 Status 列时，此方式生效）。

显示

症状问题会显示在原因问题下方，并在 *Monitoring* > *Problems*（以及 *Problems* 仪表板小部件）中相应标记 - 使用图标、更小的字体和不同的背景。



在折叠视图中，只能看到原因问题；症状问题的存在会通过行首的数字以及用于展开视图的图标来表示。



也可以将症状问题以正常字体并单独一行的方式额外显示。为此，请在过滤器设置或小部件配置中选择 *Show symptoms*。

恢复为原因问题

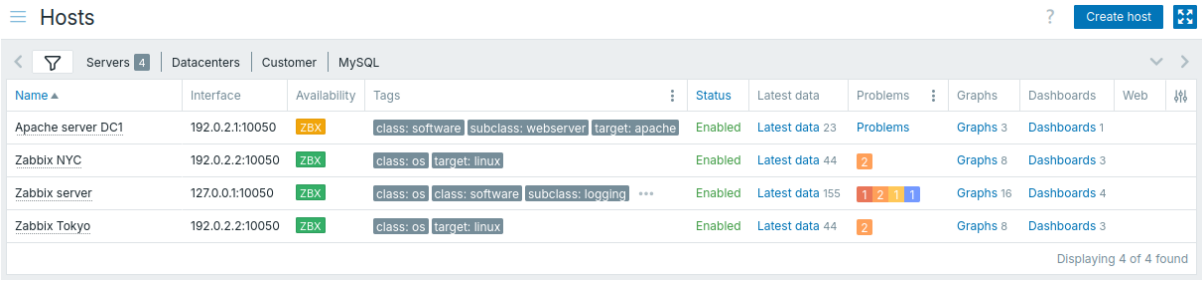
症状问题可以恢复为原因问题。要执行此操作，可以：

- 在症状问题的上下文菜单中点击 标记为原因选项；
- 在问题更新 界面中勾选 转换为原因选项，然后点击 更新（如果选择了多个问题，此选项也同样适用）。

主机

概述

Monitoring > Hosts 部分显示受监控主机的完整列表，其中包含主机接口、可用性、标签、当前问题、状态（启用/禁用），以及可轻松跳转到主机最新数据、问题历史、图形、仪表板和 web 场景的链接。



主机详细信息以可自定义的表格显示。表格标题中的 *Customize table* 按钮可让你显示、隐藏或重新排序列，并将布局重置为默认值。你也可以通过拖动列右边框来调整每列宽度；双击边框可重置。所有自定义设置都会立即生效，并始终为当前用户在活动的filter 选项卡（默认或已保存）中记住。

可用的列如下：

Column	Description
Name	显示的主机名称。点击名称会打开host menu。 名称后面的橙色扳手图标 表示该主机处于维护中。 点击列标题可按名称升序或降序对主机排序。
Interface	显示主机的主接口。

Column	Description
Availability	显示按已配置接口划分的主机可用性。 可用性图标表示主机接口在 Zabbix 服务器上的当前状态。因此，如果你在前端中禁用主机，其可用性会在 Zabbix 服务器同步配置更改后更新。同样，如果你启用主机，其可用性会在 Zabbix 服务器同步配置更改并轮询主机后更新。 可用性图标仅表示已配置的接口类型（Agent、SNMP、IPMI、JMX）。 将鼠标悬停在图标上会显示一个弹出窗口，其中列出同类型的所有接口及其详细信息、状态和错误。对于 Agent 接口，弹出窗口会显示接口（被动）和主动检查。如果主机仅配置了主动检查，即使没有配置 Agent 接口，也会显示 Agent 接口图标。 对于没有接口的主机，该列为空。 单个主机接口的状态由使用该接口的已启用监控项与主机之间的连接决定。状态可以是： **Available** - 与主机的连接成功； **Not available** - 与主机的连接失败（超时、防火墙问题等）； **Unknown** - 尚未尝试连接主机，或结果未知。 有关 Zabbix 服务器如何确定接口状态的更多详情，请参见Unknown interface status 和Unreachable/unavailable host interface settings。 单一类型（Agent、SNMP、IPMI、JMX）的所有主机接口状态由至少被一个已启用监控项使用的这些接口决定。状态由图标颜色表示： **Green** - 所有接口均可用； **Yellow** - 至少有一个接口不可用，且至少有一个接口可用或未知； **Red** - 所有接口均不可用； **Gray** - 至少有一个接口未知，但没有不可用的接口。 主动检查可用性。如果主机上至少启用了一个主动检查，则主动检查也会影响上述 Agent 接口的总体可用性。为确定主动检查可用性，心跳消息会在 agent 主动检查线程中发送。心跳消息的频率由 Zabbix agent 或agent 2 配置中的 `HeartbeatFrequency` 参数控制（默认 60 秒，范围 0-3600）。当主动检查心跳的时间早于 2 x HeartbeatFrequency 秒时，主动检查会被视为不可用。 注意： 6.2.x 之前版本的 Zabbix agent 不会发送主动检查心跳，因此其主机的可用性仍显示为未知。
Tags	显示主机tags。 将鼠标悬停或点击显示标签后面的 ••• 三点图标，会打开一个包含所有标签的弹出窗口。 点击列标题中的 *Options* 按钮，可设置显示的标签数量（1–3）、选择标签名称的显示方式（完整、缩写或不显示）、设置标签显示优先级，或复制该列（复制后的列可重命名或删除）。
Tag value	显示主机tag 值。 点击列标题中的 *Options* 按钮，可重命名该列、选择要显示的 tag 值，或复制该列（复制后的列可重命名或删除）。
Status	主机状态 - *Enabled* 或 *Disabled*。 点击列标题可按状态升序或降序对主机排序。
Latest data	点击链接将打开 *Monitoring* > *Latest data* 部分，显示从主机收集到的所有最新数据。 显示有最新数据的监控项数量，使用灰色字体。
Problems	显示主机当前未解决问题的数量，并按严重性排序。 每个问题图标的颜色表示严重性，数字表示该严重性的问题数量。点击图标会打开该主机的 *Monitoring* > *Problems*；如果主机没有问题，则会显示 *Problems* 链接。 点击列标题中的 *Options* 按钮，可使用 *Show suppressed problems* 选项，该选项会包含原本会因主机维护或单个problem suppression 而被抑制（隐藏）的问题。
Graphs	点击链接将显示为主机配置的图形。显示图形数量，使用灰色字体。 如果主机没有图形，则不显示链接。
Dashboards	点击链接将显示为主机配置的仪表板。显示仪表板数量，使用灰色字体。 如果主机没有仪表板，则不显示链接。
Web	点击链接将显示为主机配置的 web 场景。显示 web 场景数量，使用灰色字体。 如果主机没有 web 场景，则不显示链接。
Custom text	显示自定义文本、宏值，或两者的组合。 点击列标题中的 *Options* 按钮，可重命名该列、输入自定义文本或宏，或复制该列（复制后的列可重命名或删除）。 Supported macros: `{HOST.*}`（除 `{HOST.METADATA}` 外）、`{INVENTORY.*}`，以及user macros。

按钮

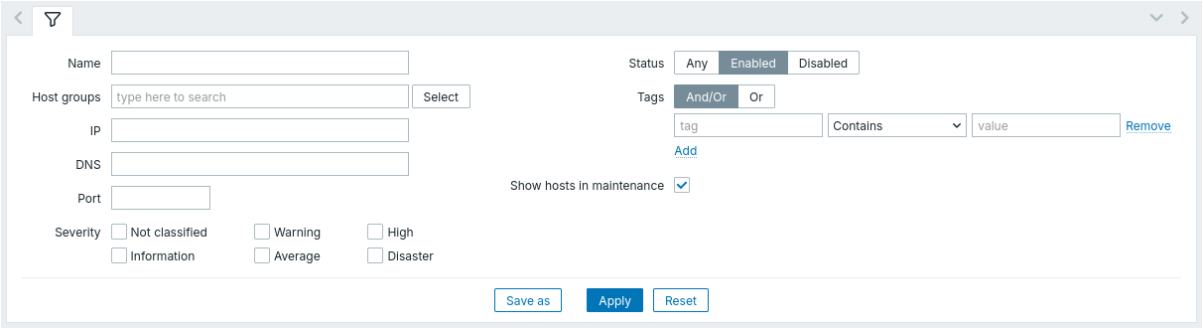
创建主机允许您创建新主机。此按钮仅对 Admin 和 Super Admin 用户可用。

所有部分通用的查看模式按钮在监控页面中进行了介绍。

使用过滤器

您可以使用过滤器仅显示您感兴趣的主机。为了获得更好的搜索性能，数据搜索时不会解析宏。

过滤器位于表格上方。可以按以下条件过滤主机：名称、主机组、IP 或 DNS、接口端口、标签、问题严重性、状态（启用/禁用/任意）；您还可以选择是否显示当前处于维护中的主机。



Parameter	Description
Name	按可见主机名称过滤。
Host groups	按一个或多个主机组过滤。 指定父主机组会隐式选择所有嵌套主机组。
IP	按 IP 地址过滤。
DNS	按 DNS 名称过滤。
Port	按端口号过滤。
Severity	按问题严重性过滤。默认显示所有严重级别的问题。未被抑制的问题会显示。
Status	按主机状态过滤。
Tags	按主机标签名称和值过滤。可以按主机级别标签以及所有已链接模板中的标签进行过滤，包括嵌套模板。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含标签值中含有输入字符串的指定标签名称（子串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除标签值中含有输入字符串的指定标签名称（子串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足其中一个条件即可
Show hosts in maintenance	勾选复选框以显示处于维护中的主机（默认显示）。

保存过滤器

收藏的过滤器设置可以保存为选项卡，然后通过单击过滤器上方相应的选项卡快速访问。

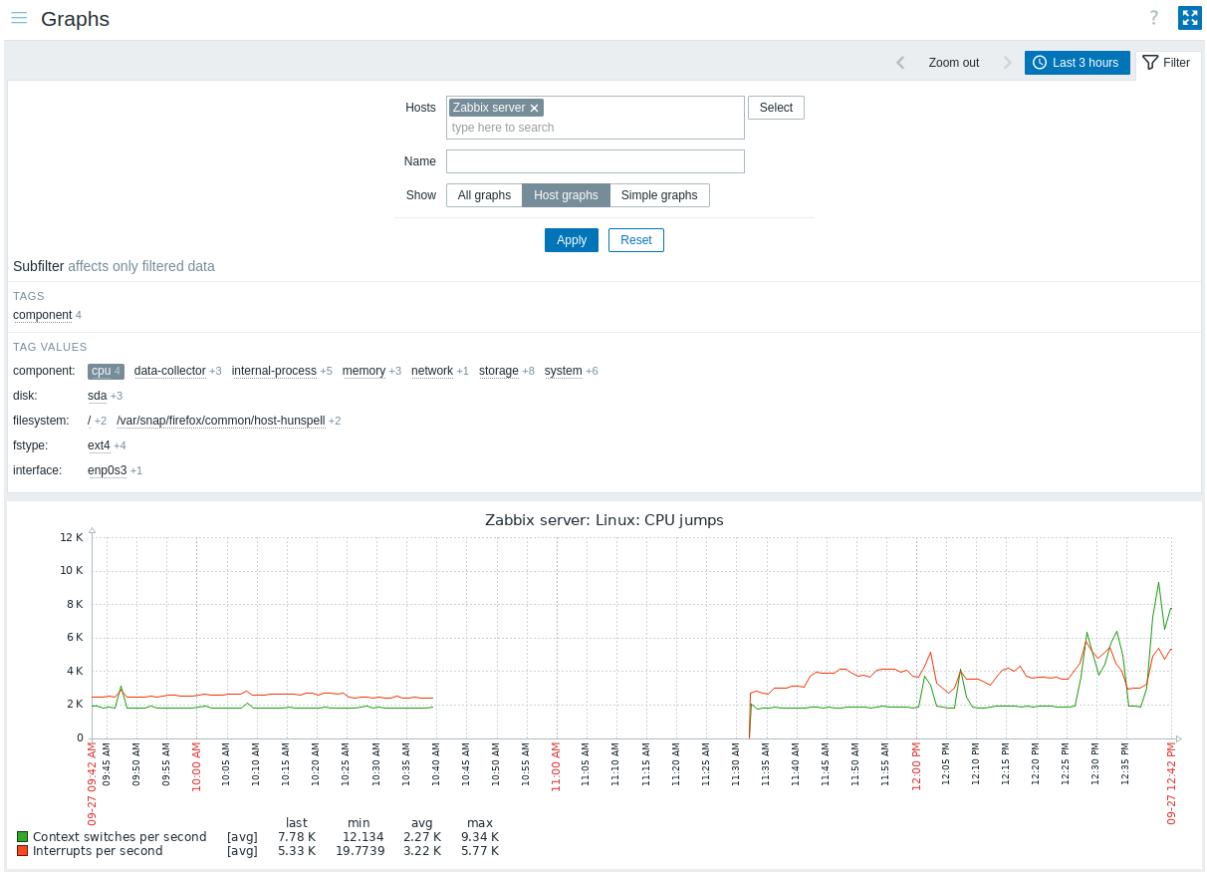
有关保存过滤器的更多详细信息，请参见。

图形

概述

可以通过 *Monitoring* > *Hosts*，点击相应主机的 Graphs 来访问主机图形。

已为主机配置的任何自定义图形 都可以显示，同时也可以显示任何简单图形。



图形按以下方式排序：

- 图形名称（自定义图形）
- 监控项名称（简单图形）

已禁用主机的图形也可以访问。

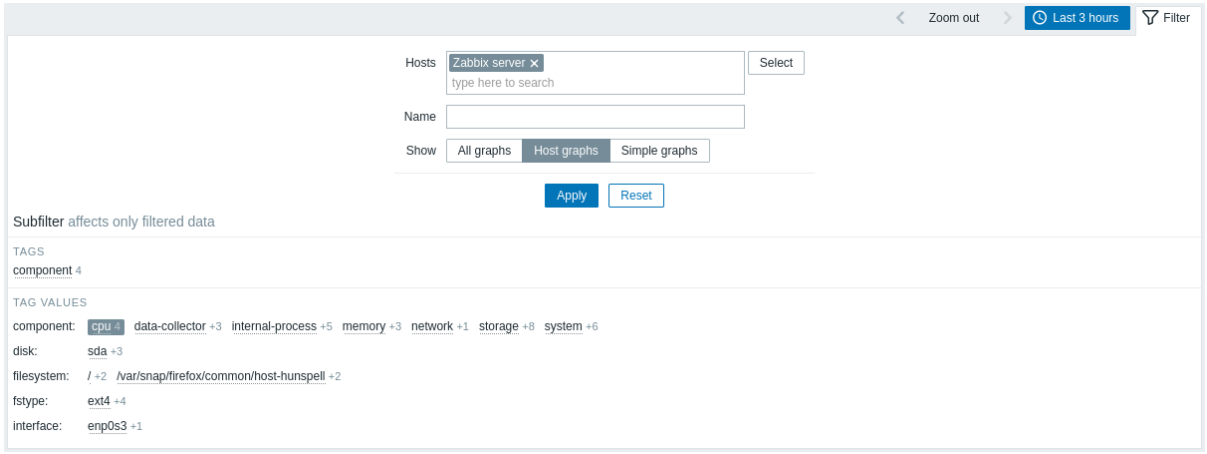
时间段选择器

图表上方的时间段选择器。允许通过鼠标单击选择经常需要的时间段。

另请参见：时间段选择器

使用筛选器

要查看特定图形，请在筛选器中选择该图形。筛选器允许您指定主机、图形名称和 显示选项（全部/主机图形/简单图形）。



如果未在筛选器中选择主机，则不会显示任何图形。

使用子过滤器

子过滤器允许进一步修改主过滤器应用的筛选条件。

其中包含可点击的链接，可快速访问相关图形。图形通过共同的实体（标签名称或值）建立关联。单击标签名称/值后，该标签名称/值会以灰色背景突出显示，并且图形会立即进行筛选（无需在主过滤器中单击应用）。单击其他标签名称/值会将其添加到筛选结果中。再次单击该标签名称/值会取消相应的筛选。

子过滤器基于筛选后的数据生成，而筛选后的数据最多包含 1000 条记录。如果希望在子过滤器中查看更多记录，需要增大搜索和过滤结果限制参数的值（位于管理 > 常规 > 前端）。

与主过滤器不同，子过滤器会随每次表格刷新请求一起更新，以始终获取可用筛选选项及其计数的最新信息。

水平显示的实体数量最多为 100 个。如果超过此数量，末尾会显示三点图标，但该图标不可点击。垂直列表（例如包含其值的标签）最多显示 20 个条目。如果超过此数量，会显示三点图标，但该图标不可点击。

每个可点击实体旁的数字表示主过滤器结果中包含该实体的图形数量。

选择一个实体后，其他可用实体旁的数字会带有加号，表示可以添加到当前选择中的图形数量。

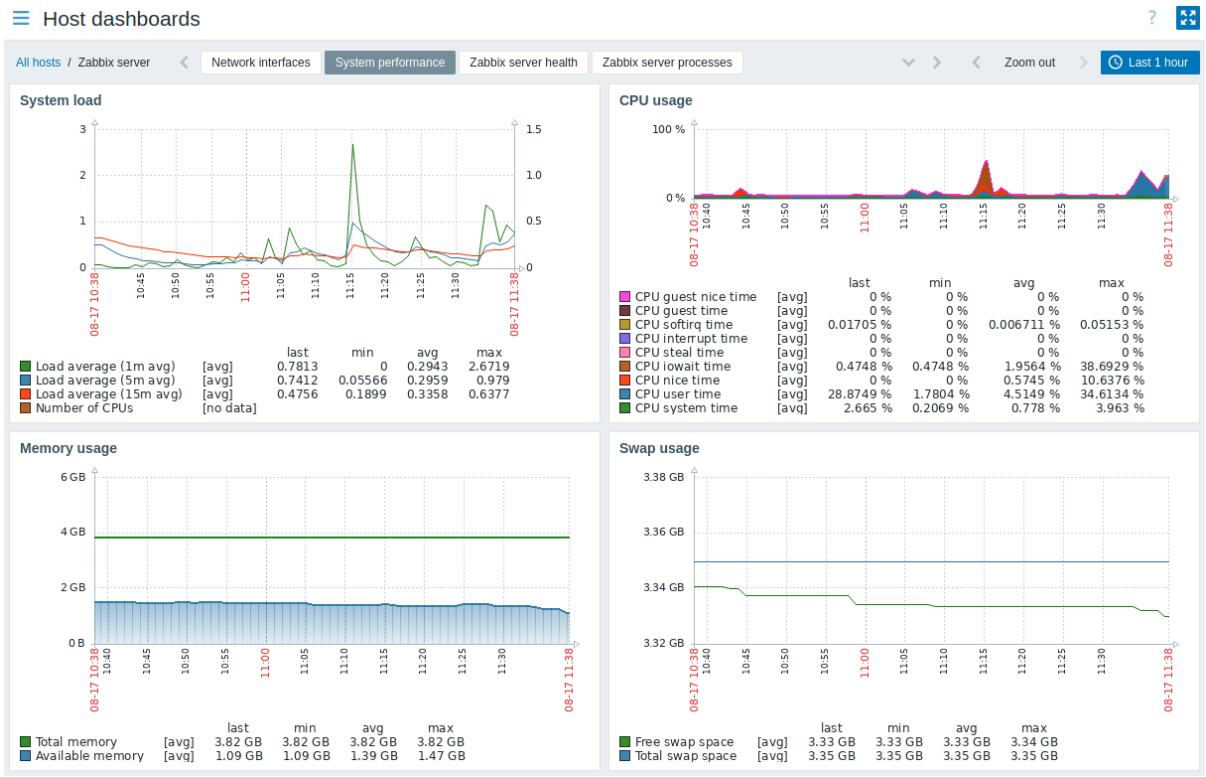
按钮

所有部分通用的视图模式按钮已在监控页面中说明。

主机仪表板

概述

主机仪表板与全局仪表板类似；但是，主机仪表板没有所有者，并且仅显示所选主机的数据。



查看主机仪表板时，可以通过以下方式在已配置的仪表板之间切换：

- 仪表板选项卡；
- 标题下方的  箭头按钮；
- 标题下方的  箭头按钮，该按钮会显示所有可用主机仪表板的完整列表。

要切换到 *Monitoring* > *Hosts* 部分，请点击左上角标题下方的 *All hosts* 导航链接。

配置

主机仪表盘在模板级别配置。一旦模板链接到主机，就会为该主机生成主机仪表盘。请注意，主机仪表盘无法在全局仪表盘部分配置，该部分是为全局仪表盘保留的。

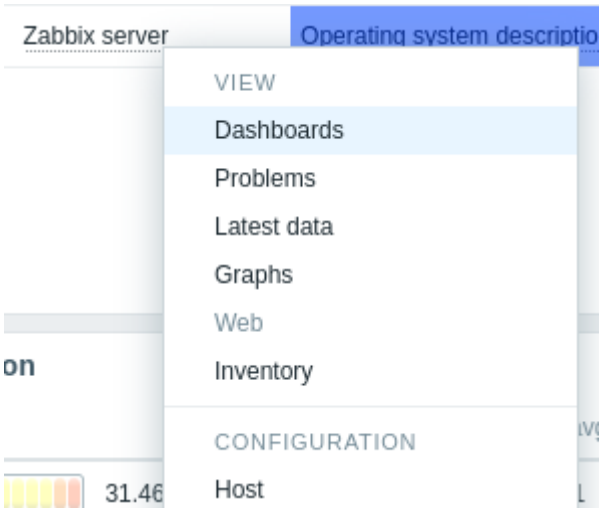
主机仪表盘的组件只能在模板 级别配置，除了更改刷新间隔。此外，主机仪表盘的组件只能复制到同一模板内的其他主机仪表盘。请注意，全局仪表盘的组件不能复制到主机仪表盘。

Note:
在 Zabbix 5.2 之前，主机仪表盘曾是主机屏幕。导入包含屏幕的旧模板时，屏幕导入将被忽略。

访问

可以通过以下方式访问主机仪表板：

- 在全局搜索 中搜索主机名称后（点击搜索结果中提供的 *Dashboards* 链接）；
- 在 *Inventory* > *Hosts* 中点击主机名称后（点击主机概览中提供的 *Dashboards* 链接）；
- 通过主机菜单 点击 *Dashboards*。

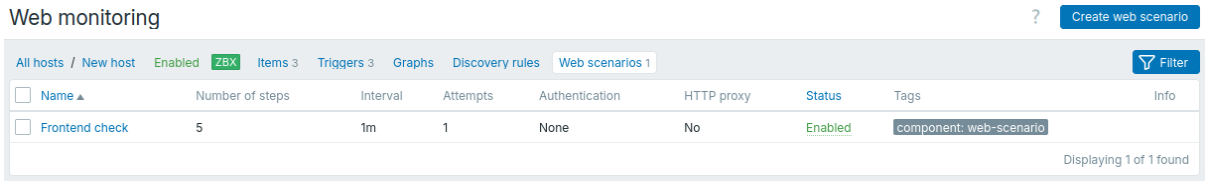


请注意，主机仪表板 不能直接在 *Dashboards* 部分访问，因为该部分保留给全局仪表板。

Web 场景

概述

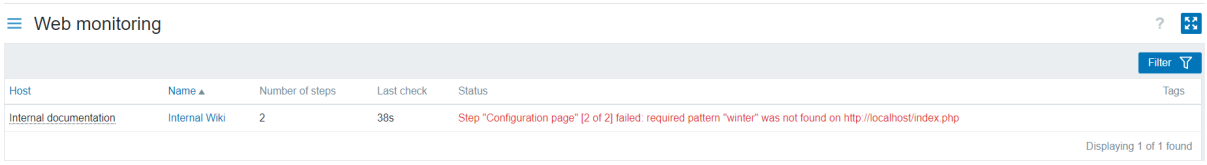
可通过 监控 > 主机，点击相应主机的 Web 来访问主机web scenario 信息。



点击主机名称会打开主机菜单。已禁用主机的数据也可以访问。已禁用主机的名称会以红色显示。

web scenario 的状态可以是：

- OK（绿色）- 所有步骤都已成功，且 Zabbix 已收集到该 scenario 的数据。
- Error（红色）- 一个或多个步骤失败（请查看错误消息，了解哪些参数可能需要修正）。
- Unknown（灰色）- 目前还没有可用的步骤状态，或者尚未收集到任何数据。



每页显示的 scenario 最大数量取决于用户配置文件中的 每页行数设置。

默认情况下，仅显示最近 24 小时内的值。引入此限制的目的是提升大型最新数据页面的初始加载速度。您可以通过在管理 > 常规 > *GUI* 菜单部分中修改 最大历史显示周期参数的值来延长此时间范围。

scenario 名称是指向其更详细统计信息的链接：



使用过滤器

该页面显示所选主机的所有 web 场景列表。要查看其他主机或主机组的 web 场景，而无需返回 *Monitoring* > *Hosts* 页面，请在过滤器中选择该主机或主机组。您还可以根据标签筛选场景。

按钮

所有分区通用的视图模式按钮已在监控 页面中说明。

最新数据

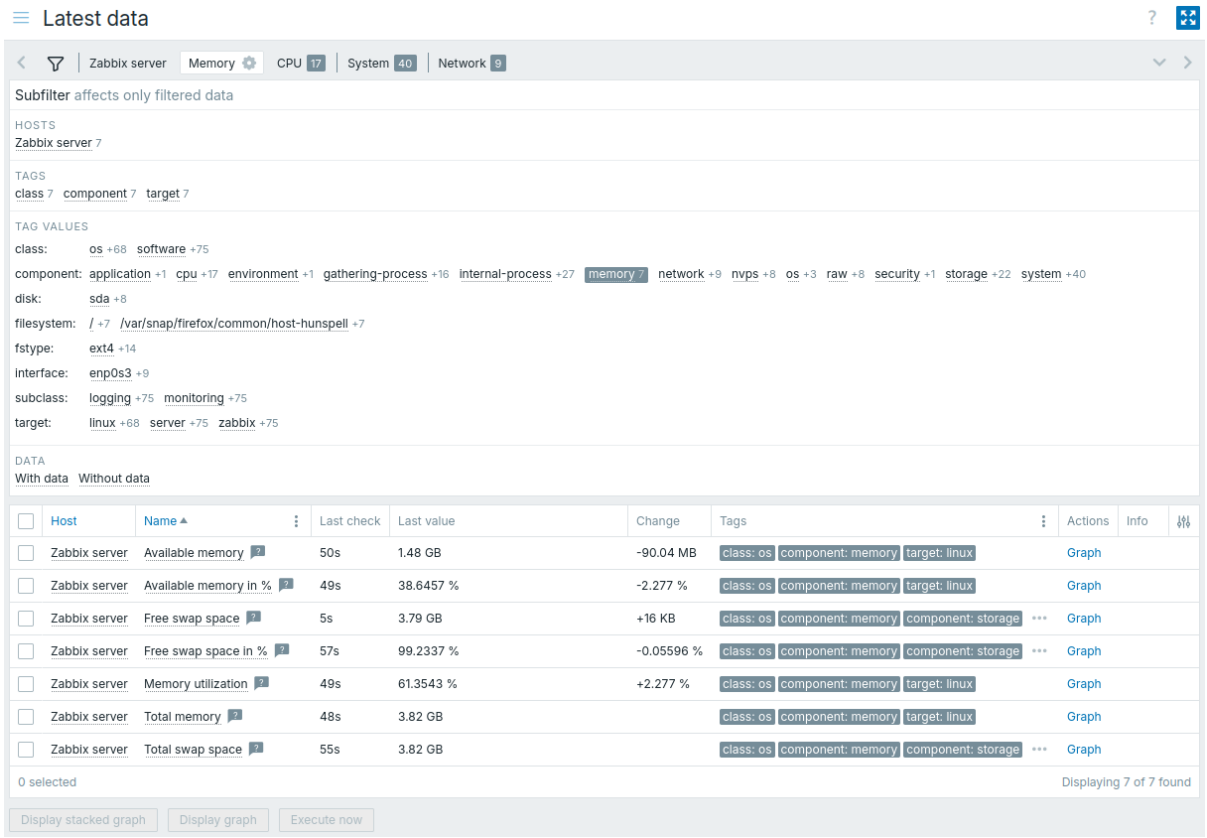
概述

监控 > 最新数据部分显示由监控项收集到的最新值。

此部分包含以下元素：

- 过滤器
- 子过滤器
- 监控项列表

Note:
仅当已设置过滤器且有可显示的结果时，才会显示子过滤器和监控项列表。



最新值详情会显示在一个可自定义的表格中。表头中的 自定义表格按钮可让你显示、隐藏或重新排序列，并将布局重置为默认值。你还可以通过拖动每列的右边框来调整列宽；双击边框可重置。所有自定义设置都会立即生效，并且始终会为当前用户在活动的过滤器 选项卡（默认或已保存）中保留。

可用列如下：

Column	Description
主机	监控项所属的主机名称。 点击名称会打开主机上下文菜单。 如果主机处于维护中，主机名称后会显示一个橙色扳手 图标。 如果主机已禁用，主机名称会以红色显示。请注意，已禁用主机的数据（包括图形和监控项值列表）仍可在 最新数据部分访问。
名称	监控项名称。 点击名称会打开监控项菜单。 对于所有带有描述的监控项，监控项名称旁都会显示一个问号 图标。将鼠标悬停在该图标上可显示包含监控项描述的工具提示。 点击列标题中的 选项按钮，可使用 显示监控项键值选项，在每个监控项名称下方显示监控项键值。
间隔	监控项的更新间隔。
历史	监控项的历史数据保存周期。
趋势	监控项的趋势数据保存周期。
类型	监控项的类型。
上次检查	距上次监控项检查的时间。
最新值	监控项的最新值。 值会应用单位转换和数值映射后显示。将鼠标悬停在该值上可显示原始数据（截断为 8192 个字符）。 二进制值会显示为占位符（*binary value*），而不是实际值。 默认情况下，仅显示最近 24 小时内接收到的值。此限制可提升大型最新数据页面的初始加载速度；如需延长该时间范围，请在 管理 > 一般 > *GUI* 中更新 最大历史数据显示周期参数的值。
变化	前一个值与最新值之间的差异。 对于更新频率为 1 天或更长的监控项，在默认设置下将不会显示变化量。在这种情况下，如果最新值是在 24 小时前接收的，则该值将完全不显示。
标签	与监控项关联的标签。 点击某个标签会在子过滤器 中启用它，从而按该标签（以及子过滤器中选择的任何其他标签）筛选监控项列表。点击后，此类标签将不再可点击；如需调整按标签筛选的条件，必须使用子过滤器。 点击列标题中的 选项按钮，可设置显示的标签数量（1-3）、选择标签名称的显示方式（完整、缩短或不显示）、设置标签显示优先级，或复制该列（复制后的列可重命名或删除）。

Column	Description
标签值	显示监控项标签 的值。 点击列标题中的 选项按钮，可重命名该列、选择要显示的标签值，或复制该列（复制后的列可重命名或删除）。
操作	指向监控项值的简单图形/历史 的链接。
信息	关于监控项的附加信息。 如果监控项存在错误（例如已变为不受支持），会显示信息 图标。将鼠标悬停在该图标上可查看详细信息。
自定义文本	显示自定义文本、宏值或两者的组合。 点击列标题中的 选项按钮，可重命名该列、输入自定义文本或宏，或复制该列（复制后的列可重命名或删除）。 支持的宏：{HOST.*}（不包括 {HOST.METADATA}）、{INVENTORY.*}、{ITEM.*}（不包括 {ITEM.STATE.ERROR}），以及用户宏。

按钮

所有部分通用的视图模式按钮已在监控页面中说明。

批量操作

列表下方的按钮提供对一个或多个已选监控项的批量操作：

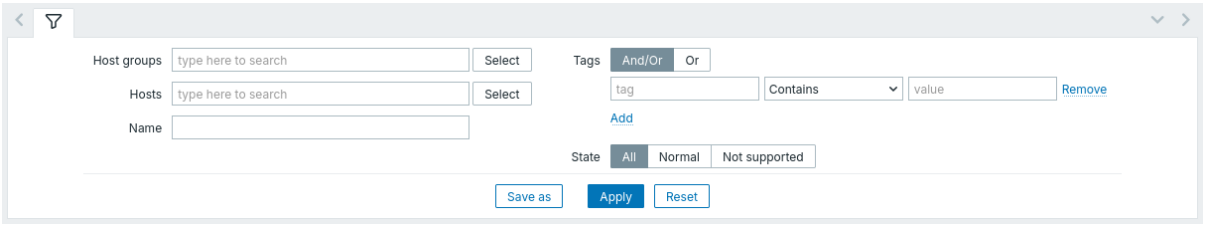
- 显示堆叠图 - 显示堆叠式临时图形。
- 显示图形 - 显示简单的临时图形。
- 立即执行 - 立即执行一次新监控项值检查。仅支持被动检查（请参见更多详情）。此选项仅适用于具有读写权限的主机。对于只读权限的主机，是否可以访问此选项取决于用户角色 中名为 在只读主机上调用 “立即执行” 的选项。

要使用这些选项，请先勾选相应监控项前的复选框，然后点击所需按钮。

使用筛选器

您可以使用筛选器，仅显示感兴趣的监控项。为提高搜索性能，搜索时不会解析宏。

筛选器  图标位于监控项列表和子筛选器上方。单击该图标可展开筛选器。



筛选器允许您按主机组、主机、监控项名称、标签和状态缩小监控项列表范围。在筛选器中指定父主机组后，会隐式选择所有嵌套的主机组。有关按标签筛选的详细信息，请参阅 监控 > 问题。

显示详细信息筛选器选项允许展开监控项显示的信息。选中该选项后，将显示监控项刷新间隔、历史记录和趋势设置、监控项类型以及监控项错误（正常/不支持）等详细信息。

保存筛选器

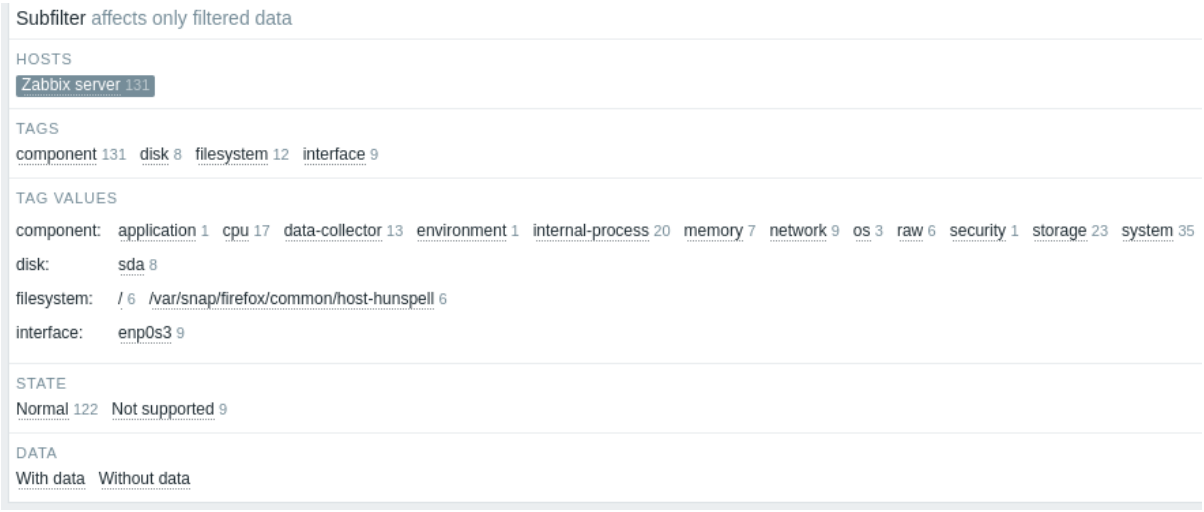
收藏筛选器设置可以保存为选项卡，然后通过单击筛选器上方相应的选项卡快速访问。

有关保存筛选器的更多详细信息。

使用子筛选器

子筛选器允许您进一步修改主筛选器中的筛选条件。

子筛选器包含可点击的链接，可快速访问相关监控项。监控项通过共同的实体进行关联，例如主机、标签名称或值、监控项状态或数据状态。单击某个实体后，该实体会以灰色背景突出显示，并立即对监控项进行筛选（无需在主筛选器中单击应用）。单击其他实体会将其添加到筛选结果中。再次单击该实体可取消相应的筛选。



子筛选器基于筛选后的数据生成，而筛选后的数据最多包含 1000 条记录。如果有 20 个主机，每个主机包含 100 个监控项（总计 2000 条记录），则子筛选器中只会显示其中一半的主机。如果希望在子筛选器中查看更多记录，需要增大搜索和筛选结果限制参数的值（位于管理 > 常规 > *GUI*）。

与主筛选器不同，子筛选器会在每次刷新表格时更新，以始终显示最新的可用筛选选项及其计数。

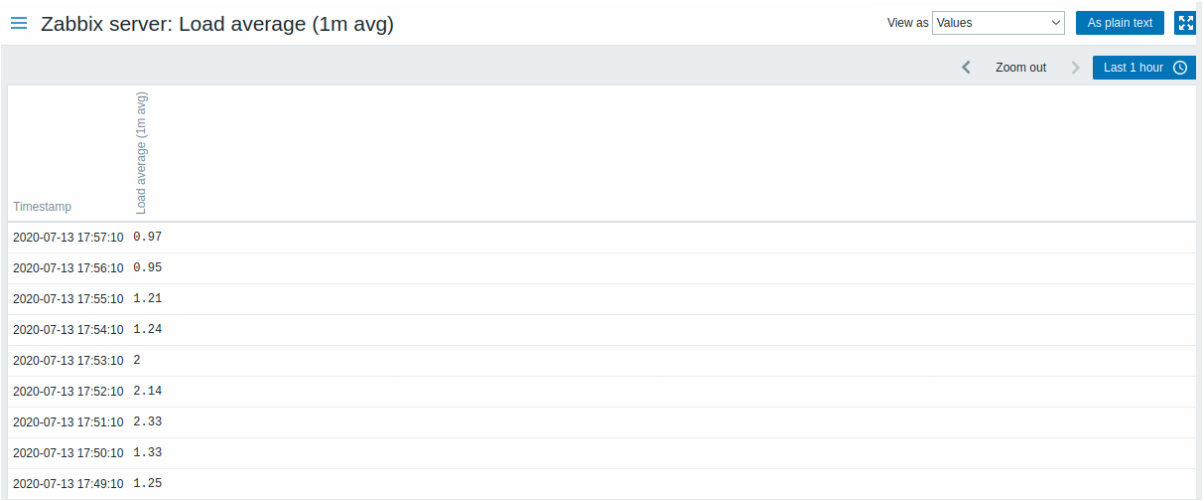
对于每个实体组（主机、标签、标签值等），最多显示 10 行实体。如果实体数量更多，可以单击列表末尾的三个点 ••• 图标，将列表展开为最多显示 1000 个条目（前端定义中的 SUBFILTER_VALUES_PER_GROUP 值）。对于标签值，列表最多可以展开显示 200 个标签名称及其对应的值。请注意，列表完全展开后无法折叠。

每个可点击实体旁边的数字表示其中包含的监控项数量（基于主筛选器的结果）。单击某个实体后，其他可用实体旁边的数字会带有加号，表示可以添加到当前选择中的监控项数量。不包含监控项的实体不会显示，除非之前已在子筛选器中选中。

图表和历史记录

监控项列表中的 *Graph/History* 列提供以下链接：

- **History** - 适用于所有文本类监控项，链接到列表（*Values/500 latest values*），显示之前监控项值的历史记录。
- **Graph** - 适用于所有数值类监控项，链接到简单图形。请注意，当图形显示时，右上角的下拉菜单还提供切换到 *Values/500 latest values* 的选项。



此列表中显示的值为原始值，即未应用任何后处理。

Note:
显示的值总数由 *Limit for search and filter results* 参数的值决定，该参数在 *Administration* > *General* > *GUI* 中设置。

地图

概述

在 *Monitoring* > *Maps* 部分，您可以配置、管理和查看网络地图。

打开此部分时，您将看到您上次访问的地图，或者您有权限访问的所有地图列表。

所有地图都可以是公开的或私有的。公开地图对所有用户可用，而私有地图仅对其所有者以及与该地图共享的用户可访问。

地图列表



显示的数据：

Column	Description
Name	地图名称。单击名称可查看该地图。
Width	显示地图宽度。
Height	显示地图高度。
Actions	提供两个操作： **Properties** - 设置地图的常规属性 **Edit** - 进入网格以添加地图元素

要配置新地图，请单击右上角的 *Create map* 按钮。要从 YAML、XML 或 JSON 文件导入地图，请单击右上角的 *Import* 按钮。导入地图的用户将被设置为其所有者。

列表下方的两个按钮提供一些批量编辑选项：

- *Export* - 将地图导出为 YAML、XML 或 JSON 文件
- *Delete* - 删除地图

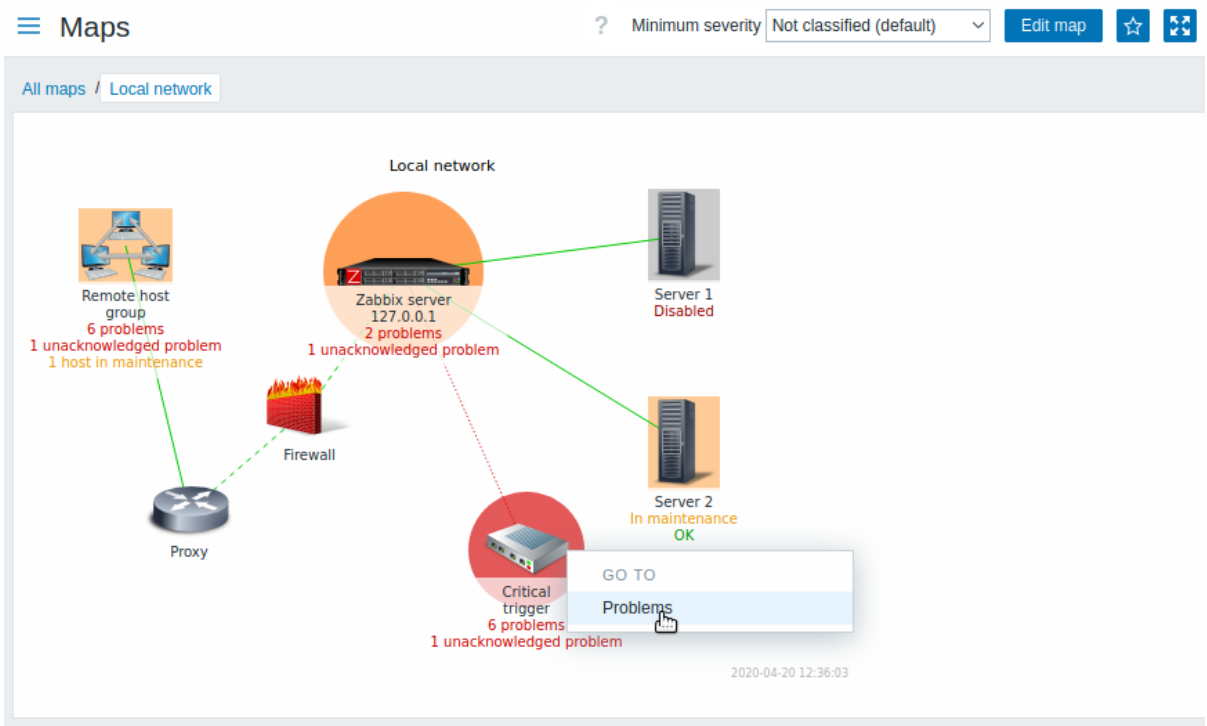
要使用这些选项，请先勾选相应地图前面的复选框，然后单击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的拓扑图。为了获得更好的搜索性能，使用未解析的宏搜索数据。

查看地图

要查看地图，请在所有地图列表中单击其名称。



您可以使用地图标题栏中的下拉菜单选择要显示的问题触发器的最低严重性级别。标记为 *default* 的严重性是地图配置中设置的级别。如果地图包含子地图，导航到子地图时将保留上一级地图的严重性级别（除非它是 *Not classified*，在这种情况下，不会将其传递给子地图）。

图标高亮

如果地图元素处于问题状态，则会以圆形高亮显示。圆形的填充颜色对应于问题的严重性颜色。只有达到或高于所选严重性级别的问题，才会与该元素一起显示。如果所有问题都已确认，则会在圆形周围显示粗绿色边框。

此外：

- 处于维护状态的主机会以橙色实心方块高亮显示。请注意，如果地图元素是主机，维护状态高亮的优先级高于问题严重性高亮。
- 已禁用（未监控）的主机会以灰色实心方块高亮显示。

如果在地图配置中选中了图标高亮复选框，则会显示高亮效果。

最近更改标记

元素周围指向内侧的红色三角形表示触发器状态最近发生了更改，即更改发生在过去 30 分钟内。

如果在地图配置中选中了在触发器状态更改时标记元素复选框，则会显示这些三角形。

链接

单击地图元素会打开一个菜单，其中包含一些可用链接。单击主机名称会打开主机菜单。

按钮

右侧的按钮提供以下选项：

Edit map	转到地图内容编辑。
	将地图添加到仪表板 中的收藏夹小组件。
	该地图已在仪表板 中的收藏夹小组件内。单击可将地图从收藏夹小组件中移除。

所有部分通用的查看模式按钮请参见监控 页面。

映射中的可读摘要

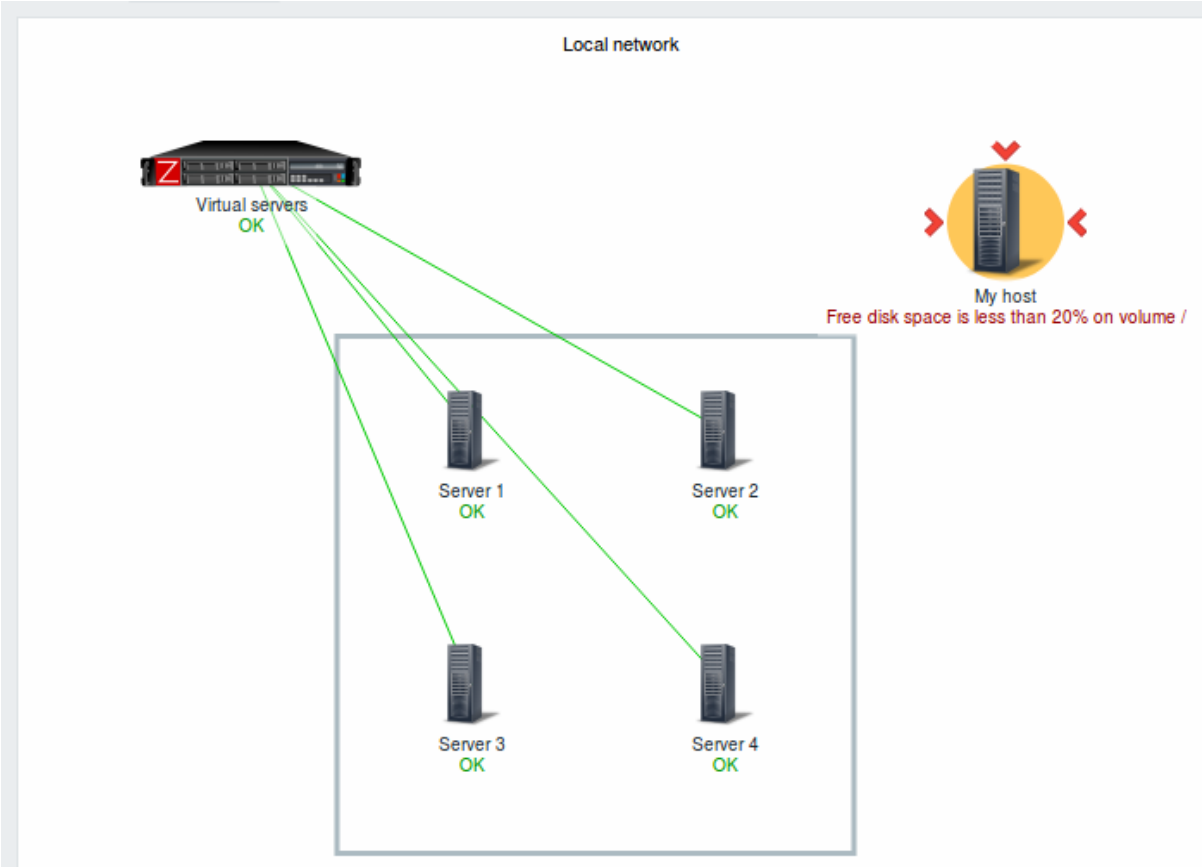
系统提供了隐藏的 “aria-label” 属性，可使用屏幕阅读器读取映射信息。可读取常规映射描述和各个元素的描述，格式如下：

- 映射描述：< 映射名称 >，< 处于问题状态的元素数 >/< 元素总数 >，共 < 问题数 > 个问题。
- 描述存在一个问题的元素：< 元素类型 >，状态 < 元素状态 >，< 元素名称 >，< 问题描述 >。
- 描述存在多个问题的元素：< 元素类型 >，状态 < 元素状态 >，< 元素名称 >，< 问题数 > 个问题。
- 描述不存在问题的元素：< 元素类型 >，状态 < 元素状态 >，< 元素名称 >。

例如，以下映射提供了这样的描述：

'本地网络，6 个元素中有 1 个处于问题状态，共 1 个问题。主机，状态为问题，我的主机，卷 \/ 上的可用磁盘空间小

对应的映射如下：



引用网络拓扑图

网络拓扑图可以被 GET 参数 sysmapid 和 mapname 引用。例如：

http://zabbix/zabbix/zabbix.php?action=map.view&mapname=Local%20network

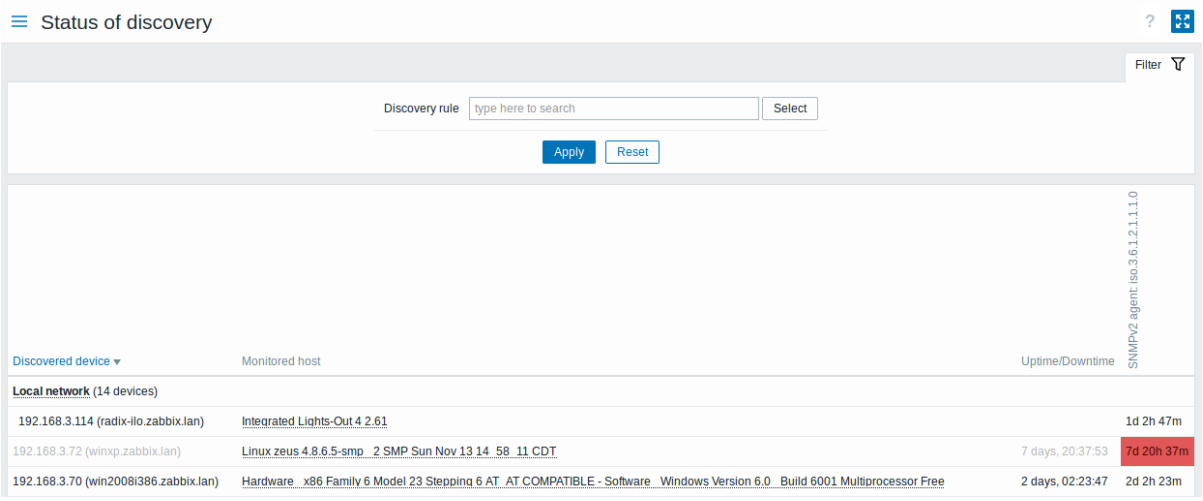
将打开名称是本地网络的拓扑图。

如果同时指定了 sysmapid （拓扑图 ID）和 mapname (拓扑图名称)，mapname 具有更高有优先级。

发现

概述

在监测 > 发现部分中显示网络发现的结果。已发现的设备按发现规则排序。



显示的数据：

列	描述
已发现的设备	已发现的设备按发现规则分组列出。单击发现规则会打开规则菜单，其中包含指向发现规则配置表单的链接。 如果某个设备有多个 IP 地址（由发现规则的设备唯一性条件确定），这些地址会列在一起：首先列出设备的主 IP 地址，随后列出其其他 IP 地址（缩进显示）。主 IP 地址是受监控的主机列中列出了主机的最低 IP 地址；如果没有这样的地址，则为设备的最低 IP 地址。 可以按主 IP 地址升序或降序排序设备；设备的其他 IP 地址始终列在其主 IP 地址下方，并按升序排列。
受监控的主机	如果设备已受到监控，则在此列中列出主机名。 单击主机名会打开主机菜单。
运行时间/停机时间	此列显示设备在上次发现后持续被发现或丢失的时长。 对于具有多个 IP 地址的设备，仅针对其主 IP 地址显示时长。
发现检查	显示每个已发现设备的单个服务（发现检查）的状态。红色单元格表示服务已停止。服务的运行时间或停机时间包含在单元格中。 仅当至少在一个已发现的设备上找到该服务时，才会显示此列。

Note:
此部分中显示的发现规则数量和已发现设备总数受管理 > 常规 > 前端中的搜索和筛选结果限制参数限制。超出限制的规则或设备不会显示，也不会对此进行任何提示。

按钮

所有部分通用的视图模式按钮在Monitoring 页面中有说明。

使用筛选器

您可以使用筛选器仅显示您感兴趣的发现规则。为获得更好的搜索性能，数据搜索时不会解析宏。

当筛选器中未选择任何内容时，将显示所有已启用的发现规则。要选择特定的发现规则进行显示，请在筛选器中开始输入其名称。所有匹配的已启用发现规则都会列出供选择。可以选择多个发现规则。

APM

概述

APM 菜单用于 Zabbix 的应用性能监控（APM）功能。它显示应用程序和服务使用 OpenTelemetry（一种用于收集遥测数据的开放标准）发送的遥测数据。这些数据可帮助您了解请求如何流经应用程序和服务、衡量其性能，并找出操作缓慢或失败的原因。

该菜单包含以下部分，分别对应 OpenTelemetry 的信号类型：

- 追踪
- 指标
- 日志

与显示由 Zabbix 监控项收集的值的监测 > 最新数据不同，*APM* 部分直接显示来自数据源的遥测数据，因此您无需创建任何监控项。要在 *APM* 部分显示数据：

1. 在管理 > 数据源中指定遥测数据源。
2. 确保遥测数据由启用 APM的 Zabbix proxy 或 OpenTelemetry collector 写入数据源。请注意，Zabbix 服务器无法接收遥测数据；只有 Zabbix proxy 可以接收。

要在触发器或图形中使用遥测数据，请使用遥测查询监控项。

跟踪

概述

APM > 跟踪部分显示从应用程序和服务中收集的跟踪。

跟踪表示请求经过应用程序和服务时所采用的路径。例如，当用户在网上商店中点击下单按钮时，请求可能会从前端服务经过结账和支付服务，最终到达数据库。

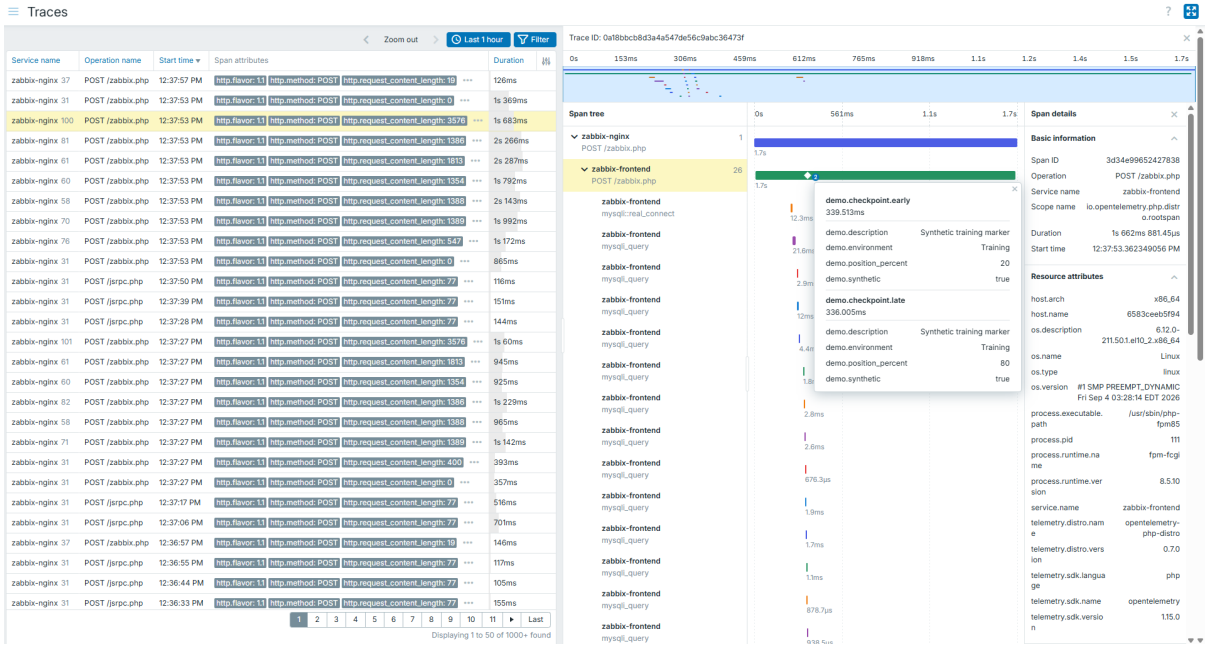
跟踪由跨度组成。每个跨度表示一个单独的操作，例如前端服务向结账服务发送的 HTTP 请求、保存订单的数据库查询，或计算运费的函数调用。

跨度构成层次结构。根跨度表示请求的第一个操作。其他每个跨度都是其操作启动期间所在跨度的子跨度。例如，接收下单请求的前端服务是根跨度，而处理订单的结账服务是其子跨度。

每个跨度由一个服务生成（例如 `zabbix-frontend`；其名称由 OpenTelemetry 的 `service.name` 资源属性指定），并由一个插桩作用域生成，该作用域表示创建该跨度的代码部分（例如用于跟踪 HTTP 请求的库）。

为了提供有关每个操作的上下文，跨度还包含键值形式的属性：跨度属性描述操作本身（例如 `http.method: POST`），而资源属性描述服务及其运行环境（例如 `host.name: eabf8c2a5ddc`）。

有关跟踪的更多详细信息，请参阅 OpenTelemetry 文档。



APM > 跟踪部分包含以下元素：

- 信息亭模式按钮 - 仅显示页面内容，不显示菜单和页眉。
- 时间段选择器 - 选择要显示跟踪的时间段。
- 筛选器 - 仅显示感兴趣的跟踪。
- 跟踪列表 - 查看跟踪，跟踪由其根跨度描述。
- 跟踪面板 - 在时间轴上查看所选跟踪的所有跨度。

跟踪列表

跟踪列表中的每一行代表一个跟踪；各列显示其根 span 的数据。

Service name	Operation name	Start time ▾	Span attributes	Duration
zabbix-nginx 42	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 796	2s 983ms
zabbix-nginx 45	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 827	2s 694ms
zabbix-nginx 41	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 789	2s 595ms
zabbix-nginx 49	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 1375	2s 964ms
zabbix-nginx 46	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 1359	3s 75ms
zabbix-nginx 46	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 1368	2s 983ms
zabbix-nginx 56	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 1821	3s 160ms
zabbix-nginx 41	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 790	2s 777ms
zabbix-nginx 46	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 803	2s 992ms
zabbix-nginx 42	POST /zabbix.php	03:40:28 PM	http.flavor: 1.1 http.method: POST http.request_content_length: 793	2s 900ms

跟踪以可自定义的表格形式显示。表头中的 自定义表格按钮可用于显示、隐藏或重新排列列，以及将布局重置为默认设置。您还可以通过拖动列的右边框来调整列宽；双击边框可将其重置。所有自定义设置会立即应用，并为当前用户保存。

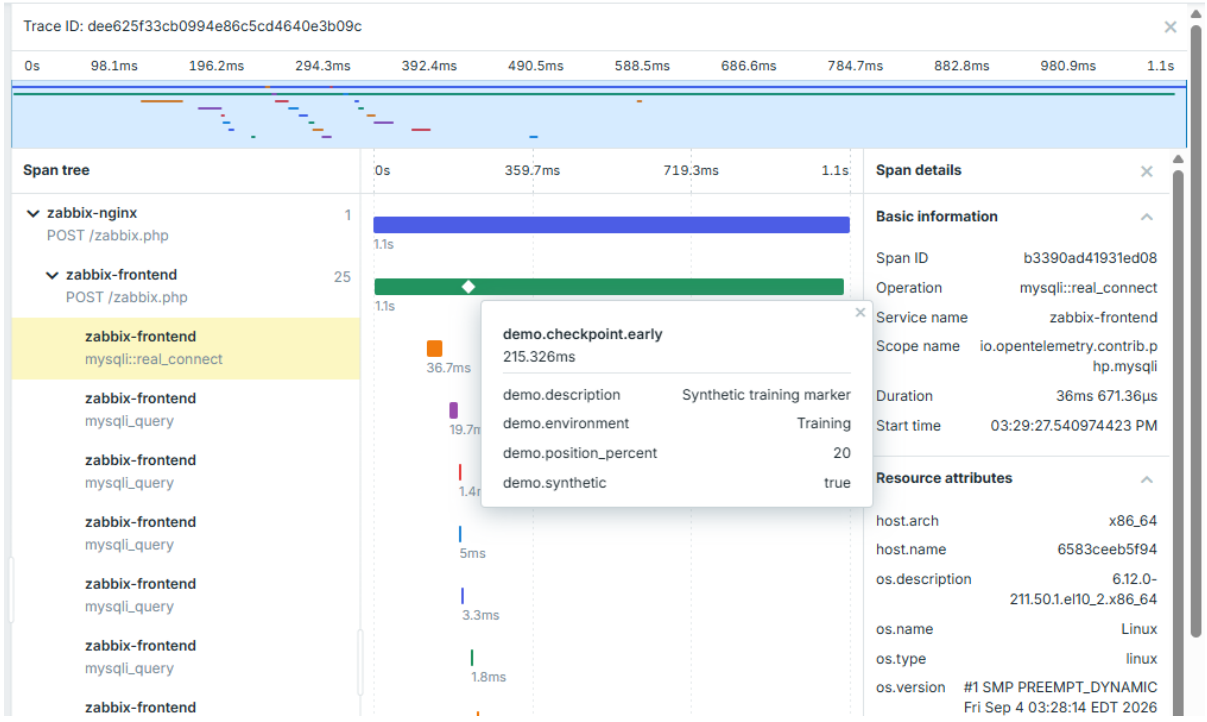
跟踪列表可以包含以下列：

Note:
每一列对应于遥测数据源中 `otel_traces` 表的一列；有关列列表，请参阅支持的遥测数据列。大多数列的名称与数据源中的名称相同（例如，跟踪 *ID* 对应于 `TraceId` 列）。如果名称不同，下表会指定数据源中的列。

列	描述
跟踪 *ID*	跟踪的 ID，属于该跟踪的所有 span 共享此 ID。
Span ID	根 span 的 ID。
跟踪状态	跟踪工具附加到跟踪的其他数据（如果有）（请参阅 W3C Trace Context）。
服务名称	生成根 span 的服务，后跟跟踪中的 span 数量。 如果任何 span 的状态代码为 *Error*（见下文），红色图标会显示这些 span 的数量。
操作名称	根 span 所表示的操作（例如，`POST /zabbix.php`）。 数据源列：`SpanName`。
Span 类型	根 span 的角色： 未指定 - 未指定。 内部 - 在应用程序内执行的操作。 服务器 - 处理传入请求。 客户端 - 传出请求。 生产者 - 创建异步操作（例如，发送消息）。 消费者 - 处理异步操作。
作用域名称	生成根 span 的检测作用域。
作用域版本	检测作用域的版本。
状态代码	根 span 所表示操作的状态： 未设置 - 未为根 span 明确分配状态。 正常 - 操作已成功完成。 错误 - 操作完成时发生错误。
状态消息	有关状态的其他信息（如果有）。
开始时间	根 span 开始的日期和时间。 数据源列：`Timestamp`。
Span 属性	根 span 的 span 属性。 单击 ••• 三点按钮可显示所有属性。
资源属性	根 span 的资源属性。 单击 ••• 三点按钮可显示所有属性。
持续时间	跟踪的持续时间；灰色条显示其相对于其他跟踪的持续时间。

跟踪面板

在跟踪列表中单击某个跟踪以打开跟踪面板，该面板在时间轴上显示该跟踪的所有跨度：执行了哪些操作、执行顺序以及每项操作耗时多久。



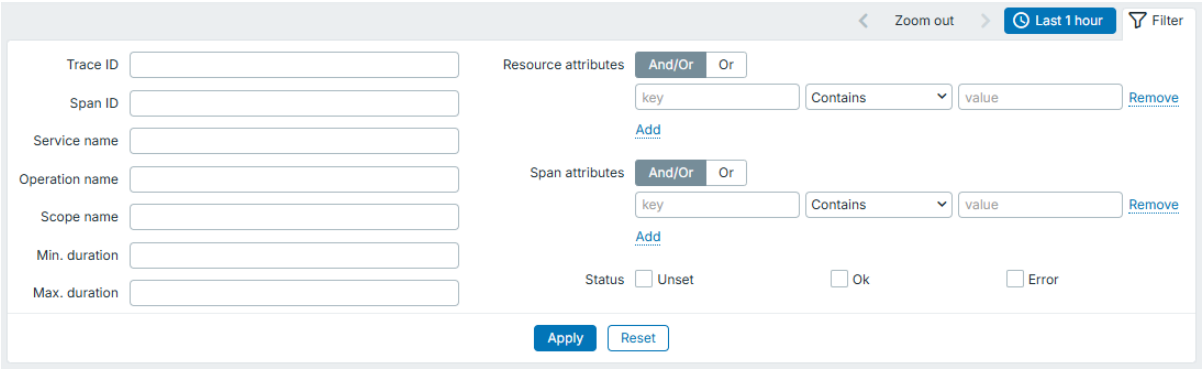
跟踪面板可能包含以下元素：

元素	描述
跟踪 *ID*	所选跟踪的 ID。

元素	描述
跟踪时间轴概览	整个跟踪，其中的跨度最多显示为 8 行（多个跨度可以共用一行）。 拖动所选范围的边界或范围本身，可将跨度时间轴缩放到跟踪的相应部分。
跨度树	按层次结构显示跨度；每个跨度显示其服务（粗体）、操作名称以及直接子跨度的数量。
跨度时间轴	以条形显示跨度：位置和长度表示每个跨度的开始时间及其耗时。 标记表示事件（例如异常）；将鼠标悬停在标记上或单击标记可查看详细信息。
跨度详细信息	在跨度树中单击某个跨度可查看其详细信息和属性。

使用筛选器

您可以使用筛选器，仅显示您感兴趣的追踪记录。



要按属性筛选，请输入属性键，选择运算符，然后输入属性值。

支持以下运算符：

- 包含 - *(默认)* 属性值包含输入的字符串。
- 等于 - 属性值等于输入的字符串。
- 不包含 - 属性值不包含输入的字符串。
- 不等于 - 属性值不等于输入的字符串。
- 存在 - 存在具有输入键的属性。
- 不存在 - 不存在具有输入键的属性。

如果添加了多个条件，请选择评估这些条件的方式：

- **And/Or** - *(默认)* 必须满足所有条件；具有相同属性键的条件使用 *Or* 组合。
- **Or** - 至少满足一个条件。

指标

概述

APM > 指标部分显示从应用程序和服务收集的指标数据点。

指标是某项测量结果（例如，请求持续时间、已处理订单数或内存使用量）。应用程序将其值作为数据点发送，其中包含在特定时间测量的值（或聚合值）。指标类似于 Zabbix 监控项，数据点类似于该监控项的值。

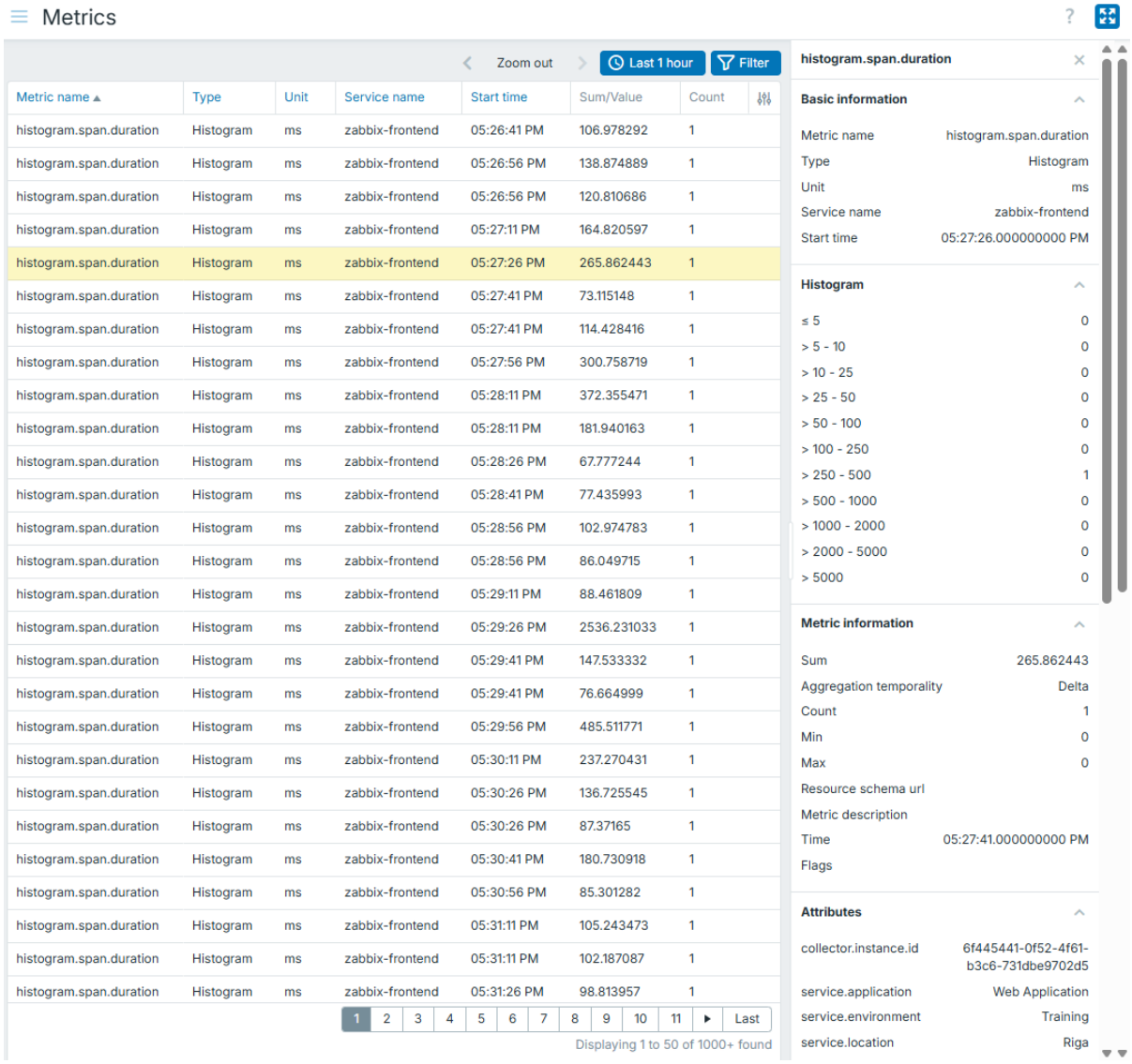
数据点可以是以下类型之一：

- 仪表盘（**Gauge**） - 在特定时间测量的值（例如，当前内存使用量）。
- 总和（**Sum**） - 一段时间内测量值的总和（例如，请求总数）。
- 直方图（**Histogram**） - 一段时间内落入多个固定值范围（称为桶）的测量值数量（例如，请求耗时分别为 0-100 毫秒、100-250 毫秒或 250-500 毫秒的数量）。
- 指数直方图（**Exponential histogram**） - 类似于直方图，但每个桶的宽度都按相同的倍数递增（例如，1-2 毫秒、2-4 毫秒、4-8 毫秒）。

每个数据点由一个服务生成（例如，`zabbix-frontend`；由 OpenTelemetry 的 `service.name` 资源属性命名），并由一个检测范围生成，即创建该数据点的代码部分（例如，用于测量 HTTP 请求持续时间的库）。

为了提供每项测量的上下文，数据点还具有键值形式的属性：数据点属性描述测量的内容，并区分同一指标的数据点（例如，`http.request.method: POST`）；资源属性描述服务及其运行环境（例如，`host.name: eabf8c2a5ddc`）。

有关指标的更多详细信息，请参阅 OpenTelemetry 文档。



APM > 指标部分包含以下元素：

- 全屏模式按钮 - 仅显示页面内容，不显示菜单和页眉。
- 时间段选择器 - 选择要显示数据点的时间段。
- 筛选器 - 仅显示您关注的数据点。
- 指标列表 - 查看数据点。
- 指标详细信息 - 查看所选数据点的详细信息。

指标列表

指标列表中的每一行代表一个数据点。

Metric name	Type	Unit	Service name	Start time ▾	Sum/Value	Cou...
histogram.span.duration	Histogram	ms	zabbix-frontend	02:28:15 PM	71.575147	1
histogram.span.duration	Histogram	ms	zabbix-frontend	02:28:15 PM	18.663078	28
histogram.span.duration	Histogram	ms	zabbix-nginx	02:28:15 PM	154	2
histogram.span.duration	Histogram	ms	zabbix-nginx	02:28:15 PM	5704	8
exponential.span.duration	Exponential histogram	ms	zabbix-frontend	02:28:15 PM	18.663078	28
exponential.span.duration	Exponential histogram	ms	zabbix-frontend	02:28:15 PM	338.947032	1
exponential.span.duration	Exponential histogram	ms	zabbix-frontend	02:28:15 PM	71.575147	1
exponential.span.duration	Exponential histogram	ms	zabbix-frontend	02:28:15 PM	2.278072	1
exponential.span.duration	Exponential histogram	ms	zabbix-nginx	02:28:15 PM	154	2
exponential.span.duration	Exponential histogram	ms	zabbix-nginx	02:28:15 PM	5704	8
nginx.connections_accepted	Summary	{connections}	Web Application	02:03:30 AM	2345	
nginx.connections_current	Summary	{connections}	Web Application	02:03:30 AM	10	

数据点显示在可自定义的表格中。表头中的 自定义表格按钮可用于显示、隐藏或重新排列列，并将布局重置为默认设置。您还可以通过拖动列的右边框来调整列宽；双击边框可将其重置。所有自定义设置会立即应用，并为当前用户保存。

指标列表可以包含以下列：

Note:
除类型外，每一列都对应于遥测数据源的指标表（otel_metrics_gauge、otel_metrics_sum、otel_metrics_histogram 和 otel_metrics_exponential_histogram）中的一列；有关列列表，请参阅支持的遥测数据列。大多数列的名称与数据源中的名称相同（例如，指标名称对应于 MetricName 列）。如果名称不同，下表中会指定数据源列。

列	描述
指标名称	数据点所属的指标（例如，http.server.duration）。
类型	数据点的类型：*Gauge*、*Sum*、*Histogram* 或 *Exponential histogram*。 由包含该数据点的指标表确定。
单位	指标值的单位（如果有）（例如，s 或 By；请参阅 UCUM）。 数据源列：MetricUnit。
服务名称	生成该数据点的服务。
范围名称	生成该数据点的检测范围。
开始时间	数据点测量周期开始的日期和时间。 数据源列：StartTimeUnix。
总和/值	所有测量值的总和（*Histogram*、*Exponential histogram*），或数据点的值（*Gauge*、*Sum*）。
计数	*Histogram* 或 *Exponential histogram* 数据点中的测量值数量。
时间	数据点的日期和时间；对于 *Sum*、*Histogram* 和 *Exponential histogram* 数据点，表示测量周期的结束时间。 数据源列：TimeUnix。
标志	数据点的标志： **0** - 未设置任何标志。 **1** - 未记录该数据点的值。
聚合临时性	*Sum*、*Histogram* 或 *Exponential histogram* 数据点的值如何随时间聚合： 未指定 - 未指定。 增量 - 聚合自上一个数据点以来期间的值。 累计 - 聚合自开始时间以来期间的值。
指标描述	指标的描述（如果有）。
属性	数据点属性。 单击 ••• 三点按钮可显示所有属性。
资源属性	数据点的资源属性。 单击 ••• 三点按钮可显示所有属性。
范围属性	检测范围的属性。 单击 ••• 三点按钮可显示所有属性。

指标详细信息

点击指标列表中的数据点，打开指标详细信息面板，其中显示有关该数据点的所有信息。

exponential.span.calls

Basic information

Metric name	exponential.span.calls
Type	Summary
Unit	
Service name	zabbix-frontend
Start time	02:22:15.000000000 PM

Metric information

Value	1
Aggregation temporality	Delta
Is monotonic	true
Resource schema url	
Metric description	
Time	02:22:30.000000000 PM
Flags	

Attributes

Resource attributes

Scope attributes

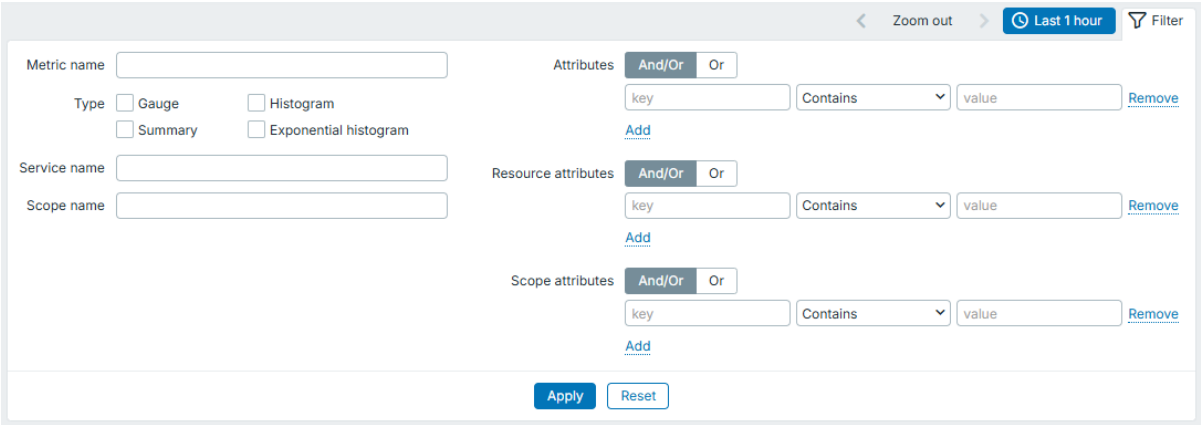
Scope

指标详细信息面板可能包含以下部分：

部分	描述
基本信息	指标名称、类型、单位、服务名称和开始时间。
直方图	存储桶（例如 > 5 - 10），以及每个存储桶中的测量数量。 仅对 直方图数据点显示。
指数直方图	存储桶（例如 >= 249.7885 - 249.7887），以及每个存储桶中的测量数量。 仅对 指数直方图数据点显示。
指标信息	数据点的值（取决于其类型），以及指标描述、时间、标志和资源架构 URL。
属性	数据点属性。
资源属性	资源属性。
范围属性	检测范围的属性（如果有）。
范围	检测范围的名称、版本和架构 URL。

使用筛选器

您可以使用筛选器，仅显示感兴趣的数据点。



要按属性筛选，请输入属性键，选择运算符，然后输入属性值。

支持以下运算符：

- 包含 - *(默认)* 属性值包含输入的字符串。
- 等于 - 属性值等于输入的字符串。
- 不包含 - 属性值不包含输入的字符串。
- 不等于 - 属性值不等于输入的字符串。
- 存在 - 存在具有输入键的属性。
- 不存在 - 不存在具有输入键的属性。

如果添加多个条件，请选择评估这些条件的方式：

- **And/Or** - *(默认)* 必须满足所有条件；具有相同属性键的条件使用 *Or* 组合。
- **Or** - 至少满足一个条件。

日志

概述

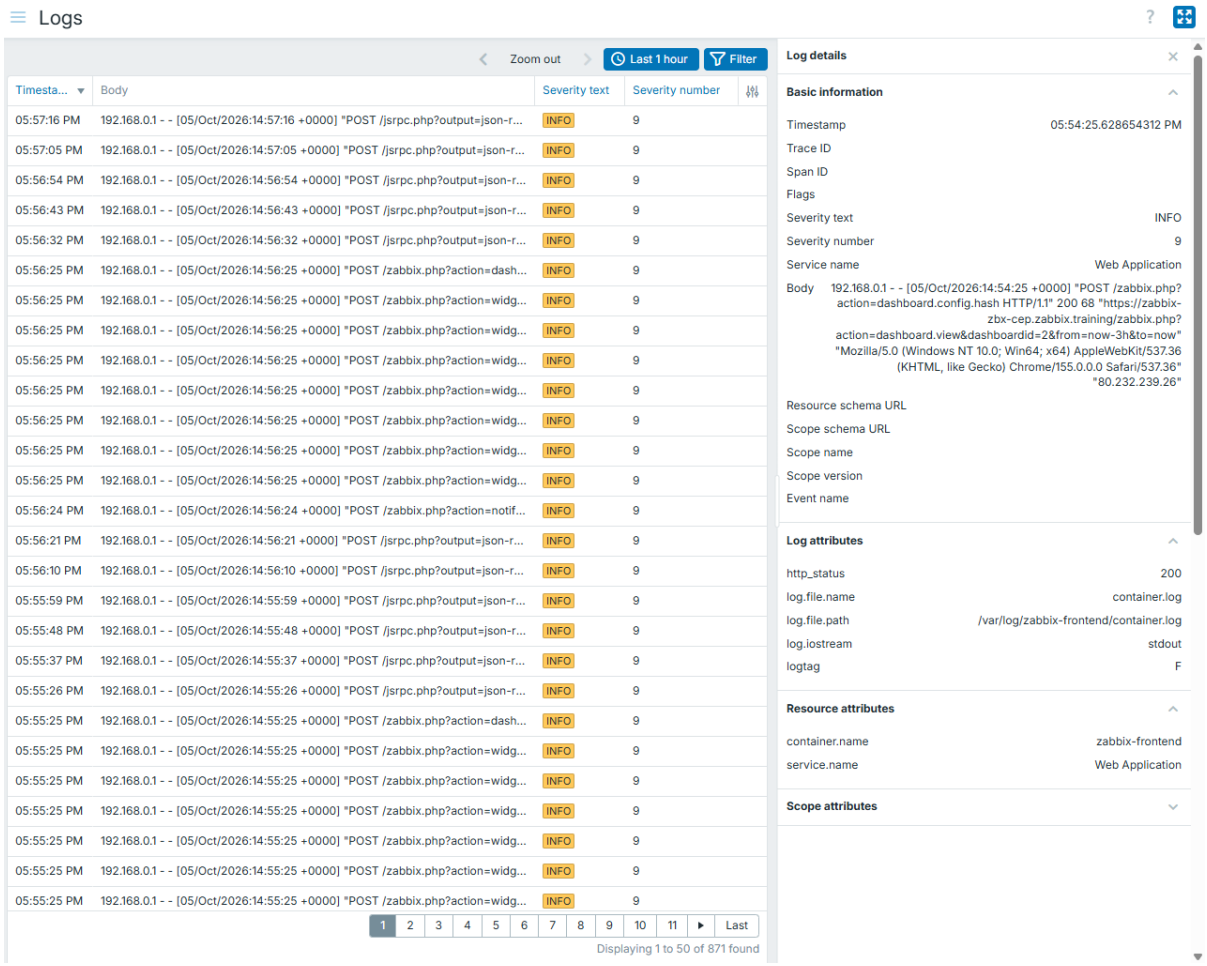
APM > 日志部分显示从应用程序和服务中收集的日志记录。

日志记录是对某个事件的文本记录（例如，支付失败或数据库超时）。每条日志记录都有严重性和消息（正文）。如果日志记录是在某个属于跟踪的操作期间生成的，则还会包含该跟踪和跨度的 ID。

每条日志记录由一个服务生成（例如，`zabbix-frontend`；其名称由 OpenTelemetry 的 `service.name` 资源属性指定），并由一个检测范围生成，该范围表示创建日志记录的代码部分（例如，日志记录库）。

为了提供有关每个事件的上下文，日志记录还包含键值形式的属性：日志属性描述事件本身（例如，`log.file.path: /var/log/zabbix/zabbix_frontend.log`），而资源属性描述服务及其运行环境（例如，`host.name: eabf8c2a5ddc`）。

有关日志的更多详细信息，请参阅 OpenTelemetry 文档。

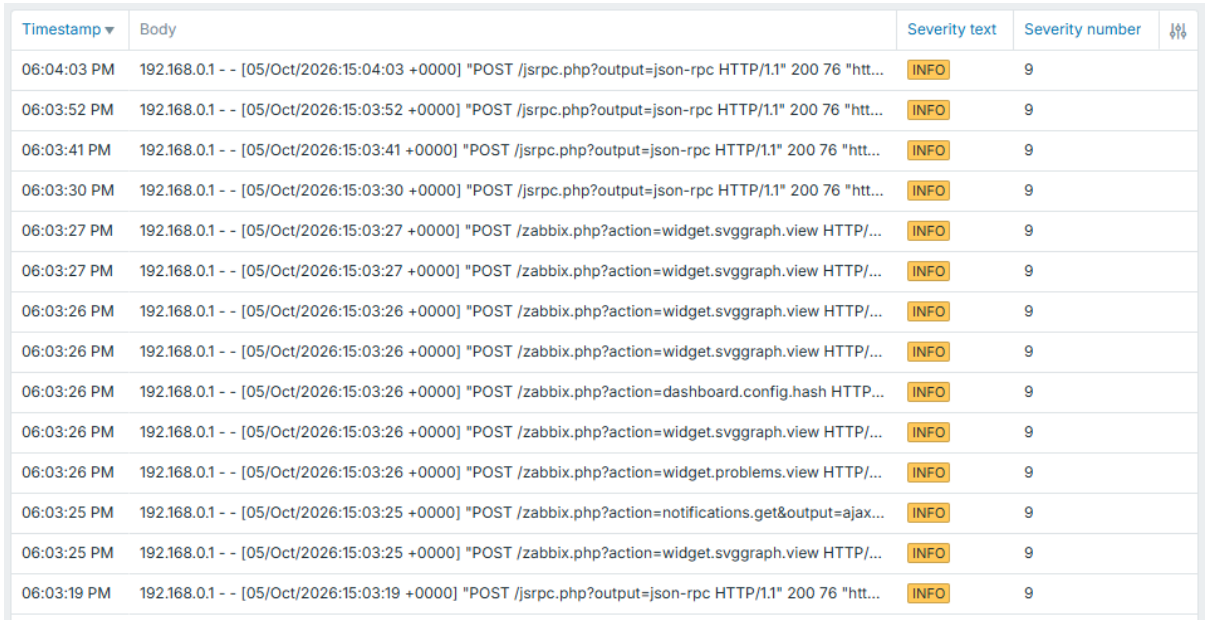


APM > 日志部分包含以下元素：

- Kiosk 模式按钮 - 仅显示页面内容，不显示菜单和页眉。
- 时间段选择器 - 选择要显示日志记录的时间段。
- 筛选器 - 仅显示您关注的日志记录。
- 日志列表 - 查看日志记录。
- 日志详细信息 - 查看所选日志记录的详细信息。

日志列表

日志列表中的每一行代表一条日志记录。



日志记录显示在可自定义的表格中。表头中的 自定义表格按钮可用于显示、隐藏或重新排列列，并将布局重置为默认设置。您还可以通过拖动每列的右边框来调整列宽；双击边框可重置列宽。所有自定义设置会立即应用，并为当前用户保存。

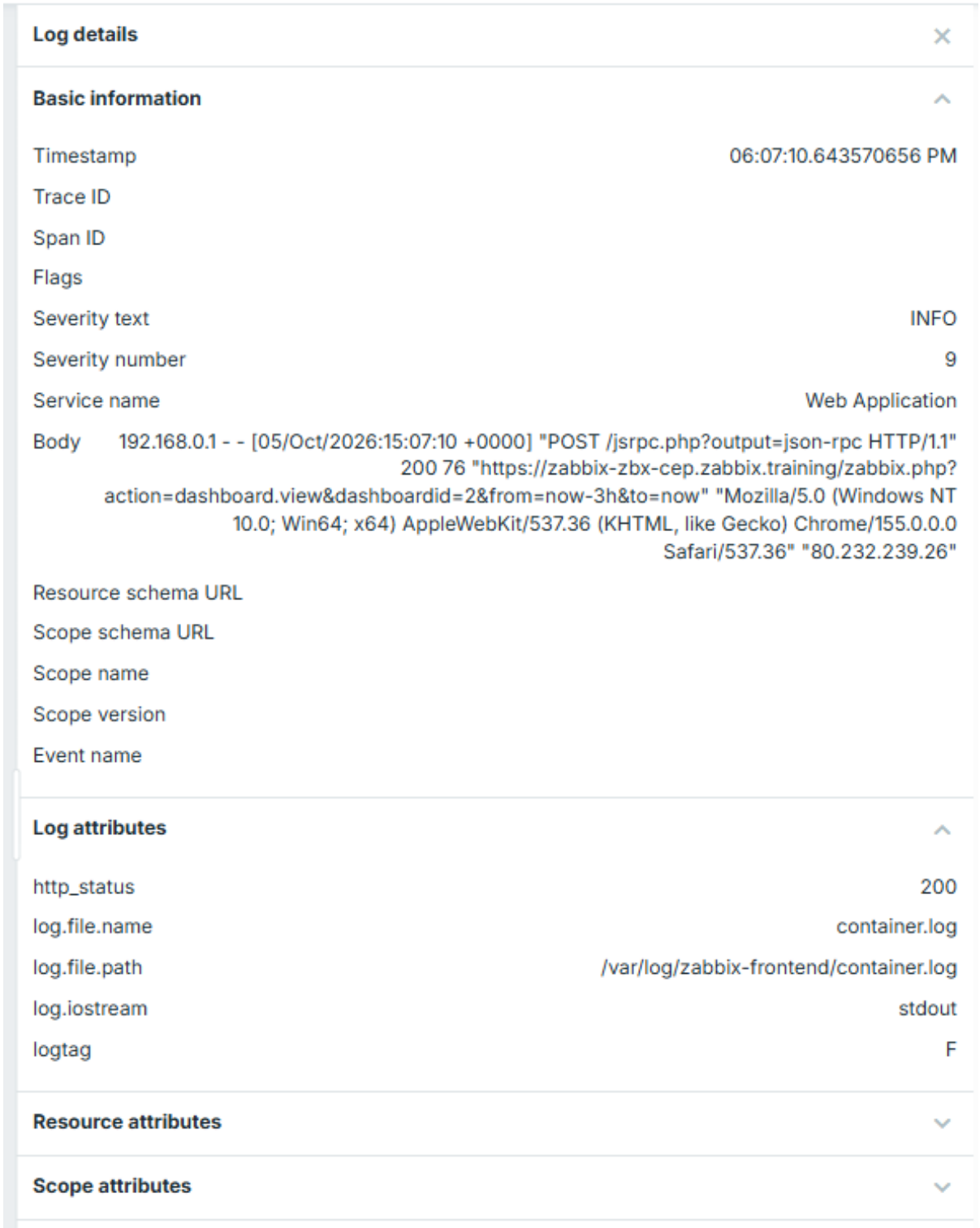
日志列表可以包含以下列：

> **Note:**
> 每列对应于遥测数据源的 `otel_logs` 表中的一列；有关列列表，请参阅支持的遥测数据列。大多数列的名称与数据源中的名称相同（例如，严重性文本对应于 `SeverityText` 列）。如果名称不同，下表会指定数据源中的列名。

列	描述
时间戳	事件发生的日期和时间。
正文	日志记录的消息。
严重性文本	应用程序提供的严重性级别（例如 `INFO`、`WARN` 或 `ERROR`）。 标签颜色取决于严重性。
严重性数值	以数字表示的严重性级别： **0** - 未指定。 **1-4** - TRACE。 **5-8** - DEBUG。 **9-12** - INFO。 **13-16** - WARN。 **17-20** - ERROR。 **21-24** - FATAL。
Trace ID	生成日志记录期间所处跟踪的 ID（如果有）。
Span ID	生成日志记录期间所处 span 的 ID（如果有）。
标志	日志记录的跟踪标志： **0** - 未对跟踪进行采样。 **1** - 已对跟踪进行采样（已记录）。 数据源列：`TraceFlags`。
服务名称	生成日志记录的服务。
范围名称	生成日志记录的检测范围。
范围版本	检测范围的版本。
事件名称	日志记录所表示的事件类型的名称（如果有）。
日志属性	日志记录的日志属性。 单击 ••• 三点按钮可显示所有属性。
资源属性	日志记录的资源属性。 单击 ••• 三点按钮可显示所有属性。
范围属性	检测范围的属性。 单击 ••• 三点按钮可显示所有属性。

日志详细信息

在日志列表中点击一条日志记录，可打开日志详细信息面板，其中显示该日志记录的所有信息。

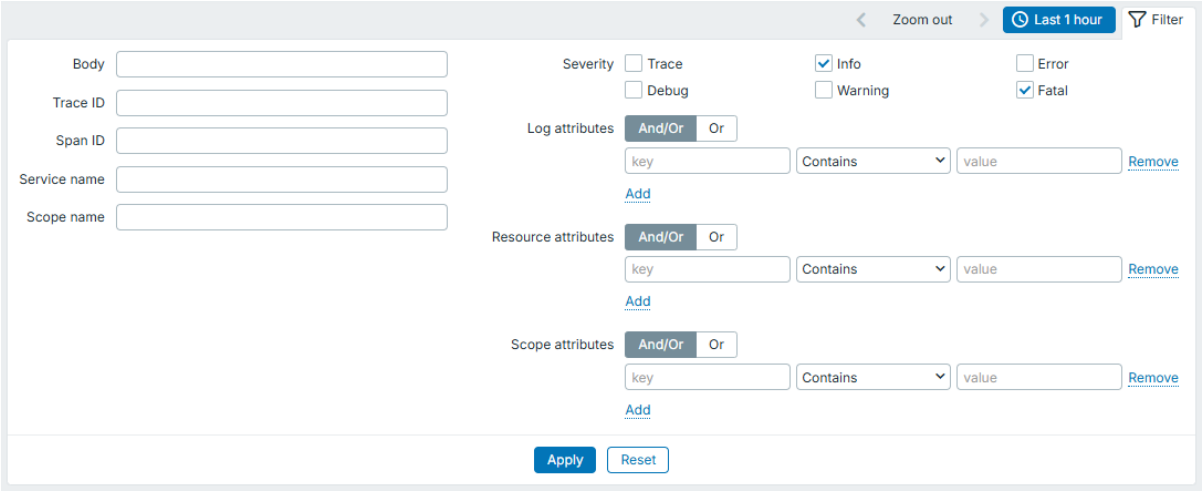


日志详细信息面板可能包含以下部分：

部分	描述
基本信息	日志记录的 Trace ID、span ID 和 trace flags。
日志属性	日志记录的日志属性。
资源属性	Schema URL 和资源属性。
作用域属性	检测作用域的 Schema URL、名称、版本和属性。

使用筛选器

您可以使用筛选器，仅显示您感兴趣的日志记录。



要按属性筛选，请输入属性键，选择运算符，然后输入属性值。

支持以下运算符：

- 包含 - （默认）属性值包含输入的字符串。
- 等于 - 属性值等于输入的字符串。
- 不包含 - 属性值不包含输入的字符串。
- 不等于 - 属性值不等于输入的字符串。
- 存在 - 具有输入键的属性存在。
- 不存在 - 具有输入键的属性不存在。

如果添加多个条件，请选择评估这些条件的方式：

- **And/Or** - （默认）必须满足所有条件；具有相同属性键的条件使用 *Or* 组合。
- **Or** - 至少满足一个条件。

服务

概述

服务菜单用于 Zabbix 的服务监控功能。

服务

概述

在本节中，您可以根据基础设施查看 Zabbix 中已配置的所有服务的总体状态。

服务可能是由多个层级的其他服务组成的层次结构，这些服务称为“子”服务，它们会影响该服务的总体状态（另请参阅服务监控功能概述）。

服务状态的主要类别为正常或问题，其中问题状态通过相应的问题严重性名称和颜色表示。

查看模式允许您监控服务及其状态和其他详细信息；此外，您还可以在本节中切换到编辑模式，以配置服务层次结构（添加/编辑服务和子服务）。

要从查看模式切换到编辑模式（或反向切换），请单击右上角相应的按钮：

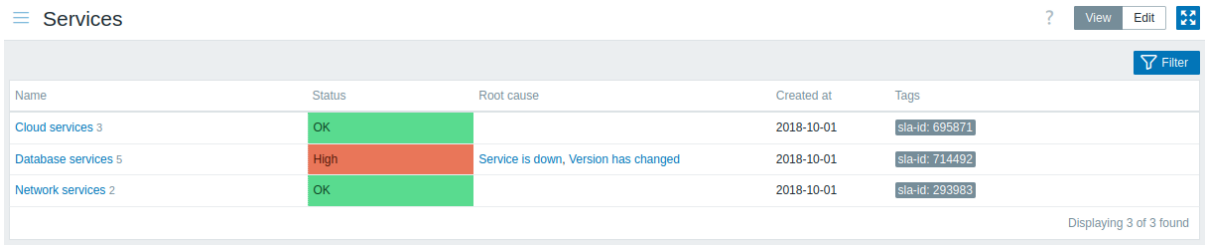
- 

- 查看服务
- 

- 添加/编辑服务和子服务

请注意，是否可以进行编辑取决于用户角色设置。

查看服务



将显示现有服务列表。

显示的数据：

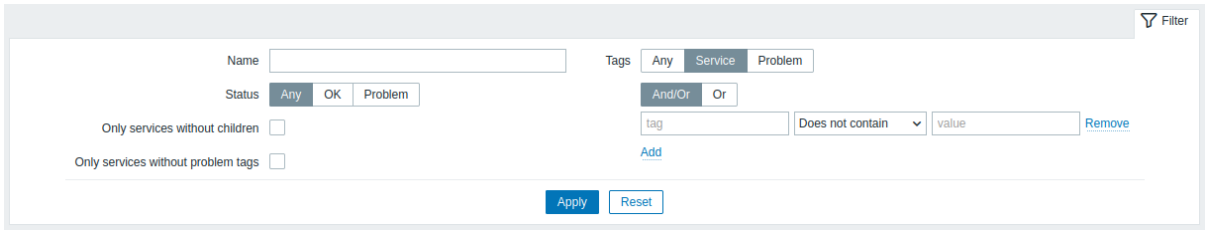
Parameter	Description
Name	服务名称。 服务名称是指向服务详情的链接。 名称后面的数字表示该服务有多少个子服务。
Status	服务状态： **OK** - 无问题 **<problem color and severity>** - 表示存在问题及其严重性。若有多个问题，则显示最严重问题的颜色和严重性。
Root cause	列出直接或间接影响服务状态的根本问题。 这些问题与 {SERVICE.ROOTCAUSE} macro 返回的内容相同。 单击问题名称可在 *Monitoring* > *Problems* 中查看其更多详情。 不影响服务状态的问题不会出现在列表中。
Created at	显示服务的创建时间。
Tags	显示该服务的标签。标签用于在服务actions和SLAs中标识服务。

按钮

所有部分通用的视图模式按钮已在监控 页面中说明。

使用筛选器

您可以使用筛选器仅显示您感兴趣的服务。

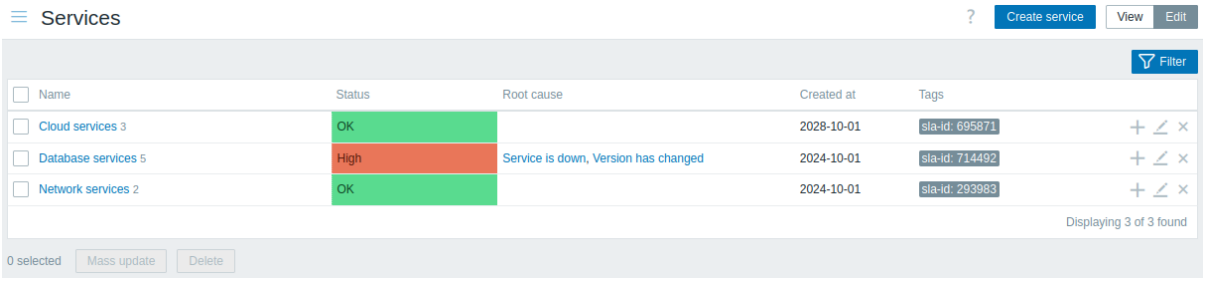


参数	描述
名称	按服务名称筛选。
状态	按服务状态筛选。
仅显示无子服务的服务	勾选此复选框以仅显示没有子服务的服务。此参数仅在服务编辑模式中可用。
仅显示无问题标签的服务	勾选此复选框以仅显示没有问题标签的服务。此参数仅在服务编辑模式中可用。
标签	按服务标签名称和值，或服务问题标签名称和值（在服务编辑模式中）进行筛选。 可以包含特定标签和标签值，也可以排除特定标签和标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件都有多个可用运算符： 存在 - 包含指定的标签名称； 等于 - 包含指定的标签名称和值（区分大小写）； 包含 - 包含指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）； 不存在 - 排除指定的标签名称； 不等于 - 排除指定的标签名称和值（区分大小写）； 不包含 - 排除指定的标签名称，且标签值包含输入的字符串（子字符串匹配，不区分大小写）。 条件有两种计算类型： 与/或 - 必须满足所有条件，具有相同标签名称的条件将按 “或” 条件分组； 或 - 只需满足一个条件即可。

编辑服务

单击 *Edit* 按钮进入编辑模式。在编辑模式下，列表项前会显示复选框，并且还会提供以下附加选项：

-  - 为此服务添加子服务
-  - 编辑此服务
-  - 删除此服务

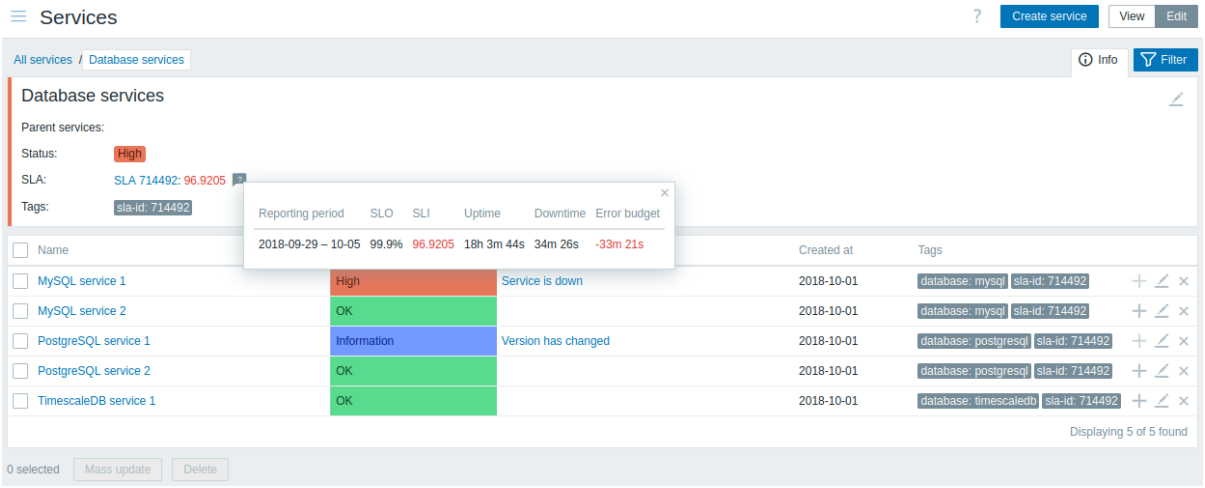


要配置新服务，请单击右上角的 *Create service* 按钮。

服务详情

要访问服务详情，请单击服务名称。要返回所有服务列表，请单击 *All services*。

服务详情包括信息框和子服务列表。



要访问信息框，请单击 *Info* 选项卡。信息框包含以下条目：

- 父服务名称（如果有）
- 此服务的当前状态
- 此服务的当前 SLA，格式为 `SLA name:service level indicator`。`SLA name` 也是指向此服务 SLA 报告的链接。如果将鼠标指针悬停在服务级别指标（SLI）旁边的信息框上，将显示一个弹出信息列表，其中包含 SLI 详细信息。服务级别指标以百分比显示当前服务级别。
- 服务标签

信息框还包含一个指向服务配置的链接。

要使用子服务筛选器，请单击 *Filter* 选项卡。

在编辑模式下，子服务列表会增加以下编辑选项：

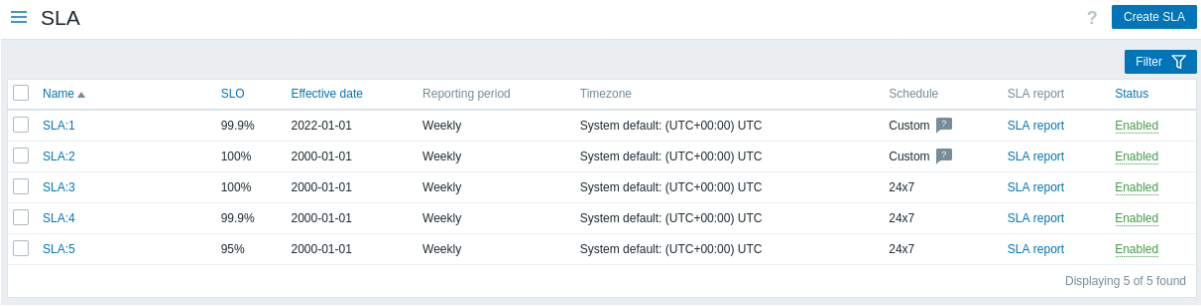
-  - 将子服务添加到此服务
-  - 编辑此服务
-  - 删除此服务

SLA

概述

此部分用于查看和配置 SLA。

SLA



将显示已配置的 SLA 列表。注意，仅会显示与用户可访问的服务相关的 SLA（以只读方式显示，除非用户角色启用了 *Manage SLA*）。

显示的数据：

Parameter	Description
Name	显示 SLA 名称。 名称是SLA 配置 的链接。
SLO	显示服务级别目标（SLO）。
Effective date	显示开始计算 SLA 的日期。
Reporting period	显示 SLA 报告中使用的周期 - *daily*、*weekly*、*monthly*、*quarterly* 或 *annually*。
Time zone	显示 SLA 时区。
Schedule	显示 SLA 日程 - 24x7 或自定义。
SLA report	单击链接查看此 SLA 的 SLA 报告。
Status	显示 SLA 状态 - 启用或禁用。

SLA 报告

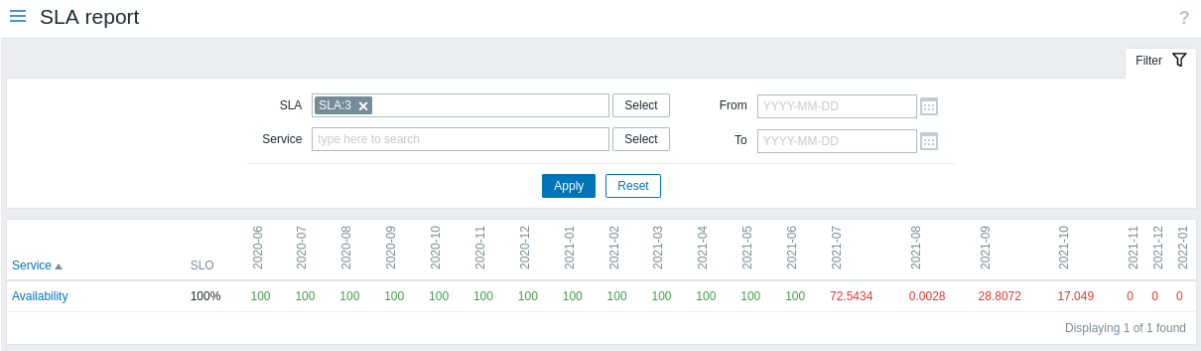
概述

此部分允许根据筛选器中选择的条件查看SLA 报告。

SLA 报告也可以显示为仪表板小组件。

报告

筛选器允许您根据 SLA 名称以及服务名称选择报告。还可以限制显示的时间段。

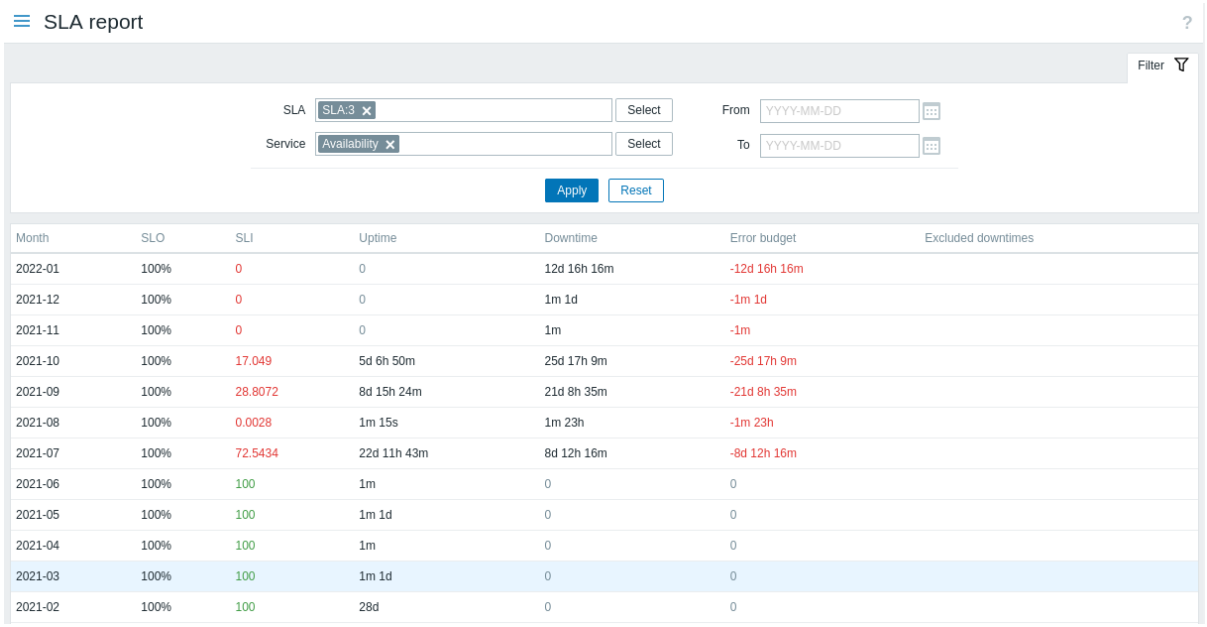


每一列（时间段）显示该时间段的 SLI。违反所设 SLO 的 SLI 将以红色突出显示。

报告中显示 20 个时间段。如果同时指定了 *From* 日期和 *To* 日期，则最多可显示 100 个时间段。

报告详情

如果您在报告中单击服务名称，您可以访问另一份报告，该报告会显示更详细的视图。



Month	SLO	SLI	Uptime	Downtime	Error budget	Excluded downtimes
2022-01	100%	0	0	12d 16h 16m	-12d 16h 16m	
2021-12	100%	0	0	1m 1d	-1m 1d	
2021-11	100%	0	0	1m	-1m	
2021-10	100%	17.049	5d 6h 50m	25d 17h 9m	-25d 17h 9m	
2021-09	100%	28.8072	8d 15h 24m	21d 8h 35m	-21d 8h 35m	
2021-08	100%	0.0028	1m 15s	1m 23h	-1m 23h	
2021-07	100%	72.5434	22d 11h 43m	8d 12h 16m	-8d 12h 16m	
2021-06	100%	100	1m	0	0	
2021-05	100%	100	1m 1d	0	0	
2021-04	100%	100	1m	0	0	
2021-03	100%	100	1m 1d	0	0	
2021-02	100%	100	28d	0	0	

请注意，负问题持续时间 不会影响 SLA 计算或报告。

库存

概述

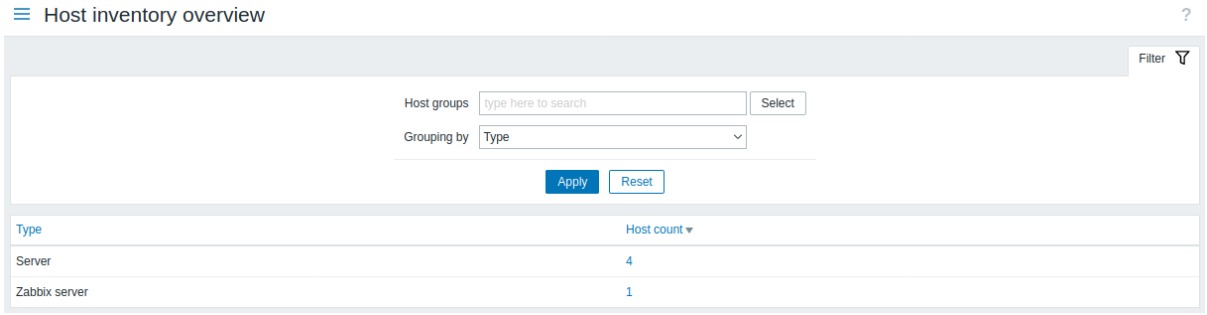
资产菜单功能支持展示已筛选的主机资产数据的概览，同时提供查看主机资产详细信息的功能。

概述

概述

Inventory > Overview 部分提供了查看主机 inventory 数据概览的方法。

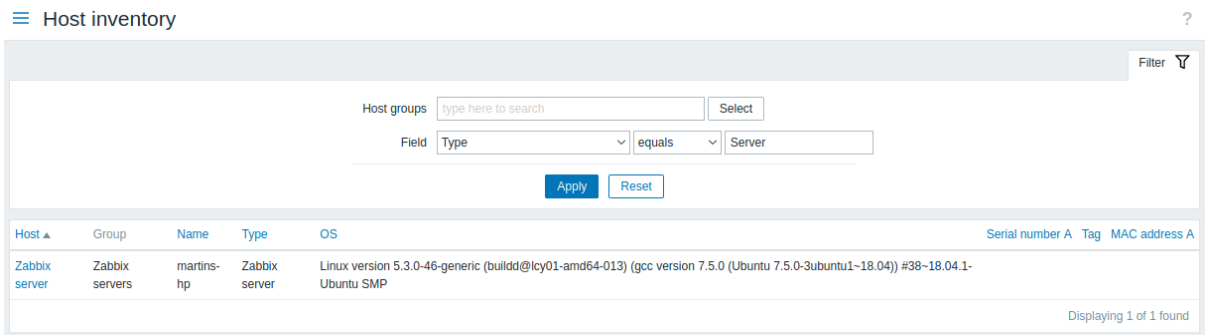
要显示概览，请选择主机组（或不选择任何主机组）以及用于显示数据的 inventory 字段。系统将显示所选字段中每个条目对应的主机数量。



Type	Host count
Server	4
Zabbix server	1

概览的完整程度取决于主机维护了多少 inventory 信息。

Host count 列中的数字是链接；点击后会在 *Host Inventories* 表中筛选出这些主机。



Host	Group	Name	Type	OS	Serial number A	Tag	MAC address A
Zabbix server	Zabbix servers	martins-hp	Zabbix server	Linux version 5.3.0-46-generic (buildd@lcy01-amd64-013) (gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #38~18.04.1-Ubuntu SMP			

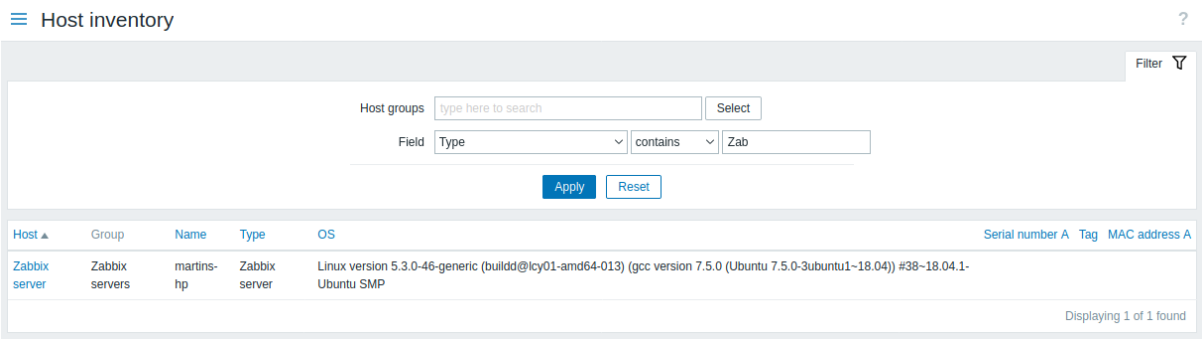
Displaying 1 of 1 found

主机

概述

在 *Inventory* > *Hosts* 部分中，会显示主机的inventory data。

您可以按主机组和任意 inventory 字段筛选主机，以便仅显示您感兴趣的主机。

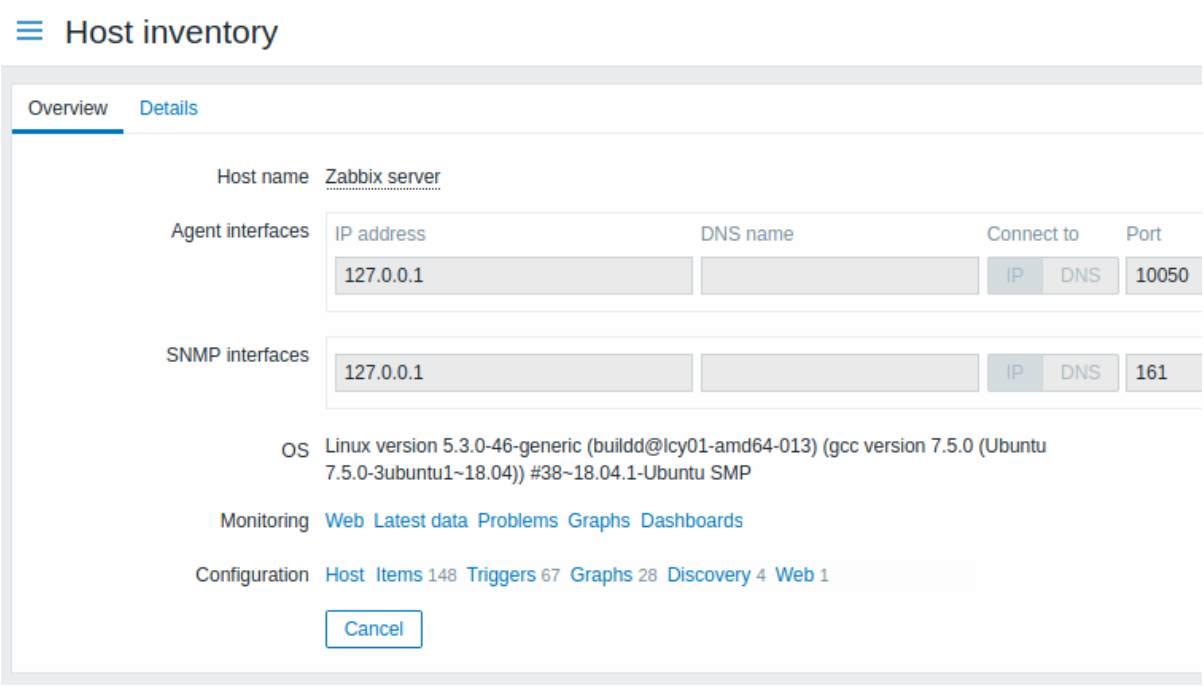


要显示所有主机 inventory，请单击 *Reset* 按钮。

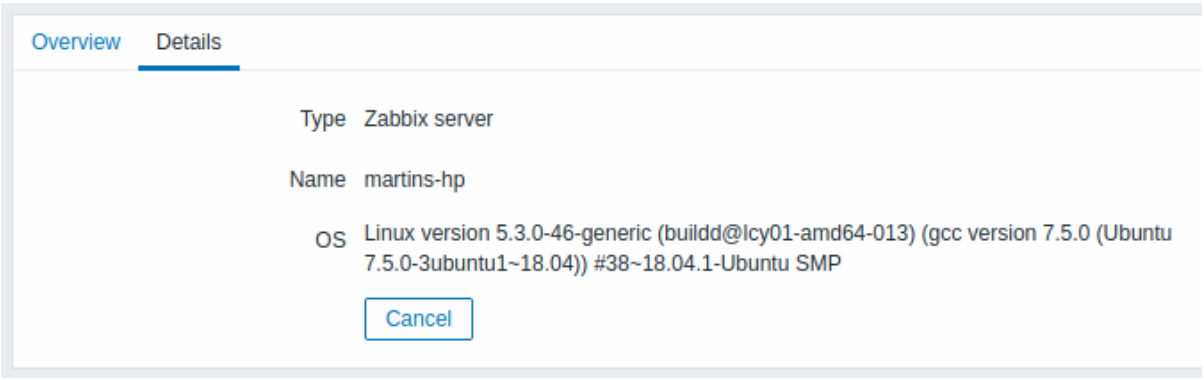
虽然表中只显示部分关键 inventory 字段，但您也可以查看该主机的所有可用 inventory 信息。为此，请单击第一列中的主机名称。

资产清单详情

概览选项卡包含有关主机的一些常规信息、最新的监控数据以及主机配置选项：



详情选项卡包含该主机所有可用的资产清单详情：



资产清单数据的完整性取决于为主机维护了多少资产清单信息。如果未维护任何信息，则 详情选项卡将被禁用。

报告

概述

“报表”菜单包含多个部分，其中包含各种预定义和用户可自定义的报表，这些报告侧重于显示系统信息，触发器和收集数据等参数的概括。

系统信息

概述

在报告 > 系统信息中，会显示 Zabbix 服务器和系统关键数据的摘要。系统数据通过内部监控项收集。

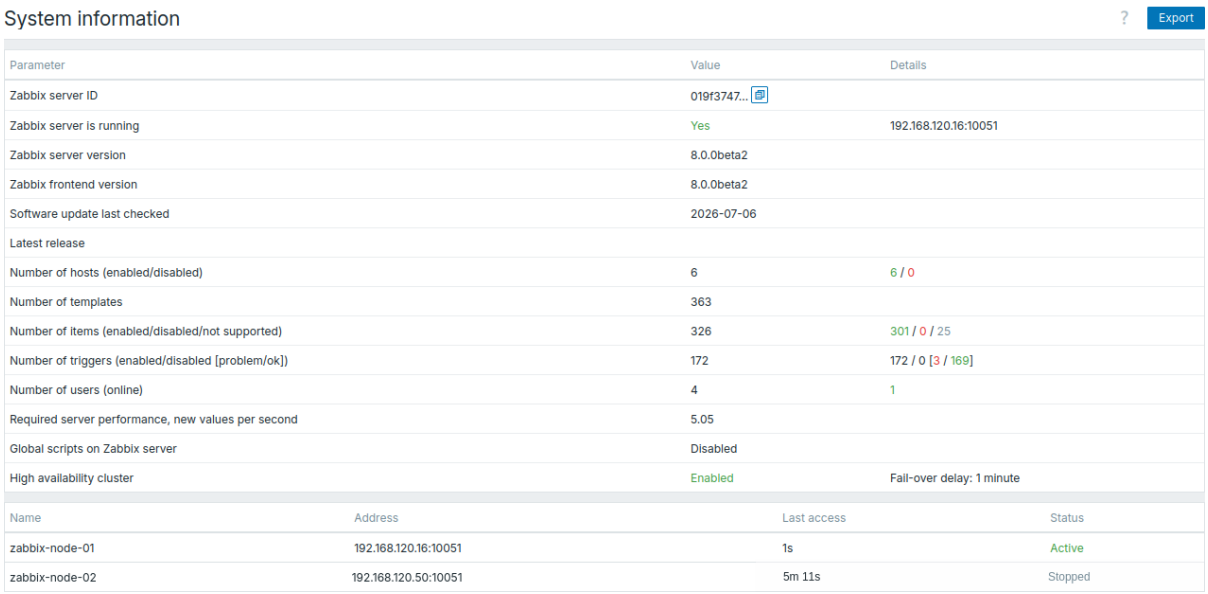
只有 Zabbix 超级管理员和 Zabbix 管理员用户可以查看系统信息报告。

请注意，在高可用性设置中，可以重定向系统信息源（服务器实例）。为此，请编辑前端配置文件文件，取消注释并将 $ZBX_SERVER，或将 $ZBX_SERVER 和 $ZBX_SERVER_PORT 同时设置为不同于当前显示为活动状态的服务器。如果只设置 $ZBX_SERVER，则 $ZBX_SERVER_PORT 将使用默认值（10051）。

启用高可用性设置后，系统统计信息下方会显示一个单独的区块，其中包含高可用性节点的详细信息。

系统信息也可作为仪表板小组件使用。

系统统计信息



显示的数据：

参数	值	详细信息
Zabbix 服务器 *ID*	Zabbix 服务器唯一标识符（UUIDv7）。	服务器 ID 在服务器首次启动时生成。界面显示前 8 个字符，后跟省略号。将鼠标悬停在或单击服务器 ID 值时，会显示包含完整值的工具提示。 单击值旁边的图标 可将 ID 复制到剪贴板。 如果启用了高可用性，每个节点都有自己的节点 ID，用于命令行寻址，但所有节点都归属于一个服务器 ID。该服务器 ID 由最先启动的节点（或之前作为独立服务器运行的节点）生成，与当前处于活动状态的节点无关。
Zabbix 服务器正在运行	Zabbix 服务器的状态： 是 - 服务器正在运行 否 - 服务器未运行 注意：要显示其余信息，Web 前端需要服务器正在运行，并且服务器上至少启动了一个 trapper 进程（`zabbix_server.conf` 文件中的 StartTrappers 参数 > 0）。	Zabbix 服务器的位置和端口。

参数	值	详细信息
Zabbix 服务器版本	显示当前服务器版本号。 注意：仅在 Zabbix 服务器运行时显示。	显示服务器版本状态： 最新 - 使用最新版本； 有新更新可用 - 有更新的版本可用； 已过期 - 此版本的完整支持期已结束。 仅当在 Zabbix 服务器配置中启用了软件更新检查时，才会提供此信息。如果上次软件更新检查是在一周以前执行的，或没有当前版本的相关数据，则不显示任何内容。
Zabbix 前端版本	显示 Zabbix 前端版本号。	显示 Zabbix 前端版本状态： 最新 - 使用最新版本； 有新更新可用 - 有更新的版本可用； 已过期 - 此版本的完整支持期已结束。 仅当在 Zabbix 服务器配置中启用了软件更新检查时，才会提供此信息。如果上次软件更新检查是在一周以前执行的，或没有当前版本的相关数据，则不显示任何内容。
上次软件更新检查时间	显示上次检查 Zabbix 软件更新的日期。 仅当在 Zabbix 服务器配置中启用了软件更新检查时，才会提供此信息。	
最新版本	显示当前 Zabbix 版本对应的较新版本号（如果有）。 仅当在 Zabbix 服务器配置中启用了软件更新检查时，才会提供此信息。如果上次软件更新检查是在一周以前执行的，或没有当前版本的相关数据，则不显示任何内容。	显示指向最新可用 Zabbix 版本发行说明的链接。
主机数量	显示已配置主机的总数。	受监控主机数/未受监控主机数。
模板数量	显示模板总数。	
监控项数量	显示监控项总数。	主机级监控项中受监控/已禁用/不受支持的监控项数量。 已禁用主机上的监控项将计为已禁用。
触发器数量	显示触发器总数。	主机级触发器中已启用/已禁用的触发器数量；已启用触发器根据 “问题”/“正常” 状态进行细分。 “正常” 状态下列出的触发器包括状态为 “未知” 的触发器。 依赖于已禁用监控项的触发器，或分配给已禁用主机的触发器，将计为已禁用。
用户数量	显示已配置用户的总数。	在线用户数。
所需服务器性能，每秒新值数	显示 Zabbix 服务器每秒处理的新值预期数量。	所需服务器性能是一个估算值，可用作参考。要获取已处理值的精确数量，请使用 `zabbix[wcache,values,all]` 内部监控项。 计算中包括受监控主机上的已启用监控项。日志监控项按每个监控项更新间隔一个值进行计数。计算常规间隔值；不计算灵活间隔值和调度间隔值。在 “无数据” 维护期间不会调整计算结果。不计算 trapper 监控项。
Zabbix 服务器上的全局脚本	如果在服务器配置中将 EnableGlobalScripts=0 设置为在 Zabbix 服务器上禁用全局脚本，则此字段显示已禁用。	
高可用性集群	Zabbix 服务器高可用性集群的状态： 已禁用 - 独立服务器 已启用 - 至少存在一个高可用性节点	如果已启用，则显示故障转移延迟。

在以下情况下，系统信息还会显示错误消息：

- 使用的数据库没有所需的字符集或排序规则（UTF-8）。
- 数据库版本低于或高于支持的范围（仅对具有超级管理员角色类型的用户可用）。
- TimescaleDB 的内务处理配置不正确（历史数据表或趋势数据表包含已压缩的数据块，但覆盖监控项历史数据周期或覆盖监控项趋势数据周期选项已禁用）。

按钮

页面右上角的按钮提供以下选项：

Export	将系统信息导出为 JSON 文件。 该文件将保存为 `Zabbix-system-information-{time}.json`，其中 `{time}` 是表示数据在服务器上准备完成时间的 Unix 时间戳。 如果启用了高可用性，还会导出高可用性节点的详细信息。

高可用性节点

如果启用了高可用性集群，则会显示一个额外的数据块，用于展示每个高可用性节点的状态。



Name	Address	Last access	Status
node-active	192.168.1.13:10051	12s	Active
node6	192.168.1.10:10053	1h 2m 40s	Unavailable
node7	192.168.1.11:10053	3m 40s	Unavailable
node4	192.168.1.8:10052	1h 34m 29s	Stopped
node5	192.168.1.9:10053	1h 9m 51s	Stopped
node8	192.168.1.12:10051	21m 16s	Stopped
node1	192.168.1.5:10051	17s	Standby
node2	192.168.1.6:10051	16s	Standby
node3	192.168.1.7:10052	16 2021-10-20 17:58:47	Standby

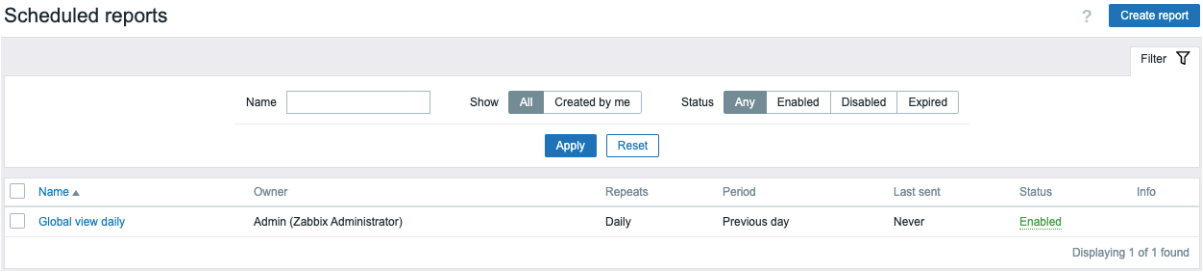
显示的数据：

Column	Description
Name	节点名称，按服务器配置中定义的内容显示。
Address	节点 IP 地址和端口。
Last access	节点最后一次访问的时间。 将鼠标悬停在该单元格上时，会以长格式显示最后访问时间戳。
Status	节点状态。表格行会按以下优先级状态排序： **Active** - 节点已启动并正常工作 **Unavailable** - 节点已超过故障切换延迟时间未被检测到（你可能需要查明原因） **Stopped** - 节点已停止或无法启动（你可能需要启动它或将其删除） **Standby** - 节点已启动并处于等待状态

定时报告

概述

在报表 > 计划报表中，具有足够权限的用户可以配置计划生成仪表板的 PDF 版本，并通过电子邮件将其发送给指定的收件人。



打开的页面会显示计划报表的信息。您可以对这些信息进行筛选，以便于导航 - 请参阅下方的使用筛选器部分。

显示的数据：

列	描述
名称	报表的名称。单击名称可打开报表配置表单。
所有者	创建报表的用户。
重复周期	报表生成频率（每天/每周/每月/每年）。
周期	生成报表所针对的时间周期（前一天、前一周、前一个月或前一年）。
上次发送时间	最新报表发送的日期和时间。

列	描述
状态	报表的当前状态（已启用/已禁用/已过期）。具有足够权限的用户可以单击状态进行更改 - 从 “已启用” 更改为 “已禁用”（反之亦然）；从 “已过期” 更改为 “已禁用”（反之亦然）。权限不足的用户无法单击状态。
信息	显示信息图标： 红色图标表示报表生成失败；将鼠标悬停在图标上时，会显示包含错误信息的工具提示。 黄色图标表示报表已生成，但发送给部分或全部收件人失败，或者报表已过期；将鼠标悬停在图标上时，会显示包含其他信息的工具提示。

使用筛选器

您可以使用筛选器来缩小报告列表的范围。为了获得更好的搜索性能，搜索数据时宏不会被解析。

以下筛选选项可用：

- 名称 - 允许部分名称匹配
- 显示 - 由当前用户创建的报告或所有报告
- 状态 - 可在 “任意”（显示所有报告）、“启用”、“禁用” 或 “已过期” 之间选择

筛选器位于 计划报告部分名称下方。点击右上角的 筛选器选项卡即可将其展开或折叠。

批量更新

有时，您可能希望一次更改多个报告状态或删除多个报告。您可以为此使用批量更新功能，而不是打开每个单独的报告进行编辑。

要批量更新某些报告，请执行以下操作：

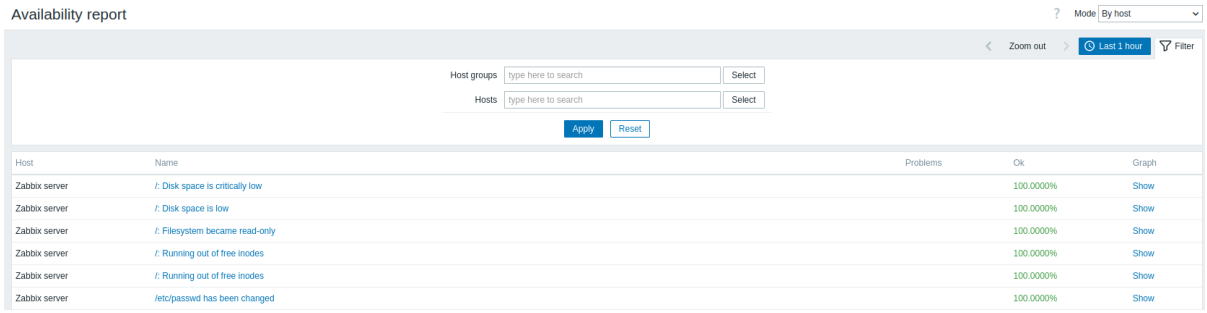
- 在列表中标记要更新的报告的复选框
- 单击列表下方的必填按钮进行更改（启用、禁用或删除）

可用性报告

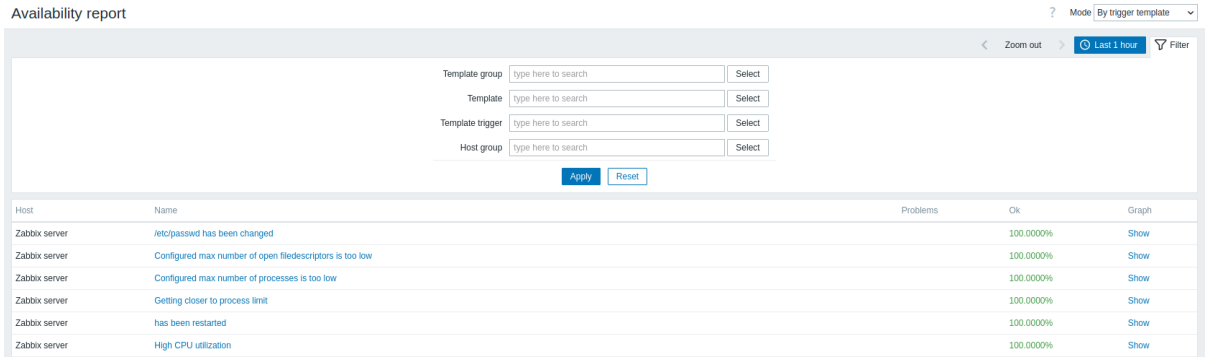
概述

在 报告 > 可用性报告中，您可以查看每个触发器处于问题/正常状态的时间占比。

对于每种状态，都会显示相应的时间百分比，从而便于确定系统中各个元素的可用性。



在右上角的下拉菜单中，您可以选择显示模式——按主机显示触发器，或按属于某个模板的触发器显示。



触发器名称是指向该触发器最新事件的链接。

使用筛选器

筛选器可以帮助缩小显示的主机和/或触发器数量。为了获得更好的搜索性能，搜索数据时不会解析宏。

筛选器位于 *Availability report* 部分名称下方。可以通过单击右侧的 *Filter* 选项卡来展开或折叠它。

按触发器模板过滤

在 按触发器模板模式下，可以使用下列一个或多个参数对结果进行过滤。

参数	描述
模板组	按从属于所选模板组中的模板继承而来的触发器过滤主机。 指定父模板组时，会隐式选择其下所有嵌套模板组。
模板	按从所选模板继承而来的触发器过滤主机，包括嵌套模板。 如果嵌套模板有其自己的触发器，这些触发器将不会显示。
模板触发器	按所选触发器过滤主机。已过滤主机的其他触发器将不会显示。
主机组	过滤属于所选主机组的主机。

按主机过滤

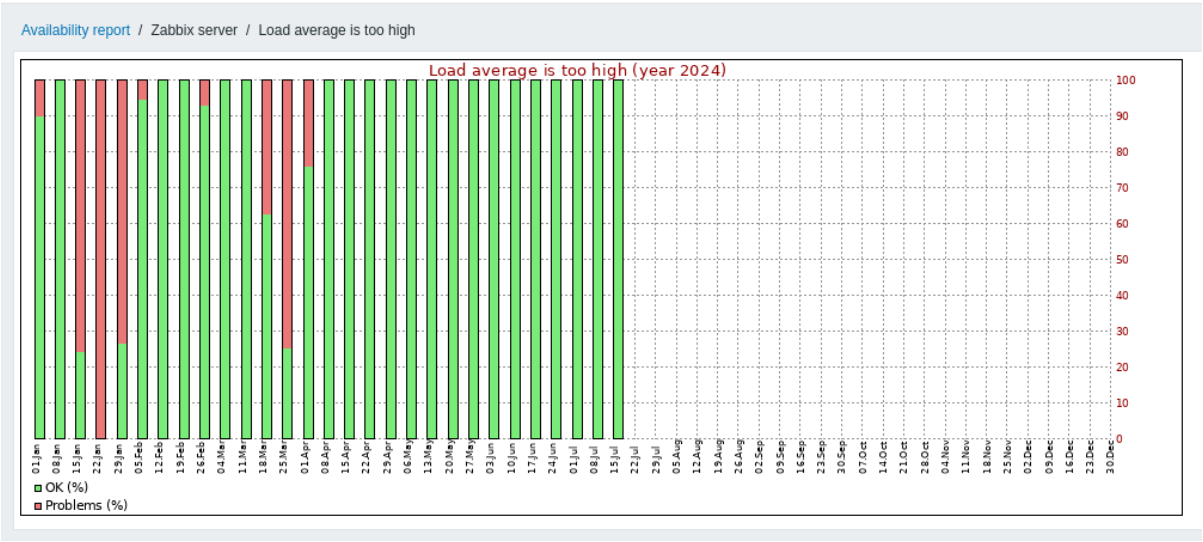
在 按主机模式下，可以按主机或主机组筛选结果。
指定父主机组会隐式选择所有嵌套的主机组。

时间段选择器

时间段选择器允许您通过一次点击选择常用的时间段。单击筛选器旁边的时间段选项卡，可以展开和折叠选择器。

单击图形列中的显示，会以条形图显示可用性信息，其中每个条形代表当前年份中已经过去的一周。

Availability report graph



条形图中的绿色表示正常时间，红色表示问题时间。

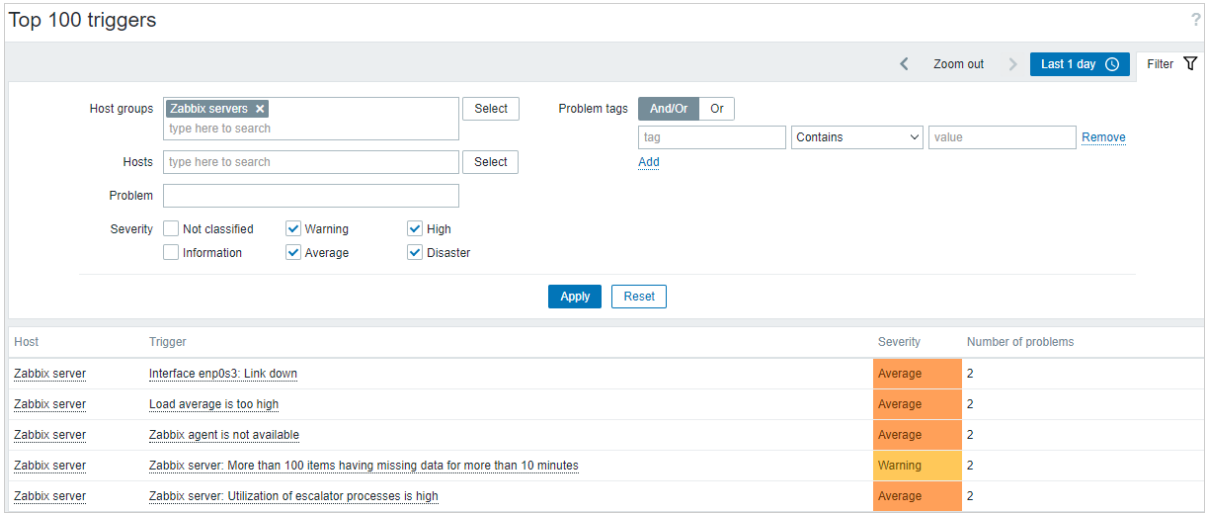
维护周期的影响

维护不会自动将时间从可用性报告中排除。只有将维护配置为 *No data collection* 时，才会停止轮询（因此不会生成问题），从而使受影响的触发器的可用性百分比保持不变。

前 **100** 个触发器

概述

在报表 > 排名前 *100* 的触发器中，您可以查看在所选时间段内检测到问题数量最多的触发器。



主机和触发器列中的条目都是链接，可提供一些实用选项：

- 对于主机 - 单击主机名称可打开主机菜单
- 对于触发器 - 单击触发器名称可打开指向最新事件的链接、每个触发器监控项的简单图形，以及触发器本身和每个触发器监控项的配置表单

使用筛选器

您可以使用筛选器按主机组、主机、问题名称、标签或触发器严重性显示触发器。指定父主机组后，会隐式选择所有嵌套的主机组。为提高搜索性能，系统会在宏未解析的情况下搜索数据。

筛选器位于排名前 *100* 的触发器部分名称下方。单击右侧的筛选器选项卡可以展开或折叠筛选器。

时间段选择器

筛选器旁边的时间段选择器允许您通过单击鼠标选择常用的时间段。

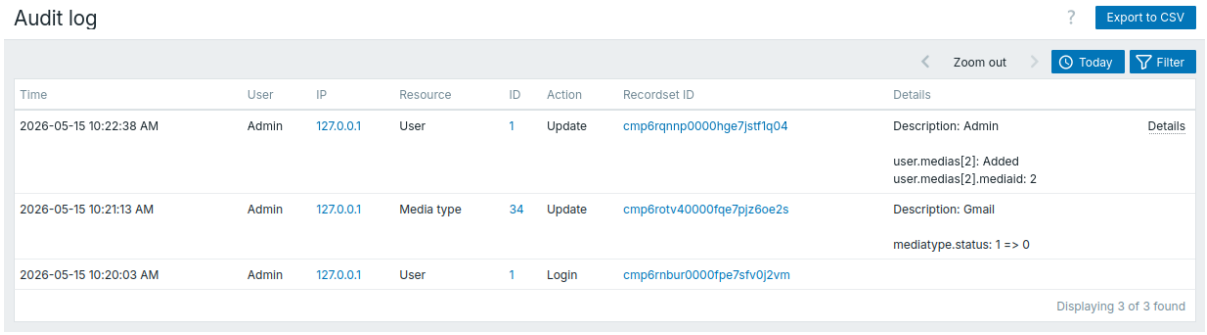
有关更多信息，请参阅时间段和主机选择器。

审计日志

概述

在 *Reports* > *Audit log* 部分，可以查看用户和系统活动的记录。

Note:
要收集并显示审计记录，必须在 *Administration* > *Audit log* 部分勾选 *Enable audit logging* 复选框。如果未启用此设置，活动历史将不会记录到数据库中，也不会显示在审计日志中。



审计日志显示以下数据：

Column	Description
Time	审计记录的时间戳。
User	执行该活动的用户。
IP	发起该活动的 IP。 点击该超链接将按此 IP 过滤审计日志记录。
Resource	受影响资源的类型（*API token*、*Action*、*Authentication*、*Autoregistration* 等）。

Column	Description
ID	受影响资源的 ID。 点击该超链接将按此资源 ID 过滤审计日志记录。
Action	活动类型（*Add*、*Configuration refresh*、*Delete*、*Execute*、*Failed login*、*History clear*、*Login*、*Logout*、*Push*、*Update*）。
Recordset ID	由同一操作生成的所有审计日志记录共享的 ID。 例如，将模板链接到主机时，每个继承的模板实体（监控项、触发器等）都会创建一条单独的审计日志记录 - 所有这些记录都将具有相同的 *Recordset ID*。 点击该超链接将按此 *Recordset ID* 过滤审计日志记录。
Details	资源的描述以及所执行活动的详细信息。 如果某条记录包含超过两行，将显示额外的 *Details* 链接。点击此链接可查看完整的更改列表。

Note:
当trapper item 或HTTP agent item（已启用 trapping）接收到数据时，只有在数据是通过`history.push` API 方法发送，而不是通过Zabbix sender 工具发送的情况下，才会在审计日志中添加一条记录。

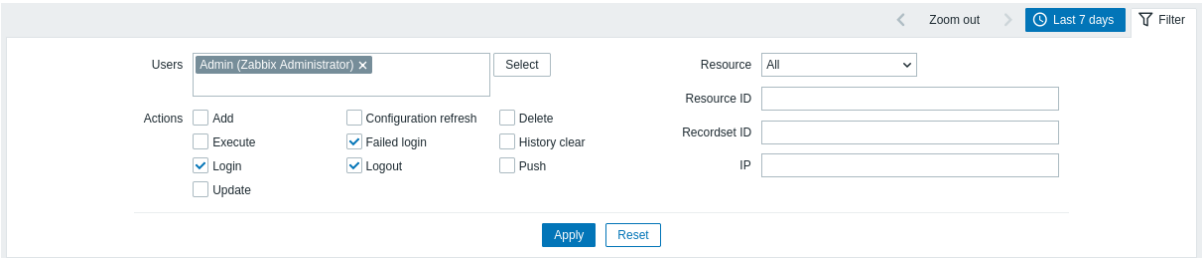
按钮

页面右上角的按钮提供以下选项：

Export to CSV	将所有页面中的审计日志记录导出为 CSV 文件。如果应用了过滤器，则只会导出筛选后的记录。

使用筛选器

筛选器位于 审计日志栏下方。单击右上角的 筛选器选项卡即可将其展开或折叠。



您可以使用筛选器按用户、受影响的资源、资源 ID、执行的操作（*Recordset ID*）和 IP 缩小记录范围。根据资源的不同，还可以在筛选器中选择一个或多个特定操作。

为了获得更好的搜索性能，所有数据在搜索时都不会解析宏。

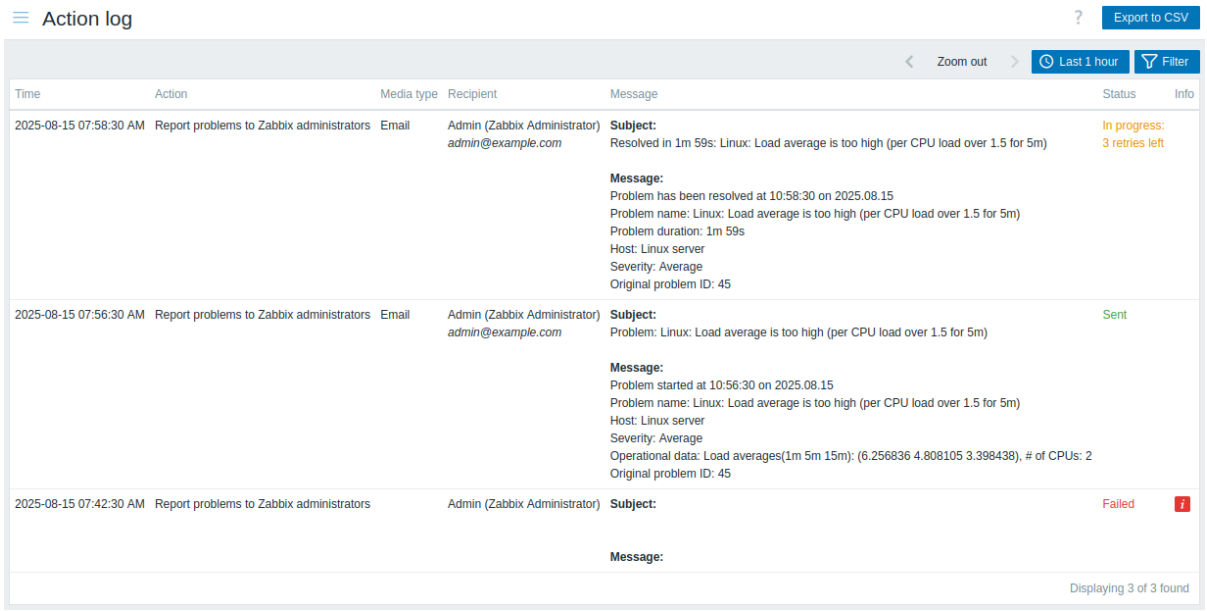
时间段选择器

筛选器旁边的时间段选择器允许您通过单击鼠标选择常用的时间段。
有关更多信息，请参阅时间段和主机选择器。

操作日志

概述

在 *Reports* > *Action log* 中，您可以查看在某个 action 中执行的operations（通知、远程命令）的详细信息。



显示的数据：

Column	Description
Time	操作的时间戳。
Action	导致执行操作的 action 名称。
Media type	用于发送通知的媒体类型（例如 Email、Jira 等）。 对于执行了远程命令的操作，此列为空。
Recipient	通知接收者的信息 - 用户名、姓名和姓氏（括号中），以及根据媒体类型提供的附加信息（email、用户名等）。 对于执行了远程命令的操作，此列为空。
Message	消息内容、远程命令或全局脚本名称。 远程命令与目标主机之间用冒号分隔：`<host>:<command>`。例如，如果远程命令是在 Zabbix server 上执行的，则信息格式如下：`Zabbix server:<command>`。
Status	操作状态： *In progress* - 发送通知的操作正在进行中（同时还会显示服务器剩余尝试发送通知的次数） *Sent* - 通知已发送 *Executed* - 远程命令已执行 *Failed* - 操作未完成
Info	有关操作执行的错误信息（如有）。

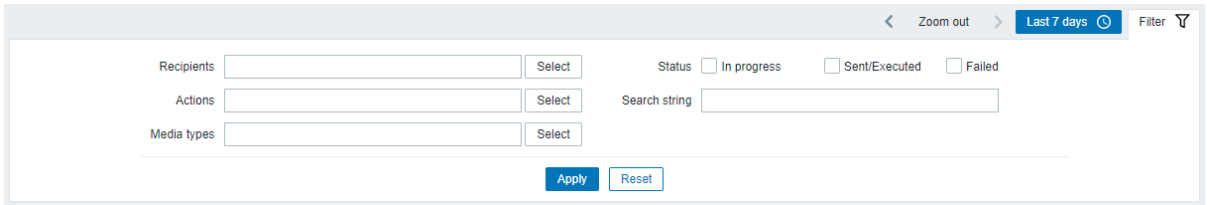
按钮

页面右上角的按钮提供以下选项：

Export to CSV	将当前页面中的动作日志记录导出为 CSV 文件。如果应用了筛选条件，则仅导出筛选后的记录。 在导出的 CSV 文件中，“Recipient” 和 “Message” 列会被拆分为多个列——“Recipient’s Zabbix username”、“Recipient’s name”、“Recipient’s surname”、“Recipient” 以及 “Subject”、“Message”、“Command”。

使用筛选器

筛选器位于 *Action log* 栏下方。可以通过单击页面右上角的 *Filter* 选项卡来展开或折叠它。



您可以使用筛选器按通知接收者、动作、媒体类型、状态，或按消息/远程命令内容（*Search string*）缩小记录范围。为获得更好的搜索性能，数据搜索时不会解析宏。

时间段选择器

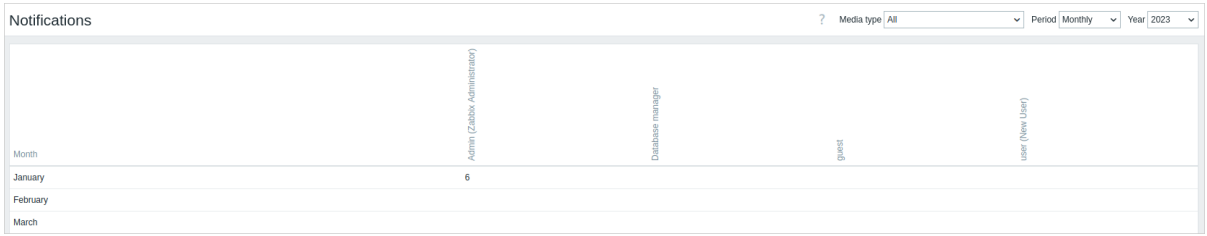
筛选器旁边的时间段选择器允许您通过单击鼠标选择常用的时间段。有关更多信息，请参阅时间段和主机选择器。

通知

概述

在 *Reports* > *Notifications* 部分，会显示一份关于发送给每个用户的通知数量的报告。

在右上角的下拉菜单中，您可以选择媒体类型（或全部）、周期（按天/周/月/年显示的数据）以及发送通知的年份。



每一列显示一个系统用户的总计。

数据采集

概览

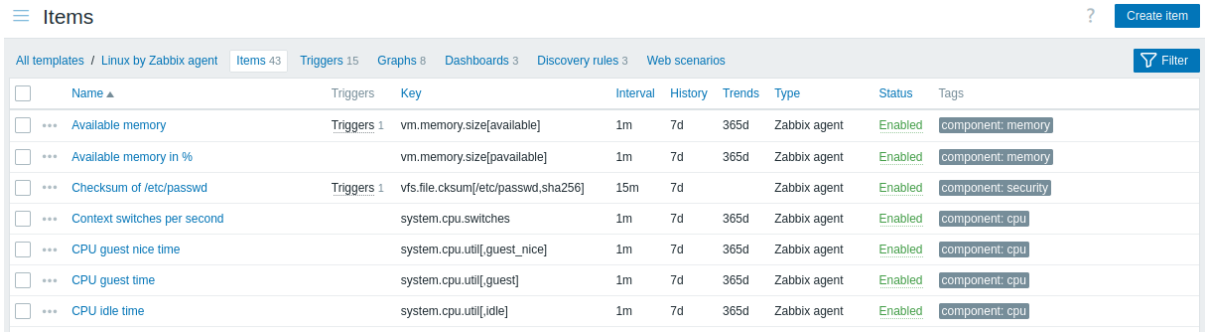
该菜单包含与配置数据收集相关的部分。

监控项

概述

可以从数据采集 > 模板访问模板的监控项列表，方法是单击相应模板的监控项。

此时会显示现有监控项列表。



显示的数据：

列	说明
监控项菜单	单击三个点图标打开监控项菜单。
模板	监控项所属的模板。 单击模板名称可打开模板配置表单。 仅当筛选器中选择了多个模板或未选择模板时，才会显示此列。
名称	监控项的名称，以蓝色链接形式显示，指向监控项详细信息。 单击监控项名称链接可打开监控项配置表单。 如果监控项继承自其他模板，则会在监控项名称前以灰色链接显示模板名称。单击模板链接可打开该模板层级的监控项列表。
触发器	将鼠标移到“触发器”上时，会显示一个信息框，其中列出与该监控项关联的触发器。 触发器数量以灰色显示。
键值	显示监控项键值。
间隔	显示检查频率。
历史	显示监控项数据历史记录的保留天数。
趋势	显示监控项趋势历史记录的保留天数。

列	说明
类型	显示监控项类型（Zabbix agent、SNMP agent、简单检查等）。
状态	显示监控项状态 - 已启用或已禁用。单击状态即可更改状态 - 从 “已启用” 更改为 “已禁用”（反之亦然）。
标签	显示监控项标签。 最多可显示三个标签（名称: 值对）。如果标签超过三个，则会显示 “...” 链接，将鼠标移到该链接上即可查看所有标签。

要配置新的监控项，请单击右上角的创建监控项按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

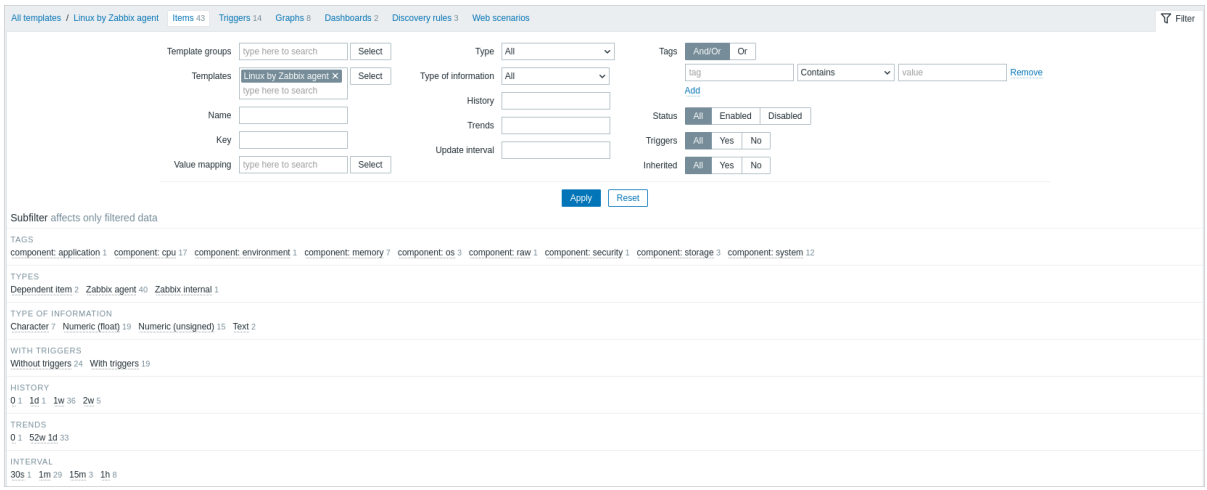
- 启用 - 将监控项状态更改为 已启用。
- 禁用 - 将监控项状态更改为 已禁用。
- 复制 - 将监控项复制到其他主机或模板。
- 批量更新 - 一次更新多个属性，适用于多个监控项。
- 删除 - 删除监控项。

要使用这些选项，请先勾选相应监控项前面的复选框，然后单击所需按钮。

使用筛选器

监控项列表可能包含大量监控项。通过使用筛选器，您可以筛选掉其中一部分，以便快速定位您要查找的监控项。为了获得更好的搜索性能，数据搜索时不会解析宏。

右上角提供了 *Filter* 图标。单击它将打开筛选器，您可以在其中指定所需的筛选条件。



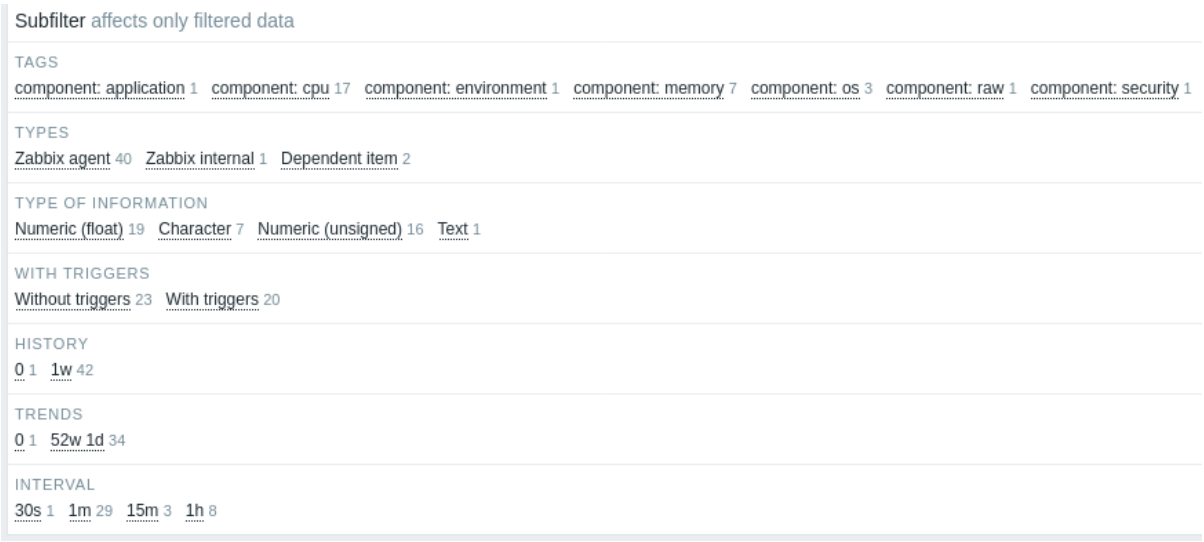
参数	描述
Template groups	按一个或多个模板组筛选。 指定父模板组会隐式选择所有嵌套组。
Templates	按一个或多个模板筛选。
Name	按监控项名称筛选。
Key	按监控项键值筛选。
Value mapping	按所使用的值映射筛选。 如果 *Templates* 选项为空，则不显示此参数。
Type	按监控项类型筛选（Zabbix agent、SNMP agent 等）。
Type of information	按信息类型筛选（无符号数值、浮点数等）。
History	按监控项历史数据的保留时长筛选。
Trends	按监控项趋势数据的保留时长筛选。
Update interval	按监控项更新间隔筛选。

参数	描述
Tags	指定标签以限制显示的监控项数量。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含标签值中含有输入字符串的指定标签名称（子字符串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除标签值中含有输入字符串的指定标签名称（子字符串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足任一条件即可
Status	按监控项状态筛选 - *Enabled* 或 *Disabled*。
Triggers	筛选带有（或不带有）触发器的监控项。
Inherited	筛选从关联模板继承（或未继承）的监控项。

使用子筛选器

子筛选器允许进一步修改主筛选器应用的筛选条件。

其中包含可点击的链接，可快速访问相关监控项。监控项通过共同实体相关联，例如标签、监控项类型、监控项更新间隔等。单击某个实体后，该实体会以灰色背景突出显示，并立即对监控项进行筛选（无需在主筛选器中单击应用）。单击另一个实体会将其添加到筛选结果中。再次单击该实体会移除相应的筛选条件。



子筛选器基于筛选后的数据生成，而筛选后的数据最多包含 1000 条记录。如果希望在子筛选器中查看更多记录，需要增大搜索和筛选结果限制参数的值（位于管理 > 常规 > *GUI*）。

与主筛选器不同，子筛选器会在每次表格刷新请求时更新，以始终显示可用筛选选项及其计数的最新信息。

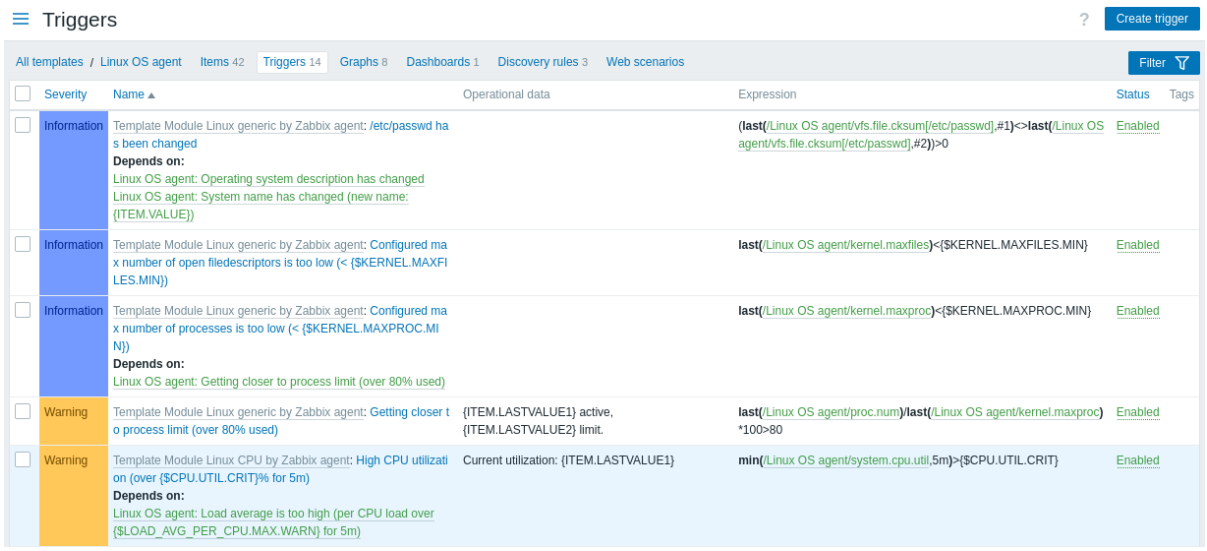
水平显示的实体数量最多为 100 个。如果实体数量更多，则末尾会显示三点图标；该图标不可点击。

每个可点击实体旁的数字表示其中包含的监控项数量（基于主筛选器的结果）。单击某个实体后，其他可用实体旁的数字会带有加号，表示可以添加到当前选择中的监控项数量。不包含监控项的实体不会显示，除非之前已在子筛选器中选中该实体。

触发器

概述

在数据采集 > 模板中，点击相应模板的触发器，即可访问模板的触发器列表。



显示的数据：

列	描述
严重性	触发器的严重性通过名称和单元格背景颜色显示。
模板	触发器所属的模板。 点击模板名称可打开模板配置表单。 仅当筛选器中选择了多个模板或未选择模板时，才会显示此列。
名称	触发器名称以蓝色链接显示，指向触发器详细信息。 点击触发器名称链接可打开触发器配置表单。 如果触发器继承自其他模板，则会在触发器名称前以灰色链接显示模板名称。点击模板链接可打开该模板级别的触发器列表。
运行数据	触发器的运行数据定义，其中包含任意字符串和宏，这些内容将在监控 > 问题中动态解析。
表达式	显示触发器表达式。表达式中的模板-监控项部分以链接形式显示，指向监控项配置表单。
状态	显示触发器状态 - 启用或禁用。点击状态即可更改状态 - 从启用改为禁用（反之亦然）。
标签	如果触发器包含标签，则在此列中显示标签名称和值。

要配置新触发器，请点击右上角的创建触发器按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

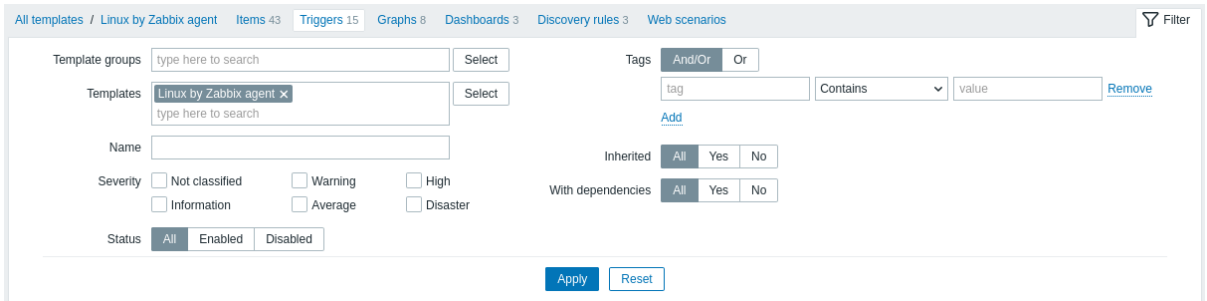
- *Enable* - 将触发器状态更改为 *Enabled*
- *Disable* - 将触发器状态更改为 *Disabled*
- *Copy* - 将触发器复制到其他主机或模板
- *Mass update* - 一次更新多个触发器的若干属性
- *Delete* - 删除触发器

要使用这些选项，请勾选相应触发器前面的复选框，然后单击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的触发器。为了获得更好的搜索性能，数据搜索时宏不会展开。

右上角提供了 *Filter* 图标。单击它将打开过滤器，您可以在其中指定所需的筛选条件。



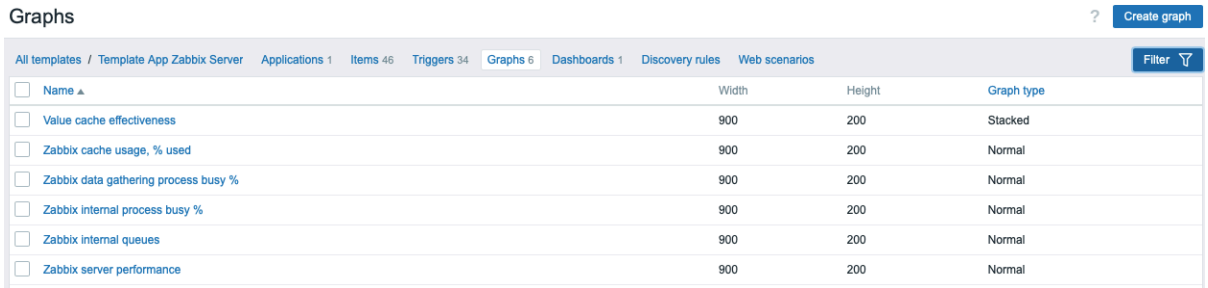
Parameter	Description
Template groups	按一个或多个模板组进行筛选。 指定父模板组会隐式选择所有嵌套组。
Templates	按一个或多个模板进行筛选。 如果上方已选择模板组，则模板选择将仅限于这些组。
Name	按触发器名称筛选。
Severity	选择按一个或多个触发器严重性进行筛选。
Status	按触发器激活状态（Enabled/Disabled）筛选。
Tags	按触发器标签名称和值进行筛选。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件有以下几种可用运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除指定的标签名称，其中标签值包含输入的字符串（子字符串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足任一条件即可 标签名称和标签值字段支持宏和macro functions。
Inherited	筛选从关联模板继承的（或未继承的）触发器。
With dependencies	筛选具有（或不具有）依赖关系的触发器。

图形

概述

可以从数据采集 > 模板访问模板的自定义图形列表，方法是点击相应模板的图形。

系统将显示现有图形列表。



显示的数据：

列	描述
模板	图形所属的模板。 点击模板名称可打开模板配置表单。 仅当筛选器中选择了多个模板或未选择模板时，才会显示此列。
名称	自定义图形的名称，以蓝色链接形式显示，指向图形详细信息。 点击图形名称链接可打开图形配置表单。 如果图形继承自其他模板，则会在图形名称前以灰色链接显示模板名称。点击模板链接可打开该模板级别的图形列表。
宽度	显示图形宽度。
高度	显示图形高度。
图形类型	显示图形类型 - 普通、堆叠、饼图或分离型。

要配置新图形，请点击右上角的创建图形按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 复制 - 将图形复制到其他主机或模板
- 删除 - 删除图形

要使用这些选项，请先勾选相应图形前面的复选框，然后点击所需的按钮。

使用筛选器

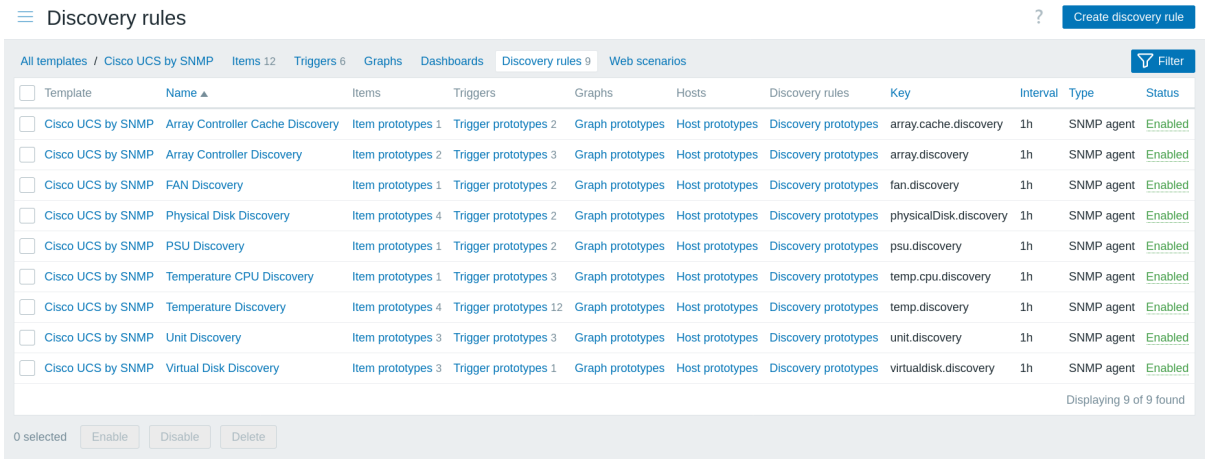
您可以按模板组和模板筛选图形。为了获得更好的搜索性能，数据会在宏未解析的情况下进行搜索。

发现规则

概述

可以在数据采集 > 模板中，点击相应模板的发现来访问模板的低级发现规则列表。

系统将显示现有低级发现规则的列表。通过更改筛选条件，也可以查看与模板无关的所有发现规则，或特定模板组中的所有发现规则。



显示的数据：

列	描述
模板	低级发现规则所属的模板。 点击模板名称可打开模板配置表单。
名称	规则名称，以蓝色链接显示。 点击规则名称可打开低级发现规则配置表单。 如果发现规则继承自其他模板，则模板名称会以灰色链接显示在规则名称之前。点击模板链接可打开该模板级别的发现规则列表。
监控项	显示监控项原型列表的链接。 现有监控项原型的数量以灰色显示。
触发器	显示触发器原型列表的链接。 现有触发器原型的数量以灰色显示。
图形	显示图形原型列表的链接。 现有图形原型的数量以灰色显示。
主机	显示主机原型列表的链接。 现有主机原型的数量以灰色显示。
键	显示用于发现的监控项键。
间隔	显示执行发现的频率。
类型	显示用于发现的监控项类型（Zabbix agent、SNMP agent 等）。
状态	显示发现规则的状态 - 已启用或已禁用。点击状态可以进行切换，即从 “已启用” 切换为 “已禁用”（反之亦然）。

要配置新的低级发现规则，请点击右上角的创建发现规则按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

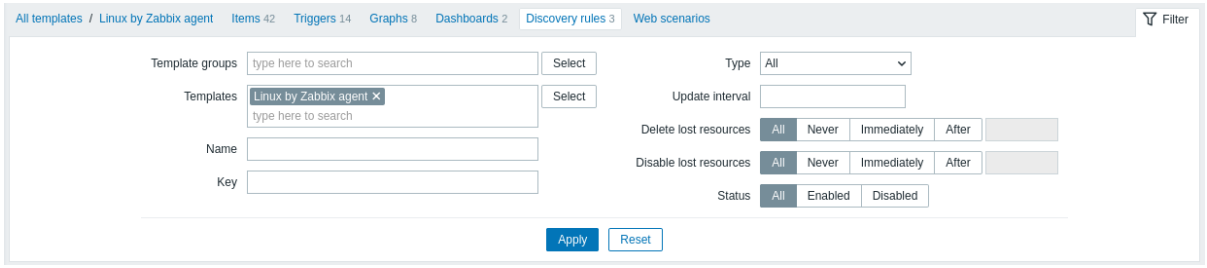
- 启用 - 将低级别发现规则状态更改为 已启用
- 禁用 - 将低级别发现规则状态更改为 已禁用
- 删除 - 删除低级别发现规则

要使用这些选项，请先勾选相应发现规则前面的复选框，然后单击所需的按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的发现规则。为了获得更好的搜索性能，数据搜索时不会解析宏。

右上角提供了 *Filter* 图标。单击它将打开一个过滤器，您可以在其中指定所需的筛选条件，例如模板、发现规则名称、监控项键、监控项类型等。



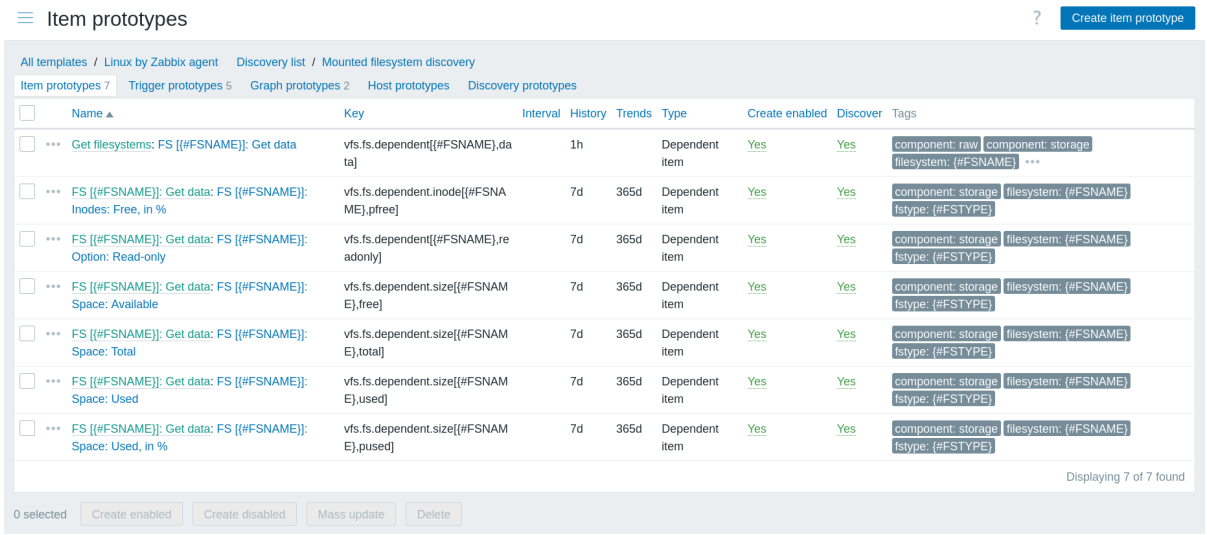
Parameter	Description
Template groups	按一个或多个模板组进行筛选。 指定父模板组会隐式选择所有嵌套组。
Templates	按一个或多个模板进行筛选。
Name	按发现规则名称进行筛选。
Key	按发现监控项键进行筛选。
Type	按发现监控项类型进行筛选。
Update interval	按更新间隔进行筛选。 不适用于 Zabbix trapper 和依赖监控项。
Delete lost resources	按 *Delete lost resources* 周期进行筛选。
Disable lost resources	按 *Disable lost resources* 周期进行筛选。
Status	按发现规则状态进行筛选（All/Enabled/Disabled）。

监控项原型

概述

本节显示模板上低级别发现规则中已配置的监控项原型。

如果模板已链接到主机，监控项原型将作为低级别发现期间创建实际主机监控项的基础。



显示的数据：

列	描述
名称	显示监控项原型的名称，并以蓝色链接显示。 单击名称可打开监控项原型配置表单。 如果监控项原型属于已链接的模板，则会在监控项原型名称前显示模板名称，并以灰色链接显示。单击模板链接可打开已链接模板级别的监控项原型列表。 如果监控项原型的类型为“依赖监控项”，则会在监控项原型名称前以绿色显示主监控项名称（与主机级别上的监控项原型相同）。
键值	显示监控项原型的键值。
间隔	显示检查频率。
历史	显示监控项数据历史记录的保留天数。
趋势	显示监控项趋势历史记录的保留天数。
类型	显示监控项原型的类型（Zabbix agent、SNMP agent、简单检查等）。
创建已启用	根据此原型创建监控项： 是 - 已启用 否 - 已禁用。单击“是”和“否”可在两者之间切换。
发现	根据此原型发现监控项： 是 - 发现 否 - 不发现。单击“是”和“否”可在两者之间切换。
标签	显示监控项原型的标签。

要配置新的监控项原型，请单击右上角的创建监控项原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 创建启用 - 将这些监控项原型创建为 启用
- 创建禁用 - 将这些监控项原型创建为 禁用
- 批量更新 - 批量更新这些监控项原型
- 删除 - 删除这些监控项原型

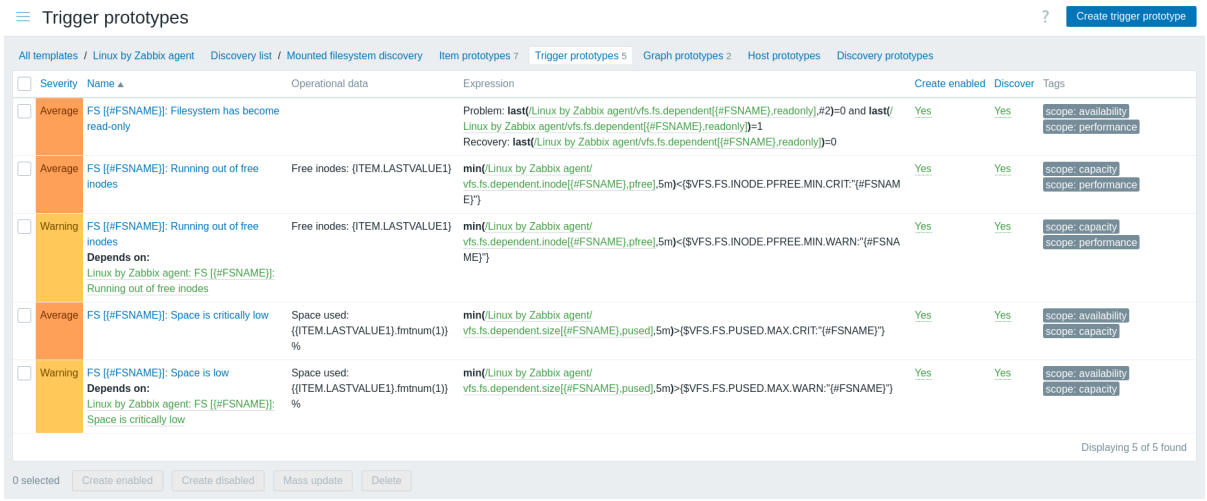
要使用这些选项，请先勾选相应监控项原型前面的复选框，然后单击所需的按钮。

触发器原型

概述

本节显示模板上低级别发现规则中已配置的触发器原型。

如果模板已链接到主机，触发器原型将作为基础，在低级别发现期间创建实际的主机触发器。



显示的数据：

列	描述
严重性	触发器的严重性同时通过名称和单元格背景颜色显示。

列	描述
名称	触发器原型的名称，以蓝色链接显示。 单击名称可打开触发器原型配置表单。 如果触发器原型属于已链接的模板，则模板名称会以灰色链接显示在触发器名称之前。单击模板链接可打开已链接模板级别的触发器原型列表。
表达式	显示触发器表达式。表达式中的模板监控项部分以链接形式显示，指向监控项配置表单。
运行数据	显示触发器运行数据的格式，其中包含任意字符串和宏，这些内容将在监测 > 问题中动态解析。
创建已启用	根据此原型创建触发器： 是 - 已启用 否 - 已禁用。单击 “是” 和 “否” 即可在两者之间切换。
发现	根据此原型发现触发器： 是 - 发现 否 - 不发现。单击 “是” 和 “否” 即可在两者之间切换。
标签	显示触发器原型的标签。

要配置新的触发器原型，请单击右上角的创建触发器原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 创建为启用 - 将这些触发器原型创建为 启用
- 创建为禁用 - 将这些触发器原型创建为 禁用
- 批量更新 - 批量更新这些触发器原型
- 删除 - 删除这些触发器原型

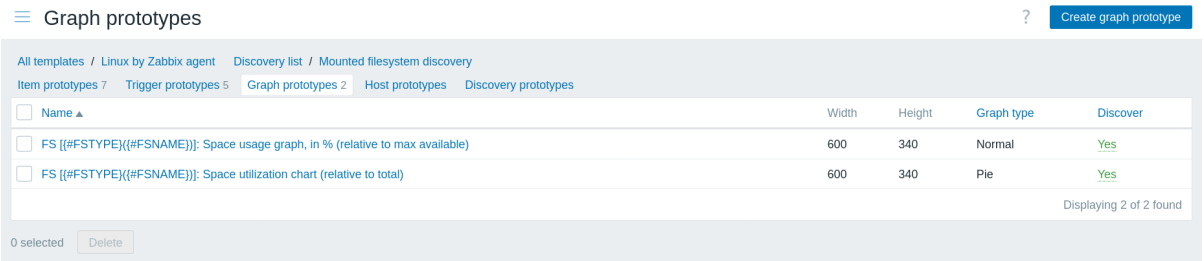
要使用这些选项，请先勾选相应触发器原型前面的复选框，然后单击所需的按钮。

图形原型

概述

本节显示模板上低级别发现规则中已配置的图形原型。

如果模板已链接到主机，图形原型将作为低级别发现期间创建实际主机图形的基础。



显示的数据：

列	描述
名称	图形原型的名称，以蓝色链接显示。 单击名称可打开图形原型配置表单。 如果图形原型属于已链接的模板，则模板名称会作为灰色链接显示在图形名称之前。单击模板链接可打开已链接模板级别的图形原型列表。
宽度	显示图形原型的宽度。
高度	显示图形原型的高度。
类型	显示图形原型的类型 - 普通、堆叠、饼图或分离型。
发现	根据此原型发现图形： 是 - 发现 否 - 不发现。单击 “是” 和 “否” 可在两者之间切换。

要配置新的图形原型，请单击右上角的创建图形原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 删除 - 删除这些图形原型

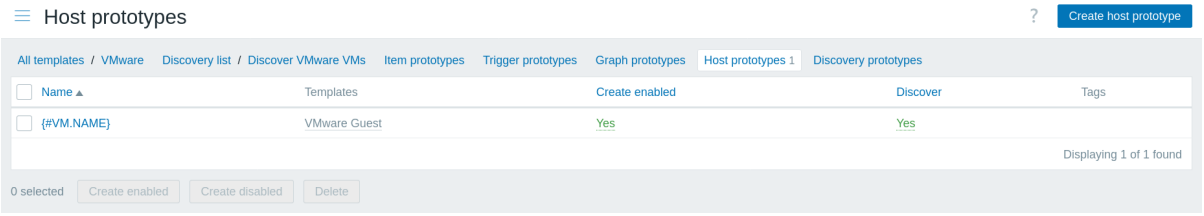
要使用这些选项，请先勾选相应图形原型前面的复选框，然后单击所需的按钮。

主机原型

概述

本节显示模板上低级别发现规则中已配置的主机原型。

如果模板已链接到主机，主机原型将作为在低级别发现期间创建实际主机的基础。



显示的数据：

Column	Description
名称	主机原型的名称，以蓝色链接显示。 单击名称可打开主机原型配置表单。 如果主机原型属于已链接的模板，则模板名称会以灰色链接显示在主机名称之前。单击模板链接可打开已链接模板级别上的主机原型列表。
模板	显示主机原型的模板。
创建已启用	根据此原型创建主机： 是 - 已启用 否 - 已禁用。单击 “是” 或 “否” 可在两者之间切换。
发现	根据此原型发现主机： 是 - 发现 否 - 不发现。单击 “是” 或 “否” 可在两者之间切换。
标签	显示主机原型的标签。

要配置新的主机原型，请单击右上角的创建主机原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 创建为启用 - 将这些主机创建为 启用
- 创建为禁用 - 将这些主机创建为 禁用
- 删除 - 删除这些主机原型

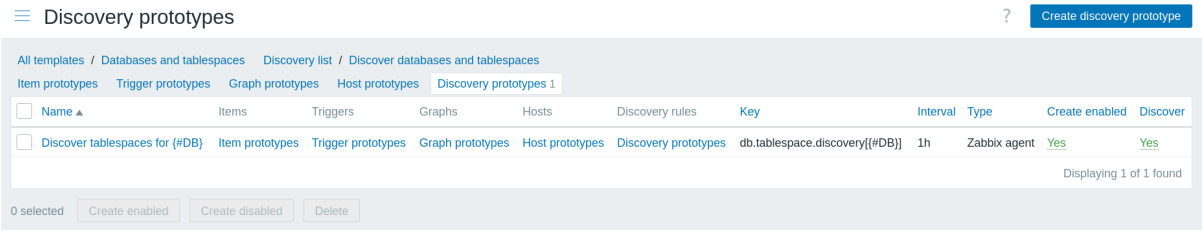
要使用这些选项，请先勾选相应主机原型前的复选框，然后单击所需的按钮。

发现原型

概述

本节显示模板上低级别发现规则中已配置的发现原型。

如果模板已链接到主机，发现原型将作为创建父发现规则的实际发现原型的基础。



显示的数据：

列	描述
名称	发现原型的名称，以蓝色链接显示。 单击名称可打开发现原型配置表单。 如果发现原型属于已链接的模板，则模板名称会以灰色链接显示在发现原型名称之前。单击模板链接可打开已链接模板级别上的发现原型列表。
监控项	显示指向监控项原型列表的链接。 现有监控项原型的数量以灰色显示。
触发器	显示指向触发器原型列表的链接。 现有触发器原型的数量以灰色显示。
图形	显示指向图形原型列表的链接。 现有图形原型的数量以灰色显示。
主机	显示指向主机原型列表的链接。 现有主机原型的数量以灰色显示。
发现规则	显示指向发现原型列表的链接。 现有发现原型的数量以灰色显示。
键值	显示用于发现的监控项键值。
间隔	显示执行发现的频率。
类型	显示用于发现的监控项类型（Zabbix agent、SNMP agent 等）。
启用创建	显示发现原型创建状态 - 启用创建（是）或禁用创建（否）。单击状态可在 “是” 和 “否” 之间切换。
发现	显示发现原型的发现状态 - 发现（是）或不发现（否）。单击状态可在 “是” 和 “否” 之间切换。

要配置新的发现原型，请单击右上角的创建发现原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 创建为启用 - 将这些发现原型创建为 启用
- 创建为禁用 - 将这些发现原型创建为 禁用
- 删除 - 删除这些发现原型

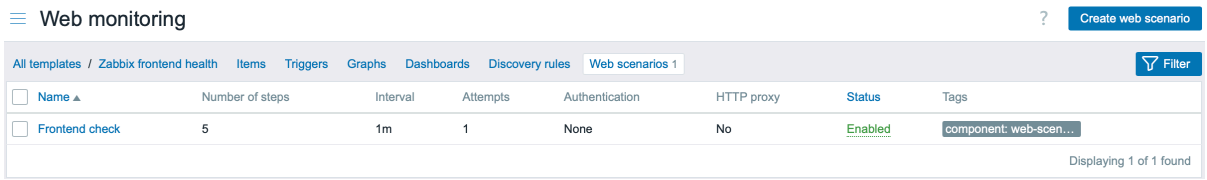
要使用这些选项，请先勾选相应发现原型前面的复选框，然后单击所需按钮。

Web 场景

概述

可以从数据采集 > 模板访问模板的Web 场景列表，方法是单击相应模板的 *Web*。

系统将显示现有 Web 场景的列表。



显示的数据：

列	描述
名称	Web 场景的名称。单击 Web 场景名称可打开Web 场景配置表单。 如果 Web 场景继承自其他模板，则模板名称会以灰色链接的形式显示在 Web 场景名称之前。单击模板链接可打开该模板级别的 Web 场景列表。
步骤数	场景包含的步骤数。
更新间隔	执行场景的频率。
尝试次数	执行 Web 场景步骤时的尝试次数。
身份验证	显示身份验证方法 - Basic、NTLM、Kerberos、Digest 或 None。
HTTP proxy	显示 HTTP proxy；如果未使用，则显示 “No”。
状态	显示 Web 场景状态 - 已启用或已禁用。 单击状态即可更改状态。

列	描述
标签	显示 Web 场景标签。 最多可显示三个标签（名称: 值对）。如果标签超过三个，则会显示 “...” 链接，将鼠标悬停在该链接上即可查看所有标签。

要配置新的 Web 场景，请单击右上角的创建 *Web* 场景按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

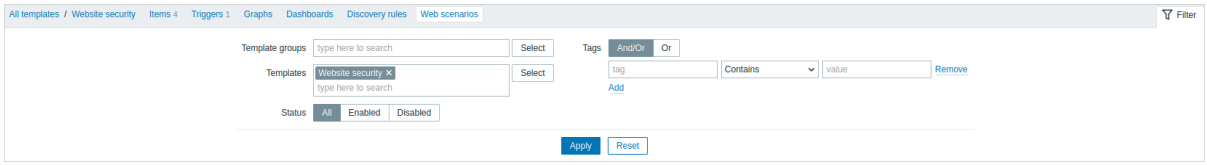
- 启用 - 将场景状态更改为 已启用
- 禁用 - 将场景状态更改为 已禁用
- 删除 - 删除 web 场景

要使用这些选项，请先勾选相应 web 场景前面的复选框，然后单击所需的按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的场景。为了获得更好的搜索性能，数据会在宏未解析的情况下进行搜索。

Filter 链接位于 web 场景列表上方。单击它后，会显示一个过滤器，您可以按模板组、模板、状态和标签筛选场景。

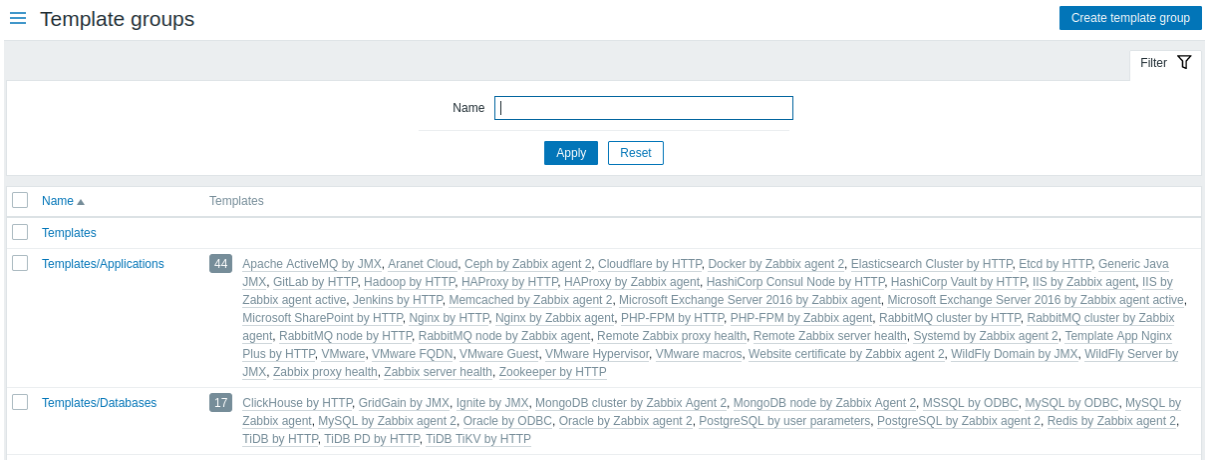


模板组

概述

在数据采集 > 模板组部分，用户可以配置和维护模板组。

页面会显示现有模板组及其详细信息列表。您可以按名称搜索和筛选模板组。



显示的数据：

列	描述
名称	模板组的名称。单击组名称可打开模板组配置表单。
模板	组中的模板数量（以灰色显示），后跟组成员列表。 单击模板名称可打开模板配置表单。 单击数量可打开此组中的模板列表。

批量编辑选项

要一次删除多个模板组，请勾选相应组前面的复选框，然后单击列表下方的 Delete 按钮。

使用过滤器

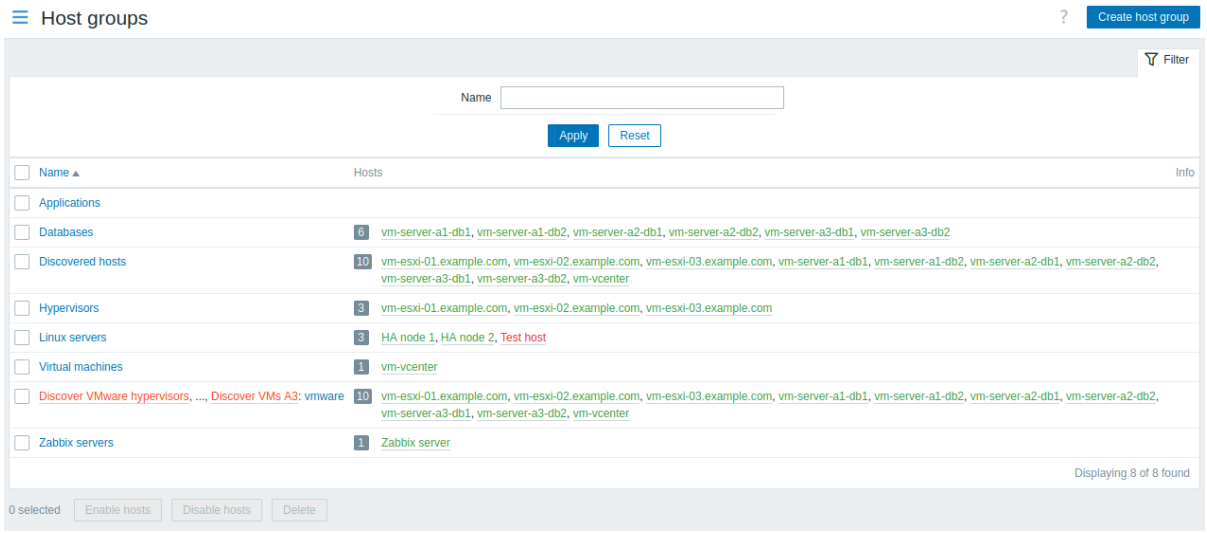
您可以使用过滤器仅显示您感兴趣的模板组。为了提高搜索性能，数据是在未解析宏的情况下进行搜索的。这意味着搜索将针对原始的、未经宏替换的字符串进行匹配。

主机组

概述

在数据采集 > 主机群组部分，用户可以配置和维护主机群组。

页面会显示现有主机群组及其详细信息的列表。您可以按名称搜索和筛选主机群组。



显示的数据：

列	描述
名称	主机群组的名称。单击群组名称可打开主机群组配置表单。 已发现的主机群组会以低级别发现规则名称作为前缀显示。单击 LLD 规则名称可打开主机原型配置表单。请注意，当发现该主机群组的所有 LLD 规则都被删除后，已发现的主机群组也会被删除。
主机	群组中的主机数量（以灰色显示），后面跟有群组成员列表。 单击主机名称可打开主机配置表单。 单击数量可在主机完整列表中筛选出属于该群组的主机。
信息	显示与主机群组相关的错误信息（如果有）。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 启用主机 - 将组中所有主机的状态更改为”Monitored”
- 禁用主机 - 将组中所有主机的状态更改为”Not monitored”
- 删除 - 删除主机组

要使用这些选项，请先勾选相应主机组前面的复选框，然后单击所需按钮。

使用过滤器

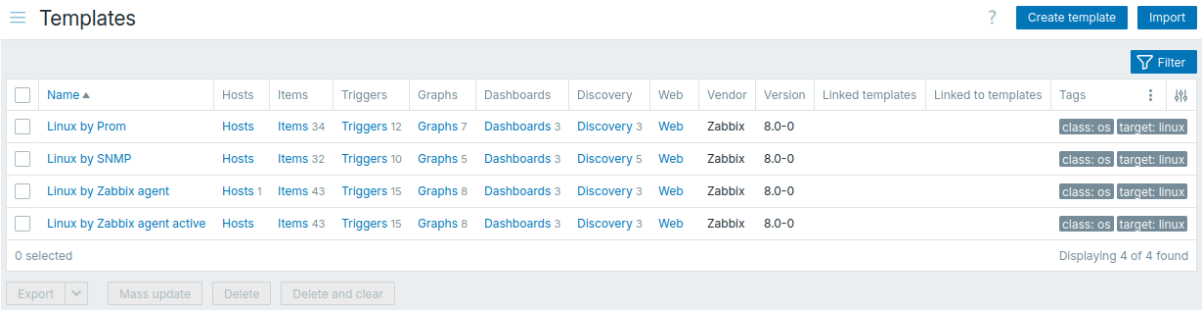
你可以通过使用过滤器仅仅展示你感兴趣的主机组。为了更好的搜索性能，搜索数据的时候不会解析宏

模板

概述

在数据采集 > 模板部分，用户可以配置和维护模板。

页面会显示现有模板及其详细信息列表。



模板详细信息显示在可自定义的表格中。表头中的 自定义表格按钮可用于显示、隐藏或重新排列列，也可以将布局重置为默认设置。您还可以通过拖动列的右边框来调整列宽；双击边框可重置列宽。所有自定义设置会立即应用，并且始终为当前用户保留。

可用列如下：

列	描述
名称	模板的名称。 单击模板名称可打开模板配置表单。
主机	与模板关联的可编辑主机数量；不包括只读主机。 单击主机将打开主机列表，其中仅筛选出与该模板关联的主机。
实体（监控项、触发器、图形、仪表板、发现、*Web*）	模板中相应实体的数量（以灰色显示）。 单击实体名称后，在该实体的完整列表中筛选出属于该模板的实体。
链接的模板	已链接到该模板的模板。
链接到的模板	该模板已链接到的模板。
厂商、版本	模板的厂商和版本；如果模板配置包含此类信息，则显示该信息，并且仅适用于开箱即用模板、导入的模板或通过Template API修改的模板。 对于开箱即用模板，版本显示格式如下：Zabbix 主版本号、分隔符（“-”）、修订号（每次发布模板的新版本时递增，并在每个 Zabbix 主版本中重置）。例如：7.0-0、7.0-3、8.0-0、8.0-3。
标签	显示模板的标签，但不解析宏。 将鼠标悬停在显示的标签后面的 三点图标上或单击该图标，可打开包含所有标签的弹出窗口。 单击列标题中的 选项按钮，可以设置显示的标签数量（1–3 个）、选择标签名称的显示方式（完整、缩短或不显示）、设置标签显示优先级，或复制该列（复制的列可以重命名或删除）。
标签值	显示模板标签的值，但不解析宏。 单击列标题中的 选项按钮，可以重命名该列、选择要显示的标签值，或复制该列（复制的列可以重命名或删除）。
自定义文本	显示自定义文本、宏值或两者的组合。 单击列标题中的 选项按钮，可以重命名该列、输入自定义文本或宏，或复制该列（复制的列可以重命名或删除）。 支持的宏：{HOST.ID}、{HOST.HOST}、{HOST.NAME}、{HOST.DESCRIPTION} 以及用户宏。

要配置新模板，请单击右上角的创建模板按钮。

要从 YAML、XML 或 JSON 文件导入模板，请单击右上角的导入按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

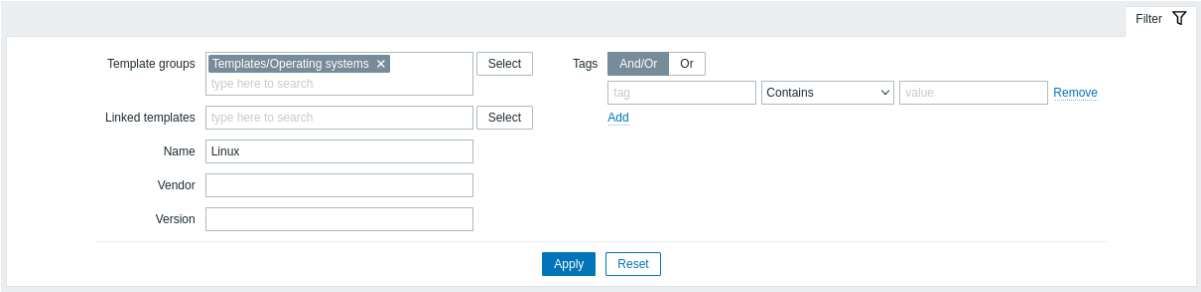
- 导出 - 将模板导出为 YAML、XML 或 JSON 文件；
- 批量更新 - 一次更新多个模板的多个属性；
- 删除 - 删除模板，同时将其关联实体（监控项、触发器等）保留在主机上；
- 删除并清除 - 删除模板及其在主机上的关联实体。

要使用这些选项，请勾选相应模板前面的复选框，然后单击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的模板。为了获得更好的搜索性能，数据搜索时宏不会解析。

Filter 链接位于 *Create template* 和 *Import* 按钮下方。单击后会显示过滤器，您可以按模板组、关联模板、名称和标签筛选模板。



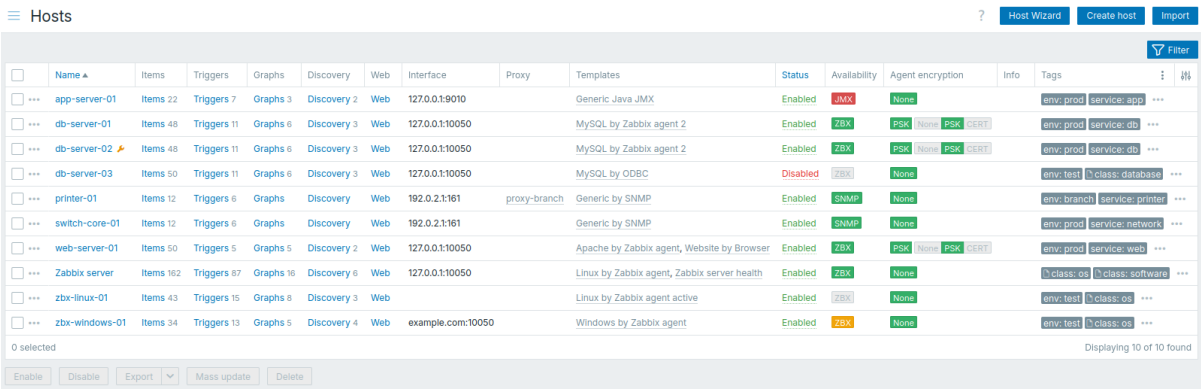
Parameter	Description
Template groups	按一个或多个模板组进行筛选。 指定父模板组会隐式选择所有嵌套组。
Linked templates	按直接关联的模板进行筛选。
Name	按模板名称进行筛选。
Vendor	按模板供应商进行筛选。
Version	按模板版本进行筛选。
Tags	按模板标签名称和值进行筛选。 只能按模板级标签进行筛选（不包括继承的标签）。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件都有以下运算符可用： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含标签值中包含所输入字符串的指定标签名称（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除标签值中包含所输入字符串的指定标签名称（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组； **Or** - 满足任一条件即可。

主机

概述

在数据采集 > 主机部分，用户可以配置和维护主机。

页面会显示现有主机及其详细信息列表。



主机详细信息显示在可自定义的表格中。表头中的 自定义表格按钮可用于显示、隐藏或重新排列列，并将布局重置为默认设置。您还可以通过拖动列的右边框来调整列宽；双击边框可重置列宽。所有自定义设置会立即应用，并始终为当前用户保留。

可用列如下：

列	描述
名称	主机的名称。 单击主机名称可打开主机配置表单。 主机名称后的橙色扳手图标 表示该主机处于维护状态。将鼠标指针悬停在图标上时会显示维护详细信息。
实体（监控项、触发器、图形、发现、*Web*）	单击实体名称可显示该主机的监控项、触发器等。 相应实体的数量以灰色显示。
接口	显示主机的主要接口。
Proxy	此列显示已分配的 proxy： <Proxy 名称 > - 主机由独立 proxy 监控（即使该 proxy 属于 proxy 组）； <Proxy 组名称: proxy 名称 > - 主机由 proxy 组监控，且 Zabbix 服务器已分配一个 proxy 来监控该主机； <Proxy 组名称 > - 主机由不包含任何 proxy 的 proxy 组监控，或 Zabbix 服务器尚未为该主机分配 proxy； 无 - 主机既不由 proxy 监控，也不由 proxy 组监控。 仅当监控方式筛选选项设置为 “任意”、“Proxy” 或 “Proxy 组” 时，才会显示此列。
模板	显示链接到主机的模板。 如果所链接的模板中包含其他模板，则这些模板会显示在括号中，并以逗号分隔。 单击模板名称可打开其配置表单。
状态	显示主机状态 - 已启用或已禁用。 单击状态可手动更改状态。 已丢失的已发现主机会标有信息图标。工具提示文本会提供其状态详细信息。
可用性	显示主机各已配置接口的可用性。 可用性图标表示主机接口在 Zabbix 服务器上的当前状态。因此，如果您在前端中禁用主机，其可用性将在 Zabbix 服务器同步配置更改后更新。同样，如果您启用主机，其可用性将在 Zabbix 服务器同步配置更改并轮询主机后更新。 可用性图标仅表示已配置的接口类型（Agent、SNMP、IPMI、JMX）。 将鼠标悬停在图标上时，会弹出一个窗口，其中列出同一类型的所有接口及其详细信息、状态和错误。 对于 Agent 接口，弹出窗口会显示接口（被动）和主动检查。如果主机仅配置了主动检查，即使主机未配置 Agent 接口，也会显示 Agent 接口图标。 没有接口的主机，此列为空。 单个主机接口的状态由使用该接口的已启用监控项与主机之间的连接决定。状态可以是： 可用 - 与主机的连接成功； 不可用 - 与主机的连接失败（超时、防火墙问题等）； 未知 - 尚未尝试连接主机，或结果未知。 有关 Zabbix 服务器如何确定接口状态的更多详细信息，请参阅未知接口状态和不可达/不可用主机接口设置。 单个类型（Agent、SNMP、IPMI、JMX）的所有主机接口的状态，由至少被一个已启用监控项使用的接口决定。状态通过图标颜色表示： 绿色 - 所有接口均可用； 黄色 - 至少一个接口不可用，且至少一个接口可用或状态未知； 红色 - 所有接口均不可用； 灰色 - 至少一个接口状态未知，但没有接口不可用。 主动检查可用性。如果主机上至少启用了一个主动检查，则主动检查也会按照上述说明影响 Agent 接口的总可用性。为确定主动检查的可用性，agent 主动检查线程会发送心跳消息。心跳消息的频率由 Zabbix agent 或agent 2 配置中的 `HeartbeatFrequency` 参数控制（默认 60 秒，范围为 0-3600）。 当主动检查心跳的时间早于 2 x HeartbeatFrequency 秒时，主动检查会被视为不可用。 注意：6.2.x 之前版本的 Zabbix agent 不会发送主动检查心跳，因此其主机的可用性会保持未知。
Agent 加密	显示与主机之间以及从主机发出的连接的加密状态： 无 - 不使用加密； **PSK** - 使用预共享密钥； 证书 - 使用证书。
信息	显示与主机相关的错误信息（如果有）。
标签	显示主机标签，宏不会解析。 将鼠标悬停在显示的标签后的 三点图标上或单击该图标，会弹出一个窗口，其中显示所有标签。 单击列标题中的 选项按钮，可以设置显示的标签数量（1–3）、选择标签名称的显示方式（完整、缩短或不显示）、设置标签显示优先级，或复制该列（复制的列可以重命名或删除）。

列	描述
标签值	显示主机标签值，宏不会解析。 单击列标题中的 选项按钮，可以重命名该列、选择要显示的标签值，或复制该列（复制的列可以重命名或删除）。
自定义文本	显示自定义文本、宏值或两者的组合。 单击列标题中的 选项按钮，可以重命名该列、输入自定义文本或宏，或复制该列（复制的列可以重命名或删除）。 支持的宏：{HOST.*}（{HOST.METADATA} 除外）、{INVENTORY.*} 以及用户宏。

要打开主机菜单，请单击主机名称前的 选项按钮。

要配置新主机，请单击右上角的主机向导或创建主机按钮。

要从 YAML、XML 或 JSON 文件导入主机，请单击右上角的导入按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

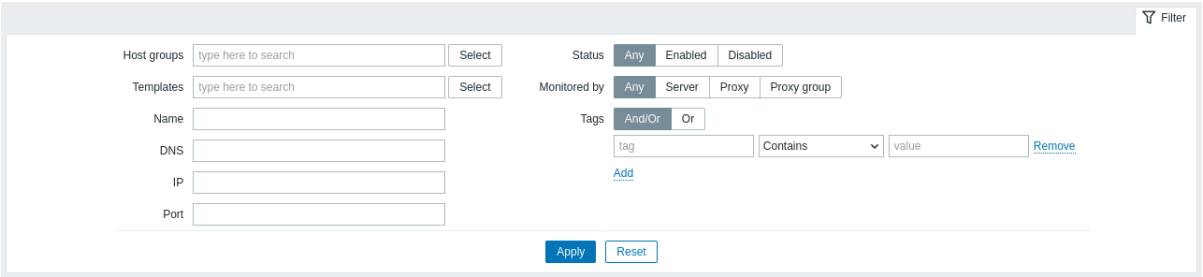
- 启用 - 将主机状态更改为 已监控；
- 禁用 - 将主机状态更改为 未监控；
- 导出 - 将主机导出为 YAML、XML 或 JSON 文件；
- 批量更新 - 一次性为多个主机更新若干属性；
- 删除 - 删除主机。

要使用这些选项，请先勾选相应主机前面的复选框，然后点击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的主机。为了提高搜索性能，数据搜索时不会解析宏。

右上角提供 *Filter* 图标。单击它将打开一个过滤器，您可以在其中指定所需的筛选条件。



Parameter	Description
Host groups	按一个或多个主机组进行筛选。 指定父主机组会隐式选择所有嵌套主机组。
Templates	按关联的模板进行筛选。
Name	按可见主机名称进行筛选。
DNS	按 DNS 名称进行筛选。
IP	按 IP 地址进行筛选。
Port	按端口号进行筛选。
Status	按主机状态进行筛选。
Monitored by	筛选由 Zabbix 服务器、proxy 或 proxy 组监控的主机。
Proxies	筛选由此处指定的 proxy 监控的主机。仅当 *Monitored by* 字段中选择了”Proxy” 时，此字段才可用。
Proxy groups	筛选由此处指定的 proxy 组监控的主机。仅当 *Monitored by* 字段中选择了”Proxy group” 时，此字段才可用。

Parameter	Description
Tags	按主机标签名称和值进行筛选。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件都有以下几种可用运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含标签值中含有输入字符串的指定标签名称（子字符串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除标签值中含有输入字符串的指定标签名称（子字符串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组； **Or** - 满足任一条件即可。

未知接口状态

在以下情况下，Zabbix 服务器会将主机接口（Agent、SNMP、IPMI、JMX）的状态显示为 “Unknown”：

- 主机已禁用。
- 主机被设置为由 proxy 监控，或由不同的 proxy 监控，或者如果之前由 proxy 监控，则改为由服务器监控。
- 主机由一个看起来已离线的 proxy 监控（在最长心跳间隔内未收到来自 proxy 的更新 - 1 小时）。
- 该接口类型对应的所有主机监控项都已禁用。
- 未配置该接口类型的轮询器（例如，`StartAgentPollers` 或`StartSNMPPollers` 服务器配置参数被设置为 0）。

在 Zabbix 服务器配置缓存同步后，接口可用性会被设置为 “Unknown”。

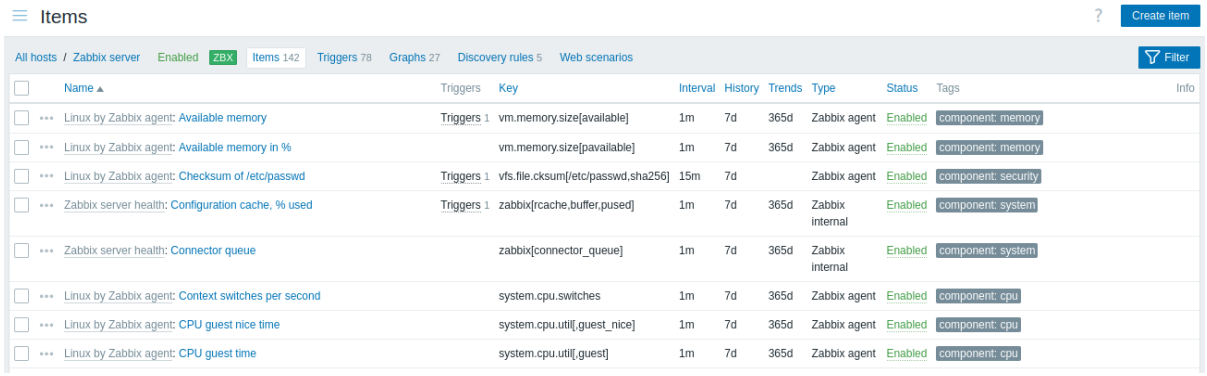
对于由 proxy 监控的主机，在 proxy 配置缓存同步后，接口可用性（Available/Not available）会恢复。

监控项

概述

可以从数据采集 > 主机访问主机的监控项列表，方法是点击相应主机的监控项。

系统将显示现有监控项的列表。



显示的数据：

列	描述
监控项菜单	点击三个点图标打开监控项菜单。
主机	监控项所属的主机。 点击主机名称可打开主机配置表单。 仅当筛选器中选择了多个主机或未选择主机时，才会显示此列。
名称	监控项的名称，以蓝色链接形式显示，指向监控项详细信息。 点击监控项名称链接可打开监控项配置表单。 如果主机监控项属于模板，则模板名称会以灰色链接形式显示在监控项名称之前。点击模板链接可打开模板级别的监控项列表。 如果监控项是从监控项原型创建的，则其名称前会以橙色显示低级别发现规则名称。点击发现规则名称可打开监控项原型列表。

列	描述
触发器	将鼠标移到 “触发器” 上时，会显示一个信息框，其中列出与监控项关联的触发器。 触发器数量以灰色显示。
键值	显示监控项键值。
间隔	显示检查频率。 注意，也可以通过点击立即执行按钮立即检查被动式监控项。
历史记录	显示监控项数据历史记录的保留天数。
趋势	显示监控项趋势历史记录的保留天数。
类型	显示监控项类型（Zabbix agent、SNMP agent、简单检查等）。
状态	显示监控项状态 - 启用、停用或不支持。点击状态即可手动更改状态：从 “启用” 更改为 “停用”（反之亦然）；从 “不支持” 更改为 “停用”（反之亦然）。 已丢失的已发现监控项会标记信息图标。工具提示文本会提供其状态的详细信息。
标签	显示监控项标签。 最多可显示三个标签（名称: 值对）。如果标签超过三个，则会显示 “...” 链接，将鼠标移到该链接上即可查看所有标签。
信息	如果监控项工作正常，此列不显示图标。如果发生错误，则会显示一个带有字母 “i” 的方形图标。将鼠标悬停在图标上，可在工具提示中查看错误描述。

要配置新的监控项，请点击右上角的创建监控项按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

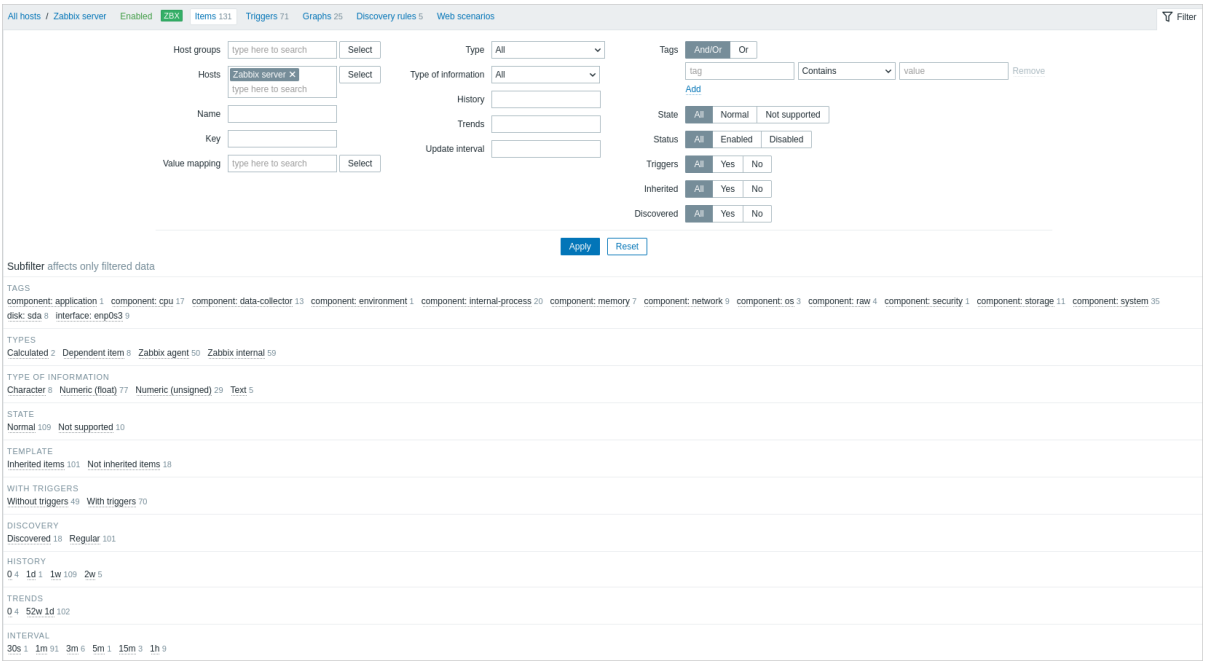
- 启用 - 将监控项状态更改为 已启用
- 禁用 - 将监控项状态更改为 已禁用
- 立即执行 - 立即执行新监控项值的检查。仅支持被动检查（请参见更多详情）。请注意，在立即检查值时，不会更新配置缓存，因此这些值不会反映监控项配置的最新更改。
- 清除历史和趋势 - 删除监控项的历史和趋势数据。
- 复制 - 将监控项复制到其他主机或模板。
- 批量更新 - 一次性为多个监控项更新若干属性。
- 删除 - 删除监控项。

要使用这些选项，请先勾选相应监控项前的复选框，然后点击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的监控项。为了获得更好的搜索性能，数据搜索时不会解析宏。

右上角提供了 *Filter* 图标。单击它将打开一个过滤器，您可以在其中指定所需的筛选条件。

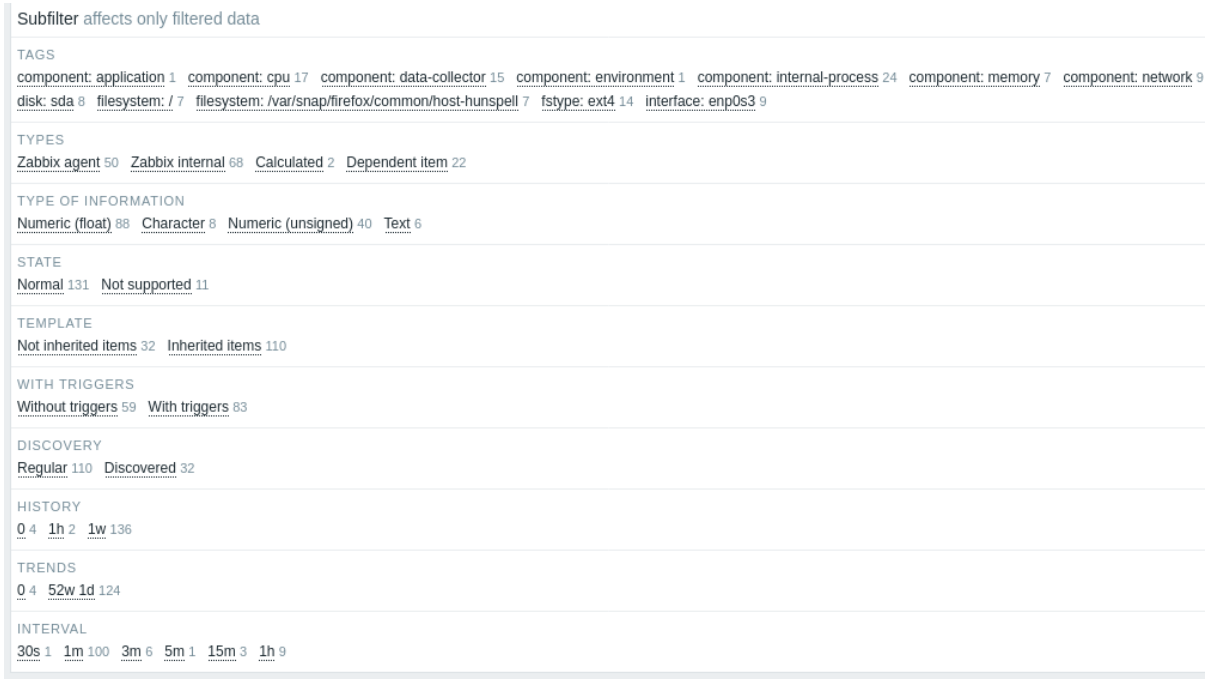


Parameter	Description
Host groups	按一个或多个主机组进行筛选。 指定父主机组会隐式选择所有嵌套主机组。 仅包含模板的主机组不能被选择。
Hosts	按一个或多个主机进行筛选。
Name	按监控项名称筛选。
Key	按监控项键值筛选。
Value mapping	按所使用的值映射筛选。 如果 *Hosts* 选项为空，则不显示此参数。
Type	按监控项类型筛选（Zabbix agent、SNMP agent 等）。
Type of information	按信息类型筛选（无符号数值、浮点数等）。
History	按监控项历史数据保留时长筛选。
Trends	按监控项趋势数据保留时长筛选。
Update interval	按监控项更新间隔筛选。
Tags	指定标签以限制显示的监控项数量。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含标签值中含有输入字符串的指定标签名称（子字符串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除标签值中含有输入字符串的指定标签名称（子字符串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足任一条件即可
State	按监控项状态筛选 - *Normal* 或 *Not supported*。
Status	按监控项状态筛选 - *Enabled* 或 *Disabled*。
Triggers	筛选带有（或不带有）触发器的监控项。
Inherited	筛选从模板继承（或未继承）的监控项。
Discovery	筛选由低级别发现发现（或未发现）的监控项。

使用子过滤器

子过滤器可进一步修改主过滤器应用的过滤条件。

其中包含可点击的链接，可快速访问相关监控项。监控项通过共同的实体进行关联，例如标签、监控项类型、监控项状态、监控项更新间隔等。单击某个实体后，该实体会以灰色背景突出显示，并立即对监控项进行过滤（无需在主过滤器中单击应用）。单击另一个实体会将其添加到过滤结果中。再次单击该实体可移除相应的过滤条件。



子过滤器基于过滤后的数据生成，而过滤后的数据最多包含 1000 条记录。如果希望在子过滤器中查看更多记录，需要增大搜索和过滤结

果限制参数的值（位于管理 > 常规 > *GUI*）。

与主过滤器不同，子过滤器会在每次表格刷新请求时更新，以始终显示可用过滤选项及其计数的最新信息。

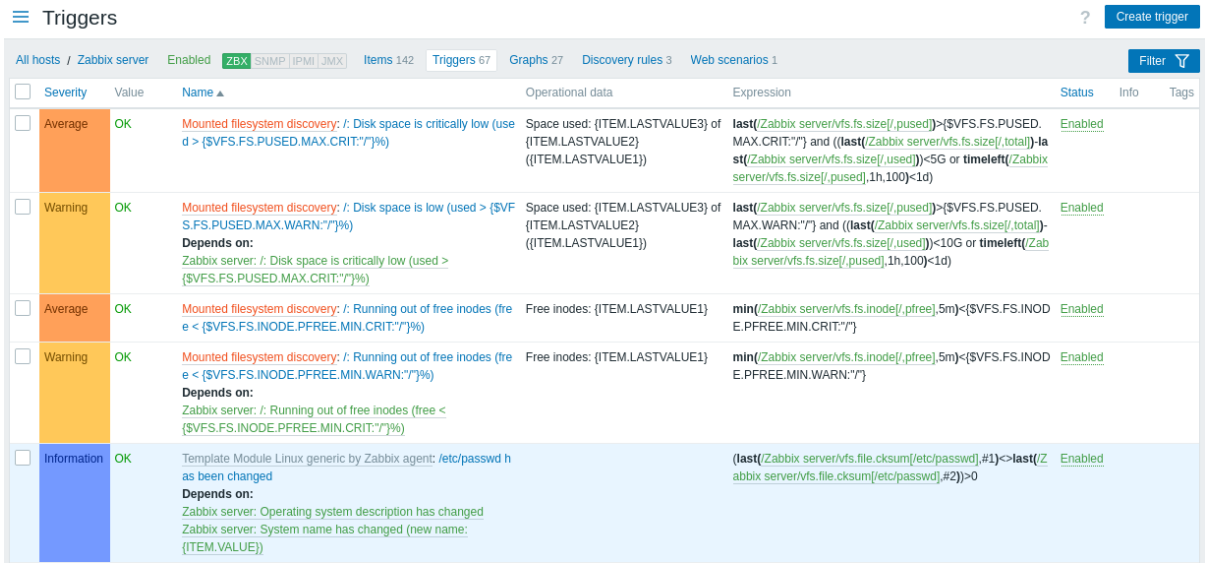
水平方向显示的实体数量最多为 100 个。如果实体数量更多，则会在末尾显示三点图标；该图标不可点击。

每个可点击实体旁的数字表示其中包含的监控项数量（基于主过滤器的结果）。单击某个实体后，其他可用实体旁的数字会带有加号，表示可以添加到当前选择中的监控项数量。不包含监控项的实体不会显示，除非之前已在子过滤器中选中该实体。

触发器

概述

可以从数据采集 > 主机访问主机的触发器列表，方法是单击相应主机的触发器。



显示的数据：

列	描述
严重性	触发器的严重性同时通过名称和单元格背景颜色显示。
值	显示触发器状态： 正常 - 正常状态 问题 - 问题状态
主机	触发器所属的主机。 单击主机名称可打开主机配置表单。 仅当筛选器中选择了多个主机或未选择主机时，才会显示此列。
名称	触发器名称，以指向触发器详细信息的蓝色链接显示。 单击触发器名称链接可打开触发器配置表单。 如果主机触发器属于模板，则会在触发器名称前以灰色链接显示模板名称。单击模板链接可打开模板级别的触发器列表。 如果触发器是根据触发器原型创建的，则其名称前会以橙色显示低级别发现规则名称。单击发现规则名称可打开触发器原型列表。
运行数据	触发器的运行数据定义，其中包含任意字符串和宏，这些内容将在监测 > 问题中动态解析。
表达式	显示触发器表达式。表达式中的主机-监控项部分显示为链接，指向监控项配置表单。
状态	显示触发器状态 - 已启用、已禁用或未知。单击状态可以手动更改状态：从 “已启用” 更改为 “已禁用”（反之亦然）；从 “未知” 更改为 “已禁用”（反之亦然）。 已禁用触发器的问题将不再显示在前端中，但不会被删除。 已丢失的已发现触发器会标记信息图标。工具提示文本会提供其状态的详细信息。
信息	如果一切正常，此列中不会显示图标。如果发生错误，则会显示带有字母 “i” 的方形图标。将鼠标悬停在图标上，可查看包含错误描述的工具提示。
标签	如果触发器包含标签，则会在此列中显示标签名称和值。

要配置新触发器，请单击右上角的创建触发器按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

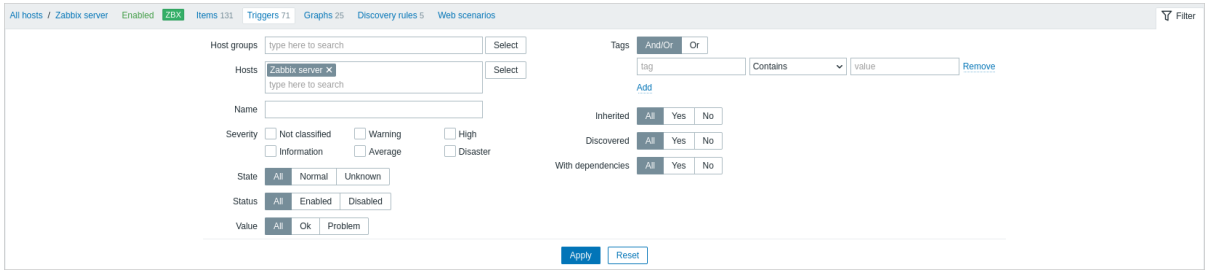
- 启用 - 将触发器状态更改为 已启用。
- 禁用 - 将触发器状态更改为 已禁用。
- 复制 - 将触发器复制到其他主机或模板。
- 批量更新 - 一次更新多个触发器的若干属性。
- 删除 - 删除触发器。

要使用这些选项，请先勾选相应触发器前面的复选框，然后单击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的触发器。为了获得更好的搜索性能，数据搜索时宏不会展开。

右上角提供了 *Filter* 图标。点击后会打开一个过滤器，您可以在其中指定所需的筛选条件。



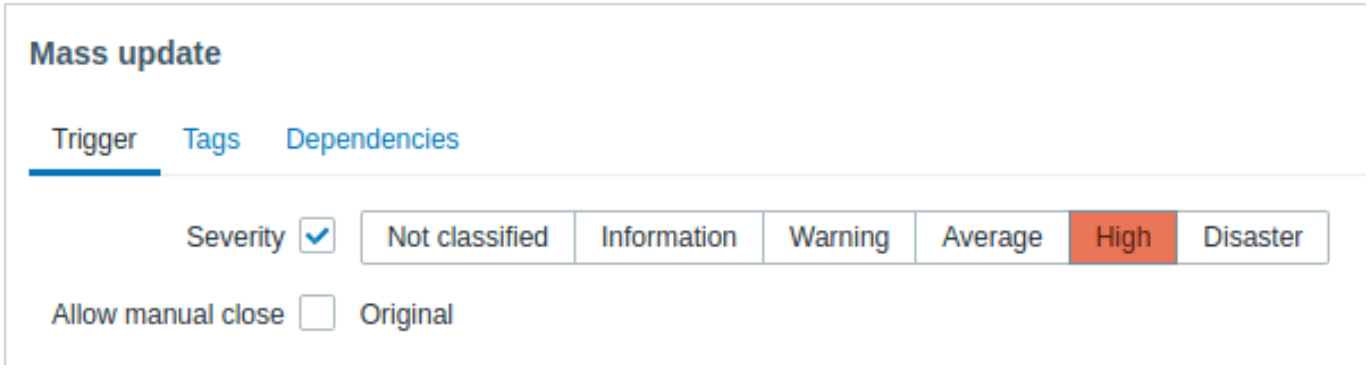
Parameter	Description
Host groups	按一个或多个主机组进行筛选。 指定父主机组会隐式选择其所有嵌套主机组。 仅包含模板的主机组无法被选择。
Hosts	按一个或多个主机进行筛选。 如果上方已选择主机组，则主机选择将仅限于这些组。
Name	按触发器名称筛选。
Severity	选择按一个或多个触发器严重性进行筛选。
State	按触发器expression state（Normal/Unknown）筛选。
Status	按触发器启用状态（Enabled/Disabled）筛选。
Value	按触发器值筛选。
Tags	按触发器标签名称和值筛选。可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称 **Equals** - 包含指定的标签名称和值（区分大小写） **Contains** - 包含标签值中含有输入字符串的指定标签名称（子串匹配，不区分大小写） **Does not exist** - 排除指定的标签名称 **Does not equal** - 排除指定的标签名称和值（区分大小写） **Does not contain** - 排除标签值中含有输入字符串的指定标签名称（子串匹配，不区分大小写） 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组 **Or** - 满足任一条件即可 标签名称和标签值字段都支持宏和macro functions。
Inherited	筛选从模板继承的（或未继承的）触发器。
Discovered	筛选由低级别发现发现的（或未被发现的）触发器。
With dependencies	筛选具有（或不具有）依赖关系的触发器。

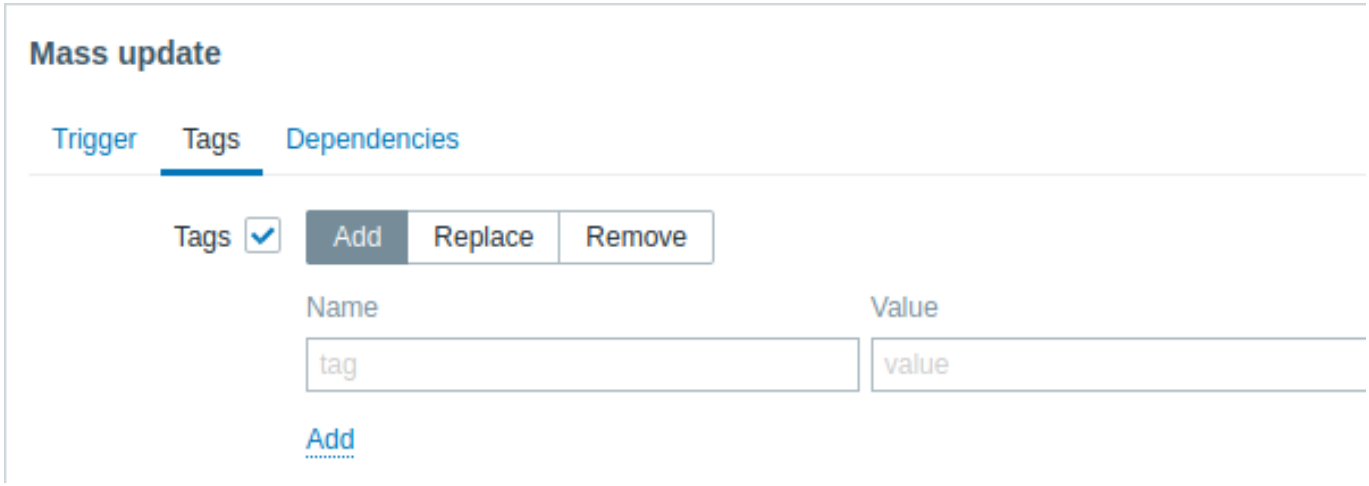
使用批量更新

使用批量更新可以一次更改多个触发器的某些属性，从而无需打开每个触发器进行编辑。

要批量更新触发器，请执行以下操作：

- 在列表中勾选要更新的触发器
- 点击列表下方的批量更新
- 导航到包含所需属性的选项卡（触发器、标签或依赖关系）
- 勾选要更新的属性对应的复选框

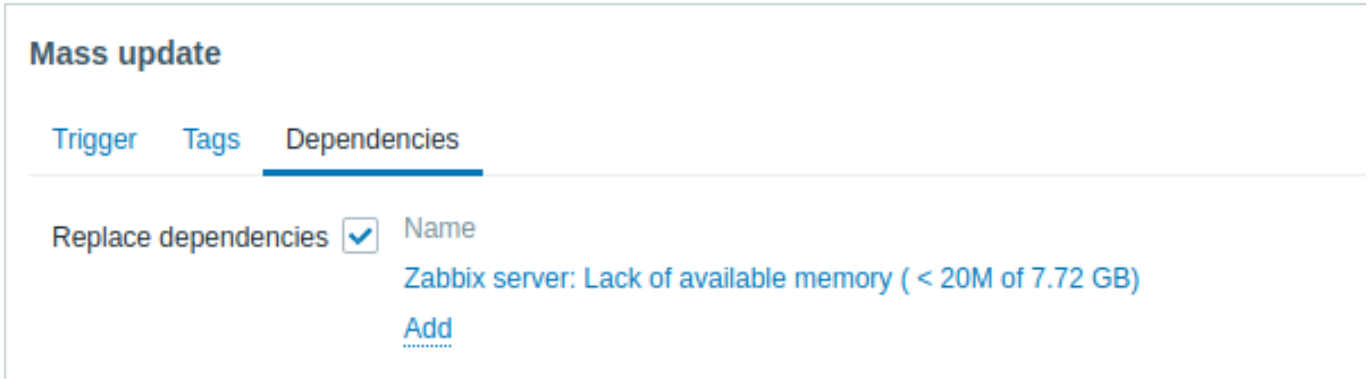




选择相应的按钮更新标签时，可以使用以下选项：

- 添加 - 为触发器添加新标签；
- 替换 - 删除触发器中现有的所有标签，并将其替换为下方指定的标签；
- 移除 - 从触发器中移除指定的标签。

请注意，名称相同但值不同的标签不被视为“重复标签”，可以添加到同一个触发器中。



替换依赖关系 - 将删除触发器中现有的所有依赖关系，并将其替换为指定的依赖关系。

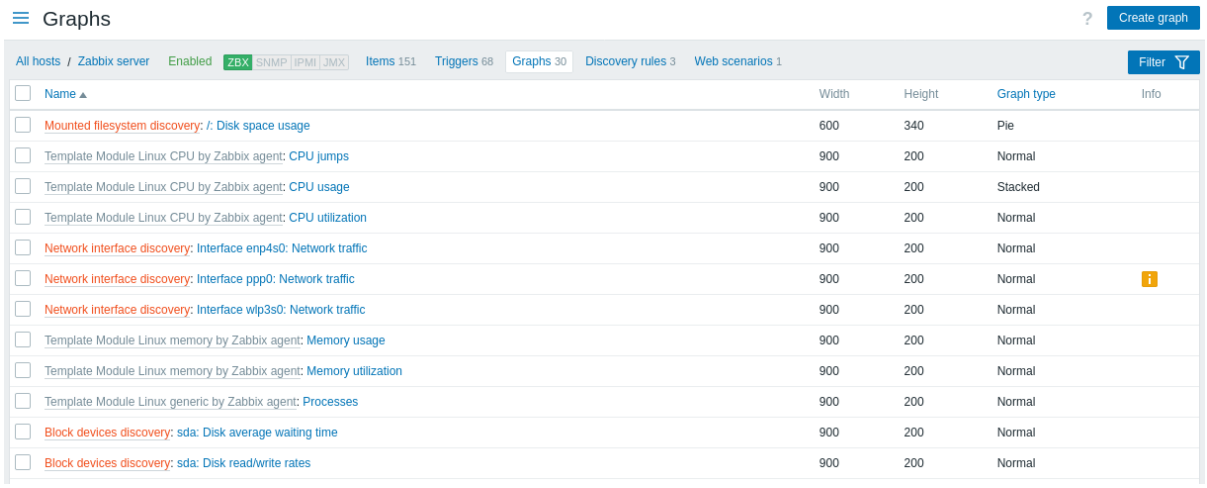
点击更新以应用更改。

图形

概述

可以在数据采集 > 主机中，点击相应主机的图形，访问该主机的自定义图形列表。

此时会显示现有图形列表。



显示的数据：

列	描述
主机	图形所属的主机。 点击主机名称可打开主机配置表单。 仅当筛选器中选择了多个主机或未选择主机时，才会显示此列。
名称	自定义图形的名称，以蓝色链接形式显示，可链接到图形详细信息。 点击图形名称链接可打开图形配置表单。 如果主机图形属于模板，则会在图形名称前显示模板名称，并以灰色链接形式显示。点击模板链接可打开模板级别的图形列表。 如果图形是根据图形原型创建的，则其名称前会以橙色显示低级别发现规则名称。点击发现规则名称可打开图形原型列表。
宽度	显示图形宽度。
高度	显示图形高度。
图形类型	显示图形类型 - 普通、堆叠、饼图或分离型。
信息	如果图形工作正常，此列中不会显示图标。如果发生错误，则会显示一个带有字母 “i” 的方形图标。将鼠标悬停在图标上，可在工具提示中查看错误描述。

要配置新图形，请点击右上角的创建图形按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 复制 - 将图形复制到其他主机或模板
- 删除 - 删除图形

要使用这些选项，请先勾选相应图形前面的复选框，然后单击所需的按钮。

使用过滤器

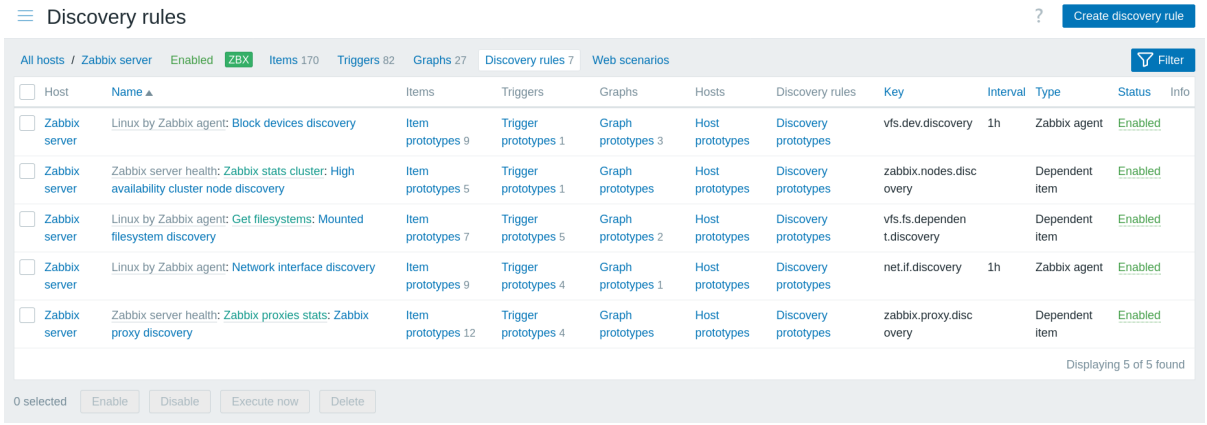
可以根据主机组和主机对图形进行过滤。为了获得更好的搜索性能，在搜索数据时不解析宏。

发现规则

概述

可以从数据采集 > 主机访问主机的低级别发现规则列表，方法是单击相应主机的发现。

系统会显示现有低级别发现规则的列表。通过更改筛选条件设置，还可以查看与主机无关的所有发现规则，或特定主机组的所有发现规则。



显示的数据：

列	描述
主机	显示主机的可见名称。 单击主机名称可打开主机配置表单。 如果没有可见的主机名称，则显示技术主机名称。
名称	显示规则名称，并以蓝色链接表示。 单击规则名称可打开低级别发现规则配置表单。 如果发现规则属于模板，则会在规则名称前显示模板名称，并以灰色链接表示。单击模板链接可打开模板级别的规则列表。
监控项	显示指向监控项原型列表的链接。 现有监控项原型的数量以灰色显示。
触发器	显示指向触发器原型列表的链接。 现有触发器原型的数量以灰色显示。
图形	显示指向图形原型列表的链接。 现有图形原型的数量以灰色显示。
主机	显示指向主机原型列表的链接。 现有主机原型的数量以灰色显示。
键	显示用于发现的监控项键值。
间隔	显示执行发现的频率。 请注意，也可以通过单击列表下方的立即执行按钮立即执行发现。
类型	显示用于发现的监控项类型（Zabbix agent、SNMP agent 等）。
状态	显示发现规则状态 - 已启用、已禁用或不支持。单击状态可以更改状态：从 “已启用” 更改为 “已禁用”（反之亦然）；从 “不支持” 更改为 “已禁用”（反之亦然）。
信息	如果一切正常，此列中不会显示图标。如果发生错误，则会显示带有字母 “i” 的方形图标。将鼠标悬停在图标上，可以查看包含错误描述的工具提示。

要配置新的低级别发现规则，请单击右上角的创建发现规则按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

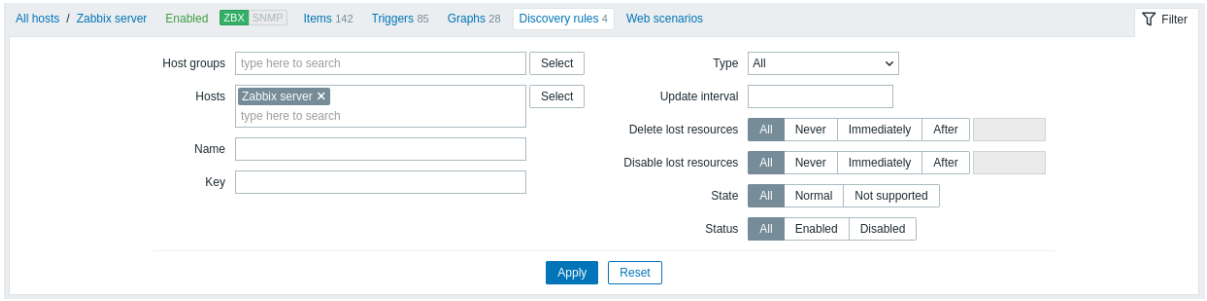
- 启用 - 将低级别发现规则状态更改为 *Enabled*。
- 禁用 - 将低级别发现规则状态更改为 *Disabled*。
- 立即执行 - 立即根据发现规则执行发现。请参见更多详情。请注意，在立即执行发现时，不会更新配置缓存，因此结果不会反映发现规则配置的最新更改。
- 删除 - 删除低级别发现规则。

要使用这些选项，请先勾选相应发现规则前的复选框，然后单击所需按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的发现规则。为了获得更好的搜索性能，数据搜索时不会解析宏。

Filter 链接位于发现规则列表上方。单击后会显示一个过滤器，您可以按主机组、主机、名称、监控项键值、监控项类型以及其他参数来筛选发现规则。

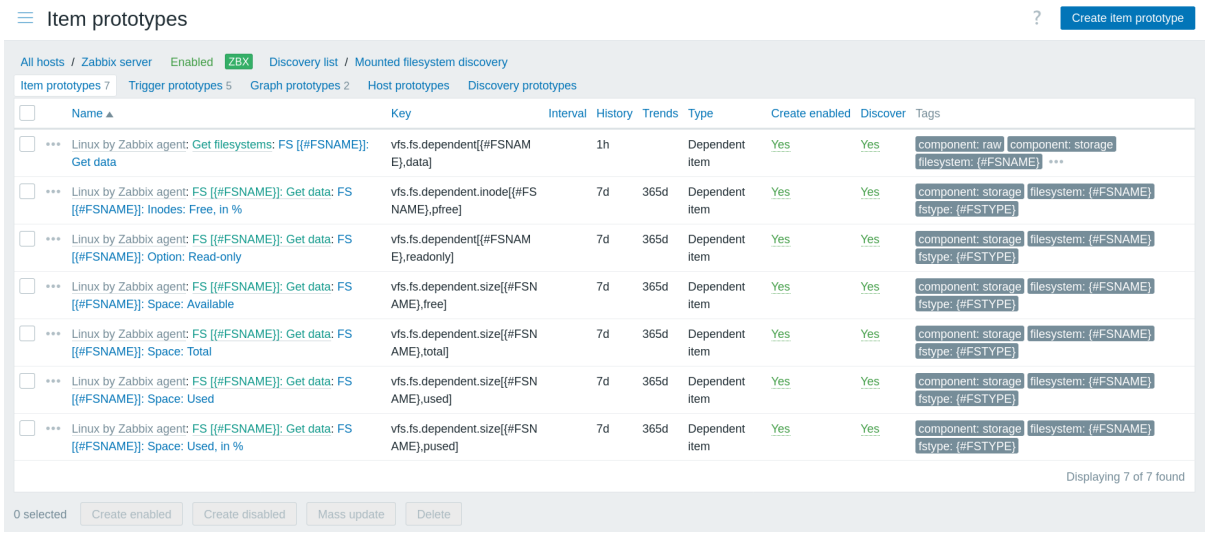


Parameter	Description
Host groups	按一个或多个主机组筛选。 指定父主机组会隐式选择所有嵌套主机组。
Hosts	按一个或多个主机筛选。
Name	按发现规则名称筛选。
Key	按发现监控项键值筛选。
Type	按发现监控项类型筛选。
Update interval	按更新间隔筛选。 对于 Zabbix trapper 和依赖监控项不可用。
Delete lost resources	按 *Delete lost resources* 周期筛选。
Disable lost resources	按 *Disable lost resources* 周期筛选。
SNMP OID	按 SNMP OID 筛选。 仅在将 *SNMP agent* 选为类型时可用。
State	按发现规则状态筛选（All/Normal/Not supported）。
Status	按发现规则状态筛选（All/Enabled/Disabled）。

监控项原型

概述

本节显示主机上低级别发现规则的监控项原型。监控项原型是通过低级别发现创建的实际主机监控项的基础。



显示的数据：

列	描述
名称	显示监控项原型的名称，并以蓝色链接显示。 单击名称可打开监控项原型配置表单。 如果监控项原型属于模板，则规则名称前会以灰色链接显示模板名称。单击模板链接可打开模板级别的监控项原型列表。 如果监控项原型的类型为“依赖监控项”，则主监控项名称会以绿色显示在监控项原型名称前。
键值	显示监控项原型的键值。
间隔	显示检查频率。

列	描述
历史数据	显示保留监控项历史数据的天数。
趋势数据	显示保留监控项趋势数据的天数。
类型	显示监控项原型的类型（Zabbix agent、SNMP agent、简单检查等）。
启用创建	根据此原型创建监控项： 是 - 已启用 否 - 已禁用。单击 “是” 和 “否” 可在两者之间切换。
发现	根据此原型发现监控项： 是 - 发现 否 - 不发现。单击 “是” 和 “否” 可在两者之间切换。
标签	显示监控项原型的标签。

要配置新的监控项原型，请单击右上角的创建监控项原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

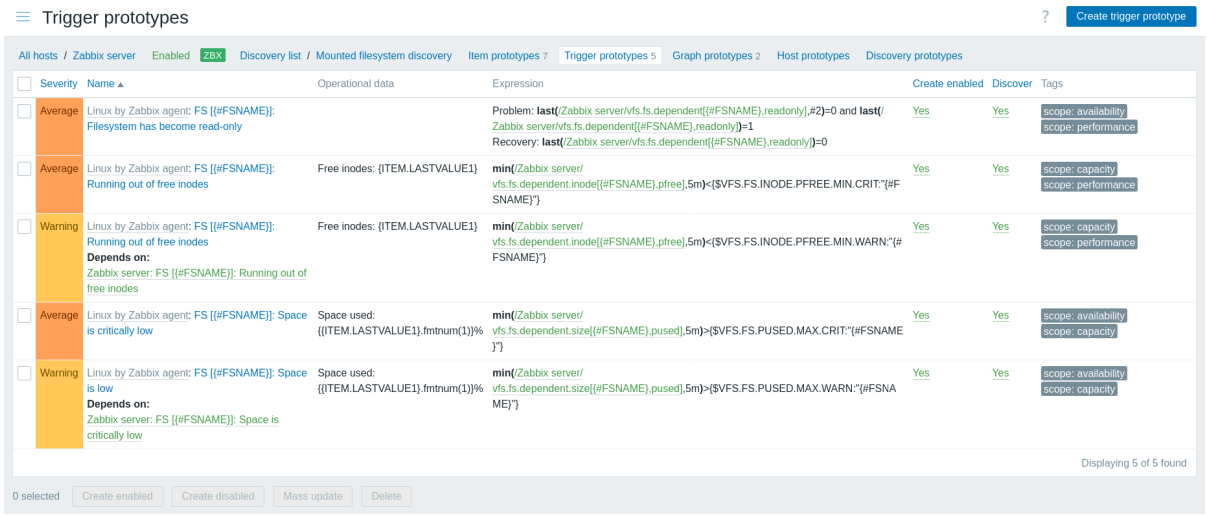
- 创建为启用 - 将这些监控项创建为 启用
- 创建为禁用 - 将这些监控项创建为 禁用
- 批量更新 - 批量更新这些监控项原型
- 删除 - 删除这些监控项原型

要使用这些选项，请先勾选相应监控项原型前面的复选框，然后单击所需的按钮。

触发器原型

概述

本节显示主机上低级别发现规则的触发器原型。触发器原型是通过低级别发现创建的实际主机触发器的基础。



显示的数据：

列	描述
名称	触发器原型的名称，以蓝色链接显示。 单击名称可打开触发器原型配置表单。 如果触发器原型属于已链接的模板，则模板名称会作为灰色链接显示在触发器名称之前。单击模板链接可打开已链接模板级别上的触发器原型列表。
运行数据	显示触发器的运行数据格式，其中包含任意字符串和宏，这些内容将在监控 > 问题中动态解析。
创建已启用	根据此原型创建触发器： 是 - 已启用 否 - 已禁用。单击 “是” 或 “否” 可在两者之间切换。
发现	根据此原型发现触发器： 是 - 发现 否 - 不发现。单击 “是” 或 “否” 可在两者之间切换。

列	描述
标签	显示触发器原型的标签。

要配置新的触发器原型，请单击右上角的创建触发器原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

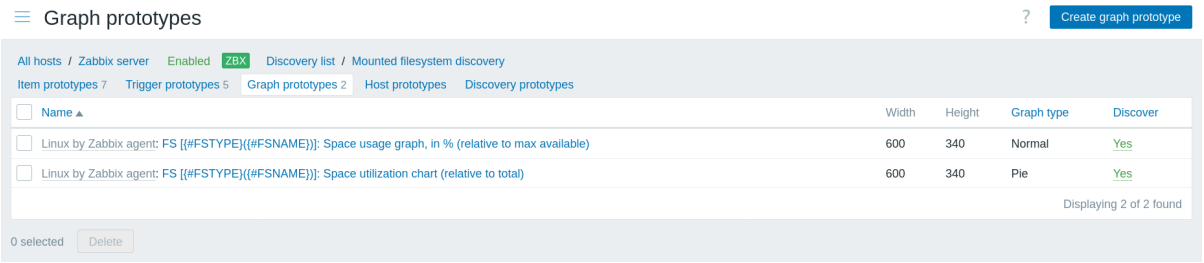
- 创建为启用 - 将这些触发器原型创建为 启用
- 创建为禁用 - 将这些触发器原型创建为 禁用
- 批量更新 - 批量更新这些触发器原型
- 删除 - 删除这些触发器原型

要使用这些选项，请先勾选相应触发器原型前面的复选框，然后单击所需的按钮。

图形原型

概述

本节显示主机上低级别发现规则的图原型。图原型是通过低级别发现创建的实际主机图形的基础。



显示的数据：

列	描述
名称	图原型的名称，以蓝色链接显示。 单击名称可打开图原型配置表单。 如果图原型属于已链接的模板，则模板名称会作为灰色链接显示在图名称之前。单击模板链接可打开已链接模板级别的图原型列表。
宽度	显示图原型的宽度。
高度	显示图原型的高度。
类型	显示图原型的类型 - 普通、堆叠、饼图或分离型。
发现	根据此原型发现图形： 是 - 发现 否 - 不发现。单击 “是” 和 “否” 可在两者之间切换。

要配置新的图原型，请单击右上角的创建图原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

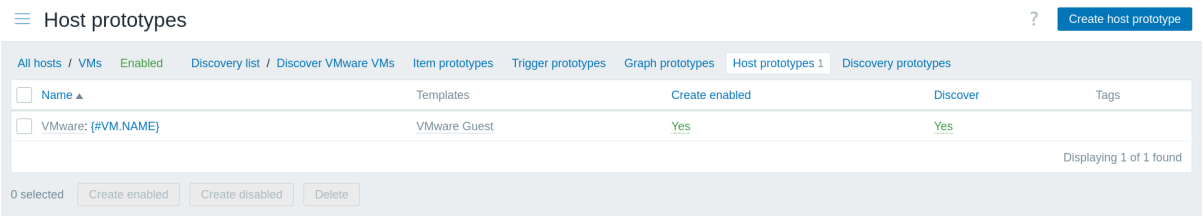
- 删除 - 删除这些图形原型

要使用这些选项，请先勾选相应图形原型前面的复选框，然后单击所需的按钮。

主机原型

概述

本节显示主机上低级别发现规则的主机原型。主机原型是低级别发现期间创建的实际主机的基础。



显示的数据：

列	描述
名称	主机原型的名称，以蓝色链接显示。 单击名称可打开主机原型配置表单。 如果主机原型属于已链接的模板，则模板名称会以灰色链接显示在主机名称之前。单击模板链接可打开已链接模板级别的主机原型列表。
模板	显示主机原型的模板。
创建已启用	根据此原型创建主机： 是 - 已启用 否 - 已禁用。单击“是”或“否”可在两者之间切换。
发现	根据此原型发现主机： 是 - 发现 否 - 不发现。单击“是”或“否”可在两者之间切换。
标签	显示主机原型的标签。

要配置新的主机原型，请单击右上角的创建主机原型按钮。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

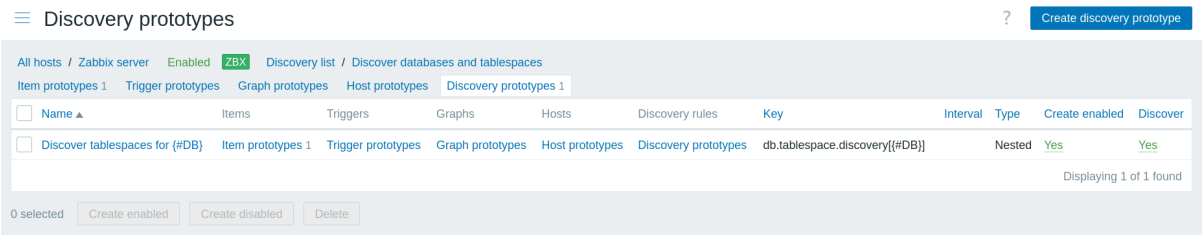
- *Create enabled* - 将这些主机创建为 *Enabled*
- *Create disabled* - 将这些主机创建为 *Disabled*
- *Delete* - 删除这些主机原型

要使用这些选项，请先勾选相应主机原型前的复选框，然后单击所需的按钮。

发现原型

概述

本节显示主机上低级别发现规则的发现原型。发现原型是父发现规则的嵌套发现规则。



显示的数据：

列	描述
名称	显示发现原型的名称，并以蓝色链接显示。 单击名称可打开发现原型配置表单。 如果发现原型属于已链接的模板，则模板名称会以灰色链接显示在主机名称之前。单击模板链接可打开已链接模板级别的发现原型列表。
监控项	显示指向监控项原型列表的链接。 现有监控项原型的数量以灰色显示。
触发器	显示指向触发器原型列表的链接。 现有触发器原型的数量以灰色显示。
图形	显示指向图形原型列表的链接。 现有图形原型的数量以灰色显示。

列	描述
主机	显示指向主机原型列表的链接。 现有主机原型的数量以灰色显示。
发现规则	显示指向发现原型列表的链接。 现有发现原型的数量以灰色显示。
键值	显示用于发现的监控项键值。
间隔	显示执行发现的频率。 注意，也可以通过单击列表下方的立即执行按钮立即执行发现。
类型	显示用于发现的监控项类型（Zabbix agent、SNMP agent 等）。
启用创建	显示发现原型创建状态 - 启用创建（是）或禁用创建（否）。单击状态可进行更改 - 从 “是” 改为 “否”（反之亦然）。
发现	显示发现原型的发现状态 - 发现（是）或不发现（否）。单击状态可进行更改 - 从 “是” 改为 “否”（反之亦然）。

要配置新的发现原型，请单击右上角的创建发现原型按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

- 创建为启用 - 将这些发现原型创建为 启用
- 创建为禁用 - 将这些发现原型创建为 禁用
- 删除 - 删除这些发现原型

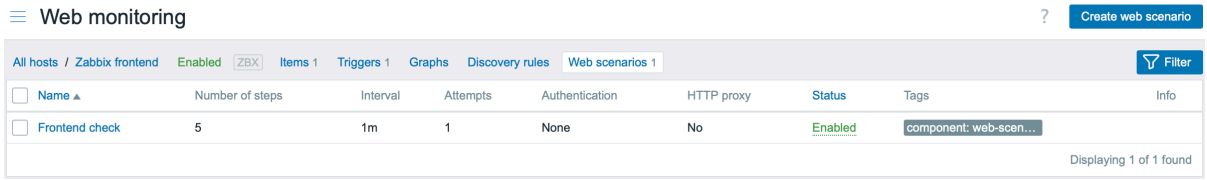
要使用这些选项，请先勾选相应发现原型前面的复选框，然后单击所需按钮。

Web 场景

概述

可以从数据采集 > 主机访问主机的Web 场景列表，方法是单击相应主机的 *Web*。

此时会显示现有 Web 场景的列表。



显示的数据：

列	描述
名称	Web 场景的名称。单击 Web 场景名称可打开Web 场景配置表单。 如果主机所属的 Web 场景来自模板，则 Web 场景名称前会显示模板名称，并以灰色链接表示。单击模板链接可打开模板级别的 Web 场景列表。
步骤数	场景包含的步骤数。
更新间隔	执行场景的频率。
尝试次数	执行 Web 场景步骤时的尝试次数。
身份验证	显示身份验证方法 - Basic、NTLM、Kerberos、Digest 或 None。
HTTP proxy	显示 HTTP proxy；如果未使用，则显示 “No”。
状态	显示 Web 场景状态 - 已启用或已禁用。 单击状态可更改状态。
标签	显示 Web 场景标签。 最多可显示三个标签（名称: 值对）。如果标签超过三个，则会显示 “...” 链接，将鼠标悬停在该链接上可查看所有标签。
信息	如果一切正常，此列中不会显示图标。如果发生错误，则会显示带有字母 “i” 的方形图标。将鼠标悬停在图标上可查看包含错误描述的工具提示。

要配置新的 Web 场景，请单击右上角的创建 *Web* 场景按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

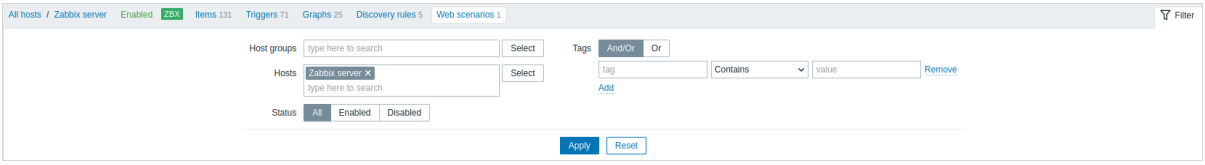
- 启用 - 将场景状态更改为 已启用
- 禁用 - 将场景状态更改为 已禁用
- 清除历史数据和趋势数据 - 清除这些场景的历史数据和趋势数据
- 删除 - 删除 web 场景

要使用这些选项，请先勾选相应 web 场景前面的复选框，然后单击所需的按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的场景。为了获得更好的搜索性能，数据搜索时宏不会解析。

Filter 链接位于 web 场景列表上方。点击它将打开一个过滤器，您可以在其中指定所需的筛选条件。



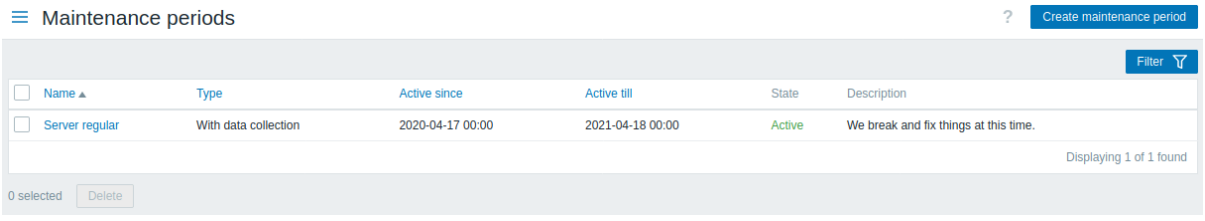
Parameter	Description
Host groups	按一个或多个主机组进行筛选。 指定父主机组会隐式选择所有嵌套的主机组。
Hosts	按一个或多个主机进行筛选。 如果上方已选择主机组，则主机选择将限制在这些组内。
Status	按 web 场景状态进行筛选。
Tags	按 web 场景标签名称和值进行筛选。 可以包含或排除特定标签及标签值。可设置多个条件。标签名称匹配始终区分大小写。 每个条件可使用以下运算符： **Exists** - 包含指定的标签名称； **Equals** - 包含指定的标签名称和值（区分大小写）； **Contains** - 包含指定的标签名称，其中标签值包含输入的字符串（子串匹配，不区分大小写）； **Does not exist** - 排除指定的标签名称； **Does not equal** - 排除指定的标签名称和值（区分大小写）； **Does not contain** - 排除指定的标签名称，其中标签值包含输入的字符串（子串匹配，不区分大小写）。 条件有两种计算类型： **And/Or** - 必须满足所有条件，具有相同标签名称的条件将按 Or 条件分组； **Or** - 满足任一条件即可。

维护

概述

在数据采集 > 维护部分，用户可以配置和维护主机的维护时段。

页面会显示现有维护时段及其详细信息列表。



显示的数据：

列	说明
名称	维护时段的名称。单击维护时段名称可打开维护时段配置表单。
类型	显示维护类型：有数据采集或无数据采集
生效时间	执行维护时段开始生效的日期和时间。 注意：此时间不会激活维护时段；需要单独设置维护时段。
结束时间	执行维护时段停止生效的日期和时间。

列	说明
状态	维护时段的状态： 即将开始 - 即将生效 活动 - 当前处于活动状态 已过期 - 不再处于活动状态
描述	显示维护时段的描述。

要配置新的维护时段，请单击右上角的创建维护时段按钮。

批量编辑选项

列表下方的按钮提供一个批量编辑选项：

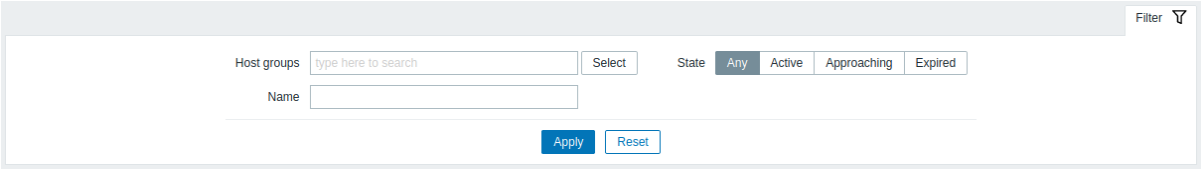
- 删除 - 删除维护周期

要使用此选项，请勾选相应维护周期前的复选框，然后单击 删除。

使用过滤器

您可以使用过滤器仅显示您感兴趣的维护周期。为了获得更好的搜索性能，数据搜索时不会解析宏。

Filter 链接位于维护周期列表上方。单击该链接后，会显示一个过滤器，您可以按主机组、名称和状态筛选维护周期。



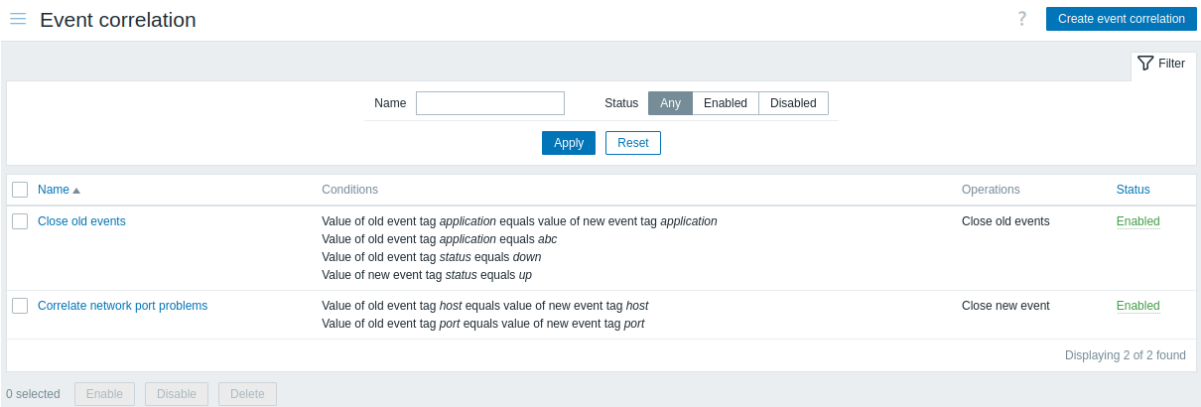
维护期间队列的计算

Attention:
Zabbix proxy 不会感知维护周期；有关详细信息，请参见维护期间队列的计算。

事件关联

概述

在数据采集 > 事件关联部分，用户可以配置和维护 Zabbix 事件的全局关联规则。



显示的数据：

列	描述
名称	关联规则的名称。单击关联规则名称可打开事件关联配置表单。
条件	显示关联规则条件。
操作	显示关联规则操作。
状态	显示关联规则状态 - 已启用或已禁用。 单击状态即可更改状态。

要配置新的关联规则，请单击右上角的创建事件关联按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

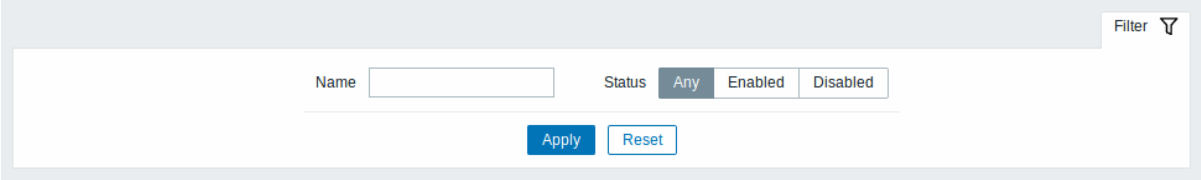
- 启用 - 将关联规则状态更改为 已启用
- 禁用 - 将关联规则状态更改为 已禁用
- 删除 - 删除关联规则

要使用这些选项，请先勾选相应关联规则前面的复选框，然后单击所需按钮。

使用筛选器

您可以使用筛选器仅显示您感兴趣的关联规则。为了获得更好的搜索性能，数据会在宏未解析的情况下进行搜索。

筛选器链接位于关联规则列表上方。单击该链接后，将显示一个筛选器，您可以按名称和状态筛选关联规则。

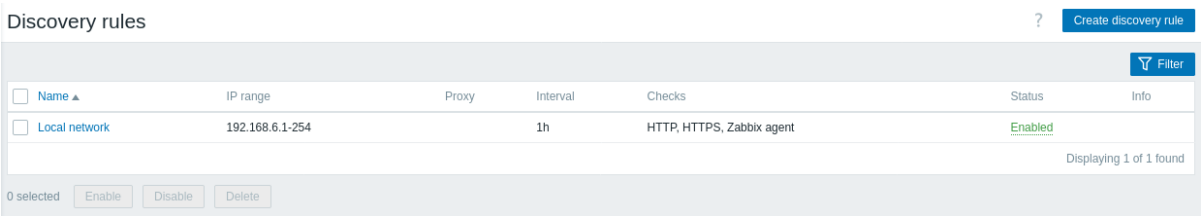


发现

概述

在数据采集 > 发现部分，用户可以配置和维护发现规则。

页面会显示现有发现规则及其详细信息列表。



显示的数据：

列	描述
名称	发现规则的名称。单击发现规则名称可打开发现规则配置表单。
IP 范围	显示用于网络扫描的 IP 地址范围。
Proxy	如果由 proxy 执行发现，则显示 proxy 名称。
间隔	显示执行发现的频率。
检查	显示用于发现的检查类型。
状态	显示发现规则的状态 - 已启用或已禁用。 单击状态可以更改状态。
信息	如果一切正常，此列为空。如果发生错误，则显示带有字母 “i” 的红色信息图标。将鼠标悬停在图标上，可在工具提示中查看错误描述。

要配置新的发现规则，请单击右上角的创建发现规则按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

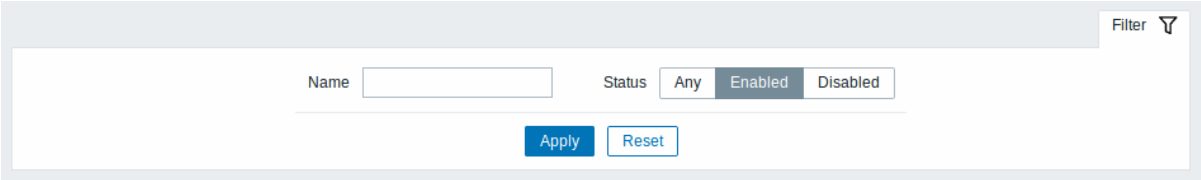
- 启用 - 将发现规则状态更改为 已启用
- 禁用 - 将发现规则状态更改为 已禁用
- 删除 - 删除发现规则

要使用这些选项，请先勾选相应发现规则前面的复选框，然后单击所需的按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的发现规则。为了获得更好的搜索性能，数据搜索时不会解析宏。

Filter 链接位于发现规则列表上方。单击它后，会显示一个过滤器，您可以按名称和状态筛选发现规则。



告警

概述

该菜单包含与在 Zabbix 中配置告警相关的部分。

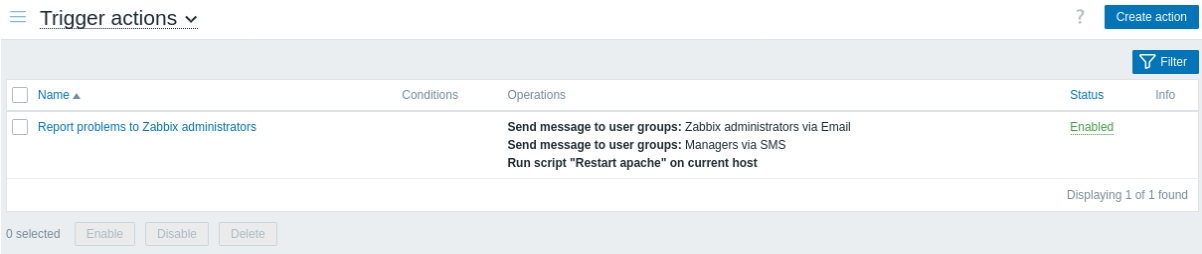
操作

概述

在告警 > 动作部分，用户可以配置和维护动作。

显示的动作是分配给所选事件源的动作（触发器、服务、发现、自动注册、内部动作）。

要切换到其他事件源，请在告警菜单部分中点击动作。也可以使用左上角的标题下拉菜单在不同事件源之间切换。



显示的数据：

列	描述
名称	动作的名称。点击动作名称可打开动作配置表单。
条件	显示动作条件。
操作	显示动作操作。 操作列表还会显示用于发送通知的媒介类型（电子邮件、SMS 或脚本），以及通知接收者的姓名和姓氏（显示在用户名后的括号中）。 根据所选的操作类型，动作操作可以是通知，也可以是远程命令。
状态	显示动作状态 - 已启用或已禁用。 点击状态即可更改状态。 有关正在进行的升级过程中禁用动作时会发生什么的详细信息，请参阅升级部分。
信息	如果一切正常，此列中不会显示图标。如果发生错误（例如升级后或低级别发现后缺少动作操作或条件），则会显示警告图标。将鼠标悬停在图标上可查看包含错误描述的工具提示。

要配置新动作，请点击右上角的创建动作按钮。

对于没有超级管理员权限的用户，动作会根据权限设置显示。这意味着在某些情况下，由于特定的权限限制，没有超级管理员权限的用户无法查看完整的动作列表。如果满足以下条件，则会向没有超级管理员权限的用户显示动作：

- 用户对动作条件中的主机组、主机、模板和触发器具有读写权限
- 用户对动作操作、恢复操作和更新操作中的主机组、主机和模板具有读写权限
- 用户对动作操作、恢复操作和更新操作中的用户组和用户具有读取权限

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

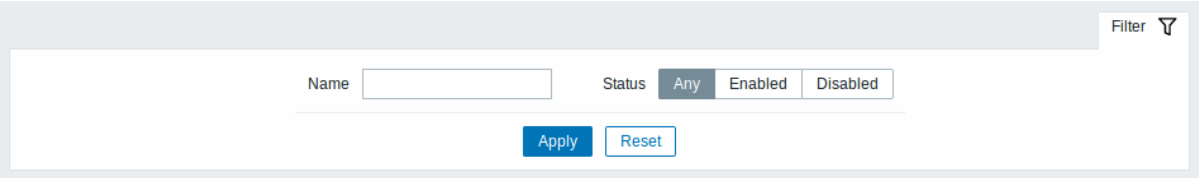
- 启用 - 将动作状态更改为 已启用
- 禁用 - 将动作状态更改为 已禁用
- 删除 - 删除这些动作

要使用这些选项，请先勾选相应动作前面的复选框，然后单击所需的按钮。

使用过滤器

您可以使用筛选器只显示您感兴趣的动作。为了获得更好的搜索性能，在搜索数据时不解析宏。

过滤器链接可以在动作列表的上方找到。如果单击它，就会出现一个过滤器，您可以根据名称和状态过滤动作。



媒体类型

概述

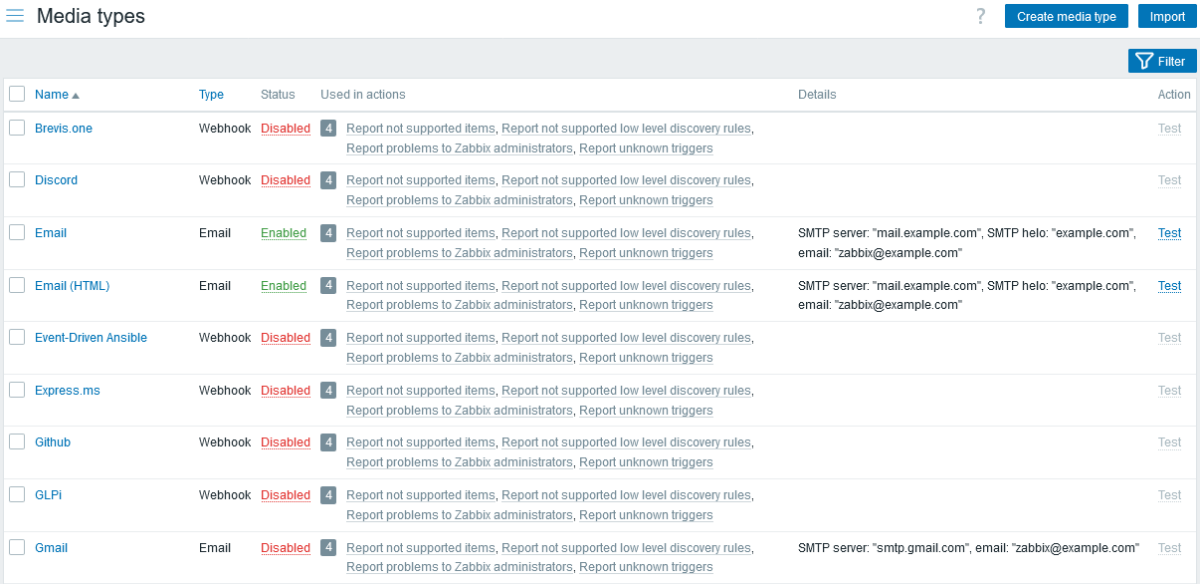
在告警 > 媒介类型部分，用户可以配置和维护媒介类型信息。

此菜单仅对SuperAdmin用户类型的用户可用。

媒介类型信息包含有关如何将媒介用作通知传送渠道的一般说明。具体详细信息（例如要向其发送通知的单个电子邮件地址）保存在各个用户的信息中。

页面会显示现有媒介类型及其详细信息的列表。

Note:
如果在前端配置文件的 `$ZBX_FEATURE_FLAGS['media_type_denylist']` 功能标志中指定了传送方式（*Email*、*SMS*、*Webhook*、*Script* 或 *Push*），则相应的预配置媒介类型会在此部分中被禁用。



显示的数据：

列	描述
名称	媒介类型的名称。单击名称可打开媒介类型配置表单。
类型	显示媒介的类型（电子邮件、SMS 等）。
状态	显示媒介类型的状态 - 已启用或已禁用。 单击状态可以更改状态。
用于动作	显示使用该媒介类型的动作，并在前面显示这些动作的总数。 单击动作名称可打开动作配置表单。如果用户没有该动作的权限，则名称不可点击。
详细信息	显示媒介类型的详细信息。
操作	可执行以下操作： 测试 - 单击可打开测试表单，在其中输入媒介类型参数（例如，包含测试主题和正文的收件人地址），并发送测试消息，以验证配置的媒介类型是否正常工作。另请参阅：媒介类型测试。

要配置新的媒介类型，请单击右上角的创建媒介类型按钮。

要导入媒介类型，请单击右上角的导入按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

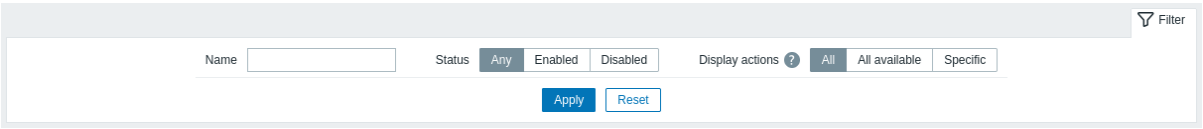
- 启用 - 将媒体类型状态更改为 已启用
- 禁用 - 将媒体类型状态更改为 已禁用
- 导出 - 将媒体类型导出为 YAML、XML 或 JSON 文件
- 删除 - 删除媒体类型

要使用这些选项，请勾选相应媒体类型前面的复选框，然后单击所需的按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的媒体类型。为了获得更好的搜索性能，数据搜索时不会解析宏。

媒体类型列表上方提供了 *Filter* 链接。单击后会显示一个过滤器，您可以按名称和状态筛选媒体类型。此外，您还可以使用过滤器根据媒体类型的使用范围，在 *Used in actions* 列中显示操作（由操作operation details 中的 *Send to media type* 参数定义）。



脚本

概述

在告警 > 脚本部分，可以配置和维护用户定义的全局脚本。

此菜单仅对SuperAdmin用户类型的用户可用。

根据配置的范围以及用户权限，全局脚本可用于执行以下操作：

- 在前端各个位置（仪表板、问题、最新数据、拓扑图等）的主机菜单中执行。
- 在事件菜单中执行。
- 作为操作的操作步骤运行。

脚本仅可由 Zabbix agent、Zabbix 服务器（proxy）或 Zabbix 服务器执行。另请参阅命令执行。

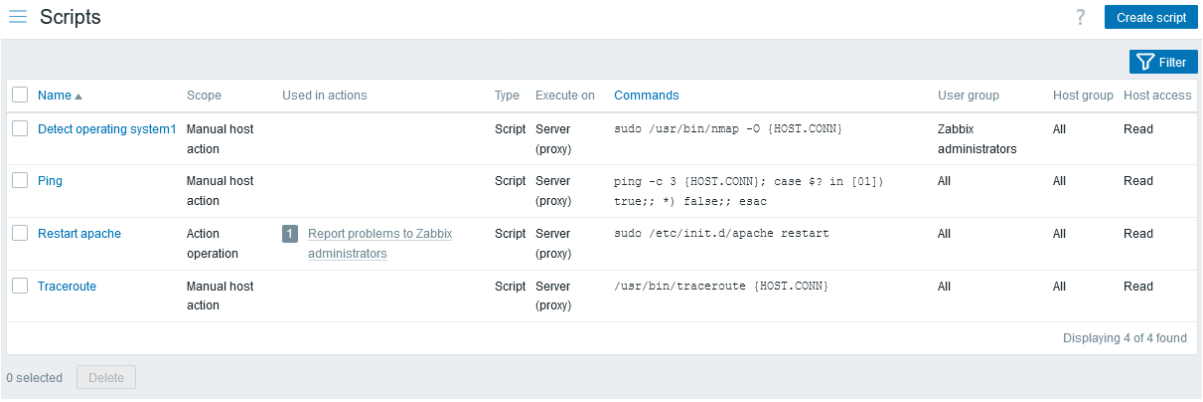
默认情况下，Zabbix agent 和 Zabbix proxy 上均禁用远程命令。可以按以下方式启用：

- 对于由 Zabbix agent 执行的远程命令，请在agent 配置中，为每个允许的命令添加 `AllowKey=system.run[<command>,*]` 或 `AllowKeyRegexp=^system\.run\[<command>,.*\]$` 参数，其中 `*` 匹配任一执行模式（`wait` 或 `nowait`）。
- 对于由 Zabbix proxy 执行的远程命令，请在proxy 配置中将 `EnableRemoteCommands` 参数设置为 1。请注意，当远程命令由 proxy 所监控主机上的 Zabbix agent 执行时，无需在 Zabbix proxy 上启用远程命令。

Attention:
除非确有必要，否则不要使用 `AllowKey=system.run[*]` 或 `AllowKeyRegexp=^system\.run\[.*\]$`，因为这会允许在主机上执行任意命令。出于安全原因，建议仅允许执行精确的命令，而 `system.run[*]` 默认被拒绝（`DenyKey=system.run[*]`）。

可以在服务器配置中将 `EnableGlobalScripts=0` 设置为禁用 Zabbix 服务器上的全局脚本执行。对于从 Zabbix 7.0 开始的新安装，Zabbix 服务器上的全局脚本执行默认处于禁用状态。

页面会显示现有脚本及其详细信息列表。



显示的数据：

列	描述
名称	脚本名称。单击脚本名称可打开脚本配置表单。
范围	脚本的范围 - 操作步骤、手动主机操作或手动事件操作。此设置决定脚本的可用位置 - 操作操作步骤中的远程命令、主机菜单或事件菜单。
用于操作	显示使用该脚本的所有操作，并在前面显示这些操作的总数。 单击操作名称可打开操作配置表单。如果用户没有该操作的权限，则操作名称不可点击。
类型	显示脚本类型 - *URL*、*Webhook*、脚本、*SSH*、*Telnet* 或 *IPMI* 命令。
执行位置	显示脚本将在 Zabbix agent、Zabbix proxy 或服务器上执行，还是仅在 Zabbix 服务器上执行。
命令	显示脚本中要执行的所有命令。 对于 webhook，此处不显示任何内容。
用户组	显示脚本可供使用的用户组（对于所有用户组则显示 所有）。
主机组	显示脚本可供使用的主机组（对于所有主机组则显示 所有）。
主机访问权限	显示主机组的权限级别 - 读取或写入。只有具有所需权限级别的用户才能执行该脚本。

要配置新脚本，请单击右上角的创建脚本按钮。

批量编辑选项

列表下方的按钮提供一个批量编辑选项：

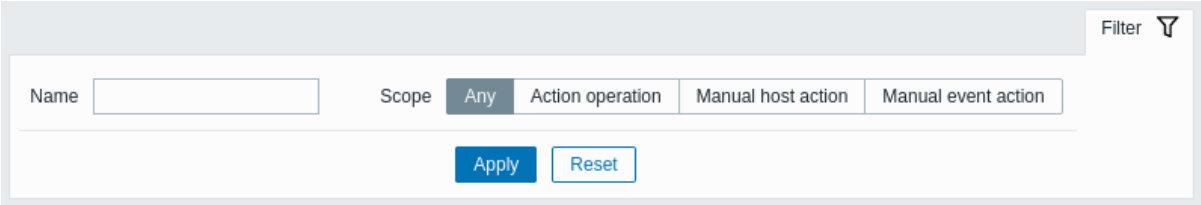
- 删除 - 删除脚本

要使用此选项，请勾选相应脚本前的复选框，然后点击 删除。

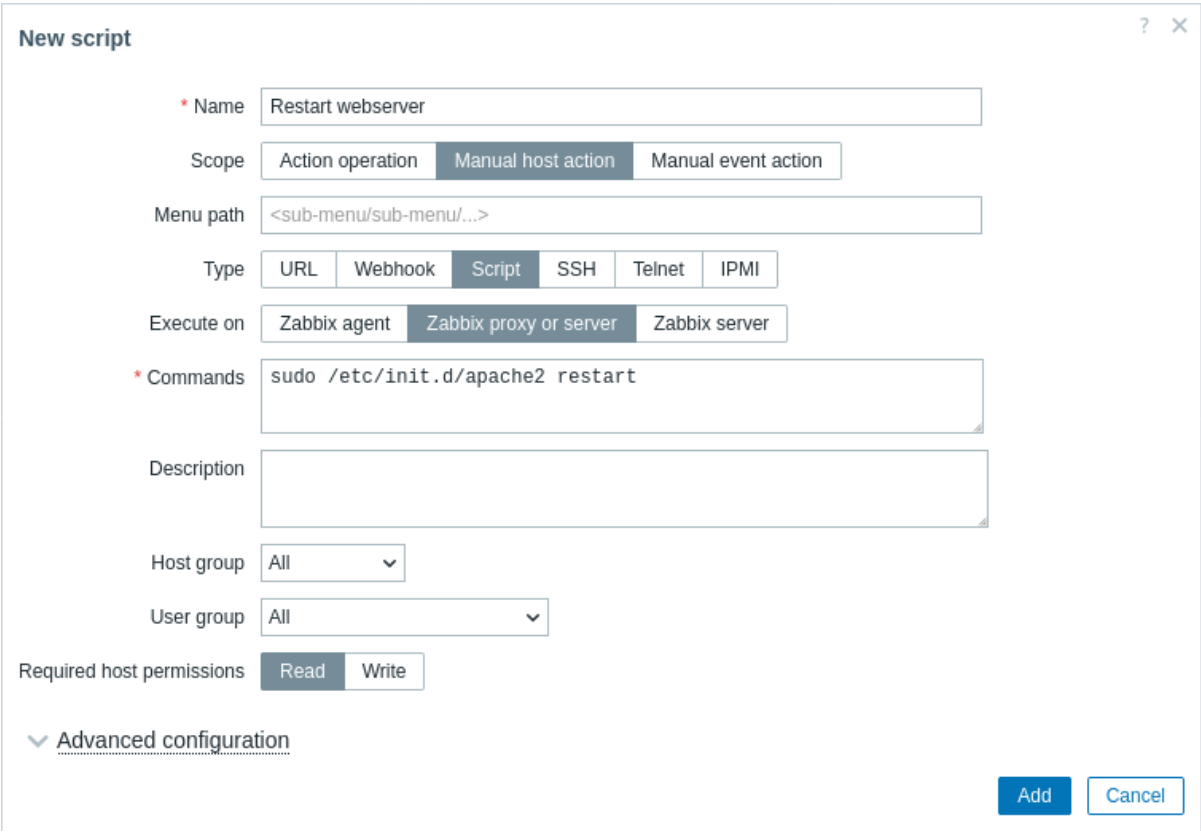
使用过滤器

您可以使用过滤器仅显示您感兴趣的脚本。为了获得更好的搜索性能，数据搜索时不会解析宏。

Filter 链接位于脚本列表上方。单击该链接后，会显示一个过滤器，您可以按名称和范围筛选脚本。



配置全局脚本



Note:
建议在全局脚本中使用宏函数而不是普通宏，以增强安全性，因为宏不会自动进行转义。

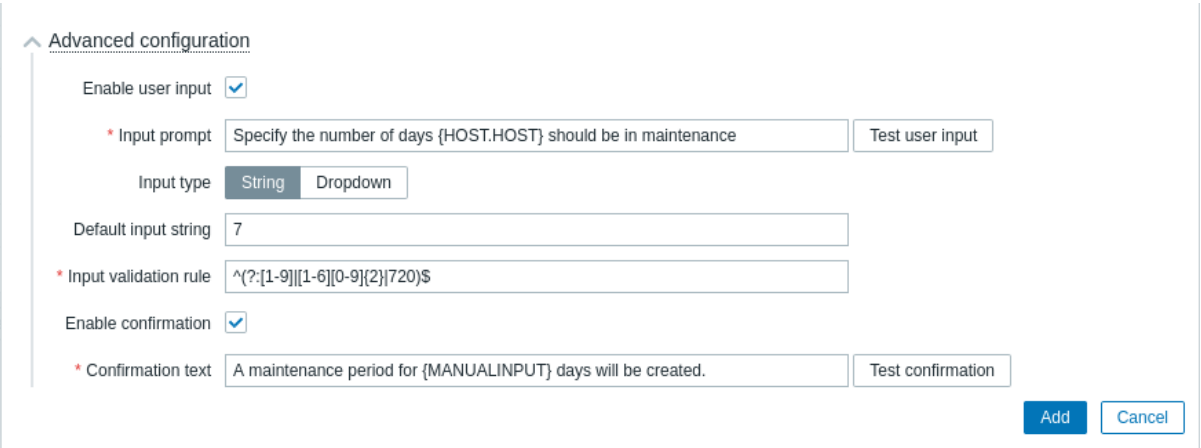
脚本属性：

参数		描述
名称		脚本的唯一名称。 例如：`Clear /tmp filesystem`
范围		脚本的范围 - 动作操作、手动主机操作或手动事件操作。此设置决定脚本的使用位置 - 分别是在动作操作的远程命令中、主机菜单中，还是事件菜单中。 将范围设置为“动作操作”后，所有有权访问告警 > 动作的用户都可以使用该脚本。 如果脚本实际用于某个动作，则无法将其范围更改为“动作操作”以外的其他范围。 宏支持 范围会影响可用宏的范围。例如，脚本支持与用户相关的宏（`{USER.*}`），以便传递启动脚本的用户信息。但是，如果脚本范围为动作操作，则不支持这些宏，因为动作操作会自动执行。 `{MANUALINPUT}` 宏允许您在脚本执行时指定手动输入。手动主机操作和手动事件操作脚本支持此宏。 要了解其他受支持的宏，请在受支持的宏表中搜索“基于触发器的通知和命令/基于触发器的命令”、“手动主机操作脚本”和“手动事件操作脚本”。请注意，如果宏可能解析为空格分隔的值（例如主机名），请务必根据需要加引号。
菜单路径		脚本所需的菜单路径。例如，`Default` 或 `Default/`会将脚本显示在相应目录中。菜单可以嵌套，例如 `Main menu/Sub menu1/Sub menu2`。通过监控部分中的主机/事件菜单访问脚本时，脚本将按照指定的目录进行组织。 仅当范围选择“手动主机操作”或“手动事件操作”时显示此字段。
类型		单击相应按钮选择脚本类型： **URL**、**Webhook**、**Script**、**SSH**、**Telnet** 或**IPMI**命令。 仅当范围选择“手动主机操作”或“手动事件操作”时，**URL** 类型可用。
	脚本类型：**URL**	
	URL	指定 URL，以便从主机菜单或事件菜单快速访问。 支持宏和自定义用户宏。宏支持取决于脚本范围（请参见上面的范围）。 在此字段中使用 `{MANUALINPUT}` 宏，可以在脚本执行时指定手动输入，例如： `http://{MANUALINPUT}/zabbix/zabbix.php?action=dashboard.view` 宏值不得进行 URL 编码。
	在新窗口中打开	确定是在新的浏览器标签页还是当前浏览器标签页中打开 URL。
	脚本类型：**Webhook**	
	参数	以属性-值对的形式指定 webhook 变量。 另请参见：Webhook媒体配置。 参数值支持宏和自定义用户宏。宏支持取决于脚本范围（请参见上面的范围）。
	脚本	单击参数字段或其旁边的铅笔图标时，在打开的模态编辑器中输入 JavaScript 代码。 宏支持取决于脚本范围（请参见上面的范围）。 另请参见：Webhook媒体配置、其他 JavaScript 对象。
	超时	JavaScript 执行超时时间（1-60 秒，默认为 30 秒）。 支持时间后缀（例如 `30s`、`1m`）。
	脚本类型：**Script**	
	执行位置	单击相应按钮，选择在哪个位置执行 Shell 脚本： **Zabbix agent** - 脚本将由 Zabbix agent 在主机上执行（前提是 system.run 监控项已被允许）。 **Zabbix proxy** 或服务器 - 脚本将由 Zabbix proxy 或服务器执行，具体取决于主机由 proxy 还是服务器监控。 如果通过EnableRemoteCommands启用，脚本将在 proxy 上执行。 如果服务器参数EnableGlobalScripts启用全局脚本，脚本将在服务器上执行。 **Zabbix** 服务器 - 脚本仅由 Zabbix 服务器执行。 如果服务器参数EnableGlobalScripts禁用全局脚本，则此选项不可用。
	命令	输入脚本中要执行的命令的完整路径。 宏支持取决于脚本范围（请参见上面的范围）。支持自定义用户宏。
	脚本类型：**SSH**	
	身份验证方法	选择身份验证方法 - 密码或公钥。
	用户名	输入用户名。
	密码	输入密码。 仅当身份验证方法选择“密码”时，此字段可用。
	公钥文件	输入公钥文件的路径。 仅当身份验证方法选择“公钥”时，此字段可用。

参数		描述
	私钥文件	输入私钥文件的路径。 仅当身份验证方法选择 “公钥” 时，此字段可用。
	密码短语	输入密码短语。 仅当身份验证方法选择 “公钥” 时，此字段可用。
	端口	输入 Zabbix 将连接的目标主机上的远程 SSH 服务端口。
	命令	输入命令。 宏支持取决于脚本范围（请参见上面的范围）。支持自定义用户宏。
	脚本类型：**Telnet**	
	用户名	输入用户名。
	密码	输入密码。
	端口	输入 Zabbix 将连接的目标主机上的远程 Telnet 服务端口。
	命令	输入命令。 宏支持取决于脚本范围（请参见上面的范围）。支持自定义用户宏。
	脚本类型：**IPMI**	
	命令	输入 IPMI 命令。 宏支持取决于脚本范围（请参见上面的范围）。支持自定义用户宏。
描述		输入脚本描述。
主机组		选择脚本可用的主机组（或选择全部以应用于所有主机组）。
用户组		选择脚本可用的用户组（或选择全部以应用于所有用户组）。 仅当范围选择 “手动主机操作” 或 “手动事件操作” 时显示此字段。
所需主机权限		选择主机组的权限级别 - 读取或写入。只有具有所需权限级别的用户才能执行脚本。 仅当范围选择 “手动主机操作” 或 “手动事件操作” 时显示此字段。
高级配置		单击高级配置标题以显示高级配置选项。 仅当范围选择 “手动主机操作” 或 “手动事件操作” 时显示此字段。

高级配置

可在可折叠的高级配置部分中使用高级配置选项：



参数	描述
启用用户输入	选中复选框，以便在执行脚本前启用手动用户输入。 手动用户输入将替换脚本中的 {MANUALINPUT} 宏值。 另请参阅：手动用户输入。

参数	描述
输入提示	输入用于提示用户输入自定义内容的文本。此文本将显示在手动输入弹出窗口的输入字段上方。 要预览手动输入弹出窗口，请单击测试用户输入。预览还可用于测试输入字符串是否符合输入验证规则(请参阅下面的参数)。 宏和用户宏的支持取决于脚本的范围（请参阅常规脚本配置参数中的范围)。
输入类型	选择手动输入类型： 字符串 - 单个字符串； 下拉列表 - 从多个下拉选项中选择值。
下拉选项	以逗号分隔的列表形式输入用户输入下拉列表中的唯一值。 要在下拉列表中包含空选项，请在列表的开头、中间或末尾添加额外的逗号。 仅当输入类型选择为“下拉列表”时显示此字段。
默认输入字符串	输入用户输入的默认字符串（或不输入任何内容)。 此字段将根据输入验证规则字段中提供的正则表达式进行验证。 此处输入的值将默认显示在手动输入弹出窗口中。 仅当输入类型选择为“字符串”时显示此字段。
输入验证规则	输入用于验证用户输入字符串的正则表达式。 支持全局正则表达式。 仅当输入类型选择为“字符串”时显示此字段。
启用确认	选中复选框，以便在执行脚本前显示确认消息。此功能对于可能具有危险性的操作（例如重启脚本）或可能需要较长时间的操作尤其有用。
确认文本	输入确认弹出窗口的自定义确认文本，该弹出窗口通过上面的复选框启用（例如，远程系统将被重启。确定要继续吗 ?)。要查看文本的显示效果，请单击字段旁边的测试确认。 支持宏和自定义用户宏。 注意：测试确认消息时不会展开宏。

如果同时配置了手动用户输入和确认消息，它们将在连续的弹出窗口中显示。

手动用户输入

手动用户输入允许在每次执行脚本时提供自定义参数。
这样可以避免仅因参数不同而创建多个类似的用户脚本。

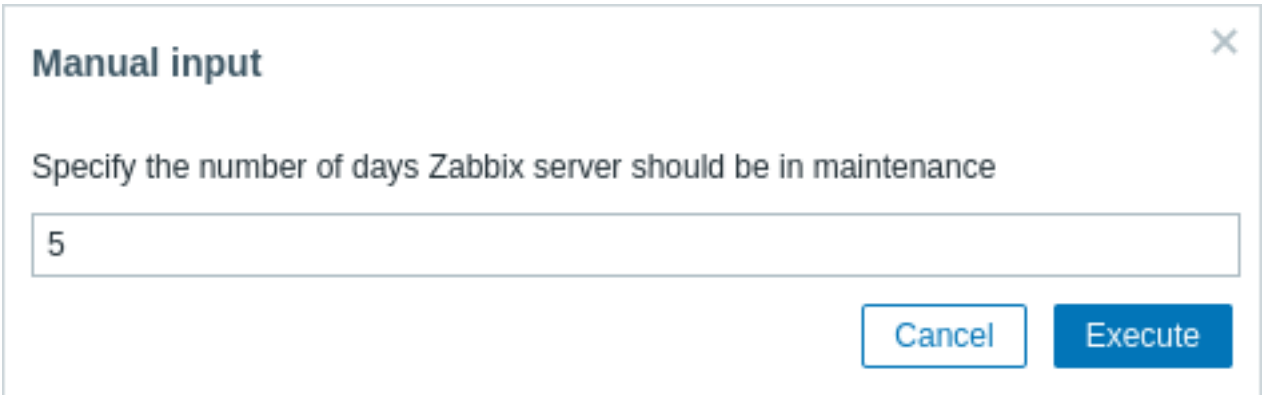
例如，您可能希望在执行脚本时提供不同的整数或不同的 URL 地址。

要启用手动用户输入：

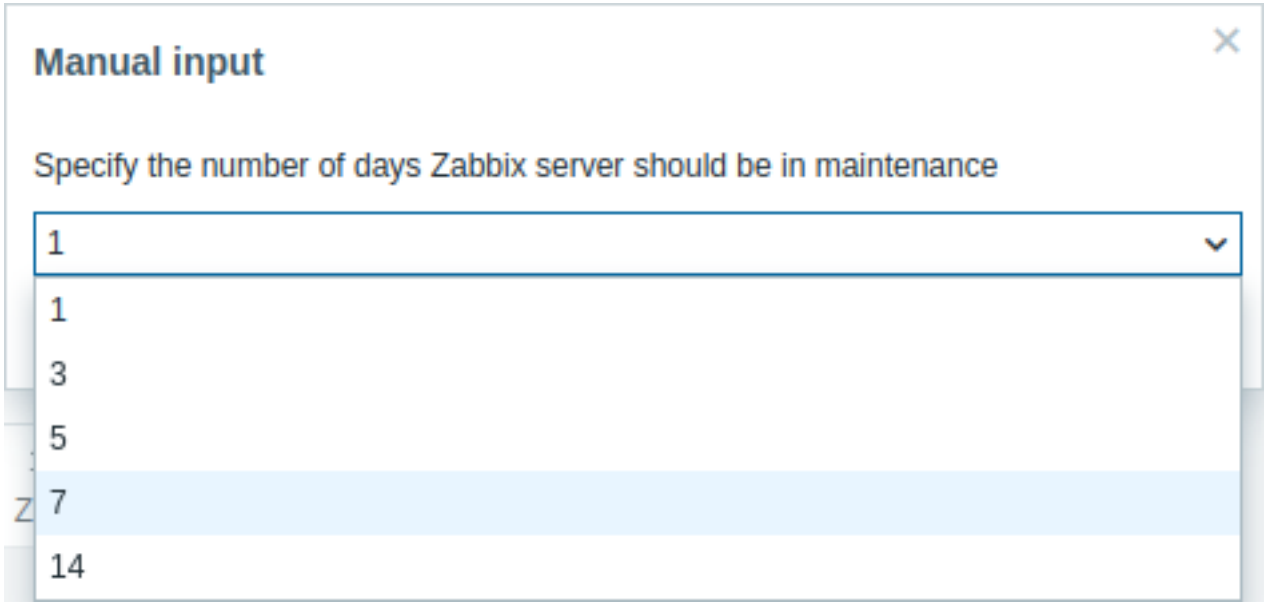
- 在脚本中需要使用的位置（命令、脚本、脚本参数）使用 {MANUALINPUT} 宏；对于 URL 脚本，则在 URL 字段中使用；
- 在高级脚本配置中启用手动用户输入并配置输入选项。

启用用户输入后，在执行脚本之前，系统会向用户显示手动输入弹窗，要求用户提供自定义值。
提供的值将替换脚本中的 {MANUALINPUT}。

根据配置，系统可能会要求用户输入字符串值：



或者从预先确定的选项下拉列表中选择值：



手动用户输入仅适用于范围为“手动主机操作”或“手动事件操作”的脚本。

脚本执行和结果

由 Zabbix 服务器运行的脚本将按照命令执行页面中描述的顺序执行。

脚本运行后，结果会显示在弹出的窗口中。脚本的返回值为标准输出：

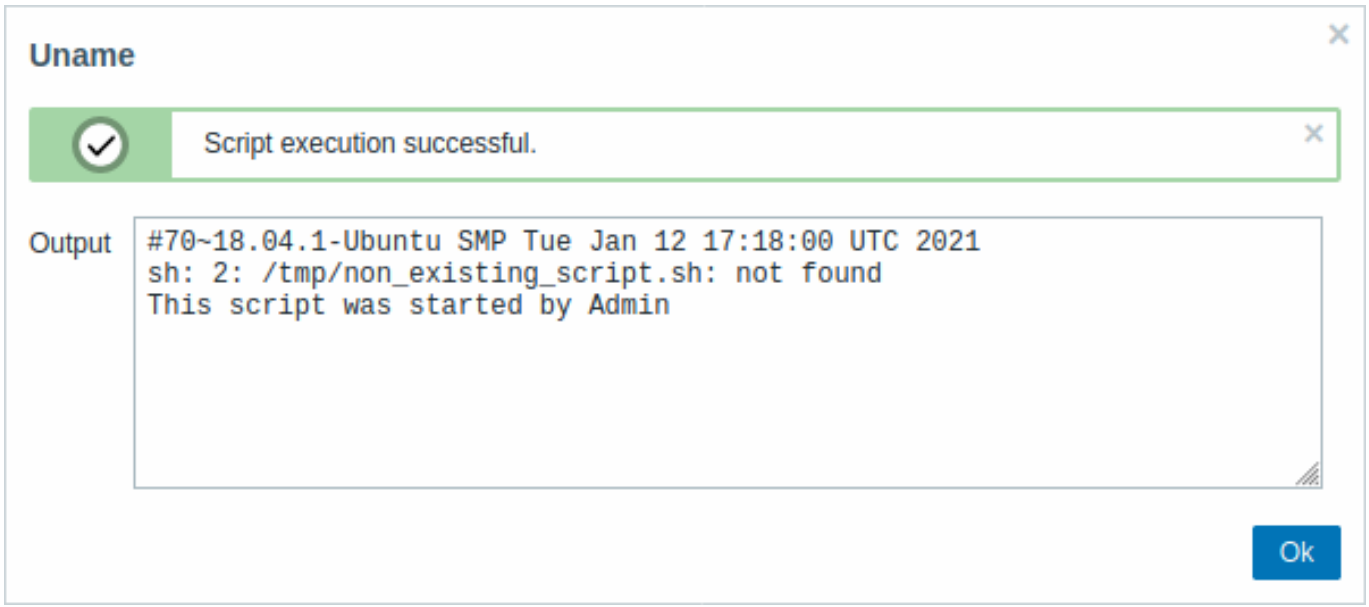
- 如果脚本成功完成（退出代码为 0），返回值限制为 16MB（包括会被截断的尾随空白字符）。
- 如果脚本因错误退出（非零退出代码），返回值为标准错误，限制为 2KB。

默认情况下，Zabbix 不会存储扩展的脚本输出。要保留完整的输出详情，可以在脚本本身中实现日志记录（例如，将输出重定向到本地日志文件）。

请注意，对于在 Zabbix 服务器或 Zabbix proxy 上执行的脚本，数据库限制同样适用。

下面是一个脚本及其结果窗口的示例：

```
uname -v
/tmp/non_existing_script.sh
echo "This script was started by {USER.USERNAME}"
```



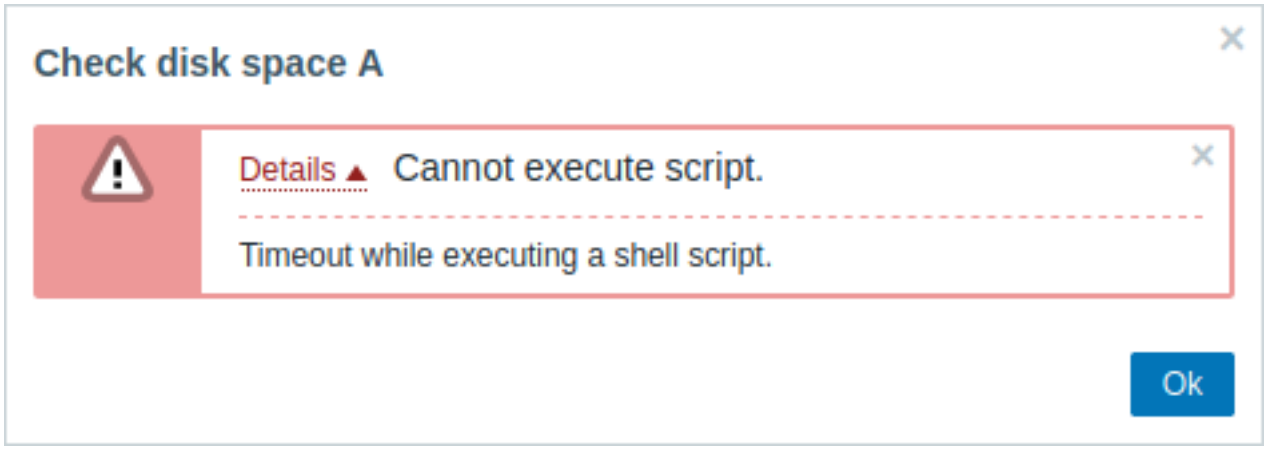
脚本结果中不会显示脚本本身。

Zabbix agent 上的脚本超时

执行脚本时可能会遇到超时的情况。

下面是一个在 Zabbix agent 上运行的脚本及其结果窗口示例：

```
sleep 5
df -h
```



在此情况下，错误消息如下：

```
Timeout while executing a shell script.
```

为避免此类情况，建议优化脚本本身（在上面的示例中为 “5”），而不是调整 Zabbix agent 配置和 Zabbix 服务器配置中的 Timeout 参数。

如果已在 Zabbix agent 配置中更改 Timeout 参数，则会显示以下错误消息：

```
Get value from agent failed: ZBX_TCP_READ() timed out.
```

这意味着已在 Zabbix agent 配置中进行了修改，但还需要在 Zabbix 服务器配置中修改 Timeout 参数。

活动 agent 上的脚本超时

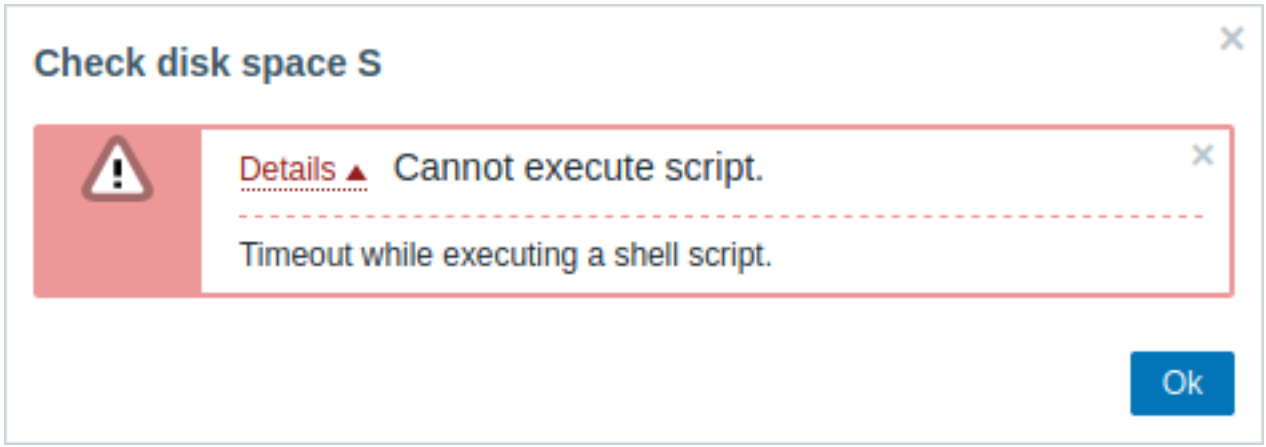
只能在以活动模式运行的 agent 上执行脚本。一旦脚本执行由动作操作 或手动脚本执行触发，该命令就会被包含在活动检查配置中，并在活动 agent 接收到后执行。

手动脚本会连同用于脚本执行的服务器/proxy 超时一起发送给活动 agent。请增加默认的服务器/proxy 脚本执行超时时间。该超时时间必须高于活动检查刷新频率，否则在活动 agent 接收到脚本并能够返回结果之前就会超时。

Zabbix 服务器/proxy 上的脚本超时

请参阅以下在 Zabbix 服务器上运行脚本及其结果窗口的示例：

```
sleep 11
df -h
```



此外，还建议优化脚本本身，而不是通过修改 Zabbix 服务器配置 将 TrapperTimeout 参数调整为相应的值（在本例中为 > 11）。

用户

概览

该菜单包含与在 Zabbix 中配置用户相关的部分。此菜单仅对超级管理员用户类型用户可用。

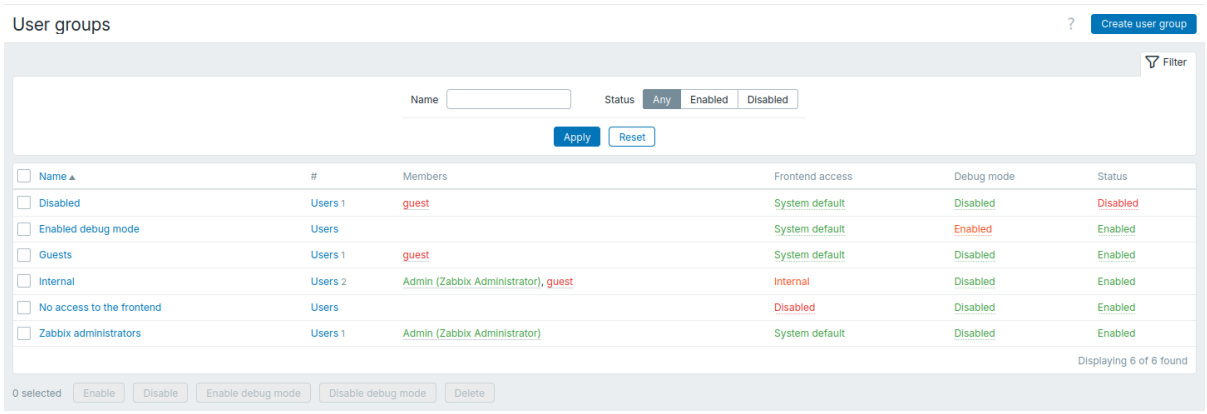
用户组

概述

在 *Users* > *User groups* 部分中维护系统的用户组。

用户组

系统会显示现有用户组及其详细信息列表。



显示的数据：

列	说明
名称	用户组的名称。点击用户组名称可打开用户组配置表单。
#	组中用户的数量。点击用户可在用户列表中显示经过筛选的相应用户。
成员	用户组中各个用户的用户名（括号中显示姓名和姓氏）。点击用户名可打开用户配置表单。已禁用组中的用户以红色显示。
前端访问	显示前端访问级别： 系统默认 - 用户通过 Zabbix、LDAP 或 HTTP 进行身份验证（具体取决于全局设置的身份验证方法）； 内部 - 用户通过 Zabbix 进行身份验证；如果 HTTP 身份验证是全局默认设置，则忽略此选项； **LDAP** - 用户通过 LDAP 进行身份验证；如果 HTTP 身份验证是全局默认设置，则忽略此选项； 已禁用 - 禁止此组访问 Zabbix 前端。 点击当前级别可对其进行更改。
调试模式	显示调试模式状态 - 已启用或已禁用。点击状态可对其进行更改。
状态	显示用户组状态 - 已启用或已禁用。点击状态可对其进行更改。

要配置新的用户组，请点击右上角的创建用户组按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

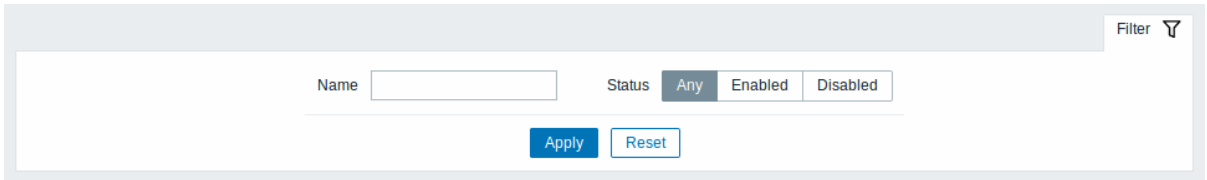
- 启用 - 将用户组状态更改为 已启用
- 禁用 - 将用户组状态更改为 已禁用
- 启用调试模式 - 为用户组启用调试模式
- 禁用调试模式 - 为用户组禁用调试模式
- 删除 - 删除用户组

要使用这些选项，请先勾选相应用户组前面的复选框，然后单击所需的按钮。

使用筛选器

您可以使用筛选器仅显示您感兴趣的用户组。为了获得更好的搜索性能，数据搜索时不会解析宏。

Filter 链接位于用户组列表上方。单击该链接后，将显示一个筛选器，您可以按名称和状态筛选用户组。



用户角色

概述

您可以在 用户 > 用户角色部分创建用户角色。

用户角色允许您根据最初选择的用户类型（用户、管理员、超级管理员）创建细粒度的权限。

选择用户类型后，该用户类型的所有可用权限都会被授予（默认处于选中状态）。

只能撤销该用户类型可用权限子集中的权限；无法扩展超出该用户类型可用范围的权限。

不可用权限对应的复选框将显示为灰色；即使用户在浏览器中直接输入该元素的 URL，也无法访问该元素。

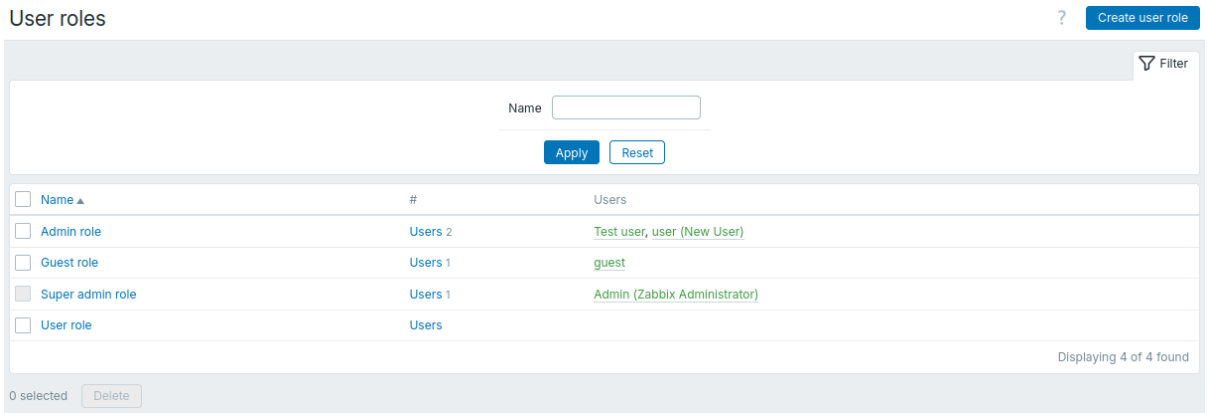
Note:
限制对某些 UI 元素的访问只能阻止打开相应页面，并不会移除用户在界面的其他部分访问底层数据的能力。

用户角色可以分配给系统用户。每个用户只能分配一个角色。

默认用户角色

默认情况下，Zabbix 配置了四种用户角色，这些角色具有预定义的权限集：

- Guest 角色
- User 角色
- Admin 角色
- Super admin 角色



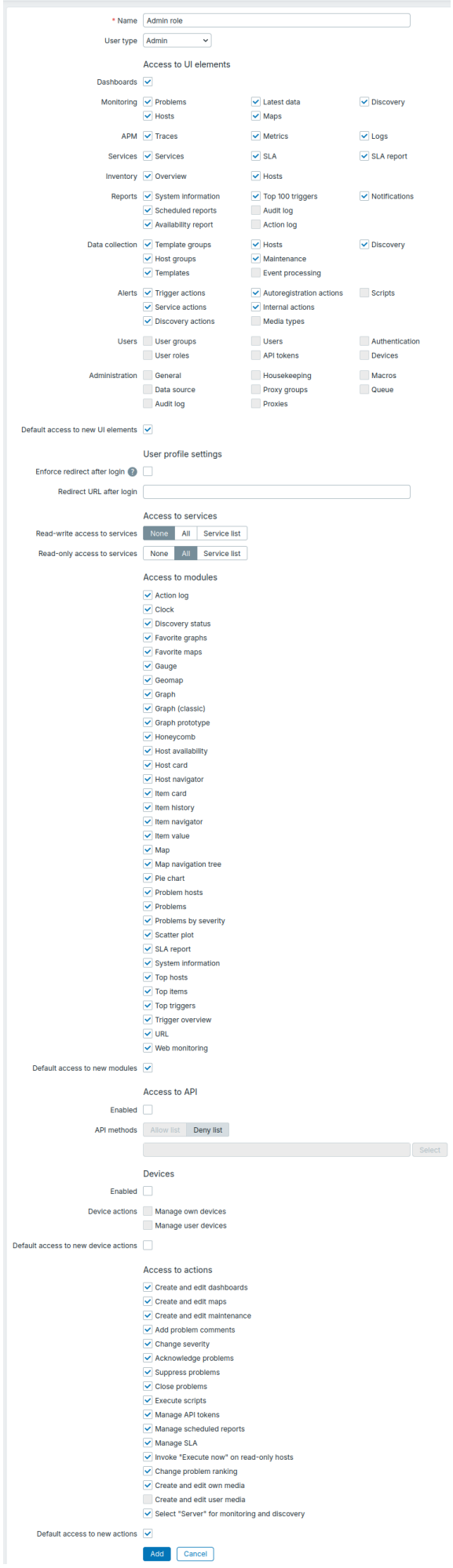
这些角色基于 Zabbix 中的主要用户类型。系统会显示分配了相应角色的所有用户列表。包含在已禁用用户组中的用户将以红色显示。
Guest 角色是一种用户类型角色，仅具有查看部分前端区域的权限。

Note:
默认的 *Super admin* 角色无法修改或删除，因为 Zabbix 中必须至少存在一个具有无限权限的 Super admin 用户。*Super admin* 类型的用户可以修改其自身角色的设置，但不能修改用户类型。

配置

要创建新角色，请点击右上角的 创建用户角色按钮。
要更新现有角色，请点击角色名称以打开配置表单。

User roles


* Name
Admin role
User type
Admin
Access to UI elements
Dashboards
Monitoring
Problems
Latest data
Discovery
Hosts
Maps
APM
Traces
Metrics
Logs
Services
Services
SLA
SLA report
Inventory
Overview
Hosts
Reports
System information
Top 100 triggers
Notifications
Scheduled reports
Audit log
Availability report
Action log
Data collection
Template groups
Hosts
Discovery
Host groups
Maintenance
Templates
Event processing
Alerts
Trigger actions
Autoregistration actions
Scripts
Service actions
Internal actions
Discovery actions
Media types
Users
User groups
Users
Authentication
User roles
API tokens
Devices
Administration
General
Housekeeping
Macros
Data source
Proxy groups
Queue
Audit log
Proxies
Default access to new UI elements
User profile settings
Enforce redirect after login
Redirect URL after login
Access to services
Read-write access to services
None
All
Service list
Read-only access to services
None
All
Service list
Access to modules
Action log
Clock
Discovery status
Favorite graphs
Favorite maps
Gauge
Geomap
Graph
Graph (classic)
Graph prototype
Honeycomb
Host availability
Host card
Host navigator
Item card
Item history
Item navigator
Item value
Map
Map navigation tree
Pie chart
Problem hosts
Problems
Problems by severity
Scatter plot
SLA report
System information
Top hosts
Top items
Top triggers
Trigger overview
URL
Web monitoring
Default access to new modules
Access to API
Enabled
API methods
Allow list
Deny list
Select
Devices
Enabled
Device actions
Manage own devices
Manage user devices
Default access to new device actions
Access to actions
Create and edit dashboards
Create and edit maps
Create and edit maintenance
Add problem comments
Change severity
Acknowledge problems
Suppress problems
Close problems
Execute scripts
Manage API tokens
Manage scheduled reports
Manage SLA
Invoke "Execute now" on read-only hosts
Change problem ranking
Create and edit own media
Create and edit user media
Select "Server" for monitoring and discovery
Default access to new actions
Add
Cancel

系统会显示可用权限。
要撤销某项权限，请取消选中其复选框。

下面介绍了 Zabbix 中每个预配置用户角色的可用权限及其默认设置。

默认权限

对 **UI** 元素的访问权限

默认情况下，对菜单部分的访问权限取决于用户类型。有关详细信息，请参阅 “权限” 页面。

对其他选项的访问权限

下表中描述的某些部分可能会被隐藏：

- 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['modules_config_enabled']` 设置为 `false`，则会隐藏访问模块。
- 如果禁用`EnableMobileDevices`，则会隐藏设备。

参数		描述	默认用户角色 超级管理员角色	管理员角色	用户角色	访客角色
新 *UI* 元素的默认访问权限		此选项指定 Zabbix 升级后如何访问新的菜单部分。模块的现有菜单部分不受影响。	是	是	是	是
对服务的访问权限	对服务的读写访问权限	选择对服务的读写访问权限： 无 - 完全无访问权限 全部 - 对所有服务具有读写访问权限 服务列表 - 选择具有读写访问权限的服务 如果授予读写访问权限，则其优先级高于只读访问权限设置，并由子服务动态继承。	全部	全部	无	无
	对带标签的服务的读写访问权限	指定标签名称以及可选的值，以额外授予对匹配该标签的服务的读写访问权限。 当对服务的读写访问权限参数选择服务列表时，此选项可用。 如果授予读写访问权限，则其优先级高于只读访问权限设置，并由子服务动态继承。				
	对服务的只读访问权限	选择对服务的只读访问权限： 无 - 完全无访问权限 全部 - 对所有服务具有只读访问权限 服务列表 - 选择具有只读访问权限的服务 只读访问权限的优先级低于读写访问权限，并由子服务动态继承。			全部	全部

	对带标签的服务的只读访问权限	指定标签名称以及可选的值，以额外授予对匹配该标签的服务的只读访问权限。 当对服务的只读访问权限参数选择服务列表时，此选项可用。 只读访问权限的优先级低于读写访问权限，并由子服务动态继承。				
对模块的访问权限						
	< 模块名称 >	允许或拒绝访问特定模块。本部分仅显示已启用的模块。 无法授予或限制对当前已禁用模块的访问权限。	是	是	是	是
	新模块的默认访问权限	此选项指定 Zabbix 升级后如何访问新的模块和小组件。 此选项也适用于在管理 > 常规 > 模块部分中添加的模块和小组件。				
对 **API** 的访问权限						
	已启用 *API* 方法	启用或禁用对 API 的访问权限。 选择允许列表以允许搜索字段中指定的 API 方法，或选择拒绝列表以拒绝这些方法。请注意，无法同时允许某些 API 方法并拒绝其他 API 方法。 在搜索字段中开始输入方法名称，然后从自动完成列表中选择方法。 也可以单击选择按钮，从适用于此用户类型的完整列表中选择方法。请注意，如果未选中 “对操作的访问权限” 块中的某个操作，用户将无法使用与该操作相关的 API 方法。 支持通配符。例如：`dashboard.*`（`dashboard.` API 服务的所有方法）、`*`（任意方法）、`*.export`（所有 API 服务中名称带有 `.export` 的方法）。 如果未指定任何方法，则会忽略允许/拒绝列表规则。	否	否	否	否
对操作的访问权限						
	创建和编辑仪表板 创建和编辑拓扑图 创建和编辑维护 添加问题评论 更改严重性 确认问题 抑制问题 关闭问题 执行脚本	清除该复选框还会撤销对相应元素使用 `.create`、`.update` 和 `.delete` API 方法的权限。 清除该复选框还会撤销通过 `event.acknowledge` API 方法执行相应操作的权限。 清除该复选框还会撤销使用 `script.execute` API 方法的权限。	是	是	是 否 是	否

	管理 API 令牌	清除该复选框还会撤销使用所有 `token.` API 方法的权限。				
	管理计划报表	清除该复选框还会撤销使用所有 `report.` API 方法的权限。			否	
	管理 SLA	启用或禁用管理SLA 的权限。				
	在只读主机上调用“立即执行”	允许用户在“最新数据”中对只读主机上的监控项使用“立即执行”选项。			是	
	更改问题排名	允许用户将问题排名从原因更改为症状，或反之。				
	创建和编辑自己的媒介	允许用户创建和编辑自己的媒介。				
	创建和编辑用户媒介	允许用户为其他用户创建和编辑媒介。此选项仅适用于超级管理员用户。		否	否	
	选择“服务器”进行监控和发现	清除该复选框会限制创建由服务器监控的主机和由服务器执行的网络发现规则，同时也会限制使用服务器测试监控项。在主机配置的监控方式参数、网络发现规则配置的发现方式参数以及监控项测试的测试方式参数中，服务器选项将被禁用。		是	否	
	新操作的默认访问权限	此选项指定 Zabbix 升级后如何访问新的操作。		是	是	
设备						
	已启用	启用或禁用对用户设置 > 设备（所有用户均可访问）以及用户 > 设备（仅超级管理员可访问）的访问权限。	是	否	否	否
	管理自己的设备	允许用户在用户设置 > 设备中管理自己的设备。				
	管理用户设备	允许用户在用户 > 设备中管理属于其他用户的设备。				
	新操作的默认访问权限	此选项指定 Zabbix 升级后如何访问新的操作。				

另请参阅：

- 配置用户

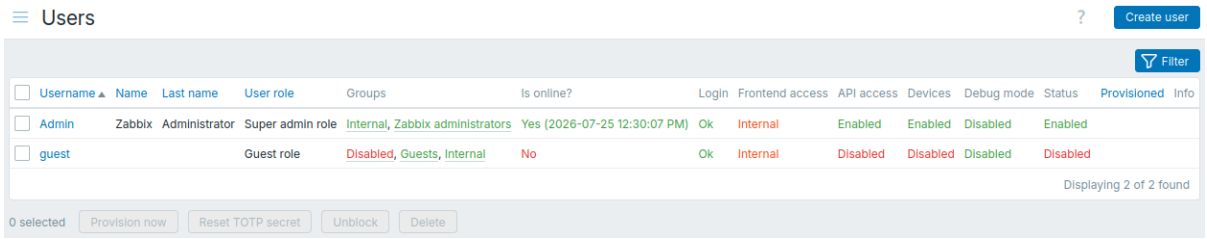
用户

概述

在用户 > 用户部分维护系统用户。

用户

系统会显示现有用户及其详细信息列表。



显示的数据：

列	描述
用户名	用于登录 Zabbix 的用户名。点击用户名可打开用户配置表单。
名字	用户的名字。
姓氏	用户的姓氏。
用户角色	显示用户角色。
用户组	列出用户所属的用户组。点击用户组名称可打开用户组配置表单。禁用的用户组以红色显示。
是否在线？	显示用户的在线状态 - 是或否。括号中显示用户上次活动的时间。
登录	显示用户的登录状态 - 正常或已阻止。如果用户超过管理 > 常规 > 其他部分中设置的失败登录尝试次数（默认为五次），用户可能会被暂时阻止。点击已阻止可以解除对用户的阻止。
前端访问	显示前端访问级别 - 系统默认、内部、*LDAP* 或已禁用，具体取决于整个用户组所设置的级别。
API 访问	显示 API 访问状态 - 已启用或已禁用，具体取决于用户角色所设置的状态。

列	描述
设备	显示设备访问状态 - 已启用或已禁用，具体取决于用户角色所设置的状态。 如果启用了`EnableMobileDevices`，则显示此列。
调试模式	显示调试模式状态 - 已启用或已禁用，具体取决于整个用户组所设置的状态。
状态	显示用户状态 - 已启用或已禁用，具体取决于整个用户组所设置的状态。
已配置	显示用户上次完成配置的日期。 用于通过 LDAP/SAML JIT 配置创建的用户。
信息	显示有关错误的信息。 对于未加入任何用户组的用户，显示黄色警告。 对于没有角色的用户，以及没有角色和用户组的用户，显示红色警告。

要配置新用户，请点击右上角的创建用户按钮。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

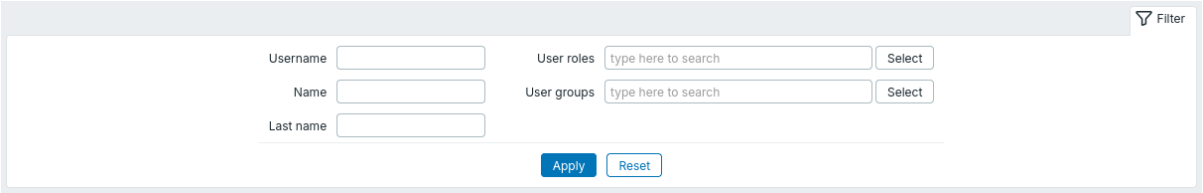
- 立即配置 - 从 LDAP 更新用户信息（仅当选择了LDAP 用户时，此选项才会启用）
- 重置 *TOTP* 密钥 - 重置所有 TOTP 方法的用户 TOTP 密钥，并删除用户会话（仅当启用了MFA 时，此选项才会启用；对于没有 TOTP 密钥的用户，不会删除其会话）
- 解除阻止 - 重新启用被阻止用户的系统访问权限
- 删除 - 删除用户

要使用这些选项，请勾选相应用户前的复选框，然后点击所需的按钮。

使用筛选器

您可以使用筛选器，仅显示您感兴趣的用户。为获得更好的搜索性能，数据将以未解析宏的形式进行搜索。

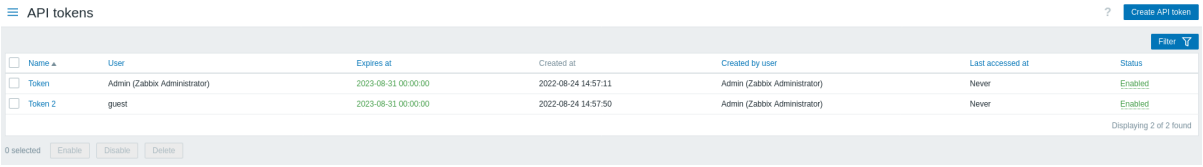
用户列表上方提供了筛选器链接。单击该链接后，将显示筛选器，您可以按用户名、名字、姓氏、用户角色和用户组筛选用户。



API 令牌

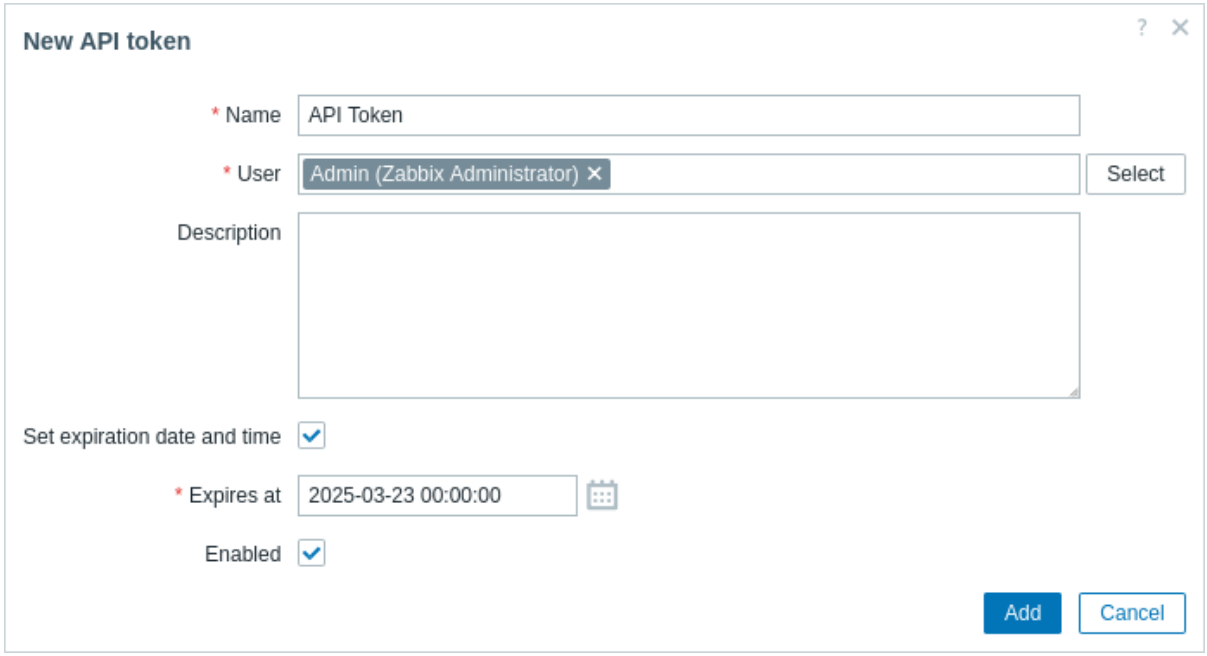
概述

此部分用于创建和管理 API 令牌。
 API 令牌用于在 Zabbix API 中进行身份验证。
 每个 API 令牌仅分配给一个用户，并继承该用户的权限。使用 API 令牌进行身份验证的请求将根据关联用户的角色和权限进行授权。



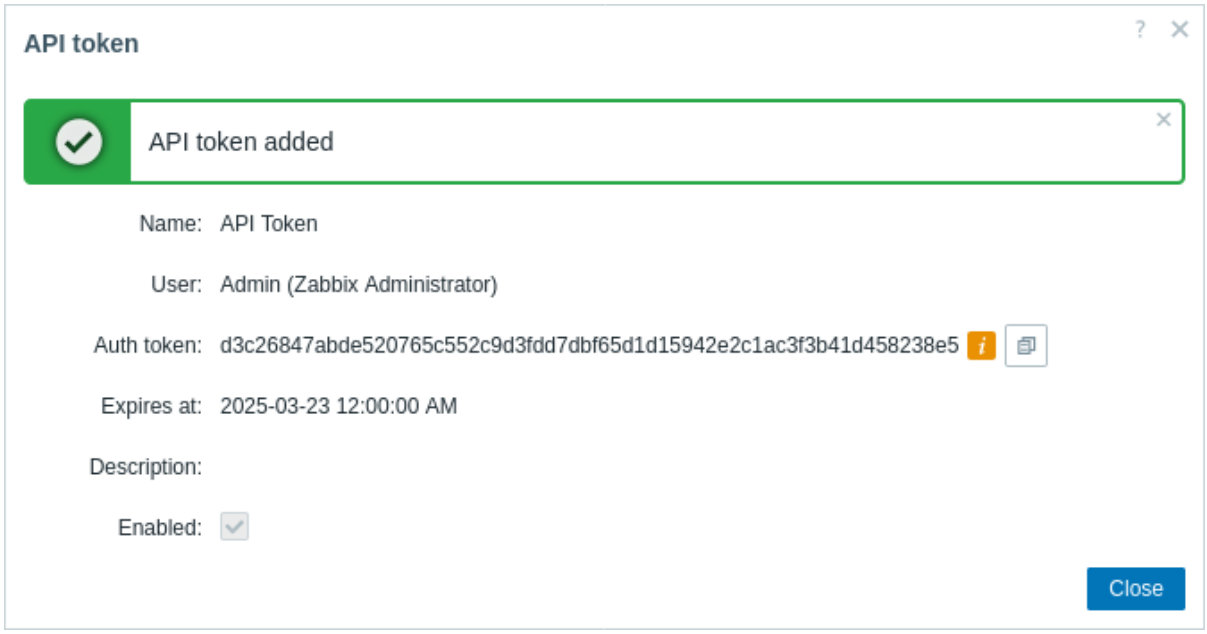
您可以按名称、令牌分配给的用户、过期日期、创建令牌的用户或状态（启用/禁用）筛选 API 令牌。单击列表中的令牌状态，可以快速启用/禁用令牌。您还可以在列表中选择令牌，然后单击列表下方的启用/禁用按钮，批量启用/禁用令牌。

要创建新令牌，请单击右上角的创建 *API* 令牌按钮，然后在令牌配置页面中填写必填字段：



参数	描述
名称	令牌的显示名称。
用户	应为令牌分配的用户。要快速选择用户，请开始输入用户名、名或姓，然后从自动完成列表中选择所需用户。或者，您可以单击选择按钮，从完整的用户列表中选择用户。一个令牌只能分配给一个用户。
描述	可选的令牌描述。
设置过期日期和时间	如果令牌不应设置过期日期，请取消选中此复选框。
过期日期	单击日历图标选择令牌过期日期，或按照 YYYY-MM-DD hh:mm:ss 格式手动输入日期。
已启用	如果需要创建处于禁用状态的令牌，请取消选中此复选框。

单击添加以创建令牌。



请在关闭页面之前复制身份验证令牌值并将其保存到安全位置，然后单击关闭。令牌将显示在列表中。

Warning:
身份验证令牌值之后无法再次查看。它仅在创建令牌后立即可用。如果丢失已保存的令牌，则必须重新生成令牌，此操作将创建新的授权字符串。

单击令牌名称可以编辑名称、描述、过期日期设置或令牌状态。请注意，无法更改令牌分配给的用户。单击更新按钮保存更改。如果令牌丢失或泄露，可以单击重新生成按钮生成新的令牌值。此时将出现确认对话框，要求您确认此操作，因为继续操作后，之前生成的令牌将失效。

没有管理菜单部分访问权限的用户，只有在其用户角色 权限中允许管理 *API* 令牌时，才能在用户配置文件 > *API* 令牌部分中查看和修改分配给他们的令牌详细信息。

身份验证

概述

用户 > 身份验证部分允许您指定 Zabbix 的用户身份验证方法和内部密码要求。

可用的身份验证方法包括内部身份验证、HTTP、LDAP、SAML 和 MFA 身份验证。

默认身份验证

默认情况下，Zabbix 对所有用户使用 内部 Zabbix 身份验证。

可以将系统范围内的默认身份验证方法更改为LDAP。为此，请转到 *LDAP* 选项卡并配置 LDAP 参数，然后返回 身份验证选项卡，将 默认身份验证选择器切换为 LDAP。

请注意，可以在用户组级别对身份验证方法进行细化配置。即使全局设置了 LDAP 身份验证，某些用户组仍可以通过 Zabbix 进行身份验证。这些用户组必须将前端访问设置为 “内部”。

如果全局使用内部身份验证，也可以仅为特定用户组启用 LDAP 身份验证。在这种情况下，可以指定 LDAP 身份验证详细信息，并将其用于特定用户组；然后必须将这些用户组的前端访问设置为 LDAP。如果某个用户属于至少一个启用了 LDAP 身份验证的用户组，则该用户将无法使用内部身份验证方法。

HTTP、SAML 2.0 和 MFA 身份验证方法可以与默认身份验证方法结合使用。

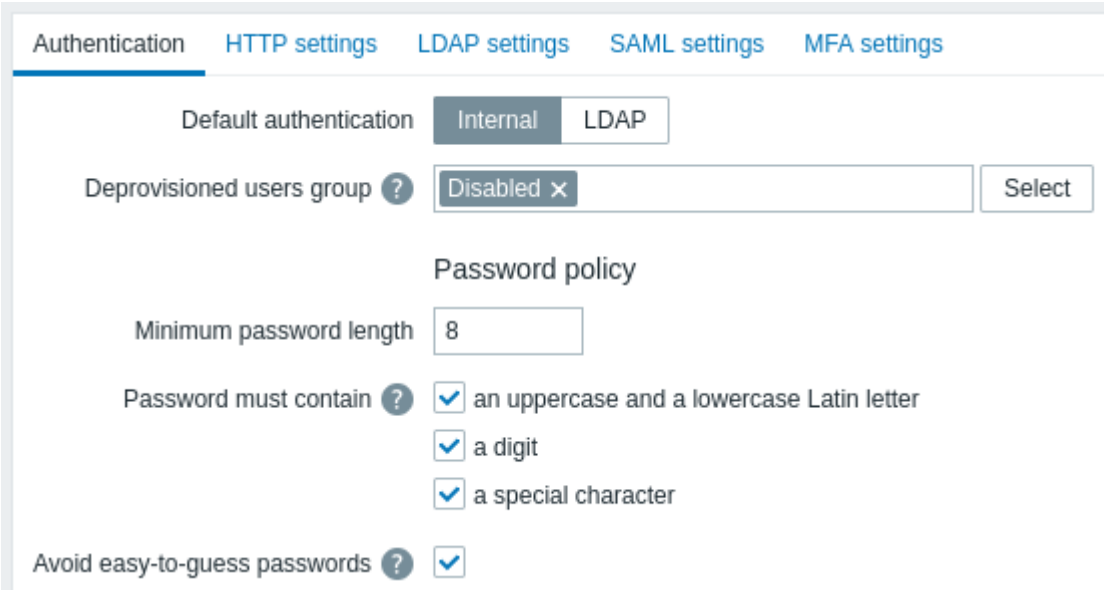
Zabbix 支持即时（JIT）配置，允许在外部用户首次进行身份验证时在 Zabbix 中创建用户账户，并为这些用户账户配置相关信息。LDAP 和 SAML 支持 JIT 配置。

另请参阅：

- HTTP 身份验证
- LDAP 身份验证
- SAML 身份验证
- MFA 身份验证

配置

身份验证选项卡允许您设置默认身份验证方法、为已取消配置的用户指定用户组，以及设置 Zabbix 用户的密码复杂度要求。



配置参数：

参数	描述
默认身份验证	选择 Zabbix 的默认身份验证方法 - 内部或 *LDAP*。
已取消配置的用户组	为已取消配置的用户指定用户组。此设置仅适用于 JIT 配置，涉及从 LDAP 或 SAML 系统在 Zabbix 中创建、但不再需要进行配置的用户。 必须指定一个已禁用的用户组。
最小密码长度	默认情况下，最小密码长度设置为 8。支持的范围：1-70。请注意，超过 72 个字符的密码将被截断。

参数	描述
密码必须包含	选中一个或多个复选框，要求密码中使用指定字符： - 一个大写和一个小写拉丁字母 - 一个数字 - 一个特殊字符 将鼠标悬停在问号上，可查看包含每个选项字符列表的提示。
避免使用易猜密码	如果选中此项，密码将根据以下要求进行检查： - 不得包含用户的名字、姓氏或用户名 - 不得为常见密码或上下文相关密码之一。 常见密码和上下文相关密码列表会根据 NCSC 的 “Top 100k passwords” 列表、SecLists 的 “Top 1M passwords” 列表以及 Zabbix 上下文相关密码列表自动生成。内部用户将无法设置包含在此列表中的密码，因为此类密码由于使用广泛而被视为弱密码。

密码复杂度要求的更改不会影响现有用户密码，但如果现有用户选择更改密码，新密码必须符合当前要求。在用户配置文件中的密码字段旁边，以及通过用户 > 用户菜单访问的用户配置表单中，将显示包含要求列表的提示。

HTTP

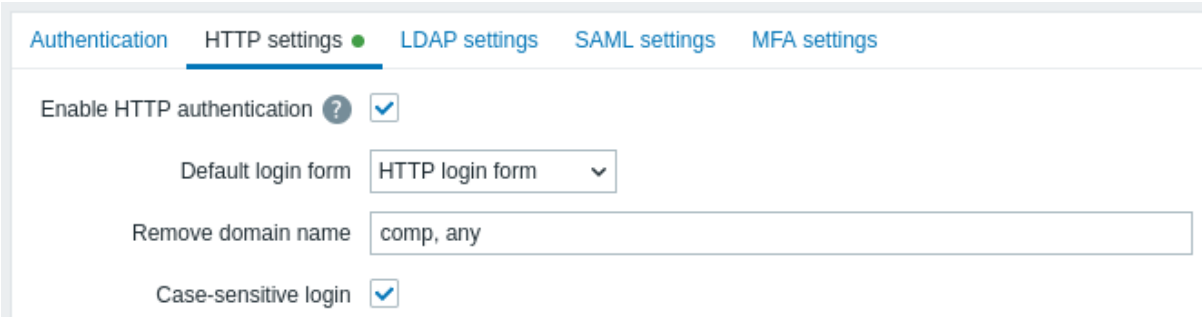
概述

可以使用基于 HTTP 或 Web 服务器的身份验证 （例如：BasicAuthentication、NTLM/Kerberos）来检查用户名和密码。但请注意，用户也必须存在于 Zabbix 中，不过不会使用其 Zabbix 密码。

Attention:
请注意！在启用 Web 服务器身份验证之前，请确保其已正确配置并正常工作。

Note:
可以在前端配置文件（`zabbix.conf.php`）中将 `$ZBX_FEATURE_FLAGS['http_auth_enabled'] = false`，以禁用 HTTP 身份验证配置。在这种情况下，包含 HTTP 身份验证选项的 *HTTP* 设置选项卡将被隐藏。

配置



配置参数：

参数	描述
启用 *HTTP* 身份验证	选中复选框以启用 HTTP 身份验证。将鼠标悬停在 上时，会显示提示框，警告在使用 Web 服务器身份验证的情况下，所有用户（即使其前端访问权限设置为 LDAP/内部）都将由 Web 服务器而不是 Zabbix 进行身份验证。
默认登录表单	指定将未通过身份验证的用户定向到： **Zabbix** 登录表单 - 标准 Zabbix 登录页面。 **HTTP** 登录表单 - HTTP 登录页面。 建议仅对 `index_http.php` 页面启用基于 Web 服务器的身份验证。如果将默认登录表单设置为 “HTTP 登录页面”，并且 Web 服务器身份验证模块在 `$_SERVER` 变量中设置了有效的用户登录名，则用户将自动登录。 支持的 `$_SERVER` 键包括 `PHP_AUTH_USER`、`REMOTE_USER`、`AUTH_USER`。

参数	描述
移除域名	以逗号分隔的域名列表，这些域名应从用户名中移除。 例如，`comp,any` - 如果用户名为'Admin@any' 或'comp\Admin'，用户将以'Admin' 身份登录；如果用户名为'notacompany\Admin'，则登录将被拒绝。
区分大小写的登录名	取消选中复选框，以禁用用户名登录的大小写敏感性（默认启用）。 禁用大小写敏感的登录名后，例如，即使 Zabbix 用户名为"Admin" 或"ADMIN"，也可以使用"admin" 登录。 请注意，如果禁用了大小写敏感的登录名，并且存在多个用户名相似的 Zabbix 用户（例如 Admin 和 admin），这些用户的登录将始终被拒绝，并显示以下错误消息："身份验证失败：提供的凭据不唯一。"

Note:
对于无法使用 HTTP 凭据登录的内部用户（默认登录表单设置为 HTTP 登录表单），如果因此出现 401 错误，可以考虑在基本身份验证指令中添加 `ErrorDocument 401 /index.php?form=default` 行，该行会将用户重定向到常规的 Zabbix 登录表单。

LDAP

概述

外部 LDAP 身份验证 可用于检查用户名和密码。

Zabbix LDAP 身份验证至少可与 Microsoft Active Directory 和 OpenLDAP 配合使用。

如果仅配置了 LDAP 登录，则用户也必须存在于 Zabbix 中，不过其 Zabbix 密码不会被使用。如果身份验证成功，Zabbix 将把本地用户名与 LDAP 返回的用户名属性进行匹配。

用户配置

可以为 LDAP 用户配置 JIT（即时）用户配置。在这种情况下，无需用户已存在于 Zabbix 中。用户首次登录 Zabbix 时，可以创建其用户账户。

当 LDAP 用户输入 LDAP 登录名和密码时，Zabbix 会检查默认 LDAP 服务器中是否存在该用户。如果用户存在且尚未在 Zabbix 中拥有账户，Zabbix 会创建一个新用户，用户即可登录。

通过 JIT 配置创建的用户与创建时设置为默认的 LDAP 服务器（目录）相关联。之后更改默认 LDAP 服务器，不会更改或更新已完成配置的用户所关联的 LDAP 服务器。

Attention:
如果启用了 JIT 配置，则必须在身份验证选项卡中指定一个用于已取消配置用户的用户组。

JIT 配置还允许根据 LDAP 中的更改更新已配置的用户账户。例如，如果用户从一个 LDAP 组移动到另一个 LDAP 组，该用户也会在 Zabbix 中从一个组移动到另一个组；如果用户从 LDAP 组中被移除，该用户也会从 Zabbix 中的相应组移除，并且如果不属于任何其他组，还会被添加到用于已取消配置用户的用户组中。已配置的用户账户会根据配置的配置周期进行更新，或在用户登录 Zabbix 时进行更新。

请注意，后台用户配置由 Zabbix 前端在用户与其交互时，或用户至少在浏览器中打开一个前端页面时执行。没有专用的后台进程用于配置用户。

LDAP 支持以下三种绑定到目录以进行身份验证和搜索的方式：

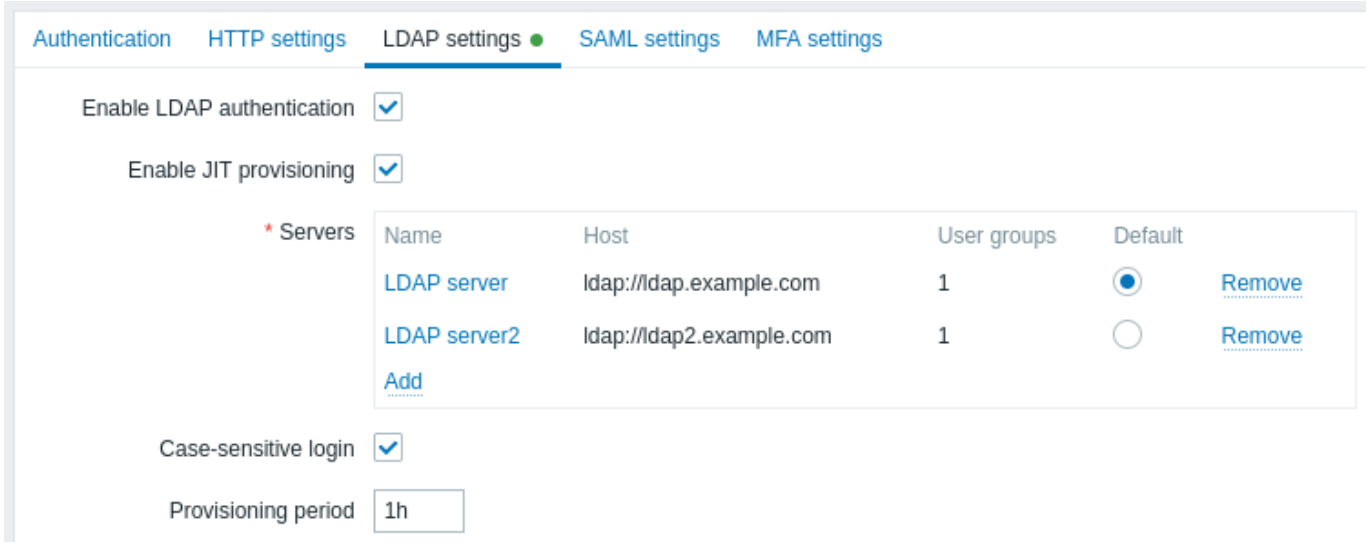
- 匿名绑定 - 不提供绑定 *DN* / 绑定密码，且 LDAP 服务器允许匿名查询。
- 专用绑定用户（服务账户）- 在绑定 *DN* / 绑定密码中设置特定的 LDAP 账户，Zabbix 使用该账户进行搜索和用户配置。这是推荐且最灵活的选项，因为 Zabbix 可以在不使用最终用户凭据的情况下执行搜索和后台用户配置。
- 直接用户绑定 - Zabbix 使用用户登录时输入的凭据执行绑定（未配置绑定 *DN* / 绑定密码）；通过在基本 DN 中包含 `uid=%{user}` 等占位符进行配置。在此模式下，Zabbix 仅在交互式登录期间访问用户密码。因此，需要在用户登录会话之外向 LDAP 进行身份验证的配置操作（例如使用立即配置按钮，或在用户未主动登录时运行后台用户配置）无法进行身份验证，因此无法正常工作。直接用户绑定的配置和更新仅在用户登录时执行。

多个服务器

如有需要，可以定义多个 LDAP 服务器。例如，可以使用不同的服务器对不同的用户组进行身份验证。配置好 LDAP 服务器后，在用户组配置中即可为相应的用户组选择所需的 LDAP 服务器。

如果某个用户属于多个用户组且涉及多个 LDAP 服务器，则将使用 LDAP 服务器列表中按名称升序排序后的第一个服务器进行身份验证。

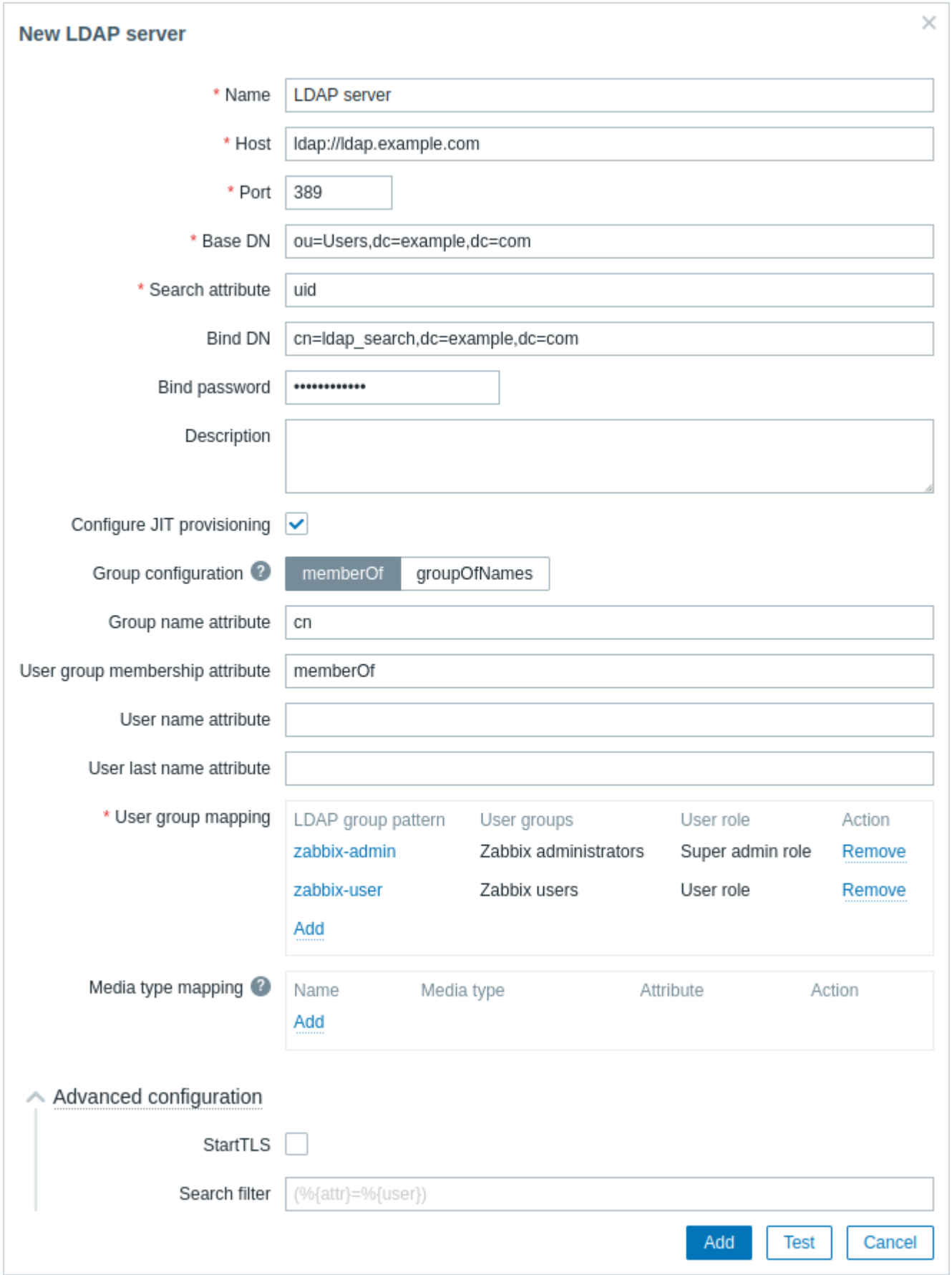
配置



配置参数：

参数	描述
启用 *LDAP* 身份验证	选中复选框以启用 LDAP 身份验证。
启用 *JIT* 配置	选中复选框以启用 JIT 配置。
服务器	点击添加以配置 LDAP 服务器（请参见下方的LDAP 服务器配置）。
区分大小写的登录	取消选中复选框，以禁用用户名登录的大小写区分功能（默认启用）。 禁用区分大小写的登录后，即使 Zabbix 用户名为”Admin” 或”ADMIN”，也可以使用”admin” 登录。 请注意，如果禁用区分大小写的登录，并且存在多个用户名相似的 Zabbix 用户（例如 Admin 和 admin），这些用户的登录将始终被拒绝，并显示以下错误消息：” 身份验证失败：提供的凭据不唯一。”。
配置周期	设置配置周期，即登录用户在使用前端期间进行配置的频率。

LDAP 服务器配置



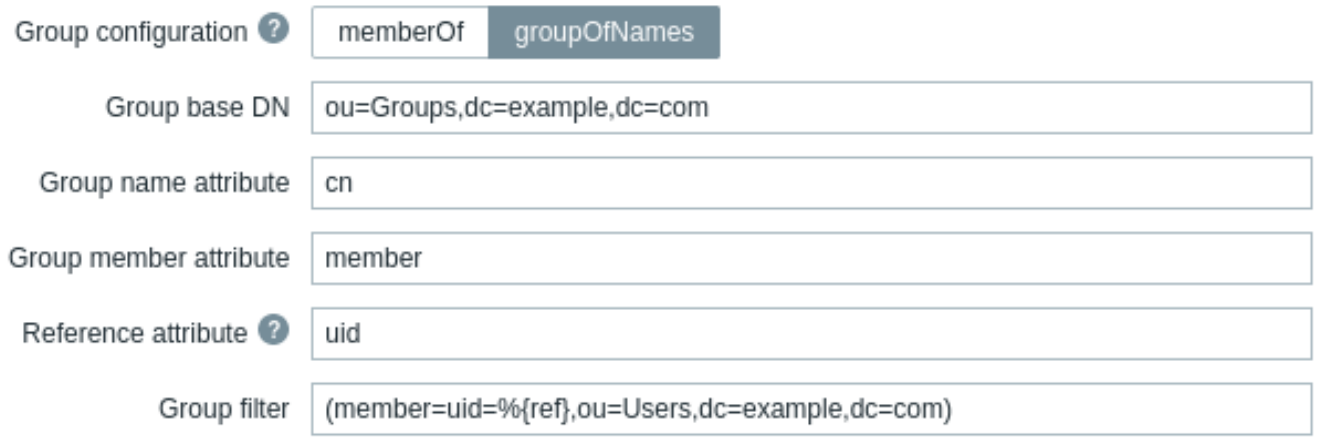
LDAP 服务器配置参数：

Parameter	Description
Name	Zabbix 配置中 LDAP 服务器的名称。

Parameter	Description
Host	LDAP 服务器的主机名、IP 或 URI。示例：ldap.example.com、127.0.0.1、ldap://ldap.example.com 对于安全 LDAP 服务器，请使用 *ldaps* 协议和主机名。示例：ldaps://ldap.example.com 在 OpenLDAP 2.x.x 及更高版本中，可以使用完整的 LDAP URI，格式为 ldap://hostname:port 或 ldaps://hostname:port。
Port	LDAP 服务器的端口。默认值为 389。 对于安全 LDAP 连接，端口号通常为 636。 使用完整 LDAP URI 时不使用此项。
Base DN	LDAP 服务器中用户账户的基础路径： ou=Users,ou=system（适用于 OpenLDAP）， DC=company,DC=com（适用于 Microsoft Active Directory） uid=%{user},dc=example,dc=com（用于直接用户绑定，见下方说明）
Search attribute	用于搜索的 LDAP 账户属性： uid（适用于 OpenLDAP）， sAMAccountName（适用于 Microsoft Active Directory）
Bind DN	用于在 LDAP 服务器上绑定和搜索的 LDAP 账户，示例： uid=ldap_search,ou=system（适用于 OpenLDAP）， CN=ldap_search,OU=user_group,DC=company,DC=com（适用于 Microsoft Active Directory） 也支持匿名绑定。请注意，匿名绑定可能会将域配置暴露给未授权用户（用户、计算机、服务器、组、服务等信息）。出于安全考虑，请在 LDAP 主机上禁用匿名绑定，并改用经过身份验证的访问。
Bind password	用于在 LDAP 服务器上绑定和搜索的该账户的 LDAP 密码。
Description	LDAP 服务器的描述。
Configure JIT provisioning	勾选此复选框以显示与 JIT provisioning 相关的选项。
Group configuration	选择组配置方法： **memberOf** - 通过搜索用户及其组成员属性 **groupOfNames** - 通过 member 属性搜索组 请注意，memberOf 更推荐，因为速度更快；如果 LDAP 服务器不支持 `memberOf` 或需要组过滤，请使用 groupOfNames。
Group name attribute	指定用于从 `memberOf` 属性中的所有对象获取组名的属性（参见 *User group membership attribute* 字段） 组名对于用户组映射是必需的。
User group membership attribute	指定包含用户所属组信息的属性（例如 `memberOf`）。 例如，memberOf 属性可能包含如下信息： `memberOf=cn=zabbix-admin,ou=Groups,dc=example,dc=com` 此字段仅适用于 memberOf 方法。
User name attribute	指定包含用户名字的属性。
User last name attribute	指定包含用户姓氏的属性。
User group mapping	将 LDAP 用户组模式映射到 Zabbix 用户组和用户角色。 这是确定已 provisioning 用户在 Zabbix 中将获得哪个用户组/角色所必需的。 单击 *Add* 以添加映射。 *LDAP group pattern* 字段支持通配符。组名必须与现有组匹配。 如果某个 LDAP 用户匹配多个 Zabbix 用户组，则该用户会成为所有这些组的成员。 如果某个用户匹配多个 Zabbix 用户角色，则该用户将获得其中权限级别最高的角色。
Media type mapping	将用户的 LDAP media 属性（例如电子邮件）映射到 Zabbix 用户 media，用于发送通知（该属性值将用作 media 的*Send to* 字段）。
Advanced configuration	单击 *Advanced configuration* 标题以显示高级配置选项（见下文）。
StartTLS	勾选复选框以在连接到 LDAP 服务器时使用 StartTLS 操作。如果服务器不支持 StartTLS，连接将失败。 StartTLS 不能与使用 *ldaps* 协议的服务器一起使用。
Search filter	在 LDAP 中对用户进行身份验证时定义自定义字符串。支持以下占位符： `%{attr}` - 搜索属性名（uid、sAMAccountName） `%{user}` - 用于身份验证的用户名值 例如，要在区分大小写的 LDAP 或 Microsoft Active Directory 环境中执行区分大小写的搜索，可以将字符串定义为： `(%{attr}:caseExactMatch:=%{user})`。 如果未自定义过滤器，LDAP 将使用默认值：`(%{attr}=%{user})`。

Note:
要为直接用户绑定配置 LDAP 服务器，请在 *Base DN* 参数中追加属性 uid=%{user}（例如，*uid=%{user},dc=example,dc=com*），并将 *BindDN* 和 *Bind password* 参数留空。进行身份验证时，占位符%{user}将被登录时输入的用户名替换。
 使用直接用户绑定时，Zabbix 只能在交互式登录期间访问用户的凭据。因此，不依赖交互式登录即可运行的 provisioning 任务（例如 *Provision now* 按钮或`user.provision` API 方法）会忽略使用直接用户绑定进行身份验证的用户，因为 Zabbix 无法代表他们进行绑定。如果你需要 provisioning 和 LDAP 搜索在前端中无需最终用户登录即可工作，请使用匿名绑定或专用绑定用户（服务账户）。

以下字段是作为 *Group configuration* 方法的"groupOfNames" 所特有的：



Parameter	Description
Group base DN	LDAP 服务器中组的基础路径。
Group name attribute	指定用于从组的指定基础路径中获取组名的属性。 组名对于用户组映射是必需的。
Group member attribute	指定包含 LDAP 中组成员信息的属性（例如 `member`）。
Reference attribute	指定组过滤器的引用属性（参见 *Group filter* 字段）。 然后在组过滤器中使用 `%{ref}` 来获取此处指定属性的值。
Group filter	指定用于检索用户所属组的过滤器。 例如，(`member=uid=%{ref},ou=Users,dc=example,dc=com`) 会在组的 member 属性为 `uid=User1,ou=Users,dc=example,dc=com` 时匹配"User1"，并返回"User1" 所属的组。

Warning:
如果证书出现问题，要使安全 LDAP 连接（ldaps）正常工作，可能需要在 `/etc/openldap/ldap.conf` 配置文件中添加一行 `TLS_REQCERT allow`。这可能会降低与 LDAP 目录连接的安全性。

Note:
建议创建一个单独的 LDAP 账户（*Bind DN*），以最小权限在 LDAP 服务器上执行绑定和搜索，而不是使用真实用户账户（用于登录 Zabbix 前端）。
 这种方式更安全，并且当用户在 LDAP 服务器上修改自己的密码时，不需要更改 *Bind password*。
 在上表中，这就是 *ldap_search* 账户名。

测试访问权限

测试按钮可用于测试用户访问权限：

参数	描述
登录名	要测试的 LDAP 用户名（预填充为当前 Zabbix 前端中的用户名）。此用户名必须存在于 LDAP 服务器中。 如果无法对测试用户进行身份验证，Zabbix 将不会启用 LDAP 身份验证。
用户密码	要测试的 LDAP 用户密码。

SAML

概述

SAML 2.0 身份验证 可用于登录 Zabbix。

如果配置 SAML 登录时未启用 JIT 用户预配，则用户必须已存在于 Zabbix 中；用户的 Zabbix 密码不会用于身份验证。在 SAML 身份验证成功后，Zabbix 会将从 SAML 接收到的用户名属性与现有的本地 Zabbix 用户名进行匹配。

用户预配

可以为 SAML 用户配置 JIT（just-in-time，实时）用户预配。
在这种情况下，不要求用户必须已存在于 Zabbix 中。
当用户首次登录 Zabbix 时，可以创建其用户账户。

> **Attention:**
> 如果启用了 JIT 预配，则必须在 *Authentication* 选项卡中为已取消预配的用户指定一个用户组。

除了 JIT 预配之外，还可以启用并配置 SCIM（System for Cross-domain Identity Management，跨域身份管理系统）预配——对那些通过用户预配创建的用户进行持续的用户账户管理。
SCIM 预配需要一个 Zabbix API token（具有 Super admin 权限）用于登录 Zabbix 时的身份验证。

例如，如果某个用户从一个 SAML 组移动到另一个组，那么该用户在 Zabbix 中也会从一个组移动到另一个组；如果某个用户从某个 SAML 组中被移除，那么该用户在 Zabbix 中也会从该组中被移除，并且如果不属于任何其他组，则会被添加到已取消预配用户的用户组中。

如果启用了并配置了 SCIM，则 SAML 用户会在登录 Zabbix 时被预配，并根据 SAML 中的变更持续更新。
已存在的 SAML 用户不会被预配，且只有已预配的用户才会被更新。
请注意，只有有效的媒介会在用户被预配或更新时添加到该用户。

如果未启用 SCIM，则 SAML 用户会在登录 Zabbix 时被预配（并在之后更新）。

> **Note:**
> 如果启用了 SAML 身份验证，用户将可以选择本地登录或通过 SAML 单点登录。
> 如果使用 JIT 预配，则只能使用单点登录。

设置身份提供商

为了与 Zabbix 配合使用，需要按以下方式配置 SAML 身份提供商（onelogin.com、auth0.com、okta.com 等）：

- *Assertion Consumer URL* 应设置为 `<path_to_zabbix_ui>/index_sso.php?acs`
- *Single Logout URL* 应设置为 `<path_to_zabbix_ui>/index_sso.php?sls`

`<path_to_zabbix_ui>` 示例：`https://example.com/zabbix/ui`，`http://another.example.com/zabbix`，`http://<any_publi`

设置 Zabbix

> **Attention:**
> 如果要在前端使用 SAML 身份验证，则必须安装 php-openssl。

要使用 SAML 身份验证，应按以下方式配置 Zabbix：

1. 提供 SP 私钥和证书。必须提供 IdP 证书（粘贴证书内容或选择文件），才能通过前端启用 SAML。如果选择了签名或加密选项，则必须提供 SP 证书和 SP 私钥。

如果在前端配置文件（`zabbix.conf.php`）中设置了 `$SSO['CERT_STORAGE'] = 'database'`，则可以在 SAML 配置期间通过前端粘贴证书文本或上传证书文件，无需在文件系统中存储文件。

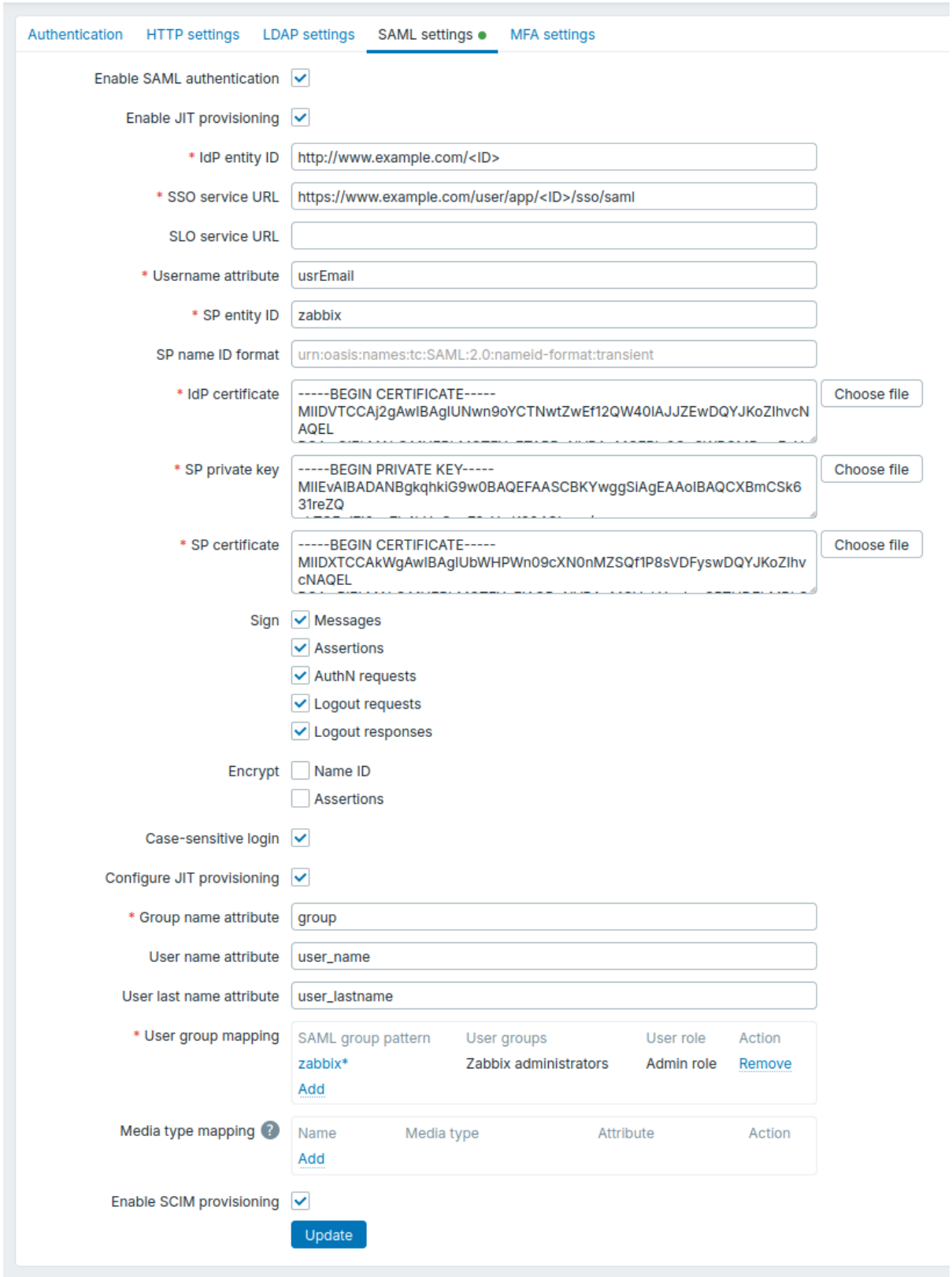
如果在配置文件中设置了 `$SSO['CERT_STORAGE'] = 'file'`，则必须在文件系统中提供证书（默认位于 `ui/conf/certs`，或位于配置文件中配置的路径），并且前端不会将证书存储在数据库中。请注意，如果未设置 `$SSO['CERT_STORAGE']` 或该设置已被注释，则默认使用文件存储。

默认情况下，Zabbix 将在以下位置查找：

- ui/conf/certs/sp.key - SP 私钥文件
- ui/conf/certs/sp.crt - SP 证书文件
- ui/conf/certs/idp.crt - IDP 证书文件

> **Attention:**
> 可以通过前端粘贴文本或上传文件来导入证书材料。尽管在某些环境中，为方便起见允许导入未加密的内容，但强烈建议使用加密传输（例如，为 Zabbix 前端启用 HTTPS/TLS）。当 SAML 证书/密钥存储在数据库中时，请为数据库连接启用 TLS/SSL，以保护应用程序与数据库之间的流量。
 请加密可能包含 SAML 证书或私钥的数据库备份，并限制对备份文件以及存储 SAML 材料的数据库表的访问权限。将数据库用户权限限制为所需的最低权限。

2. 所有最重要的设置都可以在 Zabbix 前端中配置。但是，也可以在配置文件中指定其他设置。



Zabbix 前端中可用的配置参数：

参数	描述
启用 *SAML* 身份验证	选中复选框以启用 SAML 身份验证。
启用 *JIT* 配置	选中复选框以启用 JIT 用户配置。
IDP 实体 *ID*	SAML 身份提供商中的唯一实体标识符。
SSO 服务 *URL*	用户登录时将被重定向到的 URL。
SLO 服务 *URL*	用户注销时将被重定向到的 URL。如果留空，则不会使用 SLO 服务。

参数	描述
用户名属性	登录 Zabbix 时用作用户名的 SAML 属性。 支持的值列表由身份提供商决定。 示例： uid userprincipalname samaccountname username userusername urn:oid:0.9.2342.19200300.100.1.1 urn:oid:1.3.6.1.4.1.5923.1.1.1.13 urn:oid:0.9.2342.19200300.100.1.44
SP 实体 *ID*	唯一的服务提供商标识符（如果不匹配，则操作将被拒绝）。 可以指定 URL 或任意字符串数据。
SP 名称 *ID* 格式	在响应中请求特定的名称 ID 格式。 示例： urn:oasis:names:tc:SAML:2.0:nameid-format:persistent urn:oasis:names:tc:SAML:2.0:nameid-format:unspecified urn:oasis:names:tc:SAML:2.0:nameid-format:transient
IdP 证书	用于设置 SAML 单点登录（SSO）服务的服务提供商（SP）证书内容。
SP 私钥	用于设置 SAML 单点登录（SSO）服务的服务提供商（SP）私钥内容。用于与身份提供商（IdP）进行安全身份验证和数据交换。
SP 证书	用于设置 SAML 单点登录（SSO）服务的服务提供商（SP）证书内容。
签名	取消选中复选框，以禁用相应实体的 SAML 签名： 消息 断言 身份验证请求 注销请求 注销响应
加密	选中复选框，以选择要启用 SAML 加密的实体： 名称 *ID* 断言
区分大小写的登录	取消选中复选框，以禁用用户名登录的大小写敏感性（默认启用）。 禁用区分大小写的登录后，例如，即使 Zabbix 用户为”Admin” 或”ADMIN”，也可以使用”admin” 登录。 请注意，如果禁用了区分大小写的登录，并且存在多个用户名相似的 Zabbix 用户（例如 Admin 和 admin），则这些用户的登录将始终被拒绝，并显示以下错误消息：”Authentication failed: supplied credentials are not unique.”
配置 *JIT* 配置	选中此复选框，以显示与 JIT 用户配置相关的选项。
组名称属性	指定用于 JIT 用户配置的组名称属性。
用户名属性	指定用于 JIT 用户配置的用户名属性。
用户姓氏属性	指定用于 JIT 用户配置的用户姓氏属性。
用户组映射	将 SAML 用户组模式映射到 Zabbix 用户组和用户角色。 这是确定配置的用户将在 Zabbix 中获得哪个用户组/角色所必需的。 点击添加以添加映射。 *SAML* 组模式字段支持通配符。组名称必须与现有组匹配。 如果 SAML 用户匹配多个 Zabbix 用户组，则该用户将成为所有这些用户组的成员。 如果用户匹配多个 Zabbix 用户角色，则该用户将获得其中最高的权限级别。
媒体类型映射	将用户的 SAML 媒体属性（例如电子邮件）映射到 Zabbix 用户媒体，以发送通知。
启用 *SCIM* 配置	选中此复选框以启用 SCIM 2.0 配置。

以下示例介绍了如何配置 SAML 身份提供商，以便登录 Zabbix 并在 Zabbix 中配置用户：

- Microsoft Azure AD
- Okta
- Onelogin

SCIM 配置说明

对于 SCIM 配置，请在身份提供商端指定 Zabbix 前端的路径，并在其后追加 `api_scim.php`，即：

`https://<path-to-zabbix-ui>/api_scim.php`

Zabbix 中使用的用户属性（用户名、用户名字、用户姓氏和媒介属性）需要添加为自定义属性；如有必要，外部命名空间应与用户架构相同：`urn:ietf:params:scim:schemas:core:2.0:User`。

高级设置

可以在 Zabbix 前端配置文件（`zabbix.conf.php`）中配置其他 SAML 参数：

- `$SSO['SP_KEY']`
- `$SSO['SP_CERT']`
- `$SSO['IDP_CERT']`
- `$SSO['SETTINGS']`

配置示例

示例 1：使用 TLS/SSL 通过负载均衡器连接到 Zabbix 实例

如果使用负载均衡器连接到 Zabbix 实例，并且负载均衡器使用 TLS/SSL 而 Zabbix 未使用，请按如下方式指定 `baseurl`、`strict` 和 `use_proxy_headers` 参数：

```
$SSO['SETTINGS'] = [
    'strict' => false,
    'baseurl' => 'https://zabbix.example.com/zabbix/',
    'use_proxy_headers' => true
];
```

示例 2：配置自定义 SAML 签名和摘要算法

```
$SSO['SETTINGS'] = [
    'security' => [
        'signatureAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha384'
        'digestAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#sha384',
        // ...
    ],
    // ...
];
```

示例 3：使用 ADFS 配置基于 Kerberos 身份验证的 SSO

```
$SSO['SETTINGS'] = [
    'security' => [
        'requestedAuthnContext' => [
            'urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos',
        ],
        'requestedAuthnContextComparison' => 'exact'
    ]
];
```

在这种情况下，在 SAML 配置的 *SP name ID* 字段中设置：

```
urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
```

MFA

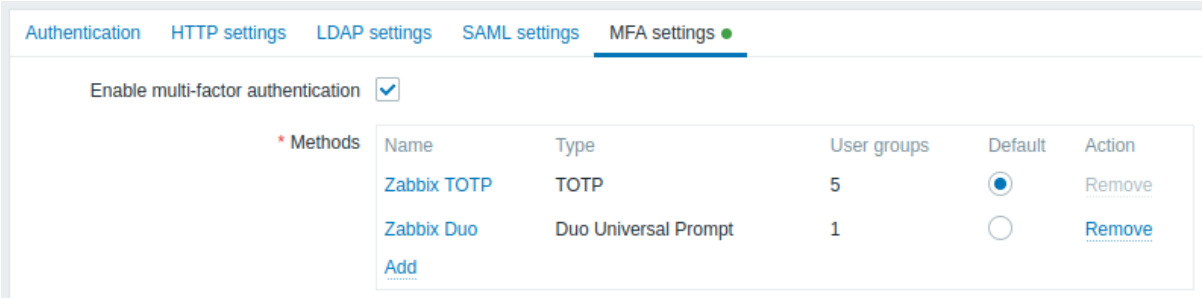
概述

多因素认证（MFA）可用于登录 Zabbix，提供比仅使用用户名和密码更高的安全层级。

使用 MFA，用户必须在 Zabbix 中存在，登录时必须提供 Zabbix 凭据，并且还必须通过其他手段证明其身份，通常是用户手机上认证器应用生成的代码。

多种 MFA 方法可供选择，允许用户选择最符合其安全需求和偏好的选项。这些方法包括基于时间的一次性密码（TOTP）和 Duo Universal Prompt。

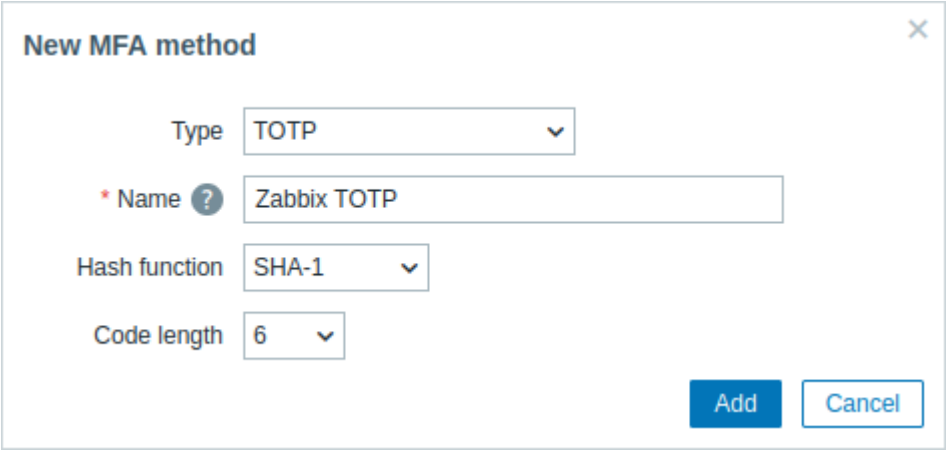
配置



配置参数：

参数	描述
启用多因素认证	勾选此框以启用多因素认证。
方法	点击 添加来配置一个多因素认证方法（参见下面的方法配置）。

方法配置



方法配置参数：

Parameter	Description
Type	选择 MFA 方法的类型： **TOTP** - 使用身份验证器应用生成基于时间的一次性密码； **Duo Universal Prompt** - 使用 Duo 身份验证服务提供多因素身份验证。
Name	输入一个名称，该名称将作为账户名称显示给身份验证器应用中的所有 MFA 用户（例如，“Zabbix”）。
Hash function	选择用于生成 TOTP 代码的哈希函数（SHA-1、SHA-256 或 SHA-512）。 如果 MFA 方法类型设置为 “TOTP”，则此参数可用。 请注意，选择 SHA-256 或 SHA-512 可能会显著限制兼容性，因为目前许多应用程序不支持这些函数。
Code length	选择验证码长度（6 或 8）。 如果 MFA 方法类型设置为 “TOTP”，则此参数可用。
API hostname	输入 Duo 身份验证服务提供的 API 主机名。 如果 MFA 方法类型设置为 “Duo Universal Prompt”，则此参数可用。
Client ID	输入 Duo 身份验证服务提供的客户端 ID。 如果 MFA 方法类型设置为 “Duo Universal Prompt”，则此参数可用。
Client secret	输入 Duo 身份验证服务提供的客户端密钥。 如果 MFA 方法类型设置为 “Duo Universal Prompt”，则此参数可用。

配置示例

本节提供了使用基于时间的一次性密码（TOTP）和Duo Universal Prompt配置 MFA 的示例。

TOTP

对于 TOTP，用户必须使用身份验证器应用验证其身份（例如，Google Authenticator 应用）。

1. 转到 Zabbix 中 *Users* > *Authentication* 下的 MFA 设置，并启用多因素身份验证。

2. 按以下配置添加一个新的 MFA 方法：

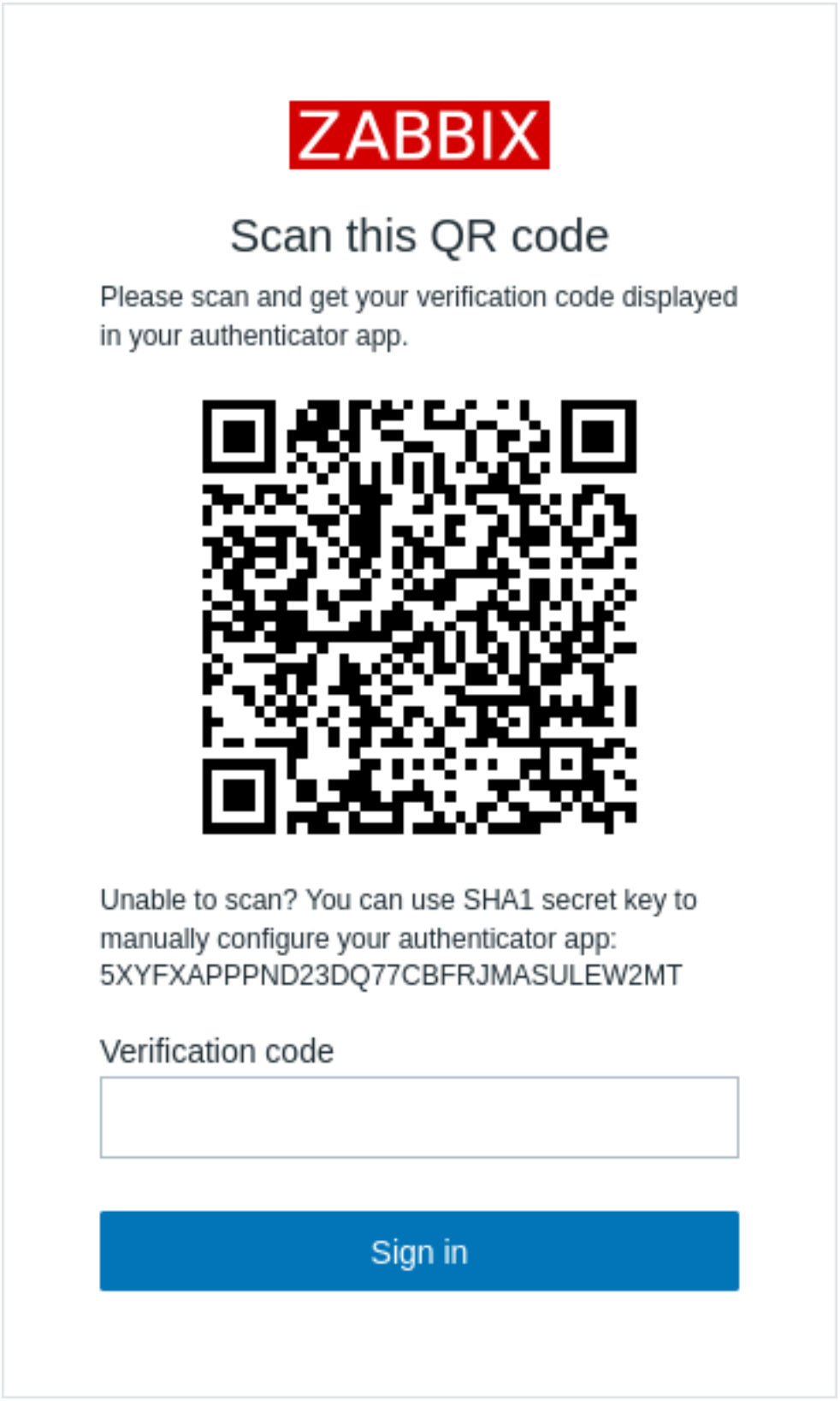
- Type: TOTP
- Name: Zabbix TOTP
- Hash function: SHA-1
- Code length: 6

3. 单击 *Add* 按钮，然后单击 *Update* 按钮。

4. 转到 *Users* > *User groups*，并按以下配置创建一个新的用户组：

- Group name: TOTP group
- Users: Admin
- Multi-factor authentication: Default（如果未将其设为默认值，则为”Zabbix TOTP”）

5. 从 Zabbix 注销，然后使用你的凭据重新登录。登录成功后，系统会提示你注册 MFA，并显示二维码和密钥。



6. 扫描二维码，或将密钥输入 Google Authenticator 应用。该应用会生成一个验证码，你需要输入该验证码以完成登录过程。

7. 之后每次登录时，从 Google Authenticator 应用中获取验证码，并在登录时输入。

Duo Universal Prompt

对于 Duo Universal Prompt，用户必须使用 Duo Mobile 身份验证器应用验证其身份。

Attention:
Duo Universal Prompt MFA 方法需要安装php-curl 扩展、可通过 HTTPS 访问 Zabbix，以及允许向 Duo 服务器发起出站连接的权限。此外，如果你已在web 服务器上启用内容安全策略（CSP），请确保在虚拟主机的配置文件中将”duo.com” 添加到 CSP 指令中。

1. 在 Duo Signup 注册一个免费的 Duo 管理员账户。

2. 打开 Duo 管理面板，转到 *Applications* > *Protect an Application*，搜索 *Web SDK* 应用，然后点击 *Protect*。

3. 记下在 Zabbix 中配置 MFA 方法所需的凭据（Client ID、Client secret、API hostname）。

4. 在 Zabbix 中转到 *Users* > *Authentication* 下的 MFA 设置，并启用多因素认证。

5. 按以下配置添加一个新的 MFA method：

- Type: Duo Universal Prompt
- Name: Zabbix Duo
- API hostname: (use API hostname from Duo)
- Client ID: (use Client ID from Duo)
- Client secret: (use Client secret from Duo)

6. 点击 *Add* 按钮，然后点击 *Update* 按钮。

7. 转到 *Users* > *User groups*，并按以下configuration 创建一个新的用户组：

- Group name: Duo group
- Users: Admin
- Multi-factor authentication: Default (or "Zabbix Duo" if it is not set as default)

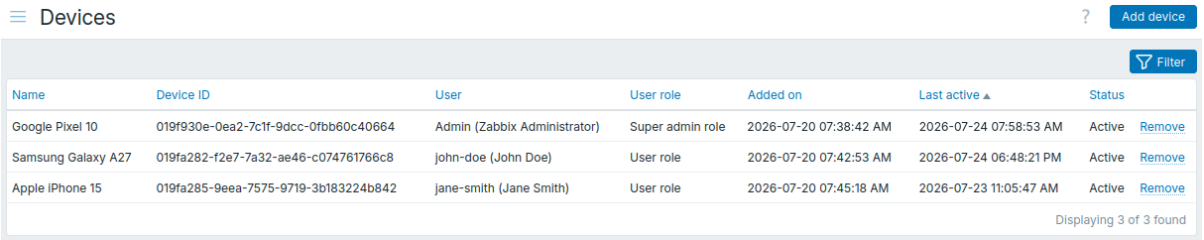
8. 从 Zabbix 注销，然后使用你的凭据重新登录。登录成功后，系统会提示你注册 MFA，并重定向到 Duo。完成 Duo 设置，并使用手机上的 Duo 应用验证你的用户以登录。

9. 对于后续登录，请使用 Duo 应用提供的相应 MFA 方法（例如获取验证码、响应推送通知或使用硬件密钥），并在登录时输入所需信息。

设备

此部分允许超级管理员管理与Zabbix 移动应用配合使用的移动设备。

启用`EnableMobileDevices` 后，此部分才会显示。



添加设备：

1. 单击添加设备。
2. 选择用户，然后单击添加。该用户必须已启用设备访问权限。
3. 此时会显示一个二维码。用户必须在其 Zabbix 移动应用中扫描此二维码。

Note:
为用户添加设备时，Zabbix 还会将推送通知媒介类型添加到其用户媒介中。移除设备时，该用户媒介也会被移除。如果配置了多个推送通知媒介类型，Zabbix 会添加已启用且按字母顺序排在最前面的媒介类型。

添加的设备会显示在列表中，您可以对列表进行筛选。

您也可以移除设备。移除设备后，其访问凭据会被撤销并从 Zabbix 中删除，此后该设备将无法再访问 Zabbix。

列表显示以下详细信息：

列	描述
名称	设备名称。
设备 *ID*	设备 ID。
用户	添加该设备时所指定的用户；如果该用户已被删除，则显示为无法访问的用户。
用户角色	用户角色。
添加时间	添加设备的日期和时间。
最后活动时间	设备最后处于活动状态的日期和时间。

列	描述
状态	设备状态： 新建 - 设备已添加但尚未扫描二维码，需要重新添加。 新建（令牌已过期） - 设备已添加但尚未扫描二维码，二维码中的令牌已过期；需要重新添加设备。 活动 - 设备已添加并已扫描二维码，可以使用 Zabbix 移动应用。 未分配 - 设备已添加，但其用户已被删除，因此无法使用 Zabbix 移动应用。

具有足够权限的用户还可以在用户设置 > 设备中管理自己的设备。

管理

概述

管理主菜单用于 Zabbix 的管理功能。此菜单仅适用于超级管理员 类型的用户。

数据源

概述

管理 > 数据源部分用于指定遥测数据源，即存储遥测数据（跟踪、指标和日志）的数据库。遥测数据用于应用程序性能监控（APM），可帮助您了解请求在应用程序和服务中的流转情况、衡量其性能，并找出操作缓慢或失败的原因。

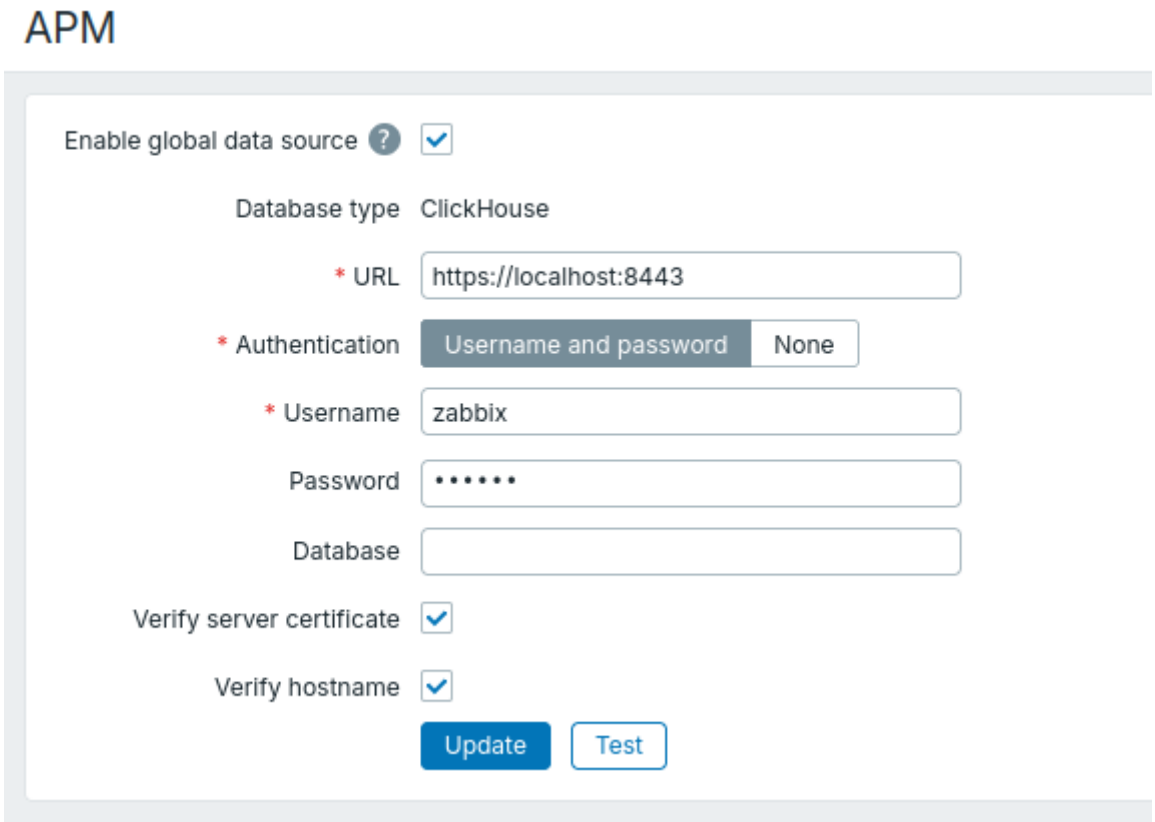
遥测数据源必须支持 OpenTelemetry。

此部分中指定的遥测数据源称为全局数据源，因为 Zabbix 服务器、proxy 和前端共享该数据源。但是，您可以通过在服务器、proxy或前端配置中指定其他遥测数据源，为其中任一组件覆盖全局数据源。

除非在组件配置中指定了其他数据源，否则遥测查询监控项会查询此数据源。

APM

要指定全局数据源，请勾选 启用全局数据源复选框，并填写以下所述参数。保存前，可以点击 测试来检查与此数据源的连接（需要php-curl 扩展）。



参数	描述
数据库类型	数据库类型固定为 *ClickHouse*，无法更改。 另请参阅：要求 和ClickHouse 设置。
URL	输入用于连接数据库的 HTTP 或 HTTPS URL。
身份验证	选择用于登录数据库的身份验证方法：用户名和密码或 无。
用户名	指定用于登录数据库的用户名。 当 身份验证设置为 用户名和密码时，此参数可用。
密码	指定用于登录数据库的密码；如果数据库用户没有密码，则留空。 更改 URL 时，密码会被重置。 当 身份验证设置为 用户名和密码时，此参数可用。
数据库	输入存储遥测数据的数据库名称。 如果留空，Zabbix 将使用 ClickHouse 中为该用户配置的默认数据库。
验证服务器证书	勾选此复选框以验证服务器证书。 如果勾选，则必须在服务器、proxy或前端配置中设置相关 TLS 选项。 当 *URL* 包含 HTTPS URL 时，此参数可用。
验证主机名	勾选此复选框，以验证 *URL* 中的主机名是否与服务器证书中的主机名匹配。 如果勾选，则必须在服务器、proxy或前端配置中设置相关 TLS 选项。 当 *URL* 包含 HTTPS URL 时，此参数可用。

常规

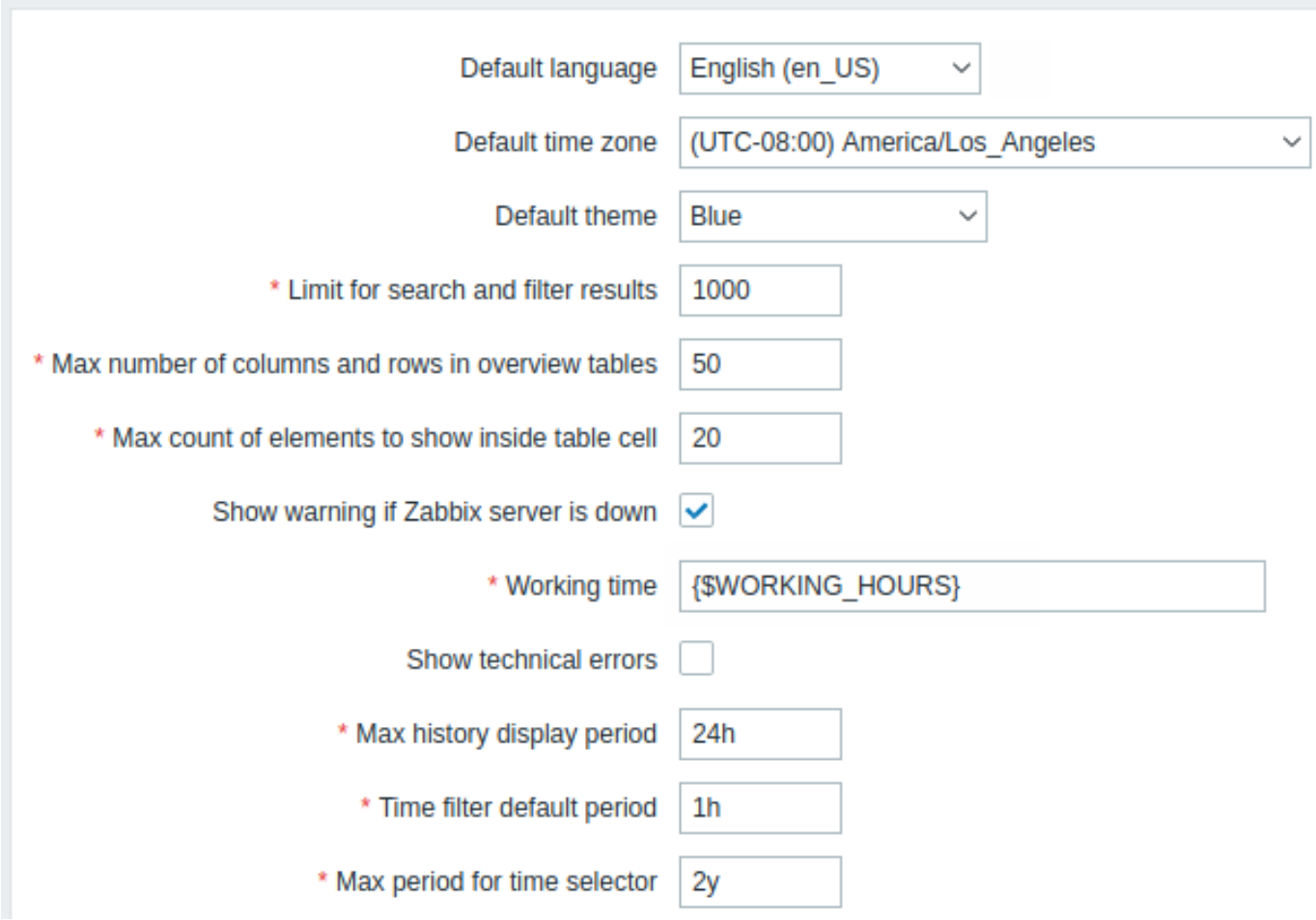
概述

Administration > *General* 部分包含多个子部分，用于设置前端相关的默认值并自定义 Zabbix。

可用子部分列表会在 *Administration* 菜单部分中单击 *General* 后显示。也可以通过使用左上角的标题下拉菜单在各子部分之间切换。

GUI

本节提供对若干与前端相关的默认设置进行自定义的说明。

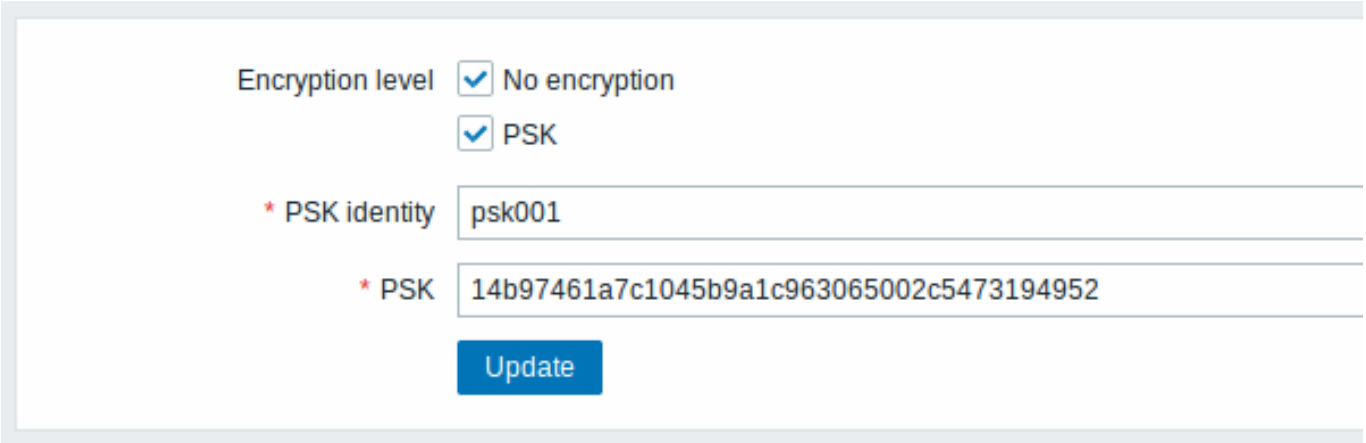


配置参数：

Parameter	Description
Default language	未在个人资料中指定语言的用户和访客用户的默认语言。 更多信息，请参见安装附加前端语言。
Default time zone	未在个人资料中指定时区的用户和访客用户的默认时区。
Default theme	未在个人资料中指定主题的用户和访客用户的默认主题。
Limit for search and filter results	将在 web 界面列表中显示的元素（行）最大数量，例如在 数据采集 > 主机中。 注意：如果设置为例如 “50”，则在所有受影响的前端列表中只会显示前 50 个元素。如果某个列表包含超过 50 个元素，则会在 *“Displaying 1 to 50 of **50+** found”* 中以 “+” 号表示。此外，如果使用了筛选且匹配项仍然超过 50 个，则只会显示前 50 个。 请注意，提高此参数的值可能会导致性能下降，并增加 web 服务器端的内存消耗。
Max number of columns and rows in overview tables	在 触发器概览仪表板小部件中显示的最大列数和行数。此限制同时适用于列和行。如果实际存在的行和/或列多于显示数量，系统会在表格底部显示警告：“并未显示所有结果。请提供更具体的搜索条件。”
Max count of elements to show inside table cell	对于显示在单个表格单元格中的条目，这里配置的显示数量不会超过上限。
Show warning if Zabbix server is down	启用此参数后，如果无法连接到 Zabbix 服务器（可能已停止运行），浏览器窗口中将显示警告消息。即使用户向下滚动页面，该消息仍会保持可见。将鼠标悬停其上时，该消息会暂时隐藏，以显示其下方内容。
Working time	此全局参数定义工作时间。在图表中，工作时间显示为白色背景，非工作时间显示为灰色。 有关时间格式的说明，请参见时间段规范页面。 支持用户宏。
Show technical errors	如果勾选，所有已注册用户都可以看到技术错误（PHP/SQL）。如果未勾选，则这些信息仅对Zabbix 超级管理员以及所属用户组启用了调试模式的用户可见。
Max history display period	在 监控 > 最新数据中显示历史数据、在 监控 > 主机中显示主机web 场景信息，以及在 *Top items* 仪表板小部件中显示历史数据的最长时间范围。 允许范围：24 小时（默认）- 1 周。支持时间后缀，例如 1w（1 周）、36h（36 小时）。
Time filter default period	图表和仪表板默认使用的时间范围。允许范围：1 分钟 - 10 年（默认：1 小时）。 支持时间后缀，例如 10m（10 分钟）、5w（5 周）。 注意：当用户在查看图表时更改时间范围后，该时间范围会作为用户偏好进行保存，并替换全局默认值或之前的用户选择。
Max period for time selector	图表和仪表板可用的最大时间范围。用户将无法查看更早的数据。允许范围：1 年 - 10 年（默认：2 年）。 支持时间后缀，例如 1y（1 年）、365w（365 周）。

自动注册

在本节中，您可以为主动 agent 自动注册配置加密级别。



带星号的参数为必填项。

配置参数：

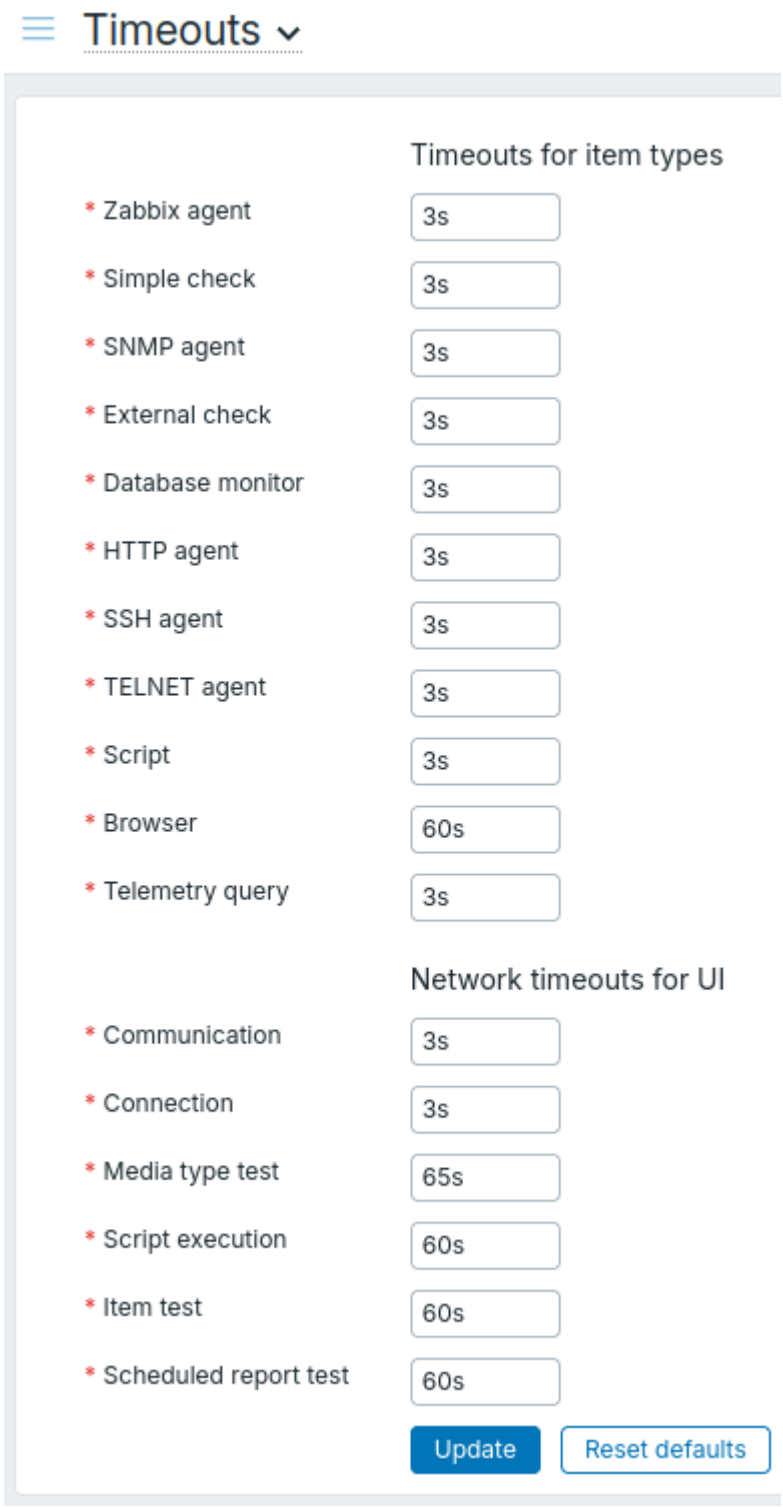
Parameter	Description
加密级别	选择一个或两个加密级别选项： 无加密 - 允许未加密连接 **PSK** - 允许使用预共享密钥的 TLS 加密连接

Parameter	Description
PSK 标识	输入预共享密钥标识字符串。 仅当将'PSK' 选为 加密级别时，此字段才可用。 请勿在 PSK 标识中放入敏感信息，因为它会以未加密方式在网络中传输，以告知接收方应使用哪个 PSK。
PSK	输入预共享密钥（偶数个十六进制字符）。 最大长度：如果 Zabbix 使用 GnuTLS 或 OpenSSL 库，则为 512 个十六进制数字（256 字节 PSK）； 如果 Zabbix 使用 mbed TLS（PolarSSL）库，则为 64 个十六进制数字（32 字节 PSK）。 示例：1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952 仅当将'PSK' 选为 加密级别时，此字段才可用。

另请参见：安全自动注册

超时

在此部分中，可以设置全局监控项类型超时和网络超时。此表单中的所有字段都是必填项。

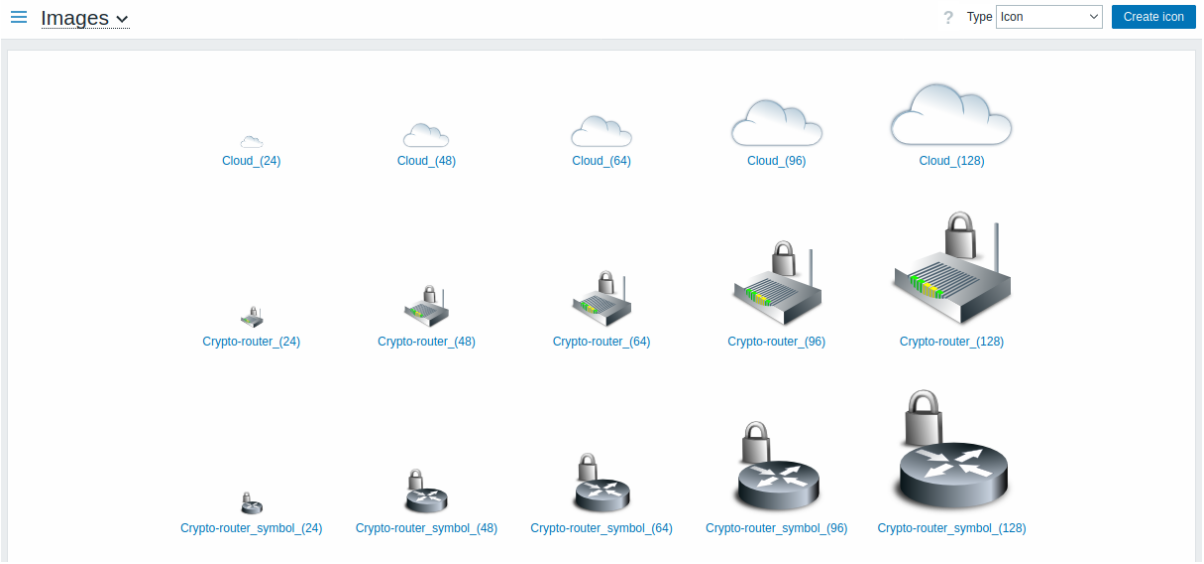


参数		说明
监控项类型超时		Zabbix 在中止检查之前应等待多少秒（基于其类型）。 更多详细信息和限制，请参见监控项超时。 允许范围：1 - 600 秒（默认：3 秒；browser 监控项的默认值：60 秒）。 支持时间后缀，例如 30s、1m，以及用户宏。 请注意，如果配置了 proxy 级别的超时，它们将覆盖这些全局设置。此外，如果配置了特定监控项的超时，则它们会同时覆盖全局和 proxy 设置。
UI 的网络超时		
	通信	在关闭空闲套接字之前等待多少秒（如果之前已建立到 Zabbix 服务器的连接，但前端在此期间无法完成数据读取/发送操作，则连接将被断开）。允许范围：1 - 300 秒（默认：3 秒）。
	连接	在停止尝试连接到 Zabbix 服务器之前等待多少秒。允许范围：1 - 30 秒（默认：3 秒）。

参数		说明
	媒体类型测试	测试媒体类型时等待响应的秒数。允许范围：1 - 300 秒（默认：65 秒）。
	脚本执行	执行脚本时等待响应的秒数。允许范围：1 - 300 秒（默认：60 秒）。 此超时适用于整个脚本执行链，其长度可能各不相同。例如，如果脚本在 agent 上执行，则它是通过服务器（可能还包括 proxy）到 agent 再返回的往返过程。 对于在 agent 上执行的脚本，将适用服务器 或proxy 超时。 对于仅在活动 agent 上执行的脚本，通常需要提高默认的服务器/proxy 超时。服务器/proxy 超时必须高于活动检查刷新频率，否则在活动 agent 甚至收到脚本之前，服务器/proxy 超时就可能已被超过。 另请参见：脚本超时。
	监控项测试	测试监控项时等待返回数据的秒数。允许范围：1 - 600 秒（默认：60 秒）。
	计划报告测试	测试计划报告时等待返回数据的秒数。允许范围：1 - 300 秒（默认：60 秒）。

图像

“图像” 部分显示 Zabbix 中可用的所有图像。
图像存储在数据库中。

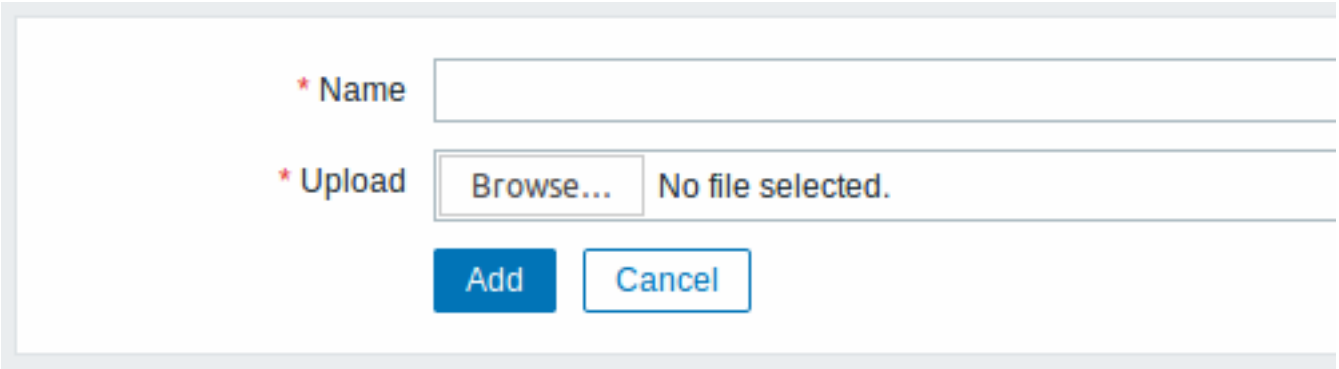


通过 类型下拉列表，可以在图标图像和背景图像之间切换：

- 图标用于显示网络拓扑图元素。
- 背景用于作为网络拓扑图的背景图像。

添加图像

您可以点击右上角的 创建图标或 创建背景按钮来添加自己的图像。



图像属性：

参数	描述
名称	图像的唯一名称。

参数	描述
上传	从本地系统中选择要上传到 Zabbix 的文件（PNG、JPEG、GIF 或 WebP）。 注意，也可以上传其他格式的文件，这些文件将在上传过程中转换为 PNG。图像处理使用 GD 库，因此支持的格式取决于所使用的库版本（Zabbix 要求使用 2.0.28 或更高版本）。

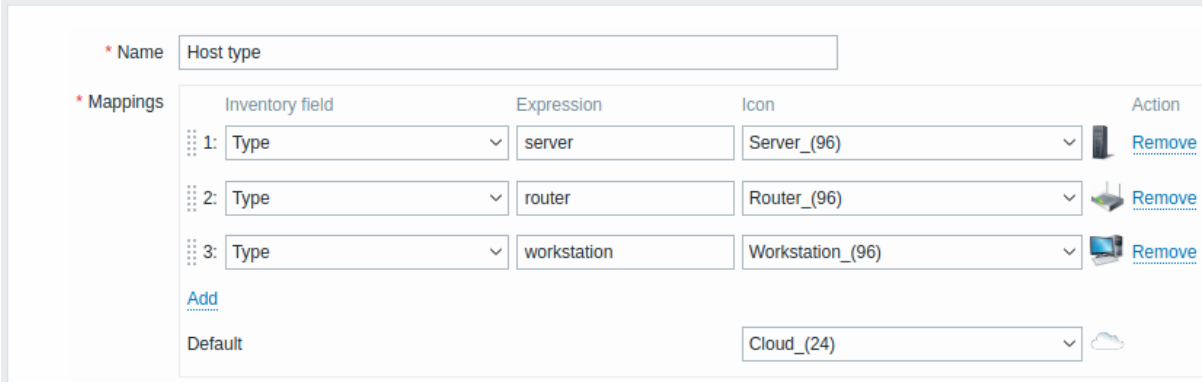
Note:
上传文件的最大大小受 ZBX_MAX_IMAGE_SIZE 的值限制，该值为 1024x1024 字节，即 1 MB。
如果图像大小接近 1 MB，且 MySQL 配置参数 max_allowed_packet 使用默认值 1 MB，则图像可能上传失败。
在这种情况下，请增大 max_allowed_packet 参数。

图标映射

本节允许创建某些主机与某些图标之间的映射。使用主机清单字段信息来创建映射。

随后，这些映射可在网络地图配置中使用，以便自动为匹配的主机分配合适的图标。

要创建新的图标映射，请点击右上角的 *Create icon map*。



配置参数：

Parameter	Description
Name	图标映射的唯一名称。
Mappings	映射列表。映射的顺序决定其优先级。你可以通过拖放在列表中上下移动映射。
Inventory field	将用于查找匹配项的主机清单字段。
Expression	描述匹配条件的正则表达式。
Icon	如果找到与表达式匹配的项，则使用该图标。
Default	要使用的默认图标。

正则表达式

本节允许创建自定义正则表达式，这些表达式可在前端的多个位置使用。有关详细信息，请参见正则表达式 部分。

触发器显示选项

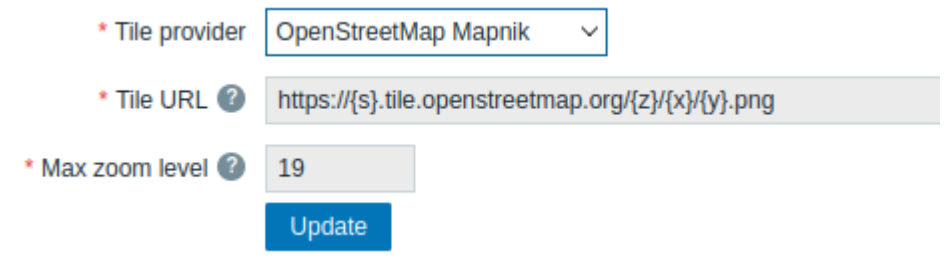
本节允许自定义在前端中显示触发器状态的方式，以及触发器严重性 名称和颜色。



Parameter	Description
Use custom event status colors	勾选此参数可启用已确认/未确认问题的颜色自定义。
Unacknowledged PROBLEM events, *Acknowledged PROBLEM events*, *Unacknowledged RESOLVED events*, *Acknowledged RESOLVED events*	输入新的颜色代码，或单击颜色，从提供的调色板中选择新的颜色。 如果勾选了 *blinking* 复选框，触发器在状态变更后会闪烁一段时间，以便更醒目。
Display OK triggers for	显示 OK 触发器的时间段。允许范围：0 - 24 小时。支持时间后缀，例如 5m、2h、1d。
On status change triggers blink for	触发器闪烁的持续时间。允许范围：0 - 24 小时。支持时间后缀，例如 5m、2h、1d。
Not classified, *Information*, *Warning*, *Average*, *High*, *Disaster*	自定义严重性名称和/或颜色，用于替代系统默认值显示。 输入新的颜色代码，或单击颜色，从提供的调色板中选择新的颜色。 请注意，此处输入的自定义严重性名称将用于所有语言环境。如果需要为某些用户将其翻译为其他语言，请参见自定义触发器严重性 页面。

地理地图

本节允许选择地理地图瓦片服务提供商，并配置 Geomap dashboard widget 的服务提供商设置。
为了通过地理地图提供可视化，Zabbix 使用开源的 JavaScript 交互式地图库 Leaflet。
请注意，Zabbix 无法控制第三方瓦片提供商所提供图像的质量，包括预定义的瓦片提供商。



Parameter	Description
Tile provider	从可用的瓦片服务提供商中选择一个，或选择 *Other* 以添加其他瓦片提供商或自托管瓦片（参见使用自定义瓦片服务提供商）。
Tile URL	用于在地理地图上加载和显示瓦片图层的 URL 模板（最多 2048 个字符）。 仅当 *Tile provider* 设置为 *Other* 时，此字段才可编辑。 支持以下占位符： *{s}* 表示可用子域名之一； *{z}* 表示 URL 中的缩放级别参数； *{x}* 和 *{y}* 表示瓦片坐标； *{r}* 可用于在 URL 中添加"@2x"，以加载 retina 瓦片。 示例：`https://{s}.example.com/{z}/{x}/{y}{r}.png`
Attribution text	要显示在地图上小文本框中的瓦片提供商署名文本。仅当 *Tile provider* 设置为 *Other* 时，此字段才可见。
Max zoom level	地图的最大缩放级别。仅当 *Tile provider* 设置为 *Other* 时，此字段才可编辑。

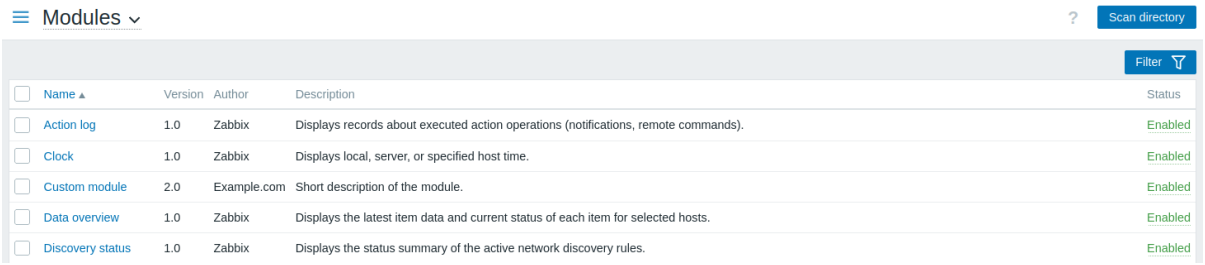
使用自定义瓦片服务提供商

Geomap 小组件能够从自托管或第三方瓦片提供商服务加载栅格瓦片图像。要使用自定义的第三方瓦片提供商服务，或自托管的瓦片文件夹或服务器，请在 *Tile provider* 字段中选择 *Other*，并在 *Tile URL* 字段中使用正确的占位符指定自定义 URL。

模块

本部分允许您管理自定义模块以及内置的前端模块。

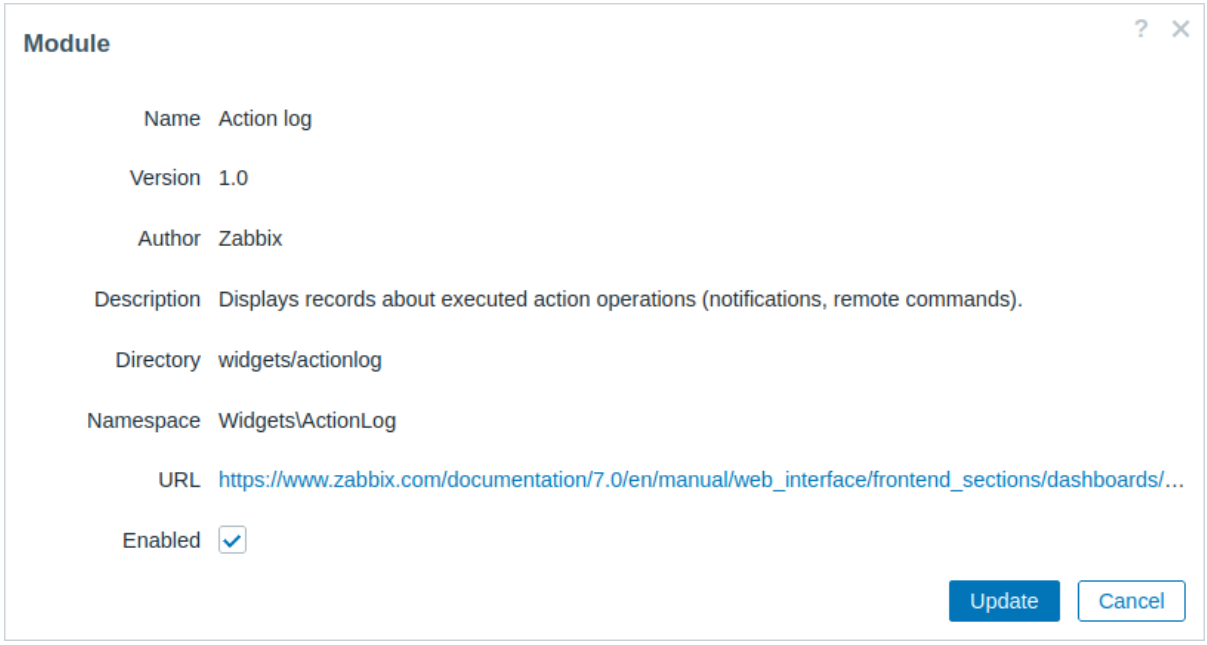
Note:
如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['modules_config_enabled']` 功能标志设置为 `false`，则无法访问本部分。



点击扫描目录，可注册或取消注册任何自定义模块。
已注册的模块将显示在列表中；未注册的模块将从列表中移除。

点击列表中的模块状态，可启用或禁用模块。
您也可以在列表中选择模块，然后点击列表下方的启用/禁用按钮，批量启用或禁用模块。

点击列表中的模块名称，可在弹出窗口中查看其详细信息。

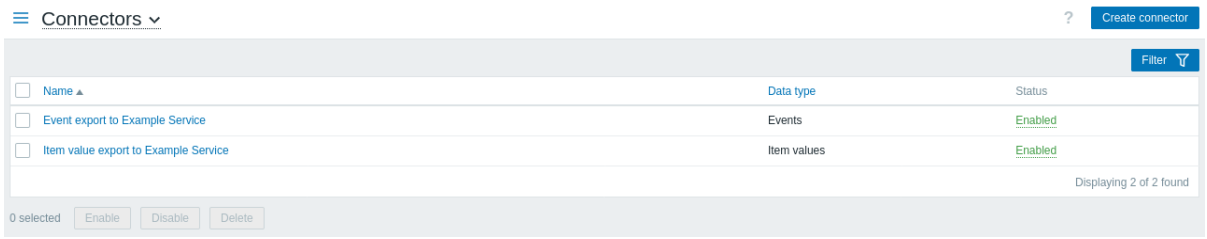


也可以在模块详细信息弹出窗口中更新模块状态；为此，选中或取消选中已启用复选框，然后点击更新。

您可以按名称或状态（已启用/已禁用）筛选模块。

连接器

本部分用于配置通过 HTTP 将 Zabbix 数据流式传输到外部系统的连接器。



点击创建连接器，配置新的连接器。

您可以按名称或状态（已启用/已禁用）筛选连接器。
点击列表中的连接器状态，可启用/禁用连接器。
您还可以通过在列表中选择连接器，然后点击列表下方的启用/禁用按钮，批量启用/禁用连接器。

其他

本节允许配置其他杂项前端参数。



Parameter	Description
Frontend URL	Zabbix Web 界面的 URL（最长 2048 个字符）。 此参数由 Zabbix Web 服务用于与前端通信，应指定此参数以启用计划报表。
Group for discovered hosts	通过网络发现和agent 自动注册发现的主机将自动放入此处所选的主机组中。
Default host inventory mode	主机资产记录的默认模式。除非在主机发现/自动注册期间通过 *Set host inventory mode* 操作覆盖，否则当服务器或前端创建新的主机或主机原型时，将遵循此模式。
User group for database down message	用于发送数据库不可用告警消息的用户组。 Zabbix 服务器依赖后端数据库的可用性。没有数据库它将无法工作。如果数据库不可用，Zabbix 可以通知所选用户。通知将使用已启用的用户媒介条目发送到此处设置的用户组。通知通过以下媒介类型发送（启用时）：电子邮件；短信；自定义告警脚本。即使配置并启用了 webhook 媒介条目，也不会触发通知。 Zabbix 服务器不会停止；它会等待数据库恢复后再继续处理。 通知包含以下内容： `[MySQL|PostgreSQL] database <DB Name> [on <DB Host>:<DB Port>] is not available: <error message depending on the type of DBMS (database)>` 如果 <DB Host> 定义为空值，则不会将其添加到消息中；如果 <DB Port> 为默认值（“0”），则也不会添加。告警管理器（Zabbix 服务器中的一个特殊进程）每 10 秒尝试重新建立与数据库的连接。如果数据库仍然不可用，告警管理器会重复发送告警，但频率不会高于每 15 分钟一次。
Log unmatched SNMP traps	如果未找到对应的 SNMP 接口，则记录SNMP trap。

授权

参数	说明
登录尝试次数	在登录被阻止之前，允许的不成功登录尝试次数。
登录阻止间隔	当超过 登录尝试次数限制时，禁止登录的时间段。允许范围：30 - 3600 秒。支持时间后缀，例如 90s、5m、1h。

密钥的存储

Parameter	Description
Vault provider	选择用于存储user macro 值的密钥管理软件 - *HashiCorp Vault*（默认）或 *CyberArk Vault*。
Resolve secret vault macros by	按以下方式解析密钥 vault 宏： **Zabbix** 服务器 - 由 Zabbix 服务器检索密钥，并在需要时转发给 proxy（默认）； **Zabbix** 服务器和 **proxy** - 由 Zabbix 服务器和 proxy 共同检索密钥，使它们能够独立解析宏。

另请参见：密钥的存储。

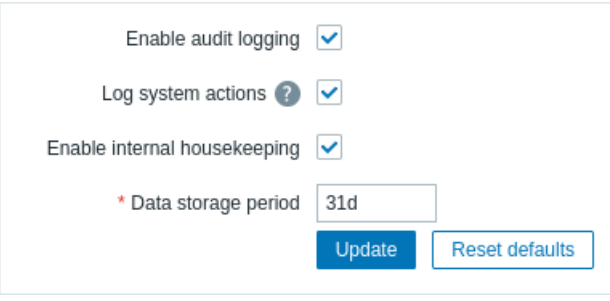
安全

参数	说明
验证 *URI* 方案	取消选中此复选框以禁用 URI 方案验证（默认启用）。 如果选中，则可以指定允许的 URI 方案的逗号分隔列表（默认值：http、https、ftp、file、mailto、tel、ssh）。适用于前端中使用 URI 的所有字段（例如，地图元素 URL）。
使用 *X-Frame-Options HTTP* 标头	取消选中此复选框以禁用 HTTP X-Frame-options 标头（不推荐）。 如果已标记，则可以指定 HTTP X-Frame-options 标头的值。支持的值： **SAMEORIGIN** (默认) 或 **'self'** (必须使用单引号) - 页面只能显示在与页面本身同源 (same origin) 的框架中; **DENY** 或 **'none'** (必须使用单引号) - 无论哪个网站试图这样做，页面都不能在框架中显示; 一串以空格分隔的主机名字符串; 将 **'self'** (必须使用单引号) 添加到列表中，可以将页面显示在与页面自身同源的框架中。 请注意，使用不带单引号的 **'self'** 或 **'none'** 将导致它们被视为主机名。
使用 *iframe* 沙盒	取消选中此复选框以禁止将检索到的 URL 内容放入沙盒（不推荐）。 如果已选中，则可以指定 iframe 沙盒例外; 未指定的限制仍将应用。如果此字段为空，则所有沙盒属性限制均适用。 详情请参阅 `sandbox` 属性说明.

审计日志

概述

本节允许配置审计日志设置。



可用参数如下：

参数	描述
启用审计日志记录	启用（默认）或禁用审计日志记录。
记录系统操作	启用（默认）或禁用对由服务器（System user）执行的低级别发现、网络发现和自动注册活动的审计日志记录。
启用内部 *housekeeping*	启用（默认）或禁用审计日志记录的内部 housekeeping。

参数	描述
数据存储周期	审计日志记录在被 housekeeper 删除之前应保留的天数。 如果启用了 housekeeping，则此项为必填。 默认值：31 天。

审计日志压缩参数位于 *Administration > Housekeeping* 部分中的 *History, trends and audit log compression* 区块内；当使用TimescaleDB 时，该区块会显示出来。

清理维护

概述

Administration > Housekeeping 部分可让你配置 Zabbix 内部 housekeeping，它会自动从数据库中删除过期或由用户删除的数据。这可以防止数据库无限增长，从而避免占用不必要的磁盘空间并降低数据库性能。

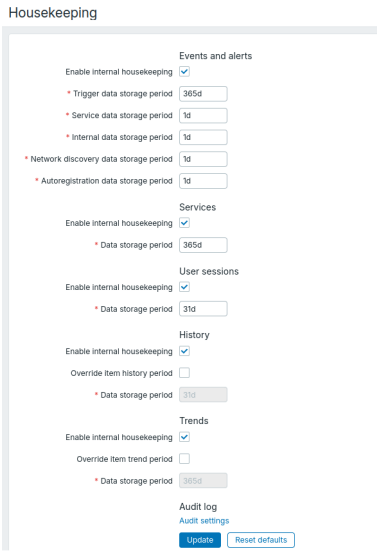
可以为以下类型的数据启用并配置 housekeeping：

- 事件和告警，来自触发器、服务、内部数据、网络发现和自动注册
- 服务
- 用户会话
- 历史数据和趋势，适用于监控项

Note:
审计日志 housekeeping 需单独配置。

配置

可用的参数如下：



Parameter	Description
Enable internal housekeeping	启用或禁用内部 housekeeping（默认已启用）。 启用后，housekeeping 过程会自动从数据库中删除超过 *Data storage period* 的数据。

Parameter	Description
Data storage period	指定在由 housekeeper 删除数据之前保留数据的时长。 如果启用了内部 housekeeping，则此项为必填。 范围：1 天（history 为 1 小时）- 25 年；或"0"。支持Time suffixes（例如 1d、1w）。 对于 *Events and alerts*，数据存储周期会分别为触发器、服务、内部数据、网络发现和自动注册单独设置。 housekeeper 只会删除那些未与问题关联的事件。例如，如果某个问题/恢复事件与未解决的问题相关联，即使它早于 *Data storage period*，也不会被删除。当 housekeeper 删除过期实体时，会先删除问题，再删除事件。 对于 *History* 和 *Trends*，即使禁用了内部 housekeeping，数据存储周期也会决定数据在 *Monitoring > Latest data* 中保持可见的时长。 请注意，当监控项或触发器被删除时，其问题及相关事件也会一并删除。
Override item history period	如果启用，将覆盖监控项配置 中指定的 history/trend 存储周期，改为使用 *Data storage period* 设置（已启用 *Do not store* 选项的监控项除外）。 即使禁用了内部 housekeeping 并使用外部 housekeeper，也可以使用此选项。
Override item trend period	

如果使用 **ClickHouse**，则会显示 *ClickHouse data storage period* 部分。其中显示每种监控项值类型的数据存储周期。如需更改该周期，请参见Configuring ClickHouse data storage period。请注意，housekeeper 不会从 ClickHouse 中删除数据。

如果使用 **Elasticsearch**，则会显示 *Elasticsearch data storage period* 部分。其中显示保存在 Elasticsearch 中的监控项值类型，数据存储周期显示为 *Unknown*。如需控制数据保留时长，请参见 Index Lifecycle Management。请注意，housekeeper 不会从 Elasticsearch 中删除数据。

如果使用 **TimescaleDB**，则会显示 *History, trends and audit log compression* 部分。

> **Note:**
> 对于TimescaleDB，请启用 *Override item history period*、*Override item trend period* 和 *Enable internal housekeeping*，以便 history 和 trends 充分受益于自动分区。如果这些选项被禁用，保存在 history 和 trends 表中的数据仍会被分区，但 housekeeper 不会删除过期分区，并且会出现配置警告。启用删除过期分区后，Zabbix 服务器和前端将不再跟踪已删除的监控项，并且当过期分区被删除时，这些监控项的 history 将被清除。

默认情况下，housekeeper 每小时自动执行一次。您可以通过设置 Zabbix 服务器配置参数`HousekeepingFrequency` 来更改执行频率。

Reset defaults 按钮可让您还原所做的任何更改。

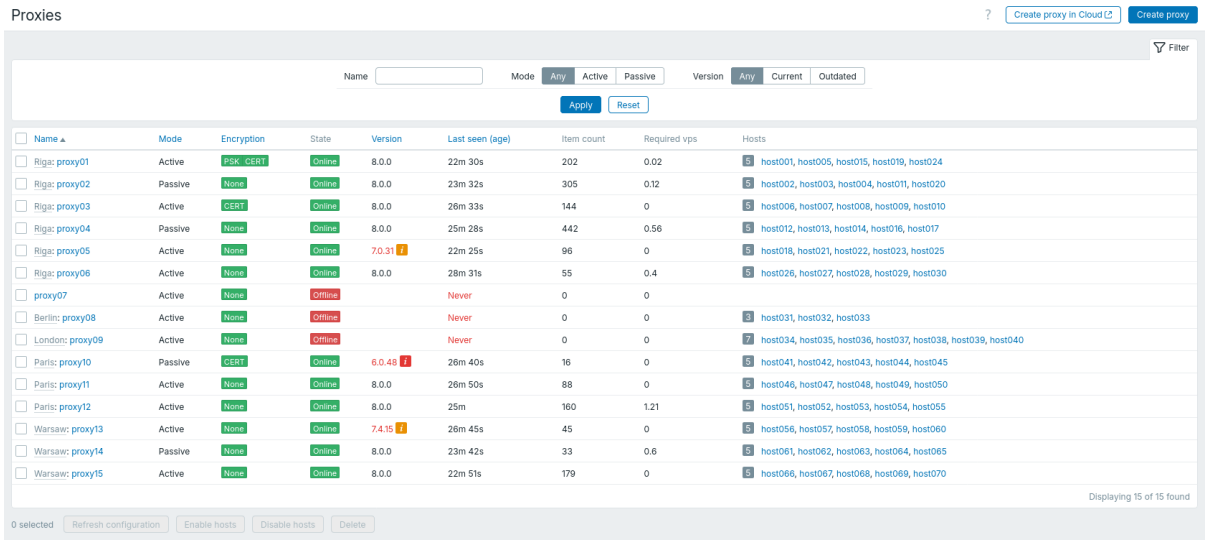
proxy

概述

在 Zabbix 前端的 管理 > *Proxy* 部分，可以配置用于分布式监控的 proxy。

proxy

此处显示现有 proxy 及其详细信息列表。



显示的数据：

列	说明
名称	proxy 的名称。 单击 proxy 名称可打开proxy 配置表单。 如果 proxy 属于 proxy 组，则 proxy 名称前会显示组名称，并以灰色链接表示。单击组名称可打开proxy 组配置表单。
模式	proxy 模式 - 主动或被动。
加密	proxy 连接的加密状态： 无 - 不加密； **PSK** - 使用预共享密钥； 证书 - 使用证书。
状态	proxy 的状态： 未知 - 在 Zabbix 服务器停止运行时创建了 proxy，或服务器尚未更新状态； 在线 - proxy 在故障转移周期内曾与 Zabbix 服务器通信（被动 proxy 响应服务器请求；主动 proxy 发送请求）； 离线 - proxy 在故障转移周期内未与 Zabbix 服务器通信。
版本	proxy 版本（三位版本号）。如果 proxy 已过时或不受支持，版本号会突出显示（红色），并显示信息状态图标（黄色或红色）。将鼠标悬停在图标上可查看详细信息。
最后发现时间（时长）	服务器最后一次发现 proxy 的时间。
监控项数量	分配给该 proxy 的已启用主机上已启用监控项的数量。
所需 *vps*	所需的 proxy 性能（每秒需要采集的值数量）。
主机	分配给该 proxy 的已启用主机数量，以及由该 proxy 监控的主机列表。 单击主机名称可打开主机配置表单。

要配置新的 proxy，请单击右上角的创建 *proxy* 按钮。要在 Zabbix Cloud 中部署新的 proxy，请单击旁边的在 *Cloud* 中创建 *proxy* 按钮。这将在新的浏览器标签页中打开 Zabbix Cloud，您可以在其中创建 proxy 节点。创建 proxy 节点后，使用创建 *proxy* 按钮配置 proxy。有关更多信息，请参阅设置 Zabbix Cloud proxy。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

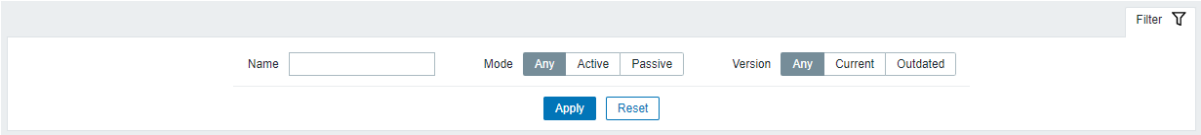
- 刷新配置 - 刷新 proxy 的配置；
- 启用主机 - 将由 proxy 监控的主机状态更改为 已监控；
- 禁用主机 - 将由 proxy 监控的主机状态更改为 未监控；
- 删除 - 删除这些 proxy。

要使用这些选项，请先勾选相应 proxy 前面的复选框，然后单击所需的按钮。

使用筛选器

您可以使用筛选器，仅显示您感兴趣的 proxy。为提高搜索性能，数据会在宏未解析的情况下进行搜索。

proxy 列表上方提供了 筛选器链接。单击该链接后，将显示筛选器，您可以按名称、模式和版本筛选 proxy。请注意，筛选选项 已过时会同时显示已过时（部分支持）和不受支持的 proxy。



proxy 组

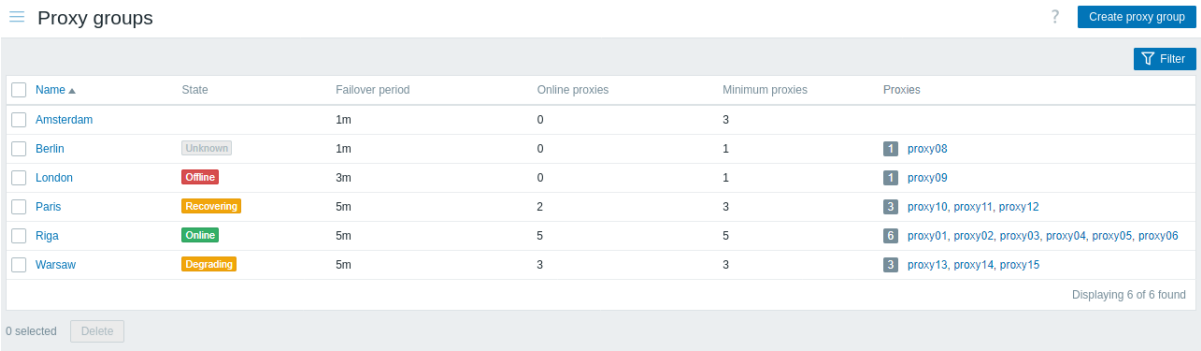
概述

在 *Administration* > *Proxy groups* 中，可以配置 proxy groups。

proxy groups 用于proxy 负载均衡，实现主机在各个 proxy 之间的自动分配，以及 proxy 之间的高可用性。

Proxy 组

系统会显示现有 proxy 组及其详细信息列表。



显示的数据：

列	描述
名称	proxy 组的名称。点击 proxy 组名称可打开proxy 组配置表单。
状态	proxy 组的状态（组中至少包含一个 proxy 时显示）： 未知 - Zabbix 服务器关闭时创建了该组，或服务器尚未更新状态； 在线 - 在故障转移周期内，至少有最少数量的 proxy 与 Zabbix 服务器通信； 降级 - 由于在线 proxy 数量低于最少 *proxy* 数量阈值，该组即将离线； 离线 - 在故障转移周期内，与 Zabbix 服务器通信的 proxy 数量少于最少数量； 恢复中 - 由于在线 proxy 数量达到最少 *proxy* 数量阈值，该组即将恢复在线状态。 可以使用zabbix[proxy group,<name>,state] 和zabbix[proxy group,discovery] 监控 proxy 组状态。
故障转移周期	proxy 组中的 proxy 必须与 Zabbix 服务器通信的周期，只有在此期间通信的 proxy 才会被视为在线。
在线 *proxy*	在线 proxy 的数量（低于组的最小数量时以红色显示）。
最少 *proxy* 数量	保持 proxy 组在线所需的最少在线 proxy数量。
Proxy	组中 proxy 的数量以及组中 proxy 的列表，并提供指向 proxy 配置表单的链接。 列出的 proxy 最大数量受表格单元格中显示的元素最大数量值限制。

要配置新的 proxy 组，请点击右上角的创建 *proxy* 组按钮。

批量编辑选项

列表下方的按钮提供一些批量编辑选项：

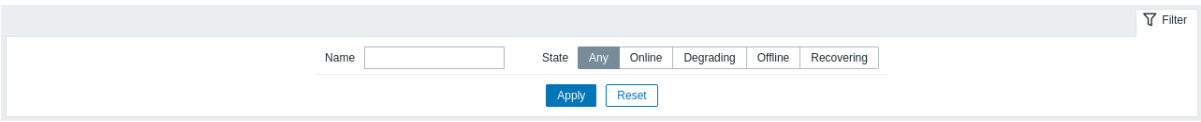
- 删除 - 删除这些 proxy 组。

要使用这些选项，请先勾选相应 proxy 组前面的复选框，然后点击所需的按钮。

使用过滤器

您可以使用过滤器仅显示您感兴趣的 proxy 组。为了获得更好的搜索性能，数据搜索时不会解析宏。

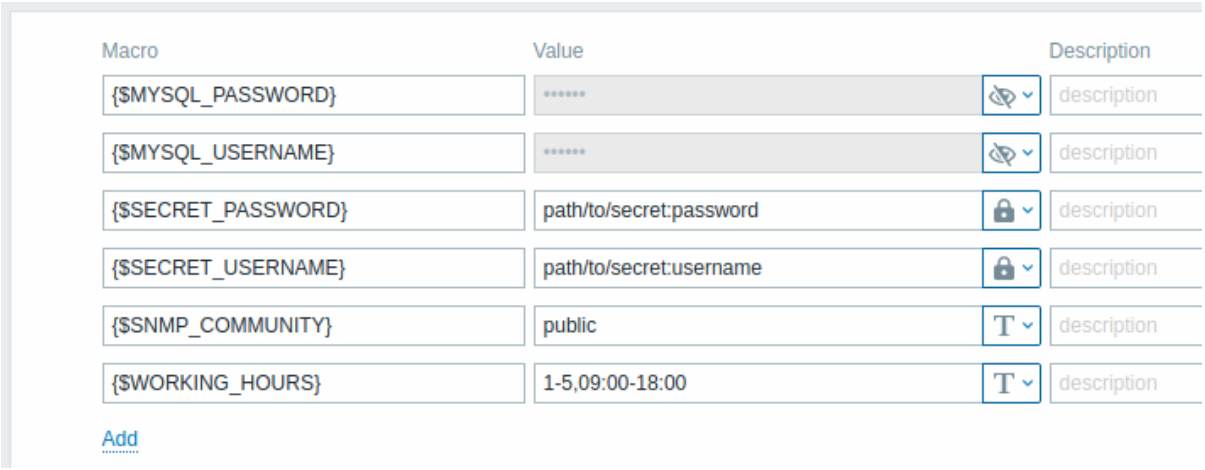
Filter 链接位于 proxy 组列表上方。单击该链接后，会显示一个过滤器，您可以按名称和状态筛选 proxy 组。



宏

概述

此部分允许您以名称-值对的形式定义系统范围的用户宏。请注意，宏值可以保存为纯文本、机密文本或 Vault 机密。同时也支持添加描述。



队列

概述

在管理 > 队列部分，会显示等待更新的监控项。

理想情况下，打开此部分时应全部显示为 “绿色”，这意味着队列中没有监控项。如果所有监控项都能及时更新，则不会有监控项等待更新。但是，由于服务器性能不足，某些监控项可能会延迟，相关信息会显示在此部分中。有关详细信息，请参阅队列部分。

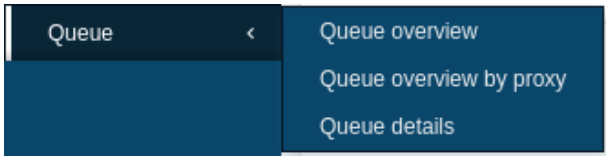
Note:
Zabbix proxy 不知道维护期；有关详细信息，请参阅维护期间的队列计算。

Note:
只有在 Zabbix 服务器运行时，队列才可用。如果监控项接口因连接问题或 agent 未正常工作而变得不可用，则该监控项不会计入队列。

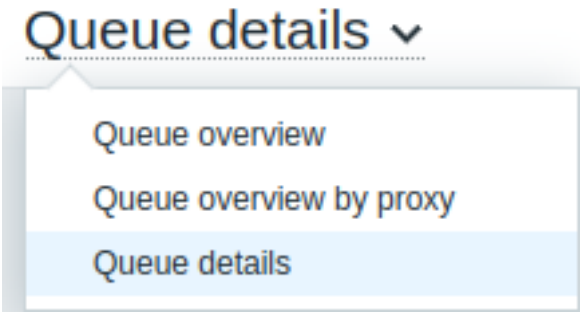
管理 > 队列部分包含以下页面：

- 队列概览 - 按监控项类型显示队列；
- 按 proxy 查看队列概览 - 按 proxy 显示队列；
- 队列详细信息 - 显示延迟监控项列表。

点击管理菜单部分中的队列后，会显示可用页面列表。也可以使用左上角的标题下拉菜单在页面之间切换。



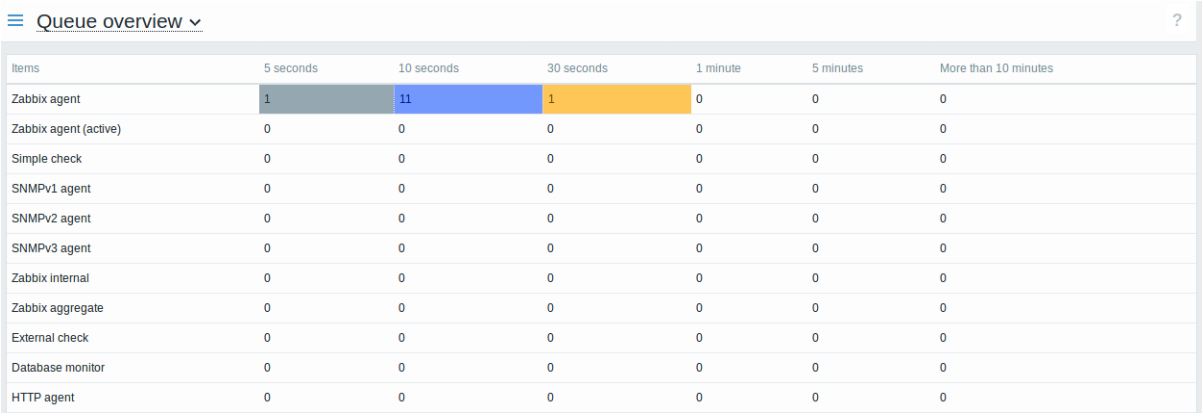
三级菜单。



标题下拉菜单。

监控项类型概览

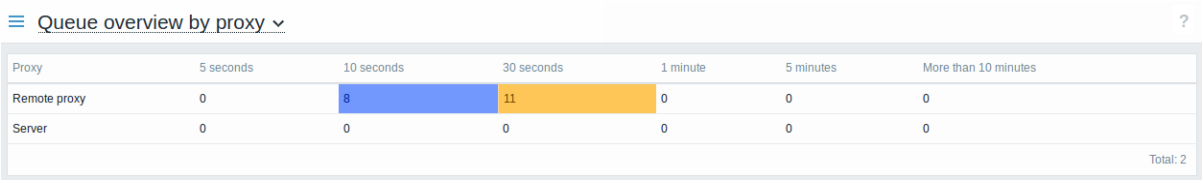
在此屏幕中，可以轻松确定问题是否与一种或多种监控项类型有关。



每行包含一个监控项类型。每列显示等待监控项的数量 - 分别等待 5-10 秒/10-30 秒/30-60 秒/1-5 分钟/5-10 分钟或超过 10 分钟。

proxy 概览

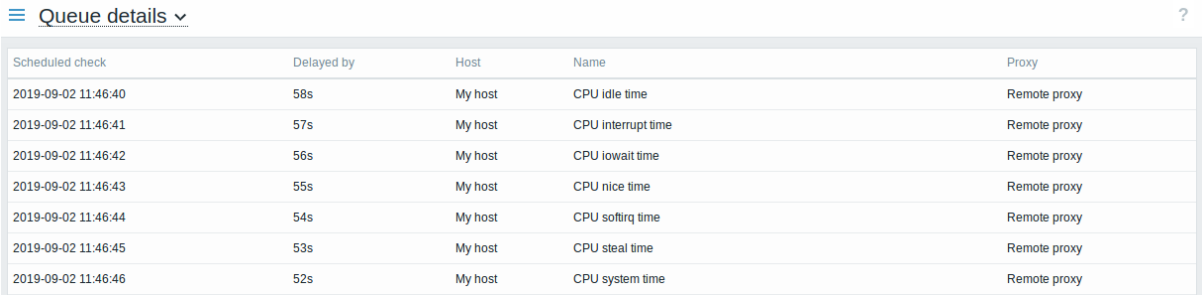
在此屏幕中，很容易找到问题是否与 proxy、server 之一有关。



每行包含一个 proxy，server 位于列表的最后。每列显示等待监控项的数量 - 分别等待 5-10 秒/10-30 秒/30-60 秒/1-5 分钟/5-10 分钟或超过 10 分钟。

等待监控项清单

在此屏幕中，将列出每个等待的监控项。

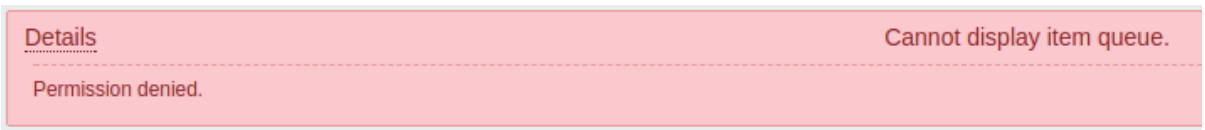


显示数据：

列	说明
定期检查	显示预期检查时间。
延迟	显示延迟的时长。
主机	显示监控项所属的主机名。
名称	显示等待监控项的名称。
Proxy 代理	如果主机通过 Proxy 代理监控，则会显示 Proxy 代理的名称

可能的错误消息

您可能会遇到未显示任何数据并出现以下错误消息的情况：



此时的错误消息如下：

```
Cannot display item queue. Permission denied
```

当前端配置文件中的 PHP 配置参数 $ZBX_SERVER，或 $ZBX_SERVER 和 $ZBX_SERVER_PORT 同时，指向使用不同数据库的现有 Zabbix 服务器时，就会发生这种情况。

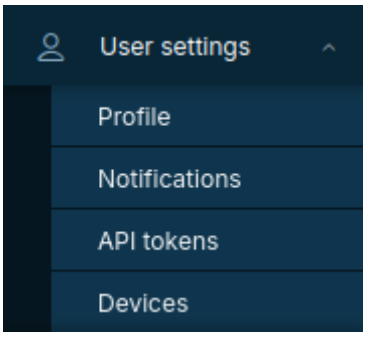
用户设置

概述

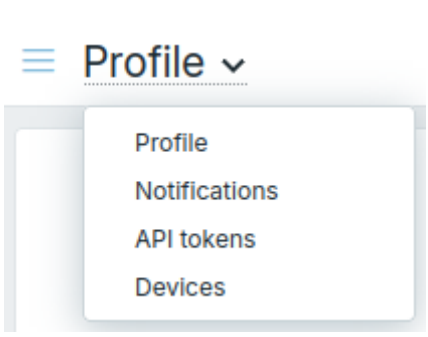
根据您的用户角色，用户设置部分可能包含以下页面：

- 个人资料或用户个人资料 - 用于自定义某些 Zabbix 前端功能；
- 通知 - 用于自定义当前用户的通知；
- *API* 令牌 - 用于管理分配给当前用户的 API 令牌；
- 设备 - 用于管理属于当前用户的移动设备。

点击 Zabbix 菜单底部附近的 用户图标后，将显示可用页面列表（*guest* 用户不可用）。也可以使用左上角的标题下拉菜单在页面之间切换。



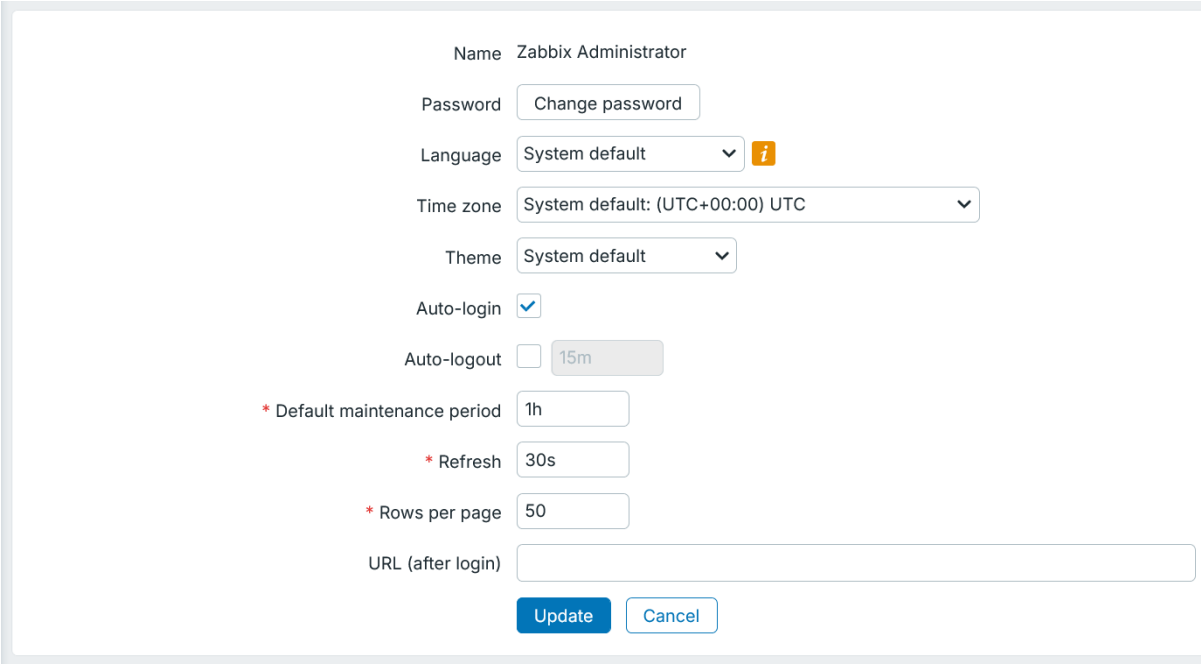
三级菜单。



标题下拉菜单。

用户配置文件

用户配置文件部分提供了设置自定义界面语言、配色主题、列表中显示的行数等选项。
此处所做的更改仅应用于当前用户。



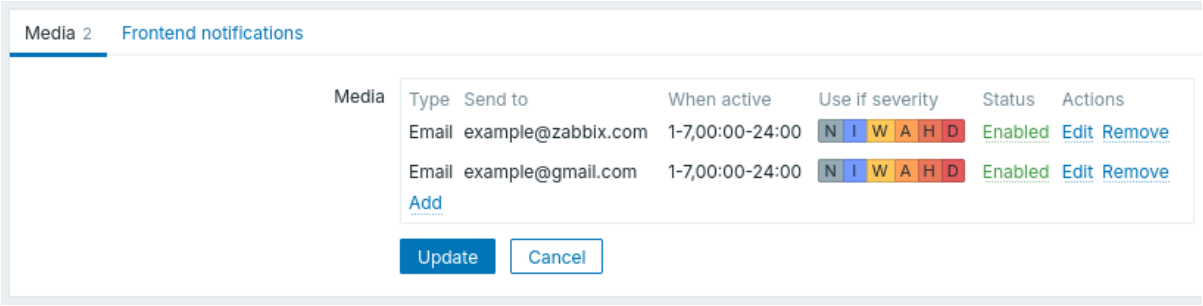
参数	描述
名称	显示用户的姓名。 如果未设置姓名，则显示用户名。
密码	单击更改密码按钮以打开三个字段：旧密码、新密码、新密码（再次输入）。 密码更改成功后，用户将从所有活动会话中退出。 请注意，仅使用 Zabbix 内部身份验证的用户可以更改密码。
语言	选择所需的界面语言，或选择系统默认以使用默认系统设置。 选择英语 *(en_US)* 还会在前端中启用美国时间/日期格式。 有关更多信息，请参阅安装其他前端语言。

参数	描述
时区	选择时区，以在用户级别覆盖全局时区，或选择系统默认以使用全局时区设置。
主题	为配置文件选择特定的配色主题： 系统默认 - 使用默认系统设置； 蓝色 - 标准蓝色主题； 蓝色（经典） - 使用旧版字体的标准蓝色主题； 深色 - 备用深色主题； 深蓝色 - 深蓝色主题； 深色（经典） - 使用旧版字体的备用深色主题； 高对比度浅色 - 高对比度浅色主题； 高对比度深色 - 高对比度深色主题。
自动登录	选中此复选框后，Zabbix 将记住您的登录信息，并在 30 天内自动登录。 使用已启用的记住 *30* 天选项登录时： - 自动注销计时器会重置（会话将持续到 30 天期限结束）。 - 在 30 天内后续访问时会启用自动登录。 不使用记住 *30* 天选项登录时： - 自动登录参数会被清除。 - 自动注销将照常遵循配置的超时时间。 此功能使用浏览器 Cookie。
自动注销	选中此复选框后，经过设定的秒数将自动注销（最少 90 秒，最多 1 天）。 支持时间后缀，例如：90s、5m、2h、1d。 请注意，在以下情况下此选项将不起作用： * 当监控菜单页面在后台刷新信息时。如果保持打开的页面会按特定时间间隔刷新数据（仪表板、图形、最新数据等），会话生命周期也会相应延长，从而禁用自动注销功能。 * 如果登录时选中了记住我 *30* 天选项；在这种情况下，自动注销计时器会被覆盖，使会话在整个期限内保持活动状态。 * 当通过 SSO 身份提供商（IdP）执行身份验证时，自动注销只会结束 Zabbix 会话。与 IdP 的 SSO 会话仍处于活动状态，因此用户可能无需输入密码即可重新登录。 自动注销也可以接受 “0”，这意味着更新配置文件设置后将禁用自动注销功能。
默认维护周期	设置临时维护的默认维护周期。
刷新	设置监控菜单页面刷新信息的频率（最少 0 秒，最多 1 小时）。 支持时间后缀，例如：30s、90s、1m、1h。
每页行数	设置列表中每页显示的行数。行数越少（需要显示的记录越少），加载速度越快。
URL（登录后）	设置登录后显示的特定 URL。它可以不是默认的仪表板，例如可以设置为监控 > 触发器的 URL。

通知

通知部分提供了用于自定义当前用户通知的选项。

介质选项卡允许您为用户指定介质详情，例如要使用的介质类型和地址，以及使用它们发送通知的时间。



如果介质类型已被禁用：

- 名称后会显示一个黄色信息图标；
- 状态列中会显示 “Disabled”。

Note:
用户是否有权限为自己更改介质详情，可根据其用户角色授予/撤销（请参见 *Create and edit own media* 选项）。超级管理员用户是否有权限为其他用户更改介质详情，也可根据其用户角色授予/撤销（请参见 *Create and edit user media* 选项）

请注意，对于已预配用户：

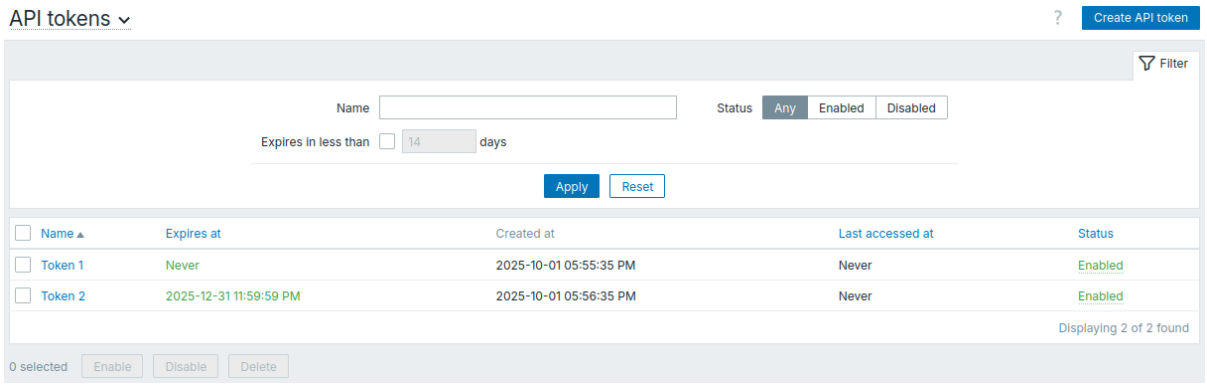
- 已预配的用户介质不能被删除；
- 已预配的用户介质可以被禁用/启用；

- 此类已预配用户介质中的 *When active*、*Use if severity* 和 *Enabled* 字段可以手动编辑；
- 可以为已预配用户手动添加额外的用户介质（例如，额外的电子邮件地址）；
- 手动添加的用户介质可以被删除。

前端通知选项卡允许您设置全局通知。

API 令牌

API 令牌部分允许您查看分配给该用户的令牌、编辑令牌详细信息以及创建新令牌。只有在用户角色 设置中允许 管理 *API* 令牌操作时，用户才能访问此部分。



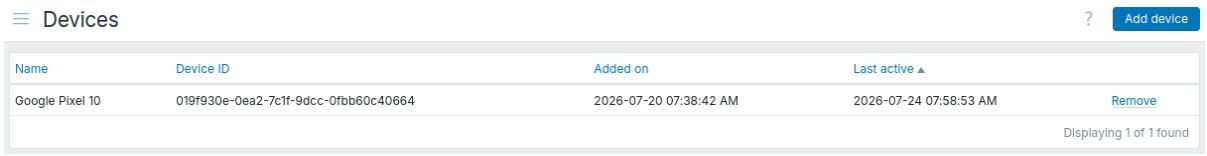
您可以按名称、过期日期或状态（启用/禁用）筛选 API 令牌。单击列表中的令牌状态可快速启用/禁用该令牌。您还可以在列表中选择多个令牌，然后单击列表下方的 启用/禁用按钮，一次性启用/禁用多个令牌。

Attention:
用户无法查看在 Zabbix 中分配给自己的 *Auth token* 值。*Auth token* 值只会显示一次——在创建令牌后立即显示。如果令牌已丢失，则必须重新生成。

设备

设备部分用于添加或移除要与Zabbix 移动应用配合使用的移动设备。

启用`EnableMobileDevices` 后，才会显示此部分。



添加设备：

1. 单击添加设备。
2. 在 Zabbix 移动应用中扫描二维码。

Note:
添加设备时，Zabbix 还会将推送通知媒体类型添加到您的用户媒体中。移除设备时，该用户媒体也会被移除。如果配置了多个推送通知媒体类型，Zabbix 会添加已启用且按字母顺序排列最靠前的媒体类型。

已添加的设备会显示在列表中。列表显示设备的名称、ID，以及设备添加时间和上次活动的日期和时间。

您也可以移除设备。移除设备后，其访问凭据将被撤销并从 Zabbix 中删除，此后该设备将无法再访问 Zabbix。

超级管理员还可以在用户 > 设备中管理其他用户的设备。

全局通知

概述

全局通知是一种在 Zabbix 前端屏幕上显示当前正在发生的问题的方式。

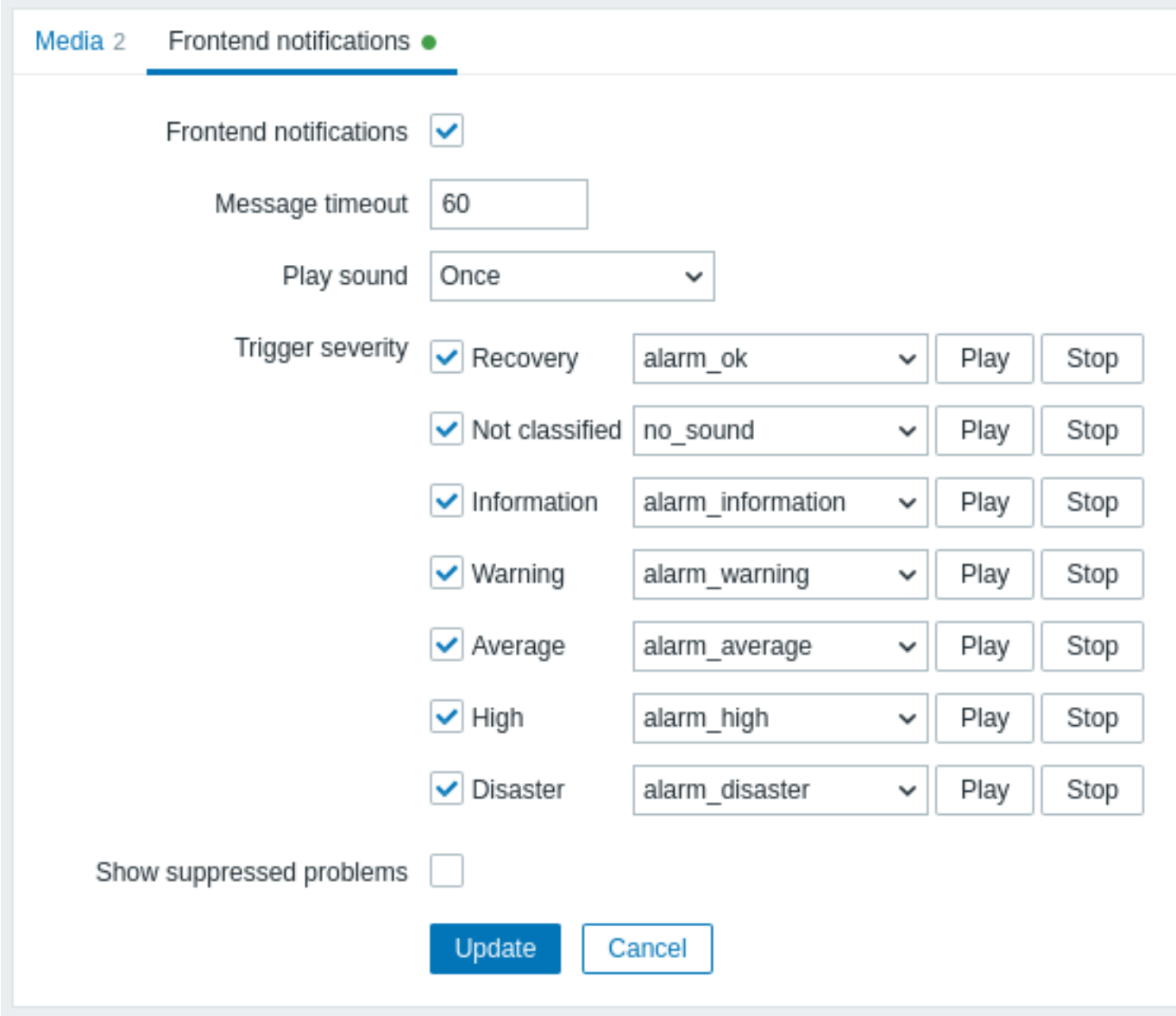
如果没有全局通知，在 问题或 仪表板之外的位置工作时，您将不会收到有关当前问题的任何信息。全局通知可确保无论您处于 Zabbix 前端中的何处，均显示此信息。

全局通知包括显示全局消息 和播放声音。

Attention:
默认情况下，最近的浏览器版本可能会禁用声音的自动播放。在这种情况下，您需要手动启用此设置。

配置

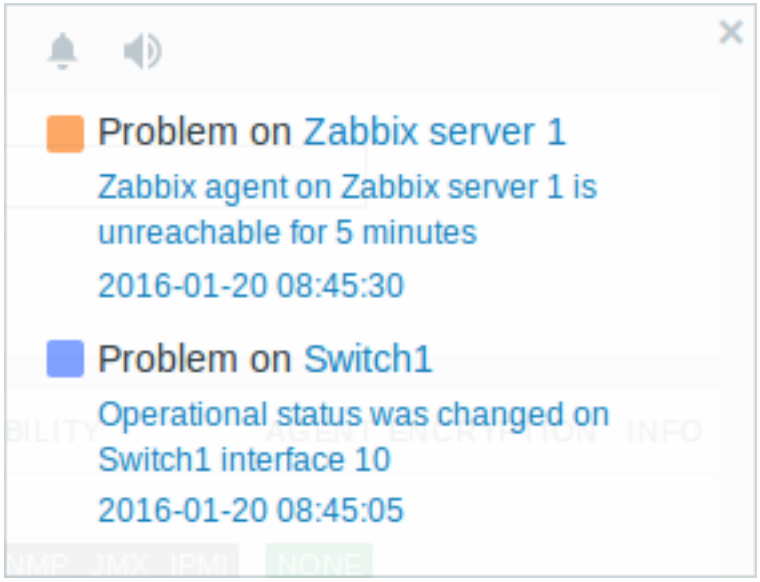
可以在Notifications 部分的 前端通知选项卡中为每个用户启用全局通知。



Parameter	Description
前端通知	勾选此复选框以启用全局通知。
消息超时	设置消息显示的持续时间。默认情况下，消息会在屏幕上保留 60 秒。 支持时间后缀，例如：30s、5m、2h、1d。
播放声音	设置声音播放的持续时间。 一次 - 声音完整播放一次； **10** 秒 - 声音会重复播放 10 秒； 消息超时 - 在消息可见期间重复播放声音。
触发器严重性	设置将激活全局通知和声音的触发器严重性。您还可以为不同严重性选择相应的声音。 如果未标记任何严重性，则不会显示任何消息。 此外，仅会为已标记的严重性显示恢复消息。例如，如果标记了 *Recovery* 和 *Disaster*，则会为 *Disaster* 严重性的触发器问题和恢复显示全局通知。
显示被抑制的问题	勾选此复选框，以显示由于主机维护而原本会被抑制（不显示）的问题通知。

显示全局消息

当消息到达时，它们会显示在右侧的浮动窗口中。该窗口可以通过拖动窗口标题自由重新定位。



对于本节，有几个控件可用：

- 延后警报按钮使当前活动的警报声音静音；
- 静音/取消静音按钮在播放和不播放警报声之间切换。

浏览器中的声音

概述

声音用于全局通知。

要在 Zabbix 前端播放声音，必须在用户配置文件的 前端通知选项卡中启用 前端通知，并选中所有触发器严重性。此外，应在全局通知弹出窗口中启用声音。

如果由于任何原因导致无法在设备上播放音频，则全局通知弹出窗口中的 按钮将永久保持” 静音” 状态，并将鼠标悬停在 按钮上时显示消息” 此设备无法支持通知音频”。

声音（包括默认音频剪辑）仅支持 MP3 格式。

Zabbix 前端的声音已经在 Linux 上最新的 Firefox 和 Opera 浏览器，以及 Windows 上的 Chrome，Firefox，Microsoft Edge 和 Opera 浏览器中测试成功。

Attention:
在最新的浏览器版本中，声音的自动播放可能会被禁用（默认情况下）。在这种情况下，您需要手动启用此设置。

全局搜索

可以在 Zabbix 前端中搜索主机、主机组、模板和模板组。

搜索输入框位于菜单中 Zabbix 标志下方。可通过按 *Enter* 键或单击 搜索图标来开始搜索。



如果某个主机的名称任意部分包含输入的字符串，则会显示一个下拉列表，列出所有此类主机（匹配部分以橙色高亮显示）。如果某个主机的可见名称与作为搜索字符串输入的技术名称匹配，下拉列表中也会列出该主机；但该匹配主机不会进行高亮显示。

可搜索的属性

可以通过以下属性搜索主机：

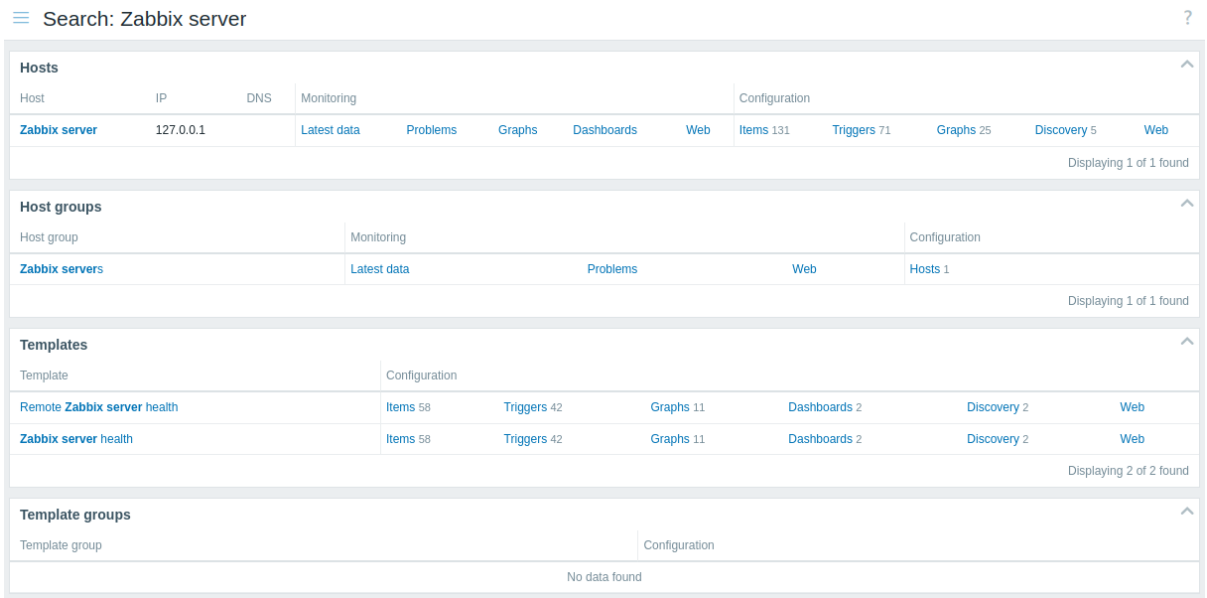
- 主机名
- 可见的名字
- IP 地址
- DNS 名称

可以按不可见名称或可见名称搜索模板。如果您按与（模板/主机的）可见名称不同的名称进行搜索，则在搜索结果中，它会显示在括号中可见名称的下方。

可以按不可见名称搜索主机组。指定父主机组会隐式选择所有嵌套主机组。

搜索结果

搜索结果由主机、主机组、模板和模板组这四个独立区块组成。



可以折叠/展开每个单独的区块。每个区块底部会显示条目计数，例如：*Displaying 13 of 13 found*。如果没有条目，则不会显示条目计数。单个区块中显示的条目总数限制为 100。

每个条目都提供指向监控和配置数据的链接。请参见链接的完整列表。

对于所有配置数据（如监控项、触发器、图形），找到的实体数量会以灰色数字显示在实体名称旁边。注意，如果实体数量为零，则不会显示数字。

已启用的主机会以蓝色显示，已禁用的主机会以红色显示。

可用链接

对于每个条目，以下链接可用：

- 主机
  - 监控
    - 最新数据
    - 问题
    - 图表
    - 主机仪表盘
    - 网络场景
  - 配置
    - 监控项
    - 触发器
    - 图表
    - 自动发现规则
    - 网络场景
- 主机组
  - 监控
    - 最新数据
    - 问题
    - 网络场景
  - 配置
    - 主机
- 模板
  - 配置
    - 监控项
    - 触发器
    - 图表
    - 模板仪表盘
    - 自动发现规则
    - 网络场景
- 模板组
  - 配置
    - 模板

前端维护模式

概述

可以临时禁用 Zabbix 前端以限制访问。这对于保护 Zabbix 数据库免受用户发起的更改、保持其完整性非常有用。

当 Zabbix 前端处于维护模式时，您可以安全地停止数据库并执行维护任务。

来自已定义 IP 地址的用户在维护模式期间仍可正常与前端交互。

配置

要启用维护模式，请打开 `maintenance.inc.php` 文件（该文件位于 Web 服务器上 Zabbix HTML 文档目录的 `/conf` 中），并取消注释以下行：

```
// 维护模式。
define('ZBX_DENY_GUI_ACCESS', 1);

// 允许连接到前端的 IP 地址数组（可选）。
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// 警告屏幕上显示的消息（可选）。
$ZBX_GUI_ACCESS_MESSAGE = '我们将升级 MySQL 数据库，直到 15:00。敬请关注……';
```

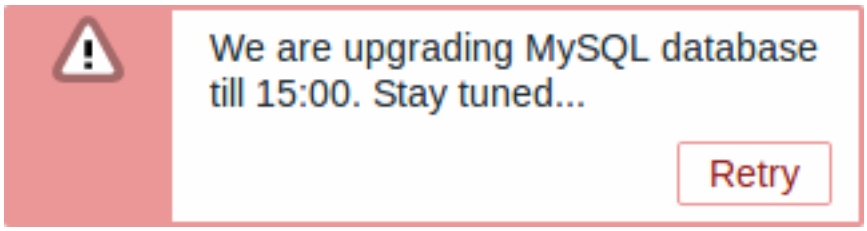
Note:
在大多数情况下，maintenance.inc.php 文件位于 Web 服务器上 Zabbix HTML 文档目录的 /conf 中。
但是，某些操作系统和 Web 服务器可能使用不同的位置。
例如，以下系统中的位置为：
- SUSE 和 RedHat：/etc/zabbix/web/maintenance.inc.php。
- 基于 Debian 的系统：/usr/share/zabbix/conf/。

另请参阅复制 PHP 文件。

Parameter	Details
ZBX_DENY_GUI_ACCESS	如果使用任意值定义此参数，将启用维护模式。 要禁用维护模式，请注释掉或删除该参数。
ZBX_GUI_ACCESS_IP_RANGE	允许连接到前端的 IP 地址数组（可选）。 例如： array('192.168.1.1', '192.168.1.2')
ZBX_GUI_ACCESS_MESSAGE	关于通知用户维护状态的消息（可选）。 如果未定义，将使用默认消息 *'Zabbix is under maintenance'*。

显示

当处于维护模式时，用户尝试访问 Zabbix 前端将看到以下界面。该界面每 30 秒刷新一次，以便在维护结束后无需用户干预即可恢复到正常状态。



在 *ZBX_GUI_ACCESS_IP_RANGE* 中定义的 IP 地址将能够像往常一样访问前端。

页面参数

概述

大多数 Zabbix Web 界面页面都支持各种 HTTP GET 参数，用于控制将显示的内容。
这些参数可以通过在 URL 后指定 parameter=value 对来传递，URL 与参数之间用问号 (?) 分隔，参数之间用与号 (&) 分隔。

监控 > 问题

支持以下参数：

Parameter	Description	Example
show	筛选选项 *Show*。 可选值： 1 - 最近的问题； 2 - 全部； 3 - 处于问题状态。	show=1
name	筛选选项 *Problem*：自由格式字符串。	name=Zabbix agent

Parameter	Description	Example
severities	筛选选项 *Severity*：按所选严重性组成的数组，格式为 `severities[*]=*`（将 `*` 替换为严重性级别）。 可选值： 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 严重； 5 - 灾难。	severities[3]=3
inventory	筛选选项 *Host inventory*：库存字段数组 `[field],[value]`	inventory[0][field]=type&invento
evaltype	筛选选项 *Tags*：标签评估方法。 可选值： 0 - 与/或； 2 - 或。	evaltype=0
tags	筛选选项 *Tags*：已定义标签数组 `[tag], [operator], [value]` 可选operator 值： 0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。	tags[0][tag]=target&tags[0][ope
show_tags	筛选选项 *Show tags*。 可选值： 0 - 不显示； 1 - 显示一个； 2 - 显示两个； 3 - 显示三个。	show_tags=3
tag_name_format	筛选选项 *Tag name*。 可选值： 0 - 全名； 1 - 缩短名称； 2 - 不显示。	tag_name_format=1
tag_priority	筛选选项 *Tag display priority*：以逗号分隔的标签显示优先级字符串	tag_priority=customer, target
show_suppressed	筛选选项 *Show suppressed problems*。 可选值： 1 - 显示； 2 - 不显示。	show_suppressed=1
acknowledgement_status	筛选选项 *Acknowledgement status*。 可选值： 0 - 全部； 1 - 未确认； 2 - 已确认。	acknowledgement_status=0
acknowledged_by_me	筛选选项 *By me*；仅在 `acknowledgement_status=2` 时支持。 可选值： 0 - 禁用； 1 - 启用。	acknowledged_by_me=1
compact_view	筛选选项 *Compact view*。 可选值： 0 - 禁用； 1 - 启用。	compact_view=1

Parameter	Description	Example
highlight_row	筛选选项 *Highlight whole row*。 可选值： 0 - 禁用； 1 - 启用。	highlight_row=1
filter_name	筛选属性选项 *Name*：自由格式字符串	filter_name=Databases
filter_show_counter	筛选属性选项 *Show number of records*。 可选值： 0 - 禁用； 1 - 启用。	filter_show_counter=1
filter_custom_time	筛选属性选项 *Set custom time period*。 可选值： 0 - 禁用； 1 - 启用。	filter_custom_time=1
sort	排序列。 可选值： clock - 按 *Time* 列排序； host - 按 *Host* 列排序； severity - 按 *Severity* 列排序； name - 按 *Problem* 列排序。	sort=clock
sortorder	结果排序顺序。 可选值： DESC - 降序； ASC - 升序。	sortorder=DESC
age_state	筛选选项 *Age less than*；仅在 `show=3` 时支持。 可选值： 0 - 禁用 age 参数； 1 - 启用 age 参数。	age_state=1
age	筛选选项 *Age less than*：整数，天数；仅在 `age_state=1` 且 `show=3` 时支持。	age=7
groupids	筛选选项 *Host groups*：主机组 ID 数组。	groupids[]=4
hostids	筛选选项 *Hosts*：主机 ID 数组	hostids[]=10084
triggerids	筛选选项 *Triggers*：触发器 ID 数组	triggerids[]=22382
show_timeline	筛选选项 *Show timeline*；不支持 `compact_view=1`。 可选值： 0 - 不显示； 1 - 显示。	show_timeline=1
details	筛选选项 *Show details*。 可选值： 0 - 不显示； 1 - 显示。	details=1
from	日期范围开始，可为相对时间（例如，now-1m）；仅在 `filter_custom_time=1` 时支持	from=now-2h
to	日期范围结束，可为相对时间（例如，now-1m）；仅在 `filter_custom_time=1` 时支持	to=now

另请参见：Problems 页面筛选选项。

信息亭模式

支持的前端页面可以使用 URL 参数激活信息亭模式。例如，在仪表板中：

- `/zabbix.php?action=dashboard.view&kiosk=1` - 激活信息亭模式
- `/zabbix.php?action=dashboard.view&kiosk=0` - 激活普通模式

幻灯片放映

可以在仪表板中启用幻灯片放映：

- `/zabbix.php?action=dashboard.view&slideshow=on` - 启用幻灯片放映

定义

概述

虽然可以使用前端本身配置前端中的许多东西，但目前只能通过编辑定义文件来进行一些自定义。

该文件是 `defines.inc.php`，位于 Zabbix HTML 文档目录的 /include 文件夹中。

参数

此文件中可能对用户有用的参数：

- ZBX_MIN_PERIOD

图形的最小周期，单位为秒。默认值为 1 分钟。

- GRAPH_YAXIS_SIDE_DEFAULT

简单图形中 Y 轴的默认位置，以及添加监控项到自定义图形时下拉框的默认值。可选值：0 - 左侧，1 - 右侧。

默认值：0

- ZBX_SESSION_NAME

用作 Zabbix 前端会话 cookie 名称的字符串。

默认值：zbx_sessionid

- ZBX_DATA_CACHE_TTL

用于使Vault 响应 数据缓存失效的 TTL 超时时间，单位为秒。设为 0 可禁用 Vault 响应缓存。

默认值：60

- SUBFILTER_VALUES_PER_GROUP

每个组的子过滤器值数量（例如，在最新数据 子过滤器中）。

默认值：1000

- ZBX_MAX_WIDGET_LINES

要显示的小组件行数上限。

默认值：1000

其他前端语言

概述　要在 Zabbix 前端中使用除英语以外的其他语言，必须在 web 服务器上安装相应的区域设置。此外，翻译功能还需要 PHP gettext 扩展。

安装 **locales**　要列出所有已安装的语言，请运行：

```
locale -a
```

如果未列出某些所需语言，请打开 `/etc/locale.gen` 文件，并取消注释所需的 locales。由于 Zabbix 使用 UTF-8 编码，您需要选择带有 UTF-8 字符集的 locales。

现在运行：

```
locale-gen
```

重启 web 服务器。请注意，如果 web 服务器安装并使用了 PHP-FPM，也需要与 web 服务器一起重启。

现在应该已经安装好 locales 了。可能还需要在浏览器中使用 `Ctrl + F5` 重新加载 Zabbix 前端页面，新的语言才会显示出来。

安装 **Zabbix**　如果直接从 Zabbix git repository 安装 Zabbix，则应手动生成翻译文件。要生成翻译文件，请运行：

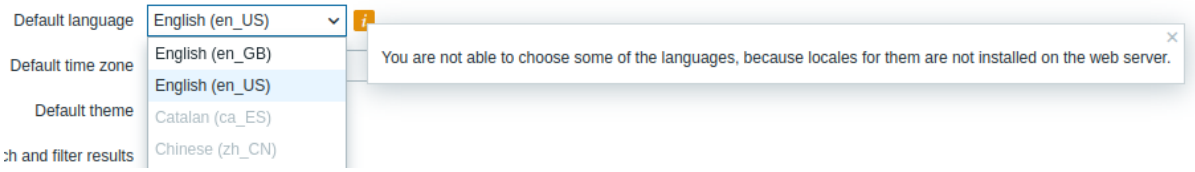
```
make gettext
locale/make_mo.sh
```

如果是从软件包或源代码 tar.gz 文件安装 Zabbix，则不需要执行此步骤。

选择语言　在 Zabbix web 界面中，有几种方式可以选择语言：

- 在安装 web 界面时 - 在前端安装向导 中选择。所选语言将被设置为系统默认语言。
- 安装完成后，可以在 *Administration* > *General* > *GUI* 中更改系统默认语言。
- 特定用户的语言可以在用户配置文件 中更改。

如果某种语言对应的 locale 未安装在该机器上，那么该语言在 Zabbix 语言选择器中将显示为灰色。如果至少有一个 locale 缺失，语言选择器旁边会显示一个橙色图标。点击该图标后，将显示以下消息："由于 web 服务器上未安装这些语言对应的 locale，您无法选择其中某些语言。"



创建你自己的主题

概述

默认情况下，Zabbix 提供了多个预定义的前端主题。要创建自定义主题，请按照此处提供的分步流程进行操作。

第 1 步：创建您的主题

要创建您自己的主题：

1. 从头创建一个 CSS 文件，或者从 assets/styles/ 目录中复制一个现有主题（例如 blue-theme.css、dark-theme.css 或其他主题）并进行修改。
2. 将您的主题（例如 custom-theme.css）保存到 assets/styles/ 目录中。

步骤 2：在主题列表中启用主题

要使自定义主题在前端中可用，必须将其添加到 APP::getThemes() 方法返回的主题列表中。具体方法是在 APP 类中重写 ZBase::getThemes() 方法，如下所示：

1. 打开 include/classes/core/APP.php 文件并找到 APP 类：

```
class APP extends ZBase {

}
```

2. 在类中，在最后一个右花括号 } 之前添加以下方法：

```
public static function getThemes() {
          return array_merge(parent::getThemes(), [
              'custom-theme' => _('Custom theme')
          ]);
}
```

其中：

- custom-theme - 主题的内部标识符，必须与 CSS 文件名（不含 .css 扩展名）一致；
- Custom theme - 在前端中显示的名称。

要添加多个主题，请将它们列在数组中，并使用逗号分隔每个条目。最后一个条目后的逗号可选。

例如：

```
      public static function getThemes() {
          return array_merge(parent::getThemes(), [
              'custom-theme' => _('Custom theme'),
              'anothertheme' => _('Another theme'),
              'onemoretheme' => _('One more theme')
```

```
            ]);
        }
```

步骤 3：更改图形颜色

要更改图形颜色，必须在数据库表 graph_theme 中为新主题添加一条记录。

向 MySQL/MariaDB 数据库添加记录的示例：

```
    mysql -u zabbix -p
    # Enter password:
```

```
mysql> USE zabbix;
mysql> INSERT INTO graph_theme (
    graphthemeid,
    theme,
    backgroundcolor,
    graphcolor,
    gridcolor,
    maingridcolor,
    gridbordercolor,
    textcolor,
    highlightcolor,
    leftpercentilecolor,
    rightpercentilecolor,
    nonworktimecolor,
    colorpalette
)
SELECT
    MAX(graphthemeid) + 1,
    'custom-theme',
    'FFFFFF',
    'FFFFFF',
    'CCD5D9',
    'ACBBC2',
    'ACBBC2',
    '1F2C33',
    'CC745E',
    '5ECCAB',
    'CC745E',
    'EBEBEB',
    '329F7E,C2583D,346D91,B26E44,CC6C91,7A6DC2,C4AA56,7C2731,BF75B8,73A350,B04833,633A60,879CCC,7FAD6C,324
FROM graph_theme;
```

步骤 4：使用新主题

在 Zabbix 前端中，您可以将此主题设置为默认主题，也可以在用户配置文件中选择该主题。

调试模式

概述

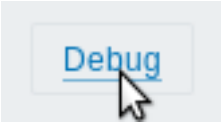
调试模式可用于诊断前端页面的性能问题。

配置

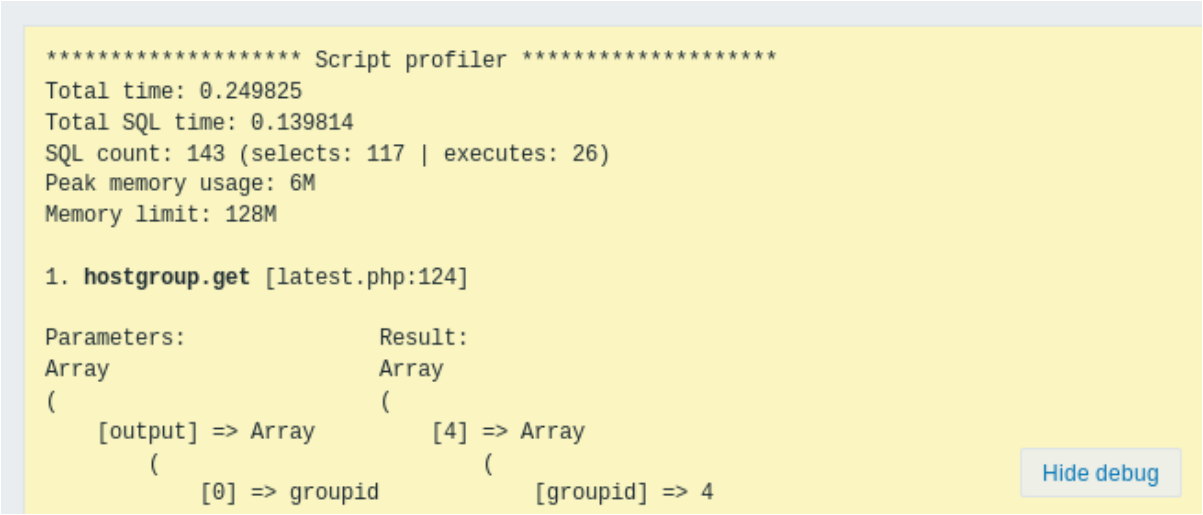
可以为属于某个用户组的单个用户启用调试模式：

- 在配置用户组 时；
- 在查看已配置的用户组 时。

当为用户组启用 调试模式时，其用户会在浏览器窗口右下角看到一个 *Debug* 按钮：



单击 *Debug* 按钮后，会在页面内容下方打开一个新窗口，其中包含该页面的 SQL 统计信息，以及 API 调用和各个 SQL 语句列表：



如果页面存在性能问题，可以使用此窗口来查找问题的根本原因。

Warning:
启用 调试模式会对前端性能产生负面影响。

Zabbix 使用的 **Cookie**

概述

本页提供 Zabbix 使用的 cookie 列表。

名称	说明	值	过期时间/Max-Age	HttpOnly[1]	Secure[2]
ZBX_SESSION_NAME	Zabbix 前端会话数据，以 base64 编码的 JSON 形式存储		会话（在浏览会话结束时过期）	+	+（仅当 Web 服务器上启用了 HTTPS 时）
tab	活动标签页编号；此 cookie 仅用于具有多个标签页的页面（例如主机、触发器或 动作配置页面），并且会在用户从主标签页导航到其他标签页（如 标签或 依赖关系标签页）时创建。 0 用于主标签页。	示例：1	会话（在浏览会话结束时过期）	-	-
browserwarning	是否忽略关于使用过时浏览器的警告。	yes	会话（在浏览会话结束时过期）	-	-
system-message-ok	页面重新加载后立即显示的消息。	纯文本消息	会话（在浏览会话结束时过期）或在页面重新加载后立即失效	+	-
system-message-error	页面重新加载后立即显示的错误消息。	纯文本消息	会话（在浏览会话结束时过期）或在页面重新加载后立即失效	+	-

Note:
不支持通过 Web 服务器指令强制为 Zabbix cookie 设置 `HttpOnly` 标志。

脚注

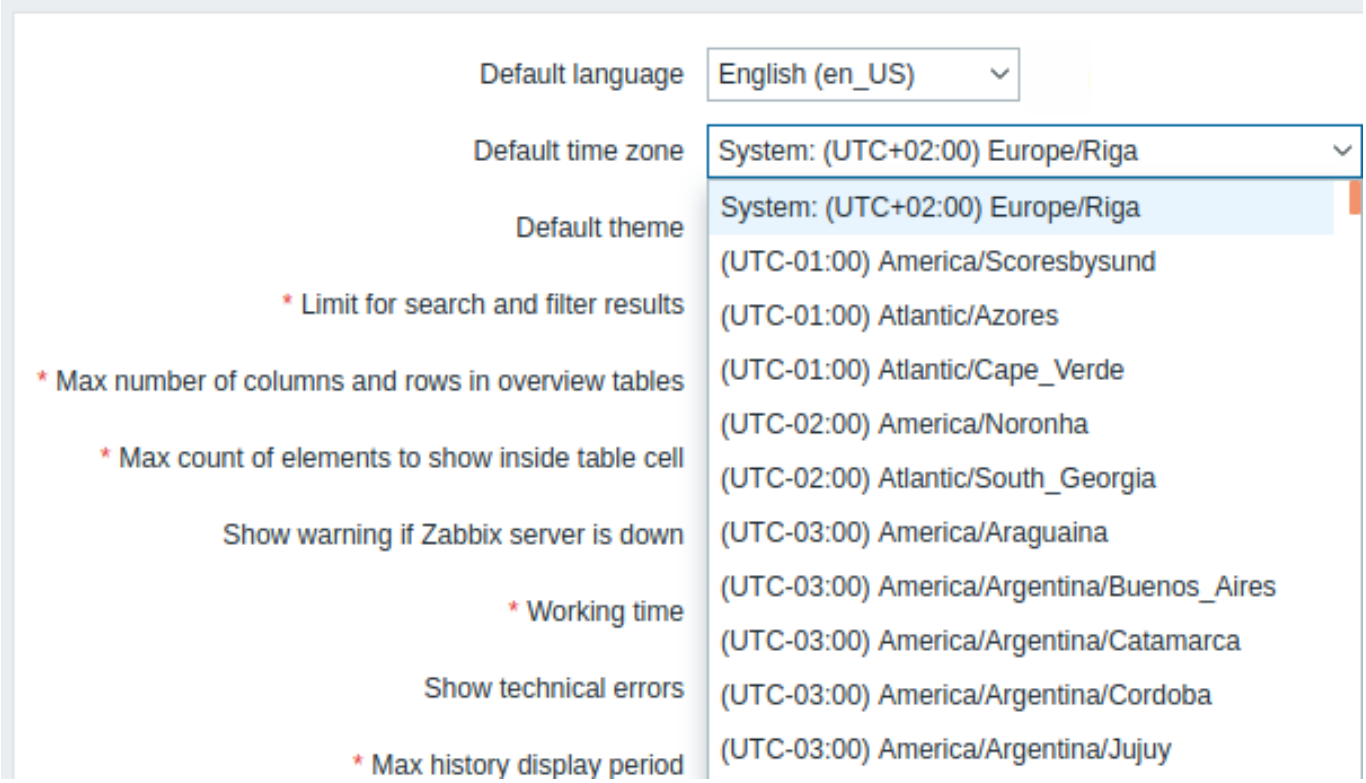
[1] 当 `HttpOnly` 为 “true” 时，cookie 将只能通过 HTTP 协议访问。这意味着脚本语言（如 JavaScript）将无法访问该 cookie。此设置可有效帮助减少通过 XSS 攻击造成的身份窃取（尽管并非所有浏览器都支持）。

[2] `Secure` 表示 cookie 只能通过客户端发起的安全 HTTPS 连接传输。设置为 “true” 时，只有在存在安全连接的情况下才会设置该 cookie。

时区

概述

可以在前端全局设置时区，也可以针对单个用户进行调整。



如果选择了 *System*，前端将使用 Web 服务器的时区（包括 `php.ini` 中 `date.timezone` 的值，如果已设置），而 Zabbix 服务器将使用其运行所在计算机的时区。

Note:
Zabbix 服务器仅在通知中展开宏时（例如，`{EVENT.TIME}` 可以根据用户展开为不同的时区）以及发送通知的时间限制中使用指定的全局/用户时区（请参阅用户媒介配置中的 “生效时间” 设置）。

时区的选择不会影响前端的日期/时间格式。相反，您可以调整界面语言（在安装期间或在用户设置中进行调整）——选择 *English (en_US)* 还会在前端启用美国日期/时间格式。

配置

全局时区：

- 可在安装前端时手动设置。
- 可在 *Administration* > *General* > *GUI* 中修改。

用户级时区：

- 可在配置/更新用户时设置。
- 每个用户都可以在其用户资料中设置。

另请参见：在使用调度间隔时对齐时区。

重置密码

概述　本节介绍在 Zabbix 中重置用户密码的步骤。

步骤　如果您忘记了 Zabbix 密码且无法登录，请联系 Zabbix 管理员。

超级管理员用户可以在用户配置表单中更改所有用户的密码。

如果超级管理员忘记了密码且无法登录，则必须运行以下 SQL 查询，为超级管理员用户应用默认密码（将 `Admin` 替换为相应的超级管理员用户名）：

```
UPDATE users SET passwd = '$2a$10$ZXIvHAEP2ZM.dLXTm6uPHOMVlARXX7cqjbhM6Fn0cANzkCQBWpMrS' WHERE username =
```

运行此查询后，用户密码将设置为 *zabbix*。请务必在首次登录时更改默认密码。

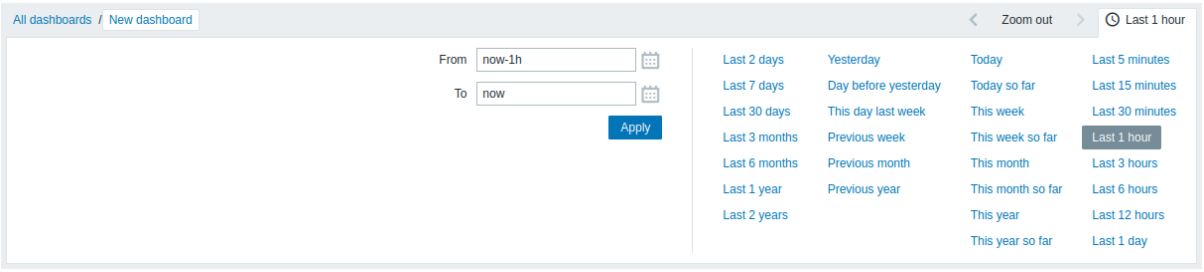
时间段和主机选择器

概述

本页面介绍仪表板上可用的时间段和主机选择器。这些选择器无需编辑各个小组件的设置，即可在所有兼容的仪表板小组件中动态调整显示的数据。

时间段选择器

时间段选择器允许您通过单击鼠标选择常用的时间段。单击仪表板右上角的时间段选择器按钮，可以展开或折叠该选择器。



Today、*This week* 等选项显示整个时间段，包括未来的小时/天。*Today so far*、*This week so far* 等选项仅显示已经过去的小时。

选择时间段后，可以通过单击  箭头按钮，在时间轴上前后移动。*Zoom out* 按钮允许您将时间段向两个方向各缩小 50%。

Note:
对于图形（使用 *Custom* 时间段的图形除外），还可以通过按住鼠标左键在图形中突出显示某个区域来选择显示的时间段。松开鼠标左键后，图形将放大到突出显示的区域。双击图形也可以缩小。

From/To 字段以绝对时间语法（格式为 `Y-m-d H:i:s`）或相对时间语法显示所选时间段。相对时间段可以包含一个或多个数学运算（`-` 或 `+`），例如 `now-1d` 或 `now-1d-2h+5m`。

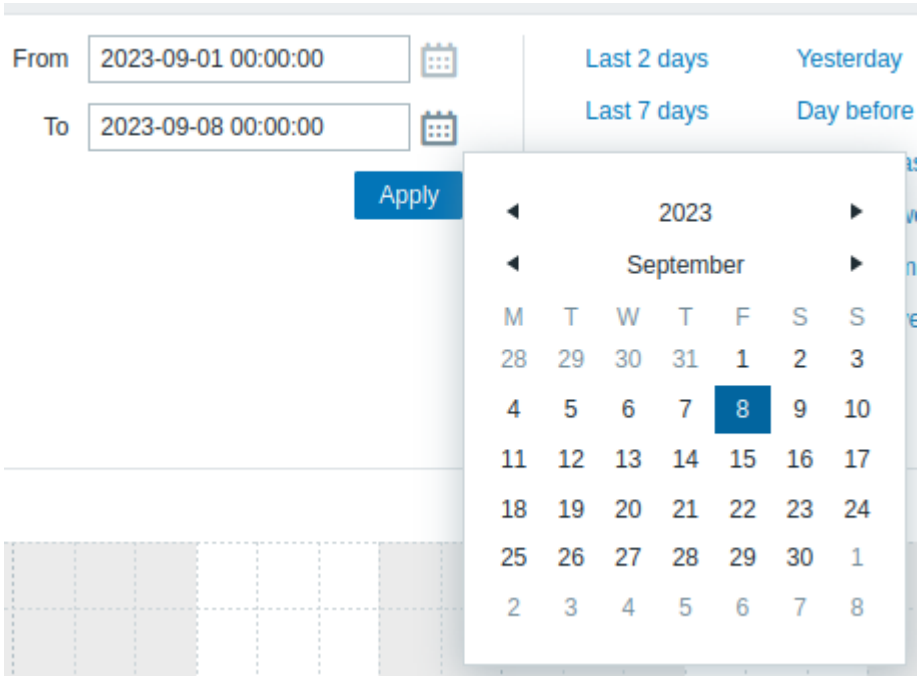
支持以下相对时间缩写：

- `now`
- `s`（秒）
- `m`（分钟）
- `h`（小时）
- `d`（天）
- `w`（周）
- `M`（月）
- `y`（年）

时间段选择器支持精度设置（例如，`now-1d/M` 中的 `/M`）。精度详情如下：

精度	From	To
`m`	Y-m-d H:m:00	Y-m-d H:m:59
`h`	Y-m-d H:00:00	Y-m-d H:59:59
`d`	Y-m-d 00:00:00	Y-m-d 23:59:59
`w`	该周星期一 00:00:00	该周星期日 23:59:59
`M`	该月第一天 00:00:00	该月最后一天 23:59:59
`y`	该年 1 月 1 日 00:00:00	该年 12 月 31 日 23:59:59

还可以使用日期选择器选择时间段。要打开日期选择器，请单击 *From/To* 字段旁边的日历图标。



Note:
在日期选择器中，您可以使用 Tab、Shift+Tab 和键盘箭头按钮在年/月/日之间导航。按 Enter 确认选择。

示例：

From	To	所选时间段
now/d	now/d	今天 00:00 - 23:59
now/d	now/d+1d	今天 00:00 - 明天 23:59
now/w	now/w	本周一 00:00:00 - 本周日 23:59:59
now-1y/w	now-1y/w	一年前该周的周一 00:00:00 - 周日 23:59:59

Attention:
对 *To* 参数使用"now/M+1M" 可能会增加 31 天，这可能导致日期根据月份天数不同而偏移 1-3 天。例如，如果在 1 月使用，结果可能是 3 月 2 日，而不是预期的 2 月 28 日。为避免此问题，请使用"now/M-3d+1M/M"，这样可以准确根据月份长度进行调整。类似地，如果配置向后回溯的 *From*，请使用"now/M+3d-1M/M"。

主机选择器

主机选择器可让您选择一个主机，以便在兼容的仪表板小组件中显示其数据。

在编辑并保存仪表板后，如果至少有一个小组件配置了 *Override host* 参数，并将仪表板设置为数据源，主机选择器便会可用。



配置参数：**Zabbix** 前端

概述　配置参数可用于自定义 Zabbix 前端行为。

在前端配置文件（zabbix.conf.php）中指定参数值。

本页面的结构如下：

- 参数概览（要查看参数的完整详细信息，请单击其名称）
- 参数详细信息

参数	描述
$DB['TYPE']	数据库类型。
$DB['SERVER']	数据库主机名或 IP 地址。
$DB['PORT']	数据库端口。
$DB['DATABASE']	Zabbix 前端用于连接已配置数据库的数据库名称。
$DB['USER']	Zabbix 前端用于连接数据库的用户名。
$DB['PASSWORD']	Zabbix 前端用于连接数据库的密码。
$DB['SCHEMA']	PostgreSQL 使用的数据库模式名称。
$DB['ENCRYPTION']	启用或禁用 Zabbix 前端与数据库之间连接的 TLS 加密。
$DB['KEY_FILE']	用于数据库 TLS 连接的客户端私钥文件路径。
$DB['CERT_FILE']	用于数据库 TLS 连接的客户端证书文件路径。
$DB['CA_FILE']	用于数据库 TLS 连接的 CA 证书文件路径。
$DB['VERIFY_HOST']	启用或禁用对 TLS 证书中数据库服务器主机名的验证。
$DB['CIPHER_LIST']	数据库连接允许使用的 TLS 密码套件列表。
$DB['VAULT']	指定用于存储数据库凭据的 Vault secrets manager。
$DB['VAULT_URL']	用于检索数据库凭据的 Vault secrets manager URL。
$DB['VAULT_PREFIX']	用于检索数据库凭据的 Vault 路径或查询的前缀。
$DB['VAULT_DB_PATH']	Vault secrets manager 中数据库凭据的路径。
$DB['VAULT_TOKEN']	用于访问 Vault secrets manager 的身份验证令牌。
$DB['VAULT_APP_ROLE_ID']	用于向 HashiCorp Vault 进行身份验证的 AppRole role ID。
$DB['VAULT_APP_SECRET_ID']	用于向 HashiCorp Vault 进行身份验证的 AppRole secret ID。
$DB['VAULT_CERT_FILE']	用于向 Vault secrets manager 进行身份验证的客户端证书文件路径。
$DB['VAULT_KEY_FILE']	用于向 Vault secrets manager 进行身份验证的客户端私钥文件路径。
$DB['VAULT_CACHE']	启用或禁用对从 Vault secrets manager 检索的凭据进行缓存。
$ZBX_SERVER	指定一个值以覆盖 Zabbix 主机名/IP 地址。
$ZBX_SERVER_PORT	指定一个值以覆盖 Zabbix 服务器端口。
$ZBX_SERVER_NAME	Zabbix 服务器名称。
$IMAGE_FORMAT_DEFAULT	默认图像格式。
$HISTORY_PROVIDERS[]	Elasticsearch 或 ClickHouse 历史数据存储提供程序的配置。
$SSO['SETTINGS']	SAML 登录的配置设置。
$SSO['CERT_STORAGE']	指定 SAML 证书的存储位置。
$SSO['SP_KEY']	当私钥存储在文件系统中时，SAML 服务提供商（SP）私钥文件的路径。
$SSO['SP_CERT']	当证书存储在文件系统中时，SAML 服务提供商（SP）证书文件的路径。
$SSO['IDP_CERT']	当证书存储在文件系统中时，SAML 身份提供商证书文件的路径。
$ZBX_FEATURE_FLAGS['banners_enabled']	启用或禁用对横幅的支持。
$ZBX_FEATURE_FLAGS['http_auth_enabled']	启用或禁用用户 HTTP 身份验证。
$ZBX_FEATURE_FLAGS['modules_config_enabled']	启用或禁用对模块的访问。
$ZBX_FEATURE_FLAGS['media_type_denylist']	禁用对特定预配置媒介类型的编辑。
$ZBX_SERVER_TLS['ACTIVE']	指定是否为连接 Zabbix 服务器启用 TLS。
$ZBX_SERVER_TLS['CA_FILE']	用于验证服务器证书的证书颁发机构（CA）证书文件路径。
$ZBX_SERVER_TLS['KEY_FILE']	与客户端证书对应的客户端私钥文件路径。
$ZBX_SERVER_TLS['CERT_FILE']	需要双向 TLS 身份验证时的客户端证书文件路径。
$ZBX_SERVER_TLS['CERTIFICATE_ISSUER']	用于与服务器证书进行匹配的颁发者可分辨名称（DN）。
$ZBX_SERVER_TLS['CERTIFICATE_SUBJECT']	用于与服务器证书进行匹配的主题可分辨名称（DN）。
$TELEMETRY_PROVIDERS[]	遥测数据源的配置。

参数详细信息

$DB['TYPE']

数据库类型。

取值：MYSQL - MySQL，POSTGRESQL - PostgreSQL。

$DB['SERVER']

数据库主机名或 IP 地址。

如果未定义 $DB['SERVER']，则使用默认的 Unix 域套接字。对于 MySQL，将 $DB['SERVER'] 设置为 localhost 也会通过默认的 Unix 扇接字建立连接。
 对于 PostgreSQL，$DB['SERVER'] 可以包含多个以逗号分隔的 host:port 条目。如果逗号分隔列表中的某个主机未定义端口，则使用 $DB['PORT'] 中定义的值作为默认端口。如果在 $DB['SERVER'] 中指定了套接字路径（例如 /var/run/pgbouncer），则使用该 Unix 域套接字。

示例：

```
$DB['SERVER'] = 'localhost';
$DB['SERVER'] = 127.0.0.1;
```

$DB['PORT']

数据库端口。

如果未定义，则使用值 0。

$DB['DATABASE']

Zabbix 前端用于连接已配置数据库的数据库名称。

示例：

```
$DB['DATABASE'] = 'zabbix';
```

$DB['USER']

Zabbix 前端用于连接数据库的用户名。

$DB['PASSWORD']

Zabbix 前端用于连接数据库的密码。

$DB['SCHEMA']

用于 PostgreSQL 的数据库架构名称。

$DB['ENCRYPTION']

启用或禁用 Zabbix 前端与数据库之间连接的加密。

取值：`true` - 启用，`false` - 禁用。

$DB['KEY_FILE']

用于数据库 TLS 连接的客户端私钥文件路径。

示例：

```
// 对于 MySQL：
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
```

$DB['CERT_FILE']

用于数据库 TLS 连接的客户端证书文件路径。

示例：

```
// 对于 MySQL：
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
```

$DB['CA_FILE']

用于数据库 TLS 连接的 CA 证书文件路径。

示例：

```
// 对于 MySQL：
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
// 对于 PostgreSQL：
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt';
```

$DB['VERIFY_HOST']

启用或禁用 TLS 证书中数据库服务器主机名的验证。

取值：`true` - 启用，`false` - 禁用。

$DB['CIPHER_LIST']

数据库连接允许使用的 TLS 加密套件列表。

示例：

```
// 对于 MySQL：
$DB['CIPHER_LIST'] = 'TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_1
```

$DB['VAULT']

指定用于存储数据库凭据的 Vault 密钥管理器。

值：CyberArk 或 HashiCorp。

示例：

```
// 对于 CyberArk：
$DB['VAULT'] = 'CyberArk';
// 对于 HashiCorp：
$DB['VAULT'] = 'HashiCorp';
```

$DB['VAULT_URL']

用于检索数据库凭据的 Vault 密钥管理器 URL。

示例：

```
// 对于 CyberArk：
$DB['VAULT_URL'] = 'https://127.0.0.1:1858';
// 对于 HashiCorp：
$DB['VAULT_URL'] = 'https://localhost:8200';
```

$DB['VAULT_PREFIX']

用于检索数据库凭据的 Vault 路径或查询的前缀。

示例：

```
// 对于 HashiCorp：
$DB['VAULT_PREFIX'] = '/v1/secret/data/zabbix/';
```

$DB['VAULT_DB_PATH']

Vault secrets manager 中数据库凭据的路径。

示例：

```
// 对于 CyberArk：
$DB['VAULT_DB_PATH'] = 'AppID=foo&Query=Safe=bar;Object=buzz';
// 对于 HashiCorp：
$DB['VAULT_DB_PATH'] = 'database';
```

$DB['VAULT_TOKEN']

用于访问 Vault 密钥管理器的身份验证令牌。

示例：

```
// 对于 HashiCorp：
$DB['VAULT_TOKEN'] = 'hvs.CAESIIG_PILmULFYOsEyWHxkZ2mF2a8VPKNLE8eHqd4autYGGh4KHGh2cy5aeTYONFNSaUp3ZnpWbDF1
```

$DB['VAULT_APP_ROLE_ID']

用于向 HashiCorp Vault 进行身份验证的 VaultAppSecretID 角色 ID。

示例：

```
$DB['VAULT_APP_ROLE_ID'] = '6f1567f1-538d-7d70-e8e4-5d3e59471b2e';
```

$DB['VAULT_APP_SECRET_ID']

用于向 HashiCorp Vault 进行身份验证的 VaultAppSecretID。

示例：

```
$DB['VAULT_APP_SECRET_ID'] = '89dac8b1-f7d6-96a7-2ecf-f94d2de3d6f1';
```

$DB['VAULT_CERT_FILE']

用于向 Vault secrets manager 进行身份验证的客户端证书文件路径。

示例：

```
// 对于 CyberArk：
$DB['VAULT_CERT_FILE'] = 'conf/certs/cyberark-cert.pem';
```

$DB['VAULT_KEY_FILE']

用于向 Vault secrets manager 进行身份验证的客户端私钥文件路径。

示例：

```
// 对于 CyberArk：
$DB['VAULT_KEY_FILE'] = 'conf/certs/cyberark-key.pem';
```

$DB['VAULT_CACHE']

启用或禁用从 Vault secrets manager 检索的凭据缓存。

值：true - 启用，false - 禁用。

$ZBX_SERVER

指定一个值以覆盖 Zabbix 主机名/IP 地址。
 如果指定了 $ZBX_SERVER 但未指定 $ZBX_SERVER_PORT，则使用默认端口（10051）。

$ZBX_SERVER_PORT

指定一个值以覆盖 Zabbix 服务器端口。
 此参数与 $ZBX_SERVER 一起使用。

$ZBX_SERVER_NAME

Zabbix 服务器名称。

$IMAGE_FORMAT_DEFAULT

默认图像格式。

默认值：IMAGE_FORMAT_PNG
 值：- IMAGE_FORMAT_PNG - PNG 格式 - IMAGE_FORMAT_JPEG - JPEG 格式 - IMAGE_FORMAT_GIF - GIF 格式

$HISTORY_PROVIDERS[]

Elasticsearch 或 ClickHouse 的历史数据存储提供程序配置。

支持的配置参数：- types - 要存储在外部存储中的数据类型数组；- provider - 历史数据提供程序类型：elasticsearch 或 clickhouse；- url - 历史数据提供程序 URL；- db - 数据库名称（用于 ClickHouse）；- username - 数据库用户（用于 ClickHouse）；- password - 数据库密码（用于 ClickHouse）。

示例：

```
// 对于 ClickHouse：
$HISTORY_PROVIDERS[] = [
    'types' => ['uint', 'dbl', 'str', 'log', 'text', 'json'],
    'provider' => 'clickhouse',
    'url' => 'http://localhost:8123',
    'db' => 'zabbix',
    'username' => 'zabbix',
    'password' => 'zabbix'
];
// 对于 Elasticsearch：
$HISTORY_PROVIDERS[] = [
    'types' => ['uint', 'dbl', 'str', 'log', 'text', 'json'],
    'provider' => 'elasticsearch',
    'url' => 'http://localhost:9200'
];
```

$SSO['SETTINGS']

SAML 登录的配置设置。$SSO['SETTINGS'] 数组必须遵循 *SAML PHP Toolkit* 库所需的结构（该库由 Zabbix 提供）。

$SSO['SETTINGS'] 中只能指定以下选项：

- strict
- baseurl
- compress
- contactPerson
- organization
- sp（仅限此列表中指定的选项）
    - attributeConsumingService
    - x509certNew
- idp（仅限此列表中指定的选项）

        - singleLogoutService（仅限一个选项）
            * responseUrl
        - certFingerprint
        - certFingerprintAlgorithm
        - x509certMulti
    • security（仅限此列表中指定的选项）
        - signMetadata
        - wantNameId
        - requestedAuthnContext
        - requestedAuthnContextComparison
        - wantXMLValidation
        - relaxDestinationValidation
        - destinationStrictlyMatches
        - rejectUnsolicitedResponsesWithInResponseTo
        - signatureAlgorithm
        - digestAlgorithm
        - lowercaseUrlencoding

所有其他选项均从数据库中获取，无法覆盖。debug 选项将被忽略。

如果 Zabbix UI 位于 proxy 或负载均衡器之后，可以使用自定义的 use_proxy_headers 选项：

- false（默认）- 忽略 X-Forwarded-* HTTP 标头；
- true - 使用 X-Forwarded-* HTTP 标头构建基础 URL。

有关可用配置选项的完整说明，请参阅官方库文档。

示例：

```
$SSO['SETTINGS'] = [
    'security' => [
        'signatureAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha384',
        'digestAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#sha384',
        // ...
    ],
    // ...
];
```

$SSO['CERT_STORAGE']

指定 SAML 证书的存储位置。

默认值：file
 值：- database - 私钥和证书存储在数据库中；- file - 私钥和证书存储在文件系统中。

示例：

```
$SSO['CERT_STORAGE'] = 'database';
```

$SSO['SP_KEY']

当私钥存储在文件系统中时，SAML 服务提供商（SP）私钥文件的路径。

示例：

```
$SSO['SP_KEY'] = 'conf/certs/sp.key';
```

$SSO['SP_CERT']

当证书存储在文件系统中时，SAML 服务提供商（SP）证书文件的路径。

示例：

```
$SSO['SP_CERT'] = 'conf/certs/sp.crt';
```

$SSO['IDP_CERT']

当证书存储在文件系统中时，SAML 身份提供商证书文件的路径。

示例：

```
$SSO['IDP_CERT'] = 'conf/certs/idp.crt';
```

$ZBX_FEATURE_FLAGS['banners_enabled']

启用或禁用对横幅的支持。

默认值：true
 值：true - 已启用，false - 已禁用。

$ZBX_FEATURE_FLAGS['http_auth_enabled']

启用或禁用用户 HTTP 身份验证。

默认值：true
 值：true - 已启用，false - 已禁用。

$ZBX_FEATURE_FLAGS['modules_config_enabled']

启用或禁用对模块的访问。

默认值：true
 值：true - 已启用，false - 已禁用。

$ZBX_FEATURE_FLAGS['media_type_denylist']

禁止编辑特定的预配置媒介类型。

值：email、script、sms、webhook、push。

示例：

```
$ZBX_FEATURE_FLAGS['media_type_denylist'] = ['webhook','push'];
```

$ZBX_SERVER_TLS['ACTIVE']

指定是否启用 Zabbix 前端与 Zabbix 服务器之间的TLS 连接。

默认值：false
 值：true - 启用，false - 禁用。

$ZBX_SERVER_TLS['CA_FILE']

用于验证服务器证书的证书颁发机构（CA）证书文件路径。

示例：

```
$ZBX_SERVER_TLS['CA_FILE'] = '/etc/zabbix/ssl/ca.crt';
```

$ZBX_SERVER_TLS['KEY_FILE']

与用于双向 TLS 身份验证的客户端证书对应的客户端私钥文件路径。

示例：

```
$ZBX_SERVER_TLS['KEY_FILE'] = '/etc/zabbix/ssl/frontend.key';
```

$ZBX_SERVER_TLS['CERT_FILE']

需要双向 TLS 身份验证时，客户端证书文件的路径。

示例：

```
$ZBX_SERVER_TLS['CERT_FILE'] = '/etc/zabbix/ssl/frontend.crt';
```

$ZBX_SERVER_TLS['CERTIFICATE_ISSUER']

用于与服务器证书进行匹配的颁发者可分辨名称（DN）。

示例：

```
$ZBX_SERVER_TLS['CERTIFICATE_ISSUER']  = 'CN=MyZabbixCA';
```

$ZBX_SERVER_TLS['CERTIFICATE_SUBJECT']

用于与服务器证书进行匹配的使用者专有名称（DN）。

示例：

```
$ZBX_SERVER_TLS['CERTIFICATE_SUBJECT'] = 'CN=zabbix-server.example.com';
```

$TELEMETRY_PROVIDERS[]

遥测数据源的配置。

ClickHouse 支持的配置参数：- url - 设置用于连接遥测数据源的 HTTP 或 HTTPS URL。- db - 设置数据库名称。- username - 设置用于登录数据库的用户名。- password - 设置用于登录数据库的密码。- ssl_cert_file、ssl_key_file、ssl_key_password、ssl_verify_peer、ssl_verify_host、ssl_ca_location、ssl_ca_file - 设置这些选项，以配置与数据库之间的安全（TLS/SSL）连接。- vault_path - 设置保存 username 和 password 值的 Vault 密钥路径。请勿与 username 和 password 选项一起使用。

示例：

```
$TELEMETRY_PROVIDERS[] = [
    'provider' => 'clickhouse',
    'url' => 'http://localhost:8123',
    'db' => 'zabbix',
    'username' => 'zabbix',
    'password' => 'zabbix'
];
```

API

概述　Zabbix API 允许您以编程方式检索和修改 Zabbix 的配置，并提供对历史数据的访问。它被广泛用于：

- 创建与 Zabbix 配合使用的新应用程序。
- 将 Zabbix 集成到第三方软件中。
- 自动化日常任务。

Zabbix API 是一个基于 HTTP 的 API，并作为 web 前端的一部分提供。它使用 JSON-RPC 2.0 协议，这意味着两点：

- API 由一组独立的方法组成。
- 客户端与 API 之间的请求和响应使用 JSON 格式进行编码。

有关该协议和 JSON 的更多信息，请参见 JSON-RPC 2.0 规范 和 JSON 格式主页。

有关将 Zabbix 功能集成到您的 Python 应用程序中的更多信息，请参见适用于 Zabbix 的 Python 库。

> **Note:**
> Zabbix 中的用户访问权限，包括配置和历史数据，取决于用户类型、分配的用户角色 以及用户组。

结构　API 由若干方法组成，这些方法在名义上被分组到不同的 API 中。每个方法都执行一项特定任务。例如，`host.create` 方法属于*host* API，用于创建新的主机。从历史上看，API 有时也被称为 “类”。

大多数 API 至少包含四个方法：`get`、`create`、`update` 和 `delete`，分别用于检索、创建、更新和删除数据，但某些 API 可能提供完全不同的一组方法。

执行请求　在设置好前端后，您可以使用远程 HTTP 请求来调用 API。为此，您需要向位于前端目录中的 `api_jsonrpc.php` 文件发送 HTTP POST 请求。
 例如，如果您的 Zabbix 前端安装在 `https://example.com/zabbix` 下，那么调用 `apiinfo.version` 方法的 HTTP 请求可能如下所示：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"apiinfo.version","params":{},"id":1}'
```

请求必须将 `Content-Type` 标头设置为以下值之一：`application/json-rpc`、`application/json` 或 `application/jsonrequest`。

请求对象必须包含以下属性：

- `jsonrpc` - API 使用的 JSON-RPC 协议版本（Zabbix API 实现的是 JSON-RPC 2.0 版本）；
- `method` - 正在调用的 API 方法；
- `params` - 将传递给 API 方法的参数；
- `id` - 请求的任意标识符（如果省略，API 会将该请求视为 notification）。

如果请求正确，API 返回的响应应如下所示：

```
{
    "jsonrpc": "2.0",
    "result": "8.0.0",
    "id": 1
}
```

响应对象同样包含以下属性：

- `jsonrpc` - JSON-RPC 协议版本；
- `result` - 方法返回的数据；

- id - 对应请求的标识符。

认证　要访问 Zabbix 中的任何数据，您需要：

- 使用在 Zabbix 前端中创建的现有 API token，位于 *Users* > *API tokens* 部分，或者使用Token API 创建的 API token。
- 使用通过user.login 方法获取的认证 token。

例如，如果您希望通过以标准 *Admin* 用户身份登录来获取新的认证 token，那么 JSON 请求如下所示：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"user.login","params":{"username":"Admin","password":"zabbix"},"id":1}
```

如果您正确提供了凭据，API 返回的响应应包含用户认证 token：

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

授权方法　通过 “Authorization” 标头

所有 API 请求都需要身份验证信息或 API token。您可以在请求中使用 Authorization 标头来提供凭据：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer 0424bd59b807674191e7d77572075f33'
```

Attention:
如果您遇到身份验证问题，请参见Authorization 标头转发。

Zabbix API 以不区分大小写的方式接受标头（例如，authorization、Authorization 和 AUTHORIZATION 会被视为相同）。

Authorization 标头支持跨源请求（CORS）。

通过 Zabbix cookie

一个 *"zbx_session"* cookie 被用来授权通过 JavaScript（来自模块或自定义控件）执行的来自 Zabbix UI 的 API 请求。

示例工作流　以下部分将更详细地介绍几个使用示例。

获取主机　现在你已经拥有一个有效的用户身份验证令牌（在以下示例中以变量表示），可用于访问 Zabbix 中的数据。
例如，你可以使用host.get 方法来检索所有已配置主机 的 ID、主机名和接口：

请求：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data @data.json
```

Note:
data.json 是一个包含 JSON 查询的文件。
你也可以不使用文件，而是在 --data 参数中直接传递查询。

data.json

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
```

```
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2
}
```

响应对象将包含所请求的主机数据：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
    "id": 2
}
```

Note:
出于性能考虑，始终建议列出你想要检索的对象属性。
这样可以避免检索全部内容。

创建新的监控项　现在，使用你从之前的 host.get 请求中获得的数据，在主机"Zabbix server" 上创建一个新的监控项。这可以通过item.create 方法完成：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"item.create","params":{"name":"Free disk space on /home/joe/","key_":
```

成功的响应将包含新创建的监控项 ID，该 ID 可用于在后续请求中引用该监控项：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 3
}
```

Note:
item.create 方法以及其他 *create methods* 也可以接受对象数组，并通过一次 API 调用创建多个监控项。

创建多个触发器　因此，如果 *create methods* 接受数组，你可以添加多个触发器，例如下面这个：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"trigger.create","params":[{"description":"Processor load is too high
```

成功响应将包含新创建的触发器 ID：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 4
}
```

更新监控项　通过将监控项的状态设置为 0 来启用它：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"item.update","params":{"itemid":"10092","status":0},"id":5}'
```

成功响应将包含已更新监控项的 ID：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 5
}
```

Note:
item.update 方法以及其他 *update methods* 也可以接受对象数组，并通过一次 API 调用更新多个监控项。

更新多个触发器　通过将多个触发器的状态设置为 0 来启用它们：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"trigger.update","params":[{"triggerid":"13938","status":0},{"triggeri
```

成功响应将包含已更新触发器的 ID：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938",
            "13939"
        ]
    },
    "id": 6
}
```

错误处理　到目前为止，你尝试过的所有操作都运行良好。
但如果你尝试对 API 发起一次错误调用，会发生什么呢？
尝试通过调用host.create 来创建另一个主机，但省略必需的 groups 参数：

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"host.create","params":{"host":"Linux server","interfaces":[{"type":1,
```

随后，响应将包含一条错误消息：

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "No groups for host \"Linux server\"."
    },
    "id": 7
}
```

如果发生了错误，响应对象中将不会包含 `result` 属性，而是包含 `error` 属性，其数据如下：

- `code` - 错误代码；
- `message` - 简短的错误摘要；
- `data` - 更详细的错误消息。

错误可能出现在多种情况下，例如使用了错误的输入值、会话超时，或者尝试访问不存在的对象。
你的应用程序应能够优雅地处理这类错误。

API 版本　为了简化 API 版本管理，自 Zabbix 2.0.4 起，API 的版本与 Zabbix 本身的版本保持一致。您可以使用`apiinfo.version` 方法来查看您正在使用的 API 版本。这对于调整您的应用程序以使用特定版本的功能可能很有帮助。

Zabbix 保证在同一主版本内功能向后兼容。在主版本之间进行不向后兼容的更改时，Zabbix 通常会在下一个版本中将旧功能标记为已弃用，并仅在再下一个版本中将其移除。偶尔，Zabbix 也可能在主版本之间移除功能，而不提供任何向后兼容性。因此，切勿依赖任何已弃用的功能，并应尽快迁移到更新的替代方案。

> **Note:**
> 您可以在API 更新日志 中查看 API 的所有变更。

延伸阅读　现在，你已经具备开始使用 Zabbix API 的足够知识，不过不要就此止步。如需进一步阅读，建议你查看可用 API 列表。

方法参考

本节概述了 Zabbix API 提供的函数，并将帮助您找到可用的类和方法。

监控　Zabbix API 允许您访问在监控期间收集的历史和其他数据。

仪表盘

管理仪表板并基于它们生成定时报表。

仪表盘 API | 模版仪表盘 API | 报表 API

高可用集群

检索服务器节点及其状态的列表。

高可用节点 API

历史记录

检索 Zabbix 监控进程收集的历史值，以便进行演示或进一步处理。

历史数据 API

趋势

检索由 Zabbix Server 计算的趋势值以进行展示或进一步处理。

趋势 API

事件

检索由触发器、网络发现和其他 Zabbix 系统生成的事件，以实现更灵活的情况管理或第三方工具集成。

事件 API

问题

根据给定的参数检索问题。

问题 API

拓扑图

配置拓扑图用于创建 IT 基础架构的详细动态展现。

拓扑图 API

任务

与 Zabbix Server task manager 交互，创建任务并检索响应。

任务 API

APM Zabbix API 允许您访问为应用性能监控（APM）收集的遥测数据。

追踪

检索追踪及其包含的跨度。

APM 追踪 API | APM 跨度 API

指标

获取指标数据点。

APM 指标 API

日志

检索日志记录。

APM 日志 API

服务 Zabbix API 允许您访问在服务监控期间收集的数据。

服务水平协议

定义服务级别目标（SLO），检索有关服务性能的详细服务级别指示器（SLI）信息。

SLA API

服务

管理用于服务级别监控的服务，并检索任何服务的详细 SLA 信息。

服务 API

数据采集 Zabbix API 允许您管理监控系统的配置。

主机和主机组

管理主机组、主机及其相关的所有内容，包括主机接口、主机宏和维护时段。

主机 API | 主机组 API | 用户宏 API | 值映射 API | 维护 API

监控项

定义要监控的监控项。

监控项 API

触发器

配置触发器，以便在系统中出现问题时通知您。
管理触发器依赖关系。

触发器 API

图形

编辑图形或单独的图形监控项，以便更好地呈现收集的数据。

图形 API | 图形监控项 API

模板和模板组

管理模板并将它们链接到主机或其他模板。

模板 API | 模板组 API | 值映射 API

低级别自动发现

配置低级发现规则以及监控项，触发器和图形原型来监视动态实体。

LLD 规则 API | 监控项原型 API | 触发器原型 API | 图形原型 API | 主机原型 API

事件关联性

创建自定义事件相关规则。

关联 API

网络发现

管理网络级发现规则以自动查找和监控新主机。获得对所发现的服务和主机的信息的完全访问。

发现规则 API | 发现检查 API | 发现的主机 API | 发现的服务 API

导入和导出

导出和导入 Zabbix 配置数据，用于配置备份，迁移或大规模配置更新。

配置 API

Web 监控

配置 Web 场景以监控 Web 应用程序和服务。

Web 场景 API

告警　Zabbix API 允许您管理监控系统的动作和告警。

动作和警报

定义动作和报警，以通知用户某些事件或自动执行远程命令。获取有关生成的警报及其接收者的信息。

动作 API | 告警 API

媒体类型

配置媒体类型以及用户将接收告警的多种方式。

媒体类型 API

脚本

配置和执行脚本以帮助您完成日常任务。

脚本 API

用户　Zabbix API 允许您管理监控系统的用户。

用户和用户组

添加将访问 Zabbix 的用户，将他们分配给用户组并授予权限。创建角色以细粒度管理用户权限。

用户 API | 用户组 API | 用户目录 API | 用户角色 API

API Tokens

管理认证 tokens.

Token API

认证

更改身份认证配置选项。

认证 API

设备

管理用于Zabbix 移动应用 的移动设备。

设备 API

管理　使用 Zabbix API，您可以更改监控系统的管理设置。

通用

更改某些全局配置选项。

自动注册 API | 图标映射 API | 图像 API | 设置 API | 正则表达式 API | 模块 API | 连接器 API

审计日志

跟踪每个用户所做的配置更改。

审计日志 API

管家

配置管家。

管家 API

Proxy 和 Proxy 组

管理分布式监控设置中使用的 Proxy。

Proxy API | Proxy 组 API

宏

管理宏。

用户宏 API

Zabbix 信息　从 Zabbix 服务器获取 Zabbix API 版本或上次更新的时间。

API info API | Server info API

API 信息

此类被设计用于检索 API 的元信息。

可用方法：

- apiinfo.version - 检索 Zabbix API 版本

版本

描述

`string apiinfo.version(array)`

此方法允许您检索 Zabbix API 的版本。

> **Attention:**
> 此方法仅对未认证用户可用。

参数

(`array`) 该方法接受一个空数组。

返回值

(`string`) 返回 Zabbix API 的版本。

> **Note:**
> 从 Zabbix 2.0.4 开始，API 的版本与 Zabbix 的版本相匹配。

示例

获取 API 的版本

获取 Zabbix API 的版本。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "apiinfo.version",
    "params": [],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": "8.0.0",
    "id": 1
}
```

源码位置

CAPIInfo::version() 在 *ui/include/classes/api/services/CAPIInfo.php* 文件中。

APM span

此类用于处理为应用程序性能监控（APM）收集的 span。

对象引用：

- APM span

可用方法：

- apmspan.get - 检索 span

APM span 对象

以下对象与 apmspan API 直接相关。

Note:
每个属性对应于遥测数据源中 otel_traces 表的一列。有关列的列表，请参阅支持的遥测数据列。

APM span

APM span 对象表示一个 span（trace 中的单个操作，例如 HTTP 请求、数据库查询或函数调用）。

Span 构成层次结构，从根 span 开始，根 span 表示请求的第一个操作。其他每个 span 都是其操作开始时所在 span 的子 span。

APM span 对象具有以下属性：

Property	Type	Description
timestamp	integer	span 开始时的 Unix 时间（以纳秒为单位）。 属性行为： - 只读
traceid	string	span 所属 trace 的 ID。 属性行为： - 只读
spanid	string	span 的 ID。 属性行为： - 只读
parent_spanid	string	父 span 的 ID。 根 span 为空。 属性行为： - 只读

Property	Type	Description
trace_state	string	跟踪工具附加到 trace 的其他数据（如果有），请参阅 W3C Trace Context。 属性行为： - 只读
span_name	string	span 表示的操作名称（例如 `mysqli_query`）。 属性行为： - 只读
span_kind	string	span 的角色。 可能的值： `Unspecified` - 未指定 span 类型； `Internal` - 在应用程序内部执行的操作； `Server` - 处理传入请求； `Client` - 传出请求； `Producer` - 创建异步操作（例如发送消息）； `Consumer` - 处理异步操作。 属性行为： - 只读
service_name	string	生成 span 的服务名称（例如 `zabbix-frontend`），取自 OpenTelemetry 的 `service.name` 资源属性。 属性行为： - 只读
resource_attributes	object	span 的键值对资源属性（例如 `host.name: eabf8c2a5ddc`）。 属性行为： - 只读
scope_name	string	生成 span 的检测范围名称。 属性行为： - 只读
scope_version	string	检测范围的版本。 属性行为： - 只读
span_attributes	object	span 的键值对 span 属性（例如 `db.system: mysql`）。 属性行为： - 只读
duration	integer	span 的持续时间（以纳秒为单位）。 属性行为： - 只读
status_code	string	span 表示的操作状态。 可能的值： `Unset` - 未向 span 分配明确状态； `Ok` - 操作已成功完成； `Error` - 操作完成时发生错误。 属性行为： - 只读
status_message	string	有关 span 状态的其他信息（如果有）。 属性行为： - 只读
events	array	span 的event 对象数组，即操作期间值得注意的时间点（例如异常）。 属性行为： - 只读

Property	Type	Description
links	array	span 的link 对象数组，这些对象指向同一 trace 或其他 trace 中的相关 span。 属性行为： - 只读

apmspan.get

描述

`integer/array apmspan.get(object parameters)`

此方法允许根据给定的参数检索 span。

Note:
任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
time_from	timestamp	仅返回在给定时间或之后开始的跨度。 参数行为： - 必需
time_till	timestamp	仅返回在给定时间之前开始的跨度。 参数行为： - 必需
traceids	string/array	仅返回属于给定跟踪的跨度。
spanids	string/array	仅返回具有给定 ID 的跨度。
parent_spanids	string/array	仅返回具有给定父跨度的跨度。
resource_attributes	object/array	仅返回具有给定资源属性的跨度。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有跨度。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
resource_attributes_evaltype	integer	`resource_attributes` 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。

参数	类型	描述
span_attributes	object/array	仅返回具有给定跨度属性的跨度。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有跨度。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
span_attributes_evaltype	integer	`span_attributes` 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。
min_duration	integer	仅返回持续时间大于或等于给定值的跨度，单位为纳秒。
max_duration	integer	仅返回持续时间小于给定值的跨度，单位为纳秒。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值为要匹配的单个值或值数组。 支持的属性：`traceid`、`spanid`、`parent_spanid`、`trace_state`、`span_name`、`span_kind`、`service_name`、`scope_name`、`scope_version`、`status_code`、`status_message`。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，则执行 `LIKE "%...%"` 搜索。 支持的属性：`trace_state`、`span_name`、`service_name`、`scope_name`、`scope_version`、`status_message`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`timestamp`、`traceid`、`spanid`、`parent_spanid`、`trace_state`、`span_name`、`span_kind`、`service_name`、`scope_name`、`scope_version`、`duration`、`status_code`、`status_message`。
offset	integer	返回结果前要跳过的记录数。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
limit	integer	
output	query	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下内容之一：

- 对象数组；
- 如果使用了 `countOutput` 参数，则返回检索到的对象数量。

示例

获取跟踪的 span 层次结构

获取跟踪"dee625f33cb0994e86c5cd4640e3b09c" 的前四个 span，并按开始时间排序。
`parent_spanid` 属性显示了各个 span 之间的关系：根 span 没有父级，而其他每个 span 都引用启动它的 span。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "apmspan.get",
    "params": {
        "output": ["spanid", "parent_spanid", "service_name", "span_name", "duration", "status_code"],
        "time_from": 1790848800,
        "time_till": 1790852400,
        "traceids": "dee625f33cb0994e86c5cd4640e3b09c",
        "sortfield": "timestamp",
        "sortorder": "ASC",
        "limit": 4
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "spanid": "7c1a9e2f4b6d8a03",
            "parent_spanid": "",
            "service_name": "zabbix-nginx",
            "span_name": "POST /zabbix.php",
            "duration": "1104826531",
            "status_code": "Unset"
        },
        {
            "spanid": "3f8b2c6d1e9a7054",
            "parent_spanid": "7c1a9e2f4b6d8a03",
            "service_name": "zabbix-frontend",
            "span_name": "POST /zabbix.php",
            "duration": "1098230418",
            "status_code": "Unset"
        },
        {
            "spanid": "b3390ad41931ed08",
            "parent_spanid": "3f8b2c6d1e9a7054",
            "service_name": "zabbix-frontend",
            "span_name": "mysqli::real_connect",
            "duration": "36671360",
            "status_code": "Ok"
        },
        {
            "spanid": "e41f7a2c9d5b3086",
            "parent_spanid": "3f8b2c6d1e9a7054",
            "service_name": "zabbix-frontend",
            "span_name": "mysqli_query",
            "duration": "19704118",
            "status_code": "Ok"
        }
    ],
    "id": 1
}
```

另请参阅

- APM trace
- APM log
- APM metric

源代码

ui/include/classes/api/services/CApmSpan.php 中的 CApmSpan::get()。

APM trace

此类用于处理为应用性能监控（APM）收集的跟踪记录。

对象引用：

- APM trace
- Event
- Link

可用方法：

- apmtrace.get - 获取跟踪记录

APM 跟踪对象

以下对象与 apmtrace API 直接相关。

Note:
除 span_count 和 error_count 外，每个属性都对应于遥测数据源中 otel_traces 表的一列。大多数属性对应于名称相同的列（例如，span_name 属性对应于 SpanName 列）。如果名称不同，下表将指定数据源中的列。有关列的列表，请参阅支持的遥测数据列。

APM 跟踪

APM 跟踪对象表示一次跟踪（请求通过您的应用程序和服务所经过的路径）。一次跟踪由一个或多个span 组成；每个 span 表示一个单独的操作。

APM 跟踪对象基于该跟踪的根 span（启动跟踪且没有父 span 的 span）。除 span_count 和 error_count 外，所有属性都包含根 span 的数据；这两个属性根据跟踪中的所有 span 计算得出。

APM 跟踪对象具有以下属性：

Property	Type	Description
timestamp	integer	根 span 启动时的 Unix 时间（以纳秒为单位）。 Property behavior： - 只读
traceid	string	跟踪的 ID，属于该跟踪的所有 span 共享此 ID。 Property behavior： - 只读
spanid	string	根 span 的 ID。 Property behavior： - 只读
trace_state	string	跟踪工具附加到跟踪的其他数据（如果有）（请参阅 W3C Trace Context）。 Property behavior： - 只读
span_name	string	根 span 所表示操作的名称（例如 POST /zabbix.php）。 Property behavior： - 只读

Property	Type	Description
span_kind	string	根 span 的角色。 可能的值： `Unspecified` - 未指定 span 类型； `Internal` - 在应用程序内执行的操作； `Server` - 处理传入请求； `Client` - 传出请求； `Producer` - 创建异步操作（例如发送消息）； `Consumer` - 处理异步操作。 Property behavior： - 只读
service_name	string	生成根 span 的服务名称（例如 `zabbix-nginx`），取自 OpenTelemetry 的 `service.name` 资源属性。 Property behavior： - 只读
resource_attributes	object	根 span 的键值对 资源属性（例如 `host.name: eabf8c2a5ddc`）。 Property behavior： - 只读
scope_name	string	生成根 span 的 检测范围名称。 Property behavior： - 只读
scope_version	string	检测范围的版本。 Property behavior： - 只读
span_attributes	object	根 span 的键值对 span 属性（例如 `http.method: POST`）。 Property behavior： - 只读
duration	integer	跟踪持续时间，单位为纳秒。 Property behavior： - 只读
status_code	string	根 span 所表示操作的状态。 可能的值： `Unset` - 未向根 span 分配明确状态； `Ok` - 操作已成功完成； `Error` - 操作完成时发生错误。 Property behavior： - 只读
status_message	string	有关根 span 状态的其他信息（如果有）。 Property behavior： - 只读
events	array	根 span 的事件 对象数组，即操作期间值得注意的时间点（例如异常）。 Property behavior： - 只读
links	array	根 span 的链接 对象数组，指向同一跟踪或其他跟踪中的相关 span。 Property behavior： - 只读
span_count	integer	跟踪中的 span 总数。 Property behavior： - 只读

Property	Type	Description
error_count	integer	跟踪中 `status_code` 设置为 `Error` 的 span 数量。 Property behavior： - 只读

事件

事件对象具有以下属性。

Property	Type	Description
timestamp	integer	事件发生时的 Unix 时间（以纳秒为单位）。 数据源列：`Events.Timestamp`
name	string	事件名称。 数据源列：`Events.Name`
attributes	object	包含事件附加信息（如有）的键值属性。 数据源列：`Events.Attributes`

链接

链接对象具有以下属性。

属性	类型	描述
traceid	string	包含所链接 span 的 trace ID。 数据源列：`Links.TraceId`
spanid	string	所链接 span 的 ID。 与 `traceid` 一起唯一标识所链接的 span。 数据源列：`Links.SpanId`
trace_state	string	所链接 span 的 trace state（如果有；请参阅 W3C Trace Context）。 数据源列：`Links.TraceState`
attributes	object	包含链接附加信息的键值属性。 数据源列：`Links.Attributes`

apmtrace.get

描述

`integer/array apmtrace.get(object parameters)`

此方法允许您根据给定参数检索追踪记录。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
time_from	timestamp	仅返回根 span 在给定时间或之后开始的 trace。 参数行为： - 必需
time_till	timestamp	仅返回根 span 在给定时间之前开始的 trace。 参数行为： - 必需
traceids	string/array	仅返回具有给定 ID 的 trace。

参数	类型	描述
resource_attributes	object/array	仅返回其根 span 具有给定资源属性的 trace。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有 trace。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
resource_attributes_evaltype	integer	`resource_attributes` 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。
span_attributes	object/array	仅返回其根 span 具有给定 span 属性的 trace。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有 trace。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
span_attributes_evaltype	integer	`span_attributes` 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。
min_duration	integer	仅返回根 span 的 `duration` 大于或等于给定值的 trace，单位为纳秒。
max_duration	integer	仅返回根 span 的 `duration` 小于给定值的 trace，单位为纳秒。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值为要匹配的单个值或值数组。 支持的属性：`traceid`、`spanid`、`trace_state`、`span_name`、`span_kind`、`service_name`、`scope_name`、`scope_version`、`status_code`、`status_message`。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，则执行 `LIKE "%...%"` 搜索。 支持的属性：`trace_state`、`span_name`、`service_name`、`scope_name`、`scope_version`、`status_message`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`timestamp`、`traceid`、`spanid`、`trace_state`、`span_name`、`span_kind`、`service_name`、`scope_name`、`scope_version`、`duration`、`status_code`、`status_message`。
offset	integer	返回结果前要跳过的记录数。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
limit	integer	
output	query	
searchByAny	boolean	

参数	类型	描述
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下内容之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

获取慢速跟踪记录

获取 zabbix-nginx 服务中最慢的两条跟踪记录。
搜索 2026 年 10 月 1 日 10:00 至 11:00（UTC）期间的跟踪记录，并且仅包含耗时达到 1 秒或更长的记录。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "apmtrace.get",
    "params": {
        "output": ["timestamp", "traceid", "span_name", "service_name", "duration", "status_code", "span_c
        "time_from": 1790848800,
        "time_till": 1790852400,
        "filter": {
            "service_name": "zabbix-nginx"
        },
        "min_duration": 1000000000,
        "sortfield": "duration",
        "sortorder": "DESC",
        "limit": 2
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "timestamp": "1790849274910226417",
            "traceid": "8a4f1c7e2b9d4e6fa0c3b5d7e9f1a2c4",
            "span_name": "POST /zabbix.php",
            "service_name": "zabbix-nginx",
            "duration": "30412907815",
            "status_code": "Error",
            "span_count": "49",
            "error_count": "2"
        },
        {
            "timestamp": "1790850512338104952",
            "traceid": "dee625f33cb0994e86c5cd4640e3b09c",
            "span_name": "POST /zabbix.php",
            "service_name": "zabbix-nginx",
            "duration": "1104826531",
            "status_code": "Unset",
            "span_count": "27",
            "error_count": "0"
        }
    ],
```

```
    "id": 1
}
```

另请参阅

- APM span
- APM metric
- APM log

来源

ui/include/classes/api/services/CApmTrace.php 中的 CApmTrace::get()。

APM 指标

此类用于处理为应用性能监控（APM）收集的指标。

对象引用：

- APM 指标
- 示例

可用方法：

- apmmetric.get - 获取指标数据点

APM 监控项对象

以下对象与 apmmetric API 直接相关。

Note:
除 type 外，每个属性都对应于遥测数据源的监控项表（otel_metrics_gauge、otel_metrics_sum、otel_metrics_histogram 和 otel_metrics_exponential_histogram）中的一列。大多数属性对应于名称相同的列（例如，metric_name 属性对应于 MetricName 列）。如果名称不同，下表将指定数据源中的列名。有关列的列表，请参阅支持的遥测数据列。

APM 指标

APM 指标对象表示指标的一个数据点（某个特定时间的测量值或聚合值）。

APM 指标对象具有以下属性：

属性	类型	描述
type	integer	数据点的类型。 可能的值： 0 - 仪表盘； 1 - 总和； 2 - 直方图； 3 - 指数直方图。 类型由存储数据点的表决定：otel_metrics_gauge、otel_metrics_sum、otel_metrics_histogram 或 otel_metrics_exponential_histogram。 属性行为： - 只读
resource_attributes	object	数据点的键值对资源属性（例如 host.name: eabf8c2a5ddc）。 属性行为： - 只读
resource_schema_url	string	资源属性对应的 OpenTelemetry 架构 URL（如果有）。 属性行为： - 只读

属性	类型	描述
scope_name	string	生成数据点的检测范围名称。 属性行为： - 只读
scope_version	string	检测范围的版本（如果有）。 属性行为： - 只读
scope_attributes	object	检测范围的键值对属性。 属性行为： - 只读
scope_schema_url	string	检测范围及其属性对应的 OpenTelemetry 架构 URL（如果有）。 属性行为： - 只读
service_name	string	生成数据点的服务名称（例如 `zabbix-frontend`），取自 OpenTelemetry `service.name` 资源属性。 属性行为： - 只读
metric_name	string	数据点所属指标的名称（例如 `http.server.request.duration`）。 属性行为： - 只读
metric_description	string	指标的描述（如果有）。 属性行为： - 只读
metric_unit	string	指标值的单位（如果有）（例如 `s` 或 `By`；请参阅 UCUM）。 属性行为： - 只读
attributes	object	键值对数据点属性（例如 `http.request.method: POST`）。 属性行为： - 只读
start_time_unix	integer	数据点测量周期开始时的 Unix 时间（以纳秒为单位）。 属性行为： - 只读
time_unix	integer	数据点的 Unix 时间（以纳秒为单位）。 对于总和、直方图和指数直方图数据点，这是测量周期的结束时间。 属性行为： - 只读
value	float	数据点的值。 适用于仪表盘和总和数据点。 默认值：0。 属性行为： - 只读
flags	integer	数据点标志的位掩码。 如果最低有效位已设置（值为 1），则未记录该数据点的值。 属性行为： - 只读

属性	类型	描述
exemplars	array	示例对象数组，即作为示例保留的单独测量值，以及记录这些测量值期间的跟踪和跨度。 属性行为： - 只读
aggregation_temporality	integer	数据点的值随时间聚合的方式。 适用于总和、直方图和指数直方图数据点。 可能的值： 0 - （默认）未指定； 1 - 增量（聚合自上一个数据点以来的周期）； 2 - 累积（聚合自 `start_time_unix` 以来的周期）。 属性行为： - 只读
is_monotonic	boolean	总和是否只能随时间增加或保持不变。 适用于总和数据点。 默认值：`false`。 属性行为： - 只读
count	integer	数据点中的测量值数量。 适用于直方图和指数直方图数据点。 默认值：0。 属性行为： - 只读
sum	float	数据点中所有测量值的总和。 适用于直方图和指数直方图数据点。 默认值：0。 属性行为： - 只读
bucket_counts	array	整数数组，表示每个直方图桶中的测量值数量。 桶由 `explicit_bounds` 定义。 适用于直方图数据点。 默认值：[]。 属性行为： - 只读
explicit_bounds	array	浮点数数组，表示直方图桶之间的边界。 适用于直方图数据点。 默认值：[]。 属性行为： - 只读
min	float	数据点中测量值的最小值。 适用于直方图和指数直方图数据点。 默认值：0。 属性行为： - 只读

属性	类型	描述
max	float	数据点中测量值的最大值。 适用于直方图和指数直方图数据点。 默认值：0。 属性行为： - 只读
scale	integer	指数直方图的缩放比例，用于确定桶边界和分辨率。 较高的缩放比例会产生更窄的桶和更高的分辨率；较低的缩放比例会产生更宽的桶和更低的分辨率。 适用于指数直方图数据点。 默认值：0。 属性行为： - 只读
zero_count	integer	指数直方图零值桶中的测量值数量。 适用于指数直方图数据点。 默认值：0。 属性行为： - 只读
positive_bucket_counts	array	整数数组，表示每个指数直方图桶中的正测量值数量。 桶边界由 `scale` 确定；第一个桶的索引在 `positive_offset` 中设置。 适用于指数直方图数据点。 默认值：`[]`。 属性行为： - 只读
positive_offset	integer	`positive_bucket_counts` 中第一个桶的索引。 适用于指数直方图数据点。 默认值：0。 属性行为： - 只读
negative_bucket_counts	array	整数数组，表示每个指数直方图桶中的负测量值数量。 桶边界由 `scale` 确定；第一个桶的索引在 `negative_offset` 中设置。 适用于指数直方图数据点。 默认值：`[]`。 属性行为： - 只读
negative_offset	integer	`negative_bucket_counts` 中第一个桶的索引。 适用于指数直方图数据点。 默认值：0。 属性行为： - 只读

示例

示例对象具有以下属性：

Property	Type	Description
traceid	string	记录测量值时所在跟踪的 ID（如果有）。 数据源列：`Exemplars.TraceId`。
spanid	string	记录测量值时所在跨度的 ID（如果有）。 数据源列：`Exemplars.SpanId`。
value	float	测量值。 数据源列：`Exemplars.Value`。
time_unix	integer	记录测量值时的 Unix 时间（以纳秒为单位）。 数据源列：`Exemplars.TimeUnix`。
filtered_attributes	object	测量值的键值属性，这些属性未包含在数据点的 `attributes` 中。 数据源列：`Exemplars.FilteredAttributes`。

apmmetric.get

描述

`integer/array apmmetric.get(object parameters)`

此方法允许您根据给定参数检索指标数据点。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
time_from	timestamp	仅返回时间大于或等于给定时间的数据点。 参数行为： - 必需
time_till	timestamp	仅返回时间早于给定时间的数据点。 参数行为： - 必需
types	integer/array	仅返回指定类型的数据点。 支持的指标类型列表，请参阅APM 指标对象页面。
with_flags_on	integer	仅返回设置了所有指定标志 的数据点。 如需指定多个标志，请使用其值的总和。
with_flags_off	integer	仅返回未设置任何指定标志 的数据点。 如需指定多个标志，请使用其值的总和。
resource_attributes	object/array	仅返回具有指定资源属性的数据点。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有数据点。 可能的运算符 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
resource_attributes_evaltype	integer	用于评估 `resource_attributes` 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。

参数	类型	描述
scope_attributes	object/array	仅返回具有指定作用域属性的数据点。 格式：[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]。 空数组将返回所有数据点。 可能的运算符 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
scope_attributes_evaltype	integer	用于评估 scope_attributes 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。
attributes	object/array	仅返回具有指定属性的数据点。 格式：[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]。 空数组将返回所有数据点。 可能的运算符 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
attributes_evaltype	integer	用于评估 attributes 中条件的评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值为要匹配的单个值或值数组。 支持的属性：resource_schema_url、scope_schema_url、scope_name、scope_version、service_name、metric_name、metric_unit、flags。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，则执行 LIKE "%…%" 搜索。 支持的属性：resource_schema_url、scope_schema_url、scope_name、scope_version、service_name、metric_name、metric_description、metric_unit。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：type、resource_schema_url、scope_schema_url、scope_name、scope_version、service_name、metric_name、metric_unit、start_time_unix、time_unix。
offset	integer	返回结果前要跳过的记录数。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
limit	integer	
output	query	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

参数	类型	描述
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下内容之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

获取直方图数据点

获取 zabbix-frontend 服务的 http.server.request.duration 直方图指标的最新数据点。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "apmmetric.get",
    "params": {
        "output": ["metric_name", "metric_unit", "time_unix", "count", "sum", "min", "max", "explicit_boun
        "time_from": 1790848800,
        "time_till": 1790852400,
        "types": 2,
        "filter": {
            "service_name": "zabbix-frontend",
            "metric_name": "http.server.request.duration"
        },
        "sortfield": "time_unix",
        "sortorder": "DESC",
        "limit": 1
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "metric_name": "http.server.request.duration",
            "metric_unit": "s",
            "time_unix": "1790852390000000000",
            "count": "1842",
            "sum": "412.773",
            "min": "0.0061",
            "max": "1.098",
            "explicit_bounds": ["0.005", "0.05", "0.1", "0.25", "0.5", "1"],
            "bucket_counts": ["0", "211", "904", "512", "167", "41", "7"],
            "exemplars": [
                {
                    "traceid": "dee625f33cb0994e86c5cd4640e3b09c",
                    "spanid": "3f8b2c6d1e9a7054",
                    "value": "1.098",
                    "time_unix": "1790850513439147623",
                    "filtered_attributes": {}
                }
            ]
        }
    ],
    "id": 1
}
```

另请参阅

- APM 跟踪
- APM span
- APM 日志

来源

ui/include/classes/api/services/CApmMetric.php 中的 CApmMetric::get()。

APM 日志

此类用于处理为应用程序性能监控（APM）收集的日志。

对象引用：

- APM 日志

可用方法：

- apmlog.get - 检索日志记录

APM 日志对象

以下对象与 `apmlog` API 直接相关。

Note:
每个属性对应于遥测数据源中 `otel_logs` 表内名称相同的列（例如，`severity_text` 属性对应于 `SeverityText` 列）。有关列的列表，请参阅支持的遥测数据列。

APM 日志

APM 日志对象表示一条日志记录（事件的文本记录，例如支付失败或数据库超时）。日志记录包含有关特定时间发生的事件的信息，包括其严重性、消息和属性。如果日志记录是在属于某个跟踪的操作期间生成的，则还包含该跟踪和跨度的 ID。

APM 日志对象具有以下属性：

Property	Type	Description
timestamp	integer	事件发生时的 Unix 时间（以纳秒为单位）。 属性行为： - 只读
traceid	string	生成日志记录期间所在跟踪的 ID（如果有）。 属性行为： - 只读
spanid	string	生成日志记录期间所在跨度的 ID（如果有）。 属性行为： - 只读
trace_flags	integer	日志记录跟踪上下文的跟踪标志，以位掩码表示。 如果设置了最低有效位（值为 1），则表示该跟踪已采样。 属性行为： - 只读
severity_text	string	应用程序提供的日志记录严重性级别（例如 `INFO`、`WARN` 或 `ERROR`）。 属性行为： - 只读

Property	Type	Description
severity_number	integer	以数字表示的日志记录严重性级别。 可能的值： 0 - UNSPECIFIED； 1-4 - TRACE； 5-8 - DEBUG； 9-12 - INFO； 13-16 - WARN； 17-20 - ERROR； 21-24 - FATAL。 属性行为： - 只读
service_name	string	生成日志记录的服务名称（例如 zabbix-frontend），取自 OpenTelemetry service.name 资源属性。 属性行为： - 只读
body	string	日志记录的消息。 属性行为： - 只读
resource_schema_url	string	资源属性的 OpenTelemetry 架构 URL（如果有）。 属性行为： - 只读
resource_attributes	object	日志记录的键值对资源属性（例如 host.name: eabf8c2a5ddc）。 属性行为： - 只读
scope_schema_url	string	检测范围及其属性的 OpenTelemetry 架构 URL（如果有）。 属性行为： - 只读
scope_name	string	生成日志记录的检测范围名称。 属性行为： - 只读
scope_version	string	检测范围的版本（如果有）。 属性行为： - 只读
scope_attributes	object	检测范围的键值对属性。 属性行为： - 只读
log_attributes	object	日志记录的键值对日志属性（例如 log.file.path: /var/log/zabbix/zabbix_frontend.log）。 属性行为： - 只读
event_name	string	日志记录所表示的事件类型名称（如果有）。 属性行为： - 只读

apmlog.get

描述

```
integer/array apmlog.get(object parameters)
```

此方法允许您根据给定参数检索日志记录。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
time_from	timestamp	仅返回在给定时间或之后发生的事件的日志记录。 参数行为： - 必需
time_till	timestamp	仅返回在给定时间之前发生的事件的日志记录。 参数行为： - 必需
traceids	string/array	仅返回在给定 trace 中生成的日志记录。
spanids	string/array	仅返回在给定 span 中生成的日志记录。
with_flags_on	integer	仅返回设置了给定全部trace 标志的日志记录。 如需指定多个标志，请使用其值之和。
with_flags_off	integer	仅返回未设置任何给定trace 标志的日志记录。 如需指定多个标志，请使用其值之和。
resource_attributes	object/array	仅返回具有给定资源属性的日志记录。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有日志记录。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
resource_attributes_evaltype	integer	`resource_attributes` 中条件的评估方法。 可能的值： 0 - （默认）*And/Or*； 2 - Or。
scope_attributes	object/array	仅返回具有给定作用域属性的日志记录。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有日志记录。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
scope_attributes_evaltype	integer	`scope_attributes` 中条件的评估方法。 可能的值： 0 - （默认）*And/Or*； 2 - Or。

参数	类型	描述
log_attributes	object/array	仅返回具有给定日志属性的日志记录。 格式：`[{"key": "<key>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有日志记录。 可能的operator 值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
log_attributes_evaltype	integer	`log_attributes` 中条件的评估方法。 可能的值： 0 - （默认）*And/Or*； 2 - Or。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值为要匹配的单个值或值数组。 支持的属性：`traceid`、`spanid`、`trace_flags`、`severity_text`、`severity_number`、`service_name`、`resource_schema_url`、`scope_schema_url`、`scope_name`、`scope_version`、`event_name`。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，则执行 `LIKE "%...%"` 搜索。 支持的属性：`severity_text`、`service_name`、`body`、`resource_schema_url`、`scope_schema_url`、`scope_name`、`scope_version`、`event_name`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`timestamp`、`traceid`、`spanid`、`trace_flags`、`severity_text`、`severity_number`、`service_name`、`resource_schema_url`、`scope_schema_url`、`scope_name`、`scope_version`、`event_name`。
offset	integer	返回结果前要跳过的记录数。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
limit	integer	
output	query	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下内容之一：

- 对象数组；
- 如果使用了 `countOutput` 参数，则返回所检索对象的数量。

示例

获取跟踪的错误日志

获取与跟踪"8a4f1c7e2b9d4e6fa0c3b5d7e9f1a2c4" 关联的 ERROR 严重性（严重性编号 17-20）的日志记录。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "apmlog.get",
    "params": {
        "output": "extend",
        "time_from": 1790848800,
        "time_till": 1790852400,
        "traceids": "8a4f1c7e2b9d4e6fa0c3b5d7e9f1a2c4",
        "filter": {
            "severity_number": [17, 18, 19, 20]
        },
        "sortfield": "timestamp",
        "sortorder": "ASC"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "timestamp": "1790849305091452376",
            "traceid": "8a4f1c7e2b9d4e6fa0c3b5d7e9f1a2c4",
            "spanid": "5d2e8f1a9c3b7046",
            "trace_flags": "1",
            "severity_text": "ERROR",
            "severity_number": "17",
            "service_name": "zabbix-frontend",
            "body": "mysqli_query(): (HY000/1205): Lock wait timeout exceeded; try restarting transaction"
            "resource_schema_url": "https://opentelemetry.io/schemas/1.26.0",
            "resource_attributes": {
                "service.name": "zabbix-frontend",
                "host.name": "6583ceeb5f94",
                "host.arch": "x86_64",
                "os.type": "linux"
            },
            "scope_schema_url": "",
            "scope_name": "zabbix-frontend",
            "scope_version": "",
            "scope_attributes": {},
            "log_attributes": {
                "log.file.path": "/var/log/zabbix/zabbix_frontend.log",
                "code.filepath": "/usr/share/zabbix/include/db.inc.php"
            },
            "event_name": ""
        }
    ],
    "id": 1
}
```

搜索日志消息

获取消息中包含 “timeout” 的最新 10 条日志记录的时间、服务和消息。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "apmlog.get",
    "params": {
        "output": ["timestamp", "service_name", "severity_text", "body"],
        "time_from": 1790848800,
        "time_till": 1790852400,
```

```
        "search": {
            "body": "timeout"
        },
        "sortfield": "timestamp",
        "sortorder": "DESC",
        "limit": 10
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "timestamp": "1790849305091452376",
            "service_name": "zabbix-frontend",
            "severity_text": "ERROR",
            "body": "mysqli_query(): (HY000/1205): Lock wait timeout exceeded; try restarting transaction"
        },
        {
            "timestamp": "1790848961730284115",
            "service_name": "zabbix-frontend",
            "severity_text": "ERROR",
            "body": "mysqli_query(): (HY000/1205): Lock wait timeout exceeded; try restarting transaction"
        }
    ],
    "id": 1
}
```

另请参阅

- APM 跟踪
- APM span
- APM 指标

来源

ui/include/classes/api/services/CApmLog.php 中的 CApmLog::get()。

LLD 规则

此类用于处理低级别发现规则。

对象引用：

- LLD 规则
  - HTTP 标头
  - HTTP 查询字段
- LLD 规则过滤器
  - LLD 规则过滤器条件
- LLD 宏路径
- LLD 规则预处理
- LLD 规则覆盖
  - LLD 规则覆盖过滤器
    - LLD 规则覆盖过滤器条件
  - LLD 规则覆盖操作
    - LLD 规则覆盖操作状态
    - LLD 规则覆盖操作发现
    - LLD 规则覆盖操作周期
    - LLD 规则覆盖操作历史记录
    - LLD 规则覆盖操作趋势
    - LLD 规则覆盖操作严重性
    - LLD 规则覆盖操作标签
    - LLD 规则覆盖操作模板

* LLD 规则覆盖操作资产清单

可用方法：

- discoveryrule.create - 创建新的 LLD 规则
- discoveryrule.delete - 删除 LLD 规则
- discoveryrule.get - 检索 LLD 规则
- discoveryrule.update - 更新 LLD 规则

LLD 规则对象

以下对象与 自动发现规则 API 直接相关。

LLD 规则

低级别发现规则对象具有以下属性。

Property	Type	Description
itemid	ID	LLD 规则的 ID。 属性行为： - 只读 - 更新操作时必需
delay	string	LLD 规则的更新间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或者可选地接受一个或多个自定义间隔，所有值以分号分隔。自定义间隔可以混合使用灵活间隔和调度间隔。 接受用户宏。如果使用，值必须是单个宏。不支持多个宏或宏与文本混合使用。灵活间隔可以写成由正斜杠分隔的两个宏（例如 {$FLEX_INTERVAL}/{$FLEX_PERIOD}）。 示例： 1h;wd1-5h9-18;{$Macro1}/1-7,00:00-24:00;0/6-7,12:00-24:00;{$Macr 属性行为： - 当 type 设置为”Zabbix agent” (0)、”Simple check” (3)、”Zabbix internal” (5)、”External check” (10)、”Database monitor” (11)、”IPMI agent” (12)、”SSH agent” (13)、”TELNET agent” (14)、”JMX agent” (16)、”HTTP agent” (19)、”SNMP agent” (20)、”Script” (21)、”Browser” (22) 时必需；或者当 type 设置为”Zabbix agent (active)” (7) 且 key_ 不包含”mqtt.get” 时必需
hostid	ID	LLD 规则所属主机的 ID。 属性行为： - 常量 - 创建操作时必需
flags	integer	发现规则的来源。 可能的值： 1 - 低级别发现规则； 5 - 从原型转换而来的低级别发现规则。 属性行为： - 只读
interfaceid	ID	LLD 规则主机接口的 ID。 属性行为： - 当 LLD 规则属于主机且 type 设置为”Zabbix agent”、”IPMI agent”、”JMX agent” 或”SNMP agent” 时必需 - 当 LLD 规则属于主机且 type 设置为”Simple check”、”External check”、”SSH agent”、”TELNET agent” 或”HTTP agent” 时支持

Property	Type	Description
key_	string	LLD 规则键。 属性行为： - 创建操作时必需 - 对继承对象只读
name	string	LLD 规则的名称。 属性行为： - 创建操作时必需 - 对继承对象只读
type	integer	LLD 规则的类型。 可能的值： 0 - Zabbix agent； 2 - Zabbix trapper； 3 - Simple check； 5 - Zabbix internal； 7 - Zabbix agent (active)； 10 - External check； 11 - Database monitor； 12 - IPMI agent； 13 - SSH agent； 14 - TELNET agent； 16 - JMX agent； 18 - Dependent item； 19 - HTTP agent； 20 - SNMP agent； 21 - Script； 22 - Browser； 23 - Nested。仅当规则的父级是模板（假定该模板在发现时链接到主机原型）或已发现主机（其根节点具有 LLD 规则）时，才允许使用此类型。 属性行为： - 创建操作时必需 - 对继承对象只读
url	string	URL 字符串。 支持用户宏、{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}。 属性行为： - 当 type 设置为”HTTP agent” 时必需 - 对继承对象只读
allow_traps	integer	允许以类似 trapper 监控项的方式填充值。 可能的值： 0 - （默认）不接受传入数据； 1 - 接受传入数据。 属性行为： - 当 type 设置为”HTTP agent” 时支持

Property	Type	Description
authtype	integer	身份验证方法。 当 type 设置为”SSH agent” 时的可能值： 0 - （默认）密码； 1 - 公钥。 当 type 设置为”HTTP agent” 时的可能值： 0 - （默认）无； 1 - 基本； 2 - NTLM； 3 - Kerberos； 4 - Digest。 属性行为： - 当 type 设置为”SSH agent” 或”HTTP agent” 时支持 - 当 type 设置为”HTTP agent” 时，对继承对象只读
description	string	LLD 规则的描述。
error	string	更新 LLD 规则值时出现问题的错误文本。 属性行为： - 只读
follow_redirects	integer	轮询数据时跟随响应重定向。 可能的值： 0 - 不跟随重定向； 1 - （默认）跟随重定向。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
headers	array	执行 HTTP 请求时发送的标头数组。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
http_proxy	string	HTTP(S) proxy 连接字符串。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
ipmi_sensor	string	IPMI 传感器。 属性行为： - 当 type 设置为”IPMI agent” 且 key_ 未设置为”ipmi.get” 时必需 - 当 type 设置为”IPMI agent” 时支持 - 对继承对象只读
jmx_endpoint	string	JMX agent 自定义连接字符串。 默认值： service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi 属性行为： - 当 type 设置为”JMX agent” 时支持
lifetime	string	不再被发现的监控项被删除前的时间段。接受秒数（例如 3600）、带有时间后缀的值（例如 3600s、180m、24h、1d）或用户宏。 默认值：7d。

Property	Type	Description
lifetime_type	integer	删除丢失 LLD 资源的方案。 可能的值： 0 - （默认）达到生命周期阈值后删除； 1 - 不删除； 2 - 立即删除。
enabled_lifetime	string	不再被发现的监控项被禁用前的时间段。接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d）或用户宏。 默认值：0。
enabled_lifetime_type	integer	禁用丢失 LLD 资源的方案。 可能的值： 0 - 达到生命周期阈值后禁用； 1 - 不禁用； 2 - （默认）立即禁用。
master_itemid	ID	主监控项的 ID。 发现规则不能作为另一个发现规则的主监控项。 属性行为： - 当 type 设置为”Dependent item” 时必需 - 对继承对象只读
output_format	integer	是否将响应转换为 JSON。 可能的值： 0 - （默认）存储原始数据； 1 - 转换为 JSON。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
params	string	取决于 LLD 规则类型的其他参数： - SSH 和 Telnet LLD 规则所执行的脚本； - 数据库监控 LLD 规则的 SQL 查询； - 计算型 LLD 规则的公式； - Script 和 Browser LLD 规则的脚本。 属性行为： - 当 type 设置为”Database monitor”、”SSH agent”、”TELNET agent”、”Script” 或”Browser” 时必需 - 当 type 设置为”Script” 或”Browser” 时，对继承对象只读
parameters	object/array	当 type 设置为”Script” 或”Browser” 时的其他参数。 由包含 name 和 value 属性的对象组成的数组，其中 name 必须唯一。 属性行为： - 当 type 设置为”Script” 或”Browser” 时支持 - 对继承对象只读
password	string	身份验证密码。 属性行为： - 当 type 设置为”JMX agent” 且设置了 username 时必需 - 当 type 设置为”Simple check”、”Database monitor”、”SSH agent”、”TELNET agent” 或”HTTP agent” 时支持 - 当 type 设置为”HTTP agent” 时，对继承对象只读

Property	Type	Description
post_type	integer	存储在 posts 属性中的 POST 数据正文类型。 可能的值： 0 - （默认）原始数据； 2 - JSON 数据； 3 - XML 数据。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
posts	string	HTTP(S) 请求正文数据。 属性行为： - 当 type 设置为”HTTP agent” 且 post_type 设置为”JSON data” 或”XML data” 时必需 - 当 type 设置为”HTTP agent” 且 post_type 设置为”Raw data” 时支持 - 对继承对象只读
privatekey	string	私钥文件的名称。 属性行为： - 当 type 设置为”SSH agent” 且 authtype 设置为”public key” 时必需
publickey	string	公钥文件的名称。 属性行为： - 当 type 设置为”SSH agent” 且 authtype 设置为”public key” 时必需
query_fields	array	执行 HTTP 请求时发送的查询字段数组。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
request_method	integer	请求方法的类型。 可能的值： 0 - （默认）GET； 1 - POST； 2 - PUT； 3 - HEAD。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
retrieve_mode	integer	应存储响应的哪一部分。 当 request_method 设置为”GET”、”POST” 或”PUT” 时的可能值： 0 - （默认）正文； 1 - 标头； 2 - 同时存储正文和标头。 当 request_method 设置为”HEAD” 时的可能值： 1 - 标头。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
snmp_oid	string	SNMP OID。 属性行为： - 当 type 设置为”SNMP agent” 时必需 - 对继承对象只读

Property	Type	Description
ssl_cert_file	string	SSL 公钥文件路径。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
ssl_key_file	string	SSL 私钥文件路径。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
ssl_key_password	string	SSL 密钥文件的密码。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
state	integer	LLD 规则的状态。 可能的值： 0 - （默认）正常； 1 - 不支持。 属性行为： - 只读
status	integer	LLD 规则的状态。 可能的值： 0 - （默认）启用的 LLD 规则； 1 - 禁用的 LLD 规则。
status_codes	string	所需 HTTP 状态码的范围，以逗号分隔。也支持将用户宏作为逗号分隔列表的一部分。 示例：200,200-{$M},{$M},200-400 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
templateid	ID	父模板 LLD 规则的 ID。 属性行为： - 只读
timeout	string	监控项数据轮询请求超时。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m）。也接受用户宏。 可能的值范围：1-600s。 默认值：”” - 使用 proxy/全局设置。 属性行为： - 当 type 设置为”Zabbix agent” (0)、”Simple check” (3) 且 key_ 不以”vmware.” 和”icmpping” 开头，”Zabbix agent (active)” (7)、”External check” (10)、”Database monitor” (11)、”SSH agent” (13)、”TELNET agent” (14)、”HTTP agent” (19)、”SNMP agent” (20) 且 snmp_oid 以”walk[” 或”get[” 开头、”Script” (21)、”Browser” (22) 时支持 - 对继承对象只读， - 如果应用默认值且监控项属于由不可访问的 proxy 或 proxy 组监控的主机，则使用全局超时。

Property	Type	Description
trapper_hosts	string	允许的主机。 属性行为： - 当 type 设置为”Zabbix trapper” 时，或当 type 设置为”HTTP agent” 且 allow_traps 设置为”Allow incoming data” 时支持。 如果省略该属性，则会自动将其设置为 {$TRAPPER.ALLOWED_HOSTS}用户宏（默认值）。默认情况下，{$TRAPPER.ALLOWED_HOSTS} 为 127.0.0.1,::1。 如果将该属性设置为空字符串，则将拒绝来自所有主机的传入连接。
username	string	身份验证用户名。 属性行为： - 当 type 设置为”SSH agent”、”TELNET agent” 时，或当 type 设置为”JMX agent” 且设置了 password 时必需 - 当 type 设置为”Simple check”、”Database monitor” 或”HTTP agent” 时支持 - 当 type 设置为”HTTP agent” 时，对继承对象只读
uuid	string	通用唯一标识符，用于将导入的 LLD 规则链接到已存在的规则。如果未提供，则自动生成。 属性行为： - 当 LLD 规则属于模板时支持
verify_host	integer	是否验证连接的主机名与主机证书中的名称匹配。 可能的值： 0 - （默认）不验证； 1 - 验证。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读
verify_peer	integer	是否验证主机证书的真实性。 可能的值： 0 - （默认）不验证； 1 - 验证。 属性行为： - 当 type 设置为”HTTP agent” 时支持 - 对继承对象只读

HTTP 标头

header 对象具有以下属性：

属性	类型	描述
name	string	HTTP 标头名称。 属性行为: - 必填
value	string	标头值。 属性行为: - 必填

HTTP 查询字段

查询字段对象定义了用于指定 URL 参数的名称和值。它具有以下属性：

属性	类型	描述
name	string	参数名称。 属性行为： - 必填
value	string	参数值。 属性行为： - 必填

LLD 规则过滤器

LLD 规则过滤器对象定义了一组可用于过滤已发现对象的条件。它具有以下属性：

属性	类型	描述
conditions	object/array	用于过滤结果的过滤条件集合。条件将按照其在公式中的排列顺序进行排序。 属性行为: - 必需
evaltype	integer	过滤条件的求值方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 属性行为: - 必需
eval_formula	string	用于评估过滤条件的生成表达式。该表达式包含通过其 `formulaid` 引用特定过滤条件的 ID。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 属性行为: - 只读
formula	string	用户定义的表达式，用于评估使用自定义表达式的过滤器条件。该表达式必须包含通过其 `formulaid` 引用特定过滤条件的 ID。表达式中使用的 ID 必须与过滤条件中定义的 ID 完全匹配：不能有任何条件未被使用或遗漏。 属性行为: - 如果 `evaltype` 设置为 “自定义表达式”，则为 必需

LLD 规则过滤条件

LLD 规则过滤条件对象定义了对 LLD 宏值执行的一项独立检查。它具有以下属性：

属性	Type	描述
macro	string	要执行检查的 LLD 宏。 属性行为： - 必填
value	string	要进行比较的值。 属性行为： - 如果 `operator` 设置为 “匹配正则表达式” 或 “不匹配正则表达式”，则为必填
formulaid	string	用于从自定义表达式中引用该条件的任意唯一 ID。只能包含大写字母。在修改过滤条件时，该 ID 必须由用户定义，但在之后请求这些条件时会重新生成。 属性行为： - 如果LLD 规则过滤对象 的 `evaltype` 设置为 “自定义表达式”，则为 必填

属性	Type	描述
operator	integer	条件运算符。 可能的值： 8 - (默认) 匹配正则表达式； 9 - 不匹配正则表达式； 12 - 存在； 13 - 不存在。

Note:
为了更好地理解如何在各种类型的表达式中使用过滤器，请参阅discoveryrule.get 和discoveryrule.create 方法页面中的示例。

LLD 宏路径

LLD 宏路径具有以下属性：

属性	Type	说明
lld_macro	string	LLD 宏。 属性行为： - 必填
path	string	将被赋值给相应宏的值选择器。 属性行为： - 必填

LLD 规则预处理

LLD 规则预处理对象具有以下属性。

属性	类型	描述
type	integer	预处理选项类型。 可能的值： 5 - 正则表达式； 11 - XML XPath； 12 - JSONPath； 14 - 匹配正则表达式； 15 - 不匹配正则表达式； 16 - 检查 JSON 中的错误； 17 - 检查 XML 中的错误； 20 - 使用心跳丢弃未更改的值； 21 - JavaScript； 23 - Prometheus 转 JSON； 24 - CSV 转 JSON； 25 - 替换； 27 - XML 转 JSON； 28 - SNMP walk 值； 29 - SNMP walk 转 JSON； 30 - SNMP get 值。 属性行为： - 必需
params	string	预处理选项使用的其他参数。多个参数使用换行符 (\n) 分隔。 属性行为： - 当 type 设置为 “正则表达式”(5)、“XML XPath”(11)、“JSONPath”(12)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“使用心跳丢弃未更改的值”(20)、“JavaScript”(21)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“替换”(25)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时必需

属性	类型	描述
error_handler	integer	预处理步骤失败时使用的操作类型。 可能的值： 0 - 错误消息由 Zabbix 服务器设置； 1 - 丢弃值； 2 - 设置自定义值； 3 - 设置自定义错误消息。 属性行为： - 当 `type` 设置为 “正则表达式”(5)、“XML XPath”(11)、“JSONPath”(12)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“XML 转 JSON”(27)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时必需
error_handler_params	string	错误处理程序参数。 属性行为： - 当 `error_handler` 设置为 “设置自定义值” 或 “设置自定义错误消息” 时必需

以下参数和错误处理程序适用于每种预处理类型。

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
5	正则表达式	pattern[1]	output[2]		0, 1, 2, 3
11	XML XPath	path[3]			0, 1, 2, 3
12	JSONPath	path[3]			0, 1, 2, 3
14	匹配正则表达式	pattern[1]			0, 1, 2, 3
15	不匹配正则表达式	pattern[1]			0, 1, 2, 3
16	检查 JSON 中的错误	path[3]			0, 1, 2, 3
17	检查 XML 中的错误	path[3]			0, 1, 2, 3
20	使用心跳丢弃未更改的值	seconds^4, 5^			
21	JavaScript	script[2]			
23	Prometheus 转 JSON	pattern^5, 6^			0, 1, 2, 3
24	CSV 转 JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	替换	search string[2]	replacement[2]		

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
27	XML 转 JSON				0, 1, 2, 3
28	SNMP walk 值	OID[2]	格式： 0 - 不变更 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数		0, 1, 2, 3
29	SNMP walk 转 JSON[7]	字段名称[2]	OID 前缀[2]	格式： 0 - 不变更 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数	0, 1, 2, 3
30	SNMP get 值	格式： 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数			0, 1, 2, 3

[1] 正则表达式
 [2] 字符串
 [3] JSONPath 或 XML XPath
 [4] 正整数（支持时间后缀，例如 30s、1m、2h、1d）
 [5] 用户宏
 [6] 遵循以下语法的 Prometheus 模式：`<metric name>{<label name>="<label value>", ...} == <value>`。Prometheus 模式的每个组成部分（指标、标签名称、标签值和指标值）都可以是用户宏。
 [7] 支持多个“字段名称、OID 前缀、格式记录”，各记录使用换行符分隔。

LLD 规则覆盖

LLD 规则覆盖对象定义了一组规则（过滤器、条件和操作），用于覆盖不同原型对象的属性。它具有以下属性：

Property	Type	Description
name	string	唯一的覆盖名称。 属性行为： - 必填
step	integer	覆盖的唯一顺序号。 属性行为： - 必填
stop	integer	如果匹配，则停止处理后续覆盖。 可能的值： 0 - (默认) 不停止处理覆盖； 1 - 如果过滤器匹配，则停止处理覆盖。
filter	object	覆盖过滤器。
operations	object/array	覆盖操作。

LLD 规则覆盖过滤器

LLD 规则覆盖过滤器对象定义了一组条件；如果这些条件与发现的对象匹配，则会应用覆盖。它具有以下属性：

Property	Type	Description
conditions	object/array	用于匹配发现对象的覆盖过滤条件集合。这些条件将按照它们在公式中的排列顺序进行排序。 属性行为: - 必需
evaltype	integer	覆盖过滤条件的求值方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 属性行为: - 必需
eval_formula	string	用于对覆盖过滤条件进行求值的生成表达式。该表达式包含通过其 `formulaid` 引用特定覆盖过滤条件的 ID。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 属性行为: - 只读
formula	string	用户定义的表达式，用于对使用自定义表达式的覆盖过滤器条件进行求值。该表达式必须包含通过其 `formulaid` 引用特定覆盖过滤条件的 ID。表达式中使用的 ID 必须与覆盖过滤条件中定义的 ID 完全匹配：任何条件都不能未被使用或被省略。 属性行为: - 如果 `evaltype` 设置为 “custom expression”，则为 必需

LLD 规则覆盖过滤条件

LLD 规则覆盖过滤条件对象定义了一个单独的检查，用于对 LLD 宏的值执行检查。
它具有以下属性：

属性	类型	描述
macro	string	要执行检查的 LLD 宏。 属性行为： - 必填
value	string	要进行比较的值。 属性行为： - 如果 `operator` 设置为 “匹配正则表达式” 或 “不匹配正则表达式”，则为必填
formulaid	string	用于从自定义表达式中引用该条件的任意唯一 ID。只能包含大写字母。在修改过滤条件时，该 ID 必须由用户定义，但在之后请求这些条件时会重新生成。 属性行为： - 如果LLD 规则覆盖过滤器对象 的 `evaltype` 设置为 “自定义表达式”，则为 必填
operator	integer	条件运算符。 可能的值： 8 - *(默认)* 匹配正则表达式； 9 - 不匹配正则表达式； 12 - 存在； 13 - 不存在。

LLD 规则覆盖操作

LLD 规则覆盖操作是对原型对象执行的一组条件和操作的组合。
它具有以下属性：

属性	Type	说明
operationobject	integer	要执行操作的发现对象类型。 可能的值： 0 - 监控项原型； 1 - 触发器原型； 2 - 图形原型； 3 - 主机原型； 4 - 发现原型。 属性行为： - 必填
operator	integer	覆盖条件的运算符。 可能的值： 0 - (默认) 等于； 1 - 不等于； 2 - 包含； 3 - 不包含； 8 - 匹配； 9 - 不匹配。
value	string	根据所选对象，用于匹配监控项、触发器、图形或主机原型名称的模式。
opstatus	object	用于监控项、触发器和主机原型对象的覆盖操作状态对象。
opdiscover	object	覆盖操作发现状态对象（适用于所有对象类型）。
opperiod	object	用于监控项原型对象的覆盖操作周期（更新间隔）对象。
ophistory	object	用于监控项原型对象的覆盖操作历史数据对象。
optrends	object	用于监控项原型对象的覆盖操作趋势数据对象。
opseverity	object	用于触发器原型对象的覆盖操作严重性对象。
optag	object/array	用于触发器和主机原型对象的覆盖操作标签对象。
optemplate	object/array	用于主机原型对象的覆盖操作模板对象。
opinventory	object	用于主机原型对象的覆盖操作资产记录对象。

LLD 规则覆盖操作状态

设置到已发现对象的 LLD 规则覆盖操作状态。它具有以下属性：

属性	Type	描述
status	integer	覆盖所选对象的状态。 可能的值： 0 - 创建为启用； 1 - 创建为禁用。 属性行为： - 必需

LLD 规则覆盖操作 discover

LLD 规则覆盖操作 discover 状态会被设置到已发现对象。它具有以下属性：

属性	Type	描述
discover	integer	覆盖所选对象的 discover 状态。 可能的值： 0 - 是，继续发现这些对象； 1 - 否，不会发现新对象，现有对象将被标记为丢失。 属性行为： - 必需

LLD 规则覆盖操作周期

LLD 规则覆盖操作周期是为发现的监控项设置的更新间隔值。它具有以下属性：

Property	Type	Description
delay	string	覆盖监控项原型的更新间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或者可选的一个或多个自定义间隔，所有值之间使用分号分隔。自定义间隔可以混合使用灵活间隔和调度间隔。 接受用户宏或 LLD 宏。如果使用宏，该值必须是单个宏。不支持多个宏，或宏与文本混合使用。灵活间隔可以写成由正斜杠分隔的两个宏（例如 {$FLEX_INTERVAL}/{$FLEX_PERIOD}）。 示例： 1h;wd1-5h9-18;{$Macro1}/1-7,00:00-24:00;0/6-7,12:00-24:00;{$Macr 属性行为： - *required*

LLD 规则覆盖操作历史记录

设置到已发现监控项的 LLD 规则覆盖操作历史记录值。它具有以下属性：

属性	Type	说明
history	string	覆盖监控项原型的历史记录，即历史数据应存储多长时间的时间单位。也接受用户宏和 LLD 宏。 属性行为: - 必需

LLD 规则覆盖操作趋势

LLD 规则覆盖操作趋势值将被设置到已发现的监控项。它具有以下属性：

属性	类型	描述
trends	string	覆盖监控项原型的趋势，即趋势数据应保留多长时间的时间单位。也接受用户宏和 LLD 宏。 属性行为: - 必填

LLD 规则覆盖操作严重性

设置到已发现触发器的 LLD 规则覆盖操作严重性值。它具有以下属性：

属性	类型	描述
severity	integer	覆盖触发器原型的严重性。 可能的值： 0 - (默认) 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 高； 5 - 灾难。 属性行为： - 必需

LLD 规则覆盖操作标签

LLD 规则覆盖操作标签对象包含设置到发现对象的标签名称和值。它具有以下属性：

Property	Type	Description
tag	string	新的标签名称。也接受用户宏和 LLD 宏。 属性行为: - *required*
value	string	新的标签值。也接受用户宏和 LLD 宏。

LLD 规则覆盖操作模板

链接到已发现主机的 LLD 规则覆盖操作模板对象。
它具有以下属性：

属性	Type	说明
templateid	ID	覆盖主机原型已链接模板的模板。 属性行为： - 必需

LLD 规则覆盖操作清单

设置到已发现主机的 LLD 规则覆盖操作清单模式值。它具有以下属性：

属性	Type	描述
inventory_mode	integer	覆盖主机原型清单模式。 可能的值： -1 - 禁用； 0 - (默认) 手动； 1 - 自动。 属性行为： - 必需

创建

描述

`object discoveryrule.create(object/array lldRules)`

此方法允许您创建新的 LLD 规则。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(object/array) 要创建的 LLD 规则。

除了标准 LLD 规则属性之外，该方法还接受以下参数。

参数	类型	描述
filter	object	LLD 规则的LLD 规则过滤器。
preprocessing	object/array	LLD 规则预处理选项。
lld_macro_paths	object/array	LLD 规则的lld_macro_path选项。
overrides	object/array	LLD 规则覆盖选项。

返回值

(object) 返回一个对象，其中包含在 itemids 属性下创建的 LLD 规则的 ID。返回 ID 的顺序与传递的 LLD 规则的顺序相匹配。

示例

创建一个 LLD 规则

创建 Zabbix agent LLD 规则以发现挂载的文件系统。发现的监控项将每 30 秒更新一次。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Mounted filesystem discovery",
        "key_": "vfs.fs.discovery",
        "hostid": "10197",
        "type": 0,
        "interfaceid": "112",
        "delay": "30s"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

使用过滤器

使用一组条件创建 LLD 规则以过滤结果。条件将使用逻辑 “和” 运算符组合在一起。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 1,
```

```
            "conditions": [
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex2",
                    "operator": "9"
                },
                {
                    "macro": "{#MACRO3}",
                    "value": "",
                    "operator": "12"
                },
                {
                    "macro": "{#MACRO4}",
                    "value": "",
                    "operator": "13"
                }
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

使用宏路径创建 LLD 规则

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "LLD rule with LLD macro paths",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}",
                "path": "$.path.1"
            },
            {
                "lld_macro": "{#MACRO2}",
                "path": "$.path.2"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

使用自定义表达式过滤器

创建一个带有过滤器的 LLD 规则，该过滤器将使用自定义表达式来评估条件。LLD 规则必须仅发现对象，其中 “{#MACRO1}” 宏的值同时匹配正则表达式 “regex1” 和 “regex2”，并且 “{#MACRO2}” 的值匹配 “regex3” 或 “regex4”。公式 ID“A”，“B”，“C” 和 “D” 是任意选择的。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 3,
            "formula": "(A and B) and (C or D)",
            "conditions": [
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1",
                    "formulaid": "A"
                },
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex2",
                    "formulaid": "B"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex3",
                    "formulaid": "C"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex4",
                    "formulaid": "D"
                }
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
```

```
            "27665"
        ]
    },
    "id": 1
}
```

使用自定义查询字段和标头

创建带有自定义查询字段和标头的 LLD 规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "hostid": "10257",
        "interfaceid": "5",
        "type": 19,
        "name": "API HTTP agent",
        "key_": "api_discovery_rule",
        "delay": "5s",
        "url": "http://127.0.0.1?discoverer.php",
        "query_fields": [
            {
                "name": "mode",
                "value": "json"
            },
            {
                "name": "elements",
                "value": "2"
            }
        ],
        "headers": [
            {
                "name": "X-Type",
                "value": "api"
            },
            {
                "name": "Authorization",
                "value": "Bearer mF_A.B5f-2.1JcM"
            }
        ],
        "allow_traps": 1,
        "trapper_hosts": "127.0.0.1"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 1
}
```

创建带预处理的 LLD 规则

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": " 带预处理的发现规则",
        "key_": "lld.with.preprocessing",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
        "delay": "60s",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": 20,
                "params": "20",
                "error_handler": 0,
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

创建带覆盖的 LLD 规则

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": " 发现数据库主机",
        "key_": "lld.with.overrides",
        "hostid": "10001",
        "type": 0,
        "delay": "60s",
        "interfaceid": "1155",
        "overrides": [
            {
                "name": " 发现 MySQL 主机",
                "step": "1",
                "stop": "1",
                "filter": {
                    "evaltype": "2",
                    "conditions": [
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^mysqld\\.service$"
                        },
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
```

```
                    "value": "^mariadb\\.service$"
                }
            ]
        },
        "operations": [
            {
                "operationobject": "3",
                "operator": "2",
                "value": " 数据库主机",
                "opstatus": {
                    "status": "0"
                },
                "optemplate": [
                    {
                        "templateid": "10170"
                    }
                ],
                "optag": [
                    {
                        "tag": "database",
                        "value": "mysql"
                    }
                ]
            }
        ]
    },
    {
        "name": " 发现 PostgreSQL 主机",
        "step": "2",
        "stop": "1",
        "filter": {
            "evaltype": "0",
            "conditions": [
                {
                    "macro": "{#UNIT.NAME}",
                    "operator": "8",
                    "value": "^postgresql\\.service$"
                }
            ]
        },
        "operations": [
            {
                "operationobject": "3",
                "operator": "2",
                "value": " 数据库主机",
                "opstatus": {
                    "status": "0"
                },
                "optemplate": [
                    {
                        "templateid": "10263"
                    }
                ],
                "optag": [
                    {
                        "tag": "database",
                        "value": "postgresql"
                    }
                ]
            }
        ]
    }
```

```
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "30980"
        ]
    },
    "id": 1
}
```

创建脚本 LLD 规则

使用脚本 LLD 规则创建一个简单的数据采集。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.lldrule",
        "hostid": "12345",
        "type": 21,
        "params": "var request = new HttpRequest();\nreturn request.post(\"https://postman-echo.com/post\"
        "parameters": [{
            "name": "host",
            "value": "{HOST.CONN}"
        }],
        "timeout": "6s",
        "delay": "30s"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

创建一个具有特定禁用时间段的 LLD 规则，且不删除规则。

创建一个自定义禁用时间周期的 LLD 规则，用于在实体不再被发现后禁用它，并设置为永不删除。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "lld disable after 1h",
        "key_": "lld.disable",
        "hostid": "10001",
        "type": 2,
```

```
        "lifetime_type": 1,
        "enabled_lifetime_type": 0,
        "enabled_lifetime": "1h"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "46864"
        ]
    },
    "id": 1
}
```

另请参阅

- LLD 规则过滤器
- LLD 宏路径
- LLD 规则预处理

来源

ui/include/classes/api/services/CDiscoveryRule.php 中的 CDiscoveryRule::create()。

删除

描述

`object discoveryrule.delete(array lldRuleIds)`

此方法允许您删除 LLD 规则。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果 LLD 规则通过无权访问的 proxy 发现实体，用户将无法删除该 LLD 规则。有关更多信息，请参阅用户组配置（*Proxy access list*）。

参数

(array) 要删除的 LLD 规则的 ID。

返回值

(object) 返回一个对象，其中包含在 `ruleids` 属性下删除的 LLD 规则的 IDs。

示例

删除多条 LLD 规则

删除两条 LLD 规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.delete",
    "params": [
        "27665",
        "27668"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "ruleids": [
            "27665",
            "27668"
        ]
    },
    "id": 1
}
```

来源

CDiscoveryRule::delete() in *ui/include/classes/api/services/CDiscoveryRule.php*.

更新

描述

`object discoveryrule.update(object/array lldRules)`

此方法允许您更新现有的 LLD 规则。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要更新的 LLD 规则属性。

必须为每个 LLD 规则定义 `itemid` 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除标准 LLD 规则属性外，该方法还接受以下参数。

参数	类型	描述
filter	object	用于替换现有过滤器的LLD 规则过滤器。
preprocessing	object/array	用于替换现有预处理选项的LLD 规则预处理 选项。 参数行为: - 继承对象为 只读
lld_macro_paths	object/array	用于替换现有 lld_macro_path 选项的 LLD 规则lld_macro_path 选项。 参数行为: - 继承对象为 只读
overrides	object/array	用于替换现有覆盖选项的LLD 规则覆盖 选项。 参数行为: - 继承对象为 只读

返回值

`(object)` 返回一个对象，其中包含 `itemids` 属性下更新的 LLD 规则的 ID。

示例

为 LLD 规则添加过滤器

添加一个过滤器，使 *{#FSTYPE}* 宏的内容与 *@File systems for discovery* 正则表达式匹配。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
```

```
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
                    "macro": "{#FSTYPE}",
                    "value": "@File systems for discovery"
                }
            ]
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

添加 LLD 宏路径

请求 :

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}",
                "path": "$.json.path"
            }
        ]
    },
    "id": 1
}
```

响应 :

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

禁用 trapping

禁用发现规则的 LLD trapping。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "28336",
```

```
        "allow_traps": 0
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 1
}
```

更新 LLD 规则预处理选项

使用预处理规则 “JSONPath” 更新一个 LLD 规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": 12,
                "params": "$.path.to.json",
                "error_handler": 2,
                "error_handler_params": "5"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

更新 LLD 规则脚本

使用不同的脚本更新一个 LLD 规则脚本，并删除之前脚本使用的不必要参数。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

更新 LLD 规则的生命周期。

更新 LLD 规则，以在实体不再被发现后 12 小时禁用它，并在 7 天后删除它。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "46864",
        "lifetime_type": 0,
        "lifetime": "7d",
        "enabled_lifetime_type": 0,
        "enabled_lifetime": "12h"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "46864"
        ]
    },
    "id": 1
}
```

来源

CDiscoveryRule::update() in *ui/include/classes/api/services/CDiscoveryRule.php*.

获取

描述

`integer/array discoveryrule.get(object parameters)`

该方法允许您根据给定参数检索 LLD 规则。

> **Note:**
> 此方法对任何类型的用户都可用。在用户角色设置中可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
itemids	ID/array	仅返回具有给定 ID 的 LLD 规则。
groupids	ID/array	仅返回属于给定组中主机的 LLD 规则。
hostids	ID/array	仅返回属于给定主机的 LLD 规则。

参数	类型	说明
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的 LLD 规则。
interfaceids	ID/array	仅返回使用给定主机接口的 LLD 规则。
monitored	boolean	如果设置为 `true`，则仅返回属于受监控主机的已启用 LLD 规则。
templated	boolean	如果设置为 `true`，则仅返回属于模板的 LLD 规则。
templateids	ID/array	仅返回属于给定模板的 LLD 规则。
selectDiscoveryData	query	返回 `discoveryData` 属性，其中包含 LLD 规则发现对象数据。LLD 规则发现对象将已发现的 LLD 规则关联到发现该规则的 LLD 规则原型。
selectDiscoveryRulePrototypes	query	返回`discoveryRulePrototypes` 属性，其中包含属于该 LLD 规则的 LLD 规则原型。 支持 `count`。
selectFilter	query	返回`filter` 属性，其中包含 LLD 规则所使用过滤器的数据。
selectGraphs	query	返回`graphs` 属性，其中包含属于该 LLD 规则的图形原型。 支持 `count`。
selectHostPrototypes	query	返回`hostPrototypes` 属性，其中包含属于该 LLD 规则的主机原型。 支持 `count`。
selectHosts	query	返回`hosts` 属性，其中包含该 LLD 规则所属主机的数组。
selectItems	query	返回`items` 属性，其中包含属于该 LLD 规则的监控项原型。 支持 `count`。
selectTriggers	query	返回`triggers` 属性，其中包含属于该 LLD 规则的触发器原型。 支持 `count`。
selectLLDMacroPaths	query	返回`lld_macro_paths` 属性，其中包含 LLD 宏列表以及分配给每个对应宏的值路径。
selectPreprocessing	query	返回`preprocessing` 属性，其中包含 LLD 规则预处理选项。
selectOverrides	query	返回`lld_rule_overrides` 属性，其中包含对原型对象执行的覆盖过滤器、条件和操作列表。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值可以是单个值或用于匹配的值数组。 不支持 `text` data type 的属性。 支持附加属性： `host` - LLD 规则所属主机的技术名称。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择：`selectItems`、`selectGraphs`、`selectTriggers`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`itemid`、`name`、`key_`、`delay`、`type`、`status`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下之一：

- 对象数组

- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

从主机中检索发现规则

检索指定主机 ID 的所有发现规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": "extend",
        "hostids": "10202"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "27425",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Network interface discovery",
            "key_": "net.if.discovery",
            "delay": "1h",
            "status": "0",
            "trapper_hosts": "",
            "templateid": "22444",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "1",
            "interfaceid": "119",
            "description": "Discovery of network interfaces as defined in global regular expression \"Netw
            "lifetime": "30d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
```

```
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "parameters": []
        },
        {
            "itemid": "27426",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Mounted filesystem discovery",
            "key_": "vfs.fs.discovery",
            "delay": "1h",
            "status": "0",
            "trapper_hosts": "",
            "templateid": "22450",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "1",
            "interfaceid": "119",
            "description": "Discovery of file systems of different types as defined in global regular expr
            "lifetime": "30d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "parameters": []
        }
    ],
```

```
    "id": 1
}
```

检索过滤条件

检索 LLD 规则 “24681” 的名称及其过滤条件。该过滤器使用 “and” 求值类型，因此 `formula` 属性为空，`eval_formula` 会自动生成。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": ["name"],
        "selectFilter": "extend",
        "itemids": ["24681"]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "24681",
            "name": "Filtered LLD rule",
            "filter": {
                "evaltype": "1",
                "formula": "",
                "conditions": [
                    {
                        "macro": "{#MACRO1}",
                        "value": "@regex1",
                        "operator": "8",
                        "formulaid": "A"
                    },
                    {
                        "macro": "{#MACRO2}",
                        "value": "@regex2",
                        "operator": "9",
                        "formulaid": "B"
                    },
                    {
                        "macro": "{#MACRO3}",
                        "value": "",
                        "operator": "12",
                        "formulaid": "C"
                    },
                    {
                        "macro": "{#MACRO4}",
                        "value": "",
                        "operator": "13",
                        "formulaid": "D"
                    }
                ],
                "eval_formula": "A and B and C and D"
            }
        }
    ],
    "id": 1
}
```

通过 URL 检索 LLD 规则

通过规则 URL 字段值检索主机的 LLD 规则。仅支持与为 LLD 规则定义的 URL 字符串完全匹配。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "hostids": "10257",
        "filter": {
            "type": 19,
            "url": "http://127.0.0.1/discoverer.php"
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28336",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10257",
            "name": "API HTTP agent",
            "key_": "api_discovery_rule",
            "delay": "5s",
            "status": "0",
            "trapper_hosts": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "1",
            "interfaceid": "5",
            "description": "",
            "lifetime": "30d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "http://127.0.0.1/discoverer.php",
            "query_fields": [
                {
                    "name": "mode",
                    "value": "json"
                },
                {
                    "name": "elements",
                    "value": "2"
                }
            ],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [
```

```
                {
                    "name" : "X-Type",
                    "value": "api"
                },
                {
                    "name": "Authorization",
                    "value": "Bearer mF_A.B5f-2.1JcM"
                }
            ],
            "retrieve_mode": "0",
            "request_method": "1",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "parameters": []
        }
    ],
    "id": 1
}
```

获取带有覆盖项的 LLD 规则

获取一个具有多种覆盖项设置的 LLD 规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": ["name"],
        "itemids": "30980",
        "selectOverrides": ["name", "step", "stop", "filter", "operations"]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "name": "Discover database host",
            "overrides": [
                {
                    "name": "Discover MySQL host",
                    "step": "1",
                    "stop": "1",
                    "filter": {
                        "evaltype": "2",
                        "formula": "",
                        "conditions": [
                            {
                                "macro": "{#UNIT.NAME}",
```

```
                        "operator": "8",
                        "value": "^mysqld\\.service$",
                        "formulaid": "A"
                    },
                    {
                        "macro": "{#UNIT.NAME}",
                        "operator": "8",
                        "value": "^mariadb\\.service$",
                        "formulaid": "B"
                    }
                ],
                "eval_formula": "A or B"
            },
            "operations": [
                {
                    "operationobject": "3",
                    "operator": "2",
                    "value": "Database host",
                    "opstatus": {
                        "status": "0"
                    },
                    "optag": [
                        {
                            "tag": "database",
                            "value": "mysql"
                        }
                    ],
                    "optemplate": [
                        {
                            "templateid": "10170"
                        }
                    ]
                }
            ]
        },
        {
            "name": "Discover PostgreSQL host",
            "step": "2",
            "stop": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "macro": "{#UNIT.NAME}",
                        "operator": "8",
                        "value": "^postgresql\\.service$",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A"
            },
            "operations": [
                {
                    "operationobject": "3",
                    "operator": "2",
                    "value": "Database host",
                    "opstatus": {
                        "status": "0"
                    },
                    "optag": [
                        {
```

```
                                    "tag": "database",
                                    "value": "postgresql"
                                }
                            ],
                            "optemplate": [
                                {
                                    "templateid": "10263"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```

另请参阅

- 图形原型
- 主机
- 监控项原型
- LLD 规则过滤器
- 触发器原型

源代码

位于 *ui/include/classes/api/services/CDiscoveryRule.php* 中的 CDiscoveryRule::get()。

LLD 规则原型

此类用于处理低级别发现规则原型。

对象参考：

- LLD 规则原型
  - HTTP 标头
  - HTTP 查询字段
- LLD 规则原型过滤器
  - LLD 规则原型过滤器条件
- LLD 原型宏路径
- LLD 规则原型预处理
- LLD 规则原型覆盖
  - LLD 规则原型覆盖过滤器
    - LLD 规则原型覆盖过滤器条件
  - LLD 规则原型覆盖操作
    - LLD 规则原型覆盖操作状态
    - LLD 规则原型覆盖操作发现
    - LLD 规则原型覆盖操作周期
    - LLD 规则原型覆盖操作历史记录
    - LLD 规则原型覆盖操作趋势
    - LLD 规则原型覆盖操作严重性
    - LLD 规则原型覆盖操作标签
    - LLD 规则原型覆盖操作模板
    - LLD 规则原型覆盖操作资产清单

可用方法：

- discoveryruleprototype.create - 创建新的 LLD 规则原型
- discoveryruleprototype.delete - 删除 LLD 规则原型
- discoveryruleprototype.get - 检索 LLD 规则原型
- discoveryruleprototype.update - 更新 LLD 规则原型

LLD 规则原型对象

以下对象与 discoveryruleprototype API 直接相关。

LLD 规则原型

低级别发现规则原型对象具有以下属性。

Property	Type	Description
itemid	ID	LLD 规则原型的 ID。 属性行为： - 只读 - 更新操作时必需
ruleid	ID	父 LLD 规则/LLD 规则原型的 ID。 属性行为： - 只读 - 创建操作时必需
delay	string	LLD 规则原型的更新间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或者可选的一个或多个自定义间隔，各项之间使用分号分隔。自定义间隔可以混合使用灵活间隔和调度间隔。 接受用户宏。如果使用，值必须是单个宏。不支持多个宏或宏与文本混合使用。灵活间隔可以写成由正斜杠分隔的两个宏（例如 {$FLEX_INTERVAL}/{$FLEX_PERIOD}）。 示例： 1h;wd1-5h9-18;{$Macro1}/1-7,00:00-24:00;0/6-7,12:00-24:00;{$Macr 属性行为： - 当 type 设置为”Zabbix agent” (0)、”Simple check” (3)、”Zabbix internal” (5)、”External check” (10)、”Database monitor” (11)、”IPMI agent” (12)、”SSH agent” (13)、”TELNET agent” (14)、”JMX agent” (16)、”HTTP agent” (19)、”SNMP agent” (20)、”Script” (21)、”Browser” (22) 时必需，或者当 type 设置为”Zabbix agent (active)” (7) 且 key_ 不包含”mqtt.get” 时必需
hostid	ID	LLD 规则原型所属主机的 ID。 属性行为： - 常量 - 创建操作时必需
flags	integer	LLD 规则原型的来源。 可能的值： 3 - 低级别发现规则原型； 7 - 已发现的低级别发现规则原型。 属性行为： - 只读
interfaceid	ID	LLD 规则原型主机接口的 ID。 属性行为： - 当 LLD 规则原型属于主机且 type 设置为”Zabbix agent”、”IPMI agent”、”JMX agent” 或”SNMP agent” 时必需 - 当 LLD 规则原型属于主机且 type 设置为”Simple check”、”External check”、”SSH agent”、”TELNET agent” 或”HTTP agent” 时支持
key_	string	LLD 规则原型键。至少需要一个 LLD 宏。 属性行为： - 创建操作时必需 - 对继承对象只读

Property	Type	Description
name	string	LLD 规则原型的名称。 属性行为： - 创建操作时必需 - 对继承对象只读
type	integer	LLD 规则原型的类型。 可能的值： 0 - Zabbix agent； 2 - Zabbix trapper； 3 - Simple check； 5 - Zabbix internal； 7 - Zabbix agent (active)； 10 - External check； 11 - Database monitor； 12 - IPMI agent； 13 - SSH agent； 14 - TELNET agent； 16 - JMX agent； 18 - Dependent item； 19 - HTTP agent； 20 - SNMP agent； 21 - Script； 22 - Browser； 23 - Nested。 属性行为： - 创建操作时必需 - 对继承对象只读
url	string	URL 字符串。 支持用户宏、{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}。 属性行为： - 当 type 设置为”HTTP agent” 时必需 - 对继承对象只读
allow_traps	integer	允许以类似 trapper 监控项的方式填充值。 可能的值： 0 - (默认) 不接受传入数据； 1 - 接受传入数据。 属性行为： - 当 type 设置为”HTTP agent” 时支持
authtype	integer	身份验证方法。 当 type 设置为”SSH agent” 时的可能值： 0 - (默认) 密码； 1 - 公钥。 当 type 设置为”HTTP agent” 时的可能值： 0 - (默认) 无； 1 - 基本； 2 - NTLM。 属性行为： - 当 type 设置为”SSH agent” 或”HTTP agent” 时支持 - 对继承对象只读（当 type 设置为”HTTP agent” 时）
description	string	LLD 规则原型的描述。

Property	Type	Description
error	string	更新 LLD 规则原型值时出现问题的错误文本。 属性行为： - 只读
follow_redirects	integer	轮询数据时跟随响应重定向。 可能的值： 0 - 不跟随重定向； 1 - *(默认)* 跟随重定向。 属性行为： - 当 `type` 设置为”HTTP agent” 时支持 - 对继承对象只读
headers	array	执行 HTTP 请求时发送的标头数组。 属性行为： - 当 `type` 设置为”HTTP agent” 时支持 - 对继承对象只读
http_proxy	string	HTTP(S) proxy 连接字符串。 属性行为： - 当 `type` 设置为”HTTP agent” 时支持 - 对继承对象只读
ipmi_sensor	string	IPMI 传感器。 属性行为： - 当 `type` 设置为”IPMI agent” 且 `key_` 未设置为”ipmi.get” 时必需 - 当 `type` 设置为”IPMI agent” 时支持 - 对继承对象只读
jmx_endpoint	string	JMX agent 自定义连接字符串。 默认值： service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi 属性行为： - 当 `type` 设置为”JMX agent” 时支持
lifetime	string	不再发现的监控项被删除前的时间段。接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或用户宏。 默认值：7d。
lifetime_type	integer	删除丢失 LLD 资源的方案。 可能的值： 0 - *(默认)* 达到生命周期阈值后删除； 1 - 不删除； 2 - 立即删除。
enabled_lifetime	string	不再发现的监控项被禁用前的时间段。接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或用户宏。 默认值：0。
enabled_lifetime_type	integer	禁用丢失 LLD 资源的方案。 可能的值： 0 - 达到生命周期阈值后禁用； 1 - 不禁用； 2 - *(默认)* 立即禁用。
master_itemid	ID	主监控项的 ID。 发现规则不能作为另一个发现规则的主监控项。 属性行为： - 当 `type` 设置为”Dependent item” 时必需 - 对继承对象只读

Property	Type	Description
output_format	integer	是否将响应转换为 JSON。 可能的值： 0 - *(默认)* 存储原始数据； 1 - 转换为 JSON。 属性行为： - 当 `type` 设置为”HTTP agent” 时支持 - 对继承对象只读
params	string	取决于 LLD 规则原型类型的其他参数： - SSH 和 Telnet LLD 规则所执行的脚本； - 数据库监控 LLD 规则的 SQL 查询； - 计算型 LLD 规则的公式； - 脚本和浏览器 LLD 规则的脚本。 属性行为： - 当 `type` 设置为”Database monitor”、”SSH agent”、”TELNET agent”、”Script” 或”Browser” 时必需 - 对继承对象只读（当 `type` 设置为”Script” 或”Browser” 时）
parameters	object/array	当 `type` 设置为”Script” 或”Browser” 时的其他参数。 由包含 `name` 和 `value` 属性的对象组成的数组，其中 `name` 必须唯一。 属性行为： - 当 `type` 设置为”Script” 或”Browser” 时支持 - 对继承对象只读
password	string	身份验证密码。 属性行为： - 当 `type` 设置为”JMX agent” 且设置了 `username` 时必需 - 当 `type` 设置为”Simple check”、”Database monitor”、”SSH agent”、”TELNET agent” 或”HTTP agent” 时支持 - 对继承对象只读（当 `type` 设置为”HTTP agent” 时）
post_type	integer	存储在 `posts` 属性中的 POST 数据正文类型。 可能的值： 0 - *(默认)* 原始数据； 2 - JSON 数据； 3 - XML 数据。 属性行为： - 当 `type` 设置为”HTTP agent” 时支持 - 对继承对象只读
posts	string	HTTP(S) 请求正文数据。 属性行为： - 当 `type` 设置为”HTTP agent” 且 `post_type` 设置为”JSON data” 或”XML data” 时必需 - 当 `type` 设置为”HTTP agent” 且 `post_type` 设置为”Raw data” 时支持 - 对继承对象只读
privatekey	string	私钥文件的名称。 属性行为： - 当 `type` 设置为”SSH agent” 且 `authtype` 设置为”public key” 时必需
publickey	string	公钥文件的名称。 属性行为： - 当 `type` 设置为”SSH agent” 且 `authtype` 设置为”public key” 时必需
query_fields	array	执行 HTTP 请求时发送的查询字段数组。 属性行为： - 当 `type` 设置为”HTTP agent” 时支持 - 对继承对象只读

Property	Type	Description
request_method	integer	请求方法类型。 可能的值： 0 - (默认) GET； 1 - POST； 2 - PUT； 3 - HEAD。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读
retrieve_mode	integer	应存储响应的哪一部分。 当 request_method 设置为"GET"、"POST" 或"PUT" 时的可能值： 0 - (默认) 正文； 1 - 标头； 2 - 同时存储正文和标头。 当 request_method 设置为"HEAD" 时的可能值： 1 - 标头。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读
snmp_oid	string	SNMP OID。 属性行为： - 当 type 设置为"SNMP agent" 时必需 - 对继承对象只读
ssl_cert_file	string	SSL 公钥文件路径。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读
ssl_key_file	string	SSL 私钥文件路径。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读
ssl_key_password	string	SSL 密钥文件密码。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读
state	integer	LLD 规则原型的状态。 可能的值： 0 - (默认) 正常； 1 - 不支持。 属性行为： - 只读
status	integer	LLD 规则原型的状态。 可能的值： 0 - (默认) 已启用的 LLD 规则原型； 1 - 已禁用的 LLD 规则原型。

Property	Type	Description
status_codes	string	所需 HTTP 状态码的范围，使用逗号分隔。也支持将用户宏作为逗号分隔列表的一部分。 示例：200,200-{$M},{$M},200-400 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读
templateid	ID	父模板 LLD 规则原型的 ID。 属性行为： - 只读
timeout	string	监控项数据轮询请求超时。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m）。也接受用户宏。 可能的值范围：1-600s。 默认值："" - 使用 proxy/全局设置。 属性行为： - 当 type 设置为"Zabbix agent" (0)、"Simple check" (3) 且 key_ 不以"vmware." 和"icmpping" 开头，"Zabbix agent (active)" (7)、"External check" (10)、"Database monitor" (11)、"SSH agent" (13)、"TELNET agent" (14)、"HTTP agent" (19)、"SNMP agent" (20) 且 snmp_oid 以"walk[" 或"get[" 开头、"Script" (21)、"Browser" (22) 时支持 - 对继承对象只读
trapper_hosts	string	允许的主机。 属性行为： - 当 type 设置为"Zabbix trapper" 时，或当 type 设置为"HTTP agent" 且 allow_traps 设置为"Allow incoming data" 时支持。 如果省略该属性，则会自动将其设置为 {$TRAPPER.ALLOWED_HOSTS} 用户宏（默认值）。默认情况下，{$TRAPPER.ALLOWED_HOSTS} 为 127.0.0.1,::1。 如果将该属性设置为空字符串，则将拒绝来自所有主机的传入连接。
username	string	身份验证用户名。 属性行为： - 当 type 设置为"SSH agent"、"TELNET agent" 时，或当 type 设置为"JMX agent" 且设置了 password 时必需 - 当 type 设置为"Simple check"、"Database monitor" 或"HTTP agent" 时支持 - 对继承对象只读（当 type 设置为"HTTP agent" 时）
uuid	string	通用唯一标识符，用于将导入的 LLD 规则原型链接到已存在的 LLD 规则原型。如果未提供，则自动生成。 属性行为： - 当 LLD 规则原型属于模板时支持
verify_host	integer	是否验证连接的主机名与主机证书中的名称匹配。 可能的值： 0 - *(默认)* 不验证； 1 - 验证。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读

Property	Type	Description
verify_peer	integer	是否验证主机证书是否真实有效。 可能的值： 0 - (默认) 不验证； 1 - 验证。 属性行为： - 当 type 设置为"HTTP agent" 时支持 - 对继承对象只读

HTTP 标头

标头对象具有以下属性：

属性	类型	说明
name	string	HTTP 标头名称。 属性行为： - 必填
value	string	标头值。 属性行为： - 必填

HTTP 查询字段

查询字段对象定义了用于指定 URL 参数的名称和值。它具有以下属性：

属性	类型	描述
name	string	参数名称。 属性行为： - 必填
value	string	参数值。 属性行为： - 必填

LLD 规则原型过滤器

LLD 规则原型过滤器对象定义了一组可用于过滤已发现对象的条件。它具有以下属性：

Property	Type	Description
conditions	object/array	用于过滤结果的过滤条件集合。这些条件将按照它们在公式中的排列顺序进行排序。 Property behavior: - 必需
evaltype	integer	过滤条件的求值方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 Property behavior: - 必需

Property	Type	Description
eval_formula	string	用于对过滤条件求值的生成表达式。该表达式包含通过其 `formulaid` 引用特定过滤条件的 ID。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 Property behavior: - 只读
formula	string	用户定义的表达式，用于对使用自定义表达式的过滤器条件进行求值。该表达式必须包含通过其 `formulaid` 引用特定过滤条件的 ID。表达式中使用的 ID 必须与过滤条件中定义的 ID 完全匹配：任何条件都不能未被使用或被省略。 Property behavior: - 如果 `evaltype` 设置为"custom expression"，则为 必需

LLD 规则原型过滤条件

LLD 规则原型过滤条件对象定义了一个单独的检查，用于对 LLD 宏的值执行检查。它具有以下属性：

Property	Type	Description
macro	string	要执行检查的 LLD 宏。 属性行为： - 必填
value	string	用于比较的值。 属性行为： - 如果 `operator` 设置为 "匹配正则表达式" 或 "不匹配正则表达式"，则为必填
formulaid	string	用于从自定义表达式中引用该条件的任意唯一 ID。只能包含大写字母。在修改过滤条件时，该 ID 必须由用户定义，但在之后请求这些条件时会重新生成。 属性行为： - 如果LLD 规则原型过滤对象 的 `evaltype` 设置为 "自定义表达式"，则为必填
operator	integer	条件运算符。 可能的值： 8 - *(默认)* 匹配正则表达式； 9 - 不匹配正则表达式； 12 - 存在； 13 - 不存在。

Note:
为了更好地理解如何将过滤器与各种类型的表达式配合使用，请参见discoveryruleprototype.get和discoveryruleprototype.create 方法页面中的示例。

LLD 宏路径

LLD 宏路径具有以下属性：

属性	Type	描述
lld_macro	string	LLD 宏。 属性行为: - 必填
path	string	将被分配给相应宏的值选择器。 属性行为: - 必填

LLD 规则原型预处理

LLD 规则原型预处理对象具有以下属性。

Property	Type	Description
type	integer	预处理选项类型。 可能的值： 5 - 正则表达式； 11 - XML XPath； 12 - JSONPath； 14 - 匹配正则表达式； 15 - 不匹配正则表达式； 16 - 检查 JSON 中的错误； 17 - 检查 XML 中的错误； 20 - 使用心跳丢弃未更改的值； 21 - JavaScript； 23 - Prometheus 转 JSON； 24 - CSV 转 JSON； 25 - 替换； 27 - XML 转 JSON； 28 - SNMP walk 值； 29 - SNMP walk 转 JSON； 30 - SNMP get 值。 Property behavior： - *required*
params	string	预处理选项使用的其他参数。多个参数使用换行符 (\n) 分隔。 Property behavior： - 当 `type` 设置为 “正则表达式”(5)、“XML XPath”(11)、“JSONPath”(12)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“使用心跳丢弃未更改的值”(20)、“JavaScript”(21)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“替换”(25)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时为 *required*
error_handler	integer	预处理步骤失败时使用的操作类型。 可能的值： 0 - 错误消息由 Zabbix 服务器设置； 1 - 丢弃值； 2 - 设置自定义值； 3 - 设置自定义错误消息。 Property behavior： - 当 `type` 设置为 “正则表达式”(5)、“XML XPath”(11)、“JSONPath”(12)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“XML 转 JSON”(27)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时为 *required*
error_handler_params	string	错误处理程序参数。 Property behavior： - 当 `error_handler` 设置为 “设置自定义值” 或 “设置自定义错误消息” 时为 *required*

以下参数和错误处理程序适用于每种预处理类型。

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
5	正则表达式	pattern[1]	output[2]		0, 1, 2, 3
11	XML XPath	path[3]			0, 1, 2, 3
12	JSONPath	path[3]			0, 1, 2, 3
14	匹配正则表达式	pattern[1]			0, 1, 2, 3
15	不匹配正则表达式	pattern[1]			0, 1, 2, 3
16	检查 JSON 中的错误	path[3]			0, 1, 2, 3
17	检查 XML 中的错误	path[3]			0, 1, 2, 3
20	使用心跳丢弃未更改的值	seconds^4, 5^			
21	JavaScript	script[2]			
23	Prometheus 转 JSON	pattern^5, 6^			0, 1, 2, 3
24	CSV 转 JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	替换	search string[2]	replacement[2]		
27	XML 转 JSON				0, 1, 2, 3
28	SNMP walk 值	OID[2]	格式： 0 - 不变更 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数		0, 1, 2, 3
29	SNMP walk 转 JSON[7]	字段名称[2]	OID 前缀[2]	格式： 0 - 不变更 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数	0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
30	SNMP get 值	格式： 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数			0, 1, 2, 3

[1] 正则表达式
 [2] 字符串
 [3] JSONPath 或 XML XPath
 [4] 正整数（支持时间后缀，例如 30s、1m、2h、1d）
 [5] 用户宏
 [6] 遵循以下语法的 Prometheus 模式：`<metric name>{<label name>="<label value>", ...} == <value>`。每个 Prometheus 模式组件（指标、标签名称、标签值和指标值）都可以是用户宏。
 [7] 支持多个 “字段名称、OID 前缀、格式记录”，各记录使用换行符分隔。

LLD 规则原型覆盖

LLD 规则原型覆盖对象定义了一组规则（过滤器、条件和操作），用于覆盖不同原型对象的属性。它具有以下属性：

Property	Type	Description
name	string	唯一的覆盖名称。 属性行为： - 必填
step	integer	覆盖的唯一顺序编号。 属性行为： - 必填
stop	integer	如果匹配，则停止处理后续覆盖。 可能的值： 0 - *(默认)* 不停止处理覆盖； 1 - 如果过滤器匹配，则停止处理覆盖。
filter	object	覆盖过滤器。
operations	object/array	覆盖操作。

LLD 规则原型覆盖过滤器

LLD 规则原型覆盖过滤器对象定义了一组条件；如果这些条件与发现的对象匹配，则会应用覆盖。它具有以下属性：

Property	Type	Description
conditions	object/array	用于匹配发现对象的覆盖过滤条件集合。这些条件将按照它们在公式中的排列顺序进行排序。 Property behavior: - *required*
evaltype	integer	覆盖过滤条件的求值方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 Property behavior: - *required*

Property	Type	Description
eval_formula	string	用于对覆盖过滤条件进行求值的生成表达式。该表达式包含一些 ID，这些 ID 通过其 `formulaid` 引用特定的覆盖过滤条件。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 Property behavior: - *read-only*
formula	string	用户定义的表达式，用于对使用自定义表达式的覆盖过滤器条件进行求值。该表达式必须包含一些 ID，这些 ID 通过其 `formulaid` 引用特定的覆盖过滤条件。表达式中使用的 ID 必须与覆盖过滤条件中定义的 ID 完全匹配：不能有任何条件未被使用或省略。 Property behavior: - 如果 `evaltype` 设置为"custom expression"，则为 *required*

LLD 规则原型覆盖过滤条件

LLD 规则原型覆盖过滤条件对象定义了一个单独的检查，用于对 LLD 宏的值执行检查。它具有以下属性：

Property	Type	Description
macro	string	要执行检查的 LLD 宏。 Property behavior: - *required*
value	string	用于比较的值。 Property behavior: - 当 `operator` 设置为 "匹配正则表达式" 或 "不匹配正则表达式" 时为 *required*
formulaid	string	用于从自定义表达式中引用该条件的任意唯一 ID。只能包含大写字母。在修改过滤条件时，该 ID 必须由用户定义，但在之后请求这些条件时会重新生成。 Property behavior: - 当LLD 规则原型覆盖过滤器对象 的 `evaltype` 设置为 "自定义表达式" 时为 *required*
operator	integer	条件运算符。 可能的值： 8 - *(default)* 匹配正则表达式； 9 - 不匹配正则表达式； 12 - 存在； 13 - 不存在。

LLD 规则原型覆盖操作

LLD 规则原型覆盖操作是对原型对象执行的一组条件和操作的组合。它具有以下属性：

属性	Type	描述
operationobject	integer	要执行操作的已发现对象类型。 可能的值： 0 - 监控项原型； 1 - 触发器原型； 2 - 图形原型； 3 - 主机原型。 属性行为： - 必填

属性	Type	描述
operator	integer	覆盖条件运算符。 可能的值： 0 - *(默认)* 等于； 1 - 不等于； 2 - 包含； 3 - 不包含； 8 - 匹配； 9 - 不匹配。
value	string	根据所选对象，用于匹配监控项、触发器、图形或主机原型名称的模式。
opstatus	object	用于监控项、触发器和主机原型对象的覆盖操作状态对象。
opdiscover	object	覆盖操作发现状态对象（适用于所有对象类型）。
opperiod	object	用于监控项原型对象的覆盖操作周期（更新间隔）对象。
ophistory	object	用于监控项原型对象的覆盖操作历史数据对象。
optrends	object	用于监控项原型对象的覆盖操作趋势数据对象。
opseverity	object	用于触发器原型对象的覆盖操作严重性对象。
optag	object/array	用于触发器和主机原型对象的覆盖操作标签对象。
optemplate	object/array	用于主机原型对象的覆盖操作模板对象。
opinventory	object	用于主机原型对象的覆盖操作资产记录对象。

LLD 规则原型覆盖操作状态

设置到已发现对象的 LLD 规则原型覆盖操作状态。它具有以下属性：

Property	Type	Description
status	integer	覆盖所选对象的状态。 可能的值： 0 - 创建为启用； 1 - 创建为禁用。 属性行为： *- required*

LLD 规则原型覆盖操作 discover

LLD 规则原型覆盖操作 discover 状态被设置为已发现对象。它具有以下属性：

属性	Type	描述
discover	integer	覆盖所选对象的 discover 状态。 可能的值： 0 - 是，继续发现对象； 1 - 否，不会发现新对象，现有对象将被标记为丢失。 属性行为： *- 必填*

LLD 规则原型覆盖操作周期

LLD 规则原型覆盖操作周期是为发现的监控项设置的更新间隔值。它具有以下属性：

属性	类型	描述
delay	string	覆盖监控项原型的更新间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或者可选地接受一个或多个自定义间隔，各项之间使用分号分隔。自定义间隔可以混合使用灵活间隔和调度间隔。 接受用户宏或 LLD 宏。如果使用宏，该值必须是单个宏。不支持多个宏，或宏与文本混合使用。灵活间隔可以写成由正斜杠分隔的两个宏（例如 {$FLEX_INTERVAL}/{$FLEX_PERIOD}）。 示例： 1h;wd1-5h9-18;{$Macro1}/1-7,00:00-24:00;0/6-7,12:00-24:00;{$Macr 属性行为： - 必需

LLD 规则原型覆盖操作历史记录

设置到已发现监控项的 LLD 规则原型覆盖操作历史记录值。它具有以下属性：

Property	Type	Description
history	string	覆盖监控项原型的历史记录，即历史数据应存储多长时间的时间单位。也接受用户宏和 LLD 宏。 属性行为: - 必填

LLD 规则原型覆盖操作趋势

LLD 规则原型覆盖操作的趋势值会设置到发现的监控项。它具有以下属性：

属性	Type	描述
trends	string	覆盖监控项原型的趋势，即趋势数据应存储多长时间的时间单位。也接受用户宏和 LLD 宏。 属性行为: - 必需

LLD 规则原型覆盖操作严重性

设置到已发现触发器的 LLD 规则原型覆盖操作严重性值。它具有以下属性：

属性	Type	描述
severity	integer	覆盖触发器原型的严重性。 可能的值： 0 - *(默认)* 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。 属性行为： - 必需

LLD 规则原型覆盖操作标签

LLD 规则原型覆盖操作标签对象包含设置到发现对象的标签名称和值。它具有以下属性：

属性	类型	描述
tag	string	新标签名称。 属性行为: - 必填
value	string	新标签值。

LLD 规则原型覆盖操作模板

链接到已发现主机的 LLD 规则原型覆盖操作模板对象。它具有以下属性：

属性	Type	描述
templateid	ID	覆盖主机原型已链接模板的模板。 属性行为: - 必需

LLD 规则原型覆盖操作清单

设置到已发现主机的 LLD 规则原型覆盖操作清单模式值。它具有以下属性：

属性	Type	描述
inventory_mode	integer	覆盖主机原型清单模式。 可能的值： -1 - 禁用； 0 - (默认) 手动； 1 - 自动。 属性行为： - 必需

discoveryruleprototype.create

描述

`object discoveryruleprototype.create(object/array lldRules)`

此方法允许您创建新的 LLD 规则原型。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的 LLD 规则原型。

除了标准 LLD 规则原型属性之外，该方法还接受以下参数。

参数	类型	说明
filter	object	LLD 规则的LLD 规则原型过滤器。
preprocessing	object/array	LLD 规则原型预处理选项。
lld_macro_paths	object/array	LLD 规则原型lld_macro_path 选项。
overrides	object/array	LLD 规则原型覆盖选项。

返回值

`(object)` 返回一个对象，其中在 `itemids` 属性下包含已创建的 LLD 规则原型的 ID。返回的 ID 顺序与传入的 LLD 规则原型顺序一致。

示例

创建 LLD 规则原型

创建一个 LLD 规则原型（类型：嵌套），用于发现数据库实例中的表空间。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryruleprototype.create",
    "params": {
        "name": " 发现 {#DB} 的表空间",
        "key_": "db.tablespace.discovery[{#DB}]",
        "hostid": "10084",
        "ruleid": "47251",
        "type": 23
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "47252"
        ]
    },
    "id": 1
}
```

另请参见

- LLD 规则原型过滤器
- LLD 宏路径
- LLD 规则原型预处理

源代码

CDiscoveryRulePrototype::create() 位于 *ui/include/classes/api/services/CDiscoveryRulePrototype.php*。

discoveryruleprototype.delete

描述

`object discoveryruleprototype.delete(array lldRuleIds)`

此方法允许您删除 LLD 规则原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(`array`) 要删除的 LLD 规则原型的 ID。

返回值

(`object`) 返回一个对象，其中在 `ruleids` 属性下包含已删除的 LLD 规则原型的 ID。

示例

删除多个 LLD 规则原型

删除两个 LLD 规则原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryruleprototype.delete",
    "params": [
```

```
        "47252",
        "47253"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "ruleids": [
            "47252",
            "47253"
        ]
    },
    "id": 1
}
```

源代码

位于 *ui/include/classes/api/services/CDiscoveryRulePrototype.php* 中的 CDiscoveryRulePrototype::delete()。

discoveryruleprototype.get

描述

`integer/array discoveryruleprototype.get(object parameters)`

该方法允许你根据给定参数检索 LLD 规则原型。

> **Note:**
> 此方法对任何类型的用户都可用。
> 可在用户角色设置中撤销调用该方法的权限。
> 有关更多信息，请参见用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
discoveryids	ID/array	仅返回属于给定 LLD 规则或 LLD 规则原型的 LLD 规则原型。
itemids	ID/array	仅返回具有给定 ID 的 LLD 规则原型。
groupids	ID/array	仅返回属于给定组中主机的 LLD 规则原型。
hostids	ID/array	仅返回属于给定主机的 LLD 规则原型。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的 LLD 规则原型。
interfaceids	ID/array	仅返回使用给定主机接口的 LLD 规则原型。
monitored	boolean	如果设置为 `true`，则仅返回属于受监控主机的已启用 LLD 规则原型。
templated	boolean	如果设置为 `true`，则仅返回属于模板的 LLD 规则原型。
templateids	ID/array	仅返回属于给定模板的 LLD 规则原型。
selectDiscoveryRule	query	返回`discoveryRule` 属性，其中包含该 LLD 规则原型的父 LLD 规则。 如果父级是发现规则原型，则将返回空响应。
selectDiscoveryRulePrototype	query	返回`discoveryRulePrototype` 属性，其中包含该 LLD 规则原型的父 LLD 规则原型。 如果父级是发现规则（非原型），则将返回空响应。
selectDiscoveryRulePrototypes	query	返回`discoveryRulePrototypes` 属性，其中包含属于该 LLD 规则原型的子 LLD 规则原型。 支持 `count`。
selectFilter	query	返回`filter` 属性，其中包含该 LLD 规则原型所使用过滤器的数据。
selectGraphs	query	返回`graphs` 属性，其中包含属于该 LLD 规则原型的图形原型。 支持 `count`。

参数	类型	说明
selectHostPrototypes	query	返回hostPrototypes 属性，其中包含属于该 LLD 规则原型的主机原型。 支持 count。
selectHosts	query	返回hosts 属性，其中包含该 LLD 规则原型所属主机的数组。
selectItems	query	返回items 属性，其中包含属于该 LLD 规则原型的监控项原型。 支持 count。
selectTriggers	query	返回triggers 属性，其中包含属于该 LLD 规则原型的触发器原型。 支持 count。
selectLLDMacroPaths	query	返回lld_macro_paths 属性，其中包含 LLD 宏列表以及分配给每个对应宏的值路径。
selectPreprocessing	query	返回preprocessing 属性，其中包含 LLD 规则原型的预处理选项。
selectOverrides	query	返回lld_rule_overrides 属性，其中包含对原型对象执行的覆盖过滤器、条件和操作列表。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值可以是单个值，也可以是用于匹配的值数组。 不支持 text 数据类型 的属性。 支持附加属性： host - 该 LLD 规则原型所属主机的技术名称。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择：selectItems、selectGraphs、selectTriggers。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：itemid、name、key_、delay、type、status。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回已检索对象的数量。

示例

从主机检索发现规则原型

检索指定主机 ID 的所有发现规则原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryruleprototype.get",
    "params": {
        "hostids": "10084"
    },
```

```
    "id": 1
}
```

响应：


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "47253",
            "type": "23",
            "snmp_oid": "",
            "hostid": "10084",
            "name": " 发现 {#DB} 的表空间",
            "key_": "db.tablespace.discovery[{#DB}]",
            "delay": "0",
            "history": "31d",
            "trends": "365d",
            "status": "0",
            "value_type": "4",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "3",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "parameters": []
```

```
        }
    ],
    "id": 1
}
```

另请参阅

- 图形原型
- 主机
- 监控项原型
- LLD 规则原型过滤器
- 触发器原型

源代码

位于 *ui/include/classes/api/services/CDiscoveryRulePrototype.php* 中的 CDiscoveryRulePrototype::get()。

discoveryruleprototype.update

描述

`object discoveryruleprototype.update(object/array lldRules)`

此方法允许您更新现有的 LLD 规则原型。

请注意，对已发现的原型进行更新是有限制的。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

`(object/array)` 要更新的 LLD 规则原型属性。

必须为每个 LLD 规则原型定义 `itemid` 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准 LLD 规则原型属性外，该方法还接受以下参数。

参数	类型	描述
filter	object	用于替换现有过滤器的LLD 规则原型过滤器。
preprocessing	object/array	用于替换现有预处理选项的LLD 规则原型预处理 选项。 参数行为： - 对继承对象为 只读
lld_macro_paths	object/array	用于替换现有 lld_macro_path 选项的 LLD 规则原型lld_macro_path 选项。 参数行为： - 对继承对象为 只读
overrides	object/array	用于替换现有覆盖选项的LLD 规则原型覆盖 选项。 参数行为： - 对继承对象为 只读

返回值

`(object)` 返回一个对象，其中在 `itemids` 属性下包含已更新的 LLD 规则原型的 ID。

示例

更新 LLD 规则原型预处理选项

使用 JSONPath 预处理规则更新一个 LLD 规则原型。请求：

```
{
    "jsonrpc": "2.0",
    "method": "discoveryruleprototype.update",
    "params": {
```

```
        "itemid": "47253",
        "preprocessing": [
            {
                "type": 12,
                "params": "$.tablespaces",
                "error_handler": 1
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "47253"
        ]
    },
    "id": 1
}
```

源代码

位于 *ui/include/classes/api/services/CDiscoveryRulePrototype.php* 中的 CDiscoveryRulePrototype::update()。

MFA

此方法旨在与 MFA (多重身份验证) 方法配合使用。

对象参考：

- MFA

可用方法：

- mfa.create - 创建新的 MFA 方法
- mfa.delete - 删除 MFA 方法
- mfa.get - 获取 MFA 方法
- mfa.update - 更新 MFA 方法

MFA 对象

以下内容是关于 `mfa` 接口。

MFA

MFA（多因素认证）对象具有以下属性。

属性	类型	说明
mfaid	ID	MFA 方法的 ID。 属性行为： - 只读 - 更新操作时必填
type	integer	MFA 方法的类型。 可能的值： 1 - TOTP（基于时间的一次性密码）； 2 - Duo Universal Prompt。
name	string	MFA 方法的唯一名称。 属性行为： - 创建操作时必填

属性	类型	说明
hash_function	integer	用于生成 TOTP 代码的哈希函数类型。 可能的值： 1 - SHA-1； 2 - SHA-256； 3 - SHA-512。 属性行为： - 当 type 设置为 “TOTP” 时，必填
code_length	integer	验证码长度。 可能的值： 6 - 6 位长度； 8 - 8 位长度。 属性行为： - 当 type 设置为 “TOTP” 时，必填
api_hostname	string	Duo 认证服务提供的 API 主机名。 属性行为： - 当 type 设置为 “Duo Universal Prompt” 时，必填
clientid	string	Duo 认证服务提供的客户端 ID。 属性行为： - 当 type 设置为 “Duo Universal Prompt” 时，必填
client_secret	string	Duo 认证服务提供的客户端密钥。 属性行为： - 只写 - 当 type 设置为 “Duo Universal Prompt” 时，必填

创建

描述

`object mfa.create(object/array MFA methods)`

此方法允许您创建新的 MFA 方法。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` MFA 方法创建。

该方法接受具有标准 MFA 方法属性的 MFA 方法。

返回值

`(object)` 返回一个对象，其中包含在 `mfaids` 属性下创建的 MFA 方法的 ID。返回的 ID 的顺序与传递的项目的顺序匹配。

示例

创建 MFA 方法

使用基于时间的一次性密码 (TOTP) 创建”Zabbix TOTP” MFA 方法，生成 TOTP 代码的哈希函数设置为 SHA-1，验证码长度设置为 6 位数字。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mfa.create",
    "params": {
```

```
        "type": 1,
        "name": "Zabbix TOTP",
        "hash_function": 1,
        "code_length": 6
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mfaids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CMfa::create() in *ui/include/classes/api/services/CMfa.php*。

删除

描述

`object mfa.delete(array mfaids)`

此方法允许您删除 MFA 方法。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`array`) 要删除的 MFA 方法的 ID 列表。

返回值

(`object`) 返回一个对象，其中包含 `mfaids` 属性下已删除的 MFA 方法的 ID。

示例

删除 MFA 方法

删除 MFA 方法。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mfa.delete",
    "params": [
        "2"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mfaids": [
            "2"
        ]
    },
```

```
    "id": 1
}
```

来源

CMfa::delete() in *ui/include/classes/api/services/CMfa.php*。

更新

描述

`object mfa.update(object/array MFA methods)`

此方法允许您更新现有的 MFA 方法。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。在用户角色设置中可以撤销调用此方法的权限。有关更多信息，请参见User roles。

参数

(object/array) 更新 MFA 方法属性。

每个项目必须定义 `mfaid` 属性，其他属性都是可选项。仅传递的属性将被更新，所有其他属性将保持不变。

该方法接受具有标准 MFA 方法属性的 MFA 方法。

返回值

(object) 返回一个对象，其中包含 `mfaids` 属性下更新的 MFA 方法的 ID。

示例

更新方法属性

更新用于生成 TOTP 代码的哈希函数以及利用基于时间的一次性密码 (TOTP) 的”Zabbix TOTP” MFA 方法的验证码长度。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mfa.update",
    "params": {
        "mfaid": "1",
        "hash_function": 3,
        "code_length": 8
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mfaids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CMfa::update() in *ui/include/classes/api/services/CMfa.php*。

获取

描述

`integer/array mfa.get(object parameters)`

该方法允许您根据给定参数检索 MFA 方法。

> **Note:**
> 此方法仅适用于 超级管理员用户类型。可在用户角色设置中撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
mfaids	ID/array	仅返回具有给定 ID 的 MFA 方法。
selectUsrgrps	query	返回 usrgrps 属性，其中包含与 MFA 方法关联的用户组。 支持 count。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 支持以下属性： mfaid - MFA 方法的 ID； type - MFA 方法的类型。
sortfield	string/array	按给定属性对结果进行排序。 可选值：name。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，将执行 LIKE "%…%" 搜索。 支持的属性：name。
countOutput	boolean	这些参数在参考说明中进行了说明。
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回两者其中任一：

- 一组对象;
- 如果已经使用了 countOutput 参数，则检索对象的计数。

示例

按名称查找 MFA 方法

检索名称中包含”Zabbix” 的所有 MFA 方法。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "mfa.get",
    "params": {
        "output": "extend",
        "search": {
            "name": "Zabbix"
        }
    },
```

```
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mfaid": "1",
            "type": "1",
            "name": "Zabbix TOTP 1",
            "hash_function": "1",
            "code_length": "6",
            "api_hostname": "",
            "clientid": ""
        },
        {
            "mfaid": "2",
            "type": "1",
            "name": "Zabbix TOTP 2",
            "hash_function": "3",
            "code_length": "8",
            "api_hostname": "",
            "clientid": ""
        }
    ],
    "id": 1
}
```

来源

CMfa::get() in *ui/include/classes/api/services/CMfa.php*。

Proxy

此类旨在与代理配合使用。

对象引用：

- Proxy

可用方法：

- proxy.create - 创建新 Proxy
- proxy.delete - 删除 Proxy
- proxy.get - 检索 Proxy
- proxy.update - 更新 Proxy

Proxy 对象

以下对象与 proxy API 直接相关。

proxy

proxy 对象具有以下属性。

属性	类型	描述
proxyid	ID	proxy 的 ID。 属性行为： - 只读 - 更新操作中为必需

属性	类型	描述
name	string	proxy 的名称。 属性行为： - 创建操作中为必需
proxy_groupid	ID	proxy 组的 ID。 0 - proxy 未分配到任何 proxy 组。
local_address	string	用于 active agent 的地址。用于连接的 IP 地址或 DNS 名称。 属性行为： - 当 `proxy_groupid` 不为 0 时为必需
local_port	string	用于连接的本地 proxy 端口号。 支持用户宏。 默认值：10051。 属性行为： - 当 `proxy_groupid` 不为 0 时支持
operating_mode	integer	proxy 类型。 可能的值： 0 - active proxy； 1 - passive proxy。 属性行为： - 创建操作中为必需
description	text	proxy 的描述。
lastaccess	timestamp	proxy 上次连接到服务器的时间。 属性行为： - 只读
address	string	用于连接的 IP 地址或 DNS 名称。 支持用户宏。 属性行为： - 当 Zabbix proxy 的运行模式为 passive 时为必需
port	string	用于连接的端口号。 支持用户宏。 默认值：10051。 属性行为： - 当 Zabbix proxy 的运行模式为 passive 时支持
allowed_addresses	string	active Zabbix proxy 的 IP 地址或 DNS 名称，以逗号分隔。
tls_connect	integer	与 proxy 的连接。 可能的值： 1 - （默认）不加密； 2 - PSK； 4 - 证书。
tls_accept	integer	来自 proxy 的连接。 可能的位图值： 1 - （默认）不加密； 2 - PSK； 4 - 证书。 这是一个位掩码字段；可以使用任意可能位图值的总和（例如，PSK 和证书使用 6）。

属性	类型	描述
tls_issuer	string	证书颁发者。
tls_subject	string	证书主题。
tls_psk_identity	string	PSK 标识；必须仅与一个 PSK 配对（在自动注册、主机和proxy中）。 请勿在 PSK 标识中包含敏感信息，因为该标识会通过网络以未加密方式发送，用于告知接收方应使用哪个 PSK。 属性行为： - 只写 - 当 `tls_connect` 设置为”PSK”，或 `tls_accept` 包含”PSK” 位时为必需
tls_psk	string	预共享密钥（PSK）；必须至少包含 32 个十六进制数字。 属性行为： - 只写 - 当 `tls_connect` 设置为”PSK”，或 `tls_accept` 包含”PSK” 位时为必需
custom_timeouts	integer	是否在 proxy 级别覆盖全局监控项超时。 可能的值： 0 - （默认）使用全局设置； 1 - 覆盖超时。
timeout_zabbix_agent	string	处理时间不超过 `timeout_*` 秒。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m）。也接受用户宏。 仅当 `custom_timeouts` 设置为”override timeouts” 时使用；否则将忽略该值。 可能的值范围：1-600s。 属性行为： - 当 `custom_timeouts` 设置为”override timeouts” 时为必需
timeout_simple_check		
timeout_snmp_agent		
timeout_external_check		
timeout_db_monitor		
timeout_http_agent		
timeout_ssh_agent		
timeout_telnet_agent		
timeout_script		
timeout_browser		
timeout_telemetry_query		
version	integer	proxy 的版本。 由三部分组成的 Zabbix 版本号，每部分使用两位十进制数字，例如 60401 表示版本 6.4.1，70002 表示版本 7.0.2，依此类推。 0 - 未知的 proxy 版本。 属性行为： - 只读
compatibility	integer	proxy 与 Zabbix 服务器版本的比较结果。 可能的值： 0 - 未定义； 1 - 当前版本（proxy 和服务器具有相同的主版本）； 2 - 过时版本（proxy 版本低于服务器版本，但仍获得部分支持）； 3 - 不支持的版本（proxy 版本低于服务器之前的 LTS 版本，或服务器主版本低于 proxy 主版本）。 属性行为： - 只读

属性	类型	描述
state	integer	proxy 的状态。 可能的值： 0 - 未知； 1 - 离线； 2 - 在线。 属性行为： - 只读

应用性能监控（APM）

APM 对象具有以下属性。

Property	Type	Description
data_collection_status	integer	是否在 proxy 上启用遥测数据收集。 可能的值： 0 - （默认）已禁用； 1 - 已启用。
max_messages_per_second	integer	proxy 每秒接受的最大消息数。 可能的值范围：0（无限制）– 2147483647。 默认值：0。 属性行为： - 如果将 `data_collection_status` 设置为 “已启用”，则为 *supported*
additional_resource_attributes	array	要添加到传入遥测数据中的资源属性数组。 属性行为： - 如果将 `data_collection_status` 设置为 “已启用”，则为 *supported*

资源属性

资源属性对象具有以下属性。

属性	类型	描述
key	string	资源属性名称。 对于指定的 `signal_type`，必须唯一。 属性行为： - 必需
value	string	资源属性值。 可以为空。
signal_type	integer	该属性适用的遥测数据类型。 可能的值： 0 - Traces； 1 - Metrics； 2 - Logs。 属性行为： - 必需

创建

描述

`object proxy.create(object/array proxies)`

此方法允许您创建新的 proxy。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`object/array`) 要创建的 proxy。

除标准 proxy 属性外，该方法还接受以下参数。

参数	类型	描述
hosts	array	由该 proxy 监控的主机。 如果某个主机已由其他 proxy 监控，则会将其重新分配给当前 proxy。 主机只能定义 `hostid` 属性。
apm	object	proxy 的应用性能监控 (APM)设置。

返回值

(`object`) 返回一个对象，其中包含 `proxyids` 属性下创建的 Proxy 的 ID。返回的 ID 的顺序与传递的 Proxy 的顺序相匹配。

示例

创建主动式 Proxy

创建一个叫做"Active Proxy" 的主动式 Proxy，并指定一个主机由其监控。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.create",
"params": {
"name": "Active Proxy",
"operating_mode": "0",
"hosts": [{
"hostid": "10279"
}]
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": [
"10280"
]
},
"id": 1
}
```

创建被动式代理

创建一个叫做 "Passive Proxy" 的被动式代理，并指定两个主机由其监控。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.create",
"params": {
"name": "Passive Proxy",
```

```
"operating_mode": "1",
"address": "127.0.0.1",
"port": "10051",
"hosts": [{
"hostid": "10192"
},
{
"hostid": "10139"
}
]
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": [
"10284"
]
},
"id": 1
}
```

创建 Proxy 并将其添加到 Proxy 组

创建主动式 Proxy “Active proxy”，并将其添加到 ID 为 “1” 的代理组。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.create",
"params": {
"name": "Active proxy",
"proxy_groupid": "1",
"operating_mode": "0"
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": ["5"]
},
"id": 1
}
```

使用基于证书加密创建 proxy

创建一个使用基于证书的加密的主动 proxy。该 proxy 配置为监控主机”10785”。

请求 :

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "name": "Active proxy",
        "operating_mode": "0",
        "hosts": [
            {
                "hostid": "10785"
```

```
            }
        ],
        "tls_accept": "4",
        "tls_issuer": "CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com",
        "tls_subject": "CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "2"
        ]
    },
    "id": 1
}
```

创建启用 APM 的 proxy

创建一个启用 APM 的 active proxy“Active proxy”。

该 proxy 每秒最多接受 1000 条消息，为追踪添加 `deployment.environment` 资源属性，并为指标添加 `cloud.region` 资源属性。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "name": "Active proxy",
        "operating_mode": "0",
        "apm": {
            "data_collection_status": "1",
            "max_messages_per_second": "1000",
            "additional_resource_attributes": [
                {
                    "key": "deployment.environment",
                    "value": "production",
                    "signal_type": "0"
                },
                {
                    "key": "cloud.region",
                    "value": "eu-central-1",
                    "signal_type": "1"
                }
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "3"
        ]
    },
    "id": 1
}
```

另请参阅

- 主机
- proxy 组

来源

ui/include/classes/api/services/CProxy.php 中的 CProxy::create()。

删除

描述

`object proxy.delete(array proxies)`

此方法允许您删除 proxies。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的 proxy IDs。

返回值

(object) 返回一个对象，其中包含 proxyids 属性下已删除代理的 ID。

示例

删除多个 Proxy

删除两个 Proxy。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.delete",
"params": [
"10286",
"10285"
],
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": [
"10286",
"10285"
]
},
"id": 1
}
```

来源

ui/include/classes/api/services/CProxy.php 中的 CProxy::delete()。

更新

描述

`object proxy.update(object/array proxies)`

此方法允许你更新现有的 proxy。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的 proxy 属性。

必须为每个 proxy 定义 proxyid 属性，所有其他属性均为可选。仅更新传入的属性，其他属性保持不变。

除标准 proxy 属性外，该方法还接受以下参数。

参数	类型	描述
hosts	array	proxy 要监控的主机。 如果某个主机已由其他 proxy 监控，则会将其重新分配给当前 proxy。 主机只能定义 hostid 属性。
apm	object	proxy 的应用性能监控 (APM)设置。

返回值

(object) 返回一个对象，该对象包含 proxyids 属性下更新的 Proxy 的 ID。

示例

更改被 Proxy 监控的主机

更新 Proxy 以监控两个给定的主机。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.update",
"params": {
"proxyid": "10293",
"hosts": [{
"hostid": "10294"
},
{
"hostid": "10295"
},
]
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": [
"10293"
]
},
"id": 1
}
```

更改 Proxy 状态

将 Proxy 更改为活动 Proxy，并将其重命名为 “Acitve Proxy”。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.update",
"params": {
```

```
"proxyid": "10293",
"name": "Acitve Proxy",
"operating_mode": "0"
},
"id": 1
}
```

响应：

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": [
"10293"
]
},
"id": 1
}
```

将 Proxy 添加到 Proxy 组

更新 ID 为 “5” 的 Proxy 并将其添加到 ID 为 “1” 的 Proxy 组。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxy.create",
"params": {
"proxyid": "5",
"proxy_groupid": "1",
"local_address": "127.0.0.1"
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxyids": [
"5"
]
},
"id": 1
}
```

另请参阅

- 主机
- proxy 组

来源

CProxy::update() in *ui/include/classes/api/services/CProxy.php*.

获取

描述

`integer/array proxy.get(object parameters)`

此方法允许根据给定参数检索 proxy。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 仅返回用户用户组的 *Proxy access list* 所允许的 proxy。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
proxyids	ID/array	仅返回具有指定 ID 的 proxy。
proxy_groupids	ID/array	仅返回属于指定 proxy 组的 proxy。
selectApm	query	返回apm 属性，其中包含 proxy 的应用性能监控（APM）设置。
selectAssignedHosts	query	返回assignedHosts 属性，其中包含分配给 proxy 的主机。 支持 count。
selectHosts	query	返回hosts 属性，其中包含由 proxy 监控的主机。 支持 count。
selectProxyGroup	query	返回proxyGroup 属性，其中包含 proxy 组对象。
sortfield	string/array	按指定属性对结果进行排序。 可能的值：proxyid、name、operating_mode。
countOutput	boolean	这些参数在参考说明中进行了介绍。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中一个：

- 一个对象数组；
- 如果使用了 countOutput 参数，则返回检索对象的数量。

示例

检索所有 proxy

检索所有已配置的 Proxy 及其接口。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.get",
    "params": {
        "output": "extend",
        "selectInterface": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "host": "Active proxy",
            "status": "5",
```

```
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30091",
            "interface": []
        },
        {
            "host": "Passive proxy",
            "status": "6",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30092",
            "interface": {
                "interfaceid": "30109",
                "hostid": "30092",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "10051"
            }
        }
    ],
    "id": 1
}
```


另请参阅

- 主机
- proxy 组

来源

CProxy::get() in *ui/include/classes/api/services/CProxy.php*.

Proxy 组

此类用于处理 proxy 组。

对象引用：

- Proxy 组

可用方法：

- proxygroup.create - 创建新的 proxy 组
- proxygroup.delete - 删除 proxy 组
- proxygroup.get - 检索 proxy 组
- proxygroup.update - 更新 proxy 组

Proxy 组对象

以下对象与 proxygroup API 直接相关。

Proxy 组

Proxy 组对象具有以下属性。

Property	Type	Description
proxy_groupid	ID	Proxy 组的 ID。 属性行为： - 只读 - 更新操作时为必需
name	string	Proxy 组的名称。 属性行为： - 创建操作时为必需
description	text	Proxy 组的描述。
failover_delay	string	Proxy 组中的 proxy 必须与 Zabbix 服务器通信的时间段，超过此时间段仍未通信时，该 proxy 将被视为在线。 支持时间后缀（例如：30s、1m）。 支持用户宏。 可能的取值：10s-15m。 默认值：1m。
min_online	string	保持 Proxy 组在线所需的最少在线 proxy 数量。 支持用户宏。 可能的取值范围：1-1000。 默认值：1。
state	integer	Proxy 组的状态。 可能的取值： 0 - 未知； 1 - 离线； 2 - 正在恢复； 3 - 在线； 4 - 性能下降。 属性行为： - 只读

创建

描述

`object proxygroup.create(object/array proxyGroups)`

此方法允许您创建新的 proxy group。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的 proxy 组。

此方法接受具有标准 proxy 组属性的 proxy 组。

返回值

`(object)` 返回一个对象，其中包含 `proxy_groupids` 属性下创建的 Proxy 组的 ID。返回的 ID 的顺序与传递的 Proxy 组的顺序相匹配。

示例

创建 Proxy 组

使用自定义设置创建 Proxy 组。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxygroup.create",
"params": {
"name": "Proxy group",
"failover_delay": "5m",
"min_online": "10"
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxy_groupids": [
"5"
]
},
"id": 1
}
```

来源

ui/include/classes/api/services/CProxyGroup.php 中的 CProxyGroup::create()。

删除

描述

`object proxygroup.delete(array proxyGroupIds)`

此方法允许您删除 proxy 组。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的 Proxy 组的 ID。

返回值

(object) 返回一个对象，其中包含 proxy_groupids 属性下已删除的 Proxy 组的 ID。

示例

删除多个 Proxy 组

删除两个 Proxy 组。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxygroup.delete",
"params": [
"5",
"10"
],
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxy_groupids": [
"5",
"10"
]
},
"id": 1
}
```

来源

ui/include/classes/api/services/CProxyGroup.php 中的 CProxyGroup::delete()。

更新

描述

`object proxygroup.update(object/array proxyGroups)`

此方法允许您更新现有的 proxy 组。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的 proxy 组属性。

必须为每个 proxy 组定义 proxy_groupid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

该方法接受具有标准 proxy 组属性的 proxy 组。

返回值

(object) 返回一个对象，其中包含 proxy_groupids 属性下已更新的 Proxy 组的 ID。

示例

更改最小在线 Proxy

更改使得 Proxy 组在线所需的最小在线 Proxy 数。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxygroup.update",
"params": {
"proxy_groupid": "5",
"min_online": "3"
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"proxy_groupids": [
"5"
]
},
"id": 1
}
```

来源

ui/include/classes/api/services/CProxyGroup.php 中的 CProxyGroup::update()。

获取

描述

`integer/array proxygroup.get(object parameters)`

此方法允许根据给定参数检索 proxy 组。

Note:
任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。仅返回用户用户组的 *Proxy access list* 所允许的 proxy 组。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
proxy_groupids	ID/array	仅返回具有给定 ID 的 proxy 组。
proxyids	ID/array	仅返回包含给定 proxy 的 proxy 组。
selectProxies	query	返回`proxies` 属性，其中包含属于该 proxy 组的 proxy。 支持 `count`。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`proxy_groupid`、`name`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下任一值：

- 包含对象的数组；
- 如果已使用 `countOutput` 参数，则返回检索到的对象的数量。

示例

检索所有 Proxy 组

检索所有已配置的 Proxy 组（含 Proxy）。

请求:

```
{
"jsonrpc": "2.0",
"method": "proxygroup.get",
"params": {
"output": "extend",
"selectProxies": ["proxyid", "name"]
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": [{
"proxy_groupid": "1",
"name": "Proxy Group 1",
"failover_delay": "1m",
"min_online": "3",
"description": "",
"state": "1",
"proxies": [{
"proxyid": "1",
"name": "Proxy 1"
},
{
"proxyid": "2",
"name": "Proxy 2"
}
]
},
{“
proxy_groupid”:“ 2”,“ name”:“ Proxy Group 2”,“ failover_delay”:“ 10 m”,“ min_online”:“ 3”,“ description”:“”,“ state”:“ 3”,
proxyid”:“ 3”,“ name”:“ Proxy 3”
}, {“
proxyid”:“ 4”,“ name”:“ Proxy 4”
}, {“
proxyid”:“ 5”,“ name”:“ Proxy 5”
}]
}
],
“id”: 1
}
```


另请参阅

- Proxy

来源

ui/include/classes/api/services/CProxyGroup.php 中的 CProxyGroup::get()。

SLA

此类旨在处理 SLA（服务级别协议）对象，这些对象用于评估 IT 基础架构和业务服务的性能。

对象参考：

- SLA
- SLA 计划
- SLA 排除的停机时间
- SLA 服务标签

可用方法：

- sla.create - 创建新的 SLA
- sla.delete - 删除 SLA
- sla.get - 检索 SLA
- sla.getsli - 检索 SLA 的服务级别指标（SLI）数据
- sla.update - 更新 SLA

SLA 对象

以下对象与 `sla` （服务水平协议）API 直接相关。

SLA

SLA 对象具有以下属性。

属性	类型	描述
slaid	ID	SLA 的 ID。 属性行为： - 只读 - 更新操作所必需
name	string	SLA 的名称。 属性行为： - 创建操作所必需
period	integer	SLA 的报告周期。 可能的值： 0 - 每日； 1 - 每周； 2 - 每月； 3 - 每季度； 4 - 每年。 属性行为： - 创建操作所必需
slo	float	以百分比表示的最低可接受服务级别目标（Service Level Objective）。如果服务级别指标（SLI）低于该值，则 SLA 被视为处于问题/未达成状态。 可能的值：0-100（最多 4 位小数）。 属性行为： - 创建操作所必需
effective_date	integer	SLA 的生效日期。 可能的值：UTC 中的日期时间戳。
timezone	string	报告时区，例如：`Europe/London`、`UTC`。 有关支持的时区完整列表，请参见 PHP documentation。 属性行为： - 创建操作所必需
status	integer	SLA 的状态。 可能的值： 0 - （默认）已禁用的 SLA； 1 - 已启用的 SLA。
description	string	SLA 的描述。

SLA 计划

SLA 计划对象定义了关联服务预定处于正常工作状态的时间段。它具有以下属性。

属性	Type	描述
period_from	integer	循环每周时间段的开始时间（包含）。 可能的值：秒数（从星期日开始计数）。 属性行为： - 必填
period_to	integer	循环每周时间段的结束时间（不包含）。 可能的值：秒数（从星期日开始计数）。 属性行为： - 必填

SLA 排除停机时间

排除的停机对象定义了计划内停机时间，这些时间不会影响 SLI 的计算，比如正在进行的计划内维护。它具有如下属性。

属性	Type	描述
name	string	排除停机时间的名称。 属性行为： - 必填
period_from	integer	排除停机时间的开始时间（包含）。 可能的值：时间戳。 属性行为： - 必填
period_to	integer	排除停机时间的结束时间（不包含）。 可能的值：时间戳。 属性行为： - 必填

SLA 服务标签

SLA 服务标签对象将服务与 SLA 计算关联起来。它具有如下属性。

属性	Type	描述
tag	string	SLA 服务标签名称。 属性行为： - 必填
operator	integer	SLA 服务标签运算符。 可能的值： 0 - *(默认)* 等于； 2 - 包含。
value	string	SLA 服务标签值。

创建

描述

`object sla.create(object/array SLAs)`

此方法允许你创建新的 SLA 对象。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的 SLA 对象。

除了标准 SLA 属性之外，该方法还接受以下参数。

参数	类型	描述
service_tags	array	要为 SLA 创建的SLA 服务标签。 参数行为： - 必需
schedule	array	要为 SLA 创建的SLA 计划。 指定空参数将被解释为 24x7 计划。 默认值：24x7 计划。
excluded_downtimes	array	要为 SLA 创建的SLA 排除的停机时间。

返回值

（对象）返回一个包含创建的 SLA 的 ID 的对象，这些 ID 在 `slaids` 属性下。返回的 ID 的顺序与传入 SLA 的顺序相匹配。

示例

创建一个 SLA

指示创建一个 SLA 条目: * 跟踪与 SQL 引擎相关的服务的正常运行时间; * 自定义时间表，所有工作日都包括在内，但不包括星期六的最后一个小时; * 生效时间为 2022 年的最后一天; * 计划停机时间为 1 小时 15 分钟，从 7 月 4 日午夜开始; * 将启用 SLA 每周报告计算; * 最低可接受的 SLO 为 99.9995%。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "sla.create",
    "params": [
        {
            "name": " 数据库正常运行时间",
            "slo": "99.9995",
            "period": "1",
            "timezone": "America/Toronto",
            "description": " 为主要数据库引擎提供出色的正常运行时间。",
            "effective_date": 1672444800,
            "status": 1,
            "schedule": [
                {
                    "period_from": 0,
                    "period_to": 601200
                }
            ],
            "service_tags": [
                {
                    "tag": "database",
                    "operator": "0",
                    "value": "mysql"
                },
                {
                    "tag": "database",
                    "operator": "0",
                    "value": "postgresql"
                }
            ],
            "excluded_downtimes": [
                {
                    "name": " 软件版本升级发布",
                    "period_from": "1648760400",
                    "period_to": "1648764900"
                }
            ]
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

来源

CSla::create() 在 *ui/include/classes/api/services/CSla.php*。

删除

描述

`object sla.delete(array slaids)`

此方法允许您删除 SLA 条目。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的 SLA 的 ID 列表。

返回值

(object) 返回一个包含已删除的 SLA 的 ID 的对象，这些 ID 在 slaids 属性下。

示例

删除多个 SLA

删除 2 个 SLA 条目。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "sla.delete",
    "params": [
        "4",
        "5"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

来源

CSla::delete() 在 *ui/include/classes/api/services/CSla.php*。

更新

描述

`object sla.update(object/array slaids)`

此方法允许您更新现有的 SLA 条目。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的 SLA 属性。

必须为每个 SLA 定义 slaid 属性，其他所有属性均为可选。只有传递的属性会被更新，其他所有属性将保持不变。

除了标准 SLA 属性之外，该方法还接受以下参数。

参数	类型	描述
service_tags	array	用于替换当前 SLA 服务标签的SLA 服务标签。
schedule	array	用于替换当前 SLA 计划的SLA 计划。 如果将该参数指定为空，则将被解释为 24x7 计划。
excluded_downtimes	array	用于替换当前 SLA 排除停机时间的SLA 排除停机时间。

返回值

(object) 在 slaids 属性下返回一个包含已更新的 SLA ID 的对象。

示例

更新服务标签

将 ID 为 “5” 的 SLA 设置为按月间隔为与 NoSQL 相关的服务进行计算，且不更改其计划或排除的停机时间；将 SLO 设置为 95%。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "sla.update",
    "params": [
        {
            "slaid": "5",
            "name": "NoSQL 数据库引擎",
            "slo": "95",
            "period": 2,
            "service_tags": [
                {
                    "tag": "database",
                    "operator": "0",
                    "value": "redis"
                },
                {
                    "tag": "database",
                    "operator": "0",
                    "value": "mongodb"
                }
            ]
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

更改 SLA 的计划

将 ID 为 “5” 的 SLA 切换为 24x7 计划。

请求：

```
{
    "jsonrpc": "2.0",
```

```
    "method": "sla.update",
    "params": {
        "slaid": "5",
        "schedule": []
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

更改 SLA 的排除停机时间

为 ID 为 “5” 的 SLA 添加一段计划中的 4 小时 RAM 升级停机时间，时间为 2022 年 4 月 6 日，同时保留（需要重新定义）此前已存在的、计划于 7 月 4 日进行的软件升级。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "sla.update",
    "params": {
        "slaid": "5",
        "excluded_downtimes": [
            {
                "name": "Software version upgrade rollout",
                "period_from": "1648760400",
                "period_to": "1648764900"
            },
            {
                "name": "RAM upgrade",
                "period_from": "1649192400",
                "period_to": "1649206800"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

来源

CSla::update() 在 *ui/include/classes/api/services/CSla.php*。

获取

描述

`integer/array sla.get(object parameters)`

该方法允许您根据给定参数检索 SLA 对象。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
slaids	ID/array	仅返回具有给定 ID 的 SLA。
serviceids	ID/array	仅返回与指定服务匹配的 SLA。
selectSchedule	query	返回包含 SLA 计划的 `schedule` 属性。 支持 `count`。
selectExcludedDowntimes	query	返回包含 SLA 排除停机时间的 `excluded_downtimes` 属性。 支持 `count`。
selectServiceTags	query	返回包含 SLA 服务标签的 `service_tags` 属性。 支持 `count`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`slaid`、`name`、`period`、`slo`、`effective_date`、`timezone`、`status`、`description`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回条目：

- 一个数组对象；
- 如果使用了 `countOutput` 参数，则返回检索到的对象数量。

示例

检索所有 SLA

检索所有 SLA 及其属性的全部数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "sla.get",
    "params": {
        "output": "extend",
        "selectSchedule": ["period_from", "period_to"],
        "selectExcludedDowntimes": ["name", "period_from", "period_to"],
        "selectServiceTags": ["tag", "operator", "value"],
```

```
        "preservekeys": true
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "1": {
            "slaid": "1",
            "name": "Database Uptime",
            "period": "1",
            "slo": "99.9995",
            "effective_date": "1672444800",
            "timezone": "America/Toronto",
            "status": "1",
            "description": " 为主要 SQL 数据库引擎提供出色的正常运行时间。",
            "service_tags": [
                {
                    "tag": "database",
                    "operator": "0",
                    "value": "mysql"
                },
                {
                    "tag": "database",
                    "operator": "0",
                    "value": "postgresql"
                }
            ],
            "schedule": [
                {
                    "period_from": "0",
                    "period_to": "601200"
                }
            ],
            "excluded_downtimes": [
                {
                    "name": " 软件版本升级发布",
                    "period_from": "1648760400",
                    "period_to": "1648764900"
                }
            ]
        }
    },
    "id": 1
}
```

来源

CSla:get() 在 *ui/include/classes/api/services/CSla.php*。

获取 **SLI**

描述

`object sla.getsli(object parameters)`

此方法允许您计算服务级别协议（SLA）的服务级别指标（SLI）数据。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 包含 SLA ID、报告周期以及可选的服务 ID 的参数——用于计算 SLI。

参数	类型	说明
slaid	ID	要返回可用性信息的 SLA 的 ID。 参数行为： - 必需
period_from	timestamp	要报告 SLI 的起始时间戳（含）。 可能的值：Unix 时间戳。
period_to	timestamp	要报告 SLI 的结束时间戳（含）。 可能的值：Unix 时间戳。
periods	integer	要报告的周期数。 可能的值：1-100
serviceids	ID/array	要返回 SLI 的服务 ID。

周期的划分

下表展示了根据参数组合返回的周期切片排列方式。

Note:
返回的周期不会早于第一个可用周期（基于 SLA 的生效日期），也不会超过当前周期。

参数			返回的周期
period_from	**period_to**	**periods**	
-	-	-	包括当前周期在内的最近 20 个周期。
-	-	N	最近 N 个周期。
-	指定	-	period_to 之前最近的 20 个周期。
-	指定	N	period_to 之前最近的 N 个周期。
指定	-	-	从 period_from 开始的前 20 个周期。
指定	-	N	从 period_from 开始的前 N 个周期。
指定	指定	-	指定范围内最多 100 个周期。
指定	指定	N	指定范围内的 N 个周期。

返回值

(object) 返回计算结果。

属性	Type	描述
periods	array	已报告周期的列表。 每个已报告周期表示为一个对象，包含： - period_from - (timestamp) 已报告周期的起始时间戳（包含）。 - period_to - (timestamp) 已报告周期的结束时间戳（不包含）。 各周期按 period_from 排序，最早的周期排在最前面。
serviceids	array	已报告周期中的服务 ID 列表。 该列表的排序顺序未定义。即使已将 serviceids 参数传递给 sla.getsli 方法也是如此。
sli	array	每个已报告周期和服务的 SLI 数据（以二维数组形式表示）。 periods 属性的索引用作 sli 属性的第一维。 serviceids 属性的索引用作 sli 属性的第二维。

SLI 数据

每个报告周期和服务返回的 SLI 数据包括：

属性	类型	说明
uptime	integer	在计划运行时间内，服务处于 *OK* 状态的时长，减去被排除的停机时间。
downtime	integer	在计划运行时间内，服务处于 *not OK* 状态的时长，减去被排除的停机时间。
sli	float	SLI（总运行时间的百分比），基于 uptime 和 downtime 计算。
error_budget	integer	错误预算（以秒为单位），基于 SLI 和 SLO 计算。
excluded_downtimes	array	此报告周期内被排除的停机时间数组。 每个对象将包含以下参数： - `name` - (`string`) 被排除停机时间的名称。 - `period_from` - (`timestamp`) 被排除停机时间的起始时间戳（包含）。 - `period_to` - (`timestamp`) 被排除停机时间的结束时间戳（不包含）。 被排除的停机时间按 `period_from` 排序，最早的时间段排在最前面。

示例

计算每日 SLA 的 SLI

检索与 ID 为 “1” 的 SLA 关联的、ID 为 “1” 和 “4” 的服务的 SLI 数据。检索截至 “1761861599”（2025 年 10 月 30 日 23:59:59 GMT+0200）的单个周期数据。由于该 SLA 的报告周期为每日，因此 SLI 数据将从 “1761775200”（2025 年 10 月 30 日 00:00:00 GMT+0200）检索到 “1761861600”（2025 年 10 月 31 日 00:00:00 GMT+0200）。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "sla.getsli",
    "params": {
        "slaid": "1",
        "serviceids": [
            1,
            4
        ],
        "periods": 1,
        "period_to": 1761861599
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "periods": [
            {
                "period_from": 1761775200,
                "period_to": 1761861600
            }
        ],
        "serviceids": [
            1,
            4
        ],
        "sli": [
            [
                {
                    "uptime": 43843,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
```

```
                            "name": "Maintenance OCT",
                            "period_from": 1761825600,
                            "period_to": 1761829200
                        }
                    ]
                },
                {
                    "uptime": 32225,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": []
                }
            ]
        ]
    },
    "id": 1
}
```

计算月度 SLA 的 SLI

检索与 ID 为 “5” 的 SLA 关联的、ID 为 “50”、“60” 和 “70” 的服务的 SLI 数据。从 “1635724800”（2021 年 11 月 01 日 00:00:00 UTC）开始，检索三个周期的数据。由于该 SLA 的报告周期为按月，因此会检索以下三个月的 SLI 数据：

- 从 “1635724800”（2021 年 11 月 01 日 00:00:00 UTC）到 “1638316800”（2021 年 12 月 01 日 00:00:00 UTC）
- 从 “1638316800”（2021 年 12 月 01 日 00:00:00 UTC）到 “1640995200”（2022 年 01 月 01 日 00:00:00 UTC）
- 从 “1640995200”（2022 年 01 月 01 日 00:00:00 UTC）到 “1643673600”（2022 年 02 月 01 日 00:00:00 UTC）

请求：

```
{
    "jsonrpc": "2.0",
    "method": "sla.getsli",
    "params": {
        "slaid": "5",
        "serviceids": [
            50,
            60,
            70
        ],
        "periods": 3,
        "period_from": 1635724800
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "periods": [
            {
                "period_from": 1635724800,
                "period_to": 1638316800
            },
            {
                "period_from": 1638316800,
                "period_to": 1640995200
            },
            {
                "period_from": 1640995200,
                "period_to": 1643673600
            }
        ],
        "serviceids": [
```

```
            50,
            60,
            70
        ],
        "sli": [
            [
                {
                    "uptime": 1186212,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Maintenance Nov 25 - Dec 01",
                            "period_from": 1637836212,
                            "period_to": 1638316800
                        }
                    ]
                },
                {
                    "uptime": 1186212,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Maintenance Nov 25 - Dec 01",
                            "period_from": 1637836212,
                            "period_to": 1638316800
                        }
                    ]
                },
                {
                    "uptime": 1186212,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Maintenance Nov 25 - Dec 01",
                            "period_from": 1637836212,
                            "period_to": 1638316800
                        }
                    ]
                }
            ],
            [
                {
                    "uptime": 1147548,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Maintenance Dec 02 - Dec 10",
                            "period_from": 1638439200,
                            "period_to": 1639109652
                        }
                    ]
                },
                {
                    "uptime": 1147548,
```

```
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Maintenance Dec 02 - Dec 10",
                            "period_from": 1638439200,
                            "period_to": 1639109652
                        }
                    ]
                },
                {
                    "uptime": 1147548,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Maintenance Dec 02 - Dec 10",
                            "period_from": 1638439200,
                            "period_to": 1639109652
                        }
                    ]
                }
            ],
            [
                {
                    "uptime": 1674000,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": []
                },
                {
                    "uptime": 1674000,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": []
                },
                {
                    "uptime": 1674000,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": []
                }
            ]
        ]
    },
    "id": 1
}
```


来源

CSla::getSli() 在 *ui/include/classes/api/services/CSla.php* 。

Token

此类用于处理令牌。

对象引用：

- Token

可用方法：

- token.create - 创建新令牌
- token.delete - 删除令牌
- token.get - 检索令牌
- token.update - 更新令牌
- token.generate - 生成令牌

令牌对象

以下对象与 令牌 API 直接相关。

Token

token 对象具有以下属性。

属性	类型	描述
tokenid	ID	token 的 ID。 属性行为： - 只读 - 更新操作时必填
name	string	token 的名称。 属性行为： - 创建操作时必填
description	text	token 的描述。
userid	ID	分配给该 token 的用户 ID。 默认值：当前用户。 属性行为： - 常量
lastaccess	timestamp	token 最近一次通过身份验证的日期和时间。 如果 token 从未通过身份验证，则为”0”。 属性行为： - 只读
status	integer	token 状态。 可能的值： 0 - （默认）已启用的 token； 1 - 已禁用的 token。
expires_at	timestamp	token 的过期日期和时间。 对于永不过期的 token，该值为”0”。
created_at	timestamp	token 的创建日期和时间。 属性行为： - 只读
creator_userid	ID	创建该 token 的用户 ID。 属性行为： - 只读

创建

描述

`object token.create(object/array tokens)`

此方法允许您创建新的 token。

Note:
用户角色需要具备 *Manage API tokens* 权限，才能管理其他用户的 token。

Attention:
通过此方法创建的 token 在可用之前，还必须先进行generated。

参数

(object/array) 要创建的令牌。

此方法接受令牌带有规范的令牌属性standard token properties。

返回值

(object) 返回一个对象其中包含在 tokenids 属性下创建的令牌的 ID。返回的 ID 的顺序与传递的令牌的顺序相匹配。

示例

创建令牌

创建一个已启用且永不过期的令牌，并对 ID 为 2 的用户进行身份验证。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "token.create",
    "params": {
        "name": "Your token",
        "userid": "2"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "188"
        ]
    },
    "id": 1
}
```

创建一个已禁用且于 2021 年 1 月 21 日过期的令牌。此令牌将对当前用户进行身份验证。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "token.create",
    "params": {
        "name": "Your token",
        "status": "1",
        "expires_at": "1611238072"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "189"
        ]
```

```
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CToken.php 中的 CToken::create()。

删除

描述

`object token.delete(array tokenids)`

此方法允许您删除令牌。

> **Note:**
> 用户角色需要具备 *Manage API tokens* 权限，才能为其他用户管理令牌。

参数

(array) 要删除的令牌的 ID。

返回值

(object) 返回一个对象，其中包含 tokenids 属性下已删除令牌的 ID。

示例

删除多个令牌

删除两个令牌。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "token.delete",
    "params": [
        "188",
        "192"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "188",
            "192"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CToken.php 中的 CToken::delete()。

更新

描述

`object token.update(object/array tokens)`

此方法允许您更新现有的 token。

Note:
用户角色需要具备 *Manage API tokens* 权限，才能为其他用户管理 token。

参数

(object/array) 要更新的令牌属性。

必须为每个令牌定义 tokenid 属性，所有其他属性都是可选的。只有被传递的属性会被更新，所有其他的将保持不变。

此方法接受具有标准的令牌属性的的令牌。

返回值

(object) 返回一个对象，其中包含 tokenids 属性下已被更新的令牌的 ID。

示例

移除令牌过期时间

从令牌中移除过期日期。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "token.update",
    "params": {
        "tokenid": "2",
        "expires_at": "0"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "2"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CToken.php 中的 CToken::update()。

生成

描述

`object token.generate(array tokenids)`

此方法允许您生成令牌。

Note:
用户角色需要具备 *Manage API tokens* 权限，才能为其他用户管理令牌。

Attention:
只有在令牌已被创建 的情况下，才能通过此方法生成该令牌。

参数

(array) 要生成的令牌的 ID。

返回值

(array) 返回一个对象数组，其中在 tokenid 属性下包含生成的令牌 ID，在 token 属性下包含生成的授权字符串。

属性	Type	描述
tokenid	ID	令牌的 ID。
token	string	为此令牌生成的授权字符串。

示例

生成多个令牌

生成两个令牌。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "token.generate",
    "params": [
        "1",
        "2"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "tokenid": "1",
            "token": "bbcfce79a2d95037502f7e9a534906d3466c9a1484beb6ea0f4e7be28e8b8ce2"
        },
        {
            "tokenid": "2",
            "token": "fa1258a83d518eabd87698a96bd7f07e5a6ae8aeb8463cae33d50b91dd21bd6d"
        }
    ],
    "id": 1
}
```

来源

ui/include/classes/api/services/CToken.php 中的 CToken::generate()。

获取

描述

`integer/array token.get(object parameters)`

该方法允许您根据给定参数检索令牌。

> **Note:**
> 只有 超级管理员用户类型才允许查看其他用户的令牌。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
tokenids	ID/array	仅返回具有给定 ID 的令牌。
userids	ID/array	仅返回为给定用户创建的令牌。
token	string	仅返回为给定的 *Auth token* 创建的令牌。
valid_at	timestamp	仅返回在给定日期和时间有效（未过期）的令牌。
expired_at	timestamp	仅返回在给定日期和时间已过期（无效）的令牌。

参数	类型	描述
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`tokenid`、`name`、`lastaccess`、`status`、`expires_at`、`created_at`。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中之一：

- 对象数组；
- 如果使用了 countOutput 参数，则检索到对象的数量。

示例

检索令牌

检索 ID 为 “2” 的令牌的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "token.get",
    "params": {
        "output": "extend",
        "tokenids": "2"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "tokenid": "1",
            "name": "The Token",
            "description": "",
            "userid": "1",
            "lastaccess": "0",
            "status": "0",
            "expires_at": "1609406220",
            "created_at": "1611239454",
            "creator_userid": "1"
        }
    ],
    "id": 1
}
```

来源

ui/include/classes/api/services/CToken.php 中的 CToken::get()。

Web 场景

此类用于处理 Web 场景。

对象参考：

- Web 场景
- 场景步骤
  - HTTP 字段
- Web 场景标签

可用方法：

- httptest.create - 创建新的 Web 场景
- httptest.delete - 删除 Web 场景
- httptest.get - 检索 Web 场景
- httptest.update - 更新 Web 场景

Web 场景对象

以下对象与 `webcheck` API 直接相关。

Web 场景

Web 场景对象具有以下属性。

Property	Type	Description
httptestid	ID	Web 场景的 ID。 Property behavior： - 只读 - 更新操作时必需
hostid	ID	Web 场景所属主机的 ID。 Property behavior： - 常量 - 创建操作时必需
name	string	Web 场景的名称。 Property behavior： - 创建操作时必需
agent	string	Web 场景使用的 User-Agent 字符串。 默认值：Zabbix
authentication	integer	Web 场景使用的身份验证方法。 可能的值： 0 - （默认）无； 1 - 基本 HTTP 身份验证； 2 - NTLM 身份验证； 3 - Kerberos 身份验证； 4 - 摘要身份验证。
delay	string	Web 场景的执行间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或用户宏。 默认值：1m。
headers	array	执行请求时发送的HTTP 标头。
http_password	string	用于基本 HTTP、NTLM、Kerberos 或摘要身份验证的密码。
http_proxy	string	Web 场景使用的 proxy，格式为 *http://[username[:password]@]proxy.example.com[:port]*。
http_user	string	用于基本 HTTP、NTLM、Kerberos 或摘要身份验证的用户名。

Property	Type	Description
retries	integer	Web 场景在失败前尝试执行每个步骤的次数。 默认值：1。
ssl_cert_file	string	用于客户端身份验证的 SSL 证书文件名称（必须为 PEM 格式）。
ssl_key_file	string	用于客户端身份验证的 SSL 私钥文件名称（必须为 PEM 格式）。
ssl_key_password	string	SSL 私钥密码。
status	integer	Web 场景是否启用。 可能的值： 0 - （默认）启用； 1 - 禁用。
templateid	ID	父模板 Web 场景的 ID。 Property behavior： - 只读
variables	array	Web 场景变量。
verify_host	integer	是否验证连接的主机名与主机证书中的名称匹配。 可能的值： 0 - （默认）跳过主机验证； 1 - 验证主机。
verify_peer	integer	是否验证主机证书的真实性。 可能的值： 0 - （默认）跳过对等方验证； 1 - 验证对等方。
uuid	string	全局唯一标识符，用于将导入的 Web 场景关联到已存在的 Web 场景。如果未提供，则自动生成。 Property behavior： - 如果 Web 场景属于模板，则支持

场景步骤

场景步骤对象定义特定的 Web 场景检查。它具有以下属性。

Property	Type	Description
name	string	场景步骤的名称。 属性行为： - 必需
no	integer	步骤在 Web 场景中的序号。 属性行为： - 必需
url	string	要检查的 URL。 属性行为： - 必需
follow_redirects	integer	是否遵循 HTTP 重定向。 可能的值： 0 - 不遵循重定向； 1 - (默认) 遵循重定向。
headers	array	执行请求时将发送的HTTP 标头。场景步骤标头将覆盖为 Web 场景指定的标头。
posts	string/array	以字符串（原始 POST 数据）或HTTP 字段 数组（表单字段数据）形式表示的 HTTP POST 变量。
required	string	响应中必须存在的文本。

Property	Type	Description
retrieve_mode	integer	场景步骤必须获取的 HTTP 响应部分。 可能的值： 0 - (默认) 仅正文； 1 - 仅标头； 2 - 标头和正文。
status_codes	string	所需 HTTP 状态码的范围，以逗号分隔。
timeout	string	请求超时时间（以秒为单位）。接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、1h）或用户宏。 默认值：15s。最大值：1h。最小值：1s。
variables	array	场景步骤变量。
query_fields	array	查询字段 - 执行请求时将添加到 URL 的HTTP 字段数组。

HTTP 字段

HTTP 字段对象定义了用于指定 web 场景变量、HTTP 标头以及 POST 字段或查询字段的名称和值。
它具有以下属性。

属性	类型	描述
name	string	标头/变量/POST 或 GET 字段的名称。 属性行为: - 必填
value	string	标头/变量/POST 或 GET 字段的值。 属性行为: - 必填

Web 场景标签

Web 场景标签对象具有以下属性。

属性	类型	描述
tag	string	Web 场景标签名称。 属性行为： - 必填
value	string	Web 场景标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板； 1 - 主机。 属性行为： - 只读。
objectid	ID	标签继承自的对象 ID。 属性行为： - 只读。

创建

描述

```
object httptest.create(object/array webScenarios)
```

此方法允许您创建新的 web 场景。

Note:
创建 web 场景时，会自动创建一组web 监控项。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。可在用户角色设置中撤销调用此方法的权限。有关更多信息，请参见用户角色。

参数

(object/array) 要创建的 Web 场景。

除了标准 Web 场景属性之外，该方法还接受以下参数。

参数	Type	说明
steps	array	场景步骤。 参数行为： - 必需
tags	array	Web 场景标签。

返回值

(object) 返回一个对象，其中包含 httptestids 属性下创建的 Web 场景的 ID。返回的 ID 的顺序与传递的 Web 场景的顺序相匹配。

示例

创建一个 Web 场景

创建一个 Web 场景用于监控公司主页。此场景具有两个步骤，检查主页和 “关于” 页，并保证它们返回 HTTP 状态码 200。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.create",
    "params": {
        "name": "Homepage check",
        "hostid": "10085",
        "steps": [
            {
                "name": "Homepage",
                "url": "http://example.com",
                "status_codes": "200",
                "no": 1
            },
            {
                "name": "Homepage / About",
                "url": "http://example.com/about",
                "status_codes": "200",
                "no": 2
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ]
```

```
    },
    "id": 1
}
```

另请参阅

- 场景步骤

源码

ui/include/classes/api/services/CHttpTest.php 中的 CHttpTest::create()。

删除

描述

`object httptest.delete(array webScenarioIds)`

此方法允许您删除 web 场景。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 需要被删除的 Web 场景的 ID。

返回值

(object) 返回一个包含被删除的 Web 场景的 ID 的对象，ID 在 `httptestids` 属性下。

示例

删除多个 Web 场景

删除两个 Web 场景。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "2",
            "3"
        ]
    },
    "id": 1
}
```

源码

ui/include/classes/api/services/CHttpTest.php 中的 CHttpTest::delete()。

更新

描述

`object httptest.update(object/array webScenarios)`

此方法允许您更新现有的 web 场景。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`object/array`) 要更新的 Web 场景属性。

必须为每个 Web 场景定义 `httptestid` 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准 Web 场景属性之外，该方法还接受以下参数。

参数	类型	描述
steps	array	用于替换现有步骤的场景步骤。
tags	array	Web 场景标签。

返回值

(`object`) 返回一个对象，其中包含 `httptestid` 属性下已更新的 Web 场景的 ID。

示例

启用一个 Web 场景

启动一个 Web 场景，即将其状态设置为 “0”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.update",
    "params": {
        "httptestid": "5",
        "status": 0
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ]
    },
    "id": 1
}
```

另请参见

- 场景步骤

源码

ui/include/classes/api/services/CHttpTest.php 中的 CHttpTest::update()。

获取

描述

`integer/array httptest.get(object parameters)`

该方法允许您根据给定参数检索 web 场景。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
groupids	ID/array	仅返回属于给定主机组的 Web 场景。
hostids	ID/array	仅返回属于给定主机的 Web 场景。
httptestids	ID/array	仅返回具有给定 ID 的 Web 场景。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的 Web 场景。
inheritedTags	boolean	返回在 template/host/linked templates 中也具有给定 `tags` 的 Web 场景。 可能的值： `true` - template/host/linked templates 也必须具有给定标签； `false` - *(默认)* 忽略来自 template/host/linked templates 的标签。
monitored	boolean	如果设置为 `true`，则仅返回属于受监控主机的已启用 Web 场景。
selectInheritedTags	query	返回`inheritedTags` 属性，其中包含位于 template/host/linked templates 上的标签。
templated	boolean	如果设置为 `true`，则仅返回属于模板的 Web 场景。
templateids	ID/array	仅返回属于给定模板的 Web 场景。
expandName	flag	展开 Web 场景名称中的宏。
expandStepName	flag	展开场景步骤名称中的宏。
evaltype	integer	标签求值方法。 可能的值： 0 - *(默认)* And/Or； 2 - Or。
tags	array	仅返回具有给定标签的 Web 场景。 格式：`[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有 Web 场景。 可能的operator 值： 0 - *(默认)* 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
selectHosts	query	以数组形式在`hosts` 属性中返回该 Web 场景所属的主机。
selectSteps	query	在`steps` 属性中返回 Web 场景步骤。 支持 `count`。
selectTags	query	在`tags` 属性中返回 Web 场景标签。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`httptestid`、`name`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	

参数	类型	描述
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

检索一个 web 场景

检索 web 场景 “9” 的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "httptest.get",
    "params": {
        "output": "extend",
        "selectSteps": "extend",
        "httptestids": "9"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "httptestid": "9",
            "name": "Homepage check",
            "delay": "1m",
            "status": "0",
            "agent": "Zabbix",
            "authentication": "0",
            "http_user": "",
            "http_password": "",
            "hostid": "10084",
            "templateid": "0",
            "http_proxy": "",
            "retries": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "uuid": "",
            "headers": [],
            "variables": [],
            "steps": [
                {
                    "httpstepid": "36",
                    "httptestid": "9",
                    "name": "Homepage",
```

```
                    "no": "1",
                    "url": "http://example.com",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "post_type": "1",
                    "headers": [],
                    "variables": [
                        {
                            "name":"{var}",
                            "value":"12"
                        }
                    ],
                    "query_fields": []
                },
                {
                    "httpstepid": "37",
                    "httptestid": "9",
                    "name": "Homepage / About",
                    "no": "2",
                    "url": "http://example.com/about",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "post_type": "1",
                    "headers": [],
                    "variables": [],
                    "query_fields": []
                }
            ]
        }
    ],
    "id": 1
}
```


另请参阅

- 主机
- 场景步骤

源码

ui/include/classes/api/services/CHttpTest.php 中的 CHttpTest::get()。

主机

此类用于处理主机。

对象引用：

- Host
- Host inventory
- Host tag

可用方法：

- host.create - 创建新主机
- host.delete - 删除主机
- host.get - 获取主机
- host.massadd - 向主机添加相关对象

- host.massremove - 从主机移除相关对象
- host.update - 更新主机

主机对象

以下对象于主机 API 直接相关。

主机

主机对象具有以下属性。

属性	类型	描述
hostid	ID	主机的 ID。 属性行为： - 只读 - 更新操作时必需
host	string	主机的技术名称。 属性行为： - 创建操作时必需
description	text	主机的描述。
flags	integer	主机的来源。 可能的值： 0 - 普通主机； 4 - 从原型转换而来的主机。 属性行为： - 只读
inventory_mode	integer	主机资产清单填充模式。 可能的值： -1 - （默认）禁用； 0 - 手动； 1 - 自动。
ipmi_authtype	integer	IPMI 身份验证算法。 可能的值： -1 - （默认）默认； 0 - 无； 1 - MD2； 2 - MD5 4 - 直连； 5 - OEM； 6 - RMCP+。
ipmi_password	string	IPMI 密码。
ipmi_privilege	integer	IPMI 权限级别。 可能的值： 1 - 回调； 2 - （默认）用户； 3 - 操作员； 4 - 管理员； 5 - OEM。
ipmi_username	string	IPMI 用户名。
maintenance_from	timestamp	当前生效维护的开始时间。 属性行为： - 只读

属性	类型	描述
maintenance_status	integer	当前生效的维护状态。 可能的值： 0 - （默认）无维护； 1 - 维护生效。 属性行为： - 只读
maintenance_type	integer	当前生效的维护类型。 可能的值： 0 - （默认）带数据采集的维护； 1 - 不带数据采集的维护。 属性行为： - 只读
maintenanceid	ID	当前在主机上生效的维护的 ID。 属性行为： - 只读
name	string	主机的可见名称。 默认值：host 属性值。
monitored_by	integer	用于监控主机的来源。 可能的值： 0 - （默认）Zabbix 服务器； 1 - proxy； 2 - proxy 组。 属性行为： - 如果select_server_for_monitoring 设置为 “禁用”，则必需。
proxyid	ID	用于监控主机的 proxy 的 ID。 属性行为： - 如果 monitored_by 设置为 “Proxy”，则必需
proxy_groupid	ID	用于监控主机的 proxy 组的 ID。 属性行为： - 如果 monitored_by 设置为 “Proxy group”，则必需
status	integer	主机的状态和功能。 可能的值： 0 - （默认）启用； 1 - 禁用。
tls_connect	integer	与主机的连接。 可能的值： 1 - （默认）不加密； 2 - PSK； 4 - 证书。
tls_accept	integer	来自主机的连接。 可能的位图值： 1 - （默认）不加密； 2 - PSK； 4 - 证书。 这是一个位掩码字段；可以使用任意可能位图值的总和（例如，PSK 和证书为 6）。
tls_issuer	string	证书颁发者。

属性	类型	描述
tls_subject	string	证书主题。
tls_psk_identity	string	PSK 身份；必须仅与一个 PSK 配对（在自动注册、主机和proxy中）。 请勿在 PSK 身份中包含敏感信息，因为该身份会通过未加密的网络发送，以告知接收方应使用哪个 PSK。 属性行为： - 只写 - 如果 `tls_connect` 设置为 “PSK”，或 `tls_accept` 包含 “PSK” 位，则必需
tls_psk	string	预共享密钥（PSK）；必须至少包含 32 个十六进制数字。 属性行为： - 只写 - 如果 `tls_connect` 设置为 “PSK”，或 `tls_accept` 包含 “PSK” 位，则必需
active_available	integer	主机 active 接口的可用性状态。 可能的值： 0 - 接口状态未知； 1 - 接口可用； 2 - 接口不可用。 属性行为： - 只读
assigned_proxyid	ID	如果主机由 proxy 组监控，则为 Zabbix 服务器分配的 proxy 的 ID。 属性行为： - 只读

主机资产记录

主机资产记录对象具有以下属性。

Note:
每个属性都有其唯一的 ID 编号，该编号用于将主机资产记录字段与监控项关联。

ID	属性	Type	描述	最大长度
4	alias	string	别名。	128 个字符
11	asset_tag	string	资产标签。	64 个字符
28	chassis	string	机箱。	64 个字符
23	contact	string	联系人。	SQL 数据库中为 65535 个字符
32	contract_number	string	合同编号。	64 个字符
47	date_hw_decomm	string	硬件退役日期。	64 个字符
46	date_hw_expiry	string	硬件维护到期日期。	64 个字符
45	date_hw_install	string	硬件安装日期。	64 个字符
44	date_hw_purchase	string	硬件采购日期。	64 个字符
34	deployment_status	string	部署状态。	64 个字符
14	hardware	string	硬件。	255 个字符
15	hardware_full	string	详细硬件信息。	SQL 数据库中为 65535 个字符
39	host_netmask	string	主机子网掩码。	39 个字符
38	host_networks	string	主机网络。	SQL 数据库中为 65535 个字符
40	host_router	string	主机路由器。	39 个字符
30	hw_arch	string	硬件架构。	32 个字符
33	installer_name	string	安装人员姓名。	64 个字符
24	location	string	位置。	SQL 数据库中为 65535 个字符
25	location_lat	string	位置纬度。	16 个字符
26	location_lon	string	位置经度。	16 个字符
12	macaddress_a	string	MAC 地址 A。	64 个字符
13	macaddress_b	string	MAC 地址 B。	64 个字符

ID	属性	Type	描述	最大长度
29	model	string	型号。	64 个字符
3	name	string	名称。	128 个字符
27	notes	string	备注。	SQL 数据库中为 65535 个字符
41	oob_ip	string	OOB IP 地址。	39 个字符
42	oob_netmask	string	OOB 主机子网掩码。	39 个字符
43	oob_router	string	OOB 路由器。	39 个字符
5	os	string	操作系统名称。	128 个字符
6	os_full	string	详细操作系统名称。	255 个字符
7	os_short	string	操作系统简称。	128 个字符
61	poc_1_cell	string	主要 POC 手机号码。	64 个字符
58	poc_1_email	string	主要电子邮件地址。	128 个字符
57	poc_1_name	string	主要 POC 姓名。	128 个字符
63	poc_1_notes	string	主要 POC 备注。	SQL 数据库中为 65535 个字符
59	poc_1_phone_a	string	主要 POC 电话 A。	64 个字符
60	poc_1_phone_b	string	主要 POC 电话 B。	64 个字符
62	poc_1_screen	string	主要 POC 屏幕名称。	64 个字符
68	poc_2_cell	string	次要 POC 手机号码。	64 个字符
65	poc_2_email	string	次要 POC 电子邮件地址。	128 个字符
64	poc_2_name	string	次要 POC 姓名。	128 个字符
70	poc_2_notes	string	次要 POC 备注。	SQL 数据库中为 65535 个字符
66	poc_2_phone_a	string	次要 POC 电话 A。	64 个字符
67	poc_2_phone_b	string	次要 POC 电话 B。	64 个字符
69	poc_2_screen	string	次要 POC 屏幕名称。	64 个字符
8	serialno_a	string	序列号 A。	64 个字符
9	serialno_b	string	序列号 B。	64 个字符
48	site_address_a	string	站点地址 A。	128 个字符
49	site_address_b	string	站点地址 B。	128 个字符
50	site_address_c	string	站点地址 C。	128 个字符
51	site_city	string	站点城市。	128 个字符
53	site_country	string	站点国家。	64 个字符
56	site_notes	string	站点备注。	SQL 数据库中为 65535 个字符
55	site_rack	string	站点机架位置。	128 个字符
52	site_state	string	站点州/省。	64 个字符
54	site_zip	string	站点 ZIP/邮政编码。	64 个字符
16	software	string	软件。	255 个字符
18	software_app_a	string	软件应用程序 A。	64 个字符
19	software_app_b	string	软件应用程序 B。	64 个字符
20	software_app_c	string	软件应用程序 C。	64 个字符
21	software_app_d	string	软件应用程序 D。	64 个字符
22	software_app_e	string	软件应用程序 E。	64 个字符
17	software_full	string	软件详细信息。	SQL 数据库中为 65535 个字符
10	tag	string	标签。	64 个字符
1	type	string	类型。	64 个字符
2	type_full	string	类型详细信息。	64 个字符
35	url_a	string	URL A。	2048 个字符
36	url_b	string	URL B。	2048 个字符
37	url_c	string	URL C。	2048 个字符
31	vendor	string	供应商。	64 个字符

主机标签

主机标签对象具有以下属性。

属性	Type	描述
tag	string	主机标签名称。 属性行为： - 必填

属性	Type	描述
value	string	主机标签值。 属性行为： - 只读。
automatic	integer	主机标签的类型。 可能的值： 0 - (默认) 手动（用户创建的标签）； 1 - 自动（由低级别发现创建的标签） 属性行为： - 只读
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板。 属性行为： - 只读。
objectid	ID	标签继承自的对象的 ID。 属性行为： - 只读。

创建

描述

`object host.create(object/array hosts)`

此方法允许您创建新的主机。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的主机。

除了标准主机属性外，该方法还接受以下参数。

参数	类型	描述
groups	object/array	要将主机添加到的主机组。 主机组必须只定义 `groupid` 属性。 参数行为： - 必需
interfaces	object/array	要为主机创建的接口。
tags	object/array	主机标签。
templates	object/array	要链接到主机的模板。 模板必须只定义 `templateid` 属性。
macros	object/array	要为主机创建的用户宏。
inventory	object	主机资产记录属性。

返回值

`(object)` 返回已创建主机的 `hostids` 属性。返回的 ID 顺序与传入的主机顺序一致。

示例

创建主机

创建一个名为 “Linux server” 的主机，为其配置 IP 接口和标签，将其添加到一个组，为其链接一个模板，并在主机资产清单中设置 MAC 地址。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "50"
            }
        ],
        "tags": [
            {
                "tag": "host-name",
                "value": "linux-server"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
        "macros": [
            {
                "macro": "{$USER_ID}",
                "value": "123321"
            },
            {
                "macro": "{$USER_LOCATION}",
                "value": "0:0:0",
                "description": "latitude, longitude and altitude coordinates"
            }
        ],
        "inventory_mode": 0,
        "inventory": {
            "macaddress_a": "01234",
            "macaddress_b": "56768"
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "107819"
```

```
        ]
    },
    "id": 1
}
```

创建具有 SNMP 接口的主机

创建一个名为"SNMP host" 的主机，并包含 SNMPv3 接口。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "SNMP host",
        "interfaces": [
            {
                "type": 2,
                "main": 1,
                "useip": 1,
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": 3,
                    "bulk": 0,
                    "securityname": "mysecurityname",
                    "contextname": "",
                    "securitylevel": 1
                }
            }
        ],
        "groups": [
            {
                "groupid": "4"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10658"
        ]
    },
    "id": 1
}
```

使用 PSK 加密创建主机

创建一个名为 "PSK host" 的主机，并配置 PSK 加密。请注意，该主机必须已预先配置为使用 PSK。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "PSK host",
        "interfaces": [
            {
```

```
                "type": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050",
                "useip": 1,
                "main": 1
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "tls_accept": 2,
        "tls_connect": 2,
        "tls_psk_identity": "PSK 001",
        "tls_psk": "1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "id": 1
}
```

创建由 proxy 监控的主机

创建主机由 ID 为"1" 的 proxy 来监控。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Host monitored by proxy",
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "monitored_by": 1,
        "proxyid": 1
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10591"
        ]
    },
    "id": 1
}
```

创建由 proxy 组监控的主机

创建由 ID 为"1" 的 proxy 组监控的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Host monitored by proxy group",
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "monitored_by": 2,
        "proxy_groupid": 1
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10592"
        ]
    },
    "id": 1
}
```

另请参见

- 主机组
- 模板
- 用户宏
- 主机接口
- 主机清单
- 主机标签
- Proxy
- Proxy 组

源码

CHost::create() in *ui/include/classes/api/services/CHost.php*.

删除

描述

`object host.delete(array hosts)`

此方法允许您删除主机。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(array)` 要删除主机的 ID。

返回值

`(object)` 返回一个在 `hostids` 属性下包含已删除主机 ID 的对象。

示例

删除多个主机

删除两个主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.delete",
    "params": [
        "13",
        "32"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "13",
            "32"
        ]
    },
    "id": 1
}
```

源码

CHost::delete() in *ui/include/classes/api/services/CHost.php*.

批量删除

描述

`object host.massremove(object parameters)`

此方法允许你从多个主机中移除相关对象。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 包含要更新的主机 ID 以及应删除的对象的参数。

参数	类型	描述
hostids	ID/array	要更新的主机的 ID。 参数行为： - 必填
groupids	ID/array	将给定主机从中移除的主机群组的 ID。
interfaces	object/array	要从给定主机中移除的主机接口。 主机接口对象必须只定义 `ip`、`dns` 和 `port` 属性。
macros	string/array	要从给定主机中删除的用户宏。
templateids	ID/array	要与给定主机取消关联的模板的 ID。
templateids_clear	ID/array	要与给定主机取消关联并清除的模板的 ID。

返回值

`(object)` 返回一个对象，该对象包含更新的主机的 `hostids` 属性。

示例

取消链接模板

取消链接两台主机的目标模板，并删除所有模板实体。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.massremove",
    "params": {
        "hostids": ["69665", "69666"],
        "templateids_clear": "325"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "69665",
            "69666"
        ]
    },
    "id": 1
}
```

另请参见

- host.update
- 用户宏
- 主机接口

源码

CHost::massRemove() in *ui/include/classes/api/services/CHost.php*.

批量添加

描述

`object host.massadd(object parameters)`

此方法允许你同时将多个相关对象添加到所有指定的主机。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 包含要更新的主机 ID 以及要添加到所有主机的对象的参数。

该方法接受以下参数。

参数	类型	描述
hosts	object/array	要更新的主机。 这些主机必须仅定义 `hostid` 属性。 参数行为： - 必填

参数	类型	描述
groups	object/array	要添加到给定主机的主机组。 这些主机组必须仅定义 groupid 属性。
interfaces	object/array	要为给定主机创建的主机接口。
macros	object/array	要为给定主机创建的用户宏。
templates	object/array	要链接到给定主机的模板。 这些模板必须仅定义 templateid 属性。

返回值

(object) 返回一个对象，该对象包含更新主机的 hostids 属性。

示例

添加宏

给两台主机添加两个新的宏。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "10160"
            },
            {
                "hostid": "10167"
            }
        ],
        "macros": [
            {
                "macro": "{$TEST1}",
                "value": "MACROTEST1"
            },
            {
                "macro": "{$TEST2}",
                "value": "MACROTEST2",
                "description": "Test description"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10160",
            "10167"
        ]
    },
    "id": 1
}
```

另请参见

- host.update
- 主机组

- 模板
- 用户宏
- 主机接口

源码

CHost::massAdd() in *ui/include/classes/api/services/CHost.php*.

更新

描述

`object host.update(object/array hosts)`

此方法允许您更新现有主机。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要更新的主机属性。

必须为每个主机定义 `hostid` 属性，其他所有属性均为可选。只有给定的属性会被更新，其他所有属性将保持不变。

但请注意，更新主机技术名称时，也会将主机的可见名称更新为主机技术名称的值（如果未单独提供可见名称）。

除标准主机属性外，该方法还接受以下参数。

参数	类型	描述
groups	object/array	用于替换主机当前所属主机组的主机组。 请求中未列出的所有主机组都将被取消关联。 主机组必须仅定义 `groupid` 属性。
interfaces	object/array	用于替换当前主机接口的主机接口。 请求中未列出的所有接口都将被移除。
tags	object/array	用于替换当前主机标签的主机标签。 请求中未列出的所有标签都将被移除。
inventory	object	主机资产记录属性。
macros	object/array	用于替换当前用户宏的用户宏。 请求中未列出的所有宏都将被移除。
templates	object/array	用于替换当前已关联模板的模板。 请求中未列出的所有模板将仅被取消关联。 模板必须仅定义 `templateid` 属性。
templates_clear	object/array	要从主机取消关联并清除的模板。 模板必须仅定义 `templateid` 属性。

返回值

`(object)` 返回一个对象，该对象包含更新的主机的 `hostids` 属性。

示例

启用主机

启用主机监控，将主机状态设为"0"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "status": 0
```

```
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

启用多个主机

启用对两个主机的监控，即将它们的状态设置为 “0”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": [
        {
            "hostid": "10127",
            "status": 0
        },
        {
            "hostid": "10128",
            "status": 0
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10127",
            "10128"
        ]
    },
    "id": 1
}
```

取消链接模板

从主机取消链接并清理两个模板。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "templates_clear": [
            {
                "templateid": "10124"
            },
            {
                "templateid": "10125"
```

```
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

更新主机宏

用两个新的宏替换所有主机宏。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "macros": [
            {
                "macro": "{$PASS}",
                "value": "password"
            },
            {
                "macro": "{$DISC}",
                "value": "sda",
                "description": "Updated description"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

更新主机资产清单

更改主机资产清单模式并添加位置信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "inventory_mode": 0,
```

```
        "inventory": {
            "location": "Latvia, Riga"
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

更新主机标签

使用一个新的标签替换所有主机标签。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "tags": {
            "tag": "os",
            "value": "rhel-7"
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

更新已发现主机的宏

改变发现规则创建的” 自动” 宏为” 手动” 宏，并将它的值改成”new-value”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "macros": {
            "hostmacroid": "5541",
            "value": "new-value",
            "automatic": "0"
        }
    },
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

更新主机加密

将主机"10590" 更新为仅对从主机到 Zabbix 服务器的连接使用 PSK 加密，并更改 PSK 标识和 PSK 密钥。
请注意，该主机必须已预先配置为使用 PSK。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10590",
        "tls_connect": 1,
        "tls_accept": 2,
        "tls_psk_identity": "PSK 002",
        "tls_psk": "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "id": 1
}
```

另请参见

- host.massadd
- host.massremove
- 主机组
- 模板
- 用户宏
- 主机接口
- 主机清单
- 主机标签
- Proxy
- Proxy 组

源码

CHost::update() in *ui/include/classes/api/services/CHost.php*.

获取

描述

`integer/array host.get(object parameters)`

该方法允许您根据给定参数检索主机。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
groupids	ID/array	仅返回属于给定组的主机。
dserviceids	ID/array	仅返回与给定已发现服务相关的主机。
graphids	ID/array	仅返回具有给定图形的主机。
hostids	ID/array	仅返回具有给定主机 ID 的主机。
httptestids	ID/array	仅返回具有给定 Web 检测的主机。
interfaceids	ID/array	仅返回使用给定接口的主机。
itemids	ID/array	仅返回具有给定监控项的主机。
maintenanceids	ID/array	仅返回受给定维护影响的主机。
monitored_hosts	flag	仅返回受监控的主机。
proxyids	ID/array	仅返回由给定 proxies 监控的主机。
proxy_groupids	ID/array	仅返回由给定 proxy 组监控的主机。
templated_hosts	flag	同时返回主机和模板。
templateids	ID/array	仅返回链接到给定模板的主机。
triggerids	ID/array	仅返回具有给定触发器的主机。
with_items	flag	仅返回具有监控项的主机。 会覆盖 `with_monitored_items` 和 `with_simple_graph_items` 参数。
with_item_prototypes	flag	仅返回具有监控项原型的主机。 会覆盖 `with_simple_graph_item_prototypes` 参数。
with_simple_graph_item_prototypes	flag	仅返回具有监控项原型的主机，这些原型已启用创建且信息类型为数值型。
with_graphs	flag	仅返回具有图形的主机。
with_graph_prototypes	flag	仅返回具有图形原型的主机。
with_httptests	flag	仅返回具有 Web 检测的主机。 会覆盖 `with_monitored_httptests` 参数。
with_monitored_httptests	flag	仅返回具有已启用 Web 检测的主机。
with_monitored_items	flag	仅返回具有已启用监控项的主机。 会覆盖 `with_simple_graph_items` 参数。
with_monitored_triggers	flag	仅返回具有已启用触发器的主机。触发器中使用的所有监控项也必须已启用。
with_simple_graph_items	flag	仅返回具有信息类型为数值型监控项的主机。
with_triggers	flag	仅返回具有触发器的主机。 会覆盖 `with_monitored_triggers` 参数。
withProblemsSuppressed	boolean	如果设置为 `true`，则仅返回具有已抑制问题的主机。
evaltype	integer	标签计算方法。 可能的值： 0 - *(默认)* And/Or； 2 - Or。
severities	integer/array	返回仅具有给定严重性的主机问题。仅当问题对象为触发器时适用。

参数	类型	描述
tags	object/array	仅返回具有给定标签的主机。 格式：[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]。 空数组返回所有主机。 可能的operator 值： 0 - *(默认)* 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
inheritedTags	boolean	返回在其所有已链接模板中也具有给定 tags 的主机。 可能的值： true - 已链接模板也必须具有给定的 tags； false - *(默认)* 忽略已链接模板的标签。
selectDiscoveryData	query	返回带有主机发现对象数据的 discoveryData 属性。主机发现对象将已发现主机关联到发现该主机的主机原型。 它具有以下属性： host - (string) 主机的 ID； parent_hostid - (string) 创建该主机所依据的主机原型 ID； status - (int) 主机发现状态： 0 - *(默认)* 主机已发现， 1 - 主机不再被发现； ts_delete - (timestamp) 不再被发现的主机将被删除的时间； ts_disable - (timestamp) 不再被发现的主机将被禁用的时间； disable_source - (int) 指示主机是由 LLD 规则禁用还是手动禁用： 0 - *(默认)* 自动禁用， 1 - 由 LLD 规则禁用。
selectDiscoveryRule	query	返回带有创建该主机的低级别发现规则的discoveryRule 属性（来自 VMware 监控中的主机原型）。
selectDiscoveryRules	query	返回带有主机 LLD 规则的discoveryRules 属性。 支持 count。
selectGraphs	query	返回带有主机图形的graphs 属性。 支持 count。
selectHostGroups	query	返回带有主机所属主机组数据的hostgroups 属性。
selectHttpTests	query	返回带有主机 Web 场景的httpTests 属性。 支持 count。
selectInterfaces	query	返回带有主机接口的interfaces 属性。 支持 count。
selectInventory	query	返回带有主机资产清单数据的inventory 属性。
selectItems	query	返回带有主机监控项的items 属性。 支持 count。
selectMacros	query	返回带有主机宏的macros 属性。
selectParentTemplates	query	返回带有主机所链接模板的 parentTemplates 属性。 除 Template 对象字段外，还包含 link_type - (integer) 模板链接到主机的方式。 可能的值： 0 - *(默认)* 手动链接； 1 - 由 LLD 自动链接。 支持 count。

参数	类型	描述
selectDashboards	query	返回dashboards 属性。 支持 count。
selectTags	query	返回带有主机标签的tags 属性。
selectInheritedTags	query	返回带有链接到主机的所有模板上标签的inheritedTags 属性。
selectTriggers	query	返回带有主机触发器的triggers 属性。 支持 count。
selectValueMaps	query	返回带有主机值映射的valuemaps 属性。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 text 数据类型 的属性。 支持附加属性：主机接口属性。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectParentTemplates - 结果将按 host 排序； selectInterfaces； selectItems - 按 name 排序； selectDiscoveryRules - 按 name 排序； selectTriggers - 按 description 排序； selectGraphs - 按 name 排序； selectDashboards - 按 name 排序。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，将执行 LIKE "%…%" 搜索。 仅支持 string 和 text 数据类型 的属性。 支持附加属性：主机接口属性。
searchInventory	object	返回资产清单数据与给定模式匹配的主机（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，将执行 LIKE "%…%" 搜索。 仅支持 string 和 text 数据类型 的属性。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：hostid、host、name、status。
countOutput	boolean	这些参数在参考说明中描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectDiscoveries	query	返回带有主机低级别发现规则的discoveries 属性。 支持 count。
selectHostDiscovery	query	此查询已弃用，请改用 selectDiscoveryRules。 返回带有主机发现对象数据的 hostDiscovery 属性。 此查询已弃用，请改用 selectDiscoveryData。

返回值

(integer/array) 返回其中之一:

- 一个对象的数组;
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

按名称检索数据

获取主机名为"Zabbix server" 和"Linux server" 的所有数据。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": [
                "Zabbix server",
                "Linux server"
            ]
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10160",
            "proxyid": "0",
            "host": "Zabbix server",
            "status": "0",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Zabbix server",
            "flags": "0",
            "description": "The Zabbix monitoring server.",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_groupid": "0",
            "monitored_by": "0",
            "inventory_mode": "1",
            "active_available": "1",
            "assigned_proxyid": "0"
        },
        {
            "hostid": "10167",
            "proxyid": "0",
            "host": "Linux server",
            "status": "0",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
```

```
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Linux server",
            "flags": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_groupid": "0",
            "monitored_by": "0",
            "inventory_mode": "1",
            "active_available": "1",
            "assigned_proxyid": "0"
        }
    ],
    "id": 1
}
```

获取主机组

获取主机"Zabbix server" 所属的主机组。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectHostGroups": "extend",
        "filter": {
            "host": [
                "Zabbix server"
            ]
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "hostgroups": [
                {
                    "groupid": "2",
                    "name": "Linux servers",
                    "flags": "0",
                    "uuid": "dc579cd7a1a34222933f24f52a68bcd8"
                },
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "flags": "0",
                    "uuid": "6f6799aa69e844b4b3918f779f2abf08"
                }
            ]
```

```
        }
    ],
    "id": 1
}
```

获取链接的模板

获取主机"10084" 链接的模板 ID 和名称。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectParentTemplates": [
            "templateid",
            "name"
        ],
        "hostids": "10084"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "parentTemplates": [
                {
                    "name": "Linux",
                    "templateid": "10001"
                },
                {
                    "name": "Zabbix Server",
                    "templateid": "10047"
                }
            ]
        }
    ],
    "id": 1
}
```

按模板获取主机

获取链接模板"10001" (*Linux by Zabbix agent*) 的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "name"],
        "templateids": "10001"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
        {
            "templateid": "10001",
            "hosts": [
                {
                    "hostid": "10084",
                    "name": "Zabbix server"
                },
                {
                    "hostid": "10603",
                    "name": "Host 1"
                },
                {
                    "hostid": "10604",
                    "name": "Host 2"
                }
            ]
        }
    ],
    "id": 1
}
```

通过主机资产清单数据查询主机

查询主机资产清单中"OS" 字段包含"Linux" 的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "host"
        ],
        "selectInventory": [
            "os"
        ],
        "searchInventory": {
            "os": "Linux"
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "inventory": {
                "os": "Linux Ubuntu"
            }
        },
        {
            "hostid": "10107",
            "host": "Linux server",
            "inventory": {
                "os": "Linux Mint"
            }
        }
    ],
    "id": 1
```

```
}
```

按主机标签搜索

检索标签”host-name” 等于”linux-server” 的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "host-name",
                "value": "linux-server",
                "operator": 1
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "tags": [
                {
                    "tag": "host-name",
                    "value": "linux-server",
                    "automatic": "0"
                },
                {
                    "tag": "os",
                    "value": "rhel-7",
                    "automatic": "0"
                }
            ]
        }
    ],
    "id": 1
}
```

检索不仅在主机级别，而且在其已链接父模板中也具有这些标签的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "tags": [
            {
                "tag": "os",
                "value": "rhel-7",
                "operator": 1
            }
        ],
```

```
        "inheritedTags": true
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10623",
            "name": "PC room 1"
        },
        {
            "hostid": "10601",
            "name": "Office"
        }
    ],
    "id": 1
}
```

检索排除带有标签”component” 的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "component",
                "value": "",
                "operator": 5
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "tags": [
                {
                    "tag": "class",
                    "value": "os",
                    "automatic": "0"
                },
                {
                    "tag": "target",
                    "value": "linux",
                    "automatic": "0"
                }
            ]
        },
        {
            "hostid": "10629",
```

```
            "tags": [
                {
                    "tag": "class",
                    "value": "application",
                    "automatic": "0"
                },
                {
                    "tag": "target",
                    "value": "browser",
                    "automatic": "0"
                }
            ]
        }
    ],
    "id": 1
}
```

使用标签和模板标签搜索主机

检索带有标签以及所有链接到父模板的标签的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "hostids": 10502,
        "selectTags": ["tag", "value"],
        "selectInheritedTags": ["tag", "value"]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10502",
            "name": "Desktop",
            "tags": [
                {
                    "tag": "host-name",
                    "value": "linux-server"
                },
                {
                    "tag": "os",
                    "value": "rhel-7"
                }
            ],
            "inheritedTags": [
                {
                    "tag": "class",
                    "value": "os"
                },
                {
                    "tag": "target",
                    "value": "linux"
                },
                {
                    "tag": "os",
                    "value": "rhel-7"
```

```
                }
            ]
        }
    ],
    "id": 1
}
```

根据问题严重程度检索主机

检索所有” 灾难” 问题的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "severities": 5
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10160",
            "name": "Zabbix server"
        }
    ],
    "id": 1
}
```

检索具有” 一般严重” 和” 严重” 问题的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "severities": [3, 4]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "20170",
            "name": "Database"
        },
        {
            "hostid": "20183",
            "name": "workstation"
        }
    ],
    "id": 1
}
```

获取主机监控项

获取 “Zabbix server” 主机的监控项。该请求仅限返回监控项属性 itemid、name 和 status。由于响应内容较大，示例中仅显示了部分监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectItems": ["itemid","name","status"],
        "filter": {
            "host": [
                "Zabbix server"
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostid": "10084",
      "items": [
        {
          "itemid": "42227",
          "name": " 正在运行的 Zabbix agent 的主机名",
          "status": "0"
        },
        {
          "itemid": "42237",
          "name": "Zabbix agent ping",
          "status": "0"
        },
        {
          "itemid": "42250",
          "name": " 正在运行的 Zabbix agent 的版本",
          "status": "0"
        },
        {
          "itemid": "42251",
          "name": " 打开文件描述符的最大数量",
          "status": "0"
        },
        {
          "itemid": "42252",
          "name": " 进程的最大数量",
          "status": "0"
        },
        {
          "itemid": "69869",
          "name": " 接口 enp0s3：丢弃的入站数据包数",
          "status": "0"
        },
        {
          "itemid": "69870",
          "name": " 接口 enp0s3：有错误的入站数据包数",
          "status": "0"
        },
        {
          "itemid": "69871",
```

```
            "name": " 接口 enp0s3：接收的比特数",
            "status": "0"
          },
          {
            "itemid": "69872",
            "name": " 接口 enp0s3：丢弃的出站数据包数",
            "status": "0"
          },
          {
            "itemid": "69873",
            "name": " 接口 enp0s3：有错误的出站数据包数",
            "status": "0"
          },
          {
            "itemid": "69874",
            "name": " 接口 enp0s3：发送的比特数",
            "status": "0"
          },
          {
            "itemid": "42253",
            "name": " 进程数量",
            "status": "0"
          },
          {
            "itemid": "42254",
            "name": " 正在运行的进程数量",
            "status": "0"
          },
          {
            "itemid": "42255",
            "name": " 系统启动时间",
            "status": "0"
          }
        ]
      }
    ],
    "id": 1
}
```


另请参见

- 主机组
- 模板
- 用户宏
- 主机接口
- Proxy
- Proxy 组

源码

CHost::get() in *ui/include/classes/api/services/CHost.php*.

主机原型

此类用于处理主机原型。

对象引用：

- 主机原型
- 组链接
- 组原型
- 主机原型标签
- 自定义接口
    - 自定义接口详情

可用方法：

- hostprototype.create - 创建新的主机原型
- hostprototype.delete - 删除主机原型
- hostprototype.get - 获取主机原型
- hostprototype.update - 更新主机原型

主机原型对象

以下对象与 hostprototype API 直接相关。

主机原型

主机原型对象具有以下属性。

Property	Type	Description
hostid	ID	主机原型的 ID。 属性行为： - 只读 - 更新操作时 必需
host	string	主机原型的技术名称。 属性行为： - 创建操作时 必需 - 对继承对象为 只读
name	string	主机原型的可见名称。 默认值：host 属性值。 属性行为： - 对继承对象为 只读
status	integer	主机原型的状态。 可能的值： 0 - （默认）已监控主机； 1 - 未监控主机。
flags	integer	主机原型的来源。 可能的值： 2 - 主机原型； 6 - 已发现的主机原型 属性行为： - 只读
inventory_mode	integer	主机资产记录填充模式。 可能的值： -1 - （默认）禁用； 0 - 手动； 1 - 自动。
templateid	ID	父模板主机原型的 ID。 属性行为： - 只读
discover	integer	主机原型发现状态。 可能的值： 0 - （默认）将发现新主机； 1 - 不会发现新主机，现有主机将被标记为丢失。

Property	Type	Description
custom_interfaces	integer	由主机原型创建的主机的自定义接口来源。 可能的值： 0 - （默认）继承父主机的接口； 1 - 使用主机原型的自定义接口。 属性行为： - 对继承对象为 只读
uuid	string	通用唯一标识符，用于将导入的主机原型链接到已存在的主机原型。如果未提供，则自动生成。 属性行为： - 如果主机原型属于模板，则 支持

组链接

组链接对象将主机原型与主机组关联起来。它具有以下属性。

Property	Type	Description
groupid	ID	主机组的 ID。 属性行为: - 必需

组原型

组原型对象定义了将为发现的主机创建的组。它具有以下属性。

属性	类型	说明
group_prototypeid	ID	组原型的 ID。 属性行为： - 只读
name	string	组原型的名称。 属性行为： - 创建操作时为必填

主机原型标签

主机原型标签对象具有以下属性。

属性	Type	说明
tag	string	主机原型标签名称。 属性行为： - 必填
value	string	主机原型标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板。 属性行为： - 只读。
objectid	ID	标签继承自的对象的 ID。 属性行为： - 只读。

自定义接口

如果主机原型对象 的 `custom_interfaces` 设置为 “使用主机原型的自定义接口”，则支持自定义接口。自定义接口对象具有以下属性。

属性	类型	描述
type	integer	接口类型。 可能的值： 1 - agent； 2 - SNMP； 3 - IPMI； 4 - JMX。 属性行为： - 必填
useip	integer	是否应通过 IP 建立连接。 可能的值： 0 - 使用主机 DNS 名称连接； 1 - 使用主机 IP 地址连接。 属性行为： - 必填
ip	string	接口使用的 IP 地址。 可包含宏。 属性行为： - 如果 `useip` 设置为 “使用主机 IP 地址连接”，则为 必填
dns	string	接口使用的 DNS 名称。 可包含宏。 属性行为： - 如果 `useip` 设置为 “使用主机 DNS 名称连接”，则为 必填
port	string	接口使用的端口号。 可包含用户宏和 LLD 宏。 属性行为： - 必填
main	integer	该接口是否用作主机上的默认接口。 在一台主机上，每种类型只能有一个接口可设置为默认。 可能的值： 0 - 非默认； 1 - 默认。 属性行为： - 必填
details	object	用于自定义接口详细信息的附加对象。 属性行为： - 如果 `type` 设置为 “SNMP”，则为 必填

自定义接口详情

details 对象具有以下属性。

属性	类型	描述
version	integer	SNMP 接口版本。 可能的值： 1 - SNMPv1； 2 - SNMPv2c； 3 - SNMPv3。 属性行为： - 必需
bulk	integer	是否使用批量 SNMP 请求。 可能的值： 0 - 不使用批量请求； 1 - *(默认)* - 使用批量请求。
community	string	SNMP community。 属性行为： - 当 `version` 设置为”SNMPv1” 或”SNMPv2c” 时为 必需
max_repetitions	integer	原生 SNMP 批量请求（GetBulkRequest-PDUs）的最大重复值。 仅用于 SNMPv2 和 v3 中的 `discovery[]` 和 `walk[]` 监控项。 默认值：10。
securityname	string	SNMPv3 安全名称。 属性行为： - 当 `version` 设置为”SNMPv3” 时 支持
securitylevel	integer	SNMPv3 安全级别。 可能的值： 0 - *(默认)* - noAuthNoPriv； 1 - authNoPriv； 2 - authPriv。 属性行为： - 当 `version` 设置为”SNMPv3” 时 支持
authpassphrase	string	SNMPv3 身份验证口令。 属性行为： - 当 `version` 设置为”SNMPv3” 且 `securitylevel` 设置为”authNoPriv” 或”authPriv” 时 支持
privpassphrase	string	SNMPv3 加密口令。 属性行为： - 当 `version` 设置为”SNMPv3” 且 `securitylevel` 设置为”authPriv” 时 支持
authprotocol	integer	SNMPv3 身份验证协议。 可能的值： 0 - *(默认)* - MD5； 1 - SHA1； 2 - SHA224； 3 - SHA256； 4 - SHA384； 5 - SHA512。 属性行为： - 当 `version` 设置为”SNMPv3” 且 `securitylevel` 设置为”authNoPriv” 或”authPriv” 时 支持

属性	类型	描述
privprotocol	integer	SNMPv3 加密协议。仅由 SNMPv3 接口使用。 可能的值： 0 - *(默认)* - DES； 1 - AES128； 2 - AES192； 3 - AES256； 4 - AES192C； 5 - AES256C。 属性行为： - 当 `version` 设置为”SNMPv3” 且 `securitylevel` 设置为”authPriv” 时支持
contextname	string	SNMPv3 上下文名称。 属性行为： - 当 `version` 设置为”SNMPv3” 时 支持

创建

描述

`object hostprototype.create(object/array hostPrototypes)`

此方法允许您创建新的主机原型。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的主机原型。

除了标准主机原型属性之外，该方法还接受以下参数。

参数	类型	说明
groupLinks	array	要为主机原型创建的组链接。 参数行为： - 必需
ruleid	ID	主机原型所属的 LLD 规则 ID。 参数行为： - 必需
groupPrototypes	array	要为主机原型创建的组原型。
macros	object/array	要为主机原型创建的用户宏。
tags	object/array	主机原型标签。
interfaces	object/array	主机原型的自定义接口。
templates	object/array	要链接到主机原型的模板。 这些模板必须仅定义 `templateid` 属性。

返回值

`(object)` 返回一个对象，其中在 `hostids` 属性下包含创建的主机原型的 ID。返回 ID 的顺序与传递的主机原型的顺序一致。

示例

创建主机原型

在 LLD 规则 “23542” 上创建主机原型 “{#VM.NAME}”，并设置组原型 “{#HV.NAME}”、标签对 “datacenter”: “{#DATACENTER.NAME}”，以及自定义 SNMPv2 接口 127.0.0.1:161，community 为 {$SNMP_COMMUNITY}。将其链接到主机组 “2”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.create",
    "params": {
        "host": "{#VM.NAME}",
        "ruleid": "23542",
        "custom_interfaces": "1",
        "groupLinks": [
            {
                "groupid": "2"
            }
        ],
        "groupPrototypes": [
            {
                "name": "{#HV.NAME}"
            }
        ],
        "tags": [
            {
                "tag": "datacenter",
                "value": "{#DATACENTER.NAME}"
            }
        ],
        "interfaces": [
            {
                "main": "1",
                "type": "2",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": "2",
                    "bulk": "1",
                    "community": "{$SNMP_COMMUNITY}"
                }
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103"
        ]
    },
    "id": 1
}
```

另请参阅

- 组链接
- 组原型
- 主机原型标签
- 自定义接口
- 用户宏

来源

CHostPrototype::create() 在 *ui/include/classes/api/services/CHostPrototype.php*。

删除

描述

`object hostprototype.delete(array hostPrototypeIds)`

此方法允许您删除主机原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的主机原型的 ID。

返回值

(object) 返回一个对象，其中在 hostids 属性下包含已删除主机原型的 ID。

示例

删除多个主机原型

删除两个主机原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.delete",
    "params": [
        "10103",
        "10105"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103",
            "10105"
        ]
    },
    "id": 1
}
```

来源

CHostPrototype::delete() 在 *ui/include/classes/api/services/CHostPrototype.php*。

更新

描述

`object hostprototype.update(object/array hostPrototypes)`

此方法允许您更新现有的主机原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的主机原型属性。

必须为每个主机原型定义 hostid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准主机原型属性之外，该方法还接受以下参数。

参数	类型	描述
groupLinks	array	用于替换主机原型当前主机组链接的主机组链接。 参数行为： - 对继承对象为只读
groupPrototypes	array	用于替换主机原型现有主机组原型的主机组原型。 请求中未列出的所有主机组原型都将被移除。 参数行为： - 对继承对象为只读
macros	object/array	用于替换当前用户宏的用户宏。 请求中未列出的所有宏都将被移除。
tags	object/array	用于替换当前标签的主机原型标签。 请求中未列出的所有标签都将被移除。 参数行为： - 对继承对象为只读
interfaces	object/array	用于替换当前接口的主机原型自定义接口。 自定义接口对象应包含其所有参数。 请求中未列出的所有接口都将被移除。 参数行为： - 当主机原型对象的 custom_interfaces 设置为 “使用主机原型自定义接口” 时，支持 - 对继承对象为只读
templates	object/array	用于替换当前已链接模板的模板。 模板必须只定义 templateid 属性。

返回值

（object）返回一个对象，其中在 hostids 属性下包含更新的主机原型的 ID。

示例

禁用主机原型

禁用主机原型，即将其状态设置为”1”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "status": 1
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
```

```
    "id": 1
}
```

更新主机原型标签

使用新的标签替换主机原型标签。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "tags": [
            {
                "tag": "datacenter",
                "value": "{#DATACENTER.NAME}"
            },
            {
                "tag": "instance-type",
                "value": "{#INSTANCE_TYPE}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

更新主机原型自定义接口

用主机原型自定义接口取代继承接口。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "custom_interfaces": "1",
        "interfaces": [
            {
                "main": "1",
                "type": "2",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": "2",
                    "bulk": "1",
                    "community": "{$SNMP_COMMUNITY}"
                }
            }
        ]
```

```
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

另请参阅

- 组链接
- 组原型
- 主机原型标签
- 自定义接口
- 用户宏

来源

CHostPrototype::update() in *ui/include/classes/api/services/CHostPrototype.php*.

获取

描述

`integer/array hostprototype.get(object parameters)`

该方法允许您根据给定参数检索主机原型。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
hostids	ID/array	仅返回具有给定 ID 的主机原型。
discoveryids	ID/array	仅返回属于给定 LLD 规则的主机原型。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的监控项。
selectDiscoveryData	query	返回一个 `discoveryData` 属性，其中包含主机原型发现对象数据。主机原型发现对象将已发现的主机原型关联到发现它的主机原型。 它具有以下属性： `host` - `(string)` 主机的 ID； `parent_hostid` - `(string)` 创建该主机原型所依据的主机原型 ID； `status` - `(int)` 主机原型发现状态： 0 - *(默认)* 主机原型已发现， 1 - 主机原型不再被发现； `ts_delete` - `(timestamp)` 不再被发现的主机原型将被删除的时间； `ts_disable` - `(timestamp)` 不再被发现的主机原型将被禁用的时间； `disable_source` - `(int)` 指示主机原型是由 LLD 规则禁用还是手动禁用： 0 - *(默认)* 自动禁用， 1 - 由 LLD 规则禁用。
selectDiscoveryRule	query	返回一个`discoveryRule` 属性，其中包含该主机原型所属的 LLD 规则。

参数	类型	描述
selectDiscoveryRulePrototype	query	返回一个discoveryRulePrototype 属性，其中包含该主机原型所属的父 LLD 规则原型。
selectInterfaces	query	返回一个interfaces 属性，其中包含主机原型的自定义接口。
selectGroupLinks	query	返回一个groupLinks 属性，其中包含主机原型的组链接。
selectGroupPrototypes	query	返回一个groupPrototypes 属性，其中包含主机原型的组原型。
selectInheritedTags	query	返回一个inheritedTags 属性，其中包含位于已链接模板上的标签。
selectMacros	query	返回一个macros 属性，其中包含主机原型宏。
selectParentHost	query	返回一个parentHost 属性，其中包含该主机原型所属的主机。
selectTags	query	返回一个tags 属性，其中包含主机原型标签。
selectTemplates	query	返回一个templates 属性，其中包含链接到该主机原型的模板。 支持 count。
sortfield	string/array	按给定属性对结果进行排序。 可选值：hostid、host、name、status、discovered。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

从 LLD 规则中获取主机原型

从 LLD 规则中获取所有主机原型，以及它们的组链接、组原型和标签。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.get",
    "params": {
        "output": "extend",
        "selectInterfaces": "extend",
        "selectGroupLinks": "extend",
        "selectGroupPrototypes": "extend",
        "selectTags": "extend",
        "discoveryids": "23554"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10092",
            "host": "{#HV.UUID}",
```

```
            "name": "{#HV.UUID}",
            "status": "0",
            "templateid": "0",
            "discover": "0",
            "custom_interfaces": "1",
            "uuid": "051a1469d4d045cbbf818fcc843a352e",
            "flags": "2",
            "inventory_mode": "-1",
            "tags": [
                {
                    "tag": "datacenter",
                    "value": "{#DATACENTER.NAME}"
                },
                {
                    "tag": "instance-type",
                    "value": "{#INSTANCE_TYPE}"
                }
            ],
            "groupLinks": [
                {
                    "group_prototypeid": "4",
                    "hostid": "10092",
                    "groupid": "7",
                    "templateid": "0"
                }
            ],
            "groupPrototypes": [
                {
                    "group_prototypeid": "7",
                    "hostid": "10092",
                    "name": "{#CLUSTER.NAME}",
                    "templateid": "0"
                }
            ],
            "interfaces": [
                {
                    "main": "1",
                    "type": "2",
                    "useip": "1",
                    "ip": "127.0.0.1",
                    "dns": "",
                    "port": "161",
                    "available": "0",
                    "error": "",
                    "errors_from": "0",
                    "disable_until": "0",
                    "details": {
                        "version": "2",
                        "bulk": "1",
                        "community": "{$SNMP_COMMUNITY}",
                        "max_repetitions": "10"
                    }
                }
            ]
        }
    ],
    "id": 1
}
```


另请参阅

- 组链接
- 组原型

- 用户宏

来源

ui/include/classes/api/services/CHostPrototype.php 中的 CHostPrototype::get()。

主机接口

此类提供对主机接口的访问。

对象引用：

- 主机接口
  - 详细信息

可用方法：

- hostinterface.get - 检索主机接口

主机接口使用主机方法创建、更新和删除。

主机接口对象

以下对象与 `hostinterface` API 直接相关。

主机接口

主机接口对象具有以下属性。

属性	类型	描述
interfaceid	ID	接口的 ID。 属性行为： - 只读 - 更新操作中为必需
available	integer	主机接口的可用性。 可能的值： 0 - （默认）未知； 1 - 可用； 2 - 不可用。 属性行为： - 只读
hostid	ID	接口所属主机的 ID。 属性行为： - 常量 - 创建操作中为必需
type	integer	接口类型。 可能的值： 1 - Agent； 2 - SNMP； 3 - IPMI； 4 - JMX。 属性行为： - 创建操作中为必需
ip	string	接口使用的 IP 地址。 如果通过 DNS 建立连接，则可以为空。 属性行为： - 创建操作中为必需

属性	类型	描述
dns	string	接口使用的 DNS 名称。 如果通过 IP 建立连接，则可以为空。 属性行为： - 创建操作中为必需
port	string	接口使用的端口号。 可以包含用户宏。 属性行为： - 创建操作中为必需
useip	integer	是否应通过 IP 建立连接。 可能的值： 0 - 使用主机 DNS 名称进行连接； 1 - 使用主机 IP 地址进行连接。 属性行为： - 创建操作中为必需
main	integer	接口是否在主机上作为默认接口使用。同一类型的接口中只能有一个可在主机上设置为默认接口。 可能的值： 0 - 非默认； 1 - 默认。 属性行为： - 创建操作中为必需
details	object	接口的其他详细信息对象。 属性行为： - 当 type 设置为”SNMP” 时为必需
disable_until	timestamp	不可用主机接口的下一次轮询时间。 属性行为： - 只读
error	string	主机接口不可用时的错误文本。 属性行为： - 只读
errors_from	timestamp	主机接口变为不可用的时间。 属性行为： - 只读

详细信息

details 对象具有以下属性。

属性	Type	描述
version	integer	SNMP 接口版本。 可能的值： 1 - SNMPv1； 2 - SNMPv2c； 3 - SNMPv3。 属性行为： - 必填

属性	Type	描述
bulk	integer	是否使用批量 SNMP 请求。 可能的值： 0 - 不使用批量请求； 1 - （默认）- 使用批量请求。
community	string	SNMP community。仅由 SNMPv1 和 SNMPv2 接口使用。
max_repetitions	integer	属性行为： - 如果 `version` 设置为"SNMPv1" 或"SNMPv2c"，则为 必填 原生 SNMP 批量请求（GetBulkRequest-PDUs）的最大重复值。 仅用于 SNMPv2 和 v3 中的 `discovery[]` 和 `walk[]` 监控项。 默认值：10。
securityname	string	SNMPv3 安全名称。仅由 SNMPv3 接口使用。
securitylevel	integer	SNMPv3 安全级别。仅由 SNMPv3 接口使用。 可能的值： 0 - （默认）- noAuthNoPriv； 1 - authNoPriv； 2 - authPriv。
authpassphrase	string	SNMPv3 身份验证口令。仅由 SNMPv3 接口使用。
privpassphrase	string	SNMPv3 隐私口令。仅由 SNMPv3 接口使用。
authprotocol	integer	SNMPv3 身份验证协议。仅由 SNMPv3 接口使用。 可能的值： 0 - （默认）- MD5； 1 - SHA1； 2 - SHA224； 3 - SHA256； 4 - SHA384； 5 - SHA512。
privprotocol	integer	SNMPv3 隐私协议。仅由 SNMPv3 接口使用。 可能的值： 0 - （默认）- DES； 1 - AES128； 2 - AES192； 3 - AES256； 4 - AES192C； 5 - AES256C。
contextname	string	SNMPv3 上下文名称。仅由 SNMPv3 接口使用。

获取

描述

`integer/array hostinterface.get(object parameters)`

此方法允许您根据给定的参数检索主机接口。

Note:
任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
hostids	ID/array	仅返回给定主机使用的主机接口。

参数	类型	描述
interfaceids	ID/array	仅返回具有给定 ID 的主机接口。
itemids	ID/array	仅返回给定监控项使用的主机接口。
triggerids	ID/array	仅返回给定触发器中的监控项所使用的主机接口。
selectItems	query	返回一个items 属性，其中包含使用该接口的监控项。 支持 count。
selectHosts	query	返回一个hosts 属性，其中包含使用该接口的主机数组。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectItems。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：interfaceid、dns、ip。
countOutput	boolean	这些参数在参考说明中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回二者之一：

- 一个对象数组；
- 如果使用了 countOutput 参数，则读取的对象数量。

示例

检索主机接口

检索有关主机 “30057” 所使用接口的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.get",
    "params": {
        "output": "extend",
        "hostids": "30057"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "interfaceid": "50039",
            "hostid": "30057",
            "main": "1",
            "type": "1",
            "useip": "0",
            "ip": "",
            "dns": "localhost",
            "port": "10050",
            "available": "0",
            "error": "",
```

```
            "errors_from": "0",
            "disable_until": "0",
            "details": []
        },
        {
            "interfaceid": "55082",
            "hostid": "30057",
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "161",
            "available": "0",
            "error": "",
            "errors_from": "0",
            "disable_until": "0",
            "details": {
                "version": "2",
                "bulk": "0",
                "community": "{$SNMP_COMMUNITY}",
                "max_repetitions": "10"
            }
        }
    ],
    "id": 1
}
```


另见

- 主机
- 监控项

来源

CHostInterface::get() 在 *ui/include/classes/api/services/CHostInterface.php*.

主机组

此类用于处理主机组。

对象参考：

- 主机组

可用方法：

- hostgroup.create - 创建新的主机组
- hostgroup.delete - 删除主机组
- hostgroup.get - 检索主机组
- hostgroup.massadd - 向主机组添加关联对象
- hostgroup.massremove - 从主机组移除关联对象
- hostgroup.propagate - 将权限和标签过滤器传播到主机组的子组
- hostgroup.update - 更新主机组

主机组对象

以下对象与 hostgroup API 直接相关。

主机组

主机组对象具有以下属性。

属性	类型	描述
groupid	ID	主机组的 ID。 属性行为： - 只读 - 更新操作时 必填
name	string	主机组名称。 属性行为： - 创建操作时 必填
flags	integer	主机组的来源。 可能的值： 0 - 普通主机组； 4 - 发现的主机组。 属性行为： - 只读
uuid	string	通用唯一标识符，用于将导入的主机组链接到已存在的主机组。如果未提供，则自动生成。

传播

描述

`object hostgroup.propagate(object parameters)`

此方法允许您将权限和标签过滤器应用于主机组的所有子组。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
groups	object/array	要传播的主机组。 主机组必须定义 `groupid` 属性。 参数行为： - 必需
permissions	boolean	设置为 “true” 以传播权限。 参数行为： - 如果未设置 `tag_filters`，则为 必需
tag_filters	boolean	设置为 “true” 以传播标签过滤器。 参数行为： - 如果未设置 `permissions`，则为 必需

返回值

`(object)` 返回一个对象，其中在 `groupids` 属性下包含传播的主机组 ID。

示例

将主机组权限和标记过滤器传播到其子组

将主机组权限和标记过滤器传播到其子组。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.propagate",
    "params": {
        "groups": [
            {
                "groupid": "6"
            }
        ],
        "permissions": true,
        "tag_filters": true
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "6",
        ]
    },
    "id": 1
}
```

另见

- hostgroup.update
- hostgroup.massadd
- 主机

来源

CHostGroup::propagate() 在 *ui/include/classes/api/services/CHostGroup.php*.

创建

描述

`object hostgroup.create(object/array hostGroups)`

此方法允许您创建新的主机组。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要创建的主机组。

该方法接受带有标准主机组属性的主机组。

返回值

(object) 返回一个对象在其 groupids 属性下包含已创建的主机组的 ID。返回主机组 ID 的顺序与传入的主机组顺序一致。

示例

创建一个主机组

创建一个名为"Linux servers" 的主机组。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.create",
```

```
    "params": {
        "name": "Linux servers"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107819"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::create() 在 *ui/include/classes/api/services/CHostGroup.php*.

删除

描述

`object hostgroup.delete(array hostGroupIds)`

此方法允许您删除主机组。

在以下情况下，无法删除主机组：

- 它包含仅属于该主机组的主机；
- 它被标记为内部；
- 它被主机原型使用；
- 它被全局脚本使用；
- 它被关联条件使用。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色 。

参数

(array) 要删除的主机组的 ID。

返回值

(object) 返回一个对象，在 groupids 属性下包含已删除的主机组的 ID。

示例

删除多个主机组

删除两个主机组。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.delete",
    "params": [
        "107824",
        "107825"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107824",
            "107825"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::delete() 在 *ui/include/classes/api/services/CHostGroup.php*.

批量删除

描述

`object hostgroup.massremove(object parameters)`

此方法允许你从多个主机组中移除相关对象。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 包含要更新的主机组 ID 以及应被移除的对象的参数。

参数	类型	描述
groupids	ID/array	要更新的主机组的 ID。 参数行为: - 必需
hostids	ID/array	要从所有主机组中移除的主机的 ID。

返回值

`(object)` 返回一个对象，其中在 `groupids` 属性下包含更新的主机组 ID。

示例

从主机组中删除主机

从给定的主机组中删除两个主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "hostids": [
            "30050",
            "30001"
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::massRemove() 在 *ui/include/classes/api/services/CHostGroup.php*.

批量添加

描述

`object hostgroup.massadd(object parameters)`

此方法允许您同时将多个相关对象添加到所有指定的主机组中。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 包含要更新的主机组 ID 以及要添加到所有主机组中的对象的参数。

该方法接受以下参数。

参数	Type	说明
groups	object/array	要更新的主机组。 主机组必须仅定义 `groupid` 属性。 参数行为： - 必需
hosts	object/array	要添加到所有主机组中的主机。 主机必须仅定义 `hostid` 属性。

返回值

`(object)` 返回一个对象，在 `groupids` 属性下包含更新的主机组 ID。

示例

将主机添加到主机组

将两台主机添加到 ID 为 5 和 6 的主机组中。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "5"
            },
            {
                "groupid": "6"
```

```
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30001"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

另见

- 主机

来源

CHostGroup::massAdd() 在 *ui/include/classes/api/services/CHostGroup.php*.

更新

描述

`object hostgroup.update(object/array hostGroups)`

此方法允许你更新现有的主机组。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的主机组属性。

必须为每个主机组定义 groupid 属性，所有其他属性均为可选。只有给定的属性会被更新，其他所有属性将保持不变。

返回值

(object) 返回一个对象，在 groupids 属性下包含更新的主机组 ID。

示例

重新命名主机组

将一个主机组重命名成”Linux hosts”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.update",
    "params": {
        "groupid": "7",
```

```
        "name": "Linux hosts"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "7"
        ]
    },
    "id": 1
}
```

重命名多个主机组

将两个主机组重命名，使其名称中包含区域信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.update",
    "params": [
        {
            "groupid": "8",
            "name": "Europe/Linux hosts"
        },
        {
            "groupid": "9",
            "name": "Europe/Windows hosts"
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "8",
            "9"
        ]
    },
    "id": 1
}
```

来源

CHostGroup::update() 在 *ui/include/classes/api/services/CHostGroup.php*.

获取

描述

`integer/array hostgroup.get(object parameters)`

该方法允许您根据给定参数检索主机组。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
graphids	ID/array	仅返回包含具有给定图形的主机的主机群组。
groupids	ID/array	仅返回具有给定主机群组 ID 的主机群组。
hostids	ID/array	仅返回包含给定主机的主机群组。
maintenanceids	ID/array	仅返回受给定维护影响的主机群组。
triggerids	ID/array	仅返回包含具有给定触发器的主机的主机群组。
with_graphs	boolean	仅返回包含具有图形的主机的主机群组。
with_graph_prototypes	boolean	仅返回包含具有图形原型的主机的主机群组。
with_hosts	boolean	仅返回包含主机的主机群组。
with_httptests	boolean	仅返回包含具有 Web 场景的主机的主机群组。 覆盖 with_monitored_httptests 参数。
with_items	boolean	仅返回包含具有监控项的主机的主机群组。 覆盖 with_monitored_items 和 with_simple_graph_items 参数。
with_item_prototypes	boolean	仅返回包含具有监控项原型的主机的主机群组。 覆盖 with_simple_graph_item_prototypes 参数。
with_simple_graph_item_prototypes	boolean	仅返回包含具有监控项原型的主机的主机群组，这些原型已启用创建且信息类型为数值型。
with_monitored_httptests	boolean	仅返回包含具有已启用 Web 场景的主机的主机群组。
with_monitored_hosts	boolean	仅返回包含受监控主机的主机群组。
with_monitored_items	boolean	仅返回包含具有已启用监控项的主机的主机群组。 覆盖 with_simple_graph_items 参数。
with_monitored_triggers	boolean	仅返回包含具有已启用触发器的主机的主机群组。触发器中使用的所有监控项也必须已启用。
with_simple_graph_items	boolean	仅返回包含具有数值型监控项的主机的主机群组。
with_triggers	boolean	仅返回包含具有触发器的主机的主机群组。 覆盖 with_monitored_triggers 参数。
selectDiscoveryRules	query	返回discoveryRules 属性，其中包含发现该主机群组的 LLD 规则。
selectDiscoveryData	query	返回 discoveryData 属性，其中包含主机群组发现对象。 每个主机群组发现对象都是一个链接到已发现主机群组的主机群组原型，并具有以下属性： parent_group_prototypeid - (ID) 发现该主机群组所依据的主机群组原型 ID； name - (string) 主机群组原型名称； ts_delete - (timestamp) 不再被发现的主机群组将被删除的时间； status - (int) 主机群组发现状态： 0 - *(默认)* 主机群组已被发现， 1 - 主机群组不再被发现。
selectHostPrototypes	query	返回hostPrototypes 属性，其中包含发现此主机群组的主机原型。
selectHosts	query	返回hosts 属性，其中包含属于该主机群组的主机。 支持 count。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectHosts - 结果将按 host 排序。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：groupid、name。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	

参数	类型	描述
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroupDiscoveries	query	返回 groupDiscoveries 属性，其中包含主机群组发现对象。 每个主机群组发现对象都是一个链接到已发现主机群组的主机群组原型。 此查询已弃用，请改用 selectDiscoveryData。

返回值

(integer/array) 返回以下之一：

- 对象数组
- 如果使用了 countOutput 参数，则返回已检索对象的数量

示例

按名称检索数据

检索名为 “Zabbix servers” 和 “Linux servers” 的两个主机组的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.get",
    "params": {
        "output": "extend",
        "filter": {
            "name": [
                "Zabbix servers",
                "Linux servers"
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "2",
            "name": "Linux servers",
            "flags": "0",
            "uuid": "dc579cd7a1a34222933f24f52a68bcd8"
        },
        {
            "groupid": "4",
            "name": "Zabbix servers",
            "flags": "0",
            "uuid": "6f6799aa69e844b4b3918f779f2abf08"
        }
    ],
```

```
    "id": 1
}
```

另请参考

- 主机

来源

CHostGroup::get() 在 *ui/include/classes/api/services/CHostGroup.php*.

事件

此类用于处理事件。

对象引用：

- 事件
- 事件标签
- 介质类型 URL

可用方法：

- event.get - 检索事件
- event.acknowledge - 确认事件

事件对象

以下对象与 `event` API 直接相关。

事件

Note:
事件由 Zabbix 服务器创建，不能通过 API 修改。

事件对象具有以下属性。

属性	类型	说明
eventid	ID	事件的 ID。
source	integer	事件类型。 可能的值： 0 - 由触发器创建的事件； 1 - 由发现规则创建的事件； 2 - 由活动 agent 自动注册创建的事件； 3 - 内部事件； 4 - 在服务状态更新时创建的事件。

属性	类型	说明
object	integer	与事件相关的对象类型。 当 source 设置为 “由触发器创建的事件” 时，可能的值： 0 - 触发器。 当 source 设置为 “由发现规则创建的事件” 时，可能的值： 1 - 已发现的主机； 2 - 已发现的服务。 当 source 设置为 “由活动 agent 自动注册创建的事件” 时，可能的值： 3 - 自动注册的主机。 当 source 设置为 “内部事件” 时，可能的值： 0 - 触发器； 4 - 监控项； 5 - LLD 规则。 当 source 设置为 “在服务状态更新时创建的事件” 时，可能的值： 6 - 服务。
objectid	ID	相关对象的 ID。
acknowledged	integer	事件是否已确认。
clock	timestamp	事件创建时间。
ns	integer	事件创建时的纳秒值。
name	string	解析后的事件名称。
value	integer	相关对象的状态。 当 source 设置为 “由触发器创建的事件” 或 “在服务状态更新时创建的事件” 时，可能的值： 0 - OK； 1 - 问题。 当 source 设置为 “由发现规则创建的事件” 时，可能的值： 0 - 主机或服务正常； 1 - 主机或服务不可用； 2 - 主机或服务已发现； 3 - 主机或服务丢失。 当 source 设置为 “内部事件” 时，可能的值： 0 - “正常” 状态； 1 - “未知” 或 “不支持” 状态。 属性行为： - 当 source 设置为 “由触发器创建的事件”、“由发现规则创建的事件”、“内部事件” 或 “在服务状态更新时创建的事件” 时，*supported*
severity	integer	事件当前严重性。 可能的值： 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 严重； 5 - 灾难。
r_eventid	ID	恢复事件的 ID。
c_eventid	ID	在全局事件关联规则下，用于覆盖（关闭）当前事件的事件 ID。请参见 correlationid 以确定具体的关联规则。 仅当事件由全局事件关联规则关闭时，才会定义此参数。
cause_eventid	ID	原因事件的 ID。
correlationid	ID	生成问题关闭操作的关联规则 ID。 仅当事件由全局事件关联规则关闭时，才会定义此参数。
userid	ID	关闭事件的用户 ID（如果事件是手动关闭的）。

属性	类型	说明
suppressed	integer	事件是否被抑制。 可能的值： 0 - 事件处于正常状态； 1 - 事件已被抑制。
opdata	string	带有已展开宏的运行数据。
urls	array	活动的媒介类型 URL。

事件标签

事件标签对象具有以下属性。

属性	类型	描述
tag	string	事件标签名称。
value	string	事件标签值。

媒介类型 URL

媒介类型 URL 对象具有以下属性。

属性	类型	描述
name	string	媒介类型中定义的 URL 名称。
url	string	媒介类型中定义的 URL 值。

结果将仅包含处于活动状态且已启用事件菜单条目的媒介类型的条目。属性中使用的宏将被展开，但如果其中一个属性包含未展开的宏，则这两个属性都将从结果中排除。有关支持的宏，请参见支持的宏。

确认

描述

`object event.acknowledge(object/array parameters)`

此方法允许您更新事件。可以执行以下更新操作：

- 关闭事件。如果事件已经解决，则会跳过此操作。
- 确认事件。如果事件已经确认，则会跳过此操作。
- 取消确认事件。如果事件未被确认，则会跳过此操作。
- 添加消息。
- 更改事件严重性。如果事件已经具有相同的严重性，则会跳过此操作。
- 抑制事件。如果事件已经被抑制，则会跳过此操作。
- 取消抑制事件。如果事件未被抑制，则会跳过此操作。
- 将事件等级更改为原因。如果事件已经被标记为原因，则会跳过此操作。
- 将事件等级更改为症状。如果事件已经被标记为症状，则会跳过此操作。

Attention:
只能更新触发器事件。
 只能更新问题事件。
 关闭事件或更改事件严重性需要触发器的读/写权限。
 要关闭事件，必须在触发器中允许手动关闭。

Note:
此方法对任何类型的用户都可用。可在用户角色设置中撤销调用此方法的权限。有关更多信息，请参见User roles。

参数

`(object/array)` 包含应执行操作的事件 ID 和更新操作的参数。

Parameter	Type	Description
eventids	ID/array	要确认的事件的 ID。 参数行为： - 必需
action	integer	事件更新操作。 可能的位图值： 1 - 关闭问题； 2 - 确认事件； 4 - 添加消息； 8 - 更改严重性； 16 - 取消确认事件； 32 - 抑制事件； 64 - 取消抑制事件； 128 - 将事件等级更改为原因； 256 - 将事件等级更改为症状。 这是一个位掩码字段；可以使用任意可能位图值的总和（例如，使用 6 确认事件并留下消息，或使用 34 确认并抑制事件）。
cause_eventid	ID	参数行为： - 必需 原因事件 ID。
message	string	参数行为： - 如果 `action` 包含 “将事件等级更改为症状” 位，则必需 消息文本。
severity	integer	参数行为： - 如果 `action` 包含 “添加消息” 位，则必需 事件的新严重性。 可能的值： 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。
suppress_until	integer	参数行为： - 如果 `action` 包含 “更改严重性” 位，则必需 必须抑制事件到的 Unix 时间戳。 如果设置为 “0”，则将无限期抑制。 参数行为： - 如果 `action` 包含 “抑制事件” 位，则必需

返回值

(`object`) 返回一个对象，其中包含在 `eventids` 属性下更新后的事件 ID。

示例

确认事件

确认单个事件并留言。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
```

```
    "params": {
        "eventids": "20427",
        "action": 6,
        "message": "Problem resolved."
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427"
        ]
    },
    "id": 1
}
```

更改事件的严重性

更改多个事件的严重性并留言。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": ["20427", "20428"],
        "action": 12,
        "message": "Maintenance required to fix it.",
        "severity": 4
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427",
            "20428"
        ]
    },
    "id": 1
}
```

无限期抑制事件

在没有过期时间的情况下抑制单个事件。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 32,
        "suppress_until": 0
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            20427
        ]
    },
    "id": 1
}
```

取消抑制事件

取消对单个先前已抑制事件的抑制。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 64
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            20427
        ]
    },
    "id": 1
}
```

确认并抑制事件

确认一个或多个事件，添加消息，并在同一请求中将其抑制。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": [
            "20427",
            "20428"
        ],
        "action": 38,
        "message": "Suppressing during planned maintenance.",
        "suppress_until": 0
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            20427,
            20428
```

```
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CEvent.php 中的 CEvent::acknowledge() 。

获取

描述

`integer/array event.get(object parameters)`

该方法允许您根据给定参数检索事件。

Attention:
如果这些事件尚未被 housekeeper 删除，则该方法可能会返回已删除实体的事件。

Note:
该方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

Parameter	Type	Description
eventids	ID/array	仅返回具有给定 ID 的事件。
groupids	ID/array	仅返回由属于给定主机组的对象创建的事件。
hostids	ID/array	仅返回由属于给定主机的对象创建的事件。
objectids	ID/array	仅返回由给定对象创建的事件。
source	integer	仅返回具有给定类型的事件。 有关支持的事件类型列表，请参见event object page。 默认值：0 - 触发器事件。
object	integer	仅返回由给定类型对象创建的事件。 有关支持的对象类型列表，请参见event object page。 默认值：0 - 触发器。
acknowledged	boolean	如果设置为 `true`，则仅返回已确认的事件。
action	integer	仅返回已执行给定event update actions的事件。对于多个操作，请使用任何可接受位图值之和作为位掩码（例如，34 表示确认并抑制事件）。
action_userids	ID/array	仅返回具有执行事件更新操作的用户给定 ID 的事件。
suppressed	boolean	如果设置为 `true`，则仅返回已抑制的事件。
symptom	boolean	如果设置为 `true`，则仅返回症状事件。
severities	integer/array	仅返回具有给定事件严重性的事件。仅在 object 为触发器时适用。
trigger_severities	integer/array	仅返回具有给定触发器严重性的事件。仅在 object 为触发器时适用。
evaltype	integer	标签evaluation method。 可能的值： 0 - *(默认)* And/Or； 2 - Or。

Parameter	Type	Description
tags	array	仅返回具有给定标签的事件。 格式：`[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组返回所有事件。 可能的operator值： 0 - *(默认)* Contains； 1 - Equals； 2 - Does not contain； 3 - Does not equal； 4 - Exists； 5 - Does not exist。
eventid_from	string	仅返回 ID 大于或等于给定 ID 的事件。
eventid_till	string	仅返回 ID 小于或等于给定 ID 的事件。
time_from	timestamp	仅返回在给定时间或之后创建的事件。
time_till	timestamp	仅返回在给定时间或之前创建的事件。
problem_time_from	timestamp	仅返回从 `problem_time_from` 起处于问题状态的事件，而不考虑其当前状态。仅在 source 为触发器事件且 object 为触发器时适用。如果指定了 `problem_time_till`，则此参数为必填。
problem_time_till	timestamp	仅返回直到 `problem_time_till` 为止处于问题状态的事件，而不考虑其当前状态。仅在 source 为触发器事件且 object 为触发器时适用。如果指定了 `problem_time_from`，则此参数为必填。
value	integer/array	仅返回具有给定值的事件。
selectAcknowledges	query	返回一个包含事件更新的 `acknowledges` 属性。事件更新按时间倒序排序。 事件更新对象具有以下属性： `acknowledgeid` - (ID) 确认的 ID； `userid` - (ID) 更新该事件的用户 ID； `clock` - (timestamp) 事件被更新的时间； `message` - (string) 消息文本； `action` - (integer) 已执行的更新操作，参见`event.acknowledge`； `old_severity` - (integer) 此更新操作之前的事件严重性； `new_severity` - (integer) 此更新操作之后的事件严重性； `suppress_until` - (timestamp) 事件将被抑制到的时间； `taskid` - (ID) 如果当前事件正在进行等级变更，则为任务 ID； `maintenanceid` - (ID) 维护的 ID； `username` - (string) 更新该事件的用户用户名； `name` - (string) 更新该事件的用户名字； `surname` - (string) 更新该事件的用户姓氏。 支持 `count`。
selectAlerts	query	返回一个包含事件生成的告警的`alerts` 属性。告警按时间倒序排序。
selectHosts	query	返回一个包含创建该事件的对象所在主机的`hosts` 属性。仅支持由触发器、监控项或 LLD 规则生成的事件。
selectRelatedObject	query	返回一个包含创建该事件的对象的 `relatedObject` 属性。返回的对象类型取决于事件类型。
selectSuppressionData	query	返回一个包含活动维护和手动抑制列表的 `suppression_data` 属性： `maintenanceid` - (ID) 维护的 ID； `userid` - (ID) 抑制该事件的用户 ID； `suppress_until` - (integer) 事件被抑制到的时间。
selectTags	query	返回一个包含事件标签的`tags` 属性。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名，值可以是单个值或用于匹配的值数组。 不支持 `text` data type 的属性。

Parameter	Type	Description
sortfield	string/array	按给定属性对结果排序。 可能的值：`eventid`、`objectid`、`clock`。 与 `groupBy` 一起使用时可能的值：`objectid`。 与 `countOutput` 和 `groupBy` 一起使用时可能的值：`objectid`、`rowscount`。
groupBy	string/array	按给定属性对结果分组。指定的属性将返回在结果中。 可能的值：`objectid`。
countOutput	boolean	这些参数在reference commentary中有说明。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下之一：

- 对象数组
- 如果使用了 `countOutput` 参数，但未使用 `groupBy` 参数，则返回检索到的对象数量
- 如果使用了 `groupBy` 参数，则返回包含聚合结果的对象数组

示例

获取触发器事件

检索触发器"22395" 的最新事件。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
        "selectAcknowledges": "extend",
        "selectSuppressionData": "extend",
        "selectTags": "extend",
        "objectids": "22395",
        "sortfield": ["clock", "eventid"],
        "sortorder": "DESC"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "20",
            "source": "0",
            "object": "0",
            "objectid": "22395",
            "clock": "1728658089",
```

```
            "value": "0",
            "acknowledged": "0",
            "ns": "461809482",
            "name": "Load average is too high (per CPU load over 1.5 for 5m)",
            "severity": "0",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "suppressed": "0",
            "cause_eventid": "0",
            "acknowledges": [],
            "opdata": "Load averages(1m 5m 15m): (1.6328 3.0522 2.2515), # of CPUs: 2",
            "suppression_data": [],
            "tags": [
                {
                    "tag": "scope",
                    "value": "capacity"
                },
                {
                    "tag": "scope",
                    "value": "performance"
                },
                {
                    "tag": "component",
                    "value": "cpu"
                },
                {
                    "tag": "class",
                    "value": "os"
                },
                {
                    "tag": "target",
                    "value": "linux"
                }
            ],
            "urls": []
        },
        {
            "eventid": "4",
            "source": "0",
            "object": "0",
            "objectid": "22395",
            "clock": "1728657737",
            "value": "1",
            "acknowledged": "1",
            "ns": "460759366",
            "name": "Load average is too high (per CPU load over 1.5 for 5m)",
            "severity": "3",
            "r_eventid": "20",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "suppressed": "1",
            "cause_eventid": "0",
            "acknowledges": [
                {
                    "acknowledgeid": "1",
                    "userid": "1",
                    "clock": "1728657938",
                    "message": "Testing environment. Please disregard this alert.",
                    "action": "38",
```

```
                    "old_severity": "0",
                    "new_severity": "0",
                    "suppress_until": "1728744338",
                    "taskid": "0",
                    "maintenanceid": "0",
                    "username": "Admin",
                    "name": "Zabbix",
                    "surname": "Administrator"
                }
            ],
            "opdata": "Load averages(1m 5m 15m): (1.6328 3.0522 2.2515), # of CPUs: 2",
            "suppression_data": [
                {
                    "maintenanceid": "0",
                    "suppress_until": "1728744338",
                    "userid": "1"
                }
            ],
            "tags": [
                {
                    "tag": "scope",
                    "value": "capacity"
                },
                {
                    "tag": "scope",
                    "value": "performance"
                },
                {
                    "tag": "component",
                    "value": "cpu"
                },
                {
                    "tag": "class",
                    "value": "os"
                },
                {
                    "tag": "target",
                    "value": "linux"
                }
            ],
            "urls": []
        }
    ],
    "id": 1
}
```


按时间段检索事件

按时间倒序检索 2012 年 10 月 17 日至 18 日之间创建的所有事件。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
        "time_from": "1350432000",
        "time_till": "1350518400",
        "sortfield": ["clock", "eventid"],
        "sortorder": "DESC"
    },
    "id": 1
}
```

响应：


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "20617",
            "source": "0",
            "object": "0",
            "objectid": "14282",
            "clock": "1350477816",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": " 历史缓存中的可用空间低于 25%",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "suppressed": "0",
            "urls": []
        },
        {
            "eventid": "20616",
            "source": "0",
            "object": "0",
            "objectid": "14281",
            "clock": "1350477814",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Zabbix trapper 进程忙碌度超过 75%",
            "severity": "0",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "suppressed": "0",
            "urls": []
        },
        {
            "eventid": "20615",
            "source": "0",
            "object": "0",
            "objectid": "14281",
            "clock": "1350477541",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": "Zabbix trapper 进程忙碌度超过 75%",
            "severity": "3",
            "r_eventid": "20616",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "suppressed": "0",
```

```
            "urls": []
        }
    ],
    "id": 1
}
```

检索由指定用户确认的事件

检索由 ID=10 的用户确认的事件

请求:

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
        "action": 2,
        "action_userids": [10],
        "selectAcknowledges": ["userid", "action"],
        "sortfield": ["eventid"],
        "sortorder": "DESC"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "503",
            "source": "0",
            "object": "0",
            "objectid": "23162",
            "clock": "1747212236",
            "value": "1",
            "acknowledged": "1",
            "ns": "413470863",
            "name": "Number of installed packages has been changed",
            "severity": "2",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "suppressed": "0",
            "cause_eventid": "0",
            "acknowledges": [
                {
                    "userid": "10",
                    "action": "2"
                }
            ],
            "opdata": "",
            "urls": []
        }
    ],
    "id": 1
}
```

获取问题事件计数最多的前几个触发器

获取严重性为 “警告” “一般” “严重” 或 “灾难” 的前 5 个触发器，以及在指定时间段内的问题事件数量。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "countOutput": true,
        "groupBy": "objectid",
        "source": 0,
        "object": 0,
        "value": 1,
        "time_from": 1672531200,
        "time_till": 1677628800,
        "trigger_severities": [2, 3, 4, 5],
        "sortfield": ["rowscount"],
        "sortorder": "DESC",
        "limit": 5
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "objectid": "232124",
            "rowscount": "27"
        },
        {
            "objectid": "29055",
            "rowscount": "23"
        },
        {
            "objectid": "253731",
            "rowscount": "18"
        },
        {
            "objectid": "254062",
            "rowscount": "11"
        },
        {
            "objectid": "23216",
            "rowscount": "7"
        }
    ],
    "id": 1
}
```

参见

- 告警
- 监控项
- 主机
- LLD 规则
- 服务
- 触发器

来源

ui/include/classes/api/services/CEvent.php 中的 CEvent::get() 。

仪表板

此类用于处理仪表板。

对象引用：

- Dashboard
- Dashboard page
    - Dashboard widget
        - Dashboard widget field
- Dashboard user group
- Dashboard user

可用方法：

- dashboard.create - 创建新的仪表板
- dashboard.delete - 删除仪表板
- dashboard.get - 检索仪表板
- dashboard.update - 更新仪表板

仪表盘对象

下列对象与 Dashboard(仪表盘) API 直接相关。

仪表板

仪表板对象具有以下属性。

Property	Type	Description
dashboardid	ID	仪表板的 ID。 属性行为： - 只读 - 更新操作时为 必需
name	string	仪表板名称。 属性行为： - 创建操作时为 必需
userid	ID	作为仪表板所有者的用户 ID。
private	integer	仪表板共享类型。 可能的值： 0 - 公共仪表板； 1 - (默认) 私有仪表板。
display_period	integer	默认幻灯片放映间隔（秒）。 可能的值：10、30、60、120、600、1800、3600。 默认值：30。
auto_start	integer	自动开始幻灯片放映。 可能的值： 0 - 不自动开始幻灯片放映； 1 - (默认) 自动开始幻灯片放映。

仪表板页面

仪表板页面对象具有以下属性。

Property	Type	Description
dashboard_pageid	ID	仪表板页面的 ID。 属性行为: - 只读

Property	Type	Description
name	string	仪表板页面名称。 默认值：空字符串。
display_period	integer	仪表板幻灯片放映间隔（秒）。 可能的值：0, 10, 30, 60, 120, 600, 1800, 3600。 默认值：0（将使用默认幻灯片放映间隔）。
widgets	array	仪表板小部件对象数组。

仪表板部件

仪表板部件对象具有以下属性。

属性	类型	描述
widgetid	ID	仪表板部件的 ID。 属性行为： - 只读
type	string	仪表板部件的类型。 可能的值： actionlog - 操作日志； clock - 时钟； discovery - 发现状态； favgraphs - 收藏图形； favmaps - 收藏地图； gauge - 仪表； geomap - 地理地图； graph - 图形（经典）； graphprototype - 图形原型； honeycomb - 蜂窝图； hostavail - 主机可用性； hostcard - 主机卡片； hostnavigator - 主机导航器； itemcard - 监控项卡片； itemhistory - 监控项历史； itemnavigator - 监控项导航器； item - 监控项值； map - 地图； navtree - 地图导航树； piechart - 饼图； problemhosts - 故障主机； problems - 故障； problemsbysv - 按严重性分类的故障； scatterplot - 散点图； slareport - SLA 报告； svggraph - 图形； systeminfo - 系统信息； tophosts - 主机排行； topitems - 监控项排行； toptriggers - 触发器排行； trigover - 触发器概览； url - URL； web - Web 监控。 属性行为： - 必需
name	string	自定义部件名称。

属性	类型	描述
x	integer	距仪表板左侧的水平位置。 可能的值范围为 0 到 71。
y	integer	距仪表板顶部的垂直位置。 可能的值范围为 0 到 63。
width	integer	部件宽度。 可能的值范围为 1 到 72。
height	integer	部件高度。 可能的值范围为 1 到 64。
view_mode	integer	部件视图模式。 可能的值： 0 - (默认) 默认部件视图； 1 - 隐藏标题；
fields	array	仪表板部件字段对象数组。 属性行为： - 请参见仪表板部件字段中的各个部件

仪表板小部件字段

仪表板小部件字段对象具有以下属性。

属性	类型	描述
type	integer	小部件字段的类型。 可能的值： 0 - 整数； 1 - 字符串； 2 - 主机组； 3 - 主机； 4 - 监控项； 5 - 监控项原型； 6 - 图形； 7 - 图形原型； 8 - 地图； 9 - 服务； 10 - SLA； 11 - 用户； 12 - 动作； 13 - 介质类型。
name	string	小部件字段名称。 属性行为： - 必需 可能的值：请参见仪表板小部件字段。
value	mixed	取决于类型的小部件字段值。 属性行为： - 必需 可能的值：请参见仪表板小部件字段。 属性行为： - 必需

仪表板用户组

基于用户组的仪表板权限列表。它具有以下属性。

属性	Type	说明
usrgrpid	ID	用户组的 ID。 属性行为: - 必填
permission	integer	权限级别类型。 可能的值： 2 - 只读； 3 - 读写。 属性行为: - 必填

仪表板用户

基于用户的仪表板权限列表。它具有以下属性。

属性	Type	说明
userid	ID	用户的 ID。 属性行为: - 必填
permission	integer	权限级别类型。 可能的值： 2 - 只读； 3 - 读写。 属性行为: - 必填

创建

描述

`object dashboard.create(object/array dashboards)`

此方法允许您创建新的仪表板。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的仪表板。

除了标准仪表板属性之外，该方法还接受以下参数。

参数	类型	说明
pages	array	要为仪表板创建的仪表板页面。仪表板页面将按照指定的顺序进行排序。 参数行为： - 必填
users	array	要在仪表板上创建的仪表板用户共享。
userGroups	array	要在仪表板上创建的仪表板用户组共享。

返回值

(object) 返回一个对象，该对象包含 dashboardids 属性，代表新创建的仪表盘的 ID。返回值中 ID 的顺序与所传递的仪表盘的顺序相匹配。

示例

创建一个仪表盘

创建一个名为"My dashboard" 的仪表盘，仪表盘中包含一个带有标签的问题组件，并使用了两种类型的共享 (用户组和用户)。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problems",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "tags.0.tag",
                                "value": "service"
                            },
                            {
                                "type": 0,
                                "name": "tags.0.operator",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "tags.0.value",
                                "value": "zabbix_server"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": "4",
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

另请参阅

- 仪表板页面
- 仪表板小部件
- 仪表板小部件字段
- 仪表板用户
- 仪表板用户组

来源

ui/include/classes/api/services/CDashboard.php 中的 CDashboard::create()

删除

描述

`object dashboard.delete(array dashboardids)`

此方法允许您删除仪表盘。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 待删除的仪表盘的 ID。

返回值

(object) 返回一个对象，该对象包含 dashboardids 属性，代表被删除的仪表盘的 ID。

示例

删除多个仪表盘

删除两个仪表盘。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.delete",
    "params": [
        "2",
        "3"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2",
            "3"
        ]
    },
```

```
    "id": 1
}
```

来源

ui/include/classes/api/services/CDashboard.php 中的 CDashboard::delete()

获取

描述

`integer/array dashboard.get(object parameters)`

该方法允许你根据给定参数检索仪表板。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
dashboardids	ID/array	仅返回具有给定 ID 的仪表板。
selectPages	query	返回`pages` 属性，其中包含已正确排序的仪表板页面。
selectUsers	query	返回`users` 属性，其中包含与该仪表板共享的用户。
selectUserGroups	query	返回`userGroups` 属性，其中包含与该仪表板共享的用户组。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`dashboardid`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回值存在以下两种可能:

- 一个对象数组；
- 如果指定 `countOutput` 参数，则返回获取到的仪表盘数量。

示例

通过 ID 获取仪表盘

获取 ID 为 1 或 2 的仪表盘的所有数据。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.get",
    "params": {
        "output": "extend",
        "selectPages": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
```

```
        "dashboardids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "1",
            "name": "Dashboard",
            "userid": "1",
            "private": "0",
            "display_period": "30",
            "auto_start": "1",
            "users": [],
            "userGroups": [],
            "pages": [
                {
                    "dashboard_pageid": "1",
                    "name": "",
                    "display_period": "0",
                    "widgets": [
                        {
                            "widgetid": "9",
                            "type": "systeminfo",
                            "name": "",
                            "x": "12",
                            "y": "8",
                            "width": "12",
                            "height": "5",
                            "view_mode": "0",
                            "fields": []
                        },
                        {
                            "widgetid": "8",
                            "type": "problemsbysv",
                            "name": "",
                            "x": "12",
                            "y": "4",
                            "width": "12",
                            "height": "4",
                            "view_mode": "0",
                            "fields": []
                        },
                        {
                            "widgetid": "7",
                            "type": "problemhosts",
                            "name": "",
                            "x": "12",
                            "y": "0",
                            "width": "12",
                            "height": "4",
                            "view_mode": "0",
                            "fields": []
                        },
                        {
                            "widgetid": "6",
```

```
                    "type": "discovery",
                    "name": "",
                    "x": "6",
                    "y": "9",
                    "width": "18",
                    "height": "4",
                    "view_mode": "0",
                    "fields": []
                },
                {
                    "widgetid": "5",
                    "type": "web",
                    "name": "",
                    "x": "0",
                    "y": "9",
                    "width": "18",
                    "height": "4",
                    "view_mode": "0",
                    "fields": []
                },
                {
                    "widgetid": "4",
                    "type": "problems",
                    "name": "",
                    "x": "0",
                    "y": "3",
                    "width": "12",
                    "height": "6",
                    "view_mode": "0",
                    "fields": []
                },
                {
                    "widgetid": "3",
                    "type": "favmaps",
                    "name": "",
                    "x": "8",
                    "y": "0",
                    "width": "12",
                    "height": "3",
                    "view_mode": "0",
                    "fields": []
                },
                {
                    "widgetid": "1",
                    "type": "favgraphs",
                    "name": "",
                    "x": "0",
                    "y": "0",
                    "width": "12",
                    "height": "3",
                    "view_mode": "0",
                    "fields": []
                }
            ]
        },
        {
            "dashboard_pageid": "2",
            "name": "",
            "display_period": "0",
            "widgets": []
        },
        {
```

```
                    "dashboard_pageid": "3",
                    "name": "Custom page name",
                    "display_period": "60",
                    "widgets": []
                }
            ]
        },
        {
            "dashboardid": "2",
            "name": "My dashboard",
            "userid": "1",
            "private": "1",
            "display_period": "60",
            "auto_start": "1",
            "users": [
                {
                    "userid": "4",
                    "permission": "3"
                }
            ],
            "userGroups": [
                {
                    "usrgrpid": "7",
                    "permission": "2"
                }
            ],
            "pages": [
                {
                    "dashboard_pageid": "4",
                    "name": "",
                    "display_period": "0",
                    "widgets": [
                        {
                            "widgetid": "10",
                            "type": "problems",
                            "name": "",
                            "x": "0",
                            "y": "0",
                            "width": "12",
                            "height": "5",
                            "view_mode": "0",
                            "fields": [
                                {
                                    "type": "2",
                                    "name": "groupids",
                                    "value": "4"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


另请参阅

- 仪表板页面
- 仪表板小部件
- 仪表板小部件字段
- 仪表板用户

- 仪表板用户组

来源

ui/include/classes/api/services/CDashboard.php 中的 CDashboard::get()

更新

描述

`object dashboard.update(object/array dashboards)`

此方法允许你更新现有的仪表板。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(`object/array`) 要更新的仪表板属性。

必须为每个仪表板定义 `dashboardid` 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除标准仪表板属性外，该方法还接受以下参数。

参数	类型	说明
pages	array	用于替换现有仪表板页面的仪表板页面。 仪表板页面通过 `dashboard_pageid` 属性进行更新。对于不包含 `dashboard_pageid` 属性的对象，将创建新的仪表板页面；如果现有仪表板页面未被复用，则将被删除。仪表板页面将按照指定的顺序排序。仅会更新仪表板页面中已指定的属性。
users	array	用于替换现有元素的仪表板用户共享。
userGroups	array	用于替换现有元素的仪表板用户组共享。

返回值

(`object`) 返回一个对象，该对象包含 `dashboardids` 属性, 代表被更新的仪表盘的 ID。

示例

重命名仪表盘

将仪表盘重命名为："SQL server status"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "name": "SQL server status"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

更新仪表盘页面

此请求包含以下操作：- 1、重命名第一个仪表盘页面；- 2、替换第二个仪表盘页面的组件；- 3、新增一个仪表盘页面作为第三个仪表盘页面；- 4、删除其他所有仪表盘页面。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "pages": [
            {
                "dashboard_pageid": 1,
                "name": "Renamed Page"
            },
            {
                "dashboard_pageid": 2,
                "widgets": [
                    {
                        "type": "clock",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 3
                    }
                ]
            },
            {
                "display_period": 60
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

修改仪表盘所有者

此请求只允许管理员和超级管理员发起。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "userid": "1"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

另请参阅

- 仪表板页面
- 仪表板小部件
- 仪表板小部件字段
- 仪表板用户
- 仪表板用户组

来源

ui/include/classes/api/services/CDashboard.php 中的 CDashboard::update()

仪表盘组件字段

本页面包含仪表板小部件参数以及相应dashboard widget field 对象可能属性值的导航链接。

要查看每个小部件的参数和属性值，请使用侧边栏导航到相应的小部件页面。

1 动作日志

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置操作日志小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新操作日志小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 操作日志小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
接收者	11	userids.0	用户 ID。 注意：要配置多个用户，请为每个用户创建一个小组件字段对象，并在属性名称中使用递增的数字。
操作	12	actionids.0	操作 ID。 注意：要配置多个操作，请为每个操作创建一个小组件字段对象，并在属性名称中使用递增的数字。

参数		类型	名称	值
媒介类型		13	mediatypeids.0	媒介类型 ID。 注意：要配置多个媒介类型，请为每个媒介类型创建一个小组件字段对象，并在属性名称中使用递增的数字。
状态		0	statuses.0	0 - 处理中； 1 - 已发送/已执行； 2 - 失败。 注意：要配置多个值，请为每个值创建一个小组件字段对象，并在属性名称中使用递增的数字。
搜索字符串		1	message	任意字符串值。
时间段		1	time_period._referenceDASHBOARD._timeperiod - 将仪表板时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其 引用参数设置为 “ABCDE”）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，您也可以仅在 起始时间和 结束时间参数中设置时间段。
	起始时间	1	time_period.from	采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 如果未设置 时间段，则支持 - 如果设置了 time_period.to，则必需
	结束时间	1	time_period.to	采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 如果未设置 时间段，则支持 - 如果设置了 time_period.from，则必需
按以下条件排序条目		0	sort_triggers	3 - 时间（升序）； 4 - *(默认)* 时间（降序）； 5 - 类型（升序）； 6 - 类型（降序）； 7 - 状态（升序）； 8 - 状态（降序）； 11 - 接收者（升序）； 12 - 接收者（降序）。
显示行数		0	show_lines	可能值的范围为 1 到 100。 默认值：25。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 动作日志组件。更多信息，请参考创建仪表盘。

配置一个动作日志组件

配置一个动作日志组件，用于展示按时间倒序排序的 10 个动作操作的详情。同时设置仅展示发送邮件给用户”1”，但是发送失败的动作。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
```

```
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "actionlog",
                        "name": "Action log",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "show_lines",
                                "value": 10
                            },
                            {
                                "type": 0,
                                "name": "sort_triggers",
                                "value": 3
                            },
                            {
                                "type": 11,
                                "name": "userids.0",
                                "value": 1
                            },
                            {
                                "type": 13,
                                "name": "mediatypeids.0",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "statuses.0",
                                "value": 2
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

2 时钟

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置时钟 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新时钟小组件，请参考下表中所述的参数行为。

参数

时钟组件支持以下参数：

参数	类型	参数名称	参数值或参数说明
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - *(默认)* 15 分钟。
时间类型	0	time_type	0 - *(默认)* 本地时间； 1 - 服务器时间； 2 - 主机时间。
时钟类型	0	clock_type	0 - *(默认)* 模拟时钟； 1 - 数字时钟。

当 时间类型设置为” 主机时间” 时，以下参数可用：

参数	类型	参数名称	参数值或参数说明
监控项	4	itemid.0	监控项 ID。 参数行为： - 必需的。

当 时钟类型设置为” 数字时钟” 时，以下参数可用：

参数	类型	参数名称	参数值或参数说明
显示	0	show.0	1 - 日期； 2 - *(默认)* 时间； 3 - 时区。 注意：如果需要配置多个值，则需要给每个值创建一个仪表盘组件字段对象，参数名称 `show.0` 中的数字按照顺序进行递增。

高级配置

当 时钟类型设置为" 数字时钟" 时，以下参数可用：

参数	类型	参数名称	参数值或参数说明
背景颜色	1	bg_color	十六进制颜色代码 (例如：FF0000)。 默认值："" (空)。

日期

如果 时钟类型设置为 "数字"，且 显示设置为 "日期"，则支持以下高级配置参数。

参数	type	name	value
粗体	0	date_bold	0 - *(默认)* 禁用； 1 - 启用。
颜色	1	date_color	十六进制颜色代码（例如 FF0000）。 默认值：""（空）。

时间

如果 *Clock type* 设置为 "Digital"，且 *Show* 设置为 "Time"，则支持以下高级配置参数。

参数	type	name	value
粗体	0	time_bold	0 - *(默认)* 禁用； 1 - 启用。
颜色	1	time_color	十六进制颜色代码（例如 FF0000）。 默认值：""（空）。
秒	0	time_sec	0 - 禁用； 1 - *(默认)* 启用。
格式	0	time_format	0 - *(默认)* 24 小时制； 1 - 12 小时制。

时区

如果 *Clock type* 设置为 "Digital"，且 *Show* 设置为 "Time zone"，则支持以下高级配置参数。

参数	type	name	value
Bold	0	tzone_bold	0 - *(默认)* 禁用； 1 - 启用。
Color	1	tzone_color	十六进制颜色代码（例如 FF0000）。 默认值：""（空）。
Time zone	1	tzone_timezone	有效的时区字符串（例如 `Europe/Riga`、`system`、`UTC` 等）。有关支持的时区完整列表，请参见 PHP documentation。 默认值：`local`。 Parameter behavior： - 如果 *Time type* 设置为 "Local time" 或 "Server time"，则 *supported*

参数	type	name	value
Format	0	tzone_format	0 - *(默认)* 短格式； 1 - 完整。 Parameter behavior： - 如果 *Time type* 设置为 “Local time” 或 “Server time”，则 *supported*

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置时钟组件。更多信息，请参考创建仪表盘。

配置 *Clock* 小组件

配置一个 *Clock* 小组件，以自定义数字时钟的形式显示本地日期、时间和时区。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "clock",
                        "name": "Clock",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "clock_type",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "show.0",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "show.1",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "show.2",
                                "value": 3
                            },
                            {
                                "type": 1,
                                "name": "date_color",
                                "value": "E1E1E1"
                            },
                            {
                                "type": 0,
                                "name": "time_bold",
```

```
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "tzone_color",
                                "value": "E1E1E1"
                            },
                            {
                                "type": 1,
                                "name": "tzone_timezone",
                                "value": "Europe/Riga"
                            },
                            {
                                "type": 0,
                                "name": "tzone_format",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```


参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

3 发现状态

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置发现状态小组件。

参数

发现状态组件支持以下参数：

参数	类型	参数名称	参数值或参数说明
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置发现状态组件。更多信息，请参考创建仪表盘。

配置一个发现状态组件

配置一个发现状态组件，并设置刷新频率为 15 分钟。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "discovery",
                        "name": "Discovery status",
                        "x": 0,
                        "y": 0,
                        "width": 18,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 900
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

4 收藏图形

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置收藏图形小组件。

参数

常用图表组件支持以下参数：

参数	类型	参数名称	参数值或参数说明
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - *(默认)* 15 分钟。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置常用图表组件。更多信息，请参考创建仪表盘。

配置一个常用图表组件

配置一个常用图表组件，并设置刷新频率为 10 分钟。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "favgraphs",
                        "name": "Favorite graphs",
```

```
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 600
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

5 个收藏的地图

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置收藏的地图小组件。

参数

常用拓扑图组件支持以下参数：

参数说明	类型	参数名称	参数值
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - *(默认)* 15 分钟。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置常用拓扑图组件。更多信息，请参考创建仪表盘。

配置一个常用拓扑图组件

配置一个常用拓扑图组件，并设置刷新频率为 10 分钟。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "favmaps",
                        "name": "Favorite maps",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 600
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

6 仪表盘组件

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置*Gauge* 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *Gauge* 小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *Gauge* 部件。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
监控项	4	itemid.0	Item ID。
Item (Widget)	1	itemid._reference	参数行为： - 如果未设置 *Item (Widget)*，则为 必填 用于代替Item ID： ABCDE._itemid - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设为监控项的数据源。
最小值	1	min	参数行为： - 如果未设置 *Item*，则为 必填 任意数值。后缀（例如"1d"、"2w"、"4K"、"8G"）受支持。 默认值："0"。
最大值	1	max	任意数值。后缀（例如"1d"、"2w"、"4K"、"8G"）受支持。 默认值："100"。
值弧	1	value_arc_color	十六进制颜色代码（例如 FF0000）。 默认值：""（空）。

参数	type	name	value
弧背景	1	empty_color	十六进制颜色代码（例如 FF0000）。 默认值："（空）。
背景	1	bg_color	十六进制颜色代码（例如 FF0000）。 默认值："（空）。
显示	0	show.0	1 - 描述； 2 - 值； 3 - 指针； 4 - 刻度； 5 - 值弧。 注意：要配置多个值，请为每个值创建一个 dashboard widget field 对象，并在属性名称中使用递增编号。 默认值：1、2、4、5。 如果同时满足以下两项，则不支持值 “指针” 和 “刻度”： - 未设置值为 “值弧” 的 显示 dashboard widget field 对象； - 显示弧高级配置 参数设置为 “已禁用”。 如果未设置具有相应值的 显示 dashboard widget field 对象，则不支持显示选项的高级配置参数。
覆盖主机	1	override_hostid._reference	ABCDE._hostid - 将兼容部件（其 *Reference* 参数设置为”ABCDE”）设为主机的数据源； DASHBOARD._hostid - 将仪表板主机选择器设为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。

高级配置

以下高级配置参数适用于 *Gauge* 小组件。

Note:
Thresholds 属性名称中的数字（例如 thresholds.0.color）表示按升序排列的列表中的阈值位置。但是，如果阈值的配置顺序不同，则在 Zabbix 前端更新小组件配置后，这些值将按升序排列（例如，"thresholds.0.threshold":"5" → "thresholds.0.threshold":"1"；"thresholds.1.threshold":"1" → "thresholds.1.threshold": "5"）。

参数	类型	名称	值
角度描述	0	angle	可能的值：180（默认）或 270。
描述	1	description	任何字符串值，包括宏。 支持的宏：{HOST.*}、{ITEM.*}、{INVENTORY.*}、用户宏。 默认值：{ITEM.NAME}。
大小	0	desc_size	可能的值范围为 1 至 100。 默认值：15。
垂直位置	0	desc_v_pos	0 - 顶部； 1 - （默认）底部。
粗体	0	desc_bold	0 - （默认）禁用； 1 - 启用。
颜色	1	desc_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。
值			

参数	类型	名称	值
小数位数	0	decimal_places	可能的值范围为 1 至 10。 默认值：2。
大小	0	value_size	可能的值范围为 1 至 100。 默认值：25。
粗体	0	value_bold	0 - （默认）禁用； 1 - 启用。
颜色	1	value_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。
单位			
单位（复选框）	0	units_show	0 - 禁用； 1 - （默认）启用。
单位（值）	1	units	任何字符串值。 参数行为： - 当单位（复选框）设置为“启用”时支持
大小	0	units_size	可能的值范围为 1 至 100。 默认值：25。 参数行为： - 当单位（复选框）设置为“启用”时支持
粗体	0	units_bold	0 - （默认）禁用； 1 - 启用。 参数行为： - 当单位（复选框）设置为“启用”时支持
位置	0	units_pos	0 - 值之前； 1 - 值上方； 2 - （默认）值之后； 3 - 值下方。 参数行为： - 当单位（复选框）设置为“启用”时支持 如果将此参数设置为以下与时间相关的单位之一，则会忽略此参数： unixtime、uptime、s。
颜色	1	units_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。
值弧			
弧大小	0	value_arc_size	可能的值范围为 1 至 100。 默认值：20。
指针			
颜色	1	needle_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。 参数行为： - 当设置了值为“值弧”的 *Show* 小组件字段对象，或将 *Show arc* 设置为“启用”时支持
刻度			
显示单位	0	scale_show_units	0 - 禁用； 1 - （默认）启用。 参数行为： - 当单位（复选框）设置为“启用”，并且设置了值为“值弧”的 *Show* 小组件字段对象，或将 *Show arc* 设置为“启用”时支持

参数	类型	名称	值
大小	0	scale_size	可能的值范围为 1 至 100。 默认值：15。 参数行为： - 当设置了值为 “值弧” 的 *Show* 小组件字段对象，或将 *Show arc* 设置为 “启用” 时支持
小数位数	0	scale_decimal_places	可能的值范围为 1 至 10。 默认值：0。 参数行为： - 当设置了值为 “值弧” 的 *Show* 小组件字段对象，或将 *Show arc* 设置为 “启用” 时支持
阈值			
颜色	1	thresholds.0.color	十六进制颜色代码（例如，FF0000）。
阈值	1	thresholds.0.threshold	任何数值。后缀（例如，“1d”、“2w”、“4K”、“8G”）均受支持。
显示标签	0	th_show_labels	0 - （默认）禁用； 1 - 启用。 参数行为： - 当设置了阈值，并且设置了值为 “值弧” 的 *Show* 小组件字段对象或将 *Show arc* 设置为 “启用” 时支持
显示弧	0	th_show_arc	0 - （默认）禁用； 1 - 启用。 参数行为： - 当设置了阈值时支持
弧大小	0	th_arc_size	可能的值范围为 1 至 100。 默认值：5。 参数行为： - 当显示弧设置为 “启用” 时支持

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置仪表盘 *(Gauge)* 组件。更多信息，请参考创建仪表盘。

配置一个仪表盘 *(Gauge)* 组件

配置一个仪表盘 *(Gauge)* 组件，用于展示监控项”44474” 上由接口 enp0s3 发送的监控项值另外，通过包含阈值在内的多个高级选项对组件进行视觉微调。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "gauge",
                        "name": "Gauge",
                        "x": 0,
                        "y": 0,
                        "width": 18,
```

```
                "height": 5,
                "view_mode": 0,
                "fields": [
                    {
                        "type": 4,
                        "name": "itemid.0",
                        "value": 44474
                    },
                    {
                        "type": 1,
                        "name": "min",
                        "value": "100000"
                    },
                    {
                        "type": 1,
                        "name": "max",
                        "value": "1000000"
                    },
                    {
                        "type": 0,
                        "name": "show.0",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "show.1",
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "show.2",
                        "value": 3
                    },
                    {
                        "type": 0,
                        "name": "show.4",
                        "value": 4
                    },
                    {
                        "type": 0,
                        "name": "show.5",
                        "value": 5
                    },
                    {
                        "type": 0,
                        "name": "angle",
                        "value": 270
                    },
                    {
                        "type": 0,
                        "name": "desc_size",
                        "value": 10
                    },
                    {
                        "type": 0,
                        "name": "desc_bold",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "decimal_places",
                        "value": 0
```

```
        },
        {
            "type": 0,
            "name": "value_bold",
            "value": 1
        },
        {
            "type": 0,
            "name": "units_size",
            "value": 15
        },
        {
            "type": 0,
            "name": "units_pos",
            "value": 3
        },
        {
            "type": 1,
            "name": "needle_color",
            "value": "3C3C3C"
        },
        {
            "type": 1,
            "name": "thresholds.0.color",
            "value": "FF465C"
        },
        {
            "type": 1,
            "name": "thresholds.0.threshold",
            "value": "700000"
        },
        {
            "type": 1,
            "name": "thresholds.1.color",
            "value": "FFD54F"
        },
        {
            "type": 1,
            "name": "thresholds.1.threshold",
            "value": "500000"
        },
        {
            "type": 1,
            "name": "thresholds.2.color",
            "value": "0EC9AC"
        },
        {
            "type": 1,
            "name": "thresholds.2.threshold",
            "value": "100000"
        },
        {
            "type": 0,
            "name": "th_show_labels",
            "value": 1
        },
        {
            "type": 0,
            "name": "th_show_arc",
            "value": 1
        },
        {
```

```
                                "type": 0,
                                "name": "th_arc_size",
                                "value": 15
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

7 地理地图

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置*Geomap*小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *Geomap* 小组件，请参考下表中所述的参数行为。

参数

以下参数适用于 *Geomap* 小组件。

参数		类型	名称	值
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个仪表板小组件字段对象，并在属性名称中使用递增的数字。 如果在模板仪表板上配置小组件，则不支持此参数。
	主机组（小组件）	1	groupids._reference	替代主机组 ID： ABCDE._hostgroupids - 将一个兼容的小组件（其 *Reference* 参数设置为"ABCDE"）设置为主机组的数据源。
主机		3	hostids.0	如果在模板仪表板上配置小组件，则不支持此参数。 主机 ID。 注意：要配置多个主机，请为每个主机创建一个仪表板小组件字段对象，并在属性名称中使用递增的数字。对于多个主机，参数 主机组必须完全不配置，或至少配置一个包含所配置主机的主机组。 如果在模板仪表板上配置小组件，则不支持此参数。
	主机（小组件/仪表板）	1	hostids._reference	替代主机 ID： DASHBOARD.hostids - 将仪表板的主机选择器设置为主机的数据源； ABCDE._hostids - 将一个兼容的小组件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源。
标签				如果在模板仪表板上配置小组件，则不支持此参数。
	评估类型	0	evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	tags.0.tag	如果在模板仪表板上配置小组件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 标签，则为 必需
	运算符	0	tags.0.operator	如果在模板仪表板上配置小组件，则不支持此参数。 0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 标签，则为 必需 如果在模板仪表板上配置小组件，则不支持此参数。

参数		类型	名称	值
	标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 标签，则为 必需
初始视图聚类		1	default_view	如果在模板仪表板上配置小组件，则不支持此参数。 以逗号分隔的 纬度、经度、缩放级别（可选，可能值范围为 0 到 30）。 示例：40.6892494,-74.0466891,10。
	聚类模式	0	clustering_mode	0 - (默认) 自动； 1 - 缩放级别。
	聚类缩放级别	0	clustering_zoom_level	可能值范围为 0 到 30。 默认值：0。 参数行为： - 当 聚类模式设置为 “缩放级别” 时，支持
引用		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的仪表板中必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置地理地图 *(Geomap)* 组件。更多信息，请参考创建仪表盘。

配置一个地理地图 *(Geomap)* 组件

配置一个地理地图 *(Geomap)* 组件，用于展示主机组”2” 和”22” 中符合以下标签配置的主机：标签名称为”component”，标签值为”node”；标签名称为”location”，标签值为”New York”。同时，将地图初始视图设置为：坐标为”40.6892494”(纬度)、”-74.04668991”(经度)，缩放级别为”10”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "geomap",
                        "name": "Geomap",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
```

```
                            "value": 22
                        },
                        {
                            "type": 2,
                            "name": "groupids.1",
                            "value": 2
                        },
                        {
                            "type": 1,
                            "name": "default_view",
                            "value": "40.6892494,-74.0466891,10"
                        },
                        {
                            "type": 0,
                            "name": "evaltype",
                            "value": 2
                        },
                        {
                            "type": 1,
                            "name": "tags.0.tag",
                            "value": "component"
                        },
                        {
                            "type": 0,
                            "name": "tags.0.operator",
                            "value": 0
                        },
                        {
                            "type": 1,
                            "name": "tags.0.value",
                            "value": "node"
                        },
                        {
                            "type": 1,
                            "name": "tags.1.tag",
                            "value": "location"
                        },
                        {
                            "type": 0,
                            "name": "tags.1.operator",
                            "value": 1
                        },
                        {
                            "type": 1,
                            "name": "tags.1.value",
                            "value": "New York"
                        }
                    ]
                }
            ]
        }
    ],
    "userGroups": [
        {
            "usrgrpid": 7,
            "permission": 2
        }
    ],
    "users": [
        {
            "userid": 1,
            "permission": 3
```

```
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

8 图形

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置*Graph* 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新 *Graph* 小组件，请参阅下表中所述的参数行为。

参数

图表组件支持以下参数：

参数	类型	参数名称	参数值或参数说明
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
标识符	1	reference	任意 5 个英文字符组成的字符串 (例如：ABCDE 或 JBPNL)。在这个组件所属的仪表盘中，这个值必须是唯一的。 参数行为： - 必需的。

数据集

以下参数支持用于配置数据集。

Note:
属性名称中的第一个数字（例如 ds.0.hosts.0、ds.0.items.0）表示特定的数据集；如果存在第二个数字，则表示已配置的主机或监控项。

参数		类型	名称	值
数据集类型		0	ds.0.dataset_type	0 - 监控项列表； 1 - (默认) 监控项模式。
	监控项	4	ds.0.itemids.0	监控项 ID。 在模板仪表板上配置小组件时，只应设置模板上配置的监控项。 注意：要配置多个监控项，请为每个监控项创建一个仪表板小组件字段对象，并在属性名称中使用递增的数字。 参数行为： - 当数据集类型设置为 “监控项列表” 且未设置监控项（小组件）时为必需
	监控项（小组件）	1	ds.0.itemids.0._reference	替代监控项 ID： ABCDE._itemid - 将兼容的小组件（其引用参数设置为 “ABCDE”）设置为监控项的数据源。 注意：要配置多个小组件，请为每个小组件创建一个仪表板小组件字段对象，并在属性名称中使用递增的数字。 参数行为： - 当数据集类型设置为 “监控项列表” 且未设置监控项时为必需
	颜色	1	ds.0.color.0	十六进制颜色代码（例如 FF0000）。 参数行为： - 当数据集类型设置为 “监控项列表” 时为必需
	主机模式	1	ds.0.hosts.0	主机名称或模式（例如 “Zabbix*”）。 参数行为： - 当数据集类型设置为 “监控项模式” 时为必需 在模板仪表板上配置小组件时，不支持此参数。
	监控项模式	1	ds.0.items.0	监控项名称或模式（例如 “*：每秒处理的 * 值数量”）。 在模板仪表板上配置小组件时，只应设置模板上配置的监控项模式。 参数行为： - 当数据集类型设置为 “监控项模式” 时为必需
	颜色	1	ds.0.color	十六进制颜色代码（例如 FF0000）。 参数行为： - 当数据集类型设置为 “监控项模式” 且未设置颜色调色板时为支持
	颜色调色板	0	ds.0.color_palette	颜色调色板索引。 可能的值范围为 0 到 11。 默认值：0。 参数行为： - 当数据集类型设置为 “监控项模式” 且未设置颜色时为支持
绘制		0	ds.0.type	0 - (默认) 线条； 1 - 点； 2 - 阶梯； 3 - 柱状图。
堆叠		0	ds.0.stacked	0 - (默认) 已禁用； 1 - 已启用。 参数行为： - 当绘制设置为 “线条”、“阶梯” 或 “柱状图” 时为支持

参数	类型	名称	值
宽度	0	ds.0.width	可能的值范围为 1 到 10。 默认值：1。 参数行为： - 当绘制设置为 “线条” 或 “阶梯” 时为支持
点大小	0	ds.0.pointsize	可能的值范围为 1 到 10。 默认值：3。 参数行为： - 当绘制设置为 “点” 时为支持
透明度	0	ds.0.transparency	可能的值范围为 1 到 10。 默认值：5。
填充	0	ds.0.fill	可能的值范围为 1 到 10。 默认值：3。 参数行为： - 当绘制设置为 “线条” 或 “阶梯” 时为支持
缺失数据	0	ds.0.missingdatafunc	0 - (默认) 无； 1 - 连接； 2 - 视为 0； 3 - 最后已知值。 参数行为： - 当绘制设置为 “线条” 或 “阶梯” 时为支持
覆盖主机	1	ds.0.override_hostid.ABCDE.1	ABCDE._hostid - 将兼容的小组件（其 “引用” 参数设置为 “ABCDE”）设置为主机的数据源； DASHBOARD._hostid - 将仪表板的主机选择器设置为主机的数据源。 在模板仪表板上配置小组件时，不支持此参数。
Y 轴	0	ds.0.axisy	0 - (默认) 左； 1 - 右。
反转值	0	ds.0.invert_values	0 - （默认）已禁用； 1 - 已启用。
时间偏移	1	ds.0.timeshift	有效的时间字符串（例如 3600、1h 等）。 可以使用时间后缀。也允许使用负值。 默认值："" （空）。
聚合函数	0	ds.0.aggregate_function	0 - (默认) 不使用； 1 - 最小值； 2 - 最大值； 3 - 平均值； 4 - 计数； 5 - 总和； 6 - 第一个； 7 - 最后一个。
聚合间隔	1	ds.0.aggregate_interval	有效的时间字符串（例如 3600、1h 等）。 可以使用时间后缀。 默认值：1h。
聚合	0	ds.0.aggregate_grouping	0 - (默认) 每个监控项； 1 - 数据集。 参数行为： - 当聚合函数设置为 “最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个” 或 “最后一个” 时为支持

参数		类型	名称	值
近似		0	ds.0.approximation	1 - 最小值； 2 - (默认) 平均值； 4 - 最大值； 7 - 全部。
数据集标签		1	ds.0.data_set_label	任意字符串值。 默认值："（空）。
监控项标签				
	评估类型	0	ds.0.item_tags_evaltype	0 - (默认) 与/或； 2 - 或。
	标签名称	1	ds.0.item_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需
	运算符	0	ds.0.item_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需
	标签值	1	ds.0.item_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需

显示选项

支持使用以下参数配置显示选项。

参数	类型	名称	值
历史数据选择	0	source	0 - （默认）自动； 1 - 历史记录； 2 - 趋势。
简单触发器	0	simple_triggers	0 - （默认）禁用； 1 - 启用。

参数	类型	名称	值
工作时间	0	working_time	0 - （默认）禁用； 1 - 启用。
标签中的主机名称	0	show_hostnames	0 - （默认）自动； 1 - 显示； 2 - 隐藏。 如果在模板仪表板上配置小组件，则不支持此参数。
百分位线(左)			
状态	0	percentile_left	0 - （默认）禁用； 1 - 启用。
值	0	percentile_left_value	参数行为： - 如果在数据集配置中将 *Y* 轴设置为 “左”，则支持可能的值范围为 1 到 100。
百分位线(右)			
状态	0	percentile_right	0 - （默认）禁用； 1 - 启用。
值	0	percentile_right_value	参数行为： - 如果在数据集配置中将 *Y* 轴设置为 “右”，则支持可能的值范围为 1 到 100。

时间段

以下参数支持用于配置 时间段。

参数	type	name	value
时间段	1	time_period._reference	DASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小部件（其 *Reference* 参数设置为 “ABCDE”）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，您也可以仅在 *From* 和 *To* 参数中设置时间段。

参数	type	name	value
From	1	time_period.from	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则 *supported* - 如果设置了 time_period.to，则 *required*
To	1	time_period.to	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则 *supported* - 如果设置了 time_period.from，则 *required*

坐标轴

以下参数支持用于配置 坐标轴。

参数	type	name	value
左 *Y* 轴	0	lefty	0 - 禁用； 1 - *(*默认*)* 启用。
右 *Y* 轴	0	righty	参数行为： - 如果 *Y* 轴（在数据集 配置中）设置为 “左侧”，则 *supported* 0 - *(*默认*)* 禁用； 1 - 启用。
刻度	0	lefty_scale	参数行为： - 如果 *Y* 轴（在数据集 配置中）设置为 “右侧”，则 *supported* 0 - *(*默认*)* 线性； 1 - 对数。
最小值	1	righty_scale lefty_min	任意数值。 默认值："" （空）。
最大值	1	righty_min lefty_max	任意数值。 默认值："" （空）。
单位（类型）	0	righty_max lefty_units	0 - *(*默认*)* 自动； 1 - 静态。
单位（值）	1	righty_units lefty_static_units	任意字符串值。 默认值："" （空）。
X 轴	0	righty_static_units xaxis	0 - 禁用； 1 - *(*默认*)* 启用。

图例

以下参数支持用于配置图例。

参数	类型	名称	值
显示图例	0	legend	0 - 已禁用； 1 - （默认）已启用。

参数	类型	名称	值
显示最小值/平均值/最大值	0	legend_statistic	0 - （默认）已禁用； 1 - 已启用。 参数行为： - 当显示图例设置为 “已启用” 时支持
显示聚合函数	0	legend_aggregation	0 - （默认）已禁用； 1 - 已启用。 参数行为： - 当显示图例设置为 “已启用” 时支持
行	0	legend_lines_mode	0 - （默认）固定； 1 - 可变。 参数行为： - 当显示图例设置为 “已启用” 时支持
行数/最大行数	0	legend_lines	可能值范围为 1 至 10。 默认值：1。 参数行为： - 当显示图例设置为 “已启用” 时支持
列数	0	legend_columns	可能值范围为 1 至 4。 默认值：4。 参数行为： - 当显示图例设置为 “已启用”，且显示最小值/平均值/最大值设置为 “已禁用” 时支持

问题

配置 问题时支持如下参数：

参数	类型	参数名称	参数值或参数说明
显示问题	0	show_problems	0 - (默认) 禁用； 1 - 启用。
仅选择的监控项问题	0	graph_item_problems	0 - 禁用； 1 - (默认) 启用。
问题主机	1	problemhosts.0	主机名。 注意：参数名称 `problemhosts.0` 中的数字与提供的主机列表需要一致，例如：总数为 3，则下标参数名称中的数字分别为：0、1、2。 如果需要配置多个主机，则需要给每个主机创建一个仪表盘组件字段对象，参数名称 `problemhosts.0` 中的数字按照顺序进行递增。 在仪表盘模板上配置组件时，这个参数不受支持。

参数		类型	参数名称	参数值或参数说明
严重性		0	severities.0	0 - 未定义； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 严重； 5 - 灾难。 默认值：空 (全部启用)。 注意：如果需要配置多个值，则需要给每个值创建一个仪表盘组件字段对象，参数名称 `severities.0` 中的数字按照顺序进行递增。
问题问题标签		1	problem_name	问题事件名称 (不区分大小写，全名或部分名称)。
	评估类型	0	evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	tags.0.tag	任意字符串。 注意：参数名称 `tags.0.tag` 中的数字与提供的参数列表需要一致，例如：总数为 3，则下标参数名称中的数字分别为：0、1、2。 参数行为： - 必需的 (配置 问题标签时)。
	操作	0	tags.0.operator	0 - 包含； 1 - 相等； 2 - 不包含； 3 - 不相等； 4 - 存在； 5 - 不存在。 注意：参数名称 `tags.0.operator` 中的数字与提供的参数列表需要一致，例如：总数为 3，则下标参数名称中的数字分别为：0、1、2。 参数行为： - 必需的 (配置 问题标签时)。
	标签值	1	tags.0.value	任意字符串。 注意：参数名称 `tags.0.value` 中的数字与提供的参数列表需要一致，例如：总数为 3，则下标参数名称中的数字分别为：0、1、2。 参数行为： - 必需的 (配置 问题标签时)。

覆盖

以下参数支持用于配置覆盖。

Note:
属性名称中的第一个数字（例如 or.0.hosts.0、or.0.items.0）表示特定数据集；如果存在第二个数字，则表示已配置的主机或监控项。

参数	类型	名称	值
主机模式	1	or.0.hosts.0	主机名称或模式（例如 Zabbix*）。 如果在模板仪表板上配置小组件，则不支持此参数。 参数行为： - 如果配置覆盖，则为必需
监控项模式	1	or.0.items.0	监控项名称或模式（例如 *: Number of processed *values per second）。 在模板仪表板上配置小组件时，只应设置模板上已配置监控项的模式。 参数行为： - 如果配置覆盖，则为必需
基本颜色	1	or.0.color	十六进制颜色代码（例如 FF0000）。
宽度	0	or.0.width	可能的值范围为 1 到 10。
绘制方式	0	or.0.type	0 - 线条； 1 - 点； 2 - 阶梯； 3 - 柱状图。
透明度	0	or.0.transparency	可能的值范围为 1 到 10。
填充	0	or.0.fill	可能的值范围为 1 到 10。
点大小	0	or.0.pointsize	可能的值范围为 1 到 10。
缺失数据	0	or.0.missingdatafunc	0 - 无； 1 - 连接； 2 - 视为 0； 3 - 最后已知值。
Y 轴	0	or.0.axisy	0 - 左侧； 1 - 右侧。
反转值	0	or.0.invert_values	0 -（默认）禁用； 1 - 启用。
时间偏移	1	or.0.timeshift	有效的时间字符串（例如 3600、1h 等）。 可以使用时间后缀。允许使用负值。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置图表组件。更多信息，请参考创建仪表盘。

配置 *Graph* 小部件

按以下方式配置 *Graph* 小部件：

- 在 1 台主机上配置 2 个数据集，总计 9 个监控项。
- 第一个数据集的类型为 “监控项列表”，由 3 个监控项组成，这些监控项以不同颜色的线条表示，但线宽、透明度和填充相同。
- 第二个数据集的类型为 “监控项模式”，由 6 个监控项组成，配置了聚合，并以具有自定义颜色、宽度、透明度和填充的线条表示。
- 第二个数据集还具有自定义数据集标签。
- 图形中的数据展示最近 3 小时的时间段。
- 图形中的问题仅针对已配置的监控项显示。
- 图形有两个 Y 轴，其中右侧 Y 轴仅显示第二个数据集的值。
- 图形图例以 4 行显示已配置的监控项，以及数据集的最小值、最大值和平均值。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "svggraph",
```

```
                "name": "Graph",
                "x": 0,
                "y": 0,
                "width": 36,
                "height": 5,
                "view_mode": 0,
                "fields": [
                    {
                        "type": 0,
                        "name": "ds.0.dataset_type",
                        "value": 0
                    },
                    {
                        "type": 4,
                        "name": "ds.0.itemids.0",
                        "value": 23264
                    },
                    {
                        "type": 1,
                        "name": "ds.0.color.0",
                        "value": "FF0000"
                    },
                    {
                        "type": 4,
                        "name": "ds.0.itemids.1",
                        "value": 23269
                    },
                    {
                        "type": 1,
                        "name": "ds.0.color.1",
                        "value": "BF00FF"
                    },
                    {
                        "type": 4,
                        "name": "ds.0.itemids.2",
                        "value": 23257
                    },
                    {
                        "type": 1,
                        "name": "ds.0.color.2",
                        "value": "0040FF"
                    },
                    {
                        "type": 0,
                        "name": "ds.0.width",
                        "value": 3
                    },
                    {
                        "type": 0,
                        "name": "ds.0.transparency",
                        "value": 3
                    },
                    {
                        "type": 0,
                        "name": "ds.0.fill",
                        "value": 1
                    },
                    {
                        "type": 1,
                        "name": "ds.1.hosts.0",
                        "value": "Zabbix server"
                    },
```

```
                    {
                        "type": 1,
                        "name": "ds.1.items.0",
                        "value": "*: Number of processed *values per second"
                    },
                    {
                        "type": 1,
                        "name": "ds.1.color",
                        "value": "000000"
                    },
                    {
                        "type": 0,
                        "name": "ds.1.transparency",
                        "value": 0
                    },
                    {
                        "type": 0,
                        "name": "ds.1.fill",
                        "value": 0
                    },
                    {
                        "type": 0,
                        "name": "ds.1.axisy",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "ds.1.aggregate_function",
                        "value": 3
                    },
                    {
                        "type": 1,
                        "name": "ds.1.aggregate_interval",
                        "value": "1m"
                    },
                    {
                        "type": 0,
                        "name": "ds.1.aggregate_grouping",
                        "value": 1
                    },
                    {
                        "type": 1,
                        "name": "ds.1.data_set_label",
                        "value": "Number of processed values per second"
                    },
                    {
                        "type": 0,
                        "name": "graph_time",
                        "value": 1
                    },
                    {
                        "type": 1,
                        "name": "time_period.from",
                        "value": "now-3h"
                    },
                    {
                        "type": 1,
                        "name": "time_period.to",
                        "value": "now"
                    },
                    {
                        "type": 0,
```

```
                                "name": "legend_statistic",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "legend_lines",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "show_problems",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "YZABC"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

9 图形（经典）

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置*Graph (classic)* 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *Graph (classic)* 小组件，请参考下表中所述的参数行为。

参数

以下参数适用于 *Graph (classic)* 部件。

参数		type	name	value
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
来源		0	source_type	0 - *(默认)* 图形； 1 - 简单图形。
图形		6	graphid.0	Graph ID。
	图形（部件）	1	graphid._reference	参数行为： - 如果 来源设置为 “图形”，则为必需 用于代替Graph ID： `ABCDE._graphid` - 将兼容部件（其 *Reference* 参数设置为”ABCDE”）设置为图形的数据源。
监控项		4	itemid.0	参数行为： - 如果 来源设置为 “简单图形” 且未设置 图形，则为必需 Item ID。
	监控项（部件）	1	itemid._reference	参数行为： - 如果 来源设置为 “简单图形” 且未设置 监控项（部件），则为必需 用于代替Item ID： `ABCDE._itemid` - 将兼容部件（其 *Reference* 参数设置为”ABCDE”）设置为监控项的数据源。
时间段		1	time_period._reference	参数行为： - 如果 来源设置为 “简单图形” 且未设置 监控项，则为必需 `DASHBOARD._timeperiod` - 将仪表板时间段选择器设置为数据源； `ABCDE._timeperiod` - 将兼容部件（其 *Reference* 参数设置为”ABCDE”）设置为数据源。 默认值：`DASHBOARD._timeperiod` 或者，也可以仅在 *From* 和 *To* 参数中设置时间段。
	From	1	time_period.from	绝对时间（`YYYY-MM-DD hh:mm:ss`）或相对时间语法（`now`、`now/d`、`now/w-1w` 等）的有效时间字符串。
	To	1	time_period.to	参数行为： - 在未设置 时间段时支持 - 如果设置了 `time_period.to`，则为必需 绝对时间（`YYYY-MM-DD hh:mm:ss`）或相对时间语法（`now`、`now/d`、`now/w-1w` 等）的有效时间字符串。
显示图例		0	show_legend	参数行为： - 在未设置 时间段时支持 - 如果设置了 `time_period.from`，则为必需 0 - 禁用； 1 - *(默认)* 启用。

参数	type	name	value
覆盖主机	1	override_hostid._reference	ABCDE._hostid - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源； DASHBOARD._hostid - 将仪表板主机选择器设置为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。
Reference	1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。该值在部件所属的仪表板内必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置图表 *(经典)* 组件。更多信息，请参考创建仪表盘。

配置一个图表 *(经典)* 组件

配置一个图表 *(经典)* 组件，用于展示监控项"42269"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "graph",
                        "name": "Graph (classic)",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "source_type",
                                "value": 1
                            },
                            {
                                "type": 4,
                                "name": "itemid.0",
                                "value": 42269
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "RSTUV"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
```

```
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

10 图形原型

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置图形原型小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新图形原型小组件，请参考下表中所述的参数行为。

参数

以下参数适用于图形原型小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
来源	0	source_type	2 - *(默认)* 图形原型； 3 - 简单图形原型。
图形原型	7	graphid.0	图形原型 ID。 参数行为： - 当来源设置为 “图形原型” 时必需

参数		类型	名称	值
监控项原型		5	itemid.0	监控项原型 ID。 参数行为： - 当来源设置为 “简单图形原型” 时必需
时间段		1	time_period._reference	DASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其引用参数设置为 “ABCDE”）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，您也可以仅在从和到参数中设置时间段。
	从	1	time_period.from	采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 当未设置时间段时支持 - 当设置了 time_period.to 时必需
	到	1	time_period.to	采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 当未设置时间段时支持 - 当设置了 time_period.from 时必需
显示图例		0	show_legend	0 - 禁用； 1 - *(默认)* 启用。
覆盖主机		1	override_hostid._reference	ABCDE._hostid - 将兼容的小组件（其引用参数设置为 “ABCDE”）设置为主机的数据源； DASHBOARD._hostid - 将仪表板的主机选择器设置为主机的数据源。 如果在模板仪表板上配置小组件，则不支持此参数。
列		0	columns	可能值的范围为 1 至 24。 默认值：2。
行		0	rows	可能值的范围为 1 至 16。 默认值：1。
引用		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的仪表板中必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 图表原型组件。更多信息，请参考创建仪表盘。

配置一个图表原型组件

配置一个图表原型组件，用于展示由低级别自动发现的监控项目原型 (ID: ”42316”) 创建的图表，同时设置布局为 3 个图表 (即 3 列，1 行)。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
```

```
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "graphprototype",
                        "name": "Graph prototype",
                        "x": 0,
                        "y": 0,
                        "width": 48,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "source_type",
                                "value": 3
                            },
                            {
                                "type": 5,
                                "name": "itemid.0",
                                "value": 42316
                            },
                            {
                                "type": 0,
                                "name": "columns",
                                "value": 3
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "OPQWX"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
```

```
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

11 Honeycomb

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置蜂窝图 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新蜂窝图小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *Honeycomb* 部件。

参数		type	名称	值
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该部件，则不支持此参数。
	主机组（部件）	1	groupids._reference	用于代替主机组 ID： ABCDE._hostgroupids - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设为主机组的数据源。
主机		3	hostids.0	如果在模板仪表板上配置该部件，则不支持此参数。 主机 ID。 注意：要配置多个主机，请为每个主机创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。对于多个主机，参数 主机组必须要么完全不配置，要么至少配置一个包含所配置主机的主机组。 如果在模板仪表板上配置该部件，则不支持此参数。
	主机（部件/仪表板）	1	hostids._reference	用于代替主机 ID： DASHBOARD.hostids - 将仪表板主机选择器设为主机的数据源； ABCDE._hostids - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设为主机的数据源。
主机标签				如果在模板仪表板上配置该部件，则不支持此参数。

参数		type	名称	值
	评估类型	0	evaltype_host	0 - (默认) And/Or； 2 - Or。
	标签名称	1	host_tags.0.tag	如果在模板仪表板上配置该部件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 主机标签，则为 必填
	运算符	0	host_tags.0.operator	如果在模板仪表板上配置该部件，则不支持此参数。 0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 主机标签，则为 必填
	标签值	1	host_tags.0.value	如果在模板仪表板上配置该部件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 主机标签，则为 必填
监控项模式		1	items.0	如果在模板仪表板上配置该部件，则不支持此参数。 监控项名称或模式。 注意：要配置多个监控项模式，请为每个监控项模式创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 参数行为： - 必填
监控项标签				
	评估类型	0	evaltype_item	0 - (默认) And/Or； 2 - Or。
	标签名称	1	item_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 监控项标签，则为 必填

参数		type	名称	值
	运算符	0	item_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 监控项标签，则为 必填
	标签值	1	item_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 监控项标签，则为 必填
显示维护中的主机		0	maintenance	0 - (默认) 禁用； 1 - 启用。
显示		0	show.0	1 - 主标签； 2 - 次标签。 注意：要配置多个值，请为每个值创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 默认值：1, 2。
Reference		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。该值在部件所属的仪表板内必须唯一。 参数行为： - 必填

高级配置

以下高级配置参数适用于 *Honeycomb* 小组件。

Note:
Thresholds 属性名称中的数字（例如 thresholds.0.color）表示按升序排列的列表中的阈值位置。但是，如果阈值的配置顺序不同，则在 Zabbix 前端更新小组件配置后，这些值将按升序排列（例如，`"thresholds.0.threshold":"5"` → `"thresholds.0.threshold":"1"`；`"thresholds.1.threshold":"1"` → `"thresholds.1.threshold": "5"`）。

Parameter	type	name	value
主标签 类型	0	primary_label_type	0 - （默认）文本； 1 - 值。
文本	1	primary_label	任意字符串值，包括宏。 支持的宏：{HOST.*}、{ITEM.*}、{INVENTORY.*}、用户宏。 默认值：{HOST.NAME} 参数行为： - 当类型设置为 “文本” 时支持

Parameter	type	name	value
小数位数	0	primary_label_decimal_places	可能的取值范围为 0 到 6。 默认值：2。 参数行为： - 当类型设置为 “值” 时支持
大小（类型）	0	primary_label_size_type	0 - （默认）自动； 1 - 自定义。
大小	0	primary_label_size	可能的取值范围为 1 到 100。 默认值：20。 参数行为： - 当大小（类型）设置为 “自定义” 时支持
粗体	0	primary_label_bold	0 - （默认）禁用； 1 - 启用。
颜色	1	primary_label_color	十六进制颜色代码（例如，FF0000）。 默认值：基于设置对象和用户对象中的 theme： 对于”blue-theme”、”blue-classic-theme” 或”hc-light”，为 1F2C33； 对于”dark-theme”、”dark-classic-theme” 或”hc-dark”，为 EEEEEE； 对于”dark-blue-theme”，为 E5F1FF。
单位（复选框）	0	primary_label_units_show	0 - 禁用； 1 - （默认）启用。 参数行为： - 当类型设置为 “值” 时支持
单位（值）	1	primary_label_units	任意字符串值。 ""（空） 参数行为： - 当类型设置为 “值” 且单位（复选框）设置为 “启用” 时支持
位置	0	primary_label_units_pos	0 - 值之前； 1 - （默认）值之后。 参数行为： - 当类型设置为 “值” 且单位（复选框）设置为 “启用” 时支持 如果设置为以下时间相关单位之一，则忽略此参数：unixtime、uptime、s。
次标签			
类型	0	secondary_label_type	0 - 文本； 1 - （默认）值。
文本	1	secondary_label	任意字符串值，包括宏。 支持的宏：{HOST.*}、{ITEM.*}、{INVENTORY.*}、用户宏。 默认值：{{ITEM.LASTVALUE}.fmtnum(2)} 参数行为： - 当类型设置为 “文本” 时支持
小数位数	0	secondary_label_decimal_places	可能的取值范围为 0 到 6。 默认值：2。 参数行为： - 当类型设置为 “值” 时支持
大小（类型）	0	secondary_label_size_type	0 - （默认）自动； 1 - 自定义。

Parameter	type	name	value
大小	0	secondary_label_size	可能的取值范围为 1 到 100。 默认值：30。 参数行为： - 当大小（类型）设置为 “自定义” 时支持
粗体	0	secondary_label_bold	0 - 禁用； 1 - （默认）启用。
颜色	1	secondary_label_color	十六进制颜色代码（例如，FF0000）。 默认值：基于设置对象和用户对象中的 theme： 对于”blue-theme”、”blue-classic-theme” 或”hc-light”，为 1F2C33； 对于”dark-theme”、”dark-classic-theme” 或”hc-dark”，为 EEEEEE； 对于”dark-blue-theme”，为 E5F1FF。
单位（复选框）	0	secondary_label_units_show	0 - 禁用； 1 - （默认）启用。 参数行为： - 当类型设置为 “值” 时支持
单位（值）	1	secondary_label_units	任意字符串值。 ""（空） 参数行为： - 当类型设置为 “值” 且单位（复选框）设置为 “启用” 时支持
位置	0	secondary_label_position	0 - 值之前； 1 - （默认）值之后。 参数行为： - 当类型设置为 “值” 且单位（复选框）设置为 “启用” 时支持 如果设置为以下时间相关单位之一，则忽略此参数：unixtime、uptime、s。
背景颜色			
背景颜色	1	bg_color	十六进制颜色代码（例如，FF0000）。 默认值：基于设置对象和用户对象中的 theme： 对于”blue-theme” 和”blue-classic-theme”，为 D9E7ED； 对于”dark-theme” 和”dark-classic-theme”，为 3D5059； 对于”dark-blue-theme”，为 004FA8； 对于”hc-light”，为 AAD7E9； 对于”hc-dark”，为 335463。
阈值			
颜色插值	0	interpolation	0 - 禁用； 1 - （默认）启用。
颜色	1	thresholds.0.color	十六进制颜色代码（例如，FF0000）。
阈值	1	thresholds.0.threshold	任意数值。后缀（例如，“1d”、“2w”、“4K”、“8G”）受支持。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 蜂窝图组件。更多信息，请参考创建仪表盘。

配置一个蜂窝图组件

配置一个蜂窝图组件，用于展示 Zabbix server 进程使用率，此外，更改蜂窝单元的主要标签，并使用阈值对组件进行视觉微调。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": "30",
```

```
        "auto_start": "1",
        "pages": [
            {
                "widgets": [
                    {
                        "type": "honeycomb",
                        "name": "Honeycomb",
                        "x": "0",
                        "y": "0",
                        "width": "24",
                        "height": "5",
                        "view_mode": "0",
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
                                "value": 4
                            },
                            {
                                "type": 3,
                                "name": "hostids.0",
                                "value": 10084
                            },
                            {
                                "type": 1,
                                "name": "items.0",
                                "value": "Zabbix server: Utilization*"
                            },
                            {
                                "type": 1,
                                "name": "primary_label",
                                "value": "{ITEM.NAME}"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.0.color",
                                "value": "0EC9AC"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.0.threshold",
                                "value": "0"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.1.color",
                                "value": "FFD54F"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.1.threshold",
                                "value": "70"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.2.color",
                                "value": "FF465C"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.2.threshold",
                                "value": "90"
```

```
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "KSTMQ"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

12 主机可用性

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置主机可用性小组件。

参数

主机可用性组件支持以下参数：

参数		类型	参数名称	参数值或参数说明
刷新频率		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - (默认) 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：如果需要配置多个主机组，则需要给每个主机组创建一个仪表盘组件字段对象，参数名称 `groupids.0` 中的数字按照顺序进行递增。 在仪表盘模板上配置组件时，这个参数不受支持。
	主机组 (组件)	1	groupids._reference	用于替代主机组 ID： `ABCDE._hostgroupids` - 将一个 标识符参数为 ABCDE 的兼容性组件作为数据源。 在仪表盘模板上配置组件时，这个参数不受支持。
接口类型		0	interface_type.0	0 - None； 1 - Zabbix agent (被动模式)； 2 - SNMP； 3 - IPMI； 4 - JMX； 5 - Zabbix agent (主动模式)。 默认值：1、2、3、4、5(均启用)。 注意：如果需要配置多个值，则需要给每个值创建一个仪表盘组件字段对象，参数名称 `interface_type.0` 中的数字按照顺序进行递增。
布局		0	layout	0 - (默认) 水平； 1 - 垂直。
展示维护中的主机		0	maintenance	0 - (默认) 禁用； 1 - 启用。
仅展示总数		0	only_totals	0 - (默认) 禁用； 1 - 启用。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置主机可用性组件。更多信息，请参考创建仪表盘。

配置一个主机可用性组件

配置一个主机可用性组件，用于展示主机组”4” 中配置了”Zabbix agent” 和”SNMP” 接口的主机的可用性。布局为垂直布局。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
```

```
        "pages": [
            {
                "widgets": [
                    {
                        "type": "hostavail",
                        "name": "Host availability",
                        "x": 0,
                        "y": 0,
                        "width": 18,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "interface_type",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "interface_type",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "layout",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
```

```
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

13 主机卡片

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置主机卡片小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新主机卡片小组件，请参考下表中所述的参数行为。

参数

以下参数适用于 主机卡片小部件。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机	3	hostid.0	主机 ID。 参数行为： - 如果未设置 主机（小部件/仪表板），则为 必填
主机（小部件/仪表板）	1	hostid._reference	如果在模板仪表板上配置该小部件，则不支持此参数。 用于代替主机 ID： `DASHBOARD.hostid` - 将仪表板主机选择器设置为主机的数据源； `ABCDE._hostid` - 将兼容的小部件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源。 参数行为： - 如果未设置 主机，则为 必填
显示已抑制的问题	0	show_suppressed	如果在模板仪表板上配置该小部件，则不支持此参数。 0 - (默认) 禁用； 1 - 启用。

参数	type	name	value
显示	0	sections.0	0 - 主机群组； 1 - 描述； 2 - 监控； 3 - 可用性； 4 - 监控方式； 5 - 模板； 6 - 资产记录； 7 - 标签。 注意：属性名称中的数字表示该部分在部分列表中的顺序。要配置多个部分，请为每个部分创建一个仪表板小部件字段对象，并在属性名称中使用递增的数字。

如果 显示设置为 “资产记录”，则支持以下参数。

参数	type	name	value
资产记录字段	0	inventory.0	资产记录 ID。 注意：要配置多个资产记录字段，请为每个资产记录字段创建一个仪表板小部件字段对象，并在属性名称中使用递增的数字。

示例

以下示例仅旨在说明 主机卡片部件的仪表板部件字段对象配置。有关配置仪表板的更多信息，请参见dashboard.create。

配置 主机卡片小部件

配置一个显示以下部分的 主机卡片小部件：“监控”、“可用性”、“监控方式”、“资产记录” 和 “标签”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "hostcard",
                        "name": "Host card",
                        "x": 0,
                        "y": 0,
                        "width": 14,
                        "height": 7,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 3,
                                "name": "hostid.0",
                                "value": 10084
                            },
                            {
                                "type": 0,
                                "name": "show_suppressed",
                                "value": 1
                            },
                            {
```

```
                                "type": 0,
                                "name": "sections.0",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "sections.1",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "sections.2",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "sections.3",
                                "value": 6
                            },
                            {
                                "type": 0,
                                "name": "sections.4",
                                "value": 7
                            },
                            {
                                "type": 0,
                                "name": "inventory.0",
                                "value": 25
                            },
                            {
                                "type": 0,
                                "name": "inventory.1",
                                "value": 26
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
```

```
    },
    "id": 1
}
```

另请参阅

- 仪表板小部件字段
- dashboard.create
- dashboard.update

14 主机导航器

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置主机导航器 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新主机导航器小组件，请参考下表中所述的参数行为。

参数

以下参数适用于主机导航小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机群组	2	groupids.0	主机群组 ID。 注意：要配置多个主机群组，请为每个主机群组创建一个小组件字段对象，并在属性名称中使用递增的数字。 如果在模板仪表板上配置小组件，则不支持此参数。
主机群组（小组件）	1	groupids._reference	替代主机群组 ID： ABCDE._hostgroupids - 将一个兼容的小组件（其引用参数设置为"ABCDE"）设置为主机群组的数据源。 如果在模板仪表板上配置小组件，则不支持此参数。
主机模式	1	hosts.0	主机名称或模式。 注意：要配置多个主机模式，请为每个主机模式创建一个小组件字段对象，并在属性名称中使用递增的数字。对于多个主机模式，参数主机群组必须完全不配置，或至少配置一个包含与所配置主机模式匹配的主机的主机群组。 如果在模板仪表板上配置小组件，则不支持此参数。
主机状态	0	status	-1 - (默认) 任意； 0 - 已启用； 1 - 已禁用。 如果在模板仪表板上配置小组件，则不支持此参数。
主机标签			

参数		类型	名称	值
	评估类型	0	host_tags_evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	host_tags.0.tag	如果在模板仪表板上配置小组件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置主机标签时必需
	运算符	0	host_tags.0.operator	如果在模板仪表板上配置小组件，则不支持此参数。 0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置主机标签时必需
	标签值	1	host_tags.0.value	如果在模板仪表板上配置小组件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置主机标签时必需
严重性		0	severities.0	如果在模板仪表板上配置小组件，则不支持此参数。 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 高； 5 - 灾难。 默认值：空（全部启用）。 注意：要配置多个值，请为每个值创建一个小组件字段对象，并在属性名称中使用递增的数字。
显示处于维护状态的主机		0	maintenance	0 - (默认) 已禁用； 1 - 已启用。
显示问题		0	show_problems	0 - 全部； 1 - (默认) 未抑制； 2 - 无。

参数		类型	名称	值
分组依据				
	属性	0	group_by.0.attribute	0 - 主机群组； 1 - 标签值； 2 - 严重性。 注意：属性名称中的数字表示分组属性列表中的属性顺序。 参数行为： - 配置分组依据时必需
	值	1	group_by.0.tag_name	任意字符串值。 注意：属性名称中的数字表示属性参数中设置的分组属性。 参数行为： - 配置分组依据且属性设置为 “标签值” 时必需
主机限制		0	show_lines	可能值范围为 1 至 9999。 默认值：100。 如果在模板仪表板上配置小组件，则不支持此参数。
引用		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的仪表板中必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 主机导航器组件。更多信息，请参考创建仪表盘。

配置 主机导航器小组件

配置一个 主机导航器小组件，使其显示按主机组分组、然后再按 “city” 标签值分组的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": "30",
        "auto_start": "1",
        "pages": [
            {
                "widgets": [
                    {
                        "type": "hostnavigator",
                        "name": "Host navigator",
                        "x": "0",
                        "y": "0",
                        "width": "12",
                        "height": "5",
                        "view_mode": "0",
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
                                "value": 2
```

```
                            },
                            {
                                "type": 2,
                                "name": "groupids.1",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "group_by.0.attribute",
                                "value": 0
                            },
                            {
                                "type": 0,
                                "name": "group_by.1.attribute",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "group_by.1.tag_name",
                                "value": "city"
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "SWKLB"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置监控项卡片 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新监控项卡片小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 监控项卡片部件。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
监控项	4	itemid.0	Item ID。 参数行为： - 如果未设置 监控项（部件），则为 必填
监控项（部件）	1	itemid._reference	用于代替Item ID： `ABCDE._itemid` - 将一个兼容的部件（其 *Reference* 参数设置为"ABCDE"）设置为监控项的数据源。 参数行为： - 如果未设置 监控项，则为 必填
显示	0	sections.0	0 - 描述； 1 - 错误文本； 2 - 指标； 3 - 最新数据； 4 - 信息类型； 5 - 触发器； 6 - 主机接口； 7 - 类型； 8 - 主机资产记录； 9 - 标签。 注意：属性名称中的数字表示该部分在部分列表中的顺序。要配置多个部分，请为每个部分创建一个仪表板部件字段对象，并在属性名称中使用递增的数字。
覆盖主机	1	override_hostid._reference	`ABCDE._hostid` - 将一个兼容的部件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源； `DASHBOARD._hostid` - 将仪表板的主机选择器设置为主机的数据源。 默认值："" （空） 如果在模板仪表板上配置该部件，则不支持此参数。

迷你图

如果将 显示设置为 “最新数据”，则支持以下参数。

参数	类型	名称	值
宽度	0	sparkline.width	有效值范围为 0 到 10。 默认值：1。
填充	0	sparkline.fill	有效值范围为 0 到 10。 默认值：3。
颜色	1	sparkline.color	十六进制颜色代码（例如：FF0000）。 默认值：42A5F5。
时间段	1	sparkline.time_period.reference	DASHBOARD._timeperiod - 将仪表板时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其 reference 参数等于 ABCDE）设置为数据源。 默认值："" （空）
从	1	sparkline.time_period.from	或者，也可以仅在 从和 到参数中设置时间段。 有效的时间字符串，可以使用绝对时间格式（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）。 默认值：now-1h。 参数行为： - 如果未设置 时间段，则支持 - 如果设置了 sparkline.time_period.to，则必需
到	1	sparkline.time_period.to	有效的时间字符串，可以使用绝对时间格式（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）。 默认值：now。 参数行为： - 如果未设置 时间段，则支持 - 如果设置了 time_period.to，则必需
历史数据	0	sparkline.history	0 - （默认）自动； 1 - 历史数据； 2 - 趋势。

示例

以下示例仅旨在说明 监控项卡片部件的仪表板部件字段对象配置。有关配置仪表板的更多信息，请参见dashboard.create。

配置监控项卡片小部件

配置一个显示以下部分的监控项卡片小部件：“描述”、“最新数据”、“触发器” 和 “标签”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "itemcard",
                        "name": "Item card",
                        "x": 0,
                        "y": 0,
```

```
                        "width": 14,
                        "height": 7,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 4,
                                "name": "itemid.0",
                                "value": 42257
                            },
                            {
                                "type": 0,
                                "name": "sections.0",
                                "value": 0
                            },
                            {
                                "type": 0,
                                "name": "sections.1",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "sections.2",
                                "value": 5
                            },
                            {
                                "type": 0,
                                "name": "sections.3",
                                "value": 9
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```


另请参见

- 仪表板小部件字段
- dashboard.create
- dashboard.update

16 监控项历史记录

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置监控项历史 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新监控项历史小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于监控项历史小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
布局	0	layout	0 - *(默认)* 水平； 1 - 垂直。
列（见下文）			
显示行数	0	show_lines	取值范围为 1 至 100。 默认值：25。
覆盖主机	1	override_hostid._reference	ABCDE._hostid - 将一个兼容的小组件（其引用参数设置为“ABCDE”）设置为主机的数据源； DASHBOARD._hostid - 将仪表板的主机选择器设置为主机的数据源。 如果在模板仪表板上配置小组件，则不支持此参数。
高级配置（见下文）			
引用	1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的仪表板中必须唯一。 参数行为： - 必需

列

列有公共参数，也有由 监控项参数配置确定的额外参数。

Note:
所有列相关的参数中的数字 (例如：columns.0.name) 依赖于配置在哪一列。示例中代表配置第一列。

所有列均支持以下参数：

参数	类型	参数名称	参数值或参数说明
名称	1	columns.0.name	任意字符串。 参数行为： - 必需的。
监控项	4	columns.0.itemid	监控项 ID。 如果是在仪表盘模板上配置组件，则仅允许设置仪表盘模板上存在的监控项。 参数行为： - 必需的。
颜色	1	columns.0.base_color	十六进制颜色代码 (例如：FF0000)。 默认值："" (空)。

当 监控项是一个数值类型的监控项时，支持以下参数：

参数		类型	参数名称	参数值或参数说明
显示		0	columns.0.display	1 - (默认) 原样显示； 2 - 条形图； 3 - 指标。
最小值		1	columns.0.min	任意数值。 参数行为： - 支持 (如果 显示配置为” 条形图” 或” 指标”)。
最大值		1	columns.0.max	任意数值。 参数行为： - 支持 (如果 显示配置为” 条形图” 或” 指标”)。
阈值				
	颜色	1	columns.0.thresholds.0.color	十六进制颜色代码 (例如：FF0000)。
	阈值	1	columns.0.thresholds.0.threshold	任意数值。支持后缀(例如：”1d”、”2w”、”4K”、”8G”)。
历史数据		0	columns.0.history	0 - (默认) 自动； 1 - 历史； 2 - 趋势。

如果配置的 监控项是字符、文本或日志类型的监控项，则支持以下列参数。

参数		type	name	value
高亮				
	高亮	1	columns.0.highlights.0.color	十六进制颜色代码（例如：FF0000）。
	阈值	1	columns.0.highlights.0.pattern	任意正则表达式。
显示		0	columns.0.display	1 - （默认）按原样显示； 4 - HTML； 5 - 单行。
	单行	0	columns.0.max_length	可能的取值范围为 1 到 500。 默认值：100。 参数行为： - 当 显示设置为 “单行” 时支持
使用等宽字体		0	columns.0.monospace_font	0 - （默认）使用默认字体； 1 - 使用等宽字体。

参数	type	name	value
显示本地时间	0	columns.0.local_time	0 - （默认）显示时间戳； 1 - 显示本地时间。 参数行为： - 当 监控项设置为日志类型监控项，且 显示时间戳设置为 “已启用” 时支持

当 监控项是一个二进制类型的监控项时，支持以下参数：

参数	类型	参数名称	参数值或参数说明
显示缩略图	1	columns.0.show_thumbnail	0 - (默认) 禁用； 1 - 启用。

高级配置

以下高级配置参数适用于 监控项历史部件。

参数	type	name	value
新值显示时间戳	0	sortorder	0 - (默认) 顶部； 1 - 底部。
显示列标题	0	show_timestamp	0 - (默认) 禁用； 1 - 启用。
时间段	1	time_period._reference	DASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的部件（其 *Reference* 参数设置为 “ABCDE”）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，也可以仅在 *From* 和 *To* 参数中设置时间段。
From	1	time_period.from	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则 支持 - 如果设置了 time_period.to，则 必需
To	1	time_period.to	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则 支持 - 如果设置了 time_period.from，则 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 监控项历史组件。更多信息，请参考创建仪表盘。

配置一个监控项历史组件

配置一个监控项历史组件，用于展示两个数值类型的监控项”42269” 和”42270” 的最新值此外，将列配置为垂直显示，列名配置为水平显示，限制仅展示 15 行数据，同时包括一个单独的时间戳列。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "itemhistory",
                        "name": "Item history",
                        "x": "0",
                        "y": "0",
                        "width": "18",
                        "height": "6",
                        "view_mode": "0",
                        "fields": [
                            {
                                "type": "0",
                                "name": "layout",
                                "value": "1"
                            },
                            {
                                "type": "1",
                                "name": "columns.0.name",
                                "value": "CPU utilization"
                            },
                            {
                                "type": "4",
                                "name": "columns.0.itemid",
                                "value": "42269"
                            },
                            {
                                "type": "1",
                                "name": "columns.1.name",
                                "value": "Memory utilization"
                            },
                            {
                                "type": "4",
                                "name": "columns.1.itemid",
                                "value": "42270"
                            },
                            {
                                "type": "0",
                                "name": "show_lines",
                                "value": "15"
                            },
                            {
                                "type": "0",
                                "name": "show_timestamp",
                                "value": "1"
                            },
                            {
                                "type": "0",
                                "name": "show_column_header",
                                "value": "1"
                            },
                            {
                                "type": "1",
```

```
                                "name": "reference",
                                "value": "KIVKD"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

17 监控项导航器

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置监控项导航器 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新监控项导航器小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于监控项导航小组件。

参数		类型	名称	值
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个 dashboard 小组件字段对象，并在属性名称中使用递增的数字。 如果在模板 dashboard上配置小组件，则不支持此参数。
	主机组（小组件）	1	groupids._reference	替代主机组 ID： `ABCDE._hostgroupids` - 将一个兼容的小组件（其引用参数设置为“ABCDE”）设置为主机组的数据源。 如果在模板 dashboard上配置小组件，则不支持此参数。
主机		3	hostids.0	主机 ID。 注意：要配置多个主机，请为每个主机创建一个 dashboard 小组件字段对象，并在属性名称中使用递增的数字。对于多个主机，参数主机组必须完全不配置，或至少配置一个所配置主机所属的主机组。 如果在模板 dashboard上配置小组件，则不支持此参数。
	主机（小组件/*dashboard*）	1	hostids._reference	替代主机 ID： `DASHBOARD.hostid` - 将 dashboard 的主机选择器设置为主机的数据源； `ABCDE._hostid` - 将一个兼容的小组件（其引用参数设置为“ABCDE”）设置为主机的数据源。 如果在模板 dashboard上配置小组件，则不支持此参数。
主机标签				
	评估类型	0	host_tags_evaltype	0 - *(默认)* And/Or； 2 - Or。 如果在模板 dashboard上配置小组件，则不支持此参数。
	标签名称	1	host_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置主机标签，则必需 如果在模板 dashboard上配置小组件，则不支持此参数。
	运算符	0	host_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置主机标签，则必需 如果在模板 dashboard上配置小组件，则不支持此参数。

参数		类型	名称	值
	标签值	1	host_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置主机标签，则必需
监控项模式监控项标签		1	items.0	如果在模板 dashboard上配置小组件，则不支持此参数。 监控项名称或模式。 注意：要配置多个监控项模式，请为每个监控项模式创建一个 dashboard 小组件字段对象，并在属性名称中使用递增的数字。
	评估类型	0	item_tags_evaltype	0 - *(默认)* And/Or； 2 - Or。
	标签名称	1	item_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置监控项标签，则必需
	运算符	0	item_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置监控项标签，则必需
	标签值	1	item_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置监控项标签，则必需
状态		0	state	-1 - *(默认)* 全部； 0 - 正常； 1 - 不支持。
显示问题分组依据		0	show_problems	0 - 全部； 1 - *(默认)* 未抑制； 2 - 无。

参数		类型	名称	值
	属性	0	group_by.0.attribute	0 - 主机组； 1 - 主机名称； 2 - 主机标签值； 3 - 监控项标签值。 注意：属性名称中的数字表示分组属性列表中的属性顺序。 参数行为： - 如果配置分组依据，则必需
	值	1	group_by.0.tag_name	任意字符串值。 注意：属性名称中的数字表示属性参数中设置的分组属性。 参数行为： - 如果配置分组依据且属性设置为 “主机标签值” 或 “监控项标签值”，则必需
监控项限制引用		0	show_lines	可能值范围为 1 至 9999。 默认值：100。
		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的 dashboard 中必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 监控项导航器组件。更多信息，请参考创建仪表盘。

配置一个监控项导航器组件

配置一个监控项导航器组件，用于展示按照主机分组后再按照标签”component” 的值分组的 1000 个监控项。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": "30",
        "auto_start": "1",
        "pages": [
            {
                "widgets": [
                    {
                        "type": "itemnavigator",
                        "name": "Item navigator",
                        "x": "0",
                        "y": "0",
                        "width": "12",
                        "height": "5",
                        "view_mode": "0",
                        "fields": [
                            {
                                "type": 0,
                                "name": "group_by.0.attribute",
                                "value": 0
                            },
                            {
```

```
                                "type": 0,
                                "name": "group_by.1.attribute",
                                "value": 3
                            },
                            {
                                "type": 1,
                                "name": "group_by.1.tag_name",
                                "value": "component"
                            },
                            {
                                "type": 0,
                                "name": "show_lines",
                                "value": 1000
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "DFNLK"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

18 监控项值

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置监控项值

小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新监控项值小组件，请参阅下表中所述的参数行为。

参数

以下参数受 监控项值部件支持。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
监控项	4	itemid.0	Item ID。 参数行为： - 如果未设置 监控项（部件），则为 必填
监控项（部件）	1	itemid._reference	用于代替Item ID： ABCDE._itemid - 将兼容的部件（其 引用参数设置为"ABCDE"）设置为监控项的数据源。 参数行为： - 如果未设置 监控项，则为 必填
显示	0	show.0	1 - 描述； 2 - 值； 3 - 时间； 4 - 变化指示器； 5 - 迷你图。 默认值：1、2、3、4。 注意：要配置多个值，请为每个值创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。
覆盖主机	1	override_hostid._reference	ABCDE._hostid - 将兼容的部件（其 引用参数设置为"ABCDE"）设置为主机的数据源； DASHBOARD._hostid - 将仪表板主机选择器设置为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。

高级配置

以下高级配置参数适用于 监控项值部件。

Note:
阈值属性名称中的数字（例如 thresholds.0.color）表示按升序排序的列表中的阈值位置。但是，如果阈值按不同顺序配置，那么在 Zabbix 前端中更新部件配置后，这些值将按升序重新排序（例如 `"thresholds.0.threshold":"5"` → `"thresholds.0.threshold":"1"`；`"thresholds.1.threshold":"1"` → `"thresholds.1.threshold": "5"`）。

参数	type	name	value
背景颜色	1	bg_color	十六进制颜色代码（例如 FF0000）。 默认值："" （空）。

参数		type	name	value
阈值				
	颜色	1	thresholds.0.color	十六进制颜色代码（例如 FF0000）。
	阈值	1	thresholds.0.threshold	任意字符串值。
聚合函数		0	aggregate_function	0 - （默认）不使用； 1 - min； 2 - max； 3 - avg； 4 - count； 5 - sum； 6 - first； 7 - last。
时间周期		1	time_period._reference	DASHBOARD._timeperiod - 将仪表板时间周期选择器设置为数据源； ABCDE._timeperiod - 将兼容的部件（其 reference 参数等于 ABCDE）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，也可以仅在 *From* 和 *To* 参数中设置时间周期。 参数行为： - 如果 聚合函数设置为”min”、”max”、”avg”、”count”、”sum”、”first”、”last”，则 *supported*
	From	1	time_period.from	绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间周期且 聚合函数设置为”min”、”max”、”avg”、”count”、”sum”、”first”、”last”，则 *supported* - 如果设置了 time_period.to，则 *required*
	To	1	time_period.to	绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间周期且 聚合函数设置为”min”、”max”、”avg”、”count”、”sum”、”first”、”last”，则 *supported* - 如果设置了 time_period.from，则 *required*
历史数据		0	history	0 - （默认）自动； 1 - History； 2 - Trends。

描述

如果将 显示设置为 “描述”，则支持以下高级配置参数。

参数	类型	名称	值
描述	1	description	任何字符串值，包括宏。 支持的宏：{HOST.*}、{ITEM.*}、{INVENTORY.*}、用户宏。 默认值：{ITEM.NAME}。
水平位置	0	desc_h_pos	0 - 左侧； 1 - （默认）居中； 2 - 右侧。 两个或多个元素（描述、值、时间）不能共享相同的水平位置和垂直位置。

参数	类型	名称	值
垂直位置	0	desc_v_pos	0 - 顶部； 1 - 中部； 2 - （默认）底部。 两个或多个元素（描述、值、时间）不能共享相同的水平位置和垂直位置。
大小	0	desc_size	可能的值范围为 1 至 100。 默认值：15。
粗体	0	desc_bold	0 - （默认）禁用； 1 - 启用。
颜色	1	desc_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。

值

如果将 显示设置为 “值”，则支持以下高级配置参数。

参数		类型	名称	值
小数位数				
	小数位数	0	decimal_places	可能值范围为 1 到 10。 默认值：2。
	大小	0	decimal_size	可能值范围为 1 到 100。 默认值：35。
位置				
	水平位置	0	value_h_pos	0 - 左侧； 1 - （默认）居中； 2 - 右侧。 两个或更多元素（描述、值、时间）不能共享相同的水平位置和垂直位置。
	垂直位置	0	value_v_pos	0 - 顶部； 1 - （默认）中部； 2 - 底部。 两个或更多元素（描述、值、时间）不能共享相同的水平位置和垂直位置。
	大小	0	value_size	可能值范围为 1 到 100。 默认值：45。
	粗体	0	value_bold	0 - 禁用； 1 - （默认）启用。
	颜色	1	value_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。
单位				
	单位（复选框）	0	units_show	0 - 禁用； 1 - （默认）启用。
	单位（值）	1	units	任意字符串值。

参数	类型	名称	值
位置	0	units_pos	0 - 值之前； 1 - 值上方； 2 - （默认）值之后； 3 - 值下方。
大小	0	units_size	可能值范围为 1 到 100。 默认值：35。
粗体	0	units_bold	0 - 禁用； 1 - （默认）启用。
颜色	1	units_color	十六进制颜色代码（例如，FF0000）。 默认值："（空）。

时间

如果将 显示设置为 “时间”，则支持以下高级配置参数。

参数	type	name	value
水平位置	0	time_h_pos	0 - 左侧； 1 - （默认）居中； 2 - 右侧。 两个或更多元素（描述、值、时间）不能共享相同的 水平位置和 垂直位置。
垂直位置	0	time_v_pos	0 - （默认）顶部； 1 - 中部； 2 - 底部。 两个或更多元素（描述、值、时间）不能共享相同的 水平位置和 垂直位置。
大小	0	time_size	可能值的范围为 1 到 100。 默认值：15。
粗体	0	time_bold	0 - （默认）禁用； 1 - 启用。
颜色	1	time_color	十六进制颜色代码（例如 FF0000）。 默认值："（空）。

变动指标

如果 显示设置为” 变动指标”，则支持以下高级配置参数：

参数	类型	参数名称	参数值或参数说明
变动指标 ↑ 颜色	1	up_color	十六进制颜色代码 (例如：FF0000)。 默认值："" (空)。
变动指标 ↓ 颜色	1	down_color	十六进制颜色代码 (例如：FF0000)。 默认值："" (空)。
变动指标 ↕ 颜色	1	updown_color	十六进制颜色代码 (例如：FF0000)。 默认值："" (空)。

迷你图

如果将 *Show* 设置为 “Sparkline”，则支持以下高级配置参数。

参数		类型	名称	值
宽度		0	sparkline.width	有效值范围为 0 到 10。 默认值：1。
填充		0	sparkline.fill	有效值范围为 0 到 10。 默认值：3。
颜色		1	sparkline.color	十六进制颜色代码（例如，FF0000）。 默认值：42A5F5。
时间段		1	sparkline.time_period.reference	DASHBOARD._timeperiod - 将仪表板时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的部件（其 reference 参数等于 ABCDE）设置为数据源。 默认值：DASHBOARD._timeperiod
	开始时间	1	sparkline.time_period.from	或者，您也可以仅在 *From* 和 *To* 参数中设置时间段。 以绝对时间格式（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）表示的有效时间字符串。 参数行为： - 如果设置了 sparkline.time_period.to，则为必需参数
	结束时间	1	sparkline.time_period.to	以绝对时间格式（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）表示的有效时间字符串。 参数行为： - 如果设置了 sparkline.time_period.from，则为必需参数
历史数据		0	sparkline.history	0 - （默认）自动； 1 - 历史数据； 2 - 趋势数据。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 监控项值组件。更多信息，请参考创建仪表盘。

配置一个监控项值组件

配置一个监控项值组件，用于展示监控项"42266" 的值 (Zabbix agent 可用)。此外，通过包含动态背景颜色在内的多个高级配置对组件进行视觉微调。基于 Zabbix agent 可用性修改监控项值。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "item",
                        "name": "Item value",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
```

```
            "type": 4,
            "name": "itemid.0",
            "value": 42266
        },
        {
            "type": 0,
            "name": "show.0",
            "value": 1
        },
        {
            "type": 0,
            "name": "show.1",
            "value": 2
        },
        {
            "type": 0,
            "name": "show.2",
            "value": 3
        },
        {
            "type": 1,
            "name": "description",
            "value": "Agent status"
        },
        {
            "type": 0,
            "name": "desc_h_pos",
            "value": 0
        },
        {
            "type": 0,
            "name": "desc_v_pos",
            "value": 0
        },
        {
            "type": 0,
            "name": "desc_bold",
            "value": 1
        },
        {
            "type": 1,
            "name": "desc_color",
            "value": "F06291"
        },
        {
            "type": 0,
            "name": "value_h_pos",
            "value": 0
        },
        {
            "type": 0,
            "name": "value_size",
            "value": 25
        },
        {
            "type": 1,
            "name": "value_color",
            "value": "FFFF00"
        },
        {
            "type": 0,
            "name": "units_show",
```

```
                            "value": 0
                        },
                        {
                            "type": 0,
                            "name": "time_h_pos",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "time_v_pos",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "time_size",
                            "value": 10
                        },
                        {
                            "type": 0,
                            "name": "time_bold",
                            "value": 1
                        },
                        {
                            "type": 1,
                            "name": "time_color",
                            "value": "9FA8DA"
                        },
                        {
                            "type": 1,
                            "name": "thresholds.0.color",
                            "value": "E1E1E1"
                        },
                        {
                            "type": 1,
                            "name": "thresholds.0.threshold",
                            "value": "0"
                        },
                        {
                            "type": 1,
                            "name": "thresholds.1.color",
                            "value": "D1C4E9"
                        },
                        {
                            "type": 1,
                            "name": "thresholds.1.threshold",
                            "value": "1"
                        }
                    ]
                }
            ]
        }
    ],
    "userGroups": [
        {
            "usrgrpid": 7,
            "permission": 2
        }
    ],
    "users": [
        {
            "userid": 1,
            "permission": 3
```

```
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

19 拓扑图

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置地图 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这允许用户修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新地图小组件，请参阅下表中所述的参数行为。

参数

拓扑图组件支持以下参数：

参数	类型	参数名称	参数值或参数说明
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - (默认) 15 分钟。
拓扑图	8	sysmapid.0	拓扑图 ID。 参数行为： - 必需的 (如果 拓扑图 (组件) 未设置)。
拓扑图 (组件)	1	sysmapid._reference	ABCDE._mapid - 将一个 标识符参数为 ABCDE 的拓扑图导航树组件作为数据源。 参数行为： - 必需的 (如果 拓扑图未设置)。
标识符	1	reference	任意 5 个英文字符组成的字符串 (例如：ABCDE 或 JBPNL)。在这个组件所属的仪表盘中，这个值必须是唯一的。 参数行为： - 必需的。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 拓扑图组件。更多信息，请参考创建仪表盘。

配置一个拓扑图组件

配置一个拓扑图组件，用于展示拓扑图"1"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "map",
                        "name": "Map",
                        "x": 0,
                        "y": 0,
                        "width": 54,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 8,
                                "name": "sysmapid.0",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

配置一个链接的拓扑图组件

配置一个拓扑图组件，链接到一个拓扑图导航树组件。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "map",
                        "name": "Map",
                        "x": 0,
                        "y": 5,
                        "width": 54,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "sysmapid._reference",
                                "value": "ABCDE._mapid"
                            }
                        ]
                    },
                    {
                        "type": "navtree",
                        "name": "Map navigation tree",
                        "x": 0,
                        "y": 0,
                        "width": 18,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "navtree.1.name",
                                "value": "Element A"
                            },
                            {
                                "type": 1,
                                "name": "navtree.2.name",
                                "value": "Element B"
                            },
                            {
                                "type": 1,
                                "name": "navtree.3.name",
                                "value": "Element C"
                            },
                            {
                                "type": 1,
                                "name": "navtree.4.name",
                                "value": "Element A1"
                            },
                            {
                                "type": 1,
                                "name": "navtree.5.name",
```

```
            "value": "Element A2"
        },
        {
            "type": 1,
            "name": "navtree.6.name",
            "value": "Element B1"
        },
        {
            "type": 1,
            "name": "navtree.7.name",
            "value": "Element B2"
        },
        {
            "type": 0,
            "name": "navtree.4.parent",
            "value": 1
        },
        {
            "type": 0,
            "name": "navtree.5.parent",
            "value": 1
        },
        {
            "type": 0,
            "name": "navtree.6.parent",
            "value": 2
        },
        {
            "type": 0,
            "name": "navtree.7.parent",
            "value": 2
        },
        {
            "type": 0,
            "name": "navtree.1.order",
            "value": 1
        },
        {
            "type": 0,
            "name": "navtree.2.order",
            "value": 2
        },
        {
            "type": 0,
            "name": "navtree.3.order",
            "value": 3
        },
        {
            "type": 0,
            "name": "navtree.4.order",
            "value": 1
        },
        {
            "type": 0,
            "name": "navtree.5.order",
            "value": 2
        },
        {
            "type": 0,
            "name": "navtree.6.order",
            "value": 1
        },
```

```
                            {
                                "type": 0,
                                "name": "navtree.7.order",
                                "value": 2
                            },
                            {
                                "type": 8,
                                "name": "navtree.6.sysmapid",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "ABCDE"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘
- 拓扑图导航树

20 地图导航树

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置地图导航树小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新地图导航树小组件，请参阅下表中所述的参数行为。

参数

以下参数支持用于 *Map navigation tree* 小组件。

Parameter	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新; 10 - 10 秒; 30 - 30 秒; 60 - 1 分钟; 120 - 2 分钟; 600 - 10 分钟; 900 - *(默认)* 15 分钟.
显示不可用的地图	1	show_unavailable	0 - *(默认)* 禁用; 1 - 启用.
引用	1	reference	任意由 5 个字符组成的字符串值（例如，`ABCDE` 或 `JBPNL`）。此值在小组件所属的前端中必须唯一。 参数行为: - 必需

以下参数支持用于配置地图导航树元素。

Parameter	type	name	value
名称	1	navtree.1.name	任意字符串值。
链接的地图	8	navtree.1.sysmapid	注意：属性名称中的数字用于设置元素编号。 地图 ID。
用于创建元素层级的参数	0	navtree.1.parent	注意：属性名称中的数字引用的是地图所链接到的元素。 父元素编号。
	0	navtree.1.order	注意：属性名称中的数字引用的是子元素。属性值引用的是父元素。 地图导航树中的元素位置。 注意：属性名称中的数字引用的是元素编号。属性值引用的是该元素在地图导航树中的位置。父元素位置在整个地图导航树中确定。子元素位置在父元素内确定。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 拓扑图导航树组件。更多信息，请参考创建仪表盘。

配置一个拓扑图导航树组件

配置一个拓扑图导航树组件，用于展示如下拓扑图导航树：

- Element A
  - Element A1
  - Element A2
- Element B
  - Element B1 (包含一个拓扑图的链接"1"，可通过链接的拓扑图组件展示)
  - Element B2
- Element C

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
```

```
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "navtree",
                        "name": "Map navigation tree",
                        "x": 0,
                        "y": 0,
                        "width": 18,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "navtree.1.name",
                                "value": "Element A"
                            },
                            {
                                "type": 1,
                                "name": "navtree.2.name",
                                "value": "Element B"
                            },
                            {
                                "type": 1,
                                "name": "navtree.3.name",
                                "value": "Element C"
                            },
                            {
                                "type": 1,
                                "name": "navtree.4.name",
                                "value": "Element A1"
                            },
                            {
                                "type": 1,
                                "name": "navtree.5.name",
                                "value": "Element A2"
                            },
                            {
                                "type": 1,
                                "name": "navtree.6.name",
                                "value": "Element B1"
                            },
                            {
                                "type": 1,
                                "name": "navtree.7.name",
                                "value": "Element B2"
                            },
                            {
                                "type": 0,
                                "name": "navtree.4.parent",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "navtree.5.parent",
                                "value": 1
                            },
                            {
```

```
                            "type": 0,
                            "name": "navtree.6.parent",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "navtree.7.parent",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "navtree.1.order",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "navtree.2.order",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "navtree.3.order",
                            "value": 3
                        },
                        {
                            "type": 0,
                            "name": "navtree.4.order",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "navtree.5.order",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "navtree.6.order",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "navtree.7.order",
                            "value": 2
                        },
                        {
                            "type": 8,
                            "name": "navtree.6.sysmapid",
                            "value": 1
                        },
                        {
                            "type": 1,
                            "name": "reference",
                            "value": "HJQXF"
                        }
                    ]
                }
            ]
        }
    ],
    "userGroups": [
        {
            "usrgrpid": 7,
```

```
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘
- 拓扑图

21 饼图

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置饼图小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新饼图小组件，请参阅下表中所述的参数行为。

参数

饼图组件支持以下参数：

参数	类型	参数名称	参数值或参数说明
刷新频率	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。

数据集

以下参数支持用于配置数据集。

Note:
属性名称中的第一个数字（例如 ds.0.hosts.0、ds.0.items.0）表示特定的数据集；如果存在第二个数字，则表示已配置的主机或监控项。

参数	类型	名称	值
数据集类型	0	ds.0.dataset_type	0 - 监控项列表； 1 - *(默认)* 监控项模式。
监控项	4	ds.0.itemids.0	监控项 ID。 在模板仪表板上配置小组件时，只应设置模板上配置的监控项。 注意：要配置多个监控项，请为每个监控项创建一个仪表板小组件字段对象，并在属性名称中使用递增的数字。 参数行为： - 当数据集类型设置为 “监控项列表” 且未设置监控项（小组件）时为必需
监控项（小组件）	1	ds.0.itemids.0._reference	替代监控项 ID： `ABCDE._itemid` - 将一个兼容的小组件（其引用参数设置为 “ABCDE”）设置为监控项的数据源。 注意：要配置多个小组件，请为每个小组件创建一个仪表板小组件字段对象，并在属性名称中使用递增的数字。 参数行为： - 当数据集类型设置为 “监控项列表” 且未设置监控项时为必需
颜色	1	ds.0.color.0	十六进制颜色代码（例如 FF0000）。 参数行为： - 当数据集类型设置为 “监控项列表” 时为必需
监控项类型	0	ds.0.type.0	0 - *(默认)* 普通； 1 - 总计。 整个图表中只能为一个监控项设置 “总计” 值。 参数行为： - 当数据集类型设置为 “监控项列表” 时支持
主机模式	1	ds.0.hosts.0	主机名称或模式（例如 “Zabbix*”）。 参数行为： - 当数据集类型设置为 “监控项模式” 时为必需 在模板仪表板上配置小组件时，不支持此参数。
监控项模式	1	ds.0.items.0	监控项名称或模式（例如 “*: 每秒处理的 * 值数量”）。 在模板仪表板上配置小组件时，只应设置模板上配置的监控项模式。 参数行为： - 当数据集类型设置为 “监控项模式” 时为必需
颜色	1	ds.0.color	十六进制颜色代码（例如 FF0000）。 参数行为： - 当数据集类型设置为 “监控项模式” 且未设置颜色调色板时支持

参数		类型	名称	值
	颜色调色板	0	ds.0.color_palette	颜色调色板索引。 可能的值范围为 0 到 11。 默认值：0。 参数行为： - 当数据集类型设置为 “监控项模式” 且未设置颜色时支持
聚合函数		0	ds.0.aggregate_function	1 - 最小值； 2 - 最大值； 3 - 平均值； 4 - 计数； 5 - 总和； 6 - 第一个； 7 - (默认) 最后一个。
数据集聚合		0	ds.0.dataset_aggregation	0 (默认) 无； 1 - 最小值； 2 - 最大值； 3 - 平均值； 4 - 计数； 5 - 总和。 参数行为： - 当监控项类型设置为 “总计” 时支持
数据集标签监控项标签		1	ds.0.data_set_label	任意字符串值。 默认值："" (空)。
	评估类型	0	ds.0.item_tags_evaltype	0 (默认) 与/或； 2 - 或。
	标签名称	1	ds.0.item_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需
	运算符	0	ds.0.item_tags.0.operator	0 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需
	标签值	1	ds.0.item_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需

显示选项

以下参数支持配置显示选项。

参数		类型	名称	值
历史数据选择绘制方式		0	source	0 - （默认）自动； 1 - 历史数据； 2 - 趋势数据。
		0	draw_type	0 - （默认）饼图； 1 - 环形图。
	宽度	0	width	20 - 半径的 20%； 30 - 半径的 30%； 40 - 半径的 40%； 50 - （默认）半径的 50%。
	描边宽度	0	stroke	参数行为： - 当绘制方式设置为 “环形图” 时支持 可能值范围为 0 至 10。 默认值：0。
	显示总值	0	total_show	参数行为： - 当绘制方式设置为 “环形图” 时支持 0 - （默认）禁用； 1 - 启用。
	大小	0	value_size_type	参数行为： - 当绘制方式设置为 “环形图” 时支持 0 - （默认）自动； 1 - 自定义。
	大小（自定义大小的值）	0	value_size	参数行为： - 当显示总值设置为 “启用” 时支持 可能值范围为 1 至 100。 默认值：20。
	小数位数	0	decimal_places	参数行为： - 当显示总值设置为 “启用” 时支持 可能值范围为 0 至 6。 默认值：2。
	单位（复选框）	0	units_show	参数行为： - 当显示总值设置为 “启用” 时支持 0 - （默认）禁用； 1 - 启用。
	单位（值）	1	units	参数行为： - 当显示总值设置为 “启用” 时支持 任意字符串值。 参数行为： - 当单位（复选框）设置为 “启用” 时支持

参数		类型	名称	值
	粗体	0	value_bold	0 - （默认）禁用； 1 - 启用。
	颜色	1	value_color	参数行为： - 当显示总值设置为 “启用” 时支持 十六进制颜色代码（例如：FF0000）。
扇区之间的间距		0	space	参数行为： - 当显示总值设置为 “启用” 时支持 可能值范围为 0 至 10。 默认值：1。
合并小于 *N*% 的扇区（复选框）		0	merge	0 - （默认）禁用； 1 - 启用。
合并小于 *N*% 的扇区（值）		0	merge_percent	可能值范围为 1 至 10。 默认值：1。 参数行为： - 当合并小于 *N*% 的扇区（复选框）设置为 “启用” 时支持
合并小于 *N*% 的扇区（颜色）		1	merge_color	十六进制颜色代码（例如：FF0000）。 参数行为： - 当合并小于 *N*% 的扇区（复选框）设置为 “启用” 时支持

时间段

以下参数支持用于配置 时间段。

参数	type	name	value
时间段	1	time_period._referenced	DASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的部件（其 *Reference* 参数设置为"ABCDE"）设置为数据源。 默认值：DASHBOARD._timeperiod
From	1	time_period.from	或者，您也可以仅在 *From* 和 *To* 参数中设置时间段。 绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则 *supported* - 如果设置了 time_period.to，则 *required*
To	1	time_period.to	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则 *supported* - 如果设置了 time_period.from，则 *required*

图例

以下参数支持配置图例。

参数	类型	名称	值
显示图例	0	legend	0 - 已禁用； 1 - （默认）已启用。
显示值	0	legend_value	0 - （默认）已禁用； 1 - 已启用。 参数行为： - 当显示图例设置为 "已启用" 时支持
显示聚合函数	0	legend_aggregation	0 - （默认）已禁用； 1 - 已启用。 参数行为： - 当显示图例设置为 "已启用" 时支持
行	0	legend_lines_mode	0 - （默认）固定； 1 - 可变。 参数行为： - 当显示图例设置为 "已启用" 时支持
行数/ 最大行数	0	legend_lines	可选值范围为 1 至 10。 默认值：1。 参数行为： - 当显示图例设置为 "已启用" 时支持
列数	0	legend_columns	可选值范围为 1 至 4。 默认值：4。 参数行为： - 当显示图例设置为 "已启用"，且显示值设置为 "已禁用" 时支持

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 饼图组件. 更多信息，请参考创建仪表盘。

配置 饼图部件

按以下方式配置 饼图部件：

- 在 1 台主机上为总计 9 个监控项配置 2 个数据集。
- 第一个数据集的类型为 “监控项列表”，由 3 个监控项组成，这些监控项的类型均为 “普通”，并以不同颜色表示。
- 第二个数据集的类型为 “监控项模式”，由 6 个监控项组成，为每个监控项配置了聚合，并以自定义颜色表示。
- 第二个数据集还具有自定义数据集标签。
- 饼图中的数据以环形图形式显示，具有自定义宽度，并在中心显示带单位的总值。
- 饼图中的数据会针对最近 3 小时的自定义时间段进行显示和聚合。
- 饼图图例以 4 行显示已配置的监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "piechart",
                        "name": "Pie chart",
                        "x": 0,
                        "y": 0,
                        "width": 24,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "ds.0.dataset_type",
                                "value": 0
                            },
                            {
                                "type": 4,
                                "name": "ds.0.itemids.0",
                                "value": 23264
                            },
                            {
                                "type": 1,
                                "name": "ds.0.color.0",
                                "value": "FF0000"
                            },
                            {
                                "type": 0,
                                "name": "ds.0.type.0",
                                "value": 0
                            },
                            {
                                "type": 4,
                                "name": "ds.0.itemids.1",
                                "value": 23269
                            },
                            {
                                "type": 1,
                                "name": "ds.0.color.1",
                                "value": "BF00FF"
                            },
                            {
                              "type": 0,
```

```
            "name": "ds.0.type.1",
            "value": 0
        },
        {
            "type": 4,
            "name": "ds.0.itemids.2",
            "value": 23257
        },
        {
            "type": 1,
            "name": "ds.0.color.2",
            "value": "0040FF"
        },
        {
            "type": 0,
            "name": "ds.0.type.2",
            "value": 0
        },
        {
            "type": 1,
            "name": "ds.1.hosts.0",
            "value": "Zabbix server"
        },
        {
            "type": 1,
            "name": "ds.1.items.0",
            "value": "*: Number of processed *values per second"
        },
        {
            "type": 1,
            "name": "ds.1.color",
            "value": "000000"
        },
        {
            "type": 0,
            "name": "ds.1.aggregate_function",
            "value": 3
        },
        {
            "type": 1,
            "name": "ds.1.data_set_label",
            "value": "Number of processed values per second"
        },
        {
            "type": 0,
            "name": "draw_type",
            "value": 1
        },
        {
            "type": 0,
            "name": "width",
            "value": 30
        },
        {
            "type": 0,
            "name": "total_show",
            "value": 1
        },
        {
            "type": 0,
            "name": "units_show",
            "value": 1
```

```
                            },
                            {
                                "type": 0,
                                "name": "graph_time",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "time_period.from",
                                "value": "now-3h"
                            },
                            {
                                "type": 1,
                                "name": "time_period.to",
                                "value": "now"
                            },
                            {
                                "type": 0,
                                "name": "legend_lines",
                                "value": 4
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

22 问题主机

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置问题主机小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新问题主机小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *Problem hosts* 部件。

参数		type	name	value
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该部件，则不支持此参数。
	主机组（部件）	1	groupids._reference	用于代替主机组 ID： ABCDE._hostgroupids - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设置为主机组的数据源。
排除主机组		2	exclude_groupids.0	如果在模板仪表板上配置该部件，则不支持此参数。 主机组 ID。 注意：要排除多个主机组，请为每个主机组创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。
主机		3	hostids.0	如果在模板仪表板上配置该部件，则不支持此参数。 主机 ID。 注意：要配置多个主机，请为每个主机创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。对于多个主机，参数 主机组必须要么完全不配置，要么至少配置一个包含所配置主机的主机组。
	主机（部件/仪表板）	1	hostids._reference	如果在模板仪表板上配置该部件，则不支持此参数。 用于代替主机 ID： DASHBOARD.hostids - 将仪表板主机选择器设置为主机的数据源； ABCDE._hostids - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源。
问题		1	problem	如果在模板仪表板上配置该部件，则不支持此参数。 问题事件名称（不区分大小写，完整名称或其中一部分）。
严重性		0	severities.0	0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。 默认值：空（全部启用）。 注意：要配置多个值，请为每个值创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。

参数		type	name	value
问题标签				
	评估类型	0	evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必需
	运算符	0	tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必需
	标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必需
显示已抑制的问题		0	show_suppressed	0 - (默认) 禁用； 1 - 启用。
隐藏没有问题的组		0	hide_empty_groups	0 - (默认) 禁用； 1 - 启用。 如果在模板仪表板上配置该部件，则不支持此参数。
问题显示		0	ext_ack	0 - (默认) 全部； 1 - 仅未确认； 2 - 分开显示。
Reference		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。该值在该部件所属的仪表板内必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 问题主机组件更多信息，请参考创建仪表盘。

配置一个问题主机组件

配置一个问题主机组件，用于展示主机组”2” 和”4” 中的问题主机。要求问题名称中包含”CPU” 并且严重性级别在以下范围内：”Warning”、”Average”、”High”、”Disaster”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problemhosts",
                        "name": "Problem hosts",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
                                "value": 2
                            },
                            {
                                "type": 2,
                                "name": "groupids.1",
                                "value": 4
                            },
                            {
                                "type": 1,
                                "name": "problem",
                                "value": "cpu"
                            },
                            {
                                "type": 0,
                                "name": "severities.0",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "severities.1",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "severities.2",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "severities.3",
                                "value": 5
                            }
                        ]
                    }
                ]
            }
        ],
```

```
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

23 问题

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置问题 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新问题小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于问题小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
显示	0	show	1 - (默认) 最近的问题； 2 - 历史记录； 3 - 问题。

参数		类型	名称	值
主机群组		2	groupids.0	主机群组 ID。 注意：要配置多个主机群组，请为每个主机群组创建一个小组件字段对象，并在属性名称中使用递增的数字。
	主机群组（小组件）	1	groupids._reference	如果在模板仪表板上配置小组件，则不支持此参数。 替代主机群组 ID： ABCDE._hostgroupids - 将一个兼容的小组件（其引用参数设置为"ABCDE"）设置为主机群组的数据源。
排除主机群组		2	exclude_groupids.0	如果在模板仪表板上配置小组件，则不支持此参数。 主机群组 ID。 注意：要排除多个主机群组，请为每个主机群组创建一个小组件字段对象，并在属性名称中使用递增的数字。
主机		3	hostids.0	如果在模板仪表板上配置小组件，则不支持此参数。 主机 ID。 注意：要配置多个主机，请为每个主机创建一个小组件字段对象，并在属性名称中使用递增的数字。对于多个主机，主机群组参数必须完全未配置，或至少配置一个所配置主机所属的主机群组。
	主机（小组件/仪表板）	1	hostids._reference	如果在模板仪表板上配置小组件，则不支持此参数。 替代主机 ID： DASHBOARD.hostids - 将仪表板的主机选择器设置为主机的数据源； ABCDE._hostids - 将一个兼容的小组件（其引用参数设置为"ABCDE"）设置为主机的数据源。
问题严重性		1	problem	如果在模板仪表板上配置小组件，则不支持此参数。 问题事件名称（不区分大小写，可以是完整名称或其中一部分）。
		0	severities.0	0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 严重； 5 - 灾难。 默认值：空（全部启用）。 注意：要配置多个值，请为每个值创建一个小组件字段对象，并在属性名称中使用递增的数字。
问题标签				
	评估类型	0	evaltype	0 - (默认) 与/或； 2 - 或。
	标签名称	1	tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置问题标签时为必需

参数		类型	名称	值
	运算符	0	tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置问题标签时为必需
	标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置问题标签时为必需
显示标签		0	show_tags	0 - (默认) 无； 1 - 1； 2 - 2； 3 - 3。
标签名称(格式)		0	tag_name_format	0 - (默认) 完整； 1 - 缩短； 2 - 无。 参数行为： - 仅当显示标签设置为”1”、”2” 或”3” 时支持
标签显示优先级		1	tag_priority	以逗号分隔的标签列表。 参数行为： - 仅当显示标签设置为”1”、”2” 或”3” 时支持
显示运行数据		0	show_opdata	0 - (默认) 无； 1 - 单独显示； 2 - 与问题名称一起显示。
显示已抑制的问题		0	show_suppressed	0 - (默认) 禁用； 1 - 启用。
确认状态		0	acknowledgement_status	0 - (默认) 全部； 1 - 未确认； 2 - 已确认。
由我		0	acknowledged_by_me	0 - (默认) 禁用； 1 - 启用。

参数	类型	名称	值
排序条目依据	0	sort_triggers	1 - 严重性（降序）； 2 - 主机（升序）； 3 - 时间（升序）； 4 - (默认) 时间（降序）； 13 - 严重性（升序）； 14 - 主机（降序）； 15 - 问题（升序）； 16 - 问题（降序）。 除”时间（降序）”和”时间（升序）”外，对于所有值，显示时间线参数必须设置为”禁用”。 如果在模板仪表板上配置小组件，则不支持”主机（升序）”和”主机（降序）”值。
显示时间线	0	show_timeline	0 - 禁用； 1 - (默认) 启用。 参数行为： - 仅当排序条目依据设置为”时间（降序）”或”时间（升序）”时支持
突出显示整行	0	highlight_row	0 - (默认) 禁用； 1 - 启用。 高对比度主题不支持此参数。
显示行数	0	show_lines	可能值的范围为 1 到 100。 默认值：25。
引用	1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的仪表板中必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 问题组件。更多信息，请参考创建仪表盘。

配置一个问题组件

配置一个问题组件，用于展示主机组”4”上满足以下条件的问题:

- 问题存在一个名称为”scope”，值为”performance”或”availability”或”capacity”的标签。
- 问题严重为以下之一：”Warning”, ”Average”, ”High”, ”Disaster”。

此外，设置组件展示标签和操作数据。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problems",
                        "name": "Problems",
```

```
                "x": 0,
                "y": 0,
                "width": 36,
                "height": 5,
                "view_mode": 0,
                "fields": [
                    {
                        "type": 2,
                        "name": "groupids.0",
                        "value": 4
                    },
                    {
                        "type": 1,
                        "name": "tags.0.tag",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.0.operator",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.0.value",
                        "value": "performance"
                    },
                    {
                        "type": 1,
                        "name": "tags.1.tag",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.1.operator",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.1.value",
                        "value": "availability"
                    },
                    {
                        "type": 1,
                        "name": "tags.2.tag",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.2.operator",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.2.value",
                        "value": "capacity"
                    },
                    {
                        "type": 0,
                        "name": "severities.0",
                        "value": 2
                    },
                    {
```

```
                                "type": 0,
                                "name": "severities.1",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "severities.2",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "severities.3",
                                "value": 5
                            },
                            {
                                "type": 0,
                                "name": "show_tags",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "show_opdata",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```


参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

24 按严重性显示的问题

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置按严重性显示问题 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新按严重性显示问题小组件，请参阅下表中所述的参数行为。

参数

按严重性显示问题小组件支持以下参数。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机群组	2	groupids.0	主机群组 ID。 注意：要配置多个主机群组，请为每个主机群组创建一个仪表板小组件字段对象，并在属性名称中使用递增的编号。
主机群组（小组件）	1	groupids._reference	如果在模板仪表板上配置该小组件，则不支持此参数。 用于代替主机群组 ID： ABCDE._hostgroupids - 将一个兼容的小组件（其 *Reference* 参数设置为"ABCDE"）设为主机群组的数据源。
排除主机群组	2	exclude_groupids.0	如果在模板仪表板上配置该小组件，则不支持此参数。 主机群组 ID。 注意：要排除多个主机群组，请为每个主机群组创建一个仪表板小组件字段对象，并在属性名称中使用递增的编号。
主机	3	hostids.0	如果在模板仪表板上配置该小组件，则不支持此参数。 主机 ID。 注意：要配置多个主机，请为每个主机创建一个仪表板小组件字段对象，并在属性名称中使用递增的编号。对于多个主机，参数 主机群组必须要么完全不配置，要么至少配置一个包含所配置主机的主机群组。
主机（小组件/仪表板）	1	hostids._reference	如果在模板仪表板上配置该小组件，则不支持此参数。 用于代替主机 ID： DASHBOARD.hostids - 将仪表板的主机选择器设为主机的数据源； ABCDE._hostids - 将一个兼容的小组件（其 *Reference* 参数设置为"ABCDE"）设为主机的数据源。
问题	1	problem	如果在模板仪表板上配置该小组件，则不支持此参数。 问题事件名称（不区分大小写，完整名称或其一部分）。

参数		type	name	value
严重性		0	severities.0	0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。 默认值：空（全部启用）。 注意：要配置多个值，请为每个值创建一个仪表板小组件字段对象，并在属性名称中使用递增的编号。
问题标签				
	评估类型	0	evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必需
	操作符	0	tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必需
	标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必需
显示		0	show_type	0 - (默认) 主机群组； 1 - 总计。 如果在模板仪表板上配置该小组件，则不支持此参数，且默认设置为 “总计”。
布局		0	layout	0 - (默认) 水平； 1 - 垂直。 参数行为： - 当 显示设置为 “总计” 时，支持
显示运行数据		0	show_opdata	0 - (默认) 无； 1 - 分开显示； 2 - 与问题名称一起显示。

参数	type	name	value
显示已抑制的问题	0	show_suppressed	0 - (默认) 禁用； 1 - 启用。
隐藏无问题的群组	0	hide_empty_groups	0 - (默认) 禁用； 1 - 启用。 参数行为： - 当 显示设置为 “主机群组” 时，支持 如果在模板仪表板上配置该小组件，则不支持此参数。
问题显示	0	ext_ack	0 - (默认) 全部； 1 - 仅未确认； 2 - 分开显示。
显示时间线	0	show_timeline	0 - 禁用； 1 - (默认) 启用。
Reference	1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。该值在小组件所属的仪表板内必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 问题严重性组件。更多信息，请参考创建仪表盘。

配置一个问题严重性组件

配置一个问题严重性组件，用于展示所有主机组中的问题总数。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problemsbysv",
                        "name": "Problems by severity",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
```

```
                                "name": "show_type",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

25 散点图

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置散点图 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新散点图小组件，请参考下表中所述的参数行为。

参数

以下参数适用于 散点图部件。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - *(默认)* 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
引用	1	reference	由 5 个字符组成的任意字符串值（例如：ABCDE 或 JBPNL）。该值在部件所属的仪表板内必须唯一。 参数行为： - 必填

数据集

支持使用以下参数配置数据集。

Note:
属性名称中的第一个数字（例如 ds.0.hosts.0、ds.0.items.0）表示特定的数据集，而第二个数字（如果存在）表示已配置的主机或监控项。

参数	type	名称	值
数据集类型 数据集：监控项列表	0	ds.0.dataset_type	0 - 监控项列表； 1 - *(默认)* 监控项模式。
颜色	1	ds.0.color.0	十六进制颜色代码（例如 FF0000）。 参数行为： - 如果数据集类型设置为“监控项列表”，则为必需
X 轴	4	ds.0.x_axis_itemids.0	监控项 ID。 在模板仪表板上配置该部件时，只应设置模板上已配置的监控项。 注意：要配置多个监控项，请为每个监控项创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 参数行为： - 如果数据集类型设置为“监控项列表”且未设置 *X* 轴（部件），则为必需
X 轴（部件）	1	ds.0.x_axis_itemids.0.reference	用于替代监控项 ID： ABCDE._itemid - 将兼容部件（其 *Reference* 参数设置为“ABCDE”）设置为监控项的数据源。 注意：要配置多个部件，请为每个部件创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 参数行为： - 如果数据集类型设置为“监控项列表”且未设置 *X* 轴，则为必需

参数	type	名称	值
Y 轴	4	ds.0.y_axis_itemids.0	监控项 ID。 在模板仪表板上配置该部件时，只应设置模板上已配置的监控项。 注意：要配置多个监控项，请为每个监控项创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 参数行为： - 如果数据集类型设置为 “监控项列表” 且未设置 *Y* 轴（部件），则为必需
Y 轴（部件）	1	ds.0.y_axis_itemids.0.reference	用于替代监控项 ID： `ABCDE._itemid` - 将兼容部件（其 *Reference* 参数设置为 “ABCDE”）设置为监控项的数据源。 注意：要配置多个部件，请为每个部件创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 参数行为： - 如果数据集类型设置为 “监控项列表” 且未设置 *Y* 轴，则为必需
覆盖主机	1	ds.0.override_hostid.reference	`ABCDE._hostid` - 将兼容部件（其 “Reference” 参数设置为 “ABCDE”）设置为主机的数据源； `DASHBOARD._hostid` - 将仪表板主机选择器设置为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。
数据集：监控项模式			
颜色	1	ds.0.color	十六进制颜色代码（例如 `FF0000`）。 参数行为： - 如果数据集类型设置为 “监控项模式” 且未设置调色板，则支持
调色板	0	ds.0.color_palette	调色板索引。 可能的值范围：0-11。 默认值：0。 参数行为： - 如果数据集类型设置为 “监控项模式” 且未设置颜色，则支持
主机模式	1	ds.0.hosts.0	主机名称或模式（例如，“Zabbix*”）。 参数行为： - 如果数据集类型设置为 “监控项模式”，则为必需 如果在模板仪表板上配置该部件，则不支持此参数。
X 轴监控项模式	1	ds.0.x_axis_items.0	监控项名称或模式（例如，“*: Number of processed *values per second”）。 在模板仪表板上配置该部件时，只应设置模板上已配置监控项的模式。 参数行为： - 如果数据集类型设置为 “监控项模式”，则为必需

参数		type	名称	值
Y 轴监控项模式		1	ds.0.y_axis_items.0	监控项名称或模式（例如，“*: Number of processed *values per second”）。 在模板仪表板上配置该部件时，只应设置模板上已配置监控项的模式。 参数行为： - 如果数据集类型设置为 “监控项模式”，则为必需
主机群组		2	groupids.0	主机群组 ID。 注意：要配置多个主机群组，请为每个主机群组创建一个仪表板部件字段对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该部件，则不支持此参数。
主机群组(部件)		1	groupids._reference	用于替代主机群组 ID： `ABCDE._hostgroupids` - 将兼容部件（其 *Reference* 参数设置为 “ABCDE”）设置为主机群组的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。
覆盖主机		1	ds.0.override_hostid._reference	`ABCDE._hostid` - 将兼容部件（其 “Reference” 参数设置为 “ABCDE”）设置为主机的数据源； `DASHBOARD._hostid` - 将仪表板主机选择器设置为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。
主机标签				
	评估类型	0	evaltype_host	0 - *(默认)* And/Or； 2 - Or。 如果在模板仪表板上配置该部件，则不支持此参数。
	标签名称	1	host_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了主机标签，则为必需 如果在模板仪表板上配置该部件，则不支持此参数。
	运算符	0	host_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了主机标签，则为必需 如果在模板仪表板上配置该部件，则不支持此参数。

参数	type	名称	值
标签值	1	host_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了主机标签，则为必需 如果在模板仪表板上配置该部件，则不支持此参数。
标记	0	ds.0.marker	0 - (默认) 椭圆； 1 - 方形； 2 - 三角形； 3 - 菱形； 4 - 星形； 5 - 十字形。
标记大小	0	ds.0.marker_size	0 - (默认) 小（6px）； 1 - 中（9px）； 2 - 大（15px）。
时间偏移	1	ds.0.timeshift	有效的时间字符串（例如 3600、1h 等）。 你可以使用时间后缀。也允许使用负值。 默认值："""（空）。
聚合间隔	1	ds.0.aggregate_interval	有效的时间字符串（例如 3600、1h 等）。 你可以使用时间后缀。 默认值：15m。
聚合函数	0	ds.0.aggregate_function	0 - 不使用； 1 - min； 2 - max； 3 - (默认) avg； 4 - count； 5 - sum； 6 - first； 7 - last。

显示选项

以下参数支持用于配置显示选项。

参数	type	name	value
历史数据选择	0	source	0 - (默认) 自动； 1 - 历史数据； 2 - 趋势。
标签中的主机名称	0	show_hostnames	0 - (默认) 自动； 1 - 显示； 2 - 隐藏。 如果在模板仪表板上配置该小组件，则不支持此参数。

时间段

以下参数支持用于配置 时间段。

参数		type	name	value
时间段		1	time_period._referenceDASHBOARD._timeperiod	DASHBOARD._timeperiod - 将仪表板时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的部件（其 *Reference* 参数设置为 "ABCDE"）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，您也可以仅在 *From* 和 *To* 参数中设置时间段。
	From	1	time_period.from	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则支持 - 如果设置了 time_period.to，则必需
	To	1	time_period.to	绝对（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 如果未设置 时间段，则支持 - 如果设置了 time_period.from，则必需

坐标轴

以下参数支持用于配置 坐标轴。

参数	type	name	value
X 轴	0	x_axis	0 - 禁用； 1 - *(*默认*)* 启用。
Y 轴	0	y_axis	0 - *(*默认*)* 禁用； 1 - 启用。
最小值	1	x_axis_min y_axis_min	任意数值。 默认值："（空）。
最大值	1	x_axis_max y_axis_max	任意数值。 默认值："（空）。
单位（类型）	0	x_axis_units y_axis_units	0 - *(*默认*)* 自动； 1 - 静态。
单位（值）	1	x_axis_static_units y_axis_static_units	任意字符串值。 默认值："（空）。

图例

以下参数可用于配置 图例。

参数	type	name	value
显示图例	0	legend	0 - 禁用； 1 - *(*默认*)* 启用。
显示聚合函数	0	legend_aggregation	0 - *(*默认*)* 禁用； 1 - 启用。
行数	0	legend_lines_mode	参数行为： - 如果 显示图例设置为 "启用"，则 *supported* 0 - *(*默认*)* 固定； 1 - 可变。 参数行为： - 如果 显示图例设置为 "启用"，则 *supported*

参数	type	name	value
行数/ 最大行数	0	legend_lines	可能的取值范围：1-10。 默认值：1。 参数行为： - 如果 显示图例设置为 “启用”，则 *supported*
列数	0	legend_columns	可能的取值范围：1-4。 默认值：4。 参数行为： - 如果 显示图例设置为 “启用”，则 *supported*

阈值

支持以下参数来配置阈值。

Note:
Thresholds 属性名称中的数字（例如 thresholds.0.color）表示按升序排序的列表中的阈值位置。但是，如果阈值以不同顺序配置，那么在 Zabbix 前端中更新部件配置后，这些值将按升序排序（例如，`"thresholds.0.threshold":"5"` → `"thresholds.0.threshold":"1"`；`"thresholds.1.threshold":"1"` → `"thresholds.1.threshold": "5"`）。

参数	type	name	value
颜色插值	0	interpolation	0 - （默认）禁用； 1 - 启用。
颜色	1	thresholds.0.color	十六进制颜色代码（例如 FF0000）。
X 轴	1	thresholds.0.x_axis_threshold	任意数值。支持后缀（例如”1d”、”2w”、”4K”、”8G”）。
Y 轴	1	thresholds.0.y_axis_threshold	任意数值。支持后缀（例如”1d”、”2w”、”4K”、”8G”）。

示例

以下示例仅用于说明 散点图部件的仪表板部件字段对象的配置。有关配置仪表板的更多信息，请参见`dashboard.create`。

配置 散点图小组件

按以下方式配置 散点图小组件：

- 5 个类型为 “监控项模式” 的数据集，每个数据集均使用默认聚合（avg，15m）和不同的标记。
- 图例以单行两列显示已配置的监控项。
- 配置阈值后，当标记在任一坐标轴上的值达到 80 时，将其着色为红色。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "scatterplot",
                        "name": "Scatter plot",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
```

```
                {
                    "type": 1,
                    "name": "ds.0.color",
                    "value": "0C5A87"
                },
                {
                    "type": 1,
                    "name": "ds.0.hosts.0",
                    "value": "de-frankfurt*"
                },
                {
                    "type": 1,
                    "name": "ds.0.x_axis_items.0",
                    "value": "Memory utilization"
                },
                {
                    "type": 1,
                    "name": "ds.0.y_axis_items.0",
                    "value": "CPU utilization"
                },
                {
                    "type": 0,
                    "name": "ds.0.marker",
                    "value": 0
                },
                {
                    "type": 0,
                    "name": "ds.0.marker_size",
                    "value": 1
                },
                {
                    "type": 1,
                    "name": "ds.1.color",
                    "value": "66B0D9"
                },
                {
                    "type": 1,
                    "name": "ds.1.hosts.0",
                    "value": "fr-paris*"
                },
                {
                    "type": 1,
                    "name": "ds.1.x_axis_items.0",
                    "value": "Memory utilization"
                },
                {
                    "type": 1,
                    "name": "ds.1.y_axis_items.0",
                    "value": "CPU utilization"
                },
                {
                    "type": 0,
                    "name": "ds.1.marker",
                    "value": 1
                },
                {
                    "type": 0,
                    "name": "ds.1.marker_size",
                    "value": 1
                },
                {
                    "type": 1,
```

```
            "name": "ds.2.color",
            "value": "0A466A"
        },
        {
            "type": 1,
            "name": "ds.2.hosts.0",
            "value": "lv-riga*"
        },
        {
            "type": 1,
            "name": "ds.2.x_axis_items.0",
            "value": "Memory utilization"
        },
        {
            "type": 1,
            "name": "ds.2.y_axis_items.0",
            "value": "CPU utilization"
        },
        {
            "type": 0,
            "name": "ds.2.marker",
            "value": 2
        },
        {
            "type": 0,
            "name": "ds.2.marker_size",
            "value": 1
        },
        {
            "type": 1,
            "name": "ds.3.color",
            "value": "3394C3"
        },
        {
            "type": 1,
            "name": "ds.3.hosts.0",
            "value": "pl-warsaw*"
        },
        {
            "type": 1,
            "name": "ds.3.x_axis_items.0",
            "value": "Memory utilization"
        },
        {
            "type": 1,
            "name": "ds.3.y_axis_items.0",
            "value": "CPU utilization"
        },
        {
            "type": 0,
            "name": "ds.3.marker",
            "value": 3
        },
        {
            "type": 0,
            "name": "ds.3.marker_size",
            "value": 1
        },
        {
            "type": 1,
            "name": "ds.4.color",
            "value": "1492C8"
```

```
            },
            {
                "type": 1,
                "name": "ds.4.hosts.0",
                "value": "se-stockholm*"
            },
            {
                "type": 1,
                "name": "ds.4.x_axis_items.0",
                "value": "Memory utilization"
            },
            {
                "type": 1,
                "name": "ds.4.y_axis_items.0",
                "value": "CPU utilization"
            },
            {
                "type": 0,
                "name": "ds.4.marker",
                "value": 4
            },
            {
                "type": 0,
                "name": "ds.4.marker_size",
                "value": 1
            },
            {
                "type": 0,
                "name": "legend_columns",
                "value": 2
            },
            {
                "type": 0,
                "name": "interpolation",
                "value": 1
            },
            {
                "type": 1,
                "name": "thresholds.0.color",
                "value": "D40000"
            },
            {
                "type": 1,
                "name": "thresholds.0.x_axis_threshold",
                "value": "80"
            },
            {
                "type": 1,
                "name": "thresholds.0.y_axis_threshold",
                "value": ""
            },
            {
                "type": 1,
                "name": "thresholds.1.color",
                "value": "D40000"
            },
            {
                "type": 1,
                "name": "thresholds.1.x_axis_threshold",
                "value": ""
            },
            {
```

```
                                "type": 1,
                                "name": "thresholds.1.y_axis_threshold",
                                "value": "80"
                            },
                            {
                                "type": 1,
                                "name": "reference",
                                "value": "JQISY"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

另请参阅

- 仪表板小部件字段
- dashboard.create
- dashboard.update

26 SLA 报告

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置*SLA* 报告小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *SLA* 报告小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *SLA* 报告小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - （默认）不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
SLA	10	slaid.0	SLA ID。 参数行为： - 必需
服务	9	serviceid.0	服务 ID。
显示周期	0	show_periods	可能值的范围为 1 到 100。 默认值：20。
起始时间	1	date_from	格式为 YYYY-MM-DD 的有效日期字符串。 支持带有 d、w、M、y 修饰符的相对日期（例如 now、now/d、now/w-1w 等）。
结束时间	1	date_to	格式为 YYYY-MM-DD 的有效日期字符串。 支持带有 d、w、M、y 修饰符的相对日期（例如 now、now/d、now/w-1w 等）。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 *SLA* 报表组件。更多信息，请参考创建仪表盘。

配置一个 *SLA* 报表组件

配置一个 *SLA* 报表组件，用于展示最近 30 天内 SLA "4" 和服务"2" 的 SLA 报表。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "slareport",
                        "name": "SLA report",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 10,
                                "name": "slaid.0",
                                "value": 4
                            },
                            {
                                "type": 9,
                                "name": "serviceid.0",
                                "value": 2
                            },
                            {
```

```
                                "type": 1,
                                "name": "date_from",
                                "value": "now-30d"
                            },
                            {
                                "type": 1,
                                "name": "date_to",
                                "value": "now"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

27 系统信息

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置系统信息小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新系统信息小组件，请参阅下表中所述的参数行为。

参数

System Information 小部件支持以下参数。

Parameter	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - *(默认)* 15 分钟。
显示	0	info_type	0 - *(默认)* 系统统计信息； 1 - 高可用节点。
显示软件更新检查详情	0	show_software_update_check_details	0 - *(默认)* 已禁用； 1 - 已启用。 参数行为： - 当 Zabbix 服务器配置中的AllowSoftwareUpdateCheck 已启用且 显示设置为"System stats" 时，*supported*。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置系统信息组件。更多信息，参考创建仪表盘。

配置一个系统信息组件

配置一个刷新频率为 10 分钟且开启了软件更新检查的系统信息组件。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "systeminfo",
                        "name": "System information",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 600
                            },
                            {
                                "type": 0,
                                "name": "show_software_update_check_details",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
```

```
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

28 热门主机

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置*Top hosts*小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这允许用户修改内置小组件并创建自定义小组件，但也会带来错误创建或更新小组件的风险。为确保成功创建或更新 *Top hosts* 小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于主机排行小组件。

参数	类型	名称	值
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机群组	2	groupids.0	主机群组 ID。 注意：要配置多个主机群组，请为每个主机群组创建一个小组件字段对象，并在属性名称中使用递增的数字。 如果在模板仪表板上配置小组件，则不支持此参数。

参数		类型	名称	值
	主机群组（小组件）	1	groupids._reference	替代主机群组 ID： ABCDE._hostgroupids - 将一个兼容的小组件（其引用参数设置为“ABCDE”）设置为主机群组的数据源。 如果在模板仪表板上配置小组件，则不支持此参数。
主机		3	hostids.0	主机 ID。 注意：要配置多个主机，请为每个主机创建一个小组件字段对象，并在属性名称中使用递增的数字。对于多个主机，参数主机群组必须完全不配置，或至少配置一个所配置主机所属的主机群组。
	主机（小组件/仪表板）	1	hostids._reference	如果在模板仪表板上配置小组件，则不支持此参数。 替代主机 ID： DASHBOARD.hostids - 将仪表板的主机选择器设置为主机的数据源； ABCDE._hostids - 将一个兼容的小组件（其引用参数设置为“ABCDE”）设置为主机的数据源。 如果在模板仪表板上配置小组件，则不支持此参数。
主机标签				
	评估类型	0	evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	tags.0.tag	如果在模板仪表板上配置小组件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置主机标签，则为必需
	运算符	0	tags.0.operator	如果在模板仪表板上配置小组件，则不支持此参数。 0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置主机标签，则为必需
	标签值	1	tags.0.value	如果在模板仪表板上配置小组件，则不支持此参数。 任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置主机标签，则为必需 如果在模板仪表板上配置小组件，则不支持此参数。

参数	类型	名称	值
显示处于维护状态的主机列(见下文)	0	maintenance	0 - (默认) 禁用； 1 - 启用。
排序依据	0	column	已配置列中的列数值。
排序顺序	0	order	2 - (默认) 前 N 个； 3 - 后 N 个。
主机限制	0	show_lines	可选值范围为 1 到 1000。 默认值：10。 如果在模板仪表板上配置小组件，则不支持此参数。

列

列有公共参数，也有由 数据参数配置确定的额外参数。

Note:
所有列相关的参数中的数字 (例如：`columns.0.name`) 依赖于配置在哪一列。示例中代表配置第一列。

所有列均支持以下参数：

参数	类型	参数名称	参数值或参数说明
名称	1	columns.0.name	任意字符串。 参数行为： - 必需的。
数据	0	columns.0.data	1 - 监控项值； 2 - 主机名称； 3 - 文本。 参数行为： - 必需的。
颜色	1	columns.0.base_color	十六进制颜色代码 (例如：FF0000)。 参数行为： - 必需的。

监控项值

如果将 *Data* 设置为 “监控项值”，则支持以下参数。

Note:
Thresholds 属性名称中的第一个数字（例如，columnsthresholds.0.color.0）引用配置阈值的列，第二个数字引用按升序排列的列表中的阈值位置。但是，如果阈值按其他顺序配置，则在 Zabbix 前端更新小组件配置后，这些值将按升序排列（例如，`"threshold.0.threshold":"5"` → `"threshold.0.threshold":"1"`；`"threshold.1.threshold":"1"` → `"threshold.1.threshold": "5"`）。

参数	类型	名称	值
监控项	1	columns.0.item	有效的监控项名称。 在模板仪表板上配置小组件时，只应设置模板上配置的监控项。
将监控项值显示为	0	columns.0.display_value	0 - （默认）数值； 1 - 文本； 2 - 二进制。
显示	0	columns.0.display	1 - （默认）原样； 2 - 条形图； 3 - 指示器； 6 - 迷你图。 参数行为： - 当将监控项值显示为设置为 “数值” 时支持
最小值	1	columns.0.min	任意数值。 参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “条形图” 或 “指示器” 时支持
最大值	1	columns.0.max	任意数值。 参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “条形图” 或 “指示器” 时支持
小数位数	0	columns.0.decimal_places	可能值范围为 0 到 10。 默认值：2。 参数行为： - 当将监控项值显示为设置为 “数值” 时支持
迷你图			
宽度	0	columns.0.sparkline.width	可能值范围为 0 到 10。 默认值：1。 参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持
填充	0	columns.0.sparkline.fill	可能值范围为 0 到 10。 默认值：3。 参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持

参数		类型	名称	值
	颜色	1	columns.0.sparkline.color	十六进制颜色代码（例如 FF0000）。 默认值：42A5F5。 参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持
	时间段	1	columns.0.sparkline.time_period.reference	DASHBOARD._timeperiod - 将仪表板时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其 reference 参数等于 ABCDE）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，也可以仅在起始时间和结束时间参数中设置时间段。
	起始时间	1	columns.0.sparkline.time_period.from	参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持 采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。
	结束时间	1	columns.0.sparkline.time_period.to	参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持 - 如果设置了 columns.0.sparkline.time_period.to，则必需 采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。
	历史数据	0	columns.0.sparkline.history	参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持 - 如果设置了 columns.0.sparkline.time_period.from，则必需 0 - （默认）自动； 1 - 历史数据； 2 - 趋势。
阈值				参数行为： - 当将监控项值显示为设置为 “数值”，且显示设置为 “迷你图” 时支持
	颜色	1	columnsthresholds.0.color	十六进制颜色代码（例如 FF0000）。 参数行为： - 当将监控项值显示为设置为 “数值” 时支持
	阈值	1	columnsthresholds.0.threshold	任意字符串值。 参数行为： - 当将监控项值显示为设置为 “数值” 时支持
高亮				
	颜色	1	columns.0.highlights.0.color	十六进制颜色代码（例如 FF0000）。 参数行为： - 当将监控项值显示为设置为 “文本” 时支持
	模式	1	columns.0.highlights.0.pattern	任意字符串值。 参数行为： - 当将监控项值显示为设置为 “文本” 时支持
显示缩略图		0	columns.0.show_thumbnail	0 - （默认）禁用； 1 - 启用。 参数行为： - 当将监控项值显示为设置为 “二进制” 时支持

参数		类型	名称	值
聚合函数		0	columns.0.aggregate_function	0 - （默认）不使用； 1 - 最小值； 2 - 最大值； 3 - 平均值； 4 - 计数； 5 - 总和； 6 - 第一个； 7 - 最后一个。
时间段		1	columns.0.time_period.reference	DASHBOARD._timeperiod - 将仪表板时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其 reference 参数等于 ABCDE）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，也可以仅在起始时间和结束时间参数中设置时间段。 参数行为： - 当聚合函数设置为 “最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个” 或 “最后一个” 时支持
	起始时间	1	columns.0.time_period.from	采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 当未设置时间段且聚合函数设置为 “最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个” 或 “最后一个” 时支持 - 如果设置了 columns.0.time_period.to，则必需
	结束时间	1	columns.0.time_period.to	采用绝对时间语法（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 当未设置时间段且聚合函数设置为 “最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个” 或 “最后一个” 时支持 - 如果设置了 columns.0.time_period.from，则必需
历史数据		0	columns.0.history	0 - （默认）自动； 1 - 历史数据； 2 - 趋势。 参数行为： - 当将监控项值显示为设置为 “数值” 时支持
引用		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。此值在小组件所属的仪表板中必须唯一。 参数行为： - 必需

文本

如果 *Data* 设置为”Text”，则支持以下参数。

Parameter	type	name	value
Text	1	columns.0.text	任意字符串值，包括宏。 支持的宏：{HOST.*}、{INVENTORY.*}。 参数行为： - 如果 *Data* 设置为”Text”，则为 *required*

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 *Top* 主机组件。更多信息，请参考创建仪表盘。

配置一个 *Top* 主机组件

配置一个 *Top* 主机组件，用于展示主机组”4” 中 CPU 利用率的 Top 主机。同时配置如下列：”Host name”, ”CPU utilization in %”, ”1m avg”, ”5m avg”, ”15m avg”, ”Processes”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "tophosts",
                        "name": "Top hosts",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
                                "value": 4
                            },
                            {
                                "type": 1,
                                "name": "columns.0.name",
                                "value": "Host"
                            },
                            {
                                "type": 0,
                                "name": "columns.0.data",
                                "value": 2
                            },
                            {
                                "type": 1,
                                "name": "columns.0.base_color",
                                "value": "FFFFFF"
                            },
                            {
                                "type": 1,
                                "name": "columns.1.name",
                                "value": "CPU utilization in %"
                            },
                            {
                                "type": 0,
                                "name": "columns.1.data",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "columns.1.base_color",
                                "value": "4CAF50"
                            },
                            {
                                "type": 1,
                                "name": "columns.1.item",
```

```
            "value": "CPU utilization"
        },
        {
            "type": 0,
            "name": "columns.1.display",
            "value": 3
        },
        {
            "type": 1,
            "name": "columns.1.min",
            "value": "0"
        },
        {
            "type": 1,
            "name": "columns.1.max",
            "value": "100"
        },
        {
            "type": 1,
            "name": "columnsthresholds.1.color.0",
            "value": "FFFF00"
        },
        {
            "type": 1,
            "name": "columnsthresholds.1.threshold.0",
            "value": "50"
        },
        {
            "type": 1,
            "name": "columnsthresholds.1.color.1",
            "value": "FF8000"
        },
        {
            "type": 1,
            "name": "columnsthresholds.1.threshold.1",
            "value": "80"
        },
        {
            "type": 1,
            "name": "columnsthresholds.1.color.2",
            "value": "FF4000"
        },
        {
            "type": 1,
            "name": "columnsthresholds.1.threshold.2",
            "value": "90"
        },
        {
            "type": 1,
            "name": "columns.2.name",
            "value": "1m avg"
        },
        {
            "type": 0,
            "name": "columns.2.data",
            "value": 1
        },
        {
            "type": 1,
            "name": "columns.2.base_color",
            "value": "FFFFFF"
        },
```

```
                {
                    "type": 1,
                    "name": "columns.2.item",
                    "value": "Load average (1m avg)"
                },
                {
                    "type": 1,
                    "name": "columns.3.name",
                    "value": "5m avg"
                },
                {
                    "type": 0,
                    "name": "columns.3.data",
                    "value": 1
                },
                {
                    "type": 1,
                    "name": "columns.3.base_color",
                    "value": "FFFFFF"
                },
                {
                    "type": 1,
                    "name": "columns.3.item",
                    "value": "Load average (5m avg)"
                },
                {
                    "type": 1,
                    "name": "columns.4.name",
                    "value": "15m avg"
                },
                {
                    "type": 0,
                    "name": "columns.4.data",
                    "value": 1
                },
                {
                    "type": 1,
                    "name": "columns.4.base_color",
                    "value": "FFFFFF"
                },
                {
                    "type": 1,
                    "name": "columns.4.item",
                    "value": "Load average (15m avg)"
                },
                {
                    "type": 1,
                    "name": "columns.5.name",
                    "value": "Processes"
                },
                {
                    "type": 0,
                    "name": "columns.5.data",
                    "value": 1
                },
                {
                    "type": 1,
                    "name": "columns.5.base_color",
                    "value": "FFFFFF"
                },
                {
                    "type": 1,
```

```
                                "name": "columns.5.item",
                                "value": "Number of processes"
                            },
                            {
                                "type": 0,
                                "name": "columns.5.decimal_places",
                                "value": 0
                            },
                            {
                                "type": 0,
                                "name": "column",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

29 最常用监控项

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置*Top items*小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *Top items* 小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *Top items* 小组件。

参数		type	name	value
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个 dashboard 小组件字段对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该小组件，则不支持此参数。
	主机组（小组件）	1	groupids._reference	用于代替主机组 ID： `ABCDE._hostgroupids` - 将兼容的小组件（其 *Reference* 参数设置为"ABCDE"）设置为主机组的数据源。 如果在模板仪表板上配置该小组件，则不支持此参数。
主机		3	hostids.0	主机 ID。 注意：要配置多个主机，请为每个主机创建一个 dashboard 小组件字段对象，并在属性名称中使用递增的编号。对于多个主机，参数 主机组必须要么完全不配置，要么至少配置一个包含所配置主机的主机组。 如果在模板仪表板上配置该小组件，则不支持此参数。
	主机（小组件/仪表板）	1	hostids._reference	用于代替主机 ID： `DASHBOARD.hostids` - 将仪表板的主机选择器设置为主机的数据源； `ABCDE._hostids` - 将兼容的小组件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源。 如果在模板仪表板上配置该小组件，则不支持此参数。
主机标签				
	评估类型	0	host_tags_evaltype	0 - (默认) And/Or； 2 - Or。 如果在模板仪表板上配置该小组件，则不支持此参数。
	标签名称	1	host_tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 主机标签，则为 必填 如果在模板仪表板上配置该小组件，则不支持此参数。

参数		type	name	value
	运算符	0	host_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 主机标签，则为 必填 如果在模板仪表板上配置该小组件，则不支持此参数。
	标签值	1	host_tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 主机标签，则为 必填 如果在模板仪表板上配置该小组件，则不支持此参数。
布局		0	layout	0 - (默认) 水平； 1 - 垂直。
显示故障监控项(见下文)		0	show_problems	0 - 全部； 1 - (默认) 未抑制； 2 - 无。

高级配置

以下高级配置参数适用于监控项小组件。

参数		类型	名称	值
主机排序排序依据		0	host_ordering_order_by	0 - （默认）主机名称； 3 - 监控项值。
	监控项模式	1	host_ordering_item.0	监控项名称或模式（例如：”*: 每秒处理的 * 值数量”）。 注意：要配置多个监控项模式，请为每个监控项模式创建一个小组件字段对象，并在属性名称中使用递增的编号。 在模板仪表板上配置小组件时，只应设置模板上已配置监控项的模式。 参数行为： - 当排序依据设置为 “监控项值” 时为必需
顺序		0	host_ordering_order	2 - （默认）前 N 项； 3 - 后 N 项。

参数		类型	名称	值
限制		0	host_ordering_limit	可能值范围为 1 至 100。 默认值：10。
监控项排序				
排序依据		0	item_ordering_order_by	1 - 主机； 2 - 监控项名称； 3 - （默认）监控项值。
	主机模式	1	item_ordering_host.0	主机名称或模式。 注意：要配置多个主机模式，请为每个主机模式创建一个小组件字段对象，并在属性名称中使用递增的编号。 参数行为： - 当排序依据设置为 “主机” 时为必需
顺序		0	item_ordering_order	2 - （默认）前 N 项； 3 - 后 N 项。
限制		0	item_ordering_limit	可能值范围为 1 至 100。 默认值：10。

列

Note:
属性名称中的第一个数字（例如 columns.0.items.0、columns.0.item_tags_evaltype）表示特定列；如果存在第二个数字，则表示已配置的实体（例如监控项模式、标签）。

参数		类型	名称	值
监控项模式		1	columns.0.items.0	监控项名称或模式（例如，”*: 每秒处理的 * 值数量”）。 注意：要配置多个监控项模式，请为每个监控项模式创建一个仪表板小组件字段对象，并在属性名称中递增第二个数字。 在模板仪表板上配置小组件时，只应设置模板上配置的监控项对应的模式。 参数行为： - 必需
监控项标签				
	评估类型	0	columns.0.item_tags_evaltype	0 - （默认） And/Or； 2 - Or。
	标签名称	1	columns.0.item_tags.0.tag	任意字符串值。 注意：属性名称中的第二个数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需

参数		类型	名称	值
	运算符	0	columns.0.item_tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的第二个数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需
	标签值	1	columns.0.item_tags.0.tag	任意字符串值。 注意：属性名称中的第二个数字表示标签评估列表中的标签顺序。 参数行为： - 配置监控项标签时为必需
基本颜色		1	columns.0.base_color	十六进制颜色代码（例如，FF0000）。
显示值为		0	columns.0.display_value_as	1 - （默认）数值； 2 - 文本。
显示		0	columns.0.display	1 - （默认）原样； 2 - 条形图； 3 - 指示器； 6 - 迷你图。
最小值		1	columns.0.min	任意数值。后缀（例如 “1d”、“2w”、“4K”、“8G”）受支持。 参数行为： - 当显示值为设置为 “数值”，且显示设置为 “条形图” 或 “指示器” 时支持
最大值		1	columns.0.max	任意数值。后缀（例如 “1d”、“2w”、“4K”、“8G”）受支持。 参数行为： - 当显示值为设置为 “数值”，且显示设置为 “条形图” 或 “指示器” 时支持
迷你图				
	宽度	0	columns.0.sparkline.width	可能值范围为 0 到 10。 默认值：1。 参数行为： - 当显示设置为 “迷你图” 时支持
	填充	0	columns.0.sparkline.fill	可能值范围为 0 到 10。 默认值：3。 参数行为： - 当显示设置为 “迷你图” 时支持
	颜色	1	columns.0.sparkline.color	十六进制颜色代码（例如，FF0000）。 默认值：42A5F5。 参数行为： - 当显示设置为 “迷你图” 时支持

参数		类型	名称	值
	时间段	1	columns.0.sparkline.time_period	DASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其 reference 参数等于 ABCDE）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，也可以仅在起始时间和结束时间参数中设置时间段。 参数行为： - 当显示设置为 “迷你图” 时支持
	起始时间	1	columns.0.sparkline.time_period.from	绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 当显示设置为 “迷你图” 时支持 - 设置了 columns.0.sparkline.time_period.to 时为必需
	结束时间	1	columns.0.sparkline.time_period.to	绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 当显示设置为 “迷你图” 时支持 - 设置了 columns.0.sparkline.time_period.from 时为必需
	历史数据	0	columns.0.sparkline.history	0 - （默认）自动； 1 - 历史数据； 2 - 趋势数据。 参数行为： - 当显示设置为 “迷你图” 时支持
阈值				
	颜色	1	columns.0.thresholds.color	十六进制颜色代码（例如，FF0000）。 参数行为： - 当显示值为设置为 “数值” 时支持
	阈值	1	columns.0.thresholds.threshold	任意数值。后缀（例如 “1d”、“2w”、“4K”、“8G”）受支持。 参数行为： - 当显示值为设置为 “数值” 时支持
高亮				
	高亮	1	columns.0.highlights.color	十六进制颜色代码（例如，FF0000）。 参数行为： - 当显示值为设置为 “文本” 时支持
	阈值	1	columns.0.highlights.pattern	任意正则表达式。 参数行为： - 当显示值为设置为 “文本” 时支持
小数位数高级配置		0	columns.0.decimal_places	可能值范围为 0 到 10。 默认值：2。

参数		类型	名称	值
聚合函数		0	columns.0.aggregate_function	如果显示值为设置为“数值”： 0 - （默认）不使用； 1 - 最小值； 2 - 最大值； 3 - 平均值； 4 - 计数； 5 - 总和； 6 - 第一个； 7 - 最后一个。 如果显示值为设置为“文本”： 0 - （默认）不使用； 4 - 计数； 6 - 第一个； 7 - 最后一个。
时间段		1	columns.0.time_period.reference	DASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其 reference 参数等于 ABCDE）设置为数据源。 默认值：DASHBOARD._timeperiod 或者，也可以仅在起始时间和结束时间参数中设置时间段。 参数行为： - 当聚合函数设置为“最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个”或“最后一个”时支持
	起始时间	1	columns.0.time_period.from	绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 未设置时间段且聚合函数设置为“最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个”或“最后一个”时支持 - 设置了 columns.0.time_period.to 时为必需
	结束时间	1	columns.0.time_period.to	绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、now/d、now/w-1w 等）中的有效时间字符串。 参数行为： - 未设置时间段且聚合函数设置为“最小值”、“最大值”、“平均值”、“计数”、“总和”、“第一个”或“最后一个”时支持 - 设置了 columns.0.time_period.from 时为必需
历史数据		0	columns.0.history	0 - （默认）自动； 1 - 历史数据； 2 - 趋势数据。 参数行为： - 当显示值为设置为“数值”时支持
聚合		0	columns.0.aggregate_column_group	0 - （默认）每个监控项 1 - 合并。 参数行为： - 当显示值为设置为“数值”，且显示设置为“原样”、“条形图”或“指示器”时支持。
合并聚合函数		0	columns.0.column_aggregate_function	1 - 最小值； 2 - 最大值； 3 - 平均值； 4 - 计数； 5 - 总和（默认）。 参数行为： - 配置聚合时为必需。

参数	类型	名称	值
合并列名称	1	columns.0.combined	任意字符串值。 参数行为： - 配置聚合时为必需。

示例

以下示例仅用于说明 *Top items* 小部件的仪表板小部件字段对象配置。有关如何配置仪表板的更多信息，请参见dashboard.create。

配置 *Top items* 小部件

配置一个 *Top items* 小部件，使其显示主机 “10084” 的数据，并且仅显示标签名称为 “component” 且其值包含 “cpu” 的监控项。此外，以主机位于顶部的方式显示数据，并使用彩色仪表条来表示单元格。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "topitems",
                        "name": "Top items",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 3,
                                "name": "hostids.0",
                                "value": 10084
                            },
                            {
                                "type": 1,
                                "name": "columns.0.items.0",
                                "value": "*"
                            },
                            {
                                "type": 1,
                                "name": "columns.0.item_tags.0.tag",
                                "value": "component"
                            },
                            {
                                "type": 0,
                                "name": "columns.0.item_tags.0.operator",
                                "value": 0
                            },
                            {
                                "type": 1,
                                "name": "columns.0.item_tags.0.value",
                                "value": "cpu"
                            },
                            {
```

```
                                "type": 0,
                                "name": "columns.0.display",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "layout",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

另请参见

- 仪表板部件字段
- dashboard.create
- dashboard.update

30 个最常用触发器

描述

这些参数以及相应仪表板小组件字段对象的可能属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置*Top triggers* 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *Top triggers* 小组件，请参考下表中所述的参数行为。

参数

以下参数适用于顶级触发器小组件。

参数		类型	名称	值
刷新间隔		0	rf_rate	0 - （默认）不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个小组件字段对象，并在属性名称中使用递增的数字。 如果在模板仪表板上配置小组件，则不支持此参数。
主机		3	hostids.0	主机 ID。 注意：要配置多个主机，请为每个主机创建一个小组件字段对象，并在属性名称中使用递增的数字。对于多个主机，参数主机组必须完全不配置，或者至少配置一个包含所配置主机的主机组。 如果在模板仪表板上配置小组件，则不支持此参数。
问题		1	problem	问题事件名称（不区分大小写，可以是完整名称或名称的一部分）。
严重性		0	severities.0	触发器严重性。 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 高； 5 - 灾难。 默认值：空（全部启用）。 注意：要配置多个值，请为每个值创建一个小组件字段对象，并在属性名称中使用递增的数字。
问题标签				
	评估类型	0	evaltype	0 - （默认）与/或； 2 - 或。
	标签名称	1	tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置问题标签时为必需
	运算符	0	tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置问题标签时为必需

参数		类型	名称	值
	标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 配置问题标签时为必需
时间段		1	time_period._referenceDASHBOARD._timeperiod - 将仪表板的时间段选择器设置为数据源； ABCDE._timeperiod - 将兼容的小组件（其引用参数设置为"ABCDE"） 设置为数据源。 默认值：DASHBOARD._timeperiod 或者，您也可以仅在从和到参数中设置时间段。
	从	1	time_period.from	采用绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、 now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 未设置时间段时支持 - 设置 time_period.to 时为必需
	到	1	time_period.to	采用绝对时间（YYYY-MM-DD hh:mm:ss）或相对时间语法（now、 now/d、now/w-1w 等）的有效时间字符串。 参数行为： - 未设置时间段时支持 - 设置 time_period.from 时为必需
触发器限制		0	show_lines	可能值的范围为 1 到 100。 默认值：10。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 *Top* 触发器组件。更多信息，请参考创建仪表盘。

配置 *Top triggers* 小组件

配置一个 *Top triggers* 小组件，使其显示主机组 "4" 的前 5 个触发器，以及每个触发器的所有问题计数。该小组件仅显示严重性为 "Warning" "Average" "High" 或 "Disaster" 的触发器，以及带有名称为 "scope" 的标签且其值包含 "performance" "availability" 或 "capacity" 的问题。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "toptriggers",
                        "name": "Top triggers",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
```

```
                    {
                        "type": 2,
                        "name": "groupids.0",
                        "value": 4
                    },
                    {
                        "type": 0,
                        "name": "severities.0",
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "severities.1",
                        "value": 3
                    },
                    {
                        "type": 0,
                        "name": "severities.2",
                        "value": 4
                    },
                    {
                        "type": 0,
                        "name": "severities.3",
                        "value": 5
                    },
                    {
                        "type": 1,
                        "name": "tags.0.tag",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.0.operator",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.0.value",
                        "value": "performance"
                    },
                    {
                        "type": 1,
                        "name": "tags.1.tag",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.1.operator",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.1.value",
                        "value": "availability"
                    },
                    {
                        "type": 1,
                        "name": "tags.2.tag",
                        "value": "scope"
                    },
                    {
                        "type": 0,
```

```
                                "name": "tags.2.operator",
                                "value": 0
                            },
                            {
                                "type": 1,
                                "name": "tags.2.value",
                                "value": "capacity"
                            },
                            {
                                "type": 0,
                                "name": "show_lines",
                                "value": 5
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

31 触发器概述

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置触发器概览 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新触发器概览小组件，请参考下表中所述的参数行为。

参数

以下参数适用于 触发器概览小部件。

参数		type	name	value
刷新间隔		0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
显示		0	show	1 - (默认) 最近的问题； 2 - 任意； 3 - 问题。
主机组		2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个 dashboard 小部件字段对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该小部件，则不支持此参数。
	主机组（小部件）	1	groupids._reference	用于代替主机组 ID： ABCDE._hostgroupids - 将兼容的小部件（其 *Reference* 参数设置为"ABCDE"）设置为主机组的数据源。
主机		3	hostids.0	如果在模板仪表板上配置该小部件，则不支持此参数。 主机 ID。 注意：要配置多个主机，请为每个主机创建一个 dashboard 小部件字段对象，并在属性名称中使用递增的编号。对于多个主机，参数 主机组必须要么完全不配置，要么至少配置一个已配置主机所属的主机组。 如果在模板仪表板上配置该小部件，则不支持此参数。
	主机（小部件/仪表板）	1	hostids._reference	用于代替主机 ID： DASHBOARD.hostids - 将仪表板主机选择器设置为主机的数据源； ABCDE._hostids - 将兼容的小部件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源。 如果在模板仪表板上配置该小部件，则不支持此参数。
问题标签				
	评估类型	0	evaltype	0 - (默认) And/Or； 2 - Or。
	标签名称	1	tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必填
	运算符	0	tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必填

参数	type	name	value
标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 问题标签，则为 必填
显示已抑制的问题	0	show_suppressed	0 - (默认) 禁用； 1 - 启用。
布局	0	layout	0 - (默认) 水平； 1 - 垂直。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 触发器概览组件。更多信息，请参考创建仪表盘。

配置一个触发器概览组件

配置一个触发器概览组件，用于展示所有主机组中包含如下标签的触发器。标签名称为：”scope”，标签值包含”availability”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "trigover",
                        "name": "Trigger overview",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "tags.0.tag",
                                "value": "scope"
                            },
                            {
                                "type": 0,
                                "name": "tags.0.operator",
                                "value": 0
                            },
                            {
                                "type": 1,
                                "name": "tags.0.value",
                                "value": "availability"
                            }
                        ]
```

```
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

32 URL

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 `dashboard.create` 和 `dashboard.update` 方法中配置*URL* 小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件 `fields` 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *URL* 小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *URL* 部件。

参数	type	name	value
刷新间隔	0	rf_rate	0 - *(默认)* 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。

参数	type	name	value
URL	1	url	有效的 URL 字符串。 参数行为： - 必填
覆盖主机	1	override_hostid._reference	ABCDE._hostid - 将兼容部件（其 *Reference* 参数设置为"ABCDE"）设置为主机的数据源； DASHBOARD._hostid - 将仪表板主机选择器设置为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 *URL* 组件。更多信息，请参考创建仪表盘。

配置 *URL* 小组件

配置一个显示 Zabbix 手册主页的 *URL* 小组件。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "url",
                        "name": "URL",
                        "x": 0,
                        "y": 0,
                        "width": 36,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "url",
                                "value": "https://www.zabbix.com/documentation/8.0/en"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
```

```
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

33 Web 监控

描述

这些参数以及相应仪表板小组件字段对象的可用属性值，可用于在 dashboard.create 和 dashboard.update 方法中配置*Web* 监控小组件。

Attention:
在创建或更新仪表板期间，不会验证小组件的 fields 属性。这使用户能够修改内置小组件并创建自定义小组件，但也带来了错误创建或更新小组件的风险。为确保成功创建或更新 *Web* 监控小组件，请参阅下表中所述的参数行为。

参数

以下参数适用于 *Web* 监测部件。

参数	type	name	value
刷新间隔	0	rf_rate	0 - 不刷新； 10 - 10 秒； 30 - 30 秒； 60 - (默认) 1 分钟； 120 - 2 分钟； 600 - 10 分钟； 900 - 15 分钟。
主机组	2	groupids.0	主机组 ID。 注意：要配置多个主机组，请为每个主机组创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该部件，则不支持此参数。
主机组（部件）	1	groupids._reference	用于替代主机组 ID： ABCDE._hostgroupids - 将一个兼容部件（其 *Reference* 参数设置为"ABCDE"）设为主机组的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。
排除主机组	2	exclude_groupids.0	主机组 ID。 注意：要排除多个主机组，请为每个主机组创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。 如果在模板仪表板上配置该部件，则不支持此参数。

参数		type	name	value
主机		3	hostids.0	主机 ID。 注意：要配置多个主机，请为每个主机创建一个 dashboard widget field 对象，并在属性名称中使用递增的编号。对于多个主机，参数 主机组要么完全不配置，要么至少配置一个包含这些已配置主机的主机组。 如果在模板仪表板上配置该部件，则不支持此参数。
	主机（部件/仪表板）	1	hostids._reference	用于替代主机 ID： `DASHBOARD.hostids` - 将仪表板的主机选择器设为主机的数据源； `ABCDE._hostids` - 将一个兼容部件（其 *Reference* 参数设置为"ABCDE"）设为主机的数据源。 如果在模板仪表板上配置该部件，则不支持此参数。
场景标签				
	评估类型	0	evaltype	0 - *(默认)* And/Or； 2 - Or。
	标签名称	1	tags.0.tag	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 场景标签，则为 必需
	运算符	0	tags.0.operator	0 - 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 场景标签，则为 必需
	标签值	1	tags.0.value	任意字符串值。 注意：属性名称中的数字表示标签评估列表中的标签顺序。 参数行为： - 如果配置了 场景标签，则为 必需
显示维护中的主机		0	maintenance	0 - 禁用； 1 - *(默认)* 启用。
引用		1	reference	由 5 个字符组成的任意字符串值（例如 ABCDE 或 JBPNL）。该值在该部件所属的仪表板内必须唯一。 参数行为： - 必需

示例

以下示例仅用于介绍如何在仪表盘组件字段对象中配置 *Web* 监测组件。更多信息，请参考创建仪表盘。

配置一个 *Web* 监测组件

配置一个 *Web* 监测组件，用于展示主机组"4" 中已启用的 Web 场景的状态概览。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "web",
                        "name": "Web monitoring",
                        "x": 0,
                        "y": 0,
                        "width": 18,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids.0",
                                "value": 4
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

参考

- 仪表盘组件字段
- 创建仪表盘
- 更新仪表盘

任务

此类用于处理任务（例如检查监控项或低级别发现规则，而无需重新加载配置）。

对象引用：

- Task
  - ’立即执行’请求对象
  - ’刷新 proxy 配置’请求对象
  - ’诊断信息’请求对象
    * 统计请求对象
    * 统计结果对象

可用方法：

- task.create - 创建新任务
- task.get - 检索任务

任务对象

以下对象与 task API 直接相关。

任务

任务对象具有以下属性：

属性	类型	描述
taskid	ID	任务的 ID。 属性行为： - 只读
type	integer	任务类型。 可能的值： 1 - 诊断信息； 2 - 刷新 proxy 配置； 6 - 立即执行。 *Admin* 和 *User* 类型的用户只能创建 “立即执行” 任务。 请注意，task.get 始终返回 “7”（任务执行摘要）。 属性行为： - *必需*
status	integer	任务状态。 可能的值： 1 - 新任务； 2 - 任务进行中； 3 - 任务已完成； 4 - 任务已过期。 属性行为： - 只读
clock	timestamp	任务创建时间。 属性行为： - 只读
ttl	integer	任务过期前的时间（以秒为单位）。 属性行为： - 只读

属性	类型	描述
proxyid	ID	收集其诊断信息统计数据的 proxy 的 ID。 属性行为： - 当 type 设置为 “诊断信息” 或 “刷新 proxy 配置” 时，支持
request	object	根据任务类型对应的任务请求对象： “立即执行” 任务的对象如下详细描述； “刷新 proxy 配置” 任务的对象如下详细描述； “诊断信息” 任务的对象如下详细描述。 属性行为： - 必需
result	object	诊断信息任务的结果对象。 如果结果尚未就绪，则可能包含 NULL。 结果对象如下详细描述。 属性行为： - 只读

“立即执行” 请求对象

“立即执行” 任务请求对象具有以下属性。

属性	类型	描述
itemid	ID	监控项和低级别发现规则的 ID。 *Admin* 和 *User* 类型的用户可对其具有 读写权限 的主机上的监控项执行 “立即执行”；如果其角色启用了 `invoke_execute_now` 操作，则具有 只读权限的用户也可执行 “立即执行”。对于低级别发现（LLD）规则，*Admin* 类型用户同样适用。

“刷新 proxy 配置” 请求对象

“刷新 proxy 配置” 任务请求对象具有以下属性。

属性	Type	说明
proxyids	array	Proxy ID。

“诊断信息” 请求对象

诊断信息任务请求对象具有以下属性。所有类型属性的统计请求对象已在下文详细说明。

属性	类型	描述
historycache	object	历史缓存统计请求。在服务器和 proxy 上可用。
valuecache	object	监控项缓存统计请求。在服务器上可用。
preprocessing	object	预处理管理器统计请求。在服务器和 proxy 上可用。
alerting	object	告警管理器统计请求。在服务器上可用。
lld	object	LLD 管理器统计请求。在服务器上可用。

统计信息请求对象

统计信息请求对象用于定义应收集有关服务器/proxy 内部进程的哪种类型的信息。它具有以下属性。

属性	类型	说明
stats	query	要返回的统计信息对象属性。 每种诊断信息统计类型的可用字段列表如下文详细说明。 默认值：`extend` 将返回所有可用的统计信息字段。

属性	类型	说明
top	object	用于对返回的统计信息值进行排序和限制的对象。 每种诊断信息统计类型的可用字段列表如下文详细说明。 示例：{ “source.alerts”: 10 }

每种诊断信息请求类型可用的统计字段列表

对于每种诊断信息请求属性，都可以请求以下统计字段。

诊断类型	可用字段	描述
historycache	items	已缓存监控项的数量。
	values	已缓存值的数量。
	memory	共享内存统计信息（可用空间、已使用块数量、空闲块数量、空闲块最大大小）。
	memory.data	历史数据缓存共享内存统计信息。
	memory.index	历史索引缓存共享内存统计信息。
valuecache	items	已缓存监控项的数量。
	values	已缓存值的数量。
	memory	共享内存统计信息（可用空间、已使用块数量、空闲块数量、空闲块最大大小）。
	mode	值缓存模式。
preprocessing	values	排队中的值数量。
	preproc.values	带有预处理步骤的排队中值数量。
alerting	alerts	排队中的告警数量。
lld	rules	排队中的规则数量。
	values	排队中的值数量。

每种诊断信息请求类型可用的排序字段列表

以下统计字段可用于对请求的信息进行排序和限制。

诊断类型	可用字段	类型
historycache	values	integer
valuecache	values	integer
	request.values	integer
preprocessing	values	integer
alerting	media.alerts	integer
	source.alerts	integer
lld	values	integer

统计结果对象

统计结果对象可在任务对象的 `result` 字段中获取。

属性	类型	描述
status	integer	任务结果的状态。 可能的值： -1 - 执行任务期间发生错误； 0 - 已创建任务结果。 属性行为： - 只读
data	string/object	特定诊断信息任务的统计请求对象对应的结果。 如果在执行任务期间发生错误，则包含错误消息字符串。

创建

描述

`object task.create(object/array tasks)`

此方法允许您创建任务。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的任务。

该方法接受具有标准任务属性的任务。

请注意，“立即执行” 任务只能为以下类型的监控项/发现规则创建：

- Zabbix agent（被动）
- 简单检查
- SNMP agent（v1/v2/v3）
- Zabbix 内部
- 外部检查
- 数据库监控
- HTTP agent
- IPMI agent
- SSH agent
- TELNET agent
- JMX agent
- 计算
- 依赖监控项
- 脚本
- 浏览器

如果监控项/发现规则的类型为 “依赖监控项”，则其主监控项也必须是上述类型之一。

返回值

(object) 在 taskids 属性下返回一个包含创建的任务 ID 的对象。每个监控项和低级别发现规则都会创建一个任务。返回的 ID 顺序与传递的 itemids 的顺序相匹配。

示例

创建任务

为一个监控项和一个低级别发现规则创建 “立即执行” 任务。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 6,
            "request": {
                "itemid": "10092"
            }
        },
        {
            "type": 6,
            "request": {
                "itemid": "10093"
            }
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "taskids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

为两个 proxy 创建 “刷新 proxy 配置” 任务。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 2,
            "request": {
                "proxyids": ["10459", "10460"]
            }
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1"
        ]
    },
    "id": 1
}
```

创建 “诊断信息” 任务。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 1,
            "request": {
                "alerting": {
                    "stats": [
                        "alerts"
                    ],
                    "top": {
                        "media.alerts": 10
                    }
                },
                "lld": {
                    "stats": "extend",
                    "top": {
                        "values": 5
                    }
                }
            },
            "proxyid": 0
```

```
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "3"
        ]
    },
    "id": 1
}
```

另请参阅

- 任务
- ' 立即执行' 请求对象
- ' 刷新 proxy 配置' 请求对象
- ' 诊断信息' 请求对象

来源

CTask::create() 在 *ui/include/classes/api/services/CTask.php*.

获取

描述

`integer/array task.get(object parameters)`

该方法允许您根据给定参数检索任务。

Note:
此方法仅适用于 *Super admin* 用户类型。在用户角色设置中可以撤销调用该方法的权限。有关更多信息，请参见User roles 。

参数

(object) 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
taskids	ID/array	仅返回具有给定 ID 的任务。
output	query	这些参数在参考说明中进行了描述。
preservekeys	boolean	

返回值

(integer/array) 返回一个对象的数组。

示例

通过 ID 获取任务

获取 ID 为 "1" 的任务的所有数据。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "task.get",
    "params": {
        "output": "extend",
        "taskids": "1"
```

```
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "taskid": "1",
            "type": "7",
            "status": "3",
            "clock": "1601039076",
            "ttl": "3600",
            "proxyid": null,
            "request": {
                "alerting": {
                    "stats": [
                        "alerts"
                    ],
                    "top": {
                        "media.alerts": 10
                    }
                },
                "lld": {
                    "stats": "extend",
                    "top": {
                        "values": 5
                    }
                }
            },
            "result": {
                "data": {
                    "alerting": {
                        "alerts": 0,
                        "top": {
                            "media.alerts": []
                        },
                        "time": 0.000663
                    },
                    "lld": {
                        "rules": 0,
                        "values": 0,
                        "top": {
                            "values": []
                        },
                        "time": 0.000442
                    }
                },
                "status": "0"
            }
        }
    ],
    "id": 1
}
```

另请参见

- 任务
- ' 立即执行' 请求对象
- ' 刷新 proxy 配置' 请求对象
- ' 诊断信息' 请求对象

来源

CTask::get() 在 *ui/include/classes/api/services/CTask.php*。

值映射

此类用于处理值映射。

对象参考：

- 值映射
  - 值映射关系

可用方法：

- valuemap.create - 创建新的值映射
- valuemap.delete - 删除值映射
- valuemap.get - 检索值映射
- valuemap.update - 更新值映射

值映射对象

以下对象与 `valuemap` API 直接相关。

值映射

值映射对象具有以下属性。

属性	类型	描述
valuemapid	ID	值映射的 ID。 属性行为： - 只读 - 更新操作时为必填
hostid	ID	值映射所属主机或模板的 ID。 属性行为： - 常量 - 创建操作时为必填
name	string	值映射的名称。 属性行为： - 创建操作时为必填
mappings	array	当前值映射的值映射关系。映射对象的详细说明见下文。 属性行为： - 创建操作时为必填
uuid	string	通用唯一标识符，用于将导入的值映射关联到已存在的值映射。如果未提供，则自动生成。 属性行为： - 如果值映射属于模板，则支持

值映射

值映射对象定义了值映射中的值映射规则。它具有以下属性。

属性	类型	描述
type	integer	映射匹配类型。 可能的值： 0 - (默认) 当值相等时应用映射； 1 - 当值大于或等于时应用映射[1]； 2 - 当值小于或等于时应用映射[1]； 3 - 当值在范围内时应用映射（范围包含边界；可定义多个范围，以逗号分隔）[1]； 4 - 当值匹配正则表达式时应用映射[2]； 5 - 如果未找到匹配项，则不应用映射，并使用默认值。 如果 type 设置为"0"、"1"、"2"、"3"、"4"，则 value 不能为空。 如果 type 设置为"5"，则 value 必须为空。
value	string	原始值。 属性行为： - 如果 type 设置为"1"、"2"、"3"、"4"，则为 必需 - 如果 type 设置为"5"，则 支持
newvalue	string	原始值将被映射为的值。 属性行为： - 必需

[1] 仅支持值类型为"numeric unsigned"、"numeric float" 的监控项。
 [2] 仅支持值类型为"character" 的监控项。

创建

描述

`object valuemap.create(object/array valuemaps)`

此方法允许您创建新的值映射。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的值映射。

该方法接受具有标准值映射属性的值映射。

返回值

`(object)` 返回一个对象，其中包含 `valuemapids` 属性所创建值映射的 ID。返回 ID 的顺序与传递的值映射的顺序相匹配。

示例

创建一个值映射

创建一个包含两个映射的值映射。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.create",
    "params": {
        "hostid": "50009",
        "name": "Service state",
        "mappings": [
            {
                "type": "1",
                "value": "1",
```

```
                "newvalue": "Up"
            },
            {
                "type": "5",
                "newvalue": "Down"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1"
        ]
    },
    "id": 1
}
```

源码

ui/include/classes/api/services/CValueMap.php 中的 CValueMap::create()。

删除

描述

`object valuemap.delete(array valuemapids)`

此方法允许您删除值映射。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 需要删除的值映射的 ID。

返回值

(object) 返回一个包含被删除的值映射 ID 的对象，ID 在 `valuemapids` 属性下。

示例

删除多个值映射

删除两个值映射。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

源码

ui/include/classes/api/services/CValueMap.php 中的 CValueMap::delete()。

更新

描述

`object valuemap.update(object/array valuemaps)`

此方法允许您更新现有的值映射。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色 。

参数

(object/array) 要更新的值映射属性。

必须为每个值映射定义 valuemapid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

返回值

(object) 返回一个对象，包含被更新的值映射的 ID，ID 在 valuemapids 属性下。

示例

更新值映射名称

将值映射名称更新为 “Device status”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "name": "Device status"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "2"
        ]
    },
    "id": 1
}
```

更改一个值映射的映射。

请求:

```
{
"jsonrpc": "2.0",
"method": "valuemap.update",
"params": {
"valuemapid": "2",
"mappings": [
{
"type": "0",
"value": "0",
"newvalue": "Online"
},
{
"type": "0",
"value": "1",
"newvalue": "Offline"
}
]
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"valuemapids": [
"2"
]
},
"id": 1
}
```

源码

ui/include/classes/api/services/CValueMap.php 中的 CValueMap::update()。

获取

描述

`integer/array valuemap.get(object parameters)`

该方法允许你根据给定参数检索值映射。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
valuemapids	ID/array	仅返回具有给定 ID 的值映射。
selectMappings	query	在当前值映射的`mappings` 属性中返回值映射条目。 支持 `count`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`valuemapid`、`name`。
countOutput	boolean	这些参数在参考注释中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	

参数	类型	描述
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下其中一种结果：

- 一个数组对象；
- 如果使用了参数 countOutput，则返回检索到的对象的数量。

示例

检索值映射

检索所有配置的值映射。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "valuemapid": "4",
            "name": "APC Battery Replacement Status"
        },
        {
            "valuemapid": "5",
            "name": "APC Battery Status"
        },
        {
            "valuemapid": "7",
            "name": "Dell Open Manage System Status"
        }
    ],
    "id": 1
}
```

根据映射关系，检索一个值映射。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend",
        "selectMappings": "extend",
        "valuemapids": ["4"]
    },
```

```
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "valuemapid": "4",
            "name": "APC Battery Replacement Status",
            "mappings": [
                {
                    "type": "0",
                    "value": "1",
                    "newvalue": "unknown"
                },
                {
                    "type": "0",
                    "value": "2",
                    "newvalue": "notInstalled"
                },
                {
                    "type": "0",
                    "value": "3",
                    "newvalue": "ok"
                },
                {
                    "type": "0",
                    "value": "4",
                    "newvalue": "failed"
                },
                {
                    "type": "0",
                    "value": "5",
                    "newvalue": "highTemperature"
                },
                {
                    "type": "0",
                    "value": "6",
                    "newvalue": "replaceImmediately"
                },
                {
                    "type": "0",
                    "value": "7",
                    "newvalue": "lowCapacity"
                }
            ]
        }
    ],
    "id": 1
}
```

源码

ui/include/classes/api/services/CValueMap.php 中的 CValueMap::get()。

关联

此类用于处理关联。

对象引用：

- 关联

- 关联操作
- 关联过滤器
  - 关联过滤器条件

可用方法：

- correlation.create - 创建新的关联
- correlation.delete - 删除关联
- correlation.get - 检索关联
- correlation.update - 更新关联

关联对象

以下对象与关联 API 直接相关。

关联

关联对象具有以下属性。

属性	类型	描述
correlationid	ID	关联的 ID。 属性行为： - 只读 - 更新操作时 必填
name	string	关联的名称。 属性行为： - 创建操作时 必填
description	string	关联的描述。
status	integer	关联是启用还是禁用。 可能的值： 0 - (默认) 启用； 1 - 禁用。

关联操作

关联操作对象定义了在执行关联时将执行的操作。它具有以下属性。

Property	Type	Description
type	integer	操作类型。 可能的值： 0 - 关闭旧事件； 1 - 关闭新事件。 属性行为： - 必填

关联过滤器

关联过滤器对象定义了一组必须满足的条件，以执行已配置的关联操作。它具有以下属性。

Property	Type	Description
conditions	array	用于筛选结果的一组过滤条件。这些条件将按照它们在公式中的排列顺序进行排序。 属性行为: - 必需

Property	Type	Description
evaltype	integer	过滤条件的求值方法。 可能的值： 0 - And/Or; 1 - And; 2 - Or; 3 - 自定义表达式。 属性行为: - 必需
eval_formula	string	用于计算过滤条件的生成表达式。该表达式包含引用特定过滤条件的 ID，这些 ID 通过其 `formulaid` 标识。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 属性行为: - 只读
formula	string	用户定义的表达式，用于计算使用自定义表达式的过滤器条件。该表达式必须包含引用特定过滤条件的 ID，这些 ID 通过其 `formulaid` 标识。表达式中使用的 ID 必须与过滤条件中定义的完全一致：不能有任何条件未使用或被省略。 属性行为: - 当 `evaltype` 设置为”custom expression” 时为必需

关联筛选条件

关联筛选条件对象定义了一个特定条件，在运行关联操作之前必须先检查该条件。

属性	类型	描述
type	integer	条件类型。 可能的值： 0 - 旧事件标签； 1 - 新事件标签； 2 - 新事件主机组； 3 - 事件标签对； 4 - 旧事件标签值； 5 - 新事件标签值。 属性行为： - 必需
tag	string	事件标签（旧或新）。 属性行为： - 如果 `type` 设置为 “旧事件标签”、“新事件标签”、“旧事件标签值” 或 “新事件标签值”，则为 必需
groupid	ID	主机组的 ID。 属性行为： - 如果 `type` 设置为 “新事件主机组”，则为 必需
oldtag	string	旧事件标签。 属性行为： - 如果 `type` 设置为 “事件标签对”，则为 必需
newtag	string	旧事件标签。 属性行为： - 如果 `type` 设置为 “事件标签对”，则为 必需

属性	类型	描述
value	string	事件标签（旧或新）的值。 属性行为： - 如果 `type` 设置为 “旧事件标签值” 或 “新事件标签值”，则为 必需
formulaid	string	用于在自定义表达式中引用该条件的任意唯一 ID。只能包含大写字母。修改筛选条件时，该 ID 必须由用户定义；但在之后请求这些条件时，将重新生成该 ID。
operator	integer	条件运算符。 属性行为： - 如果 `type` 设置为 “新事件主机组”、“旧事件标签值” 或 “新事件标签值”，则为 必需

Note:
为了更好地理解如何将筛选器与各种类型的表达式配合使用，请参见correlation.get 和correlation.create 方法页面中的示例。

以下是每种条件类型支持的运算符和值。

条件	条件名称	支持的运算符	预期值
2	主机组	=, <>	主机组 ID。
4	旧事件标签值	=, <>, like, not like	string
5	新事件标签值	=, <>, like, not like	string

创建

描述

`object correlation.create(object/array correlations)`

此方法允许您创建新的关联。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

`(object/array)` 要创建的关联。

除了标准关联属性之外，该方法还接受以下参数。

参数	类型	描述
operations	array	要为关联创建的关联操作。 参数行为: - 必需
filter	object	关联的关联过滤器对象。 参数行为: - 必需

返回值

`(object)` 返回一个对象，该对象包含 `correlationids` 属性下创建的关联的 ID。返回的 ID 的顺序与所传递的关联的顺序相匹配。

示例

创建新的事件标签关联

使用评估方法 `AND/OR` 创建一个包含一个条件和一个操作的关联。默认情况下，该关联将处于启用状态。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new event tag correlation",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "type": 1,
                    "tag": "ok"
                }
            ]
        },
        "operations": [
            {
                "type": 0
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

使用自定义表达式过滤器

创建一个将使用自定义过滤条件的事件关联。公式 ID “A” 或 “B” 是任意选择的。条件类型将为 “主机群组”，运算符为 “<>”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new host group correlation",
        "description": "a custom description",
        "status": 0,
        "filter": {
            "evaltype": 3,
            "formula": "A or B",
            "conditions": [
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "A"
                },
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "B"
                }
            ]
        },
        "operations": [
```

```
            {
                "type": 1
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "2"
        ]
    },
    "id": 1
}
```

另请参阅

- 关联过滤器
- 关联操作

来源

ui/include/classes/api/services/CCorrelation.php 中的 CCorrelation::create()。

删除

描述

`object correlation.delete(array correlationids)`

此方法允许您删除关联。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 要删除的关联 ID。

返回值

(object) 返回一个对象，该对象包含 correlationids 属性下删除的关联 ID。

示例

删除多个事件关联

删除两个事件关联。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "correlation.delete",
    "params": [
        "1",
        "2"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CCorrelation.php 中的 CCorrelation::delete()。

更新

描述

`object correlation.update(object/array correlations)`

此方法允许您更新现有的关联。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的关联属性。

必须为每个关联定义 correlationid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准关联属性之外，该方法还接受以下参数。

参数	Type	说明
filter	object	用于替换当前过滤器的关联过滤器对象。
operations	array	用于替换现有操作的关联操作。

返回值

(object) 返回一个对象，该对象包含 “correlationids” 属性下更新的关联的 ID。

示例

禁用关联

请求:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "status": "1"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
```

```
    "id": 1
}
```

替换条件，但保留评估方法

请求:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "filter": {
            "conditions": [
                {
                    "type": 3,
                    "oldtag": "error",
                    "newtag": "ok"
                }
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

另请参阅

- 关联过滤器
- 关联操作

来源

ui/include/classes/api/services/CCorrelation.php 中的 CCorrelation::update()。

获取

描述

`integer/array correlation.get(object parameters)`

该方法允许你根据给定参数检索关联关系。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
correlationids	ID/array	仅返回具有给定 ID 的事件关联。
selectFilter	query	返回一个包含事件关联条件的`filter` 属性。
selectOperations	query	返回一个包含事件关联操作的`operations` 属性。

参数	类型	描述
sortfield	string/array	按给定属性对结果进行排序。 可选值：`correlationid`、`name`、`status`。
countOutput	boolean	这些参数在参考注释中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回其中之一：

- 一个对象数组；
- 如果使用了 countOutput 参数，将返回获取对象的数量。

示例

检索关联关系

检索所有已配置的关联关系及其关联条件和操作。该过滤器使用 “and/or” 求值类型，因此 `formula` 属性为空，`eval_formula` 会自动生成。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectFilter": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "correlationid": "1",
            "name": "Correlation 1",
            "description": "",
            "status": "0",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "type": "3",
                        "oldtag": "error",
                        "newtag": "ok",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A"
            },
```

```
            "operations": [
                {
                    "type": "0"
                }
            ]
        }
    ],
    "id": 1
}
```

另请参阅

- 关联筛选器
- 关联操作

来源

ui/include/classes/api/services/CCorrelation.php 中的 CCorrelation::get()。

动作

此类用于处理动作。

对象引用：

- 动作
- 动作操作
  - 动作操作消息
  - 动作操作条件
- 动作恢复操作
- 动作更新操作
- 动作过滤器
  - 动作过滤器条件

可用方法：

- action.create - 创建新动作
- action.delete - 删除动作
- action.get - 检索动作
- action.update - 更新动作

动作对象

以下对象与动作 API 直接相关。

Action

action 对象具有以下属性。

Property	Type	Description
actionid	ID	action 的 ID。 属性行为： - 只读 - 更新操作中为 必需
esc_period	string	默认操作步骤持续时间。必须至少为 60 秒。可接受秒数（例如 60、300）、带有时间后缀的值（例如 30s、1m、2h）或用户宏。 属性行为： - 当 `eventsource` 设置为 “由触发器创建的事件”、“内部事件” 或 “服务状态更新时创建的事件” 时支持

Property	Type	Description
eventsource	integer	action 将处理的事件类型。 支持的事件类型列表请参阅事件的 source 属性。 属性行为： - 常量 - 创建操作中为 必需
name	string	action 的名称。 属性行为： - 创建操作中为 必需
status	integer	action 是否启用。 可能的值： 0 - （默认）启用； 1 - 禁用。
pause_symptoms	integer	如果事件为症状事件，是否暂停升级。 可能的值： 0 - 不暂停症状问题的升级； 1 - （默认）暂停症状问题的升级。 属性行为： - 当 eventsource 设置为 “由触发器创建的事件” 时支持
pause_suppressed	integer	在维护期间是否暂停升级。 可能的值： 0 - 不暂停升级； 1 - （默认）暂停升级。 属性行为： - 当 eventsource 设置为 “由触发器创建的事件” 时支持
notify_if_canceled	integer	升级取消时是否发送通知。 可能的值： 0 - 升级取消时不发送通知； 1 - （默认）升级取消时发送通知。 属性行为： - 当 eventsource 设置为 “由触发器创建的事件” 时支持

操作

操作对象定义了在执行操作时将执行的操作。它具有以下属性。

属性	类型	描述
operationtype	integer	操作类型。 可能的值： 0 - 发送消息； 1 - 全局脚本； 2 - 添加主机； 3 - 移除主机； 4 - 添加到主机组； 5 - 从主机组中移除； 6 - 链接模板； 7 - 取消链接模板； 8 - 启用主机； 9 - 禁用主机； 10 - 设置主机资产清单模式； 13 - 添加主机标签； 14 - 移除主机标签。 如果Action object 的 `eventsource` 设置为“由触发器创建的事件”或“服务状态更新时创建的事件”，则可能的值： 0 - “发送消息”； 1 - “全局脚本”。 如果Action object 的 `eventsource` 设置为“内部事件”，则可能的值： 0 - “发送消息”。 属性行为： - 必需
esc_period	string	升级步骤的持续时间，单位为秒。必须大于 60 秒。接受秒数（例如 60、300）、带有时间后缀的值（例如 30s、1m、2h）或用户宏。如果设置为 0 或 0s，则使用默认的操作升级周期。 默认值：0s。 属性行为： - 当Action object 的 `eventsource` 设置为“由触发器创建的事件”、“内部事件”或“服务状态更新时创建的事件”时，支持
esc_step_from	integer	开始升级的步骤。 默认值：1。 属性行为： - 当Action object 的 `eventsource` 设置为“由触发器创建的事件”、“内部事件”或“服务状态更新时创建的事件”时，支持
esc_step_to	integer	结束升级的步骤。 默认值：1。 属性行为： - 当Action object 的 `eventsource` 设置为“由触发器创建的事件”、“内部事件”或“服务状态更新时创建的事件”时，支持
evaltype	integer	操作条件的评估方法。 可能的值： 0 - *(默认)* And/Or； 1 - And； 2 - Or。
opcommand	object	要执行的全局脚本。 全局脚本必须定义 `scriptid` 属性。 属性行为： - 当 `operationtype` 设置为“全局脚本”时，必需

属性	类型	描述
opcommand_grp	array	要在其上运行全局脚本的主机组。 主机组必须定义 groupid 属性。 属性行为： - 当 operationtype 设置为 “全局脚本” 且未设置 opcommand_hst 时，必需
opcommand_hst	array	要在其上运行全局脚本的主机。 主机必须定义 hostid 属性。 属性行为： - 当 operationtype 设置为 “全局脚本” 且未设置 opcommand_grp 时，必需
opconditions	array	用于触发器操作的操作条件。 操作条件对象在下文中有详细说明。
opgroup	array	要将主机添加到的主机组。 主机组必须定义 groupid 属性。 属性行为： - 当 operationtype 设置为 “添加到主机组” 或 “从主机组中移除” 时，必需
opmessage	object	包含操作发送的消息数据的对象。 操作消息对象在下文中有详细说明。 属性行为： - 当 operationtype 设置为 “发送消息” 时，必需
opmessage_grp	array	要向其发送消息的用户组。 用户组必须定义 usrgrpid 属性。 属性行为： - 当 operationtype 设置为 “发送消息” 且未设置 opmessage_usr 时，必需
opmessage_usr	array	要向其发送消息的用户。 用户必须定义 userid 属性。 属性行为： - 当 operationtype 设置为 “发送消息” 且未设置 opmessage_grp 时，必需
optemplate	array	要链接到主机的模板。 模板必须定义 templateid 属性。 属性行为： - 当 operationtype 设置为 “链接模板” 或 “取消链接模板” 时，必需
opinventory	object	要为主机设置的资产清单模式。 资产清单必须定义 inventory_mode 属性。 属性行为： - 当 operationtype 设置为 “设置主机资产清单模式” 时，必需

属性	类型	描述
optag	array	要添加或移除的主机标签。 标签必须定义 `tag` 属性。 `value` 属性是可选的。 属性行为： - 当 `operationtype` 设置为 “添加主机标签” 或 “移除主机标签” 时，支持

操作消息

操作消息对象包含将由该操作发送的消息相关数据。它具有以下属性。

属性	类型	描述
default_msg	integer	是否使用默认的动作消息文本和主题。 可能的值： 0 - 使用操作中的数据； 1 - *(默认)* 使用媒体类型中的数据。
mediatypeid	ID	将用于发送消息的媒体类型 ID。 属性行为： - 如果动作操作对象、动作恢复操作对象 或动作更新操作对象 的 `operationtype` 设置为 “发送消息”，或者动作更新操作对象 的 `operationtype` 设置为 “通知所有相关人员”，则为 *supported*
message	string	操作消息文本。 属性行为： - 如果 `default_msg` 设置为 “使用操作中的数据”，则为 *supported*
subject	string	操作消息主题。 属性行为： - 如果 `default_msg` 设置为 “使用操作中的数据”，则为 *supported*

操作条件

操作条件对象定义了执行当前操作必须满足的条件。它具有以下属性。

属性	Type	描述
conditiontype	integer	条件类型。 可能的值： 14 - 事件已确认。 属性行为： - 必填
value	string	用于比较的值。 属性行为： - 必填
operator	integer	条件运算符。 可能的值： 0 - *(默认)* =

以下是每种操作条件类型支持的运算符和值。

条件	条件名称	支持的运算符	预期值
14	事件已确认	=	事件是否已确认。 可能的值： 0 - 未确认； 1 - 已确认。

动作恢复操作

动作恢复操作对象定义了在问题被解决时将执行的操作。
恢复操作仅适用于触发器、内部和服务动作。
它具有以下属性。

属性	类型	说明
operationtype	integer	操作类型。 如果Action object 的 eventsource 设置为 “由触发器创建的事件” 或 “在服务状态更新时创建的事件”，则可能的值为： 0 - 发送消息； 1 - 全局脚本； 11 - 通知所有相关人员。 如果Action object 的 eventsource 设置为 “内部事件”，则可能的值为： 0 - 发送消息； 11 - 通知所有相关人员。 属性行为： - 必需
opcommand	object	要执行的全局脚本。 全局脚本必须定义 scriptid 属性。 属性行为： - 如果 operationtype 设置为 “全局脚本”，则为 必需
opcommand_grp	array	要在其上运行全局脚本的主机组。 主机组必须定义 groupid 属性。 属性行为： - 如果Action object 的 eventsource 设置为 “由触发器创建的事件”，且 operationtype 设置为 “全局脚本”，并且未设置 opcommand_hst，则为 必需
opcommand_hst	array	要在其上运行全局脚本的主机。 主机必须定义 hostid 属性。 属性行为： - 如果Action object 的 eventsource 设置为 “由触发器创建的事件”，且 operationtype 设置为 “全局脚本”，并且未设置 opcommand_grp，则为 必需
opmessage	object	包含恢复操作所发送消息数据的对象。 操作消息对象已在上文详细说明。 属性行为： - 如果 operationtype 设置为 “发送消息”，则为 必需
opmessage_grp	array	要向其发送消息的用户组。 用户组必须定义 usrgrpid 属性。 属性行为： - 如果 operationtype 设置为 “发送消息” 且未设置 opmessage_usr，则为 必需

属性	类型	说明
opmessage_usr	array	要向其发送消息的用户。 用户必须定义 `userid` 属性。 属性行为： - 如果 `operationtype` 设置为 “发送消息” 且未设置 `opmessage_grp`，则为 必需

操作更新操作

操作更新操作对象定义了在问题被更新时将执行的操作（添加评论、确认、严重性变更或手动关闭）。更新操作仅适用于触发器和服务动作。它具有以下属性。

Property	Type	Description
operationtype	integer	操作类型。 可能的值： 0 - 发送消息； 1 - 全局脚本； 12 - 通知所有相关人员。 属性行为： - 必填
opcommand	object	要执行的全局脚本。 全局脚本必须定义 `scriptid` 属性。 属性行为： - 如果 `operationtype` 设置为 “全局脚本”，则为 必填
opcommand_grp	array	要在其上运行全局脚本的主机组。 主机组必须定义 `groupid` 属性。 属性行为： - 如果Action object 的 `eventsource` 设置为 “由触发器创建的事件”，且 `operationtype` 设置为 “全局脚本”，并且未设置 `opcommand_hst`，则为 必填
opcommand_hst	array	要在其上运行全局脚本的主机。 主机必须定义 `hostid` 属性。 属性行为： - 如果Action object 的 `eventsource` 设置为 “由触发器创建的事件”，且 `operationtype` 设置为 “全局脚本”，并且未设置 `opcommand_grp`，则为 必填
opmessage	object	包含更新操作所发送消息数据的对象。 操作消息对象在上文中有详细说明。
opmessage_grp	array	要发送消息到的用户组。 用户组必须定义 `usrgrpid` 属性。 属性行为： - 如果 `operationtype` 设置为 “发送消息” 且未设置 `opmessage_usr`，则为 必填

Property	Type	Description
opmessage_usr	array	要发送消息到的用户。 用户必须定义 `userid` 属性。 属性行为： - 如果 `operationtype` 设置为 “发送消息” 且未设置 `opmessage_grp`，则为 必填

操作筛选器

操作筛选器对象定义了一组必须满足的条件，以执行已配置的操作。它具有以下属性。

Property	Type	Description
conditions	array	用于筛选结果的筛选条件集合。条件将按照其在公式中的排列顺序进行排序。 属性行为: - 必需
evaltype	integer	筛选条件的求值方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 属性行为: - 必需
eval_formula	string	用于对筛选条件进行求值的生成表达式。该表达式包含通过其 `formulaid` 引用特定筛选条件的 ID。对于使用自定义表达式的筛选器，`eval_formula` 的值等于 `formula` 的值。 属性行为: - 只读
formula	string	用户定义的表达式，用于对使用自定义表达式的筛选器条件进行求值。该表达式必须包含通过其 `formulaid` 引用特定筛选条件的 ID。表达式中使用的 ID 必须与筛选条件中定义的 ID 完全匹配：不能有任何条件未被使用或省略。 属性行为: - 如果 `evaltype` 设置为 “自定义表达式”，则为 必需

操作筛选条件

操作筛选条件对象定义了在运行操作之前必须检查的特定条件。

属性	类型	描述
conditiontype	integer	条件类型。 如果Action 对象的 `eventsource` 设置为“由触发器创建的事件”，可能的值为： 0 - 主机组； 1 - 主机； 2 - 触发器； 3 - 事件名称； 4 - 触发器严重性； 6 - 时间段； 13 - 主机模板； 16 - 问题已被抑制； 25 - 事件标签； 26 - 事件标签值。 如果Action 对象的 `eventsource` 设置为“由发现规则创建的事件”，可能的值为： 7 - 主机 IP； 8 - 已发现服务类型； 9 - 已发现服务端口； 10 - 发现状态； 11 - 正常运行或停机持续时间； 12 - 接收值； 18 - 发现规则； 19 - 发现检查； 20 - proxy； 21 - 发现对象； 29 - proxy 组。 如果Action 对象的 `eventsource` 设置为“由 active agent 自动注册创建的事件”，可能的值为： 20 - proxy； 22 - 主机名称； 24 - 主机元数据； 29 - proxy 组。 如果Action 对象的 `eventsource` 设置为“内部事件”，可能的值为： 0 - 主机组； 1 - 主机； 13 - 主机模板； 23 - 事件类型； 25 - 事件标签； 26 - 事件标签值。 如果Action 对象的 `eventsource` 设置为“服务状态更新时创建的事件”，可能的值为： 25 - 事件标签； 26 - 事件标签值； 27 - 服务； 28 - 服务名称。 属性行为： - 必需
value	string	用于比较的值。 属性行为： - 必需

属性	类型	描述
value2	string	用于比较的次要值。 属性行为： - 当Action 对象的 eventsource 设置为 “由触发器创建的事件”、conditiontype 设置为触发器操作的任意可能值，且条件类型（见下文）为 “26” 时，必需 - 当Action 对象的 eventsource 设置为 “内部事件”、conditiontype 设置为内部操作的任意可能值，且条件类型（见下文）为 “26” 时，必需 - 当Action 对象的 eventsource 设置为 “服务状态更新时创建的事件”、conditiontype 设置为服务操作的任意可能值，且条件类型（见下文）为 “26” 时，必需
formulaid	string	用于从自定义表达式中引用条件的任意唯一 ID。只能包含大写字母。修改筛选条件时，ID 必须由用户定义；但之后请求这些条件时，ID 将重新生成。
operator	integer	条件运算符。 可能的值： 0 - *(默认)* 等于； 1 - 不等于； 2 - 包含； 3 - 不包含； 4 - 属于； 5 - 大于或等于； 6 - 小于或等于； 7 - 不属于； 8 - 匹配； 9 - 不匹配； 10 - 是； 11 - 否。

Note:
要更好地了解如何将筛选器与各种类型的表达式结合使用，请参阅action.get 和action.create 方法页面中的示例。

以下运算符和值适用于每种条件类型。

条件	条件名称	支持的运算符	预期值
0	主机组	等于、 不等于	主机组 ID。
1	主机	等于、 不等于	主机 ID。
2	触发器	等于、 不等于	触发器 ID。
3	事件名称	包含、 不包含	事件名称。
4	触发器严重性	等于、 不等于、 大于或等于、 小于或等于	触发器严重性。有关支持的触发器严重性列表，请参阅触发器 severity 属性。
5	触发器值	等于	触发器值。有关支持的触发器值列表，请参阅触发器 value 属性。
6	时间段	属于、不属于	事件触发时所在的时间段。
7	主机 IP	等于、 不等于	要检查的一个或多个 IP 范围，以逗号分隔。有关支持的 IP 范围格式的更多信息，请参阅网络发现配置部分。
8	已发现服务类型	等于、 不等于	已发现服务的类型。服务类型与用于检测该服务的发现检查类型相匹配。有关支持的类型列表，请参阅发现检查 type 属性。
9	已发现服务端口	等于、 不等于	一个或多个端口范围，以逗号分隔。

条件	条件名称	支持的运算符	预期值
10	发现状态	等于	已发现对象的状态。 可能的值： 0 - 主机或服务正常运行； 1 - 主机或服务停机； 2 - 已发现主机或服务； 3 - 主机或服务丢失。
11	正常运行或停机持续时间	大于或等于、 小于或等于	表示已发现对象处于当前状态的时长，单位为秒。
12	接收值	等于、 不等于、 大于或等于、 小于或等于、 包含、 不包含	执行 Zabbix agent、SNMPv1、SNMPv2 或 SNMPv3 发现检查时返回的值。
13	主机模板	等于、 不等于	已链接模板的 ID。
16	问题已被抑制	是、否	无需值：使用 “是” 运算符表示问题必须被抑制；“否” 表示问题未被抑制。
18	发现规则	等于、 不等于	发现规则的 ID。
19	发现检查	等于、 不等于	发现检查的 ID。
20	proxy	等于、 不等于	proxy 的 ID。
21	发现对象	等于	触发发现事件的对象类型。 可能的值： 1 - 已发现主机； 2 - 已发现服务。
22	主机名称	包含、 不包含、 匹配、 不匹配	主机名称。 在自动注册条件中，运算符匹配和不匹配支持使用正则表达式。
23	事件类型	等于	特定的内部事件。 可能的值： 0 - 监控项处于 “不支持” 状态； 1 - 监控项处于 “正常” 状态； 2 - LLD 规则处于 “不支持” 状态； 3 - LLD 规则处于 “正常” 状态； 4 - 触发器处于 “未知” 状态； 5 - 触发器处于 “正常” 状态。
24	主机元数据	包含、 不包含、 匹配、 不匹配	自动注册主机的元数据。 支持在运算符匹配和不匹配中使用正则表达式。
25	标签	等于、 不等于、 包含、 不包含	事件标签。
26	标签值	等于、 不等于、 包含、 不包含	事件标签值。
27	服务	等于、 不等于	服务 ID。
28	服务名称	等于、 不等于	服务名称。
29	proxy 组	等于、 不等于	proxy 组的 ID。

创建

描述

`object action.create(object/array actions)`

此方法允许你创建新的 action。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`object/array`) 要创建的动作。

除了标准动作属性外，该方法还接受以下参数。

参数	类型	描述
filter	object	该动作的动作过滤器对象。
operations	array	要为该动作创建的动作操作。
recovery_operations	array	要为该动作创建的动作恢复操作。
update_operations	array	要为该动作创建的动作更新操作。

返回值

(`object`) 返回一个对象，该对象在 `actionids` 属性下包含已创建动作的 ID。返回的 ID 顺序与传递的动作顺序相匹配。

示例

创建触发器动作

创建一个触发器动作，该动作将在主机 ID 为"10084" 上的触发器（名称中包含"memory"）进入 PROBLEM 状态时开始执行。该动作将配置 4 个操作。- 第一个也是立即执行的操作，将通过媒体类型"1" 向用户组 ID 为"7" 中的所有用户发送消息。- 如果事件在 30 分钟内未解决，第二个操作将在组"2" 中的所有主机上运行脚本"5"（来自" 动作操作" 按钮创建的脚本）。- 如果事件得到解决，所有与该问题相关的用户将收到一个恢复操作通知。- 如果事件被更新，所有与该问题相关的用户将收到一个确认/更新操作通知。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "esc_period": "30m",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084"
                },
                {
                    "conditiontype": 3,
                    "operator": 2,
                    "value": "memory"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_step_from": 1,
                "esc_step_to": 1,
```

```
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            },
            {
                "operationtype": 1,
                "esc_step_from": 2,
                "esc_step_to": 2,
                "opconditions": [
                    {
                        "conditiontype": 14,
                        "operator": 0,
                        "value": "0"
                    }
                ],
                "opcommand_grp": [
                    {
                        "groupid": "2"
                    }
                ],
                "opcommand": {
                    "scriptid": "5"
                }
            }
        ],
        "recovery_operations": [
            {
                "operationtype": "11",
                "opmessage": {
                    "default_msg": 1
                }
            }
        ],
        "update_operations": [
            {
                "operationtype": "12",
                "opmessage": {
                    "default_msg": 0,
                    "message": "Custom update operation message body",
                    "subject": "Custom update operation message subject"
                }
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17"
        ]
    },
    "id": 1
```

```
}
```

创建一个发现动作

创建一个发现动作，该动作将模板"10001" 链接到已发现的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Discovery action",
        "eventsource": 1,
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "operator": 0,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "operator": 0,
                    "value": "2"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 6,
                "optemplate": [
                    {
                        "templateid": "10001"
                    }
                ]
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

使用自定义表达式过滤

创建一个触发器动作，该动作使用自定义表达式 - "A and (B or C)" - 来评估动作条件。一旦来自主机"10084" 或主机"10106" 的告警级别大于等于"Warning" 触发器将进入 PROBLEM 状态，该动作将通过媒体类型"1" 向用户组"7" 中的所有用户发送消息。公式 ID "A"、"B" 和"C" 是任意选择的。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
```

```
        "name": "Trigger action",
        "eventsource": 0,
        "esc_period": "15m",
        "filter": {
            "evaltype": 3,
            "formula": "A and (B or C)",
            "conditions": [
                {
                    "conditiontype": 4,
                    "operator": 5,
                    "value": "2",
                    "formulaid": "A"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084",
                    "formulaid": "B"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10106",
                    "formulaid": "C"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_step_from": 1,
                "esc_step_to": 1,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

创建 agent 自动注册规则

创建一个自动注册动作，当主机名包含 “SRV” 或元数据包含 “AlmaLinux” 时，将该主机添加到主机组 “2” 中。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Register Linux servers",
        "eventsource": "2",
        "filter": {
            "evaltype": "2",
            "conditions": [
                {
                    "conditiontype": "22",
                    "operator": "2",
                    "value": "SRV"
                },
                {
                    "conditiontype": "24",
                    "operator": "2",
                    "value": "AlmaLinux"
                }
            ]
        },
        "operations": [
            {
                "operationtype": "4",
                "opgroup": [
                    {
                        "groupid": "2"
                    }
                ]
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            19
        ]
    },
    "id": 1
}
```

创建带有主机标签的 agent 自动注册规则

创建一个自动注册动作，将主机添加到主机组 “2”，并添加两个主机标签。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Register Linux servers with tags",
        "eventsource": "2",
        "operations": [
            {
                "operationtype": "4",
                "opgroup": [
                    {
                        "groupid": "2"
```

```
                    }
                ]
            },
            {
                "operationtype": "13",
                "optag": [
                    {
                        "tag": "location",
                        "value": "office"
                    },
                    {
                        "tag": "city",
                        "value": "Riga"
                    }
                ]
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            20
        ]
    },
    "id": 1
}
```

另请参见

- 动作过滤器
- 动作操作
- 脚本

源码位置

`CAction::create()` 在 *ui/include/classes/api/services/CAction.php* 文件中。

删除

描述

`object action.delete(array actionIds)`

此方法允许您删除操作。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果某个操作的过滤条件中包含无权访问的 proxy 或 proxy 组，用户将无法删除该操作。有关更多信息，请参阅用户组配置（*Proxy access list*）。

参数

(`array`) 要删除的动作的 ID。

返回值

(`object`) 返回一个对象，该对象在 `actionids` 属性下包含已删除动作的 ID。

示例

删除多个动作

删除两个动作。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.delete",
    "params": [
        "17",
        "18"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17",
            "18"
        ]
    },
    "id": 1
}
```

源码位置

CAction::delete() 在 *ui/include/classes/api/services/CAction.php* 文件中。

更新

描述

`object action.update(object/array actions)`

此方法允许您更新现有的动作。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要更新的动作属性。

必须为每个动作定义 `actionid` 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除标准动作属性外，该方法还接受以下参数。

参数	类型	描述
filter	object	用于替换当前过滤条件的动作过滤器对象。
operations	array	用于替换现有操作的动作操作。
recovery_operations	array	用于替换现有恢复操作的动作恢复操作。
update_operations	array	参数行为： - 如果动作对象的 `eventsource` 设置为 “由触发器创建的事件”、“内部事件” 或 “在服务状态更新时创建的事件”，则支持 用于替换现有更新操作的动作更新操作。 参数行为： - 如果动作对象的 `eventsource` 设置为 “由触发器创建的事件” 或 “在服务状态更新时创建的事件”，则支持

返回值

(object) 返回一个对象，其中包含 actionids 属性下已更新的动作的 ID。

示例

禁用动作

禁用一个动作，即将其状态设置为"1"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.update",
    "params": {
        "actionid": "2",
        "status": "1"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "2"
        ]
    },
    "id": 1
}
```

另请参见

- 动作过滤器
- 动作操作

源码位置

CAction::update() 在 *ui/include/classes/api/services/CAction.php* 文件中。

获取

描述

`integer/array action.get(object parameters)`

该方法允许您根据给定参数检索动作。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
actionids	ID/array	仅返回具有给定 ID 的动作。
groupids	ID/array	仅返回在动作条件中使用给定主机组的动作。
hostids	ID/array	仅返回在动作条件中使用给定主机的动作。
triggerids	ID/array	仅返回在动作条件中使用给定触发器的动作。
mediatypeids	ID/array	仅返回使用给定媒介类型发送消息的动作。
usrgrpids	ID/array	仅返回配置为向给定用户组发送消息的动作。
userids	ID/array	仅返回配置为向给定用户发送消息的动作。
scriptids	ID/array	仅返回配置为运行给定脚本的动作。
selectFilter	query	返回包含动作条件过滤器的`filter` 属性。
selectOperations	query	返回包含动作操作的`operations` 属性。

参数	类型	描述
selectRecoveryOperations	query	返回包含动作恢复操作的recovery_operations 属性。
selectUpdateOperations	query	返回包含动作更新操作的update_operations 属性。
sortfield	string/array	按给定属性对结果进行排序。 可选值：actionid、name、status。
countOutput	boolean	这些参数在参考说明中进行了说明。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

（整数/数组）返回以下之一：

- 一个对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

检索触发器动作

检索所有已配置的触发器动作及其动作条件和操作。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectRecoveryOperations": "extend",
        "selectUpdateOperations": "extend",
        "selectFilter": "extend",
        "filter": {
            "eventsource": 0
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "3",
            "name": " 向 Zabbix 管理员报告问题",
            "eventsource": "0",
            "status": "1",
            "esc_period": "1h",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [],
                "eval_formula": ""
            },
```

```
        "operations": [
            {
                "operationid": "3",
                "actionid": "3",
                "operationtype": "0",
                "esc_period": "0",
                "esc_step_from": "1",
                "esc_step_to": "1",
                "evaltype": "0",
                "opconditions": [],
                "opmessage": [
                    {
                        "default_msg": "1",
                        "subject": "",
                        "message": "",
                        "mediatypeid" => "0"
                    }
                ],
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ]
            }
        ],
        "recovery_operations": [
            {
                "operationid": "7",
                "actionid": "3",
                "operationtype": "11",
                "evaltype": "0",
                "opconditions": [],
                "opmessage": {
                    "default_msg": "0",
                    "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
                    "message": " 触发器：{TRIGGER.NAME}\r\n触发器状态：{TRIGGER.STATUS}\r\n触发器严重性：{TR
                    "mediatypeid": "0"
                }
            }
        ],
        "update_operations": [
            {
                "operationid": "31",
                "operationtype": "12",
                "evaltype": "0",
                "opmessage": {
                    "default_msg": "1",
                    "subject": "",
                    "message": "",
                    "mediatypeid": "0"
                }
            },
            {
                "operationid": "32",
                "operationtype": "0",
                "evaltype": "0",
                "opmessage": {
                    "default_msg": "0",
                    "subject": " 已更新：{TRIGGER.NAME}",
                    "message": "{USER.FULLNAME} 于 {EVENT.UPDATE.DATE} {EVENT.UPDATE.TIME} 更新了问题，并
                    "mediatypeid": "1"
                },
```

```
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ],
                    "opmessage_usr": []
                },
                {
                    "operationid": "33",
                    "operationtype": "1",
                    "evaltype": "0",
                    "opcommand": {
                        "scriptid": "3"
                    },
                    "opcommand_hst": [
                        {
                            "hostid": "10084"
                        }
                    ],
                    "opcommand_grp": []
                }
            ]
        }
    ],
    "id": 1
}
```

检索发现动作

从发现配置中检索动作，同时返回动作操作对象和动作过滤对象。该过滤使用 “和” 评估类型，因此 `formula` 属性为空，而 `eval_formula` 则自动生成。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectFilter": "extend",
        "filter": {
            "eventsource": 1
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "2",
            "name": "Auto discovery. Linux servers.",
            "eventsource": "1",
            "status": "1",
            "esc_period": "0s",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
```

```
                        "conditiontype": "10",
                        "operator": "0",
                        "value": "0",
                        "value2": "",
                        "formulaid": "B"
                    },
                    {
                        "conditiontype": "8",
                        "operator": "0",
                        "value": "9",
                        "value2": "",
                        "formulaid": "C"
                    },
                    {
                        "conditiontype": "12",
                        "operator": "2",
                        "value": "Linux",
                        "value2": "",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A and B and C"
            },
            "operations": [
                {
                    "operationid": "1",
                    "actionid": "2",
                    "operationtype": "6",
                    "esc_period": "0s",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "optemplate": [
                        {
                            "templateid": "10001"
                        }
                    ]
                },
                {
                    "operationid": "2",
                    "actionid": "2",
                    "operationtype": "4",
                    "esc_period": "0s",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "opgroup": [
                        {
                            "groupid": "2"
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


另请参见

- 动作过滤器

- 动作操作

源码位置

CAction::get() 在 *ui/include/classes/api/services/CAction.php* 文件中。

历史记录

此类用于处理历史数据。

对象引用：

- 浮点型历史记录
- 整型历史记录
- 字符串历史记录
- 文本历史记录
- 日志历史记录

可用方法：

- history.clear - 清除历史数据
- history.get - 检索历史数据
- history.push - 将历史数据发送到 Zabbix 服务器

历史数据对象

下列对象与 `history` API 直接相关。

> **Note:**
> 历史数据对象会因为监控项的数据类型而有所不同。它们都是 Zabbix Server 创建的，不能通过 API 进行修改。

浮点数历史记录

浮点数历史记录对象具有以下属性。

属性	类型	描述
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
ns	integer	接收到该值时的纳秒数。
value	float	接收到的值。

整数历史记录

整数历史记录对象具有以下属性。

属性	类型	描述
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
ns	integer	接收到该值时的纳秒数。
value	integer	接收到的值。

字符串历史记录

字符串历史记录对象具有以下属性。

属性	类型	描述
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
ns	integer	接收到该值时的纳秒数。
value	string	接收到的值。

文本历史记录

文本历史记录对象具有以下属性。

属性	类型	描述
id	ID	历史记录条目的 ID。
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
ns	integer	接收到该值时的纳秒数。
value	text	接收到的值。

日志历史记录

日志历史记录对象具有以下属性。

属性	类型	描述
id	ID	历史记录条目的 ID。
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
logeventid	integer	Windows 事件日志条目 ID。
ns	integer	接收到该值时的纳秒数。
severity	integer	Windows 事件日志条目级别。
source	string	Windows 事件日志条目来源。
timestamp	timestamp	Windows 事件日志条目时间。
value	text	接收到的值。

二进制历史记录

二进制历史记录对象具有以下属性。

属性	类型	描述
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
ns	integer	接收到该值时的纳秒数。
value	text	接收到的值。

JSON 历史记录

JSON 历史记录对象具有以下属性。

属性	类型	描述
clock	timestamp	接收到该值的时间。
itemid	ID	相关监控项的 ID。
ns	integer	接收到该值时的纳秒数。
value	text	接收到的值。

推送

描述

`object history.push(object/array itemHistoryData)`

此方法允许发送监控项历史数据给 Zabbix server。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用该方法的权限。有关更多信息，请参阅用户角色。

参数

`(object/array)` 要发送的监控项历史数据。

该方法支持以下参数。

参数	类型	描述
itemid	ID	相关监控项的 ID。 参数行为： - 如果未设置 `host` 和 `key`，则为必填
host	string	主机的技术名称。 参数行为： - 如果未设置 `itemid`，则为必填
key	string	监控项键值。 参数行为： - 如果未设置 `itemid`，则为必填
value	mixed	监控项值。 参数行为： - 必填
clock	timestamp	接收到该值的时间。
ns	integer	接收到该值时的纳秒数。

返回值

(object) 数据发送操作的返回结果。

示例

发送监控项历史数据

发送监控项"10600", "10601" 和"999999" 的历史数据给 Zabbix server。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "history.push",
    "params": [
        {
            "itemid": 10600,
            "value": 0.5,
            "clock": 1690891294,
            "ns": 45440940
        },
        {
            "itemid": 10600,
            "value": 0.6,
            "clock": 1690891295,
            "ns": 312431
        },
        {
            "itemid": 10601,
            "value": "[Tue Aug 01 15:01:35 2023] [error] [client 1.2.3.4] File does not exist: /var/www/ht
        },
        {
            "itemid": 999999,
            "value": 123
        }
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "data": [
            {
                "itemid": "10600"
```

```
            },
            {
                "itemid": "10600"
            },
            {
                "itemid": "10601",
                "error": "Item is disabled."
            },
            {
                "error": "No permissions to referred object or it does not exist."
            }
        ]
    },
    "id": 1
}
```

另见

- 采集器监控项
- HTTP 代理 监控项
- 主机
- 监控项

源码

CHistory::push() in *ui/include/classes/api/services/CHistory.php*.

清理

描述

`object history.clear(array itemids)`

此方法允许您清除监控项历史数据。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 需要清理历史数据的监控项 ID。

返回值

(object) 返回一个包含 itemids 属性（其中包含成功清除历史数据的监控项 ID）的对象。

示例

清理历史数据

请求:

```
{
    "jsonrpc": "2.0",
    "method": "history.clear",
    "params": [
        "10325",
        "13205"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
```

```
            "10325",
            "13205"
        ]
    },
    "id": 1
}
```

源码

CHistory::clear() in *ui/include/classes/api/services/CHistory.php*.

获取

描述

`integer/array history.get(object parameters)`

该方法允许你根据给定参数检索历史数据。

Attention:
如果该数据尚未被 housekeeper 删除，则此方法可能会返回已删除实体的历史数据。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

Parameter	Type	Description
history	integer	要返回的历史数据对象类型。 可能的值： 0 - 数值浮点； 1 - 字符； 2 - 日志； 3 - *(默认)* 数值无符号； 4 - 文本； 5 - 二进制； 6 - JSON。
hostids	ID/array	仅返回来自指定主机的历史数据。
itemids	ID/array	仅返回来自指定监控项的历史数据。
maxValueSize	integer	在 `value` 属性中返回的最大字节数。 可能的取值范围：1-134217728（1 字节到 128MiB）。 如果设置为 `null`，则不应用限制。 默认值：65536（64KiB）。 参数行为： - *supported* 如果 `history` 设置为”binary” 或”JSON”
time_from	timestamp	仅返回在指定时间之后或该时间时已接收的值。
time_till	timestamp	仅返回在指定时间之前或该时间时已接收的值。
sortfield	string/array	按指定属性对结果排序。 可能的值：`itemid`、`clock`、`ns`。 如果历史数据存储在ClickHouse 中，则不支持按 `ns` 排序。

Parameter	Type	Description
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名，值为要搜索的字符串。如果未提供其他选项，则执行 LIKE "%…%" 搜索。 如果 history 设置为 6（JSON），则不支持按 value 搜索。
countOutput	boolean	这些参数在参考说明 中有描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

返回（integer/array）其中之一：

- 一个对象的数组；
- 如果使用了 countOutput 参数，则为检索到的对象的数量。

示例

检索监控项的历史数据

返回从一个 numeric(float) 监控项收到的 10 个最新值。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "history.get",
    "params": {
        "output": "extend",
        "history": 0,
        "itemids": "23296",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 10
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23296",
            "clock": "1351090996",
            "value": "0.085",
            "ns": "563157632"
        },
        {
            "itemid": "23296",
            "clock": "1351090936",
            "value": "0.16",
            "ns": "549216402"
        },
        {
            "itemid": "23296",
```

```
            "clock": "1351090876",
            "value": "0.18",
            "ns": "537418114"
        },
        {
            "itemid": "23296",
            "clock": "1351090816",
            "value": "0.21",
            "ns": "522659528"
        },
        {
            "itemid": "23296",
            "clock": "1351090756",
            "value": "0.215",
            "ns": "507809457"
        },
        {
            "itemid": "23296",
            "clock": "1351090696",
            "value": "0.255",
            "ns": "495509699"
        },
        {
            "itemid": "23296",
            "clock": "1351090636",
            "value": "0.36",
            "ns": "477708209"
        },
        {
            "itemid": "23296",
            "clock": "1351090576",
            "value": "0.375",
            "ns": "463251343"
        },
        {
            "itemid": "23296",
            "clock": "1351090516",
            "value": "0.315",
            "ns": "447947017"
        },
        {
            "itemid": "23296",
            "clock": "1351090456",
            "value": "0.275",
            "ns": "435307141"
        }
    ],
    "id": 1
}
```


源码

CHistory::get() in *ui/include/classes/api/services/CHistory.php*.

发现检查

此类用于处理发现检查。

对象引用：

- 发现检查

可用方法：

- dcheck.get - 检索发现检查

发现检查对象

以下对象与 dcheck API 直接相关。

发现检查

发现检查对象定义了由网络发现规则执行的一项特定检查。它具有以下属性。

属性	类型	描述
dcheckid	ID	发现检查的 ID。
druleid	ID	该检查所属的发现规则的 ID。
key_	string	监控项键值（如果 type 设置为 “Zabbix agent”）或 SNMP OID（如果 type 设置为 “SNMPv1 agent”、“SNMPv2 agent” 或 “SNMPv3 agent”）。 属性行为： - 如果 type 设置为 “Zabbix agent”、“SNMPv1 agent”、“SNMPv2 agent” 或 “SNMPv3 agent”，则为 必需
ports	string	要检查的一个或多个端口范围，以逗号分隔。 默认值：0。
snmp_community	string	属性行为： - 如果 type 设置为 “SSH”(0)、“LDAP”(1)、“SMTP”(2)、“FTP”(3)、“HTTP”(4)、“POP”(5)、“NNTP”(6)、“IMAP”(7)、“TCP”(8)、“Zabbix agent”(9)、“SNMPv1 agent”(10)、“SNMPv2 agent”(11)、“SNMPv3 agent”(13)、“HTTPS”(14) 或 “Telnet”(15)，则 支持 SNMP community。
snmpv3_authpassphrase	string	属性行为： - 如果 type 设置为 “SNMPv1 agent” 或 “SNMPv2 agent”，则为 必需 认证口令。
snmpv3_authprotocol	integer	属性行为： - 如果 type 设置为 “SNMPv3 agent” 且 snmpv3_securitylevel 设置为 “authNoPriv” 或 “authPriv”，则 支持 认证协议。 可选值： 0 - (默认) MD5； 1 - SHA1； 2 - SHA224； 3 - SHA256； 4 - SHA384； 5 - SHA512。
snmpv3_contextname	string	属性行为： - 如果 type 设置为 “SNMPv3 agent” 且 snmpv3_securitylevel 设置为 “authNoPriv” 或 “authPriv”，则 支持 SNMPv3 上下文名称。
snmpv3_privpassphrase	string	属性行为： - 如果 type 设置为 “SNMPv3 agent”，则 支持 隐私口令。 属性行为： - 如果 type 设置为 “SNMPv3 agent” 且 snmpv3_securitylevel 设置为 “authPriv”，则 支持

属性	类型	描述
snmpv3_privprotocol	integer	隐私协议。 可选值： 0 - *(默认)* DES； 1 - AES128； 2 - AES192； 3 - AES256； 4 - AES192C； 5 - AES256C。 属性行为： - 如果 `type` 设置为 “SNMPv3 agent” 且 `snmpv3_securitylevel` 设置为 “authPriv”，则 支持
snmpv3_securitylevel	string	安全级别。 可选值： 0 - noAuthNoPriv； 1 - authNoPriv； 2 - authPriv。 属性行为： - 如果 `type` 设置为 “SNMPv3 agent”，则 支持
snmpv3_securityname	string	安全名称。 属性行为： - 如果 `type` 设置为 “SNMPv3 agent”，则 支持
type	integer	检查类型。 可选值： 0 - SSH； 1 - LDAP； 2 - SMTP； 3 - FTP； 4 - HTTP； 5 - POP； 6 - NNTP； 7 - IMAP； 8 - TCP； 9 - Zabbix agent； 10 - SNMPv1 agent； 11 - SNMPv2 agent； 12 - ICMP ping； 13 - SNMPv3 agent； 14 - HTTPS； 15 - Telnet。 属性行为： - 必需
uniq	integer	是否将此检查用作设备唯一性判定条件。一个发现规则只能配置一个唯一检查。 可选值： 0 - *(默认)* 不将此检查用作唯一性判定条件； 1 - 将此检查用作唯一性判定条件。 属性行为： - 如果 `type` 设置为 “Zabbix agent”、“SNMPv1 agent”、“SNMPv2 agent” 或 “SNMPv3 agent”，则 支持

属性	类型	描述
host_source	integer	主机名称的来源。 可选值： 1 - *(默认)* DNS； 2 - IP； 3 - 此检查的发现值。
name_source	integer	可见名称的来源。 可选值： 0 - *(默认)* 未指定； 1 - DNS； 2 - IP； 3 - 此检查的发现值。
allow_redirect	integer	允许被 ICMP ping 的目标从不同的 IP 地址响应。 可选值： 0 - *(默认)* 将重定向响应视为目标主机宕机（失败）； 1 - 将重定向响应视为目标主机在线（成功）。 属性行为： - 如果 `type` 设置为 “ICMP ping”，则 支持

获取

描述

`integer/array dcheck.get(object parameters)`

该方法允许你根据给定参数检索发现检查。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
dcheckids	ID/array	仅返回具有给定 ID 的发现检查。
druleids	ID/array	仅返回属于给定发现规则的发现检查。
dserviceids	ID/array	仅返回已检测到给定已发现服务的发现检查。
selectDRules	query	返回与发现检查相关的发现规则。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`dcheckid`、`druleid`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中一种结果：

- 一个对象数组；
- 如果使用了参数 `countOutput`，则返回检索到的对象的数量。

示例

检索发现规则的发现检查

检索被发现规则 “6” 使用的所有发现检查。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dcheck.get",
    "params": {
        "output": "extend",
        "dcheckids": "6"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dcheckid": "6",
            "druleid": "4",
            "type": "3",
            "key_": "",
            "snmp_community": "",
            "ports": "21",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "uniq": "0",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "host_source": "1",
            "name_source": "0"
        }
    ],
    "id": 1
}
```

来源

ui/include/classes/api/services/CDCheck.php 中的 CDCheck::get()。

发现规则

此类用于处理网络发现规则。

> **Note:**
> 此 API 用于处理网络发现规则。有关低级别发现规则，请参见LLD 规则 API。

对象引用：

- 发现规则

可用方法：

- drule.create - 创建新的网络发现规则
- drule.delete - 删除网络发现规则

- drule.get - 检索网络发现规则
- drule.update - 更新网络发现规则

发现规则对象

以下对象与 drule API 直接相关。

发现规则

发现规则对象定义了一个网络发现规则。它具有以下属性。

Property	Type	Description
druleid	ID	发现规则的 ID。 属性行为： - 只读 - 更新操作时必需
iprange	string	要检查的一个或多个 IP 范围，使用逗号分隔。 有关支持的 IP 范围格式的更多信息，请参阅网络发现配置部分。
name	string	属性行为： - 创建操作时必需 发现规则的名称。
delay	string	属性行为： - 创建操作时必需 发现规则的执行间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d）或用户宏。 默认值：1h。
proxyid	ID	用于发现的 proxy 的 ID。
status	integer	发现规则是否启用。 可能的值： 0 - （默认）启用； 1 - 禁用。
concurrency_max	integer	每个发现规则的最大并发检查数。 可能的值： 0 - （默认）不限制检查数； 1 - 一次检查； 2-999 - 自定义检查数。
error	string	执行发现规则时出现任何问题时的错误文本。 属性行为： - 只读

创建

描述

`object drule.create(object/array discoveryRules)`

此方法允许您创建新的发现规则。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的发现规则。

除了标准发现规则属性之外，该方法还接受以下参数。

参数	类型	描述
dchecks	array	要为发现规则创建的发现检查。 参数行为： - 必需

返回值

(object) 返回一个对象，该对象包含 druleids 属性下创建的发现规则的 ID。返回 ID 的顺序与传递的发现规则的顺序相匹配。

示例

创建发现规则

创建一个发现规则来查找在内部网络中运行 Zabbix agent 的机器。该规则必须在端口 10050 上使用单个 Zabbix agent 检查。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "drule.create",
    "params": {
        "name": "Zabbix agent discovery",
        "iprange": "192.168.1.1-255",
        "dchecks": [
            {
                "type": "9",
                "key_": "system.uname",
                "ports": "10050",
                "uniq": "0"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

另请参阅

- 发现检查

来源

ui/include/classes/api/services/CDRule.php 中的 CDRule::create()。

删除

描述

`object drule.delete(array discoveryRuleIds)`

此方法允许你删除发现规则。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的发现规则的 ID。

返回值

(object) 返回一个对象，该对象包含 druleids 属性下已删除的发现规则的 ID。

示例

删除多个发现规则

删除两个发现规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "drule.delete",
    "params": [
        "4",
        "6"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "4",
            "6"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CDRule.php 中的 CDRule::delete()。

更新

描述

`object drule.update(object/array discoveryRules)`

此方法允许你更新现有的发现规则。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的发现规则属性。

必须为每个发现规则定义 druleid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准发现规则属性之外，该方法还接受以下参数。

参数	类型	描述
dchecks	array	用于替换现有检查的发现检查。

返回值

(object) 返回一个对象，该对象包含 druleids 属性下更新的发现规则的 ID。

示例

更改发现规则的 IP 范围

将发现规则的 IP 范围更改为 “192.168.2.1-255”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "drule.update",
    "params": {
        "druleid": "6",
        "iprange": "192.168.2.1-255"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

另请参见

- 发现检查

来源

ui/include/classes/api/services/CDRule.php 中的 CDRule::update()。

获取

描述

`integer/array drule.get(object parameters)`

该方法允许你根据给定参数检索发现规则。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
dhostids	ID/array	仅返回创建了给定已发现主机的发现规则。
druleids	ID/array	仅返回具有给定 ID 的发现规则。
dserviceids	ID/array	仅返回创建了给定已发现服务的发现规则。

参数	类型	描述
selectDChecks	query	返回dchecks 属性，其中包含发现规则使用的发现检查。 支持 count。
selectDHosts	query	返回dhosts 属性，其中包含由发现规则创建的已发现主机。 支持 count。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectDChecks - 结果将按 dcheckid 排序； selectDHosts - 结果将按 dhostsid 排序。
sortfield	string/array	按给定属性对结果进行排序。 可选值：druleid、name。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中一种结果：

- 一个对象数组；
- 如果使用了参数 countOutput，则返回检索到的对象的数量。

示例

获取所有发现规则

获取所有配置的发现规则及其使用的发现检查。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "drule.get",
    "params": {
        "output": "extend",
        "selectDChecks": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "druleid": "2",
            "proxyid": "0",
            "name": "Local network",
            "iprange": "192.168.3.1-255",
            "delay": "5s",
            "status": "0",
            "concurrency_max": "0",
```

```
            "error": "",
            "dchecks": [
                {
                    "dcheckid": "7",
                    "druleid": "2",
                    "type": "3",
                    "key_": "",
                    "snmp_community": "",
                    "ports": "21",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0",
                    "snmpv3_contextname": "",
                    "host_source": "1",
                    "name_source": "0",
                    "allow_redirect": "0"
                },
                {
                    "dcheckid": "8",
                    "druleid": "2",
                    "type": "4",
                    "key_": "",
                    "snmp_community": "",
                    "ports": "80",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0",
                    "snmpv3_contextname": "",
                    "host_source": "1",
                    "name_source": "0",
                    "allow_redirect": "0"
                }
            ]
        },
        {
            "druleid": "6",
            "proxyid": "0",
            "name": "Zabbix agent discovery",
            "iprange": "192.168.1.1-255",
            "delay": "1h",
            "status": "0",
            "concurrency_max": "10",
            "error": "",
            "dchecks": [
                {
                    "dcheckid": "10",
                    "druleid": "6",
                    "type": "9",
                    "key_": "system.uname",
                    "snmp_community": "",
                    "ports": "10050",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
```

```
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0",
                    "snmpv3_contextname": "",
                    "host_source": "2",
                    "name_source": "3",
                    "allow_redirect": "0"
                }
            ]
        }
    ],
    "id": 1
}
```


另请参见

- 已发现的主机
- 发现检查

来源

ui/include/classes/api/services/CDRule.php 中的 CDRule::get()。

告警

这个类是为处理告警而设计的。

对象引用：

- 告警

可用方法：

- alert.get - 检索告警

告警对象

以下对象与 `alert` API 直接相关。

告警

Note:
告警由 Zabbix 服务器创建，且无法通过 API 修改。

告警对象包含有关某些动作操作是否已成功执行的信息。它具有以下属性。

属性	类型	描述
alertid	ID	告警的 ID。
actionid	ID	生成该告警的动作 ID。
alerttype	integer	告警类型。 可能的值： 0 - 消息； 1 - 远程命令。
clock	timestamp	生成告警的时间。
error	string	发送消息或运行命令时如果出现问题的错误文本。
esc_step	integer	生成该告警时的动作升级步骤。
eventid	ID	触发该动作的事件 ID。
mediatypeid	ID	用于发送消息的媒介类型 ID。
message	text	消息文本。 属性行为： - 如果 `alerttype` 设置为 “message”，则为 *supported*
retries	integer	Zabbix 尝试发送该消息的次数。

属性	类型	描述
sendto	string	收件人的地址、用户名或其他标识符。
status	integer	属性行为： - 如果 `alerttype` 设置为 “message”，则为 *supported* 表示动作操作是否已成功执行的状态。 当 `alerttype` 设置为 “message” 时，可能的值： 0 - 消息未发送； 1 - 消息已发送； 2 - 重试若干次后失败； 3 - 新告警尚未由告警管理器处理。 当 `alerttype` 设置为 “remote command” 时，可能的值： 0 - 命令未运行； 1 - 命令已运行； 2 - 已尝试在 Zabbix agent 上运行该命令，但其不可用。
subject	string	消息主题。 属性行为： - 如果 `alerttype` 设置为 “message”，则为 *supported*
userid	ID	接收该消息的用户 ID。
p_eventid	ID	生成该告警的问题事件 ID。
acknowledgeid	ID	生成该告警的确认 ID。

获取

描述

`integer/array alert.get(object parameters)`

该方法允许您根据给定参数检索告警。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
alertids	ID/array	仅返回具有给定 ID 的告警。
actionids	ID/array	仅返回由给定动作生成的告警。
eventids	ID/array	仅返回由给定事件生成的告警。
groupids	ID/array	仅返回由给定主机组中的对象生成的告警。
hostids	ID/array	仅返回由给定主机中的对象生成的告警。
mediatypeids	ID/array	仅返回使用给定媒介类型的消息告警。
objectids	ID/array	仅返回由给定对象生成的告警。
userids	ID/array	仅返回发送给给定用户的消息告警。
eventobject	integer	仅返回由与给定类型对象相关的事件生成的告警。 支持的对象类型列表请参见 event `object`。 默认值：0 - 触发器。
eventsource	integer	仅返回由给定类型事件生成的告警。 支持的事件类型列表请参见 event `source`。 默认值：0 - 触发器事件。
time_from	timestamp	仅返回在给定时间之后生成的告警。

参数	类型	说明
time_till	timestamp	仅返回在给定时间之前生成的告警。
selectHosts	query	返回hosts 属性，其中包含触发该动作操作的主机数据。
selectMediatypes	query	返回mediatypes 属性，其中包含用于该消息告警的媒介类型数组。 有关基于用户类型的限制，请参见mediatype.get。
selectUsers	query	返回users 属性，其中包含该消息所发送给的用户数组。 有关基于用户类型的限制，请参见user.get。
sortfield	string/array	按给定属性对结果进行排序。 可选值：alertid、clock、eventid、mediatypeid、sendto、status。
countOutput	boolean	这些参数在参考注释中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下两者之一：

- 一个对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象的数量。

示例

按动作 ID 检索告警

检索由动作 “3” 生成的所有告警。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "alert.get",
    "params": {
        "output": "extend",
        "actionids": "3"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "alertid": "1",
            "actionid": "3",
            "eventid": "21243",
            "userid": "1",
            "clock": "1362128008",
            "mediatypeid": "1",
            "sendto": "support@company.com",
            "subject": "PROBLEM: Linux 服务器上的 Zabbix agent 已有 5 分钟无法访问：",
            "message": " 触发器：Linux 服务器上的 Zabbix agent 已有 5 分钟无法访问：\n触发器状态：PROBLEM\n触发器严重
```

```
            "status": "0",
            "retries": "3",
            "error": "",
            "esc_step": "1",
            "alerttype": "0",
            "p_eventid": "0",
            "acknowledgeid": "0"
        }
    ],
    "id": 1
}
```

另请参阅

- 主机
- 介质类型
- 用户

源码位置

CAlert::get() 在 *ui/include/classes/api/services/CAlert.php* 文件中。

图像

此类旨在处理图像。

对象引用：

- 图像

可用方法：

- image.create - 创建新图像
- image.delete - 删除图像
- image.get - 获取图像
- image.update -更新图像

图像对象

以下对象与 image API 直接相关。

图像

图像对象具有以下属性。

属性	类型	描述
imageid	ID	图像的 ID。 属性行为： - 只读 - 更新操作时 必填
name	string	图像名称。 属性行为： - 创建操作时 必填
imagetype	integer	图像类型。 可能的值： 1 - （默认）图标； 2 - 背景图像。 属性行为： - 常量 - 创建操作时 必填

属性	类型	描述
image	string	Base64 编码的图像。 编码后图像的最大大小为 1 MB。可通过修改 ZBX_MAX_IMAGE_SIZE 常量值来调整最大大小。 支持的图像格式：PNG、JPEG、GIF 和 WebP。 属性行为： - 创建操作时 必填

创建

描述

`object image.create(object/array images)`

此方法允许您创建新图像。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 创建的目标图像。

该方法接受具有标准图像属性的图像。

返回值

(object) 返回一个对象，该对象包含 imageids 属性下创建的图像的 ID。返回 ID 的顺序与传递图像的顺序一致。

示例

创建图标图像

创建一个云图标。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "image.create",
    "params": {
        "imagetype": 1,
        "name": "Cloud_(24)",
        "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAAApgB
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "188"
        ]
    },
    "id": 1
}
```

创建背景图片

创建一个小型背景图片，用作图案。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "image.create",
    "params": {
        "imagetype": 2,
        "name": "background_image",
        "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAIAAABvFaqvAAABhWlDQ1BJQ0MgcHJvZmlsZQAAKJF9kb1Lw0AYxp+m
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "189"
        ]
    },
    "id": 1
}
```

来源

CImage::create() in *ui/include/classes/api/services/CImage.php*.

删除

描述

`object image.delete(array imageIds)`

此方法允许您删除图像。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色 。

参数

(array) 要删除图像的 ID。

返回值

(object) 返回一个对象，该对象包含 imageids 属性下已删除图像的 ID。

示例

批量删除图像

删除两个图像。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "image.delete",
    "params": [
        "188",
        "192"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "imageids": [
            "188",
            "192"
        ]
    },
    "id": 1
}
```

来源

CImage::delete() in *ui/include/classes/api/services/CImage.php*.

更新

描述

`object image.update(object/array images)`

此方法允许您更新现有图像。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 更新的目标图像属性。

每个图象都必须定义 imageid 属性，其余属性为可选项。只有传递的属性将被更新，其他属性保持不变。

此方法接受具有标准图像属性的图像。

返回值

(object) 返回一个对象，该对象包含 imageids 属性下更新的图像的 ID。

示例

重命名图像

将图像重命名为 “Cloud icon” 。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "image.update",
    "params": {
        "imageid": "2",
        "name": "Cloud icon"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "2"
        ]
    },
    "id": 1
}
```

来源

CImage::update() in *ui/include/classes/api/services/CImage.php*.

获取

描述

`integer/array image.get(object parameters)`

该方法允许您根据给定参数检索图像。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
imageids	ID/array	仅返回具有给定 ID 的图像。
sysmapids	ID/array	返回在给定拓扑图中使用的图像。
select_image	flag	返回一个 image 属性，其中包含经过 Base64 编码的图像。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：imageid、name。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中之一：

- 一组对象；
- 如果使用了 countOutput 参数，则检索对象的计数。

示例

检索图像

检索 ID 为 “2” 的图像的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "image.get",
    "params": {
        "output": "extend",
        "select_image": true,
        "imageids": "2"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
        {
            "imageid": "2",
            "imagetype": "1",
            "name": "Cloud_(24)",
            "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAA
        }
    ],
    "id": 1
}
```


来源

CImage::get() in *ui/include/classes/api/services/CImage.php*.

图形原型

此类用于处理图形原型。

对象引用：

- 图形原型

可用方法：

- graphprototype.create - 创建新的图形原型
- graphprototype.delete - 删除图形原型
- graphprototype.get - 检索图形原型
- graphprototype.update - 更新图形原型

图形原型对象

以下对象与 图形原型 API 直接相关。

图形原型

图形原型对象具有以下属性。

属性	类型	描述
graphid	ID	图形原型的 ID。 属性行为： - 只读 - 更新操作时为 必需
height	integer	图形原型的高度（以像素为单位）。 属性行为： - 创建操作时为 必需
name	string	图形原型的名称。 属性行为： - 创建操作时为 必需
width	integer	图形原型的宽度（以像素为单位）。 属性行为： - 创建操作时为 必需
flags	integer	图形原型的来源。 可能的值： 2 - 图形原型； 6 - 发现的图形原型 属性行为： - 只读

属性	类型	描述
graphtype	integer	图形原型的布局类型。 可能的值： 0 - （默认）普通； 1 - 堆叠； 2 - 饼图； 3 - 分离。
percent_left	float	左百分位。 默认值：0。
percent_right	float	右百分位。 默认值：0。
show_3d	integer	是否以 3D 方式显示发现的饼图和分离图。 可能的值： 0 - （默认）以 2D 显示； 1 - 以 3D 显示。
show_legend	integer	是否在发现的图形上显示图例。 可能的值： 0 - 隐藏； 1 - （默认）显示。
show_work_period	integer	是否在发现的图形上显示工作时间。 可能的值： 0 - 隐藏； 1 - （默认）显示。
templateid	ID	父模板图形原型的 ID。 属性行为： - 只读
yaxismax	float	Y 轴的固定最大值。
yaxismin	float	Y 轴的固定最小值。
ymax_itemid	ID	用作 Y 轴最大值的监控项 ID。 如果用户无权访问指定的监控项，则图形将按 `ymax_type` 设置为“calculated” 时的方式进行渲染。
ymax_type	integer	Y 轴最大值的计算方法。 可能的值： 0 - （默认） calculated； 1 - fixed； 2 - item。
ymin_itemid	ID	用作 Y 轴最小值的监控项 ID。 如果用户无权访问指定的监控项，则图形将按 `ymin_type` 设置为“calculated” 时的方式进行渲染。
ymin_type	integer	Y 轴最小值的计算方法。 可能的值： 0 - （默认） calculated； 1 - fixed； 2 - item。
discover	integer	图形原型发现状态。 可能的值： 0 - （默认）将发现新图形； 1 - 将不发现新图形，现有图形将被标记为丢失。

属性	类型	描述
uuid	string	通用唯一标识符，用于将导入的图形原型链接到已存在的图形原型。如果未提供，则自动生成。 属性行为： - 如果图形原型属于模板，则 支持

创建

描述

`object graphprototype.create(object/array graphPrototypes)`

此方法允许您创建新的图形原型。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的图形原型。

除了标准图形原型属性之外，该方法还接受以下参数。

参数	Type	说明
gitems	array	要为图形原型创建的图形监控项。图形监控项既可以引用监控项，也可以引用监控项原型，但必须至少包含一个监控项原型。 参数行为: - 必需

返回值

(object) 返回一个对象，包含在 graphids 属性下创建的图形原型的 ID。返回的 ID 的顺序与传递的图形原型的顺序相匹配。

示例

创建图形原型

创建具有两个监控项的图形原型。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.create",
    "params": {
        "name": "Disk space usage {#FSNAME}",
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

另请参阅

- 图形监控项

源码

CGraphPrototype::create() in *ui/include/classes/api/services/CGraphPrototype.php*.

删除

描述

`object graphprototype.delete(array graphPrototypeIds)`

此方法允许您删除图形原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(array) 要删除的图形原型的 ID。

返回值

(object) 返回一个对象，该对象包含 graphids' 属性下已删除的图形原型的 ID。

示例

删除多个图形原型

删除两个图形原型。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.delete",
    "params": [
        "652",
        "653"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

源码

CGraphPrototype::delete() in *ui/include/classes/api/services/CGraphPrototype.php*.

更新

描述

`object graphprototype.update(object/array graphPrototypes)`

此方法允许您更新现有的图形原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`object/array`) 要更新的图形原型属性。

必须为每个图形原型定义 `graphid` 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准图形原型属性之外，该方法还接受以下参数。

参数	Type	说明
gitems	array	用于替换现有图形项的图形项。如果图形项定义了 `gitemid` 属性，则会更新该图形项；否则将创建一个新的图形项。

返回值

(`object`) 返回一个包含 'graphids' 属性下更新的图形原型的 ID 的对象。

示例

改变图形原型的大小

将图形原型的大小改为 1100 * 400 像素。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.update",
    "params": {
        "graphid": "439",
        "width": 1100,
        "height": 400
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

源码

CGraphPrototype::update() in *ui/include/classes/api/services/CGraphPrototype.php*.

获取

描述

`integer/array graphprototype.get(object parameters)`

该方法允许你根据给定参数检索图形原型。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
discoveryids	ID/array	仅返回属于给定发现规则的图形原型。
graphids	ID/array	仅返回具有给定 ID 的图形原型。
groupids	ID/array	仅返回属于给定主机组或模板组中的主机或模板的图形原型。
hostids	ID/array	仅返回属于给定主机的图形原型。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的图形原型。
itemids	ID/array	仅返回包含给定监控项原型的图形原型。
templated	boolean	如果设置为 `true`，则仅返回属于模板的图形原型。
templateids	ID/array	仅返回属于给定模板的图形原型。
selectDiscoveryData	query	返回一个 `discoveryData` 属性，其中包含图形原型发现对象数据。图形原型发现对象将已发现的图形原型链接到发现它的图形原型。 它具有以下属性： `parent_graphid` - `(ID)` 创建该图形所依据的图形原型的 ID； `status` - `(int)` 图形发现状态： 0 - *(默认)* 图形原型已发现， 1 - 图形原型不再被发现； `ts_delete` - `(timestamp)` 不再被发现的图形原型将被删除的时间。
selectDiscoveryRule	query	返回一个`discoveryRule` 属性，其中包含该图形原型所属的 LLD 规则。
selectDiscoveryRulePrototype	query	返回一个`discoveryRulePrototype` 属性，其中包含该图形原型所属的父 LLD 规则原型。
selectGraphItems	query	返回一个`gitems` 属性，其中包含图形原型中使用的图形项。
selectHostGroups	query	返回一个`hostgroups` 属性，其中包含该图形原型所属的主机组。
selectHosts	query	返回一个`hosts` 属性，其中包含该图形原型所属的主机。
selectItems	query	返回一个 `items` 属性，其中包含图形原型中使用的监控项和监控项原型。
selectTemplateGroups	query	返回一个`templategroups` 属性，其中包含该图形原型所属的模板组。
selectTemplates	query	返回一个`templates` 属性，其中包含该图形原型所属的模板。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 `text` 数据类型 的属性。 支持附加属性： `host` - 该图形原型所属主机的技术名称； `hostid` - 该图形原型所属主机的 ID。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`graphid`、`name`、`graphtype`、`discovered`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	

参数	类型	描述
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

返回（integer/array）其中之一：

- 一个对象的数组；
- 如果使用了 countOutput 参数，则为检索到的对象的数量。

示例

从 LLD 规则中检索图形原型

检索某个 LLD 规则中的所有图形原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "1017",
            "name": "Disk space usage {#FSNAME}",
            "width": "600",
            "height": "340",
            "yaxismin": "0.0000",
            "yaxismax": "0.0000",
            "templateid": "442",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "2"
            "discover": "0"
        }
    ],
    "id": 1
}
```

另请参见

- 发现规则
- 图形监控项
- 监控项
- 主机

- 主机组
- 模板
- 模板组

源码

CGraphPrototype::get() in *ui/include/classes/api/services/CGraphPrototype.php*.

图形监控项

此类用于处理图形监控项。

对象引用：

- 图形监控项

可用方法：

- graphitem.get - 检索图形监控项

图形监控项对象

以下对象与 图形监控项 API 直接相关。

图形监控项

Note:
图形监控项只能通过 `graph` API 进行修改。

图形监控项对象具有以下属性。

属性	类型	说明
gitemid	ID	图形监控项的 ID。 属性行为： - 只读
color	string	图形监控项的绘制颜色，以十六进制颜色代码表示。 属性行为： - 创建操作时必填
itemid	ID	监控项的 ID。 属性行为： - 创建操作时必填
calc_fnc	integer	将显示的监控项值。 可选值： 1 - 最小值； 2 - *(默认)* 平均值； 4 - 最大值； 7 - 所有值； 9 - 最新值，仅用于饼图和分离饼图。
drawtype	integer	图形监控项的绘制样式。 可选值： 0 - *(默认)* 线条； 1 - 填充区域； 2 - 粗线； 3 - 点； 4 - 虚线； 5 - 渐变线。
graphid	ID	该图形监控项所属图形的 ID。

属性	类型	说明
sortorder	integer	监控项在图形中的位置。 默认值：从 “0” 开始，每增加一项递增 1。
type	integer	图形监控项的类型。 可选值： 0 - *(默认)* 简单； 2 - 图形总和，仅用于饼图和分离饼图。
yaxisside	integer	绘制图形监控项 Y 轴刻度的一侧。 可选值： 0 - *(默认)* 左侧； 1 - 右侧。

获取

描述

`integer/array graphitem.get(object parameters)`

该方法允许您根据给定参数检索图形监控项。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
graphids	ID/array	仅返回属于给定图形的图形监控项。
itemids	ID/array	仅返回具有给定监控项 ID 的图形监控项。
type	integer	仅返回具有给定类型的图形监控项。 支持的图形监控项类型列表请参见图形监控项对象页面。
selectGraphs	query	返回一个`graphs` 属性，其中包含该监控项所属图形的数组。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`gitemid`。
countOutput	boolean	这些参数在参考注释中进行了说明。
editable	boolean	
limit	integer	
output	query	
preservekeys	boolean	
sortorder	string/array	

返回值

返回值是 `(integer/array)` 其中之一：

- 一个对象的数组；
- 如果使用了 `countOutput` 参数，则为检索到的对象的数量。

示例

从图形中检索图形监控项

检索图形中使用的所有图形监控项，以及相关的监控项和主机的额外信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "graphitem.get",
```

```
    "params": {
        "output": "extend",
        "graphids": "387"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "gitemid": "1242",
            "graphid": "387",
            "itemid": "22665",
            "drawtype": "1",
            "sortorder": "1",
            "color": "FF5555",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
            "color": "55FF55",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        },
        {
            "gitemid": "1244",
            "graphid": "387",
            "itemid": "22671",
            "drawtype": "1",
            "sortorder": "3",
            "color": "009999",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        }
    ],
    "id": 1
}
```

另见

- 图形

源码

CGraphItem::get() in *ui/include/classes/api/services/CGraphItem.php*.

图标映射

此类用于处理图标映射。

对象引用：

- 图标映射
- 图标映射关系

可用方法：

- iconmap.create - 创建新的图标映射
- iconmap.delete - 删除图标映射
- iconmap.get - 获取图标映射
- iconmap.update - 更新图标映射

图标映射对象

以下对象与 `iconmap` API 直接相关。

图标映射

图标映射对象具有以下属性。

属性	类型	描述
iconmapid	ID	图标映射的 ID。 属性行为： - 只读 - 更新操作时 必需
default_iconid	ID	默认图标的 ID。 属性行为： - 创建操作时 必需
name	string	图标映射的名称。 属性行为： - 创建操作时 必需

图标映射

图标映射对象定义了一个特定图标，用于具有某个清单字段值的主机。它具有以下属性。

属性	类型	描述
iconid	ID	图标映射使用的图标 ID。 属性行为： - 必需
expression	string	用于与清单字段进行匹配的表达式。 属性行为： - 必需
inventory_link	integer	主机清单字段的 ID。 支持的清单字段列表请参见主机清单对象。 属性行为： - 必需
sortorder	integer	图标映射在图标地图中的位置。 属性行为： - 只读

创建

描述

`object iconmap.create(object/array iconMaps)`

此方法允许您创建新的图标映射。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的图标映射。

除了标准图标映射属性外，该方法还接受以下参数。

参数	类型	描述
mappings	array	要为该图标映射创建的图标映射关系。 参数行为: - 必需

返回值

(object) 返回一个如下对象：在 iconmapids 属性下包含所创建图标映射的 ID。返回的多个 ID 的顺序与传参里多个图标映射的顺序相匹配。

示例

创建一个图标映射

创建一个图标映射以展现不同类型的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.create",
    "params": {
        "name": "Type icons",
        "default_iconid": "2",
        "mappings": [
            {
                "inventory_link": 1,
                "expression": "server",
                "iconid": "3"
            },
            {
                "inventory_link": 1,
                "expression": "switch",
                "iconid": "4"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2"
        ]
    },
    "id": 1
}
```

另请参见

- 图标映射

来源

CIconMap::create() in *ui/include/classes/api/services/CIconMap.php*.

删除

描述

`object iconmap.delete(array iconMapIds)`

此方法允许您删除图标映射。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。在用户角色设置中可以撤销调用此方法的权限。有关更多信息，请参见用户角色。

参数

(array) 要删除的多个图标映射的 ID。

返回值

(object) 返回一个如下对象：在 iconmapids 属性下包含所删除的多个图标映射的 ID。

示例

删除多个图标映射

删除两个图标映射。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.delete",
    "params": [
        "2",
        "5"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2",
            "5"
        ]
    },
    "id": 1
}
```

来源

CIconMap::delete() in *ui/include/classes/api/services/CIconMap.php*.

更新

描述

`object iconmap.update(object/array iconMaps)`

此方法允许你更新现有的图标映射。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的图标映射属性。

必须为每个图标映射定义 iconmapid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准图标映射属性之外，该方法还接受以下参数。

参数	类型	说明
mappings	array	用于替换现有图标映射的图标映射关系。

返回值

(object) 返回一个如下对象：在 iconmapids 属性下包含所更新的多个图标映射的 ID。

示例

重命名一个图标映射

重命名一个图标映射为 “OS icons”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.update",
    "params": {
        "iconmapid": "1",
        "name": "OS icons"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "1"
        ]
    },
    "id": 1
}
```

另请参见

- 图标映射

来源

CIconMap::update() in *ui/include/classes/api/services/CIconMap.php*。

获取

描述

integer/array iconmap.get(object parameters)

该方法允许您根据给定参数检索图标映射。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
iconmapids	ID/array	仅返回具有给定 ID 的图标映射。
sysmapids	ID/array	仅返回在给定网络地图中使用的图标映射。
selectMappings	query	返回一个mappings 属性，其中包含所使用的图标映射。

参数	类型	说明
sortfield	string/array	按给定属性对结果进行排序。 可选值：iconmapid、name。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回任一：

- 对象数组；
- 如果已使用 countOutput 参数，则检索对象的计数。

例如

检索图标映射

检索所有关于图标映射"3" 的数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.get",
    "params": {
        "iconmapids": "3",
        "output": "extend",
        "selectMappings": "extend"
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mappings": [
                {
                    "iconmappingid": "3",
                    "iconmapid": "3",
                    "iconid": "6",
                    "inventory_link": "1",
                    "expression": "server",
                    "sortorder": "0"
                },
                {
                    "iconmappingid": "4",
                    "iconmapid": "3",
                    "iconid": "10",
                    "inventory_link": "1",
                    "expression": "switch",
                    "sortorder": "1"
                }
            ],
```

```
            "iconmapid": "3",
            "name": "Host type icons",
            "default_iconid": "2"
        }
    ],
    "id": 1
}
```

另请参见

- 图标映射

来源

CIconMap::get() in *ui/include/classes/api/services/CIconMap.php*.

图表

此类设计用于图表操作。

对象参考：

- 图表

可用方法：

- graph.create - 创建图表
- graph.delete - 删除图表
- graph.get - 获取图表
- graph.update - 更新图表

图表对象

以下对象与 graph API 直接相关。

图形

图形对象具有以下属性。

Property	Type	Description
graphid	ID	图形的 ID。 属性行为： - 只读 - 更新操作时 必需
height	integer	图形的高度（像素）。 属性行为： - 创建操作时 必需
name	string	图形名称。 属性行为： - 创建操作时 必需
width	integer	图形的宽度（像素）。 属性行为： - 创建操作时 必需
flags	integer	图形的来源。 可能的值： 0 - *(默认)* 普通图形； 4 - 从原型转换而来的图形。 属性行为： - 只读

Property	Type	Description
graphtype	integer	图形的布局类型。 可能的值： 0 - *(默认)* 普通； 1 - 堆叠； 2 - 饼图； 3 - 分离。
percent_left	float	左百分位。 默认值：0。
percent_right	float	右百分位。 默认值：0。
show_3d	integer	是否以 3D 显示饼图和分离图。 可能的值： 0 - *(默认)* 以 2D 显示； 1 - 以 3D 显示。
show_legend	integer	是否在图形上显示图例。 可能的值： 0 - 隐藏； 1 - *(默认)* 显示。
show_work_period	integer	是否在图形上显示工作时间。 可能的值： 0 - 隐藏； 1 - *(默认)* 显示。
show_triggers	integer	是否在图形上显示触发器线。 可能的值： 0 - 隐藏； 1 - *(默认)* 显示。
templateid	ID	父模板图形的 ID。 属性行为： - 只读
yaxismax	float	Y 轴的固定最大值。 默认值：100。
yaxismin	float	Y 轴的固定最小值。 默认值：0。
ymax_itemid	ID	用作 Y 轴最大值的监控项 ID。 如果用户无权访问指定的监控项，则图形将按 `ymax_type` 设置为“calculated”进行渲染。
ymax_type	integer	Y 轴最大值的计算方法。 可能的值： 0 - *(默认)* calculated； 1 - fixed； 2 - item。
ymin_itemid	ID	用作 Y 轴最小值的监控项 ID。 如果用户无权访问指定的监控项，则图形将按 `ymin_type` 设置为“calculated”进行渲染。

Property	Type	Description
ymin_type	integer	Y 轴最小值的计算方法。 可能的值： 0 - *(默认)* calculated； 1 - fixed； 2 - item。
uuid	string	通用唯一标识符，用于将导入的图形链接到已存在的图形。如果未提供，则自动生成。 属性行为： - 如果图形属于模板，则 支持

创建

描述

`object graph.create(object/array graphs)`

此方法允许您创建新的图形。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的图形。

除了标准图形属性之外，该方法还接受以下参数。

参数	Type	描述
gitems	array	要为该图形创建的图形元素。 参数行为： - 必需

返回值

(object) 返回一个对象，包含在 graphids 属性下创建的图表的 ID。返回的 ID 的顺序与传递的图表的顺序相匹配。

示例

创建一个图表

创建一个具有两个监控项的图表。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "graph.create",
    "params": {
        "name": "MySQL bandwidth",
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
```

```
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

另请参阅

- 图形监控项

来源

ui/include/classes/api/services/CGraph.php 中的 CGraph::create()。

删除

描述

`object graph.delete(array graphIds)`

此方法允许您删除图形。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的图表的 ID。

返回值

(object) 返回一个对象，该对象包含 graphids 属性下已删除图表的 ID。

示例

删除多个图表

删除两个图表。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "graph.delete",
    "params": [
        "652",
        "653"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
```

```
            "653"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CGraph.php 中的 CGraph::delete()。

更新

描述

`object graph.update(object/array graphs)`

此方法允许你更新现有的图形。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的图形属性。

必须为每个图形定义 graphid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除标准图形属性外，该方法还接受以下参数。

参数	类型	描述
gitems	array	用于替换现有图形项的图形项。如果图形项定义了 gitemid 属性，则会更新该图形项；否则将创建一个新的图形项。

返回值

(object) 返回一个对象，该对象包含 graphids 属性下更新的图表的 ID。

示例

设置 Y 刻度的最大值

将 Y 刻度的最大值设置为固定值 100。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "graph.update",
    "params": {
        "graphid": "439",
        "ymax_type": 1,
        "yaxismax": 100
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CGraph.php 中的 CGraph::update()。

获取

描述

`integer/array graph.get(object parameters)`

该方法允许你根据给定参数检索图形。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
graphids	ID/array	仅返回具有给定 ID 的图形。
groupids	ID/array	仅返回属于给定主机组或模板组中的主机或模板的图形。
templateids	ID/array	仅返回属于给定模板的图形。
hostids	ID/array	仅返回属于给定主机的图形。
itemids	ID/array	仅返回包含给定监控项的图形。
templated	boolean	如果设置为 `true`，则仅返回属于模板的图形。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的图形。
expandName	flag	展开图形名称中的宏。
selectDiscoveryData	query	返回一个带有图形发现对象数据的 `discoveryData` 属性。图形发现对象将已发现的图形关联到发现该图形所基于的图形原型。 它具有以下属性： `parent_graphid` - (ID) 创建该图形所基于的图形原型 ID； `status` - (int) 图形发现状态： 0 - (默认) 图形已发现， 1 - 图形不再被发现； `ts_delete` - (timestamp) 不再被发现的图形将被删除的时间。
selectDiscoveryRule	query	返回一个`discoveryRule` 属性，其中包含创建该图形的低级别发现规则。
selectHostGroups	query	返回一个`hostgroups` 属性，其中包含该图形所属的主机组。
selectTemplateGroups	query	返回一个`templategroups` 属性，其中包含该图形所属的模板组。
selectTemplates	query	返回一个`templates` 属性，其中包含该图形所属的模板。
selectHosts	query	返回一个`hosts` 属性，其中包含该图形所属的主机。
selectItems	query	返回一个`items` 属性，其中包含图形中使用的监控项。
selectGraphItems	query	返回一个`gitems` 属性，其中包含图形中使用的监控项。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 `text` 数据类型 的属性。 支持以下附加属性： `host` - 图形所属主机的技术名称； `hostid` - 图形所属主机的 ID。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`graphid`、`name`、`graphtype`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	

参数	类型	说明
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGraphDiscovery	query	返回一个带有图形发现对象的 graphDiscovery 属性。图形发现对象将图形关联到创建该图形所基于的图形原型。 此查询已弃用，请改用 selectDiscoveryData。

返回值

(integer/array) 返回其中之一：

- 一个对象的数组；
- 如果使用了 countOutput 参数，则为检索到的对象的数量。

示例

从主机上检索图表

检索主机"10107 " 的所有图表，并按名称排序。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "graph.get",
    "params": {
        "output": "extend",
        "hostids": 10107,
        "sortfield": "name"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "612",
            "name": "CPU jumps",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "439",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "613",
            "name": "CPU load",
```

```
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "433",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "1",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "614",
            "name": "CPU utilization",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "387",
            "show_work_period": "1",
            "show_triggers": "0",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "1",
            "ymax_type": "1",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "645",
            "name": "Disk space usage /",
            "width": "600",
            "height": "340",
            "yaxismin": "0",
            "yaxismax": "0",
            "templateid": "0",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "4"
        }
    ],
    "id": 1
```

```
}
```

另请参见

- 发现规则
- 图形监控项
- 监控项
- 主机
- 主机组
- 模板
- 模板组

来源

ui/include/classes/api/services/CGraph.php 中的 CGraph::get()。

媒介类型

此类用于处理媒介类型。

对象参考：

- 媒介类型
  - Webhook 参数
  - 脚本参数
- 消息模板

可用方法：

- mediatype.create - 创建新的媒介类型
- mediatype.delete - 删除媒介类型
- mediatype.get - 检索媒介类型
- mediatype.update - 更新媒介类型

媒介类型对象

以下对象与 `mediatype` API 直接相关。

媒体类型

媒体类型对象具有以下属性。

属性	类型	描述
mediatypeid	ID	媒体类型的 ID。 属性行为： - 只读 - 更新操作时必需
name	string	媒体类型的名称。 属性行为： - 创建操作时必需
type	integer	媒体类型使用的传输方式。 可能的值： 0 - 电子邮件； 1 - 脚本； 2 - SMS； 4 - webhook； 5 - 推送。 属性行为： - 创建操作时必需

属性	类型	描述
exec_path	string	脚本文件的名称（例如 notification.sh）。该文件位于服务器配置参数AlertScriptsPath 指定的目录中。 属性行为： - `type` 设置为 “Script” 时必需
gsm_modem	string	GSM 调制解调器的串行设备名称。 属性行为： - `type` 设置为 “SMS” 时必需
passwd	string	身份验证密码。 属性行为： - `smtp_authentication` 设置为 “Normal password” 时支持
provider	integer	电子邮件提供商。 可能的值： 0 - （默认）通用 SMTP； 1 - Gmail； 2 - Gmail 中继； 3 - Office365； 4 - Office365 中继。
smtp_email	string	用于发送通知的电子邮件地址。 属性行为： - `type` 设置为 “Email” 时必需
smtp_helo	string	SMTP HELO。 属性行为： - `type` 设置为 “Email” 时支持
smtp_server	string	SMTP 服务器。 属性行为： - `type` 设置为 “Email” 时必需
smtp_port	integer	要连接的 SMTP 服务器端口。 默认值：25。 属性行为： - `type` 设置为 “Email” 时支持
smtp_security	integer	要使用的 SMTP 连接安全级别。 可能的值： 0 - （默认）无； 1 - STARTTLS； 2 - SSL/TLS。 属性行为： - `type` 设置为 “Email” 时支持
smtp_verify_host	integer	SMTP 的 SSL 主机验证。 可能的值： 0 - （默认）否； 1 - 是。 属性行为： - `smtp_security` 设置为 “STARTTLS” 或 “SSL/TLS” 时支持

属性	类型	描述
smtp_verify_peer	integer	SMTP 的 SSL 对等方验证。 可能的值： 0 - （默认）否； 1 - 是。
smtp_authentication	integer	属性行为： - `smtp_security` 设置为 “STARTTLS” 或 “SSL/TLS” 时支持 要使用的 SMTP 身份验证方法。 可能的值： 0 - （默认）无； 1 - 普通密码； 2 - OAuth 令牌。 Office365 中继电子邮件提供商不允许使用 OAuth 身份验证。
redirection_url	string	属性行为： - `type` 设置为 “Email” 时支持 用于重定向 OAuth 授权的 Zabbix 前端 URL。 默认值： API 设置属性 `url` 的值，并附加 `zabbix.php?action=oauth.authorize` 部分
client_id	string	属性行为： - `smtp_authentication` 设置为 “OAuth token” 时必需 在 OAuth 授权服务器中注册的客户端标识符。
client_secret	string	属性行为： - `smtp_authentication` 设置为 “OAuth token” 时必需 在 OAuth 授权服务器中注册的客户端密钥。仅超级管理员类型的用户可访问。
authorization_url	string	属性行为： - `smtp_authentication` 设置为 “OAuth token” 时必需 用于获取访问令牌和刷新令牌的 OAuth URL（包含参数）。
token_url	string	属性行为： - `smtp_authentication` 设置为 “OAuth token” 时必需 用于将授权令牌交换为访问令牌和刷新令牌的 OAuth URL。 服务器也使用此 URL 刷新无效的访问令牌。
tokens_status	integer	属性行为： - `smtp_authentication` 设置为 “OAuth token” 时必需 令牌状态的位掩码。 可能的值： 0 - （默认）两个令牌均包含无效值； 1 - 访问令牌包含有效值； 2 - 刷新令牌包含有效值； 3 - 两个令牌均包含有效值。
access_token	string	属性行为： - `smtp_authentication` 设置为 “OAuth token” 时支持 OAuth 访问令牌值。
		属性行为： - `smtp_authentication` 设置为 “OAuth token” 时必需

属性	类型	描述
access_token_updated	timestamp	服务器使用 refresh_token 刷新令牌时，或 API 更改令牌时，对 access_token 执行最后一次修改的时间戳。 属性行为： - smtp_authentication 设置为 “OAuth token” 时支持
access_expires_in	integer	access_token 失效并需要向 refresh_url 发出请求前的时间（以秒为单位）。 由 Zabbix 服务器在刷新 access_token 时设置，或由 API 在令牌更改时设置。 时间戳通过将 access_token_updated 的值相加计算得出。 属性行为： - smtp_authentication 设置为 “OAuth token” 时支持
refresh_token	string	OAuth 刷新令牌值。 属性行为： - smtp_authentication 设置为 “OAuth token” 时必需
status	integer	媒体类型是否启用。 可能的值： 0 - （默认）已启用； 1 - 已禁用。
username	string	用户名。 属性行为： - smtp_authentication 设置为 “Normal password” 时支持
maxsessions	integer	可并行处理的最大告警数量。 当 type 设置为 “SMS” 时，可能的值：1。 当 type 设置为 “Email”、“Script” 或 “Webhook” 时，可能的值：0-100。
maxattempts	integer	默认值：1。 发送告警的最大尝试次数。 可能的值：1-100。
attempt_interval	string	默认值：3。 重试尝试之间的间隔。 接受秒数（例如 1、30、60），或带有时间后缀的值（例如 30s、1m、1h）。 可能的值：0-1h。
message_format	integer	默认值：10s。 消息格式。 可能的值： 0 - 纯文本； 1 - （默认）HTML。
script	text	属性行为： - type 设置为 “Email” 时支持 webhook 脚本正文（JavaScript）。 属性行为： - type 设置为 “Webhook” 时必需

属性	类型	描述
timeout	string	webhook 脚本超时时间。 接受秒数（例如 1、60），或带有时间后缀的值（例如 30s、1m） 可能的值：1-60s。 默认值：30s。
process_tags	integer	属性行为： - type 设置为 “Webhook” 时支持 将 webhook 脚本响应中的 JSON 属性值作为标签进行处理。这些标签会添加到任何现有的问题标签中。 可能的值： 0 - （默认）忽略 webhook 脚本响应； 1 - 将 webhook 脚本响应作为标签进行处理。
show_event_menu	integer	属性行为： - type 设置为 “Webhook” 时支持 在事件菜单中包含一个指向自定义 URL 的条目。同时将 urls 属性添加到problem.get 和event.get 的输出中。 可能的值： 0 - （默认）不包含事件菜单条目或 urls 属性； 1 - 包含事件菜单条目和 urls 属性。
event_menu_url	string	属性行为： - type 设置为 “Webhook” 时支持 用于事件菜单条目以及problem.get 和event.get 返回的 urls 属性中的 URL。
event_menu_name	string	属性行为： - show_event_menu 设置为 “Include event menu entry and urls property” 时支持 用于事件菜单条目以及problem.get 和event.get 返回的 urls 属性中的名称。
parameters	array	属性行为： - show_event_menu 设置为 “Include event menu entry and urls property” 时支持 Webhook 或脚本参数。
description	text	属性行为： - type 设置为 “Webhook” 或 “Script” 时支持 媒体类型描述。

webhook 参数

webhook 参数具有以下属性。

属性	Type	说明
name	string	参数名称。 属性行为： - 必填
value	string	参数值，支持宏。 支持的宏请参见支持的宏页面。

脚本参数

脚本参数具有以下属性。

属性	类型	说明
sortorder	integer	参数值将作为命令行参数传递给脚本时的顺序，从 0 开始，0 为第一个。 属性行为: - 必填
value	string	参数值，支持宏。 支持的宏请参见支持的宏页面。

消息模板

消息模板对象定义了一个模板，该模板将用作动作操作发送通知时的默认消息。它具有以下属性。

属性	类型	描述
eventsource	integer	事件源。 可能的值： 0 - 触发器； 1 - 发现； 2 - 自动注册； 3 - 内部； 4 - 服务。 属性行为： - 必需
recovery	integer	操作模式。 可能的值： 0 - 操作； 1 - 恢复操作； 2 - 更新操作。 属性行为： - 必需
subject	string	消息主题。
message	string	消息文本。

创建

描述

`object mediatype.create(object/array mediaTypes)`

此方法用于创建新的媒介类型。

Note:
此方法仅适用于超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件的 `$ZBX_FEATURE_FLAGS['media_type_denylist']` 功能标志中指定了投递方式，则用户不允许调用对应媒介类型的方法。

参数

`(object/array)` 要创建的媒介类型。

除了标准媒介类型属性之外，该方法还接受以下参数。

参数	Type	说明
message_templates	array	要为该媒介类型创建的消息模板。

返回值

`(object)` 返回一个对象，其中包含在 `mediatypeids` 属性下创建的媒体类型的 ID。返回的 ID 的顺序与传递的媒体类型的顺序匹配。

示例

创建电子邮件媒介类型

使用自定义 SMTP 端口和消息模板创建新的电子邮件媒介类型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "0",
        "name": "Email",
        "smtp_server": "mail.example.com",
        "smtp_helo": "example.com",
        "smtp_email": "zabbix@example.com",
        "smtp_port": "587",
        "message_format": "1",
        "message_templates": [
            {
                "eventsource": "0",
                "recovery": "0",
                "subject": "Problem: {EVENT.NAME}",
                "message": "Problem \"{EVENT.NAME}\" on host \"{HOST.NAME}\" started at {EVENT.TIME}."
            },
            {
                "eventsource": "0",
                "recovery": "1",
                "subject": "Resolved in {EVENT.DURATION}: {EVENT.NAME}",
                "message": "Problem \"{EVENT.NAME}\" on host \"{HOST.NAME}\" has been resolved at {EVENT.R
            },
            {
                "eventsource": "0",
                "recovery": "2",
                "subject": "Updated problem in {EVENT.AGE}: {EVENT.NAME}",
                "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem \"{EVENT.NAME}\" on host \"{HOST
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "7"
        ]
    },
    "id": 1
}
```

创建脚本媒介类型

使用自定义数量值创建新的脚本媒体类型尝试次数以及尝试次数之间的间隔。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "1",
        "name": "Push notifications",
        "exec_path": "push-notification.sh",
        "maxattempts": "5",
```

```
        "attempt_interval": "11s",
        "parameters": [
            {
                "sortorder": "0",
                "value": "{ALERT.SENDTO}"
            },
            {
                "sortorder": "1",
                "value": "{ALERT.SUBJECT}"
            },
            {
                "sortorder": "2",
                "value": "{ALERT.MESSAGE}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "8"
        ]
    },
    "id": 1
}
```

创建 webhook 媒介类型

创建新的 Webhook 媒介类型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "4",
        "name": "Webhook",
        "script": "var Webhook = {\r\n    token: null,\r\n    to: null,\r\n    subject: null,\r\n    messa
        "parameters": [
            {
                "name": "Message",
                "value": "{ALERT.MESSAGE}"
            },
            {
                "name": "Subject",
                "value": "{ALERT.SUBJECT}"
            },
            {
                "name": "To",
                "value": "{ALERT.SENDTO}"
            },
            {
                "name": "Token",
                "value": "<Token>"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "9"
        ]
    },
    "id": 1
}
```

来源

CMediaType::create() in *ui/include/classes/api/services/CMediaType.php*。

删除

描述

`object mediatype.delete(array mediaTypeIds)`

此方法允许您删除媒体类型。

> **Note:**
> 此方法仅适用于超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件的 `$ZBX_FEATURE_FLAGS['media_type_denylist']` 功能标志中指定了传递方法，则不允许用户调用对应媒体类型的方法。

参数

(array) 要删除的媒介类型的 ID。

返回值

(object) 返回一个对象，该对象包含 mediatypeids 属性下已删除媒介类型的 ID。

示例

删除多个媒介类型

删除两个媒介类型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.delete",
    "params": [
        "3",
        "5"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "3",
            "5"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CMediaType.php 中的 CMediaType::delete()。

更新

描述

`object mediatype.update(object/array mediaTypes)`

此方法允许更新现有的媒体类型。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
> 如果在前端配置文件的 `$ZBX_FEATURE_FLAGS['media_type_denylist']` 功能标志中指定了投递方式，则不允许用户为相应的媒体类型调用此方法。

参数

(object/array) 要更新的媒介类型属性。

必须为每个媒介类型定义 `mediatypeid` 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准媒介类型属性之外，该方法还接受以下参数。

参数	类型	描述
message_templates	array	用于替换当前消息模板的消息模板。

返回值

(object) 返回包含 `mediatypeids` 属性下所更新 IDs 的对象。

示例

启用媒介类型

启用媒体类型，即将其状态设置为"0"。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.update",
    "params": {
        "mediatypeid": "6",
        "status": "0"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "6"
        ]
    },
    "id": 1
}
```

来源

ui/include/classes/api/services/CMediaType.php 中的 CMediaType::update()。

获取

描述

`integer/array mediatype.get(object parameters)`

该方法允许根据给定的参数检索媒介类型。

> **Note:**
> 任何类型的用户都可以使用此方法。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件的 `$ZBX_FEATURE_FLAGS['media_type_denylist']` 功能标志中指定了投递方式，则不允许用户调用对应媒介类型的方法。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

> **Note:**
> 请求媒体类型的用户相关信息时，*Admin* 和 *User* 类型的用户只能检索其自身用户的数据。示例请参见以 *Admin* 身份检索媒体类型。

参数	类型	描述
mediatypeids	ID/array	仅返回具有给定 ID 的媒体类型。
mediaids	ID/array	仅返回被给定媒介使用的媒体类型。
userids	ID/array	仅返回被给定用户使用的媒体类型。
selectActions	query	返回一个`actions` 属性，其中包含使用该媒体类型的动作。
selectMessageTemplates	query	返回一个`message_templates` 属性，其中包含媒体类型消息数组。
selectUsers	query	参数行为： - 对 *Super admin* 类型用户支持 返回一个`users` 属性，其中包含使用该媒体类型的用户。 有关基于用户类型的限制，请参见`user.get`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`mediatypeid`。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 `text` 数据类型 的属性。
output	query	*Admin* 和 *User* 类型用户可用的媒体类型对象 属性：`mediatypeid`、`name`、`type`、`status`、`maxattempts`。 要返回的媒体类型对象属性。 *Admin* 和 *User* 类型用户只能检索以下属性：`mediatypeid`、`name`、`type`、`status`、`maxattempts`、`description`。示例请参见以 *Admin* 身份检索媒体类型。
search	object	默认值：`extend`。 返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，将执行 `LIKE "%...%"` 搜索。 仅支持 `string` 和 `text` 数据类型 的属性。
countOutput	boolean	*Admin* 和 *User* 类型用户可用的媒体类型对象 属性：`name`、`description`。 这些参数在参考说明中进行了描述。

参数	类型	描述
editable	boolean	
excludeSearch	boolean	
limit	integer	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

（整型/数组）返回其中之一：

- 一个对象数组；
- 如果使用 `countOutput` 参数，返回被检索对象的数量。

示例

检索媒介类型

检索所有已配置的媒介类型。以下示例返回两种媒介类型：

- 电子邮件媒介类型；
- SMS 媒介类型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "selectMessageTemplates": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "name": "Email",
            "smtp_server": "mail.example.com",
            "smtp_helo": "example.com",
            "smtp_email": "zabbix@example.com",
            "exec_path": "",
            "gsm_modem": "",
            "username": "",
            "passwd": "",
            "status": "0",
            "smtp_port": "25",
            "smtp_security": "0",
            "smtp_verify_peer": "0",
            "smtp_verify_host": "0",
            "smtp_authentication": "0",
            "maxsessions": "1",
            "maxattempts": "3",
            "attempt_interval": "10s",
            "message_format": "0",
            "script": "",
            "timeout": "30s",
```

```
        "process_tags": "0",
        "show_event_menu": "1",
        "event_menu_url": "",
        "event_menu_name": "",
        "description": "",
        "provider": "0",
        "message_templates": [
            {
                "eventsource": "0",
                "recovery": "0",
                "subject": "Problem: {EVENT.NAME}",
                "message": "Problem started at {EVENT.TIME} on {EVENT.DATE}\r\nProblem name: {EVENT.NA
            },
            {
                "eventsource": "0",
                "recovery": "1",
                "subject": "Resolved: {EVENT.NAME}",
                "message": "Problem has been resolved at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE
            },
            {
                "eventsource": "0",
                "recovery": "2",
                "subject": "Updated problem: {EVENT.NAME}",
                "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVEN
            },
            {
                "eventsource": "1",
                "recovery": "0",
                "subject": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}",
                "message": "Discovery rule: {DISCOVERY.RULE.NAME}\r\n\r\nDevice IP: {DISCOVERY.DEVICE.
            },
            {
                "eventsource": "2",
                "recovery": "0",
                "subject": "Autoregistration: {HOST.HOST}",
                "message": "Host name: {HOST.HOST}\r\nHost IP: {HOST.IP}\r\nAgent port: {HOST.PORT}"
            }
        ],
        "parameters": []
    },
    {
        "mediatypeid": "3",
        "type": "2",
        "name": "SMS",
        "smtp_server": "",
        "smtp_helo": "",
        "smtp_email": "",
        "exec_path": "",
        "gsm_modem": "/dev/ttyS0",
        "username": "",
        "passwd": "",
        "status": "0",
        "smtp_port": "25",
        "smtp_security": "0",
        "smtp_verify_peer": "0",
        "smtp_verify_host": "0",
        "smtp_authentication": "0",
        "maxsessions": "1",
        "maxattempts": "3",
        "attempt_interval": "10s",
        "message_format": "1",
        "script": "",
```

```
            "timeout": "30s",
            "process_tags": "0",
            "show_event_menu": "1",
            "event_menu_url": "",
            "event_menu_name": "",
            "description": "",
            "provider": "0",
            "message_templates": [
                {
                    "eventsource": "0",
                    "recovery": "0",
                    "subject": "",
                    "message": "{EVENT.SEVERITY}: {EVENT.NAME}\r\nHost: {HOST.NAME}\r\n{EVENT.DATE} {EVENT
                },
                {
                    "eventsource": "0",
                    "recovery": "1",
                    "subject": "",
                    "message": "RESOLVED: {EVENT.NAME}\r\nHost: {HOST.NAME}\r\n{EVENT.DATE} {EVENT.TIME}"
                },
                {
                    "eventsource": "0",
                    "recovery": "2",
                    "subject": "",
                    "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVEN
                },
                {
                    "eventsource": "1",
                    "recovery": "0",
                    "subject": "",
                    "message": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}"
                },
                {
                    "eventsource": "2",
                    "recovery": "0",
                    "subject": "",
                    "message": "Autoregistration: {HOST.HOST}\r\nHost IP: {HOST.IP}\r\nAgent port: {HOST.P
                }
            ],
            "parameters": []
        }
    ],
    "id": 1
}
```


以 *Admin* 身份检索媒体类型

作为 *Admin* 类型用户，检索所有已启用的媒体类型，以及使用这些媒体类型的用户。以下示例返回两种媒体类型：

- 电子邮件媒体类型，包含一个用户（仅 *Admin* 类型用户自己的用户）；
- SMS 媒体类型，不包含任何用户。

请求：


```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "filter": {
            "status": 0
        },
        "selectUsers": "extend"
    },
    "id": 1
```

```
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "name": "Email",
            "status": "0",
            "description": "",
            "maxattempts": "3",
            "users": [
                {
                    "userid": "3",
                    "username": "database-admin",
                    "name": "John",
                    "surname": "Doe",
                    "url": "",
                    "autologin": "0",
                    "autologout": "0",
                    "lang": "default",
                    "refresh": "30s",
                    "theme": "default",
                    "attempt_failed": "0",
                    "attempt_ip": "",
                    "attempt_clock": "0",
                    "rows_per_page": "50",
                    "timezone": "default",
                    "roleid": "2",
                    "provisioned": "0"
                }
            ]
        },
        {
            "mediatypeid": "3",
            "type": "2",
            "name": "SMS",
            "status": "0",
            "description": "",
            "maxattempts": "3",
            "users": []
        }
    ],
    "id": 1
}
```

检索脚本和 webhook 媒介类型

以下示例返回三种媒介类型：

- 带参数的脚本媒介类型;
- 不带参数的脚本媒介类型;
- 带参数的 Webhook 媒介类型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": ["mediatypeid", "name", "parameters"],
        "filter": {
            "type": [1, 4]
```

```
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "10",
            "name": "Script with parameters",
            "parameters": [
                {
                    "sortorder": "0",
                    "value": "{ALERT.SENDTO}"
                },
                {
                    "sortorder": "1",
                    "value": "{EVENT.NAME}"
                },
                {
                    "sortorder": "2",
                    "value": "{ALERT.MESSAGE}"
                },
                {
                    "sortorder": "3",
                    "value": "Zabbix alert"
                }
            ]
        },
        {
            "mediatypeid": "13",
            "name": "Script without parameters",
            "parameters": []
        },
        {
            "mediatypeid": "11",
            "name": "Webhook",
            "parameters": [
                {
                    "name": "alert_message",
                    "value": "{ALERT.MESSAGE}"
                },
                {
                    "name": "event_update_message",
                    "value": "{EVENT.UPDATE.MESSAGE}"
                },
                {
                    "name": "host_name",
                    "value": "{HOST.NAME}"
                },
                {
                    "name": "trigger_description",
                    "value": "{TRIGGER.DESCRIPTION}"
                },
                {
                    "name": "trigger_id",
                    "value": "{TRIGGER.ID}"
                },
                {
                    "name": "alert_source",
```

```
                    "value": "Zabbix"
                }
            ]
        }
    ],
    "id": 1
}
```

参见

- 用户

来源

ui/include/classes/api/services/CMediaType.php 中的 CMediaType::get()。

审计日志

此类用于处理审计日志。

对象引用：

- 审计日志

可用方法：

- auditlog.get - 检索审计日志记录

审计日志对象

以下对象与 `auditlog` API 直接相关。

审计日志

审计日志对象包含有关用户操作的信息。它具有以下属性。

属性	类型	描述
auditid	ID	审计日志条目的 ID。使用 CUID 算法生成。
userid	ID	审计日志条目作者的用户 ID。
username	string	审计日志条目作者的用户名。
clock	timestamp	审计日志条目的创建时间戳。
ip	string	审计日志条目作者的 IP 地址。
action	integer	审计日志条目的操作。 可能的值： 0 - 添加； 1 - 更新； 2 - 删除； 4 - 登出； 7 - 执行； 8 - 登录； 9 - 登录失败； 10 - 清除历史记录； 11 - 刷新配置； 12 - 推送。

属性	类型	描述
resourcetype	integer	审计日志条目的资源类型。 可能的值： 0 - 用户； 3 - 介质类型； 4 - 主机； 5 - 动作； 6 - 图形； 11 - 用户组； 13 - 触发器； 14 - 主机组； 15 - 监控项； 16 - 图像； 17 - 值映射； 18 - 服务； 19 - 拓扑； 22 - Web 场景； 23 - 发现规则； 25 - 脚本； 26 - Proxy； 27 - 维护； 28 - 正则表达式； 29 - 宏； 30 - 模板； 31 - 触发器原型； 32 - 图标映射； 33 - 仪表板； 34 - 事件关联； 35 - 图形原型； 36 - 监控项原型； 37 - 主机原型； 38 - 自动注册； 39 - 模块； 40 - 设置； 41 - Housekeeping； 42 - 身份验证； 43 - 模板仪表板； 44 - 用户角色； 45 - API 令牌； 46 - 计划报表； 47 - 高可用节点； 48 - SLA； 49 - 用户目录； 50 - 模板组； 51 - 连接器； 52 - LLD 规则； 53 - 历史记录； 54 - 多因素认证； 55 - Proxy 组； 56 - LLD 规则原型。
resourceid	ID	审计日志条目的资源标识符。
resource_cuid	ID	审计日志条目的资源唯一标识符，使用 CUID 算法生成。
resourcename	string	审计日志条目资源的人类可读名称。
recordsetid	ID	审计日志条目的记录集 ID。在同一操作期间创建的审计日志记录将具有相同的记录集 ID。使用 CUID 算法生成。

属性	类型	描述
details	text	审计日志条目详情。详情以 JSON 对象形式存储，其中每个属性名都是发生更改的属性或嵌套对象的路径，而每个值都包含有关该属性或嵌套对象中更改的数据（采用数组格式）。 可能的值格式： ["add"] - 已添加嵌套对象； ["add", "<value>"] - 已添加对象的属性等于 <value>； ["update"] - 已更新嵌套对象； ["update", "<new value>", "<old value>"] - 已更新对象的属性已从 <old value> 更改为 <new value>； ["delete"] - 已删除嵌套对象。

获取

描述

`integer/array auditlog.get(object parameters)`

该方法允许您根据给定参数检索审计日志记录。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
auditids	ID/array	仅返回具有给定 ID 的审计日志。
userids	ID/array	仅返回由给定用户创建的审计日志。
time_from	timestamp	仅返回在给定时间之后或该时间点创建的审计日志条目。
time_till	timestamp	仅返回在给定时间之前或该时间点创建的审计日志条目。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`auditid`、`userid`、`clock`。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下之一：

- 对象数组；
- 如果使用了 `countOutput` 参数，则返回已检索对象的数量。

示例

检索审计日志

检索最近两条审计日志记录。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": "extend",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 2
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "auditid": "cksstgfam0001yhdcc41y20q2",
            "userid": "1",
            "username": "Admin",
            "clock": "1629975715",
            "ip": "127.0.0.1",
            "action": "1",
            "resourcetype": "0",
            "resourceid": "0",
            "resource_cuid": "0",
            "resourcename": "Jim",
            "recordsetid": "cksstgfal0000yhdcso67ondl",
            "details": "{\"user.name\":[\"update\",\"Jim\",\"\"],\"user.medias[37]\":[\"add\"],\"user.medi
        },
        {
            "auditid": "ckssofl0p0001yhdcqxclsg8r",
            "userid": "1",
            "username": "Admin",
            "clock": "1629967278",
            "ip": "127.0.0.1",
            "action": "0",
            "resourcetype": "0",
            "resourceid": "20",
            "resource_cuid": "0",
            "resourcename": "John",
            "recordsetid": "ckssofl0p0000yhdcpxyo1jgo",
            "details": "{\"user.username\":[\"add\",\"John\"], \"user.userid:\":[\"add\",\"20\"],\"user.us
        }
    ],
    "id": 1
}
```

另请参阅

- 审计日志对象

源码位置

CAuditLog::get() 在 *ui/include/classes/api/services/CAuditLog.php* 文件中。

已发现主机

此类用于处理已发现的主机。

对象引用：

- 已发现主机

可用方法：

- dhost.get - 检索已发现的主机

发现的主机对象

以下对象与 dhost API 直接相关。

已发现主机

Note:
已发现主机由 Zabbix 服务器创建，且不能通过 API 修改。

已发现主机对象包含由网络发现规则发现的主机信息。它具有以下属性。

Property	Type	Description
dhostid	ID	已发现主机的 ID。
druleid	ID	检测到该主机的发现规则的 ID。
lastdown	timestamp	已发现主机上次变为不可用的时间。
lastup	timestamp	已发现主机上次变为可用的时间。
status	integer	已发现主机是可用还是不可用。如果主机至少有一个处于活动状态的已发现服务，则该主机为可用。 可能的值： 0 - 主机可用； 1 - 主机不可用。

获取

描述

`integer/array dhost.get(object parameters)`

该方法允许你根据给定参数检索已发现的主机。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
dhostids	ID/array	仅返回具有给定 ID 的已发现主机。
druleids	ID/array	仅返回由给定发现规则创建的已发现主机。
dserviceids	ID/array	仅返回正在运行给定服务的已发现主机。
selectDRules	query	返回`drules` 属性，其中包含检测到该主机的发现规则数组。
selectDServices	query	返回`dservices` 属性，其中包含在该主机上运行的已发现服务。 支持 `count`。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： `selectDServices` - 结果将按 `dserviceid` 排序。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`dhostid`、`druleid`。
countOutput	boolean	这些参数在参考说明中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	

参数	类型	说明
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中一种结果：

- 一个对象数组；
- 如果使用了参数 countOutput，则返回检索到的对象的数量。

示例

通过发现规则检索发现的主机

检索发现规则"4" 检测到的所有主机及其正在运行的已发现服务。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dhost.get",
    "params": {
        "output": "extend",
        "selectDServices": "extend",
        "druleids": "4"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dservices": [
                {
                    "dserviceid": "1",
                    "dhostid": "1",
                    "type": "4",
                    "key_": "",
                    "value": "",
                    "port": "80",
                    "status": "0",
                    "lastup": "1337697227",
                    "lastdown": "0",
                    "dcheckid": "5",
                    "ip": "192.168.1.1",
                    "dns": "station.company.lan"
                }
            ],
            "dhostid": "1",
            "druleid": "4",
            "status": "0",
            "lastup": "1337697227",
            "lastdown": "0"
        },
        {
            "dservices": [
                {
```

```
                "dserviceid": "2",
                "dhostid": "2",
                "type": "4",
                "key_": "",
                "value": "",
                "port": "80",
                "status": "0",
                "lastup": "1337697234",
                "lastdown": "0",
                "dcheckid": "5",
                "ip": "192.168.1.4",
                "dns": "john.company.lan"
            }
        ],
        "dhostid": "2",
        "druleid": "4",
        "status": "0",
        "lastup": "1337697234",
        "lastdown": "0"
    },
    {
        "dservices": [
            {
                "dserviceid": "3",
                "dhostid": "3",
                "type": "4",
                "key_": "",
                "value": "",
                "port": "80",
                "status": "0",
                "lastup": "1337697234",
                "lastdown": "0",
                "dcheckid": "5",
                "ip": "192.168.1.26",
                "dns": "printer.company.lan"
            }
        ],
        "dhostid": "3",
        "druleid": "4",
        "status": "0",
        "lastup": "1337697234",
        "lastdown": "0"
    },
    {
        "dservices": [
            {
                "dserviceid": "4",
                "dhostid": "4",
                "type": "4",
                "key_": "",
                "value": "",
                "port": "80",
                "status": "0",
                "lastup": "1337697234",
                "lastdown": "0",
                "dcheckid": "5",
                "ip": "192.168.1.7",
                "dns": "mail.company.lan"
            }
        ],
        "dhostid": "4",
        "druleid": "4",
```

```
            "status": "0",
            "lastup": "1337697234",
            "lastdown": "0"
        }
    ],
    "id": 1
}
```

另请参阅

- 已发现的服务
- 发现规则

来源

ui/include/classes/api/services/CDHost.php 中的 CDHost::get()。

已发现服务

此类用于处理已发现的服务。

对象引用：

- 已发现服务

可用方法：

- dservice.get - 检索已发现的服务

发现的服务对象

以下对象与 `dservice` API 直接相关。

已发现服务

Note:
已发现服务由 Zabbix 服务器创建，无法通过 API 修改。

已发现服务对象包含由主机上的网络发现规则发现的服务信息。它具有以下属性。

属性	类型	描述
dserviceid	ID	已发现服务的 ID。
dcheckid	ID	用于检测该服务的发现检查 ID。
dhostid	ID	运行该服务的已发现主机 ID。
dns	string	运行该服务的主机的 DNS。
ip	string	运行该服务的主机的 IP 地址。
lastdown	timestamp	已发现服务上次停止运行的时间。
lastup	timestamp	已发现服务上次恢复运行的时间。
port	integer	服务端口号。
status	integer	服务状态。 可能的值： 0 - 服务运行； 1 - 服务停止。
value	string	执行 Zabbix agent、SNMPv1、SNMPv2 或 SNMPv3 发现检查时，该服务返回的值。

获取

描述

`integer/array dservice.get(object parameters)`

该方法允许您根据给定参数检索已发现的服务。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
dserviceids	ID/array	仅返回具有给定 ID 的已发现服务。
dhostids	ID/array	仅返回属于给定已发现主机的已发现服务。
dcheckids	ID/array	仅返回由给定发现检查检测到的已发现服务。
druleids	ID/array	仅返回由给定发现规则检测到的已发现服务。
selectDRules	query	返回drules 属性，其中包含检测到该服务的发现规则数组。
selectDHosts	query	返回dhosts 属性，其中包含该服务所属的已发现主机数组。
selectHosts	query	返回hosts 属性，其中包含与该服务具有相同 IP 地址和 proxy 的主机。 支持 count。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectHosts - 结果将按 hostid 排序。
sortfield	string/array	按给定属性对结果进行排序。 可选值：dserviceid、dhostid、ip。
countOutput	boolean	这些参数在参考说明中有描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中一种结果：

- 一个对象数组；
- 如果使用了参数 countOutput，则返回检索到的对象的数量。

示例

检索在主机上被发现的服务

检索在被发现主机 “11” 上检测出的所有被发现的服务。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "dservice.get",
    "params": {
        "output": "extend",
        "dhostids": "11"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dserviceid": "12",
            "dhostid": "11",
            "value": "",
            "port": "80",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650607",
            "dcheckid": "5",
            "ip": "192.168.1.134",
            "dns": "john.local"
        },
        {
            "dserviceid": "13",
            "dhostid": "11",
            "value": "",
            "port": "21",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650610",
            "dcheckid": "6",
            "ip": "192.168.1.134",
            "dns": "john.local"
        }
    ],
    "id": 1
}
```


另请参阅

- 已发现的主机
- 发现检查
- 主机

来源

ui/include/classes/api/services/CDService.php 中的 CDService::get()。

报表

此类用于处理计划报表。

对象引用：

- 报表
- 用户
- 用户组

可用方法：

- report.create - 创建新的计划报表
- report.delete - 删除计划报表
- report.get - 检索计划报表
- report.update - 更新计划报表

报告对象

接下来介绍有关 `report`（报告） API 的相关内容。

报告

报告对象具有以下属性：

属性	类型	描述
reportid	ID	报告的 ID。 属性行为： - 只读 - 更新操作时必需
userid	ID	创建报告的用户的 ID。 属性行为： - 创建操作时必需
name	string	报告的唯一名称。 属性行为： - 创建操作时必需
dashboardid	ID	报告所基于的仪表板的 ID。 属性行为： - 创建操作时必需
period	integer	准备报告的周期。 可能的值： 0 - （默认）前一天； 1 - 前一周； 2 - 前一个月； 3 - 前一年。
cycle	integer	周期性重复计划。 可能的值： 0 - （默认）每天； 1 - 每周； 2 - 每月； 3 - 每年。
start_time	integer	准备发送报告的时间，以当天经过的秒数表示。 默认值：0。
weekdays	integer	发送报告的星期。 可能的位图值： 1 - 星期一； 2 - 星期二； 4 - 星期三； 8 - 星期四； 16 - 星期五； 32 - 星期六； 64 - 星期日。 这是一个位掩码字段；可以使用任意可能的位图值之和（例如，21 表示星期一、星期三和星期五）。 默认值：0。 属性行为： - 当 `cycle` 设置为”weekly” 时必需。
active_since	string	开始日期。 可能的值： 空字符串 - （默认）未指定（存储为 0）； YYYY-MM-DD 格式的具体日期（存储为当天开始时间（00:00:00）的时间戳）。

属性	类型	描述
active_till	string	结束日期。 可能的值： 空字符串 - （默认）未指定（存储为 0）； YYYY-MM-DD 格式的具体日期（存储为当天结束时间（23:59:59）的时间戳）。
subject	string	报告消息主题。
message	string	报告消息文本。
status	integer	报告是否启用。 可能的值： 0 - （默认）启用； 1 - 禁用。
description	text	报告的描述。
state	integer	报告的状态。 可能的值： 0 - （默认）报告尚未处理； 1 - 报告已生成并成功发送给所有收件人； 2 - 报告生成失败；”info” 包含错误信息； 3 - 报告已生成，但发送给部分（或全部）收件人失败；”info” 包含错误信息。 属性行为： - 只读
lastsent	timestamp	上次成功发送报告的 Unix 时间戳。 属性行为： - 只读
info	string	错误描述或其他信息。 属性行为： - 只读

用户

users 对象具有以下属性：

属性	类型	描述
userid	ID	要向其发送报告的用户 ID。 属性行为： - 必填
access_userid	ID	将以其名义生成报告的用户 ID。 0 - *(默认)* 由收件人生成报告。
exclude	integer	是否将该用户排除在邮件列表之外。 可能的值： 0 - *(默认)* 包含； 1 - 排除。

用户组

用户组对象具有以下属性：

属性	Type	描述
usrgrpid	ID	要将报告发送到的用户组 ID。 属性行为： - 必填
access_userid	ID	将代表其生成报告的用户 ID。 0 - *(默认)* 由收件人生成报告。

创建

描述

`object report.create(object/array reports)`

此方法允许您创建新的计划报告。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的计划报告。

除了标准计划报告属性外，该方法还接受以下参数。

参数	类型	描述
users	object/array	要将报告发送给的用户。 参数行为： - 如果未设置 user_groups，则为必填
user_groups	object/array	要将报告发送给的用户组。 参数行为： - 如果未设置 users，则为必填

返回值

(object) 返回一个对象，其中包含 reportids 属性下已创建的计划报告的 ID。返回的 ID 的顺序与传递的计划报告的顺序相匹配。

参考示例

创建计划报告

创建一个周报，从 2021-04-01 到 2021-08-31，每周一到周五 12:00，发送上一周的报告。

请求:

```
{
"jsonrpc": "2.0",
"method": "report.create",
"params": {
"userid": "1",
"name": "Weekly report",
"dashboardid": "1",
"period": "1",
"cycle": "1",
"start_time": "43200",
"weekdays": "31",
"active_since": "2021-04-01",
"active_till": "2021-08-31",
"subject": "Weekly report",
"message": "Report accompanying text",
"status": "1",
"description": "Report description",
"users": [{
"userid": "1",
"access_userid": "1",
"exclude": "0"
},
{
"userid": "2",
"access_userid": "0",
"exclude": "1"
}
],
```

```
"user_groups": [{
"usrgrpid": "7",
"access_userid": "0"
}]
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
```

响应：

```
{
"jsonrpc": "2.0",
"result": {
"reportids": [
"1"
]
},
"id": 1
}
```

另请参阅

- 用户
- 用户组

参考来源

CReport::create() in *ui/include/classes/api/services/CReport.php*.

删除

描述

`object report.delete(array reportids)`

此方法允许您删除计划报告。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 将要删除的规划报告 ID。

返回值

根据 `reportids` 的特性，(object) 返回一个包含已删除的规划报告 ID 的对象。

参考示例

删除多个计划报表

删除两个计划报表。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "report.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "reportids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

参考来源

CReport::delete() in *ui/include/classes/api/services/CReport.php*.

更新

描述

`object report.update(object/array reports)`

此方法允许您更新现有的计划报告。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`object/array`) 要更新的计划报告属性。

必须为每个计划报告定义 `reportid` 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除标准计划报告属性外，该方法还接受以下参数。

参数	类型	描述
users	object/array	用于替换当前分配给计划报告的用户。 参数行为： - 如果未设置 `user_groups`，则为必需
user_groups	object/array	用于替换当前分配给计划报告的用户组。 参数行为： - 如果未设置 `users`，则为必需

返回值

根据 `reportids` 的特性，(`object`) 返回一个包含已更新计划报表 ID 的对象。

参考示例

禁用计划报表

请求:

```
{
"jsonrpc": "2.0",
"method": "report.update",
"params": {
"reportid": "1",
"status": "0"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
```

响应：

```
{
"jsonrpc": "2.0",
"result": {
"reportids": ["1"]
},
"id": 1
}
```

另请参阅

- 用户
- 用户组

参考来源

CReport::update() in *ui/include/classes/api/services/CReport.php*.

获取

描述

`integer/array report.get(object parameters)`

该方法允许您根据给定参数检索计划报告。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
reportids	ID/array	仅返回具有给定报告 ID 的计划报告。
expired	boolean	如果设置为 `true`，则仅返回已过期的计划报告。
selectUsers	query	返回`users` 属性，其中包含该报告配置要发送给的用户。
selectUserGroups	query	返回`user_groups` 属性，其中包含该报告配置要发送给的用户组。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`reportid`、`name`、`status`。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回下列两种数值之一：

- 一组对象；
- 在 `countOutput` 参数应用的情况下，返回检索对象的数量。

参考示例

获取报告数据

请求:

```
{
    "jsonrpc": "2.0",
    "method": "report.get",
    "params": {
        "output": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "reportids": ["1", "2"]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "reportid": "1",
            "userid": "1",
            "name": "Weekly report",
            "dashboardid": "1",
            "period": "1",
            "cycle": "1",
            "start_time": "43200",
            "weekdays": "31",
            "active_since": "2021-04-01",
            "active_till": "2021-08-31",
            "subject": "Weekly report",
            "message": "Report accompanying text",
            "status": "1",
            "description": "Report description",
            "state": "1",
            "lastsent": "1613563219",
            "info": "",
            "users": [
                {
                    "userid": "1",
                    "access_userid": "1",
                    "exclude": "0"
                },
                {
                    "userid": "2",
                    "access_userid": "0",
                    "exclude": "1"
                }
            ],
            "user_groups": [
                {
                    "usrgrpid": "7",
                    "access_userid": "0"
                }
            ]
        },
        {
            "reportid": "2",
            "userid": "1",
            "name": "Monthly report",
            "dashboardid": "2",
            "period": "2",
            "cycle": "2",
            "start_time": "0",
            "weekdays": "0",
```

```
            "active_since": "2021-05-01",
            "active_till": "",
            "subject": "Monthly report",
            "message": "Report accompanying text",
            "status": "1",
            "description": "",
            "state": "0",
            "lastsent": "0",
            "info": "",
            "users": [
                {
                    "userid": "1",
                    "access_userid": "1",
                    "exclude": "0"
                }
            ],
            "user_groups": []
        }
    ],
    "id": 1
}
```


另请参阅

- 用户
- 用户组

参考来源

CReport::get() in *ui/include/classes/api/services/CReport.php*.

拓扑图

此类用于处理拓扑图。

对象引用：

- 拓扑图
- 拓扑图元素
  - 拓扑图元素主机
  - 拓扑图元素主机组
  - 拓扑图元素拓扑图
  - 拓扑图元素触发器
  - 拓扑图元素标签
  - 拓扑图元素 URL
- 拓扑图链接
  - 拓扑图链接触发器
  - 拓扑图链接指示器
- 拓扑图 URL
- 拓扑图用户
- 拓扑图用户组
- 拓扑图形状
- 拓扑图线条

可用方法：

- map.create - 创建新的拓扑图
- map.delete - 删除拓扑图
- map.get - 检索拓扑图
- map.update - 更新拓扑图

拓扑图对象

以下对象与 拓扑图 API 直接相关。

拓扑图

拓扑图对象具有以下属性。

属性	类型	描述
sysmapid	ID	拓扑图的 ID。 属性行为： - 只读 - 更新操作时必填
height	integer	拓扑图的高度（像素）。 属性行为： - 创建操作时必填
name	string	拓扑图名称。 属性行为： - 创建操作时必填
width	integer	拓扑图的宽度（像素）。 属性行为： - 创建操作时必填
backgroundid	ID	用作拓扑图背景的图像 ID。
background_scale	integer	是否启用背景图像按比例缩放。 可能的值： 0 - 禁用缩放； 1 - *(默认)* 启用图像缩放。
expand_macros	integer	配置拓扑图时，是否展开标签中的宏。 可能的值： 0 - *(默认)* 不展开宏； 1 - 展开宏。
expandproblem	integer	对于只有一个问题的元素，是否显示问题触发器。 可能的值： 0 - 始终显示问题数量； 1 - *(默认)* 如果只有一个问题，则显示问题触发器。
grid_align	integer	是否启用网格对齐。 可能的值： 0 - 禁用网格对齐； 1 - *(默认)* 启用网格对齐。
grid_show	integer	是否在拓扑图上显示网格。 可能的值： 0 - 不显示网格； 1 - *(默认)* 显示网格。
grid_size	integer	拓扑图网格大小（像素）。 支持的值：20、40、50、75 和 100。 默认值：50。
highlight	integer	是否启用图标高亮。 可能的值： 0 - 禁用高亮； 1 - *(默认)* 启用高亮。
iconmapid	ID	拓扑图中使用的图标映射 ID。
label_format	integer	是否启用高级标签。 可能的值： 0 - *(默认)* 禁用高级标签； 1 - 启用高级标签。

属性	类型	描述
label_location	integer	拓扑图元素标签的位置。 可能的值： 0 - *(默认)* 底部； 1 - 左侧； 2 - 右侧； 3 - 顶部。
label_string_host	string	主机元素的自定义标签。 属性行为： - 如果 `label_type_host` 设置为 “custom”，则必填
label_string_hostgroup	string	主机组元素的自定义标签。 属性行为： - 如果 `label_type_hostgroup` 设置为 “custom”，则必填
label_string_image	string	图像元素的自定义标签。 属性行为： - 如果 `label_type_image` 设置为 “custom”，则必填
label_string_map	string	拓扑图元素的自定义标签。 属性行为： - 如果 `label_type_map` 设置为 “custom”，则必填
label_string_trigger	string	触发器元素的自定义标签。 属性行为： - 如果 `label_type_trigger` 设置为 “custom”，则必填
label_type	integer	拓扑图元素标签类型。 可能的值： 0 - 标签； 1 - IP 地址； 2 - *(默认)* 元素名称； 3 - 仅状态； 4 - 无。
label_type_host	integer	主机元素的标签类型。 可能的值： 0 - 标签； 1 - IP 地址； 2 - *(默认)* 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义。
label_type_hostgroup	integer	主机组元素的标签类型。 可能的值： 0 - 标签； 2 - *(默认)* 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义。
label_type_image	integer	主机组元素的标签类型。 可能的值： 0 - 标签； 2 - *(默认)* 元素名称； 4 - 无； 5 - 自定义。

属性	类型	描述
label_type_map	integer	拓扑图元素的标签类型。 可能的值： 0 - 标签； 2 - (默认) 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义。
label_type_trigger	integer	触发器元素的标签类型。 可能的值： 0 - 标签； 2 - (默认) 元素名称； 3 - 仅状态； 4 - 无； 5 - 自定义。
markelements	integer	是否高亮最近状态发生变化的拓扑图元素。 可能的值： 0 - (默认) 不高亮元素； 1 - 高亮元素。
severity_min	integer	将在拓扑图上显示的触发器的最低严重性。 有关支持的触发器严重性列表，请参见触发器 severity 属性。
show_element_label	integer	默认如何显示元素标签。 可能的值： 0 - 始终显示； 1 - (默认) 自动隐藏。
show_link_label	integer	默认如何显示链路标签。 可能的值： 0 - 始终显示； 1 - (默认) 自动隐藏。
show_unack	integer	应如何显示问题。 可能的值： 0 - (默认) 显示所有问题的计数； 1 - 仅显示未确认问题的计数； 2 - 分别显示已确认和未确认问题的计数。
userid	ID	作为拓扑图所有者的用户 ID。
private	integer	拓扑图共享类型。 可能的值： 0 - 公共拓扑图； 1 - (默认) 私有拓扑图。
show_suppressed	integer	是否显示已抑制的问题。 可能的值： 0 - (默认) 隐藏已抑制的问题； 1 - 显示已抑制的问题。

地图元素

地图元素对象定义了显示在地图上的对象。它具有以下属性。

属性	类型	描述
selementid	ID	地图元素的 ID。 属性行为： - 只读
elements	array	元素数据对象。 属性行为： - 如果 `elementtype` 设置为 “主机”、“地图”、“触发器” 或 “主机组”，则为必需
elementtype	integer	地图元素的类型。 可能的值： 0 - 主机； 1 - 地图； 2 - 触发器； 3 - 主机组； 4 - 图像。 属性行为： - 必需
iconid_off	ID	用于以默认状态显示该元素的图像 ID。 属性行为： - 必需
areatype	integer	应如何显示主机组中的各个主机。 可能的值： 0 - *(默认)* 主机组元素将占据整个地图； 1 - 主机组元素将具有固定大小。
elementsubtype	integer	应如何在地图上显示主机组元素。 可能的值： 0 - *(默认)* 将主机组显示为单个元素； 1 - 分别显示组中的每个主机。
evaltype	integer	地图元素标签过滤条件的评估方法。 可能的值： 0 - *(默认)* And/Or； 2 - Or。
height	integer	固定大小主机组元素的高度（像素）。 默认值：200。
iconid_disabled	ID	用于显示已禁用地图元素的图像 ID。 属性行为： - 如果 `elementtype` 设置为 “主机”、“地图”、“触发器” 或 “主机组”，则支持
iconid_maintenance	ID	用于显示维护中的地图元素的图像 ID。 属性行为： - 如果 `elementtype` 设置为 “主机”、“地图”、“触发器” 或 “主机组”，则支持
iconid_on	ID	用于显示存在问题的地图元素的图像 ID。 属性行为： - 如果 `elementtype` 设置为 “主机”、“地图”、“触发器” 或 “主机组”，则支持
label	string	元素的标签。

属性	类型	描述
label_location	integer	地图元素标签的位置。 可能的值： -1 - *(默认)* 默认位置； 0 - 底部； 1 - 左侧； 2 - 右侧； 3 - 顶部。
permission	integer	权限级别类型。 可能的值： -1 - 无； 2 - 只读； 3 - 读写。
show_label	integer	元素标签的显示方式。 可能的值： -1 - *(默认)* 地图默认； 0 - 始终显示； 1 - 自动隐藏。
sysmapid	ID	该元素所属地图的 ID。 属性行为： - 只读
urls	array	地图元素 URL。 地图元素 URL 对象的详细说明见下文。
use_iconmap	integer	是否必须对主机元素使用图标映射。 可能的值： 0 - 不使用图标映射； 1 - *(默认)* 使用图标映射。
viewtype	integer	主机组元素放置算法。 可能的值： 0 - *(默认)* 网格。
width	integer	固定大小主机组元素的宽度（像素）。 默认值：200。
x	integer	元素的 X 坐标（像素）。 默认值：0。
y	integer	元素的 Y 坐标（像素）。 默认值：0。
zindex	integer	用于对地图元素进行排序的值（z-index）。 默认值：0。

地图元素主机

地图元素主机对象定义一个主机元素。

属性	Type	描述
hostid	ID	主机的 ID。

地图元素主机组

地图元素主机组对象定义了一个主机组元素。

属性	Type	说明
groupid	ID	主机组的 ID。

地图元素 Map

地图元素 Map 对象定义了一个地图元素。

属性	类型	描述
sysmapid	ID	地图的 ID。

地图元素触发器

地图元素 Trigger 对象定义一个或多个触发器元素。

属性	Type	说明
triggerid	ID	触发器的 ID。

地图元素标签

地图元素标签对象具有以下属性。

属性	类型	描述
tag	string	地图元素标签名称。 属性行为： - 必填
operator	integer	地图元素标签条件运算符。 可能的值： 0 - (默认) 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
value	string	地图元素标签值。

地图元素 URL

地图元素 URL 对象定义了一个可点击的链接，该链接将可用于特定的地图元素。它具有以下属性：

属性	类型	描述
sysmapelementurlid	ID	地图元素 URL 的 ID。 属性行为： - 只读
name	string	链接标题。 属性行为： - 必需
url	string	链接 URL。 属性行为： - 必需
selementid	ID	该 URL 所属地图元素的 ID。

地图链接

地图链接对象定义了两个地图元素之间的链接。它具有以下属性。

属性	类型	描述
linkid	ID	地图链接的 ID。 属性行为： - 只读
sysmapid	ID	该链接所属地图的 ID。
selementid1	ID	链接一端的第一个地图元素的 ID。 属性行为： - 必填
selementid2	ID	链接另一端的第一个地图元素的 ID。 属性行为： - 必填
drawtype	integer	链接线条的绘制样式。 可能的值： 0 - *(默认)* 实线； 2 - 粗线； 3 - 点线； 4 - 虚线。
color	string	线条颜色，使用十六进制颜色代码表示。 默认值：000000。
label	string	链接标签。
show_label	integer	如何显示链接标签。 可能的值： -1 - *(默认)* 地图默认值； 0 - 始终显示； 1 - 自动隐藏。
indicator_type	integer	选择链接指示器类型。 可能的值： 0 - *(默认)* 静态链接； 1 - 触发器； 2 - 监控项值。
linktriggers	array	用作链接状态指示器的地图链接触发器。 地图链接触发器对象的详细说明见下文。 属性行为： - 如果 `indicator_type` 设置为 “触发器”(1)，则为 必填。
itemid	ID	监控项的 ID。 属性行为： - 如果 `indicator_type` 设置为 “监控项值”(2)，则为 必填。
highlights	array	用作链接状态指示器的地图链接高亮。 地图链接指示器对象的详细说明见下文。 属性行为： - 如果 `indicator_type` 设置为 “监控项值”(2)，则为 必填。
thresholds	array	用作链接状态指示器的地图链接阈值。 地图链接指示器对象的详细说明见下文。 属性行为： - 如果 `indicator_type` 设置为 “监控项值”(2)，则为 必填。

属性	类型	描述
permission	integer	权限级别类型。 可能的值： -1 - 无； 2 - 只读； 3 - 读写。

地图链接触发器

地图链接触发器对象根据触发器的状态定义地图链接状态指示器。它具有以下属性：

属性	类型	描述
triggerid	ID	用作链接指示器的触发器 ID。 属性行为： - 必填
color	string	指示器颜色，使用十六进制颜色代码表示。 默认值：DD0000。
drawtype	integer	指示器绘制样式。 可能的值： 0 - (默认) 线； 2 - 粗线； 3 - 点线； 4 - 虚线。

地图链接指示器

地图链接指示器对象基于监控项值定义地图链接状态指示器。它具有以下属性：

属性	类型	说明
pattern	string	用于比较的正则表达式。仅适用于高亮。
sortorder	integer	用于定义高亮的排序顺序。仅适用于高亮。
threshold	string	用于比较的数值。仅适用于阈值。
drawtype	integer	指示器绘制样式。 可能的值： 0 - (默认) 线； 2 - 粗线； 3 - 点线； 4 - 虚线。
color	string	指示器颜色，以十六进制颜色代码表示。 默认值：DD0000。 属性行为： - 必需

地图 URL

地图 URL 对象定义了一个可点击链接，该链接将对地图上特定类型的所有元素可用。它具有以下属性：

属性	Type	说明
sysmapurlid	ID	地图 URL 的 ID。 属性行为： - 只读

属性	Type	说明
name	string	链接标题。 属性行为： - 必需
url	string	链接 URL。 属性行为： - 必需
elementtype	integer	URL 可用于哪种类型的地图元素。 支持的类型列表请参见地图元素 type 属性。 默认值：0。
sysmapid	ID	该 URL 所属地图的 ID。

地图用户

基于用户的地图权限列表。它具有以下属性：

属性	类型	描述
sysmapuserid	ID	地图用户的 ID。 属性行为： - 只读
userid	ID	用户的 ID。 属性行为： - 必需
permission	integer	权限级别类型。 可能的值： 2 - 只读； 3 - 读写。 属性行为： - 必需

地图用户组

基于用户组的地图权限列表。它具有以下属性：

属性	Type	描述
sysmapusrgrpid	ID	地图用户组的 ID。 Property behavior： - *read-only*
usrgrpid	ID	用户组的 ID。 Property behavior： - *required*
permission	integer	权限级别类型。 可能的值： 2 - 只读； 3 - 读写。 Property behavior： - *required*

地图形状

地图形状对象定义了在地图上显示的几何形状（带文本或不带文本）。它具有以下属性：

属性	类型	描述
sysmap_shapeid	ID	地图形状元素的 ID。 属性行为: - 只读
type	integer	地图形状元素的类型。 可能的值： 0 - 矩形； 1 - 椭圆。 创建新形状时，此属性为必填。 属性行为: - 必需
x	integer	形状的 X 坐标，以像素为单位。 默认值：0。
y	integer	形状的 Y 坐标，以像素为单位。 默认值：0。
width	integer	形状的宽度，以像素为单位。 默认值：200。
height	integer	形状的高度，以像素为单位。 默认值：200。
text	string	形状的文本。
font	integer	形状内文本的字体。 可能的值： 0 - Georgia, serif 1 - ”Palatino Linotype”, ”Book Antiqua”, Palatino, serif 2 - ”Times New Roman”, Times, serif 3 - Arial, Helvetica, sans-serif 4 - ”Arial Black”, Gadget, sans-serif 5 - ”Comic Sans MS”, cursive, sans-serif 6 - Impact, Charcoal, sans-serif 7 - ”Lucida Sans Unicode”, ”Lucida Grande”, sans-serif 8 - Tahoma, Geneva, sans-serif 9 - ”Trebuchet MS”, Helvetica, sans-serif 10 - Verdana, Geneva, sans-serif 11 - ”Courier New”, Courier, monospace 12 - ”Lucida Console”, Monaco, monospace 默认值：9。
font_size	integer	字体大小，以像素为单位。 默认值：11。
font_color	string	字体颜色。 默认值：000000。
text_halign	integer	文本的水平对齐方式。 可能的值： 0 - 居中； 1 - 左对齐； 2 - 右对齐。 默认值：0。

属性	类型	描述
text_valign	integer	文本的垂直对齐方式。 可能的值： 0 - 中间； 1 - 顶部； 2 - 底部。 默认值：0。
border_type	integer	边框类型。 可能的值： 0 - 无； 1 - ——————； 2 - ┅； 3 - – – –。 默认值：0。
border_width	integer	边框宽度，以像素为单位。 默认值：0。
border_color	string	边框颜色。 默认值：000000。
background_color	string	背景颜色（填充颜色）。 默认值：(empty)。
zindex	integer	用于对所有形状和线条进行排序的值（z-index）。 默认值：0。

地图连线

地图连线对象定义了显示在地图上的一条线。它具有以下属性：

属性	类型	描述
sysmap_shapeid	ID	地图形状元素的 ID。 属性行为： - 只读
x1	integer	线条第 1 个点的 X 坐标，单位为像素。 默认值：0。
y1	integer	线条第 1 个点的 Y 坐标，单位为像素。 默认值：0。
x2	integer	线条第 2 个点的 X 坐标，单位为像素。 默认值：200。
y2	integer	线条第 2 个点的 Y 坐标，单位为像素。 默认值：200。
line_type	integer	线条类型。 可能的值： 0 - 无； 1 - ——————； 2 - ┅； 3 - – – –。 默认值：0。

属性	类型	描述
line_width	integer	线条宽度，单位为像素。 默认值：0。
line_color	string	线条颜色。 默认值：000000。
zindex	integer	用于对所有形状和线条进行排序的值（z-index）。 默认值：0。

创建

描述

`object map.create(object/array maps)`

此方法允许您创建新的地图。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object/array)` 要创建的地图。

除了标准地图属性之外，该方法还接受以下参数。

参数	Type	描述
links	array	要在地图上创建的地图链接。
selements	array	要在地图上创建的地图元素。
urls	array	要在地图上创建的地图 URL。
users	array	要在地图上创建的地图用户共享。
userGroups	array	要在地图上创建的地图用户组共享。
shapes	array	要在地图上创建的地图形状。
lines	array	要在地图上创建的地图线条。

> **Note:**
> 要创建地图链接，您需要先将某个地图元素的 `selementid` 设置为任意值，然后在链接的 `selementid1` 或 `selementid2` 属性中使用该值来引用此元素。创建该元素时，此值将被替换为 Zabbix 生成的正确 ID。请参见示例。

返回值

（对象）返回一个对象，该对象包含在"sysmapid" 属性下创建的拓扑图的 id。返回 id 的顺序与传递的拓扑图的顺序相匹配。

案例

创建一个空拓扑图

创建一个没有元素的拓扑图。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map",
        "width": 600,
        "height": 600
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

创建主机映射

创建一个包含两个主机元素和它们之间链接的拓扑图。请注意在拓扑图链接对象中使用临时”selementid1” 和”selementid2” 值来引用拓扑图元素。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "selementid": "1",
                "elements": [
                    {"hostid": "1033"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
            },

            {
                "selementid": "2",
                "elements": [
                    {"hostid": "1037"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
            }
        ],
        "links": [
            {
                "selementid1": "1",
                "selementid2": "2"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

创建一个触发器拓扑图

创建一个包含两个触发器元素的拓扑图。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Trigger map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "elements": [
                    {"triggerid": "12345"},
                    {"triggerid": "67890"}
                ],
                "elementtype": 2,
                "iconid_off": "2"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

拓扑图分享

创建具有两种共享类型（用户和用户组）的拓扑图。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map sharing",
        "width": 600,
        "height": 600,
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

拓扑图形状

创建一个带有主题的拓扑图。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "shapes": [
            {
                "type": 0,
                "x": 0,
                "y": 0,
                "width": 600,
                "height": 11,
                "text": "{MAP.NAME}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

拓扑图线

创建拓扑图线。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map API lines",
        "width": 500,
        "height": 500,
        "lines": [
            {
                "x1": 30,
                "y1": 10,
```

```
                "x2": 100,
                "y2": 50,
                "line_type": 1,
                "line_width": 10,
                "line_color": "009900"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "11"
        ]
    },
    "id": 1
}
```

另请参见

- 地图元素
- 地图链接
- 地图 URL
- 地图用户
- 地图用户组
- 地图形状
- 地图线条

来源

CMap::create() in *ui/include/classes/api/services/CMap.php*.

删除

描述

`object map.delete(array mapIds)`

此方法允许您删除地图。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 要删除的拓扑图的 id 列表.

返回值

返回包含"sysmapid" 属性下的已删除拓扑图的 IDs 的对象。

案例

删除多个拓扑图

删除 2 个拓扑图。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.delete",
    "params": [
        "12",
```

```
            "34"
        ],
        "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "12",
            "34"
        ]
    },
    "id": 1
}
```

来源

CMap::delete() in *ui/include/classes/api/services/CMap.php*.

更新

描述

`object map.update(object/array maps)`

此方法允许您更新现有的映射。

> **Note:**
> 任何类型的用户都可以使用此方法。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object/array)` 要更新的拓扑图属性映射。

必须为每个拓扑图定义 `mapid` 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准拓扑图属性外，该方法还接受以下参数。

参数	类型	说明
links	array	用于替换现有链接的拓扑图链接。
selements	array	用于替换现有元素的拓扑图元素。
urls	array	用于替换现有 URL 的拓扑图 URL。
users	array	用于替换现有元素的拓扑图用户共享。
userGroups	array	用于替换现有元素的拓扑图用户组共享。
shapes	array	用于替换现有形状的拓扑图形状。
lines	array	用于替换现有线条的拓扑图线条。

> **Note:**
> 要在新的拓扑图元素之间创建拓扑图链接，您需要将元素的 `selementid` 设置为任意值，然后使用该值在链接的 `selementid1` 或 `selementid2` 属性中引用此元素。创建元素时，该值将被替换为 Zabbix 生成的正确 ID。请参见 map.create 的示例。

返回值

`(object)` 返回一个对象，该对象包含 `sysmapids` 属性下更新的映射 ID。

案例

调整拓扑图大小

将拓扑图大小更改为 1200x1200 像素。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "8",
        "width": 1200,
        "height": 1200
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

改变拓扑图的拥有者

仅适用于管理员和超级管理员。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "9",
        "userid": "1"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

另请参阅

- 地图元素
- 地图链接
- 地图 URL
- 地图用户
- 地图用户组
- 地图形状
- 地图线条

来源

CMap::update() in *ui/include/classes/api/services/CMap.php*.

获取

描述

`integer/array map.get(object parameters)`

该方法允许您根据给定参数检索地图。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
sysmapids	ID/array	仅返回具有给定 ID 的地图。
userids	ID/array	仅返回属于给定用户 ID 的地图。
expandUrls	flag	将全局地图 URL 添加到相应的地图元素，并展开所有地图元素 URL 中的宏。
selectIconMap	query	返回一个`iconmap` 属性，其中包含地图上使用的图标映射。
selectLinks	query	返回一个`links` 属性，其中包含元素之间的地图链接。
selectSelements	query	返回一个`selements` 属性，其中包含地图元素。
selectUrls	query	返回一个`urls` 属性，其中包含地图 URL。
selectUsers	query	返回一个`users` 属性，其中包含与该地图共享的用户。
selectUserGroups	query	返回一个`userGroups` 属性，其中包含与该地图共享的用户组。
selectShapes	query	返回一个`shapes` 属性，其中包含地图形状。
selectLines	query	返回一个`lines` 属性，其中包含地图线条。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`name`、`width`、`height`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(整型/数组) 返回其中之一：

- 一个对象数组；
- 如果使用 `countOutput` 参数, 查询对象的数量.

案例

检索地图

检索地图 “3” 的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "map.get",
    "params": {
        "output": "extend",
        "selectSelements": "extend",
        "selectLinks": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "selectShapes": "extend",
```

```
        "selectLines": "extend",
        "sysmapids": "3"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "selements": [
                {
                    "selementid": "10",
                    "sysmapid": "3",
                    "elementtype": "4",
                    "evaltype": "0",
                    "iconid_off": "1",
                    "iconid_on": "0",
                    "label": "Zabbix server",
                    "label_location": "3",
                    "x": "11",
                    "y": "141",
                    "iconid_disabled": "0",
                    "iconid_maintenance": "0",
                    "elementsubtype": "0",
                    "areatype": "0",
                    "width": "200",
                    "height": "200",
                    "viewtype": "0",
                    "use_iconmap": "1",
                    "show_label": "-1",
                    "zindex":"0",
                    "elements": [],
                    "urls": [],
                    "tags": [
                        {
                            "tag": "service",
                            "value": "mysqld",
                            "operator": "0"
                        }
                    ]
                },
                {
                    "selementid": "11",
                    "sysmapid": "3",
                    "elementtype": "4",
                    "evaltype": "0",
                    "iconid_off": "1",
                    "iconid_on": "0",
                    "label": "Web server",
                    "label_location": "3",
                    "x": "211",
                    "y": "191",
                    "iconid_disabled": "0",
                    "iconid_maintenance": "0",
                    "elementsubtype": "0",
                    "areatype": "0",
                    "width": "200",
                    "height": "200",
                    "viewtype": "0",
                    "use_iconmap": "1",
```

```
                "show_label": "0",
                "zindex":"0",
                "elements": [],
                "urls": [],
                "tags": []
            },
            {
                "selementid": "12",
                "sysmapid": "3",
                "elementtype": "0",
                "evaltype": "0",
                "iconid_off": "185",
                "iconid_on": "0",
                "label": "{HOST.NAME}\r\n{HOST.CONN}",
                "label_location": "0",
                "x": "111",
                "y": "61",
                "iconid_disabled": "0",
                "iconid_maintenance": "0",
                "elementsubtype": "0",
                "areatype": "0",
                "width": "200",
                "height": "200",
                "viewtype": "0",
                "use_iconmap": "0",
                "show_label": "1",
                "zindex":"0",
                "elements": [
                    {
                        "hostid": "10084"
                    }
                ],
                "urls": [],
                "tags": []
            }
        ],
        "links": [
            {
                "linkid": "23",
                "sysmapid": "3",
                "selementid1": "10",
                "selementid2": "11",
                "drawtype": "0",
                "color": "00CC00",
                "label": "",
                "show_label": "1",
                "indicator_type": "0",
                "itemid": "0",
                "linktriggers": [],
                "thresholds": [],
                "highlights": []
            }
        ],
        "users": [
            {
                "sysmapuserid": "1",
                "userid": "2",
                "permission": "2"
            }
        ],
        "userGroups": [
            {
```

```
                "sysmapusrgrpid": "1",
                "usrgrpid": "7",
                "permission": "2"
            }
        ],
        "shapes":[
            {
                "sysmap_shapeid":"1",
                "type":"0",
                "x":"0",
                "y":"0",
                "width":"680",
                "height":"15",
                "text":"{MAP.NAME}",
                "font":"9",
                "font_size":"11",
                "font_color":"000000",
                "text_halign":"0",
                "text_valign":"0",
                "border_type":"0",
                "border_width":"0",
                "border_color":"000000",
                "background_color":"",
                "zindex":"0"
            }
        ],
        "lines":[
            {
                "sysmap_shapeid":"2",
                "x1": 30,
                "y1": 10,
                "x2": 100,
                "y2": 50,
                "line_type": 1,
                "line_width": 10,
                "line_color": "009900",
                "zindex":"1"
            }
        ],
        "sysmapid": "3",
        "name": "Local network",
        "width": "400",
        "height": "400",
        "backgroundid": "0",
        "background_scale": "1",
        "label_type": "2",
        "label_location": "3",
        "show_element_label": "0",
        "show_link_label": "1",
        "highlight": "1",
        "expandproblem": "1",
        "markelements": "0",
        "show_unack": "0",
        "grid_size": "50",
        "grid_show": "1",
        "grid_align": "1",
        "label_format": "0",
        "label_type_host": "2",
        "label_type_hostgroup": "2",
        "label_type_trigger": "2",
        "label_type_map": "2",
        "label_type_image": "2",
```

```
            "label_string_host": "",
            "label_string_hostgroup": "",
            "label_string_trigger": "",
            "label_string_map": "",
            "label_string_image": "",
            "iconmapid": "0",
            "expand_macros": "0",
            "severity_min": "0",
            "userid": "1",
            "private": "1",
            "show_suppressed": "1"
        }
    ],
    "id": 1
}
```


另请参见

- 图标映射
- 地图元素
- 地图链接
- 地图 URL
- 地图用户
- 地图用户组
- 地图形状
- 地图线条

来源

CMap::get() in *ui/include/classes/api/services/CMap.php*.

数据清理

该类被设计用于和数据清理一起使用。

对象参考：

- 数据清理

可用的方法：

- housekeeping.get - 检索数据清理
- housekeeping.update - 更新数据清理

数据清理对象

以下对象与 housekeepingAPI 直接相关。

数据保留清理

设置对象具有以下属性。

属性	类型	描述
hk_events_mode	integer	启用事件和告警的内部数据保留清理。 可能的值： 0 - 禁用； 1 - （默认）启用。
hk_events_trigger	string	触发器数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：365d。
hk_events_service	string	服务数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：1d。

属性	类型	描述
hk_events_internal	string	内部数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：1d。
hk_events_discovery	string	网络发现数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：1d。
hk_events_autoreg	string	自动注册数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：1d。
hk_services_mode	integer	启用服务的内部数据保留清理。 可能的值： 0 - 禁用； 1 - （默认）启用。
hk_services	string	服务数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：365d。
hk_audit_mode	integer	启用审计的内部数据保留清理。 可能的值： 0 - 禁用； 1 - （默认）启用。
hk_audit	string	审计数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：31d。
hk_sessions_mode	integer	启用会话的内部数据保留清理。 可能的值： 0 - 禁用； 1 - （默认）启用。
hk_sessions	string	会话数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：31d。
hk_history_mode	integer	启用历史数据的内部数据保留清理。 可能的值： 0 - 禁用； 1 - （默认）启用。
hk_history_global	integer	覆盖监控项历史数据周期。 可能的值： 0 - 不覆盖； 1 - （默认）覆盖。
hk_history	string	历史数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：31d。
hk_trends_mode	integer	启用趋势数据的内部数据保留清理。 可能的值： 0 - 禁用； 1 - （默认）启用。

属性	类型	描述
hk_trends_global	integer	覆盖监控项趋势数据周期。 可能的值： 0 - 不覆盖； 1 - （默认）覆盖。
hk_trends	string	趋势数据存储周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：365d。
db_extension	string	配置标志 DB extension。如果此标志设置为”timescaledb”，则服务器会更改其数据保留清理和监控项删除行为。 属性行为： - 只读
compression_availability	integer	数据库（或其扩展）是否支持数据压缩。 可能的值： 0 - 不可用； 1 - 可用。 属性行为： - 只读
compression_status	integer	为历史数据和趋势数据启用 TimescaleDB 压缩。 可能的值： 0 - （默认）关闭； 1 - 开启。
compress_older	string	压缩早于指定周期的历史数据和趋势数据记录。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：7d。

更新

描述

`object housekeeping.update(object housekeeping)`

此方法允许您更新现有的 housekeeping 设置。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 要更新的数据清理属性。

返回值

`(array)` 返回包含更新参数名称的数组。

示例

请求：

```
{
    "jsonrpc": "2.0",
    "method": "housekeeping.update",
    "params": {
        "hk_events_mode": "1",
        "hk_events_trigger": "200d",
        "hk_events_internal": "2d",
        "hk_events_discovery": "2d"
    },
```

```
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        "hk_events_mode",
        "hk_events_trigger",
        "hk_events_internal",
        "hk_events_discovery"
    ],
    "id": 1
}
```

来源

CHousekeeping::update() 在 *ui/include/classes/api/services/CHousekeeping.php*.

获取

描述

`object housekeeping.get(object parameters)`

该方法允许你根据给定参数检索 housekeeping 对象。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法仅支持一个参数。

参数	Type	描述
output	query	该参数在参考注释中进行了说明。

返回值

`(object)` 返回数据清理对象。

示例

请求：

```
{
    "jsonrpc": "2.0",
    "method": "housekeeping.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "hk_events_mode": "1",
        "hk_events_trigger": "365d",
        "hk_events_service": "1d",
        "hk_events_internal": "1d",
        "hk_events_discovery": "1d",
```

```
        "hk_events_autoreg": "1d",
        "hk_services_mode": "1",
        "hk_services": "365d",
        "hk_audit_mode": "1",
        "hk_audit": "31d",
        "hk_sessions_mode": "1",
        "hk_sessions": "365d",
        "hk_history_mode": "1",
        "hk_history_global": "0",
        "hk_history": "31d",
        "hk_trends_mode": "1",
        "hk_trends_global": "0",
        "hk_trends": "365d",
        "db_extension": "",
        "compression_status": "0",
        "compress_older": "7d"
    },
    "id": 1
}
```


来源

CHousekeeping ::get() 在 *ui/include/classes/api/services/CHousekeeping.php*.

服务

此类用于处理 IT 基础架构/业务服务。

对象引用：

- Service
- Status rule
- Service tag
- Service alarm
- Problem tag

可用方法：

- service.create - 创建新服务
- service.delete - 删除服务
- service.get - 检索服务
- service.update - 更新服务

服务对象

以下对象与 `service` API 直接关联

Service

service 对象具有以下属性。

Property	Type	Description
serviceid	ID	service 的 ID。 Property behavior: - *read-only* - *required* for update operations

Property	Type	Description
algorithm	integer	状态计算规则。仅在存在子 service 时适用。 Possible values: 0 - 将状态设置为 OK； 1 - 如果所有子项都有问题，则取最严重的状态； 2 - 取子 service 中最严重的状态。 Property behavior: - *required* for create operations
name	string	service 的名称。 Property behavior: - *required* for create operations
sortorder	integer	用于排序的 service 位置。 Possible values: 0-999. Property behavior: - *required* for create operations
weight	integer	service 权重。 Possible values: 0-1000000. Default: 0.
propagation_rule	integer	状态传播规则。 Possible values: 0 - *(default)* 按原样传播 service 状态 - 不做任何更改； 1 - 将传播后的状态按给定的 `propagation_value` 提高（提高 1 到 5 个严重级别）； 2 - 将传播后的状态按给定的 `propagation_value` 降低（降低 1 到 5 个严重级别）； 3 - 忽略此 service - 状态根本不会传播到父 service； 4 - 使用给定的 `propagation_value` 设置固定的 service 状态。 Property behavior: - *required* if `propagation_value` is set
propagation_value	integer	状态传播值。 Possible values if `propagation_rule` is set to ”0” or ”3”: 0 - 未分类。 Possible values if `propagation_rule` is set to ”1” or ”2”: 1 - 信息； 2 - 警告； 3 - 一般； 4 - 严重； 5 - 灾难。 Possible values if `propagation_rule` is set to ”4”: -1 - OK； 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 严重； 5 - 灾难。 Property behavior: - *required* if `propagation_rule` is set

Property	Type	Description
status	integer	service 处于 OK 还是问题状态。 如果 service 处于问题状态，`status` 等于最严重问题的严重级别，或处于问题状态的子 service 的最高状态。 如果 service 处于 OK 状态，`status` 等于 -1。 Property behavior: - *read-only*
description	string	service 的描述。
uuid	string	通用唯一标识符，用于将导入的 service 链接到已存在的 service。若未提供，则自动生成。
created_at	integer	service 创建时的 Unix 时间戳。
readonly	integer	对 service 的访问权限。 Possible values: 0 - 可读写； 1 - 只读。 Property behavior: - *read-only*

状态规则

状态规则对象具有以下属性。

属性	类型	描述
type	integer	用于设置（新状态）状态的条件。 可能的值： 0 - 如果至少有 (N) 个子服务处于（状态）状态或更高； 1 - 如果至少有 (N%) 的子服务处于（状态）状态或更高； 2 - 如果少于 (N) 个子服务处于（状态）状态或更低； 3 - 如果少于 (N%) 的子服务处于（状态）状态或更低； 4 - 如果处于（状态）状态或更高的子服务权重至少为 (W)； 5 - 如果处于（状态）状态或更高的子服务权重至少为 (N%)； 6 - 如果处于（状态）状态或更低的子服务权重少于 (W)； 7 - 如果处于（状态）状态或更低的子服务权重少于 (N%)。 其中： - N (W) 为 `limit_value`； - （状态）为 `limit_status`； - （新状态）为 `new_status`。 属性行为： - 必需
limit_value	integer	限制值。 可能的值： - 对于 N 和 W：1-100000； - 对于 N%：1-100。 属性行为： - 必需

属性	类型	描述
limit_status	integer	限制状态。 可能的值： -1 - 正常； 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。 属性行为： - 必需
new_status	integer	新状态值。 可能的值： 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 严重； 5 - 灾难。 属性行为： - 必需

服务标签

服务标签对象具有以下属性。

属性	Type	说明
tag	string	服务标签名称。 属性行为： - 必填
value	string	服务标签值。

服务告警

Note:
无法通过 Zabbix API 直接创建、更新或删除服务告警。

服务告警对象表示服务状态的变更。它具有以下属性。

Property	Type	Description
clock	timestamp	服务状态变更发生的时间。
value	integer	服务的状态。 有关可能值的列表，请参阅service `status` property。

问题标签

问题标签允许将服务与问题事件关联起来。问题标签对象具有以下属性。

Property	Type	Description
tag	string	问题标签名称。 Property behavior: - *required*

Property	Type	Description
operator	integer	映射条件operator。 Possible values: 0 - *(default)* Equals; 2 - Contains.
value	string	问题标签值。

创建

描述

`object service.create(object/array services)`

此方法允许您创建新的服务。

Note:
此方法适用于任何类型的用户。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(`object/array`) 要创建的服务。

除了标准服务属性之外，该方法还接受以下参数。

参数	类型	描述
children	array	要关联到该服务的子服务。 子服务必须仅定义 `serviceid` 属性。
parents	array	要关联到该服务的父服务。 父服务必须仅定义 `serviceid` 属性。
tags	array	要为该服务创建的服务标签。
problem_tags	array	要为该服务创建的问题标签。
status_rules	array	要为该服务创建的状态规则。

返回值

（对象）返回一个 `serviceids` 属性包含被创建服务 ID 的对象。返回的 ID 顺序与传入服务的顺序一致。

示例

创建服务

创建将切换到问题状态的服务（如果至少有）子服务问题。请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.create",
    "params": {
        "name": "Server 1",
        "algorithm": 1,
        "sortorder": 1
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
```

```
    "id": 1
}
```

来源

CService::create() in *ui/include/classes/api/services/CService.php*.

删除

描述

`object service.delete(array serviceIds)`

此方法允许您删除服务。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

（数组）要删除的服务 ID。

返回值

（对象）返回一个 `serviceids` 属性包含被删除服务 ID 的对象。

示例

删除多个服务

删除两个服务请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.delete",
    "params": [
        "4",
        "5"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

来源

CService::delete() in *ui/include/classes/api/services/CService.php*.

更新

描述

`object service.update(object/array services)`

此方法允许您更新现有服务。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的服务属性。

必须为每个服务定义 serviceid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他属性将保持不变。

除了标准服务属性之外，该方法还接受以下参数。

参数	类型	说明
children	array	用于替换当前子服务的子服务。 子服务必须仅定义 serviceid 属性。
parents	array	用于替换当前父服务的父服务。 父服务必须仅定义 serviceid 属性。
tags	array	用于替换当前服务标签的服务标签。
problem_tags	array	用于替换当前问题标签的问题标签。
status_rules	array	用于替换当前状态规则的状态规则。

返回值

(对象) 返回一个 serviceids 属性包含了被更新服务 ID 的对象。

示例

设置服务的父服务

使 ID 为 "3" 的服务成为 ID 为 "5" 的服务的父服务。请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "5",
        "parents": [
            {
                "serviceid": "3"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

增加关机计划

给 ID 为 "4" 的服务增加每周一 22:00 到周二 10:00 的关机计划。请求:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "4",
        "times": [
            {
```

```
                "type": "1",
                "ts_from": "165600",
                "ts_to": "201600"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4"
        ]
    },
    "id": 1
}
```

来源

CService::update() in *ui/include/classes/api/services/CService.php*.

获取

描述

`integer/array service.get(object parameters)`

该方法允许你根据给定参数检索服务。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
serviceids	ID/array	仅返回具有给定 ID 的服务。
parentids	ID/array	仅返回链接到给定父服务的服务。
deep_parentids	flag	返回所有直接和间接子服务。与 `parentids` 一起使用。
childids	ID/array	仅返回链接到给定子服务的服务。
evaltype	integer	标签评估方法。 可能的值： 0 - *(默认)* And/Or； 2 - Or。
tags	object/array	仅返回具有给定标签的服务。 格式：`[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有服务。 可能的operator值： 0 - *(默认)* 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。

参数	类型	描述
problem_tags	object/array	仅返回具有给定问题标签的服务。 格式：[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]。 空数组将返回所有服务。 可能的operator值： 0 - *(默认)* 包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
without_problem_tags	flag	仅返回不带问题标签的服务。
slaids	ID/array	仅返回链接到特定 SLA 的服务。
selectChildren	query	返回一个 children 属性，其中包含子服务。 支持 count。
selectParents	query	返回一个 parents 属性，其中包含父服务。 支持 count。
selectTags	query	返回一个tags 属性，其中包含服务标签。 支持 count。
selectProblemEvents	query	返回一个 problem_events 属性，其中包含问题事件对象数组。 问题事件对象具有以下属性： eventid - (ID) 事件 ID； severity - (string) 当前事件严重性； name - (string) 已恢复事件名称。 支持 count。
selectProblemTags	query	返回一个problem_tags 属性，其中包含问题标签。 支持 count。
selectStatusRules	query	返回一个status_rules 属性，其中包含状态规则。 支持 count。
selectStatusTimeline	object/array	返回一个 status_timeline 属性，其中包含给定时间段内的服务状态变更。 格式：[{"period_from": "<period_from>", "period_to": "<period_to>"}, ...] - 其中 period_from 为你感兴趣时间段的开始日期（包含；integer 时间戳），period_to 为结束日期（不包含；integer 时间戳）。 返回一个条目数组，其中包含 start_value 属性以及一个alarms 数组，用于表示指定时间段内的状态变更。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：serviceid、name、status、sortorder、created_at。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	

参数	类型	描述
startSearch	boolean	

返回值

（整型/数组）返回其中之一：- 一个对象数组；- 如果使用 `countOutput` 参数，被检索对象的数量。

示例

检索所有服务

检索所有服务及其依赖关系的全部数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "service.get",
    "params": {
        "output": "extend",
        "selectChildren": "extend",
        "selectParents": ["serviceid", "name"]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "serviceid": "1",
            "name": "Zabbix cluster",
            "status": "-1",
            "algorithm": "2",
            "sortorder": "0",
            "weight": "0",
            "propagation_rule": "0",
            "propagation_value": "0",
            "description": "",
            "uuid": "8d77bd91b62347e4b79382912eb5df95",
            "created_at": "1761742392",
            "readonly": false,
            "parents": [],
            "children": [
                {
                    "serviceid": "2",
                    "name": "Zabbix server node 1",
                    "status": "-1",
                    "algorithm": "2",
                    "sortorder": "0",
                    "weight": "0",
                    "propagation_rule": "0",
                    "propagation_value": "0",
                    "description": "",
                    "uuid": "195911d26d7f4e218d6217079bcd5929",
                    "created_at": "1761742623",
                    "readonly": false
                },
                {
                    "serviceid": "3",
                    "name": "Zabbix server node 2",
                    "status": "-1",
                    "algorithm": "2",
```

```
                    "sortorder": "0",
                    "weight": "0",
                    "propagation_rule": "0",
                    "propagation_value": "0",
                    "description": "",
                    "uuid": "9fc659a30fe244f690dff25fc2a9db5c",
                    "created_at": "1761742654",
                    "readonly": false
                }
            ]
        },
        {
            "serviceid": "2",
            "name": "Zabbix server node 1",
            "status": "-1",
            "algorithm": "2",
            "sortorder": "0",
            "weight": "0",
            "propagation_rule": "0",
            "propagation_value": "0",
            "description": "",
            "uuid": "195911d26d7f4e218d6217079bcd5929",
            "created_at": "1761742623",
            "readonly": false,
            "parents": [
                {
                    "serviceid": "1",
                    "name": "Zabbix cluster"
                }
            ],
            "children": []
        },
        {
            "serviceid": "3",
            "name": "Zabbix server node 2",
            "status": "-1",
            "algorithm": "2",
            "sortorder": "0",
            "weight": "0",
            "propagation_rule": "0",
            "propagation_value": "0",
            "description": "",
            "uuid": "9fc659a30fe244f690dff25fc2a9db5c",
            "created_at": "1761742654",
            "readonly": false,
            "parents": [
                {
                    "serviceid": "1",
                    "name": "Zabbix cluster"
                }
            ],
            "children": []
        }
    ],
    "id": 1
}
```


来源

CService::get() in *ui/include/classes/api/services/CService.php*.

服务器信息

此类用于获取 Zabbix 服务器的信息。

可用方法：

- serverinfo.get - 获取 Zabbix 服务器的上次更新时间

serverinfo.get

此方法用于检索指定 Zabbix 服务器的上次更新时间。

User、*Admin*、*Super admin* 和未经过身份验证的用户均可使用此方法。

方法签名：`object serverinfo.get(object parameters)`

方法源代码：`ui/include/classes/api/services/CServerInfo.php` 中的 `CServerInfo::get()`

参数

Parameter	Type	Description
serverid	ID	Zabbix 服务器的 ID。 参数行为： - 必需
output	query	此参数在参考说明中进行了描述。

返回值

属性	类型	描述
serverid	ID	Zabbix 服务器 ID。
lastaccess	timestamp	Zabbix 服务器上次更新的时间。 对于高可用性（HA）集群，此项显示活动节点的 `lastaccess` 值。

示例

获取服务器信息

获取指定 Zabbix 服务器的信息。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "serverinfo.get",
    "params": {
        "serverid": "019fb1f9-22bd-74ea-bc03-51efb8b76aba",
        "output": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "serverid": "019fb1f9-22bd-74ea-bc03-51efb8b76aba",
        "lastaccess": 1785337245
    },
    "id": 1
}
```

模块

此类旨在与前端模块一起使用。

对象参考：

- Module

可用方法：

- module.create - 安装新模块
- module.delete - 卸载模块
- module.get - 获取模块
- module.update - 更新模块

模块对象

以下内容将详细介绍有关 模块 API 的相关功能。

模块

module 对象具有以下属性。

Property	Type	Description
moduleid	ID	模块在数据库中存储的 ID。 Property behavior: - 只读 - 在更新操作中为 必需
id	string	由开发者在模块的`manifest.json` 文件中定义的唯一模块 ID。 内置模块的可能值： 请参见Dashboard widget 中 “type” 属性的说明。 Property behavior: - 在创建操作中为 必需
relative_path	string	相对于 Zabbix 前端目录的模块目录路径。 可能的值： `widgets/*` - 用于内置 widget 模块； `modules/*` - 用于第三方模块。 Property behavior: - 在创建操作中为 必需
status	integer	模块是启用还是禁用。 可能的值： 0 - *(默认)* 已禁用； 1 - 已启用。
config	object	模块配置。

创建

描述

`object module.create(object/array modules)`

此方法允许安装新的前端模块。

Note:
此方法仅适用于 超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['modules_config_enabled']` 设置为 `false`，则不允许用户调用此方法。

Attention:
必须手动将模块文件解压到正确的子目录中，并与模块的 `relative_path` 属性相匹配。

参数

`(object/array)` 要创建的模块。

此方法接受模块带有规范的对象属性standard token properties。

返回值

(object) 返回一个对象，其中包含 moduleids 属性下已安装模块的 ID。返回的 ID 的顺序与传递的模块的顺序匹配。

示例

安装模块

安装状态为" 已启用" 的模块。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "module.create",
    "params": {
        "id": "example_module",
        "relative_path": "modules/example_module",
        "status": 1
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "moduleids": [
            "25"
        ]
    },
    "id": 1
}
```

另请参阅

- 模块
- 前端模块

来源

CModule::create() in *ui/include/classes/api/services/CModule.php*。

删除

描述

`object module.delete(array moduleids)`

此方法允许您卸载模块。

> **Note:**
> 此方法仅适用于超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件中将 $ZBX_FEATURE_FLAGS['modules_config_enabled'] 设置为 false，则不允许用户调用此方法。

> **Attention:**
> 必须手动删除模块文件。

参数

(array) 要卸载模块的 ID。

返回值

(object) 返回一个对象，其中包含 moduleids 属性下已卸载模块的 ID。

示例

卸载多个模块

卸载模块"27" 和"28" 。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "module.delete",
    "params": [
        "27",
        "28"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "moduleids": [
            "27",
            "28"
        ]
    },
    "id": 1
}
```

来源

CModule::delete() in *ui/include/classes/api/services/CModule.php*。

更新

描述

`object module.update(object/array modules)`

此方法允许更新现有模块。

> **Note:**
> 此方法仅适用于超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['modules_config_enabled']` 设置为 `false`，则不允许用户调用此方法。

参数

(object/array) 要更新的模块属性。

必须为每个模块定义 `moduleid` 属性，所有其他属性都是可选的。仅更新指定的属性。

该方法接受具有标准模块属性的模块。

返回值

(object) 返回一个对象，其中包含 `moduleids` 属性下已更新模块的 ID。

示例

禁用模块

禁用模块"25"。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "module.update",
    "params": {
        "moduleid": "25",
        "status": 0
```

```
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "moduleids": [
            "25"
        ]
    },
    "id": 1
}
```

另请参阅

- 模块
- 前端模块

来源

CModule::update() in *ui/include/classes/api/services/CModule.php*。

获取

描述

`integer/array module.get(object parameters)`

该方法允许根据给定参数检索模块。

Note:
此方法仅适用于超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。
 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['modules_config_enabled']` 设置为 `false`，则不允许用户调用此方法。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
moduleids	ID/array	仅返回具有给定 ID 的模块。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`moduleid`、`relative_path`。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回两者其中任一：

- 一组对象数组；
- 如果已经使用了 `countOutput` 参数，则检索对象的计数。

示例

通过 ID 检索模块

检索有关模块"1"、"2" 和"25" 的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "module.get",
    "params": {
        "output": "extend",
        "moduleids": [
            "1",
            "2",
            "25"
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "moduleid": "1",
            "id": "actionlog",
            "relative_path": "widgets/actionlog",
            "status": "1",
            "config": []
        },
        {
            "moduleid": "2",
            "id": "clock",
            "relative_path": "widgets/clock",
            "status": "1",
            "config": []
        },
        {
            "moduleid": "25",
            "id": "example",
            "relative_path": "modules/example_module",
            "status": "1",
            "config": []
        }
    ],
    "id": 1
}
```

另请参阅

- 模块
- 仪表板小部件
- 前端模块

来源

CModule::get() in *ui/include/classes/api/services/CModule.php*。

模板

此类设计用于处理模板。

对象引用：

- 模板
- 模板标签

可用方法：

- template.create - 创建新模板
- template.delete - 删除模板
- template.get - 检索模板
- template.massadd - 向模板添加相关对象
- template.massremove - 从模板中移除相关对象
- template.update - 更新模板

模版对象

以下对象都是与 `template` 直接相关的 API。

模板

模板对象具有以下属性。

属性	类型	描述
templateid	ID	模板的 ID。 属性行为： - 只读 - 更新操作时为 必填
description	text	模板的描述。
host	string	模板的技术名称。 属性行为： - 创建操作时为 必填
name	string	模板的可见名称。 默认值：`host` 属性值。
readme	text	显示在主机向导中的模板专用配置说明。支持 Markdown 格式。
uuid	string	通用唯一标识符，用于将导入的模板链接到已存在的模板。如果未提供，则自动生成。
vendor_name	string	模板供应商名称。 对于创建操作，`vendor_name` 和 `vendor_version` 必须同时设置或同时留空。对于更新操作，如果数据库中已有值，则 `vendor_version` 可以留空。
vendor_version	string	模板供应商版本。 对于创建操作，`vendor_name` 和 `vendor_version` 必须同时设置或同时留空。对于更新操作，如果数据库中已有值，则 `vendor_name` 可以留空。
wizard_ready	integer	该模板是否可在主机向导中选择。 可能的值： 0 - *(默认)* 不可用。 1 - 可用。

模板标签

模板标签对象具有以下属性。

属性	类型	描述
tag	string	模板标签名称。 属性行为： - 必填

属性	类型	描述
value	string	模板标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板。 属性行为： - 只读。
objectid	ID	标签继承自的对象的 ID。 属性行为： - 只读。

创建

描述

`object template.create(object/array templates)`

此方法允许您创建新的模板。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的模板。

除了标准模板属性之外，该方法还接受以下参数。

参数	类型	描述
groups	object/array	要将模板添加到的模板组。 模板组必须仅定义 `groupid` 属性。 参数行为： - 必需
tags	object/array	模板标签。
templates	object/array	要链接到该模板的模板。 这些模板必须仅定义 `templateid` 属性。
macros	object/array	要为该模板创建的用户宏。

返回值

`(object)` 在 `templateids` 属性下返回一个包含创建的模板的 ID 的对象。返回的 ID 顺序与传入模板的顺序相匹配。

示例

创建模板

创建一个带有标签的模板，并将两个模板链接到该模板。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.create",
    "params": {
        "host": "Linux template",
        "groups": {
            "groupid": 1
        },
```

```
        "templates": [
            {
                "templateid": "11115"
            },
            {
                "templateid": "11116"
            }
        ],
        "tags": [
            {
                "tag": "host-name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "11117"
        ]
    },
    "id": 1
}
```

来源

CTemplate::create() 在 *ui/include/classes/api/services/CTemplate.php*.

删除

描述

`object template.delete(array templateIds)`

此方法允许您删除模板。

删除模板时，也会从其链接到的任何主机或模板中移除该模板的所有实体（监控项、触发器、图形等）。如果要保留这些实体，请先使用`host.update`、`host.massremove`、`template.update` 或`template.massremove` 方法，将模板从相关主机或模板中解除链接。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。在用户角色设置中，可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

(array) 要删除的模板的 ID。

返回值

(object) 返回一个对象，该对象包含 templateids 属性下已删除模板的 ID。

示例

删除多个模版

删除 2 个模版。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.delete",
```

```
    "params": [
        "13",
        "32"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "13",
            "32"
        ]
    },
    "id": 1
}
```

来源

CTemplate::delete() 在 *ui/include/classes/api/services/CTemplate.php*.

批量删除

描述

`object template.massremove(object parameters)`

此方法允许你从多个模板中移除相关对象。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 包含要更新的模板 ID 以及应删除的对象的参数。

参数	类型	描述
templateids	ID/array	要更新的模板的 ID。 参数行为： - 必需
groupids	ID/array	要从中移除给定模板的模板组的 ID。
macros	string/array	要从给定模板中删除的用户宏的 ID。
templateids_clear	ID/array	要从给定模板中取消链接并清除的模板的 ID（上游）。
templateids_link	ID/array	要从给定模板中取消链接的模板的 ID（上游）。

返回值

`(object)` 返回一个对象，该对象包含 `templateids` 属性下已更新模板的 ID。

示例

从一个组中移除模版

将两个模版从模版组”2” 中移除。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": [
```

```
            "10085",
            "10086"
        ],
        "groupids": "2"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

从主机上取消链接模版

取消模版”10085” 上的链接的模版”10106” 和”10104” 。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "templateids_link": [
            "10106",
            "10104"
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085"
        ]
    },
    "id": 1
}
```

另请参阅

- template.update
- 用户宏

来源

CTemplate::massRemove() 在 *ui/include/classes/api/services/CTemplate.php*.

批量添加

描述

`object template.massadd(object parameters)`

此方法允许你同时向指定模板添加多个相关对象。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(object) 包含要更新的模板 ID 以及要添加到模板中的对象的参数。

该方法接受以下参数。

参数	类型	描述
templates	object/array	要更新的模板。 这些模板必须仅定义 templateid 属性。 参数行为： - 必需
groups	object/array	要将给定模板添加到的模板组。 这些模板组必须仅定义 groupid 属性。
macros	object/array	要为给定模板创建的用户宏。
templates_link	object/array	要链接到给定模板的模板。 这些模板必须仅定义 templateid 属性。

返回值

(object) 返回一个对象，该对象包含 templateids 属性下已更新模板的 ID。

示例

将一个模板组链接到模板

将模板组"2" 添加到两个模板中。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
```

```
        ]
    },
    "id": 1
}
```

将两个模板链接到一个模板

将模板”10106” 和”10104” 链接到模板”10073”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10073"
            }
        ],
        "templates_link": [
            {
                "templateid": "10106"
            },
            {
                "templateid": "10104"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10073"
        ]
    },
    "id": 1
}
```

另请参阅

- template.update
- 主机
- 模板组
- 用户宏

来源

CTemplate::massAdd() 在 *ui/include/classes/api/services/CTemplate.php*.

更新

描述

`object template.update(object/array templates)`

此方法允许您更新现有模板。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(object/array) 要更新的模板属性。

必须为每个模板定义 templateid 属性，其他所有属性均为可选。只有给定的属性会被更新，其他所有属性将保持不变。

除了标准模板属性之外，该方法还接受以下参数。

参数	类型	说明
groups	object/array	用于替换模板当前所属模板组的模板组。 模板组必须仅定义 groupid 属性。
tags	object/array	用于替换当前模板标签的模板标签。
macros	object/array	用于替换给定模板上当前用户宏的用户宏。
templates	object/array	用于替换当前已链接模板的模板。未传递的模板将仅被取消链接。 模板必须仅定义 templateid 属性。
templates_clear	object/array	要从给定模板中取消链接并清除的模板。 模板必须仅定义 templateid 属性。

返回值

(object) 返回一个对象，该对象包含 templateids 属性下已更新模板的 ID。

示例

更改标准模板属性

将模板的技术名称更改为"Linux by Zabbix agent Custom"，将可见名称更改为"My template"，并更新模板描述。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "host": "Linux by Zabbix agent Custom",
        "name": "My template",
        "description": "This is a custom Linux template."
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

更新模板组

将给定模板的所有模板组替换为另一个模板组。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "groups": [
```

```
            {
                "groupid": "24"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

更新多个模板组

将多个模板的所有模板组替换为不同的模板组。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": [
        {
            "templateid": "10086",
            "groups": [
                {
                    "groupid": "24"
                }
            ]
        },
        {
            "templateid": "10087",
            "groups": [
                {
                    "groupid": "1"
                },
                {
                    "groupid": "12"
                }
            ]
        },
        {
            "templateid": "10088",
            "groups": [
                {
                    "groupid": "1"
                },
                {
                    "groupid": "12"
                }
            ]
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086",
            "10087",
            "10088"
        ]
    },
    "id": 1
}
```

更新模板标签

将所有模板标签替换为另一个标签。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "tags": [
            {
                "tag": "host-name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

更新模板宏

将所有模板宏替换为另一个不同的宏。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "macros": [
            {
                "macro": "{$MY_MACRO}",
                "value": "new_value"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

更新多个模板宏

将多个模板中的所有用户宏替换为给定的用户宏。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": [
            {
                "templateid": "10087",
                "macros": [
                    {
                        "macro": "{$AGENT.TIMEOUT}",
                        "value": "5m",
                        "description": "agent 被视为不可用前的超时时间。仅适用于可从 Zabbix 服务器/proxy 访问的 agen
                    }
                ]
            },
            {
                "templateid": "10088",
                "macros": [
                    {
                        "macro": "{$AGENT.TIMEOUT}",
                        "value": "5m",
                        "description": "agent 被视为不可用前的超时时间。仅适用于可从 Zabbix 服务器/proxy 访问的 agen
                    }
                ]
            }
        ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10087",
            "10088"
        ]
    },
    "id": 1
}
```

更新模板的已链接模板

取消给定模板与所有模板的链接（不清除），并为其链接另一个模板。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
```

```
        "templateid": "10086",
        "templates": [
            {
                "templateid": "10001"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

清除多个模板的已链接模板

从给定模板中取消链接并清除指定的已链接模板。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": [
        {
            "templateid": "10087",
            "templates_clear": [
                {
                    "templateid": "10001"
                }
            ]
        },
        {
            "templateid": "10088",
            "templates_clear": [
                {
                    "templateid": "10001"
                }
            ]
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10087",
            "10088"
        ]
    },
    "id": 1
}
```

来源

CTemplate::update() 在 *ui/include/classes/api/services/CTemplate.php*.

获取

描述

`integer/array template.get(object parameters)`

该方法允许您根据给定参数检索模板。

> **Note:**
> 此方法对任何类型的用户都可用。可在用户角色设置中撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
templateids	ID/array	仅返回具有给定模板 ID 的模板。
groupids	ID/array	仅返回属于给定模板组的模板。
parentTemplateids	ID/array	仅返回给定模板所链接到的模板。
hostids	ID/array	仅返回链接到给定主机/模板的模板。
graphids	ID/array	仅返回包含给定图形的模板。
itemids	ID/array	仅返回包含给定监控项的模板。
triggerids	ID/array	仅返回包含给定触发器的模板。
with_items	flag	仅返回具有监控项的模板。
with_triggers	flag	仅返回具有触发器的模板。
with_graphs	flag	仅返回具有图形的模板。
with_httptests	flag	仅返回具有 Web 场景的模板。
evaltype	integer	标签评估方法。 可选值： 0 - *(默认)* And/Or； 2 - Or。
tags	object/array	仅返回具有给定标签的模板。 格式：`[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组将返回所有模板。 可选的operator值： 0 - *(默认)* Contains； 1 - Equals； 2 - Does not contain； 3 - Does not equal； 4 - Exists； 5 - Does not exist。
inheritedTags	boolean	返回在链接模板中也具有给定 `tags` 的模板。 可选值： `true` - 链接模板也必须具有给定标签； `false` - *(默认)* 忽略来自链接模板的标签。
selectInheritedTags	query	返回`inheritedTags` 属性，其中包含链接模板上的标签。
selectTags	query	在`tags` 属性中返回模板标签。
selectDiscoveryRules	query	返回`discoveryRules` 属性，其中包含模板 LLD 规则。 支持 `count`。
selectHosts	query	在`hosts` 属性中返回链接到该模板的主机。 支持 `count`。
selectTemplateGroups	query	在`templategroups` 属性中返回该模板所属的模板组。

参数	类型	说明
selectTemplates	query	在templates 属性中返回给定模板所链接到的模板。 支持 count。
selectParentTemplates	query	在 parentTemplates 属性中返回链接到给定模板的模板。 支持 count。
selectHttpTests	query	在httpTests 属性中返回模板中的 Web 场景。 支持 count。
selectItems	query	在items 属性中返回模板中的监控项。 支持 count。
selectTriggers	query	在triggers 属性中返回模板中的触发器。 支持 count。
selectGraphs	query	在graphs 属性中返回模板中的图形。 支持 count。
selectMacros	query	在macros 属性中返回模板中的宏。
selectDashboards	query	在dashboards 属性中返回模板中的仪表板。 支持 count。
selectValueMaps	query	返回valuemaps 属性，其中包含模板值映射。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectTemplates - 结果将按 name 排序； selectHosts - 按 host 排序； selectParentTemplates - 按 host 排序； selectItems - 按 name 排序； selectDiscoveryRules - 按 name 排序； selectTriggers - 按 description 排序； selectGraphs - 按 name 排序； selectDashboards - 按 name 排序。
sortfield	string/array	按给定属性对结果进行排序。 可选值：hostid、host、name、status。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectDiscoveries	query	在 discoveries 属性中返回模板中的低级别发现。 支持 count。 此查询已弃用，请改用 selectDiscoveryRules。

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回已检索对象的数量。

示例

按名称检索模板

检索名为 “Linux by Zabbix agent” 和 “Windows by Zabbix agent” 的两个模板的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "Linux by Zabbix agent",
                "Windows by Zabbix agent"
            ]
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "proxyid": "0",
            "host": "Linux by Zabbix agent",
            "status": "3",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Linux by Zabbix agent",
            "flags": "0",
            "templateid": "10001",
            "description": " 这是一个官方 Linux 模板。它需要 Zabbix agent 8.0 或更高版本。\r\n\r\n有关文件系统（FS）
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "custom_interfaces": "0",
            "uuid": "f8f7908280354f2abeed07dc788c3747",
            "vendor_name": "Zabbix",
            "vendor_version": "8.0-2",
            "proxy_groupid": "0",
            "monitored_by": "0",
            "wizard_ready": "1",
            "readme": "## 概述\r\n\r\n这是一个官方 Linux 模板。它需要 Zabbix agent 8.0 或更高版本。\r\n\r\n#### 有
        },
        {
            "proxyid": "0",
            "host": "Windows by Zabbix agent",
            "status": "3",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
```

```
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Windows by Zabbix agent",
            "flags": "0",
            "templateid": "10081",
            "description": " 这是一个官方 Windows 模板。它需要 Zabbix agent 8.0 或更高版本。\r\n\r\n您可以在我们的论
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "custom_interfaces": "0",
            "uuid": "13b06904a6bf41cbb795e3193d896340",
            "vendor_name": "Zabbix",
            "vendor_version": "8.0-2",
            "proxy_groupid": "0",
            "monitored_by": "0",
            "wizard_ready": "1",
            "readme": "## 概述\r\n\r\n这是一个官方 Windows 模板。它需要 Zabbix agent 8.0 或更高版本。"
        }
    ],
    "id": 1
}
```


检索模板组

检索模板”Linux by Zabbix agent” 所属的模板组。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTemplateGroups": "extend",
        "filter": {
            "host": [
                "Linux by Zabbix agent"
            ]
        }
    },
    "id": 1
}
```

响应:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10001",
            "templategroups": [
                {
                    "groupid": "10",
                    "name": "Templates/Operating systems",
                    "uuid": "846977d1dfed4968bc5f8bdb363285bc"
                }
            ]
        }
    ],
```

```
    "id": 1
}
```

按模板检索主机

检索已链接到"10001"（即 *Linux by Zabbix agent*）模板的主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "templateid",
        "templateids": "10001",
        "selectHosts": ["hostid", "name"]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10001",
            "hosts": [
                {
                    "hostid": "10084",
                    "name": "Zabbix server"
                },
                {
                    "hostid": "10603",
                    "name": "Host 1"
                },
                {
                    "hostid": "10604",
                    "name": "Host 2"
                }
            ]
        }
    ],
    "id": 1
}
```

按模板标签搜索

检索具有标签 "host-name" 且其值等于 "{HOST.NAME}" 的模板。

请求 :

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "host-name",
                "value": "{HOST.NAME}",
                "operator": 1
            }
        ]
    },
```

```
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10402",
            "tags": [
                {
                    "tag": "host-name",
                    "value": "{HOST.NAME}"
                }
            ]
        }
    ],
    "id": 1
}
```

另请参见

- 模板组
- 模板
- 用户宏
- 主机接口

来源

CTemplate::get() 在 *ui/include/classes/api/services/CTemplate.php*.

模板仪表板

此类用于处理模板仪表板。

对象参考：

- 模板仪表板
- 模板仪表板页面
    - 模板仪表板小部件
        * 模板仪表板小部件字段

可用方法：

- templatedashboard.create - 创建新的模板仪表板
- templatedashboard.delete - 删除模板仪表板
- templatedashboard.get - 检索模板仪表板
- templatedashboard.update - 更新模板仪表板

模版仪表盘对象

以下对象与 `templatedashboard` API 直接相关。

模板仪表板

模板仪表板对象具有以下属性。

Property	Type	Description
dashboardid	ID	模板仪表板的 ID。 Property behavior: - 只读 - 更新操作时必需

Property	Type	Description
name	string	模板仪表板的名称。
templateid	ID	Property behavior: - 创建操作时必需 该仪表板所属模板的 ID。
display_period	integer	Property behavior: - 常量 - 创建操作时必需 默认幻灯片播放间隔（秒）。 Possible values: 10, 30, 60, 120, 600, 1800, 3600. Default: 30.
auto_start	integer	自动开始幻灯片播放。 Possible values: 0 - 不自动开始幻灯片播放； 1 - *(default)* 自动开始幻灯片播放。
uuid	string	通用唯一标识符，用于将导入的模板仪表板链接到已存在的模板仪表板。如果未提供，则自动生成。

模板仪表板页面

模板仪表板页面对象具有以下属性。

Property	Type	Description
dashboard_pageid	ID	仪表板页面的 ID。
name	string	Property behavior: - *read-only* 仪表板页面名称。 默认值：空字符串。
display_period	integer	仪表板幻灯片播放间隔（秒）。 可能的值：0, 10, 30, 60, 120, 600, 1800, 3600。 默认值：0（将使用默认幻灯片播放间隔）。
widgets	array	模板仪表板小部件 对象数组。

模板仪表板部件

模板仪表板部件对象具有以下属性。

Property	Type	Description
widgetid	ID	仪表板部件的 ID。 Property behavior： - *read-only*

Property	Type	Description
type	string	仪表板部件的类型。 可能的值： actionlog - 操作日志； clock - 时钟； discovery - 发现状态； favgraphs - 收藏图形； favmaps - 收藏地图； gauge - 仪表； graph - 图形（经典）； graphprototype - 图形原型； honeycomb - 蜂窝图； hostavail - 主机可用性； hostcard - 主机卡片； hostnavigator - 主机导航器； itemcard - 监控项卡片； itemnavigator - 监控项导航器； item - 监控项值； map - 地图； navtree - 地图导航树； piechart - 饼图； plaintext - 纯文本； problemhosts - 故障主机； problems - 故障； problemsbysv - 按严重性分类的故障； scatterplot - 散点图； slareport - SLA 报告； svggraph - 图形； systeminfo - 系统信息； tophosts - 主机排行； topitems - 监控项排行； toptriggers - 触发器排行； trigover - 触发器概览； url - URL； web - Web 监控。 Property behavior： - *required*
name	string	自定义部件名称。
x	integer	距仪表板左侧的水平位置。 可能的值范围为 0 到 71。
y	integer	距仪表板顶部的垂直位置。 可能的值范围为 0 到 63。
width	integer	部件宽度。 可能的值范围为 1 到 72。
height	integer	部件高度。 可能的值范围为 1 到 64。
view_mode	integer	部件视图模式。 可能的值： 0 - *(default)* 默认部件视图； 1 - 隐藏标题；
fields	array	模板仪表板部件字段对象数组。

模板仪表板小部件字段

模板仪表板小部件字段对象具有以下属性。

属性	类型	描述
type	integer	小部件字段的类型。 可能的值： 0 - 整数； 1 - 字符串； 4 - 监控项； 5 - 监控项原型； 6 - 图形； 7 - 图形原型； 8 - 拓扑图； 9 - 服务； 10 - SLA； 11 - 用户； 12 - 操作； 13 - 媒介类型。 属性行为： - 必需
name	string	小部件字段名称。 可能的值：请参见仪表板小部件字段。请注意，在模板仪表板上配置小部件时，某些与主机相关的参数（例如，问题小部件中的 主机组、排除的主机组和 主机，主机可用性小部件中的 主机组等）不可用。这是因为模板仪表板仅显示模板所链接主机的数据。 属性行为： - 必需
value	mixed	取决于类型的小部件字段值。 可能的值：请参见仪表板小部件字段。请注意，在模板仪表板上配置小部件时，某些与主机相关的参数（例如，问题小部件中的 主机组、排除的主机组和 主机，主机可用性小部件中的 主机组等）不可用。这是因为模板仪表板仅显示模板所链接主机的数据。 属性行为： - 必需

创建

描述

`object templatedashboard.create(object/array templateDashboards)`

此方法允许您创建新的模板仪表板。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的模板仪表板。

除了标准模板仪表板属性之外，该方法还接受以下参数。

参数	类型	说明
pages	array	要为仪表板创建的模板仪表板页面。仪表板页面将按照指定的顺序进行排序。 参数行为: - 必需

返回值

(object) 返回一个对象，该对象包含 dashboardids 属性下创建的模板仪表盘的 ID。返回 ID 的顺序与传递的模板仪表盘的顺序匹配。

示例

创建模版仪表盘

在单个仪表盘页面上使用图形组件创建一个名为 “Graphs” 的模板仪表盘。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.create",
    "params": {
        "templateid": "10318",
        "name": "Graphs",
        "pages": [
            {
                "widgets": [
                    {
                        "type": "graph",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 6,
                                "name": "graphid",
                                "value": "1123"
                            }
                        ]
                    }
                ]

            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "32"
        ]
    },
    "id": 1
}
```

另请参阅

- 模板仪表板页面
- 模板仪表板小部件
- 模板仪表板小部件字段

来源

CTemplateDashboard::create() 在 *ui/include/classes/api/services/CTemplateDashboard.php*.

删除

描述

`object templatedashboard.delete(array templateDashboardIds)`

此方法允许您删除模板仪表板。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 需要删除的模板仪表盘 ID。

返回值

(object) 返回一个对象，该对象包含 dashboardids 属性下已删除模板仪表盘的 ID。

示例

删除多个模版仪表盘

删除两个模版仪表盘。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.delete",
    "params": [
        "45",
        "46"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "45",
            "46"
        ]
    },
    "id": 1
}
```

来源

CTemplateDashboard::delete() 在 *ui/include/classes/api/services/CTemplateDashboard.php*.

更新

描述

`object templatedashboard.update(object/array templateDashboards)`

此方法允许您更新现有的模板仪表板。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(object/array) 要更新的模板仪表板属性。

必须为每个仪表板指定 dashboardid 属性，所有其他属性均为可选。只有指定的属性会被更新。

除了标准模板仪表板属性之外，该方法还接受以下参数。

参数	类型	说明
pages	array	用于替换现有仪表板页面的模板仪表板页面。 仪表板页面通过 dashboard_pageid 属性进行更新。对于不包含 dashboard_pageid 属性的对象，将创建新的仪表板页面；如果现有仪表板页面未被复用，则将被删除。仪表板页面将按照指定的顺序排序。只有仪表板页面中指定的属性会被更新。pages 属性至少需要一个仪表板页面对象。

返回值

(object) 返回一个对象，该对象包含 dashboardids 属性下更新的模板仪表盘的 ID。

示例

重命名模版仪表盘

将一个模板仪表盘重命名为"Performance graphs"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.update",
    "params": {
        "dashboardid": "23",
        "name": "Performance graphs"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "23"
        ]
    },
    "id": 1
}
```

更新模版仪表盘页面

重命名第一个仪表盘页面，替换第二个仪表盘页面上的组件，并添加一个新页面作为第三个页面。删除所有其他仪表盘页面。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.update",
    "params": {
        "dashboardid": "2",
        "pages": [
            {
                "dashboard_pageid": 1,
                "name": "Renamed Page"
            },
            {
                "dashboard_pageid": 2,
                "widgets": [
                    {
                        "type": "clock",
                        "x": 0,
                        "y": 0,
                        "width": 12,
```

```
                        "height": 3
                    }
                ]
            },
            {
                "display_period": 60
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

另请参阅

- 模板仪表板小部件
- 模板仪表板小部件字段

来源

CTemplateDashboard::update() 在 *ui/include/classes/api/services/CTemplateDashboard.php*.

获取

描述

`integer/array templatedashboard.get(object parameters)`

该方法允许您根据给定参数检索模板仪表板。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
dashboardids	ID/array	仅返回具有给定 ID 的模板仪表板。
templateids	ID/array	仅返回属于给定模板的模板仪表板。
selectPages	query	返回`pages` 属性，其中包含已正确排序的模板仪表板页面。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`dashboardid`、`name`。
countOutput	boolean	这些参数在参考注释中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

参数	类型	描述
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回两者其一：

- 一个对象数组；
- 如果使用了'countOutput' 参数，则对检索对象进行计数。

示例

检索模版仪表盘

检索指定模板的所有带有组件的模板仪表盘。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.get",
    "params": {
        "output": "extend",
        "selectPages": "extend",
        "templateids": "10001"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "23",
            "name": "Docker overview",
            "templateid": "10001",
            "display_period": "30",
            "auto_start": "1",
            "uuid": "6dfcbe0bc5ad400ea9c1c2dd7649282f",
            "pages": [
                {
                    "dashboard_pageid": "1",
                    "name": "",
                    "display_period": "0",
                    "widgets": [
                        {
                            "widgetid": "220",
                            "type": "graph",
                            "name": "",
                            "x": "0",
                            "y": "0",
                            "width": "36",
                            "height": "5",
                            "view_mode": "0",
                            "fields": [
                                {
                                    "type": "6",
                                    "name": "graphid",
                                    "value": "1125"
                                }
                            ]
                        },
                        {
```

```
                "widgetid": "221",
                "type": "graph",
                "name": "",
                "x": "12",
                "y": "0",
                "width": "36",
                "height": "5",
                "view_mode": "0",
                "fields": [
                    {
                        "type": "6",
                        "name": "graphid",
                        "value": "1129"
                    }
                ]
            },
            {
                "widgetid": "222",
                "type": "graph",
                "name": "",
                "x": "0",
                "y": "5",
                "width": "36",
                "height": "5",
                "view_mode": "0",
                "fields": [
                    {
                        "type": "6",
                        "name": "graphid",
                        "value": "1128"
                    }
                ]
            },
            {
                "widgetid": "223",
                "type": "graph",
                "name": "",
                "x": "12",
                "y": "5",
                "width": "36",
                "height": "5",
                "view_mode": "0",
                "fields": [
                    {
                        "type": "6",
                        "name": "graphid",
                        "value": "1126"
                    }
                ]
            },
            {
                "widgetid": "224",
                "type": "graph",
                "name": "",
                "x": "0",
                "y": "10",
                "width": "36",
                "height": "5",
                "view_mode": "0",
                "fields": [
                    {
                        "type": "6",
```

```
                                    "name": "graphid",
                                    "value": "1127"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```

另请参阅

- 模板仪表板页面
- 模板仪表板小部件
- 模板仪表板小部件字段

来源

CTemplateDashboard::get() 在 *ui/include/classes/api/services/CTemplateDashboard.php*.

模板组

此类用于处理模板组。

对象引用：

- 模板组

可用方法：

- templategroup.create - 创建新的模板组
- templategroup.delete - 删除模板组
- templategroup.get - 检索模板组
- templategroup.massadd - 向模板组添加相关对象
- templategroup.massremove - 从模板组中移除相关对象
- templategroup.propagate - 将权限传播到模板组的子组
- templategroup.update - 更新模板组

模板组对象

以下对象都是与 `templategroup` 直接相关的 API.

模板组

模板组对象具有以下属性。

属性	类型	描述
groupid	ID	模板组的 ID。 属性行为: - 只读 - 更新操作时 必填
name	string	模板组名称。 属性行为: - 创建操作时 必填
uuid	string	通用唯一标识符，用于将导入的模板组链接到已存在的模板组。如果未提供，则自动生成。

传播

描述

`object templategroup.propagate(object parameters)`

此方法允许你将权限应用到所有模板组的子组。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
groups	object/array	要传播的模板组。 模板组必须仅定义 groupid 属性。 参数行为： - 必需
permissions	boolean	如果需要传播权限，则设置为 true。 参数行为： - 必需

返回值

(object) 返回一个对象，该对象包含 groupids 属性下已被下发权限的模板组的 ID.

示例

将权限下发到模板组的子组.

将权限下发到模板组的子组.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.propagate",
    "params": {
        "groups": [
            {
                "groupid": "15"
            }
        ],
        "permissions": true
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "15",
        ]
    },
    "id": 1
}
```

参阅

- templategroup.update
- templategroup.massadd
- Template

来源

CTemplateGroup::propagate() in *ui/include/classes/api/services/CTemplateGroup.php*.

创建

描述

`object templategroup.create(object/array templateGroups)`

此方法允许您创建新的模板组。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要创建的模板组。该方法接受具有标准模板组属性的模板组。

返回值

(object) 返回一个对象，该对象包含 groupids 属性下创建的模板组的 ID. 返回 ID 的顺序与传递模板组的顺序匹配.

示例

创建一个模板组

创建一个名为"Templates/Databases" 的模板组.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.create",
    "params": {
        "name": "Templates/Databases"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107820"
        ]
    },
    "id": 1
}
```

来源

CTemplateGroup::create() in *ui/include/classes/api/services/CTemplateGroup.php*.

删除

描述

`object templategroup.delete(array templateGroupIds)`

此方法允许您删除模板组。

如果某个模板组仅包含属于该组的模板，则无法删除该模板组。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 需要删除的模板组的 ID.

返回值

(object) 返回一个对象，该对象包含 groupids 属性下的被删除的模板组的 ID.

示例

删除多个模板组

删除两个模板组.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.delete",
    "params": [
        "107814",
        "107815"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107814",
            "107815"
        ]
    },
    "id": 1
}
```

来源

CTemplateGroup::delete() in *ui/include/classes/api/services/CTemplateGroup.php*.

批量删除

描述

`object templategroup.massremove(object parameters)`

此方法允许你从多个模板组中移除相关对象。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(object) 包含要更新的模板组 ID 以及应被移除的对象的参数。

参数	类型	描述
groupids	ID/array	要更新的模板组的 ID。 参数行为: - 必需
templateids	ID/array	要从所有模板组中移除的模板的 ID。 参数行为: - 必需

返回值

(object) 返回一个对象，该对象包含 groupids 属性下已更新的模板组的 ID.

示例

从模板组中删除模板

从给出的模板组中删除两个模板.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "templateids": [
            "30050",
            "30001"
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

来源

CTemplateGroup::massRemove() in *ui/include/classes/api/services/CTemplateGroup.php*.

批量添加

描述

`object templategroup.massadd(object parameters)`

此方法允许您同时将多个相关对象添加到所有给定的模板组中。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 包含要更新的模板组 ID 以及要添加到所有模板组中的对象的参数。

该方法接受以下参数。

参数	类型	说明
groups	object/array	要更新的模板组。 模板组必须仅定义 groupid 属性。 参数行为: - 必需
templates	object/array	要添加到所有模板组中的模板。 模板必须仅定义 templateid 属性。 参数行为: - 必需

返回值

(object) 返回一个对象，该对象包含 groupids 属性下已更新的模板组的 ID.

示例

将模板添加到模板组中

将两个模板添加到 ID 为 12 和 13 的模板组中.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "12"
            },
            {
                "groupid": "13"
            }
        ],
        "templates": [
            {
                "templateid": "10486"
            },
            {
                "templateid": "10487"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "12",
            "13"
        ]
    },
    "id": 1
}
```

参阅

- 模板

来源

CTemplateGroup::massAdd() in *ui/include/classes/api/services/CTemplateGroup.php*.

更新

描述

`object templategroup.update(object/array templateGroups)`

此方法允许你更新现有的模板组。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(object/array) 要更新的模板组属性。

必须为每个模板组定义 groupid 属性，所有其他属性均为可选。只有给定的属性会被更新，所有其他属性将保持不变。

返回值

(object) 返回一个对象，该对象包含 groupids 属性下已更新的模板组的 ID.

示例

重命名一个模板组

将一个模板组重命名为"Templates/Databases"

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.update",
    "params": {
        "groupid": "7",
        "name": "Templates/Databases"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "7"
        ]
    },
    "id": 1
}
```

来源

CTemplateGroup::update() in *ui/include/classes/api/services/CTemplateGroup.php*.

获取

描述

`integer/array templategroup.get(object parameters)`

该方法允许您根据给定参数检索模板组。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
graphids	ID/array	仅返回包含具有给定图形的模板的模板组。
groupids	ID/array	仅返回具有给定模板组 ID 的模板组。
templateids	ID/array	仅返回包含给定模板的模板组。
triggerids	ID/array	仅返回包含具有给定触发器的模板的模板组。
with_graphs	boolean	仅返回包含带有图形的模板的模板组。
with_graph_prototypes	boolean	仅返回包含带有图形原型的模板的模板组。
with_httptests	boolean	仅返回包含带有 Web 场景的模板的模板组。
with_items	boolean	仅返回包含带有监控项的模板的模板组。 会覆盖 with_simple_graph_items 参数。
with_item_prototypes	boolean	仅返回包含带有监控项原型的模板的模板组。 会覆盖 with_simple_graph_item_prototypes 参数。
with_simple_graph_item_prototypes	boolean	仅返回包含带有监控项原型的模板的模板组，这些原型已启用创建且信息类型为数值型。
with_simple_graph_items	boolean	仅返回包含带有数值型监控项的模板的模板组。
with_templates	boolean	仅返回包含模板的模板组。
with_triggers	boolean	仅返回包含带有触发器的模板的模板组。
selectTemplates	query	返回属于该模板组的templates 属性。 支持 count。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectTemplates - 结果将按 template 排序。
sortfield	string/array	按给定属性对结果进行排序。 可选值：groupid、name。
countOutput	boolean	这些参数在参考说明中有描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

按照名字检索模板组

检索名为”Templates/Databases” 和”Templates/Modules” 的两个模板组的数据.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.get",
    "params": {
        "output": "extend",
        "filter": {
            "name": [
                "Templates/Databases",
                "Templates/Modules"
            ]
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "13",
            "name": "Templates/Databases",
            "uuid": "748ad4d098d447d492bb935c907f652f"
        },
        {
            "groupid": "8",
            "name": "Templates/Modules",
            "uuid": "57b7ae836ca64446ba2c296389c009b7"
        }
    ],
    "id": 1
}
```

参阅

- 模板

来源

CTemplateGroup::get() in *ui/include/classes/api/services/CTemplateGroup.php*.

正则表达式

此类用于处理全局正则表达式。

对象参考：

- 正则表达式
- 表达式

可用方法：

- regexp.create - 创建新的正则表达式
- regexp.delete - 删除正则表达式
- regexp.get - 检索正则表达式
- regexp.update - 更新正则表达式

正则表达式对象

以下对象与 regexp API 直接相关。

正则表达式

全局正则表达式对象具有以下属性。

Property	Type	Description
regexpid	ID	正则表达式的 ID。 Property behavior: - *read-only* - *required* for update operations.
name	string	正则表达式的名称。 Property behavior: - *required* for create operations.
description	text	正则表达式的描述。
test_string	string	测试字符串。

表达式

expressions 对象具有以下属性。

Property	Type	Description
expression	string	正则表达式。 Property behavior: - *required*
expression_type	integer	正则表达式类型。 可能的值： 0 - 包含字符串； 1 - 包含列表中的任意子字符串； 2 - 不包含字符串； 3 - 匹配正则表达式； 4 - 不匹配正则表达式。 Property behavior: - *required*
exp_delimiter	string	表达式分隔符。 默认值：","。 可能的值：","、"."或"/"。 Property behavior: - 当 `expression_type` 设置为 *Contains any substring from list* 时，*supported*。
case_sensitive	integer	大小写敏感性。 默认值：0。 可能的值： 0 - 不区分大小写； 1 - 区分大小写。

创建

描述

`object regexp.create(object/array regularExpressions)`

此方法允许您创建新的全局正则表达式。

Note:
此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色 。

参数

(object/array) 要创建的正则表达式。

除了标准属性外，该方法还接受以下参数。

参数	类型	说明
expressions	array	Expressions 选项。 参数行为: - 必需

返回值

(object) 返回一个对象，其中包含 regexpids 属性下创建的正则表达式的 ID。

示例

创建新的全局正则表达式。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "regexp.create",
    "params": {
        "name": "Storage devices for SNMP discovery",
        "description": " 用于发现并识别受监控主机上的内存资源的全局正则表达式",
        "test_string": "/boot",
        "expressions": [
            {
                "expression": "^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$"
                "expression_type": "4",
                "case_sensitive": "1"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "regexpids": [
            "9"
        ]
    },
    "id": 1
}
```

参考来源

CRegexp::create() in *ui/include/classes/api/services/CRegexp.php*.

删除

描述

`object regexp.delete(array regexpids)`

此方法允许您删除全局正则表达式。

Note:
此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 代表计划删除的正则表达式的 ID 号。

返回数值

根据 regexpids 的特性 (object) 会反馈一个包含已删除正则表达式的 ID 对象。

示例

删除多个全局正则表达式。

请求:

```
{
"jsonrpc": "2.0",
"method": "regexp.delete",
"params": [
"16",
"17"
],
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": {
"regexpids": [
"16",
"17"
]
},
"id": 1
}
```

参考来源

CRegexp::delete() in *ui/include/classes/api/services/CRegexp.php*.

更新

描述

`object regexp.update(object/array regularExpressions)`

此方法允许您更新现有的全局正则表达式。

> **Note:**
> 此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的正则表达式属性。

必须为每个对象定义 regexpid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准属性之外，该方法还接受以下参数。

参数	类型	说明
expressions	array	表达式 选项。

返回值

根据 regexpids 的特性，(object) 会返回一个对象，其包含已升级的正则表达式 ID。

参考示例

更新用于文件系统发现的全局正则表达式。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "regexp.update",
    "params": {
        "regexpid": "1",
        "name": " 用于发现的文件系统",
        "description": " 用于发现并识别受监控主机上的文件系统的全局正则表达式",
        "test_string": "",
        "expressions": [
            {
                "expression": "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|zfs)$
                "expression_type": "3",
                "case_sensitive": "0"
            },
            {
                "expression": "^(ntfs|fat32|fat16)$",
                "expression_type": "3",
                "case_sensitive": "0"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "regexpids": [
            "1"
        ]
    },
    "id": 1
}
```

参考来源

CRegexp::update() in *ui/include/classes/api/services/CRegexp.php*.

获取

描述

`integer/array regexp.get(object parameters)`

该方法允许您根据给定参数检索全局正则表达式。

> **Note:**
> 此方法仅对 *Super Admin* 可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
regexpids	ID/array	仅返回具有给定 ID 的正则表达式。
selectExpressions	query	返回`expressions` 属性。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`regexpid`、`name`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	

参数	类型	描述
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下任一值：

- 对象数组；
- 如果已使用 countOutput 参数，则返回检索到的对象的数量。

参考示例

获取全局正则表达式。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "regexp.get",
    "params": {
        "output": ["regexpid","name","description"],
        "selectExpressions": ["expression","expression_type"],
        "regexpids": [1,2],
        "preservekeys": true
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "1": {
            "regexpid": "1",
            "name": "File systems for discovery",
            "description": "",
            "expressions": [
                {
                    "expression": "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|n
                    "expression_type": "3"
                }
            ]
        },
        "2": {
            "regexpid": "2",
            "name": "Network interfaces for discovery",
            "description": "",
            "expressions": [
                {
                    "expression": "^Software Loopback Interface",
                    "expression_type": "4"
                },
                {
                    "expression": "^(In)?[Ll]oop[Bb]ack[0-9._]*$",
                    "expression_type": "4"
                },
```

```
                {
                    "expression": "^NULL[0-9.]*$",
                    "expression_type": "4"
                },
                {
                    "expression": "^[Ll]o[0-9.]*$",
                    "expression_type": "4"
                },
                {
                    "expression": "^[Ss]ystem$",
                    "expression_type": "4"
                },
                {
                    "expression": "^Nu[0-9.]*$",
                    "expression_type": "4"
                }
            ]
        }
    },
    "id": 1
}
```

参考来源

CRegexp::get() in *ui/include/classes/api/services/CRegexp.php*.

用户

这个类用于操作用户。

对象属性:

- 用户
- 媒介

可用方法:

- user.checkauthentication -检查并延长用户会话
- user.create - 创建用户
- user.delete - 删除用户
- user.get - 查询用户
- user.login - 登录用户
- user.logout - 注销用户
- user.provision - 设置 LDAP 用户
- user.resettotp - 重置用户 TOTP 密码
- user.unblock - 解锁用户
- user.update - 更新用户

用户对象

以下对象都是与 `user` 相关的 API。

用户

用户对象具有以下属性。

Property	Type	Description
userid	ID	用户的 ID。 属性行为： - 只读 - 更新操作时必需

Property	Type	Description
username	string	用户的名称。 属性行为： - 创建操作时必需 - 对于已配置用户，如果用户已链接到用户目录（`userdirectoryid` 已设置为非"0" 的有效值），且用户目录配置状态已启用（用户目录对象的 `provision_status` 设置为"1"），并且所有 LDAP 或 SAML 配置的身份验证状态已启用（身份验证对象的 `ldap_jit_status` 设置为"Enabled for configured LDAP IdPs"，或身份验证对象的 `saml_jit_status` 设置为"Enabled for configured SAML IdPs"），则为只读
passwd	string	用户的密码。 如果用户已链接到用户目录，则此参数的值可以为空字符串。 属性行为： - 只写
roleid	ID	用户角色的 ID。 请注意，没有角色的用户只能使用LDAP 或SAML 身份验证登录 Zabbix，前提是其 LDAP/SAML 信息与 Zabbix 中配置的用户组映射相匹配。
attempt_clock	timestamp	上次登录失败尝试的时间。 属性行为： - 只读
attempt_failed	integer	最近登录失败尝试的次数。 属性行为： - 只读
attempt_ip	string	上次登录失败尝试的来源 IP 地址。 属性行为： - 只读
autologin	integer	是否启用自动登录。 可能的值： 0 - （默认）禁用自动登录； 1 - 启用自动登录。
autologout	string	用户会话生命周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h）。如果设置为 0s，会话将永不过期。 默认值：15m。
default_maintenance_period	string	通过快捷方式创建的维护的默认维护周期。接受秒数或带后缀的时间单位（300s、5m、1h、1d、1w）。给定值将向下取整为整分钟数。 默认值：1h。
lang	string	用户语言的语言代码，例如 `en_US`。 默认值：`default` - 系统默认值。
name	string	用户的姓名。
provisioned	integer	用户是否已配置。 可能的值： 0 - 未配置； 1 - 已配置。 属性行为： - 只读
refresh	string	自动刷新周期。接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、1h）。 默认值：30s。

Property	Type	Description
rows_per_page	integer	每页显示的对象行数。 默认值：50。
surname	string	用户的姓氏。
theme	string	用户的主题。 可能的值： `default` - （默认）系统默认值； `blue-theme` - 蓝色； `blue-classic-theme` - 蓝色（经典）； `dark-theme` - 深色； `dark-blue-theme` - 深蓝色； `dark-classic-theme` - 深色（经典）； `hc-light` - 高对比度浅色； `hc-dark` - 高对比度深色。
ts_provisioned	timestamp	最近一次配置操作的时间。 属性行为： - 只读 - 对于 *Super admin* 类型的用户支持
url	string	用户登录后重定向到的页面 URL。
userdirectoryid	ID	用户所链接到的用户目录的 ID。 用于配置（创建或更新），以及登录链接到用户目录的用户。 对于登录操作，此属性的值优先于用户所属用户组的 `userdirectoryid` 属性。 默认值：0。 属性行为： - 只读 - 对于 *Super admin* 类型的用户支持
timezone	string	用户的时区，例如 `Europe/London`、`UTC`。 默认值：`default` - 系统默认值。 有关支持的完整时区列表，请参阅 PHP 文档。

媒体

媒体对象具有以下属性。

Property	Type	Description
mediaid	ID	用户媒体的 ID。 属性行为： - 只读
mediatypeid	ID	用户媒体所使用的媒体类型的 ID。 属性行为： - 必需

Property	Type	Description
sendto	string/array	接收者的地址、用户名或其他标识符。 如果媒体类型的 type 设置为”Email” 或”Push”，则值表示为数组。对于”Push”，数组可以包含通配符 (*)，或一个或多个移动设备 ID。 如果媒体类型的 type 设置为”Script”、”SMS” 或”Webhook”，则值表示为字符串。 属性行为： - 必需
active	integer	媒体是否已启用。 可能的值： 0 - (默认) 已启用； 1 - 已禁用。
severity	integer	需要发送通知的触发器严重性。 可能的位图值： 1 - 未分类； 2 - 信息； 4 - 警告； 8 - 一般； 16 - 严重； 32 - 灾难。 这是一个位掩码字段；可以使用任意可能位图值的总和（例如，48 表示一般、严重和灾难）。 默认值：63。
period	string	可以发送通知的时间，格式为时间段或以分号分隔的用户宏。 默认值：1-7,00:00-24:00。
provisioned	integer	用户是否已配置。 可能的值： 0 - 未配置； 1 - 已配置。
userdirectory_mediaid	ID	已配置媒体的用户目录媒体映射 ID。 属性行为： - 只读 - 超级管理员类型用户支持

创建

描述

`object user.create(object/array users)`

此方法允许您创建新用户。

> **Note:**
> 此方法仅适用于超级管理员用户类型。可以在用户角色设置中撤销调用此方法的权限。有关更多信息，请参阅用户角色。

> **Note:**
> 用户密码的强度将根据 Authentication API 定义的密码策略规则进行验证。有关更多信息，请参阅authentication 对象。

参数

`(object/array)` 要创建的用户。

除了标准用户属性之外，该方法还接受以下参数。

参数	类型	描述
usrgrps	array	要将用户添加到的用户组。 用户组必须仅定义 `usrgrpid` 属性。
medias	array	要创建的用户介质。

返回值

(object) 返回一个包含创建值的 ID 的对象映射 `userids` 属性，返回的 ID 顺序与传入的用户顺序相匹配。

示例

创建用户

创建一个新用户，将其添加到用户组，并为其创建新的媒介。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "user.create",
    "params": {
        "username": "John",
        "passwd": "Doe12345",
        "roleid": "5",
        "usrgrps": [
            {
                "usrgrpid": "7"
            }
        ],
        "medias": [
            {
                "mediatypeid": "1",
                "sendto": [
                    "support@company.com"
                ],
                "active": 0,
                "severity": 63,
                "period": "1-7,00:00-24:00"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
        ]
    },
    "id": 1
}
```

另请参见

- 身份验证
- 媒介
- 用户组
- 角色

来源

CUser::create() in *ui/include/classes/api/services/CUser.php*.

删除

描述

`object user.delete(array users)`

此方法允许您删除用户。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

(array) 要删除用户的 ID。

返回值

(object) 返回一个带有 userids 属性（其中包含被删除用户 ID）的对象。

示例

批量删除用户

删除两个用户。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.delete",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

来源

CUser::delete() in *ui/include/classes/api/services/CUser.php*.

更新

描述

`object user.update(object/array users)`

此方法允许您更新现有用户。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

> **Note:**
> 用户密码强度会根据 Authentication API 定义的密码策略规则进行验证。有关更多信息，请参见Authentication API。

参数

(object/array) 要更新的用户属性。

必须为每个用户定义 userid 属性，其他所有属性均为可选。只有传递的属性会被更新，其他所有属性将保持不变。

除了标准用户属性之外，该方法还接受以下参数。

参数	类型	描述
current_passwd	string	用户当前的密码。 如果用户已关联到用户目录，则该参数的值可以为空字符串。 参数行为： - 仅写 - 如果设置了用户对象的 passwd 且用户正在修改自己的用户密码，则为必填
usrgrps	array	用于替换现有用户组的用户组。 用户组必须仅定义 usrgrpid 属性。
medias	array	用于替换现有的、非预配媒体的用户媒体。更新媒体时，可以省略已预配的媒体。

返回值

(object) 返回一个带有 userids 属性（其中包含被更新用户 ID）的对象。

示例

重命名用户

把用户重命名为 John Doe。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "1",
        "name": "John",
        "surname": "Doe"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1"
        ]
    },
    "id": 1
}
```

变更用户角色

变更一个用户的角色。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "12",
```

```
        "roleid": "6"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
        ]
    },
    "id": 1
}
```

参考

- 认证

来源

CUser::update() in *ui/include/classes/api/services/CUser.php*.

检查认证

描述

`object user.checkAuthentication`

此方法检查并延长用户会话。

Attention:
默认情况下，使用参数'sessionid' 调用'user.checkAuthentication' 会延长用户会话。

参数

该方法接受以下参数。

Parameter	Type	Description
extend	boolean	是否延长用户会话。 默认值："true"。 将该值设置为"false" 可在不延长会话的情况下检查用户会话。
sessionid	string	参数行为： - 如果 `sessionid` 已设置，则为 *supported* 用户authentication token。
token	string	参数行为： - 如果未设置 `token`，则为 *required* 用户API token。
		参数行为： - 如果未设置 `sessionid`，则为 *required*

返回值

(`object`) 返回一个包含用户信息的对象。

除标准用户属性外，还会返回以下信息。

属性	Type	描述
auth_type	integer	用户的默认认证方式。 有关可能的值列表，请参见认证对象的 authentication_type 属性。
debug_mode	integer	是否为用户启用或禁用调试模式。 有关可能的值列表，请参见用户组对象的 debug_mode 属性。
deprovisioned	boolean	用户是否属于已取消配置用户组。
gui_access	string	用户访问前端的认证方法。 有关可能的值列表，请参见用户组对象的 gui_access 属性。
secret	string	随机的 32 个字符字符串。在用户登录时生成。 如果使用 API token 检查用户会话，则不会返回 secret 属性。
sessionid	string	认证令牌，必须在后续 API 请求中使用。 如果使用 API token 检查用户会话，则不会返回 sessionid 属性。
type	integer	用户类型。 有关可能的值列表，请参见角色对象的 type 属性。
userip	string	用户的 IP 地址。

示例

使用认证令牌检查身份验证

使用用户认证令牌检查和延长用户会话，并返回有关用户的其他信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.checkAuthentication",
    "params": {
        "sessionid": "673b8ba11562a35da902c66cf5c23fa2"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "timezone": "Europe/Riga",
        "roleid": "3",
        "userdirectoryid": "0",
        "ts_provisioned": "0",
        "type": 3,
        "userip": "127.0.0.1",
```

```
        "debug_mode": 0,
        "gui_access": "0",
        "deprovisioned": false,
        "auth_type": 0,
        "sessionid": "673b8ba11562a35da902c66cf5c23fa2",
        "secret": "0e329b933e46984e49a5c1051ecd0751"
    },
    "id": 1
}
```

使用 API 令牌检查身份验证

使用用户 API 令牌检查用户会话，并返回有关该用户的附加信息。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "user.checkAuthentication",
    "params": {
        "token": "00aff470e07c12d707e50d98cfe39edef9e6ec349c14728dbdfbc8ddc5ea3eae"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919338",
        "rows_per_page": "50",
        "timezone": "Europe/Riga",
        "roleid": "3",
        "userdirectoryid": "0",
        "ts_provisioned": "0",
        "debug_mode": 0,
        "deprovisioned": false,
        "gui_access": "1",
        "mfaid": 0,
        "auth_type": 0,
        "type": 3,
        "userip": "127.0.0.1"
    },
    "id": 1
}
```

源代码

位于 *ui/include/classes/api/services/CUser.php* 中的 CUser::checkAuthentication()。

注销

描述

`string/object user.logout(array)`

此方法允许您退出 API，并使当前认证令牌失效。

> **Note:**
> 此方法适用于任何类型的用户。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 此方法接受一个空数组。

返回值

(boolean) 如果用户成功注销，返回 `true`。

示例

注销

通过 API 注销。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.logout",
    "params": [],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

参考

- 登录

来源

CUser::login() in *ui/include/classes/api/services/CUser.php*.

登录

描述

`string/object user.login(object parameters)`

此方法允许您登录到 API 并生成身份验证令牌。

> **Warning:**
> 使用此方法时，您还需要执行user.logout，以防止生成大量未关闭的会话记录。

> **Attention:**
> 此方法仅适用于未认证用户，且这些用户不属于任何启用了多因素认证的user group。

参数

(object) 包含用户名和密码的参数。

该方法接受以下参数。

参数	Type	描述
password	string	用户密码。 参数行为： - 必填
username	string	用户名。 参数行为： - 必填
userData	flag	返回已认证用户的信息。

返回值

(string/object) 如果使用了 userData 参数，则返回一个对象，其中包含有关已认证用户的信息。

除标准用户属性外，还会返回以下信息：

属性	类型	描述
auth_type	integer	用户的默认认证方式。 可能的值列表请参见认证对象的 authentication_type 属性。
debug_mode	integer	是否为用户启用调试模式。 可能的值列表请参见用户组对象的 debug_mode 属性。
deprovisioned	boolean	用户是否属于已取消配置用户组。
gui_access	string	用户访问前端的认证方式。 可能的值列表请参见用户组对象的 gui_access 属性。
mfaid	integer	用户登录时使用的MFA 方法 ID。 如果全局禁用了 MFA，或者用户所属的所有用户组都禁用了 MFA，则返回“0”。
secret	string	随机的 32 个字符字符串。在用户登录时生成。
sessionid	string	认证令牌，必须在后续 API 请求中使用。
type	integer	用户类型。 可能的值列表请参见角色对象的 type 属性。
userip	string	用户的 IP 地址。

Note:
如果用户在一次或多次认证失败后成功通过认证，该方法将返回 attempt_clock、attempt_failed 和 attempt_ip 属性的当前值，然后将其重置。

如果未使用 userData 参数，则该方法返回一个用于认证的认证令牌。

示例

单用户认证

单用户认证。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "username": "Admin",
        "password": "zabbix"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

请求已验证用户的信息

验证并返回有关用户的附加信息。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "username": "Admin",
        "password": "zabbix",
        "userData": true
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "timezone": "Europe/Riga",
        "roleid": "3",
        "type": 3,
        "debug_mode": 0,
        "userip": "127.0.0.1",
        "gui_access": "0",
        "sessionid": "5b56eee8be445e98f0bd42b435736e42"
    },
    "id": 1
}
```

参考

- 注销

来源

CUser::login() in *ui/include/classes/api/services/CUser.php*.

获取

描述

```
integer/array user.get(object parameters)
```

该方法允许你根据给定参数检索用户。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

> **Note:**
> 请求用户媒介、权限或角色信息时，*Admin* 和 *User* 类型的用户只能检索其自身用户的数据。示例请参见以 *Admin* 身份检索用户。

参数	类型	描述
mediaids	ID/array	仅返回使用给定媒介的用户。
mediatypeids	ID/array	仅返回使用给定媒介类型的用户。
userids	ID/array	仅返回具有给定 ID 的用户。
usrgrpids	ID/array	仅返回属于给定用户组的用户。
getAccess	flag	添加有关用户权限的附加信息。 为每个用户添加以下属性： gui_access - (integer) 用户的前端认证方式。可能的值列表请参见用户组对象的 gui_access 属性。 debug_mode - (integer) 指示是否为该用户启用调试。可能的值：0 - 禁用调试，1 - 启用调试。 users_status - (integer) 指示该用户是否被禁用。可能的值：0 - 用户已启用，1 - 用户已禁用。
selectMedias	query	在medias 属性中返回用户使用的媒介。
selectMediatypes	query	在mediatypes 属性中返回用户使用的媒介类型。 有关基于用户类型的限制，请参见mediatype.get。
selectUsrgrps	query	在usrgrps 属性中返回该用户所属的用户组。 有关基于用户类型的限制，请参见usergroup.get。
selectRole	query	在role 属性中返回用户角色。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 text 数据类型 的属性。 当 *Admin* 和 *User* 类型的用户请求其用户组内用户的数据时，可用的用户对象 属性为：userid、name、surname、username。
output	query	要返回的用户对象 属性。 *Admin* 和 *User* 类型的用户只能检索以下属性： - 对于其自身用户：userid、attempt_clock、attempt_failed、attempt_ip、autologin、autologout、lang、name、provisioned、refresh、roleid、rows_per_page、surname、theme、timezone、url、username。 - 对于其用户组中的用户：userid、name、surname、username。 默认值：extend。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，将执行 LIKE "%...%" 搜索。 仅支持 string 和 text 数据类型 的属性。 当 *Admin* 和 *User* 类型的用户请求其用户组内用户的数据时，可用的用户对象 属性为：name、surname、username。

参数	类型	描述
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`userid`、`username`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回:

- 一个对象数组；
- 检索对象的计数（使用 countOutput 参数时）。

示例

获取用户

获取所有已配置的用户。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "username": "Admin",
            "name": "Zabbix",
            "surname": "Administrator",
            "url": "",
            "autologin": "1",
            "autologout": "0",
            "lang": "en_US",
            "default_maintenance_period": "1h",
            "refresh": "30s",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50",
            "timezone": "default",
            "roleid": "3",
            "userdirectoryid": "0",
            "ts_provisioned": "0",
            "provisioned": "0"
        },
        {
```

```
            "userid": "2",
            "username": "guest",
            "name": "",
            "surname": "",
            "url": "",
            "autologin": "0",
            "autologout": "15m",
            "lang": "default",
            "default_maintenance_period": "1h",
            "refresh": "30s",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50",
            "timezone": "default",
            "roleid": "4",
            "userdirectoryid": "0",
            "ts_provisioned": "0",
            "provisioned": "0"
        }
    ],
    "id": 1
}
```

以 *Admin* 身份检索用户

作为 *Admin* 类型的用户，您可以检索自己的详细用户数据，以及用户组中其他用户的有限数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend",
        "getAccess": true,
        "selectMedias": "extend",
        "selectMediatypes": "extend",
        "selectUsrgrps": "extend",
        "selectRole": "extend"
    },
    "id": 1
}
```

响应：

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "userid": "1",
      "username": "Admin",
      "name": "Zabbix",
      "surname": "Administrator",
      "url": "",
      "autologin": "1",
      "autologout": "0",
      "lang": "default",
      "default_maintenance_period": "1h",
      "refresh": "30s",
      "theme": "default",
      "attempt_failed": "0",
      "attempt_ip": "",
      "attempt_clock": "0",
```

```
      "rows_per_page": "50",
      "timezone": "default",
      "roleid": "3",
      "userdirectoryid": "0",
      "ts_provisioned": "0",
      "provisioned": "0",
      "gui_access": "1",
      "debug_mode": "0",
      "users_status": "0",
      "usrgrps": [
        {
          "usrgrpid": "7",
          "name": "Zabbix administrators",
          "gui_access": "0",
          "users_status": "0",
          "debug_mode": "0",
          "userdirectoryid": "0",
          "mfa_status": "0",
          "mfaid": "0",
          "proxy_mode": "0",
          "proxy_group_mode": "0"
        },
        {
          "usrgrpid": "13",
          "name": "Internal",
          "gui_access": "1",
          "users_status": "0",
          "debug_mode": "0",
          "userdirectoryid": "0",
          "mfa_status": "0",
          "mfaid": "0",
          "proxy_mode": "0",
          "proxy_group_mode": "0"
        }
      ],
      "medias": [],
      "mediatypes": [],
      "role": {
        "roleid": "3",
        "name": "Super admin role",
        "type": "3",
        "readonly": "1"
      }
    },
    {
      "userid": "2",
      "username": "guest",
      "name": "",
      "surname": "",
      "url": "",
      "autologin": "0",
      "autologout": "15m",
      "lang": "default",
      "default_maintenance_period": "1h",
      "refresh": "30s",
      "theme": "default",
      "attempt_failed": "0",
      "attempt_ip": "",
      "attempt_clock": "0",
      "rows_per_page": "50",
      "timezone": "default",
      "roleid": "4",
```

```
      "userdirectoryid": "0",
      "ts_provisioned": "0",
      "provisioned": "0",
      "gui_access": "1",
      "debug_mode": "0",
      "users_status": "1",
      "usrgrps": [
        {
          "usrgrpid": "8",
          "name": "Guests",
          "gui_access": "0",
          "users_status": "0",
          "debug_mode": "0",
          "userdirectoryid": "0",
          "mfa_status": "0",
          "mfaid": "0",
          "proxy_mode": "0",
          "proxy_group_mode": "0"
        },
        {
          "usrgrpid": "9",
          "name": "Disabled",
          "gui_access": "0",
          "users_status": "1",
          "debug_mode": "0",
          "userdirectoryid": "0",
          "mfa_status": "0",
          "mfaid": "0",
          "proxy_mode": "0",
          "proxy_group_mode": "0"
        },
        {
          "usrgrpid": "13",
          "name": "Internal",
          "gui_access": "1",
          "users_status": "0",
          "debug_mode": "0",
          "userdirectoryid": "0",
          "mfa_status": "0",
          "mfaid": "0",
          "proxy_mode": "0",
          "proxy_group_mode": "0"
        }
      ],
      "medias": [],
      "mediatypes": [],
      "role": {
        "roleid": "4",
        "name": "Guest role",
        "type": "1",
        "readonly": "0"
      }
    }
  ],
  "id": 1
}
```


获取用户数据

获取 ID 为 “2” 的用户数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": ["userid", "username"],
        "selectRole": "extend",
        "userids": "2"
    },
    "id": 1
}
```

响应：

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "userid": "2",
      "username": "guest",
      "role": {
        "roleid": "4",
        "name": "Guest role",
        "type": "1",
        "readonly": "0"
      }
    }
  ],
  "id": 1
}
```

另请参阅

- 媒体
- 媒体类型
- 用户组
- 角色

来源

CUser::get() in *ui/include/classes/api/services/CUser.php*.

解锁

描述

`object user.unblock(array userids)`

此方法允许你解除用户的阻止状态。

> **Note:**
> 此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 需要解锁用户的 ID。

返回值

(object) 返回一个带有 userids 属性（其中包含被解锁用户 ID）的对象。

示例

批量解锁用户

解锁两个用户。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.unblock",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

来源

CUser::unblock() in *ui/include/classes/api/services/CUser.php*.

配置

描述

`object user.provision(object/array users)`

此方法允许你为 LDAP 用户进行配置。

> **Note:**
> 此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 要设置的用户的 id。

返回值

(object) 在 userids 属性下返回一个包含已配置用户 id 的对象。

示例

配置多用户

创建两个用户账号。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.provision",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

来源

CUser::provision() in *ui/include/classes/api/services/CUser.php*.

重置 **TOTP**

描述

`object user.resettotp(object/array users)`

此方法允许您重置用户的 TOTP 密钥。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要重置 TOTP 密码的用户的 ID。

> **Note:**
> 指定用户的用户会话也将被删除（发送请求的用户除外）。

返回值

(object) 此函数返回一个对象，该对象在其 `userids` 属性下包含了已重置 TOTP 密钥的用户 ID。

示例

批量重置用户 TOTP 密钥

为两个用户重置 TOTP 密钥。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "user.resettotp",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

参考

- MFA 对象

来源

CUser::resettotp() in *ui/include/classes/api/services/CUser.php*.

用户宏

此类用于处理主机级和全局用户宏。

对象引用：

- 全局宏
- 主机宏
    - 宏配置

可用方法：

- usermacro.create - 创建新的主机宏
- usermacro.createglobal - 创建新的全局宏
- usermacro.delete - 删除主机宏
- usermacro.deleteglobal - 删除全局宏
- usermacro.get - 检索主机宏和全局宏
- usermacro.update - 更新主机宏
- usermacro.updateglobal - 更新全局宏

用户宏对象

以下对象均与 `usermacro` 接口相关。

全局宏

全局宏对象具有以下属性。

属性	类型	描述
globalmacroid	ID	全局宏的 ID。 属性行为： - 只读 - 更新操作时必填
description	string	宏的描述。
macro	string	宏字符串。 属性行为： - 创建操作时必填
type	integer	宏的类型。 可能的值： 0 - （默认）文本宏； 1 - 密文宏； 2 - Vault 密钥。
value	string	宏的值。 属性行为： - 如果 `type` 设置为 “密文宏”，则为 只写 - 创建操作时必填

主机宏

主机宏对象定义了在主机、主机原型或模板上可用的宏。它具有以下属性。

属性	类型	描述
hostmacroid	ID	宏的 ID。 属性行为： - 只读 - 更新操作时必填
automatic	integer	定义该宏是否由发现规则控制。 可能的值： 0 - *(默认)* 宏由用户管理； 1 - 宏由发现规则管理。 不允许用户创建自动宏。 要更新自动宏，必须先将其转换为手动。
config	object/array	负责定义宏在主机向导中如何显示的宏配置。
description	string	宏的描述。
hostid	ID	该宏所属主机、主机原型或模板的 ID。 属性行为： - 常量 - 创建操作时必填
macro	string	宏字符串。 属性行为： - 创建操作时必填
type	integer	宏的类型。 可能的值： 0 - *(默认)* 文本宏； 1 - 密文宏； 2 - Vault 密钥。
value	string	宏的值。 属性行为： - 当 `type` 设置为 “密文宏” 时为 只写 - 创建操作时必填

宏配置

宏配置对象定义了宏在主机向导 中的显示方式。

Property	Type	Description
type	integer	宏输入字段的类型。 可能的值： 0 - 宏不在主机向导中使用； 1 - 文本框； 2 - 列表； 3 - 复选框。 Property behavior： - *required*
label	string	宏输入字段的标签。 Property behavior： - 如果 `type` 设置为 “Textbox”、“List” 或 “Checkbox”，则为 *required*
description	text	显示在宏输入字段旁边的帮助文本。支持 Markdown 格式。
priority	integer	宏输入字段在宏列表中的位置。 如果未设置 `priority`，则该宏会被添加到未分组宏列表的末尾。

Property	Type	Description
required	integer	将宏标记为必填。 可能的值： 0 - 非必填； 1 - 必填。 Property behavior： - 如果 type 设置为 “Textbox” 或 “List”，则 *supported*
regex	string	用于验证文本框字段中用户输入内容的正则表达式。 Property behavior： - 如果 type 设置为 “Textbox”，则 *supported*
section_name	string	宏所属可折叠分区的标签。 如果未设置 section_name，则该宏不分组。未分组的宏会优先显示；已分组的宏显示在下方，并在各自的可折叠分区内按其 priority 排序。
options	text	定义列表项或复选框值的 JSON 字符串。 对于列表：包含 value 和 text 属性的对象数组。 示例：[{"value": "http", "text": "HTTP"}, {"value": "https", "text": "HTTPS"}] 对于复选框：包含 checked 和 unchecked 属性的对象。 示例：{"checked": true, "unchecked": false} Property behavior： - 如果 type 设置为 “List” 或 “Checkbox”，则为 *required*

创建

描述

`object usermacro.create(object/array hostMacros)`

此方法允许您创建新的主机宏。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的主机宏。

该方法接受具有标准主机宏属性的主机宏。

返回值

(object) 返回包含 hostmacroids 属性（其中包含被创建主机宏的 ID）的对象. 返回的主机宏 ID 顺序与传入的主机宏顺序相同。

示例

创建主机宏

在主机”10198” 上创建值为”public” 的主机宏 {$SNMP_COMMUNITY}。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.create",
    "params": {
        "hostid": "10198",
        "macro": "{$SNMP_COMMUNITY}",
        "value": "public"
```

```
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "11"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::create() in *ui/include/classes/api/services/CUserMacro.php*.

创建全局宏

描述

`object usermacro.createglobal(object/array globalMacros)`

此方法允许您创建新的全局宏。

> **Note:**
> 此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要创建的全局宏。

此方法接受具有标准全局宏属性的全局宏。

返回值

(object) 返回包含 globalmacroids 属性（其中包含被创建全局宏的 ID）的对象，返回的主机宏 ID 顺序与传入的主机宏顺序相同。

示例

创建全局宏

创建一个值为"public" 的全局宏 {$SNMP_COMMUNITY}。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.createglobal",
    "params":  {
        "macro": "{$SNMP_COMMUNITY}",
        "value": "public"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "6"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::createGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

删除

描述

`object usermacro.delete(array hostMacroIds)`

此方法允许您删除主机宏。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除主机宏的 ID。

返回值

(object) 返回包含 hostmacroids 属性（其中包含被删除主机宏的 ID）的对象。

示例

批量删除主机宏

删除两个主机宏。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.delete",
    "params": [
        "32",
        "11"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::delete() in *ui/include/classes/api/services/CUserMacro.php*.

删除全局宏

描述

`object usermacro.deleteglobal(array globalMacroIds)`

此方法允许您删除全局宏。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 要删除的全局宏 ID。

返回值

(object) 返回包含 globalmacroids 属性（其中包含被删除全局宏的 ID）的对象。

示例

批量删除全局宏

删除两个全局宏。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.deleteglobal",
    "params": [
        "32",
        "11"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::deleteGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

更新

描述

`object usermacro.update(object/array hostMacros)`

此方法允许你更新现有的主机宏。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的主机宏属性。

必须为每个主机宏定义 hostmacroid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

返回值

(object) 返回包含 hostmacroids 属性（其中包含被更新主机宏的 ID）的对象。

示例

更改一个主机宏的值

更改一个主机宏"public" 的值。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.update",
    "params": {
        "hostmacroid": "1",
        "value": "public"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

更改自动发现规则创建的宏值

将 “automatic” 宏创建的发现规则转换为 “manual”，并将其值更改为 “new-value”。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.update",
    "params": {
        "hostmacroid": "1",
        "value": "new-value",
        "automatic": "0"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::update() in *ui/include/classes/api/services/CUserMacro.php*.

更新全局宏

描述

`object usermacro.updateglobal(object/array globalMacros)`

此方法允许您更新现有的全局宏。

> **Note:**
> 此方法仅适用于 超级管理员用户类型。调用此方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的全局宏属性。

必须为每个全局宏定义 globalmacroid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

返回值

(object) 返回包含 globalmacroids 属性（其中包含被更新全局宏的 ID）的对象。

示例

变更一个全局宏的值

变更全局宏"public" 的值。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.updateglobal",
    "params": {
        "globalmacroid": "1",
        "value": "public"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

来源

CUserMacro::updateGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

获取

描述

`integer/array usermacro.get(object parameters)`

该方法允许您根据给定参数检索主机和全局宏。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
globalmacro	flag	返回全局宏而不是主机宏。
globalmacroids	ID/array	仅返回具有给定 ID 的全局宏。
groupids	ID/array	仅返回属于给定主机组或模板组中的主机或模板的主机宏。
hostids	ID/array	仅返回属于给定主机或模板的宏。
hostmacroids	ID/array	仅返回具有给定 ID 的主机宏。
inherited	boolean	如果设置为 true，则仅返回从模板继承的主机原型用户宏。
selectHostGroups	query	在hostgroups 属性中返回主机宏所属的主机组。 仅在检索主机宏时使用。

参数	类型	描述
selectHosts	query	在hosts 属性中返回主机宏所属的主机。
selectTemplateGroups	query	仅在检索主机宏时使用。 在templategroups 属性中返回模板宏所属的模板组。
selectTemplates	query	仅在检索模板宏时使用。 在templates 属性中返回主机宏所属的模板。
sortfield	string/array	仅在检索主机宏时使用。 按给定属性对结果进行排序。
countOutput	boolean	可能的值：macro。 这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回二者之一:

- 一个对象列表;
- 检索对象的计数（使用 countOutput 参数时）。

示例

查询一个主机的主机宏

查询主机"10198" 所有的主机宏。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "hostids": "10198"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostmacroid": "9",
            "hostid": "10198",
            "macro": "{$INTERFACE}",
            "value": "eth0",
            "description": "",
            "type": "0",
            "automatic": "0"
        },
        {
```

```
            "hostmacroid": "11",
            "hostid": "10198",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public",
            "description": "",
            "type": "0",
            "automatic": "0"
        }
    ],
    "id": 1
}
```

检索模板的主机宏

检索为模板 “10265” 定义的、宏名称中包含 “STATUS” 的主机宏。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "hostids": "10265",
        "search": {
            "macro": "STATUS"
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostmacroid": "6709",
            "hostid": "10265",
            "macro": "{$APACHE.STATUS.HOST}",
            "value": "",
            "description": "Apache 状态页主机的主机名或 IP 地址。",
            "type": "0",
            "automatic": "0",
            "config": {
                "type": "1",
                "priority": "1",
                "section_name": "",
                "label": "Apache 状态主机",
                "description": "Apache 状态页主机的主机名或 IP 地址。",
                "required": "1",
                "regex": "",
                "options": []
            }
        },
        {
            "hostmacroid": "814",
            "hostid": "10265",
            "macro": "{$APACHE.STATUS.PATH}",
            "value": "server-status?auto",
            "description": "Apache 状态页的 URL 路径。",
            "type": "0",
            "automatic": "0",
            "config": {
                "type": "1",
```

```
                "priority": "3",
                "section_name": "",
                "label": "Apache 状态页路径",
                "description": "Apache 状态页的 URL 路径。",
                "required": "1",
                "regex": "",
                "options": []
            }
        },
        {
            "hostmacroid": "815",
            "hostid": "10265",
            "macro": "{$APACHE.STATUS.PORT}",
            "value": "80",
            "description": "Apache 状态页的端口。",
            "type": "0",
            "automatic": "0",
            "config": {
                "type": "1",
                "priority": "2",
                "section_name": "",
                "label": "Apache 状态页端口",
                "description": " 范围为 1 到 65535 (含)。",
                "required": "1",
                "regex": "^-?([0-9]+|(([0-9]+)\\.([0-9]+)))$",
                "options": []
            }
        },
        {
            "hostmacroid": "816",
            "hostid": "10265",
            "macro": "{$APACHE.STATUS.SCHEME}",
            "value": "http",
            "description": " 请求方案，可以是 HTTP 或 HTTPS。",
            "type": "0",
            "automatic": "0",
            "config": {
                "type": "2",
                "priority": "4",
                "section_name": "",
                "label": " 请求方案",
                "description": " 请求方案，可以是 HTTP 或 HTTPS。",
                "required": "0",
                "regex": "",
                "options": [
                    {
                        "value": "http",
                        "text": "HTTP"
                    },
                    {
                        "value": "https",
                        "text": "HTTPS"
                    }
                ]
            }
        }
    ],
    "id": 1
}
```


查询全局宏

查询所有的全局宏。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "globalmacro": true
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "globalmacroid": "6",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public",
            "description": "",
            "type": "0"
        }
    ],
    "id": 1
}
```

来源

CUserMacro::get() in *ui/include/classes/api/services/CUserMacro.php*.

用户目录

此类用于处理用户目录。

对象参考：

- 用户目录
  - 介质类型映射
  - Provisioning 组映射

可用方法：

- userdirectory.create - 创建新的用户目录
- userdirectory.delete - 删除用户目录
- userdirectory.get - 获取用户目录
- userdirectory.update - 更新用户目录
- userdirectory.test - 测试用户目录连接

用户目录对象

以下对象与 `userdirectory` API 直接相关。

用户目录

用户目录对象具有以下属性。

属性	类型	描述
userdirectoryid	ID	用户目录的 ID。 如果删除用户目录，则对于所有链接到已删除用户目录的用户，用户对象属性 `userdirectoryid` 的值将设置为”0”。 属性行为： - 只读 - 更新操作时必需
idp_type	integer	用户目录的身份提供商所使用的身份验证协议类型。 请注意，只能存在一个 SAML 类型的用户目录。 可能的值： 1 - LDAP 类型的用户目录； 2 - SAML 类型的用户目录。 属性行为： - 创建操作时必需
group_name	string	LDAP/SAML 用户目录属性，其中包含用于在 LDAP/SAML 用户目录与 Zabbix 之间映射组的组名称。 示例：*cn*
user_username	string	属性行为： - 当 `provision_status` 设置为”Enabled”，且身份验证对象的 `saml_jit_status` 设置为”Enabled for configured SAML IdPs” 时必需 LDAP/SAML 用户目录属性（如果 `scim_status` 设置为”SCIM provisioning is enabled”，则也包括 SCIM 属性），其中包含用户名称。配置用户时，该名称将用作用户对象属性 `name` 的值。 示例：*cn*、*commonName*、*displayName*、*name*
user_lastname	string	LDAP/SAML 用户目录属性（如果 `scim_status` 设置为”SCIM provisioning is enabled”，则也包括 SCIM 属性），其中包含用户姓氏。配置用户时，该姓氏将用作用户对象属性 `surname` 的值。 示例：*sn*、*surname*、*lastName*
provision_status	integer	用户目录的配置状态。 可能的值： 0 - *(默认)* 已禁用（已禁用配置此用户目录创建的用户）； 1 - 已启用（已启用配置此用户目录创建的用户；此外，还必须启用 LDAP 或 SAML 配置（身份验证对象的 `ldap_jit_status` 或 `saml_jit_status`）。）
provision_groups	array	用于将 LDAP/SAML 用户组模式映射到 Zabbix 用户组和用户角色的配置组映射对象数组。 属性行为： - 当 `provision_status` 设置为”Enabled” 时必需
provision_media	array	用于将用户的 LDAP/SAML 媒体属性（例如电子邮件）映射到 Zabbix 用户媒体，以发送通知的媒体类型映射对象数组。
LDAP 专用属性：		
name	string	用户目录的唯一名称。 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时必需

属性	类型	描述
host	string	LDAP 服务器的主机名、IP 或 URI。 URI 必须包含架构（`ldap://` 或 `ldaps://`）、主机和端口（可选）。 示例： *host.example.com* *127.0.0.1* *ldap://ldap.example.com:389* 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时必需
port	integer	LDAP 服务器的端口。 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时必需
base_dn	string	LDAP 用户目录中用户账户的基本路径。 示例： *ou=Users,dc=example,dc=org* *ou=Users,ou=system*（适用于 OpenLDAP） *DC=company,DC=com*（适用于 Microsoft Active Directory） *uid=%{user},dc=example,dc=com*（适用于直接用户绑定；占位符”*%{user}*” 为必需） 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时必需
search_attribute	string	LDAP 用户目录属性，用于根据登录请求中提供的信息识别用户账户。 示例： *uid*（适用于 OpenLDAP） *sAMAccountName*（适用于 Microsoft Active Directory） 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时必需
bind_dn	string	用于绑定并搜索 LDAP 服务器的 LDAP 服务器账户。 对于直接用户绑定和匿名绑定，`bind_dn` 必须为空。 示例： *uid=ldap_search,ou=system*（适用于 OpenLDAP） *CN=ldap_search,OU=user_group,DC=company,DC=com*（适用于 Microsoft Active Directory） *CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local* 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时支持
bind_password	string	用于绑定并搜索 LDAP 服务器的账户的 LDAP 密码。 对于直接用户绑定和匿名绑定，`bind_password` 必须为空。 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时支持
description	string	用户目录的描述。 属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时支持

属性	类型	描述
group_basedn	string	LDAP 用户目录中组的基本路径；用于配置 LDAP 用户目录中的用户成员资格检查。 如果设置了 `group_membership`，则在配置用户时忽略此属性。 示例：*ou=Groups,dc=example,dc=com* 属性行为： - 当 `idp_type` 设置为"User directory of type LDAP" 时支持
group_filter	string	用于检索用户所属 LDAP 用户目录组的过滤器字符串；用于配置 LDAP 用户目录中的用户成员资格检查。 如果设置了 `group_membership`，则在配置用户时忽略此属性。 支持的 `group_filter` 占位符： *%{attr}* - 搜索属性（替换为 `search_attribute` 属性值）； *%{groupattr}* - 组属性（替换为 `group_member` 属性值）； *%{host}* - LDAP 服务器的主机名、IP 或 URI（替换为 `host` 属性值）； *%{user}* - Zabbix 用户名。 默认值：*(%{groupattr}=%{user})* 示例： - *(member=uid=%{ref},ou=Users,dc=example,dc=com)*：如果 LDAP 组对象包含值为"*uid=User1,ou=Users,dc=example,dc=com*" 的"*member*" 属性，则匹配"User1"，并返回"User1" 所属的组； - *(%{groupattr}=cn=%{ref},ou=Users,ou=Zabbix,DC=example,DC=com)*：如果 LDAP 组对象包含 `group_member` 属性中指定的属性，且其值为"*cn=User1,ou=Users,ou=Zabbix,DC=example,DC=com*"，则匹配"User1"，并返回"User1" 所属的组。 属性行为： - 当 `idp_type` 设置为"User directory of type LDAP" 时支持
group_member	string	LDAP 用户目录属性，其中包含有关组成员的信息；用于配置 LDAP 用户目录中的用户成员资格检查。 如果设置了 `group_membership`，则在配置用户时忽略此属性。 属性行为： - 当 `idp_type` 设置为"User directory of type LDAP" 时支持
group_membership	string	LDAP 用户目录属性，其中包含有关用户所属组的信息。 示例：*memberOf* 属性行为： - 当 `idp_type` 设置为"User directory of type LDAP" 时支持
search_filter	string	用于根据登录请求中提供的信息，在 LDAP 用户目录中定位并验证用户身份的自定义过滤器字符串。 支持的 `search_filter` 占位符： *%{attr}* - 搜索属性名称（例如 *uid*、*sAMAccountName*）； *%{user}* - Zabbix 用户名。 默认值：*(%{attr}=%{user})* 属性行为： - 当 `idp_type` 设置为"User directory of type LDAP" 时支持

属性	类型	描述
start_tls	integer	LDAP 服务器配置选项，允许使用传输层安全性（TLS）保护与 LDAP 服务器之间的通信。 请注意，对于使用 `ldaps://` 协议的主机，`start_tls` 必须设置为”Disabled”。 可能的值： 0 - *(默认)* 已禁用； 1 - 已启用。
user_ref_attr	string	属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时支持 用于引用用户对象的 LDAP 用户目录属性。`user_ref_attr` 的值用于从用户目录中指定的属性获取值，并将其替换到 `group_filter` 字符串中的 *%{ref}* 占位符。 示例：*cn*、*uid*、*member*、*uniqueMember*
SAML 专用属性：		属性行为： - 当 `idp_type` 设置为”User directory of type LDAP” 时支持
idp_entityid	string	用于标识身份提供商，并用于在 SAML 消息中与身份提供商通信的 URI。 示例：*https://idp.example.com/idp*
sp_entityid	string	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时必需 用于标识身份提供商服务提供商的 URL 或任意字符串。 示例： *https://idp.example.com/sp* *zabbix*
username_attribute	string	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时必需 SAML 用户目录属性（如果 `scim_status` 设置为”SCIM provisioning is enabled”，则也包括 SCIM 属性），其中包含用户的用户名。身份验证时，该用户名将与用户对象属性 `username` 的值进行比较。 示例：*uid*、*userprincipalname*、*samaccountname*、*username*、*userusername*、*urn:oid:0.9.2342.19200300.100.1.1*、*urn:oid:1.3.6.1.4.1.5923.1.1.1.13*、*urn:oid:0.9.2342.19200300.100.1.44*
sso_url	string	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时必需 身份提供商的 SAML 单点登录服务 URL，Zabbix 将向该 URL 发送 SAML 身份验证请求。 示例：*http://idp.example.com/idp/sso/saml*
slo_url	string	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时必需 身份提供商的 SAML 单点注销服务 URL，Zabbix 将向该 URL 发送 SAML 注销请求。 示例：*https://idp.example.com/idp/slo/saml*
		属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持

属性	类型	描述
encrypt_nameid	integer	是否应加密 SAML 名称 ID。 可能的值： 0 - *(默认)* 不加密名称 ID； 1 - 加密名称 ID。
encrypt_assertions	integer	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持 是否应加密 SAML 断言。 可能的值： 0 - *(默认)* 不加密断言； 1 - 加密断言。
nameid_format	string	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持 SAML 身份提供商服务提供商的名称 ID 格式。 示例： *urn:oasis:names:tc:SAML:2.0:nameid-format:persistent* *urn:oasis:names:tc:SAML:2.0:nameid-format:transient* *urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos* *urn:oasis:names:tc:SAML:2.0:nameid-format:entity*
scim_status	integer	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持 是否启用或禁用 SAML 的 SCIM 配置。 可能的值： 0 - *(默认)* SCIM 配置已禁用； 1 - SCIM 配置已启用。
sign_assertions	integer	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持 是否应使用 SAML 签名对 SAML 断言进行签名。 可能的值： 0 - 不对断言进行签名； 1 - *(默认)* 对断言进行签名。
sign_authn_requests	integer	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持 是否应使用 SAML 签名对 SAML AuthN 请求进行签名。 可能的值： 0 - 不对 AuthN 请求进行签名； 1 - *(默认)* 对 AuthN 请求进行签名。
sign_messages	integer	属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持 是否应使用 SAML 签名对 SAML 消息进行签名。 可能的值： 0 - 不对消息进行签名； 1 - *(默认)* 对消息进行签名。 属性行为： - 当 `idp_type` 设置为”User directory of type SAML” 时支持

属性	类型	描述
sign_logout_requests	integer	是否应使用 SAML 签名对 SAML 注销请求进行签名。 可能的值： 0 - 不对注销请求进行签名； 1 - *(默认)* 对注销请求进行签名。 属性行为： - 当 `idp_type` 设置为"User directory of type SAML" 时支持
sign_logout_responses	integer	是否应使用 SAML 签名对 SAML 注销响应进行签名。 可能的值： 0 - 不对注销响应进行签名； 1 - *(默认)* 对注销响应进行签名。 属性行为： - 当 `idp_type` 设置为"User directory of type SAML" 时支持
idp_certificate	string	用于设置 SAML 单点登录（SSO）服务的服务提供商（SP）证书内容。 示例： *-----BEGIN CERTIFICATE----- ...more encoded data... -----END CERTIFICATE-----* 属性行为： - 当 `idp_type` 设置为"User directory of type SAML"，且前端配置文件中的 `$SSO['CERT_STORAGE']` 设置为 `database` 时支持 - 只写
sp_private_key	string	用于设置 SAML 单点登录（SSO）服务的服务提供商（SP）私钥内容。 用于与身份提供商（IdP）进行安全身份验证和数据交换。 示例： *-----BEGIN CERTIFICATE----- ...more encoded data... -----END CERTIFICATE-----* 属性行为： - 当 `idp_type` 设置为"User directory of type SAML"，且前端配置文件中的 `$SSO['CERT_STORAGE']` 设置为 `database` 时支持 - 只写
sp_certificate	string	用于设置 SAML 单点登录（SSO）服务的服务提供商（SP）证书内容。 示例： *-----BEGIN CERTIFICATE----- ...more encoded data... -----END CERTIFICATE-----* 属性行为： - 当 `idp_type` 设置为"User directory of type SAML"，且前端配置文件中的 `$SSO['CERT_STORAGE']` 设置为 `database` 时支持 - 只写
idp_certificate_hash	string	`idp_certificate` 值的 md5 哈希。对于空的 `idp_certificate`，将返回空字符串。 属性行为： - 当 `idp_type` 设置为"User directory of type SAML"，且前端配置文件中的 `$SSO['CERT_STORAGE']` 设置为 `database` 时支持 - 只读

属性	类型	描述
sp_private_key_hash	string	sp_private_key 值的 md5 哈希。对于空的 sp_private_key，将返回空字符串。 属性行为： - 当 idp_type 设置为"User directory of type SAML"，且前端配置文件中的 $SSO['CERT_STORAGE'] 设置为 database 时支持 - 只读
sp_certificate_hash	string	sp_certificate 值的 md5 哈希。对于空的 sp_certificate，将返回空字符串。 属性行为： - 当 idp_type 设置为"User directory of type SAML"，且前端配置文件中的 $SSO['CERT_STORAGE'] 设置为 database 时支持 - 只读

媒介类型映射

媒介类型映射对象具有以下属性。

属性	类型	描述
userdirectory_mediaid	ID	媒介类型映射 ID。 属性行为: - 只读
name	string	在媒介类型映射列表中显示的名称。 属性行为: - 必填
mediatypeid	ID	要创建的媒介类型 ID；用作Media object 属性 mediatypeid 的值。 属性行为: - 必填
attribute	string	包含用户媒介的 LDAP/SAML 用户目录属性（如果 scim_status 设置为 "已启用 SCIM 配置"，也可以是 SCIM 属性，例如 *user@example.com*），该值将用作Media object 属性 sendto 的值。 如果该属性存在于从 LDAP/SAML 身份提供方接收的数据中，且其值非空，则会为已配置的用户触发媒介创建。 属性行为: - 必填
active	integer	为已配置的用户创建媒介时，用户媒介 active 属性的值。 可能的值： 0 - (默认) 已启用； 1 - 已禁用。
severity	integer	为已配置的用户创建媒介时，用户媒介 severity 属性的值。 默认值：63。
period	string	为已配置的用户创建媒介时，用户媒介 period 属性的值。 默认值：1-7,00:00-24:00。

Provisioning 组映射

Provisioning 组映射具有以下属性。

属性	类型	描述
name	string	LDAP/SAML 用户目录中的组全名（例如 *Zabbix administrators*）（如果 `scim_status` 设置为 “已启用 SCIM provisioning”，也包括 SCIM）。 支持通配符字符 “*”。 在所有 provisioning 组映射中必须唯一。 属性行为： - 必填
roleid	ID	要分配给用户的用户角色 ID。 如果匹配到多个 provisioning 组映射，则将最高用户类型（*User*、*Admin* 或 *Super admin*）的角色分配给该用户。如果存在多个具有相同用户类型的角色，则将第一个角色（按字母顺序排序）分配给该用户。 属性行为： - 必填
user_groups	array	Zabbix 用户组 ID 对象数组。每个对象具有以下属性： `usrgrpid` - (ID) 要分配给用户的 Zabbix 用户组 ID。 如果匹配到多个 provisioning 组映射，则会将所有匹配映射中的 Zabbix 用户组分配给该用户。 属性行为： - 必填

创建

描述

`object userdirectory.create(object/array userDirectory)`

此方法允许您创建新的用户目录。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。

参数

`(object/array)` 要创建的用户目录。该方法接受具有标准用户目录属性的用户目录。

返回值

`(object)` 返回一个对象，该对象包含在 `userdirectoryids` 属性下创建的用户目录 id。返回的 id 的顺序与传递的用户目录的顺序匹配。

示例

创建一个用户目录

创建一个用户目录，以通过 LDAP 使用 StartTLS 对用户进行身份验证。请注意，通过 LDAP 对用户进行身份验证时，LDAP authentication必须启用。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.create",
    "params": {
        "idp_type": "1",
        "name": "LDAP API server #1",
        "host": "ldap://local.ldap",
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "ldapsecretpassword",
        "search_attribute": "uid",
        "start_tls": "1"
```

```
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "3"
        ]
    },
    "id": 1
}
```

创建一个用户目录 (已启用 JIT 设置)

创建用户目录以通过 LDAP 对用户进行身份验证 (已启用 JIT 设置)。请注意，通过 LDAP 对用户进行身份验证时，LDAP 身份认证必须开启。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.create",
    "params": {
            "idp_type": "1",
            "name": "AD server",
            "provision_status": "1",
            "description": "",
            "host": "host.example.com",
            "port": "389",
            "base_dn": "DC=zbx,DC=local",
            "search_attribute": "sAMAccountName",
            "bind_dn": "CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local",
            "start_tls": "0",
            "search_filter": "",
            "group_basedn": "OU=Zabbix,DC=zbx,DC=local",
            "group_name": "CN",
            "group_member": "member",
            "group_filter": "(%{groupattr}=CN=%{ref},OU=Users,OU=Zabbix,DC=zbx,DC=local)",
            "group_membership": "",
            "user_username": "givenName",
            "user_lastname": "sn",
            "user_ref_attr": "CN",
            "provision_media": [
                {
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "user@example.com"
                }
            ],
            "provision_groups": [
                {
                    "name": "*",
                    "roleid": "4",
                    "user_groups": [
                        {
                            "usrgrpid": "8"
                        }
                    ]
                },
                {
                    "name": "Zabbix administrators",
                    "roleid": "2",
```

```
                    "user_groups": [
                        {
                            "usrgrpid": "7"
                        },
                        {
                            "usrgrpid": "8"
                        }
                    ]
                }
            ]
        },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "2"
        ]
    },
    "id": 1
}
```

来源

CUserDirectory::create() in *ui/include/classes/api/services/CUserDirectory.php*.

删除

描述

`object userdirectory.delete(array userDirectoryIds)`

此方法允许您删除用户目录。当用户目录至少被一个用户组直接使用时，无法删除该用户目录。
 当 `authentication.ldap_configured` 设置为 1，或者剩余的用户目录数量不止一个时，默认的 LDAP 用户目录无法删除。

Note:
此方法仅适用于 超级管理员用户类型。

参数

要删除的用户目录的 （array）id。

返回值

（object）返回一个对象，该对象包含 `userdirectoryids` 属性下已删除用户目录的 id。

示例

批量删除用户目录

删除两个用户目录。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.delete",
    "params": [
        "2",
        "12"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "2",
            "12"
        ]
    },
    "id": 1
}
```

来源

CUserDirectory::delete() in *ui/include/classes/api/services/CUserDirectory.php*.

更新

描述

`object userdirectory.update(object/array userDirectory)`

此方法允许您更新现有的用户目录。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。

参数

要更新的 (object/array)用户目录属性。

必须为每个用户目录定义 userdirectoryid 属性，其他所有属性都是可选的。注意只有给定的属性将被更新，其他所有属性将保持不变。

返回值

(object) 在 userdirectoryids 属性下，返回一个包含更新的用户目录 id 的对象。

示例

更新用户目录的绑定密码

为用户目录设置新的绑定密码。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.update",
    "params": {
        "userdirectoryid": "3",
        "bind_password": "newldappassword"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "3"
        ]
    },
    "id": 1
}
```

更新用户目录的映射

更新用户目录”2” 的预配置组映射和媒介类型映射。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.update",
    "params": {
        "userdirectoryid": "2",
        "provision_media": [
                {
                    "userdirectory_mediaid": "2"
                }
            ],
            "provision_groups": [
                {
                    "name": "Zabbix administrators",
                    "roleid": "2",
                    "user_groups": [
                        {
                            "usrgrpid": "7"
                        },
                        {
                            "usrgrpid": "8"
                        },
                        {
                            "usrgrpid": "11"
                        }
                    ]
                }
            ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "2"
        ]
    },
    "id": 1
}
```

来源

CUserDirectory::update() in *ui/include/classes/api/services/CUserDirectory.php*.

测试

描述

`object userdirectory.test(array userDirectory)`

此方法允许您测试用户目录连接设置。

Note:
此方法还允许您测试已配置的数据与用户目录设置在用户 provisioning 中的匹配情况（例如，将为用户分配什么用户角色、用户组、用户媒介）。对于这种类型的测试，API 请求应针对 `provision_status` 设置为启用的user directory 发起。

Note:
此方法仅适用于 *Super admin* 用户类型。

参数

(object) 用户目录属性。

由于 userdirectory.get API 不返回 bind_password 字段，因此应提供 userdirectoryid 和/或 bind_password。
 除标准用户目录属性外，该方法还接受以下参数。

参数	类型	说明
test_username	string	要在用户目录中测试的用户名。
test_password	string	与该用户名关联、用于在用户目录中测试的密码。

返回值

(bool) 成功时返回 true。

示例

测试现有用户的用户目录

使用"user1" 测试用户目录"3"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "3",
        "host": "127.0.0.1",
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "search_attribute": "uid",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "password",
        "test_username": "user1",
        "test_password": "password"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

测试非现有用户的用户目录

测试非现有用户 "user2" 的用户目录"3"。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "3",
        "host": "127.0.0.1",
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "search_attribute": "uid",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "test_username": "user2",
        "test_password": "password"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32500,
        "message": "Application error.",
        "data": "Incorrect user name or password or account is temporarily blocked."
    },
    "id": 1
}
```

测试用户配置的用户目录

测试用户目录"3"，了解配置的数据与"user3" 设置的用户目录设置相匹配 (例如，什么用户角色、用户组、用户媒介将被分配给用户)。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "2",
        "host": "host.example.com",
        "port": "389",
        "base_dn": "DC=zbx,DC=local",
        "search_attribute": "sAMAccountName",
        "bind_dn": "CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local",
        "test_username": "user3",
        "test_password": "password"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "username": "user3",
        "name": "John",
        "surname": "Doe",
        "medias": [],
        "usrgrps": [
            {
                "usrgrpid": "8"
            },
            {
                "usrgrpid": "7"
            }
        ],
        "roleid": "2",
        "userdirectoryid": "2"
    },
    "id": 1
}
```

来源

CUserDirectory::test() in *ui/include/classes/api/services/CUserDirectory.php*.

获取

描述

`integer/array userdirectory.get(object parameters)`

该方法允许您根据给定参数检索用户目录。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
userdirectoryids	ID/array	仅返回具有给定 ID 的用户目录。
selectUsrgrps	query	返回 usrgrps 属性，其中包含与用户目录关联的用户组。 支持 count。
selectProvisionMedia	query	返回 provision_media 属性，其中包含与用户目录关联的媒体类型映射。
selectProvisionGroups	query	返回 provision_groups 属性，其中包含与用户目录关联的预配组映射。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：name。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值可以是单个值或值数组。 支持的属性：userdirectoryid、idp_type、provision_status。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称，值为要搜索的字符串。如果未提供其他选项，将执行 LIKE "%…%" 搜索。 支持的属性：name、description。 SAML 类型的用户目录，其 name 和 description 属性值都将为空。这两个属性都可以通过 userdirectory.update 操作进行更改。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组
- 如果使用了 countOutput 参数，则返回已检索对象的数量

示例

检索用户目录

检索具有其他属性的所有用户目录，这些属性显示与每个用户目录关联的介质类型映射和调配组映射。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.get",
    "params": {
```

```
        "output": "extend",
        "selectProvisionMedia": "extend",
        "selectProvisionGroups": "extend"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
       {
            "userdirectoryid": "1",
            "idp_type": "2",
            "name": "",
            "provision_status": "1",
            "description": "",
            "group_name": "groups",
            "user_username": "",
            "user_lastname": "",
            "idp_entityid": "http://example.com/simplesaml/saml2/idp/metadata.php",
            "sso_url": "http://example.com/simplesaml/saml2/idp/SSOService.php",
            "slo_url": "",
            "username_attribute": "uid",
            "sp_entityid": "zabbix",
            "nameid_format": "",
            "sign_messages": "0",
            "sign_assertions": "0",
            "sign_authn_requests": "0",
            "sign_logout_requests": "0",
            "sign_logout_responses": "0",
            "encrypt_nameid": "0",
            "encrypt_assertions": "0",
            "scim_status": "1",
            "provision_media": [
                {
                    "userdirectory_mediaid": "1",
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "user@example.com",
                    "active": "0",
                    "severity": "63",
                    "period": "1-7,00:00-24:00"
                }
            ],
            "provision_groups": [
                {
                    "name": "*",
                    "roleid": "1",
                    "user_groups": [
                        {
                            "usrgrpid": "13"
                        }
                    ]
                }
            ]
        },
        {
            "userdirectoryid": "2",
            "idp_type": "1",
            "name": "AD server",
            "provision_status": "1",
```

```
            "description": "",
            "host": "host.example.com",
            "port": "389",
            "base_dn": "DC=zbx,DC=local",
            "search_attribute": "sAMAccountName",
            "bind_dn": "CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local",
            "start_tls": "0",
            "search_filter": "",
            "group_basedn": "OU=Zabbix,DC=zbx,DC=local",
            "group_name": "CN",
            "group_member": "member",
            "group_filter": "(%{groupattr}=CN=%{ref},OU=Users,OU=Zabbix,DC=zbx,DC=local)",
            "group_membership": "",
            "user_username": "givenName",
            "user_lastname": "sn",
            "user_ref_attr": "CN",
            "provision_media": [
                {
                    "userdirectory_mediaid": "2",
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "user@example.com",
                    "active": "0",
                    "severity": "63",
                    "period": "1-7,00:00-24:00"
                }
            ],
            "provision_groups": [
                {
                    "name": "*",
                    "roleid": "4",
                    "user_groups": [
                        {
                            "usrgrpid": "8"
                        }
                    ]
                },
                {
                    "name": "Zabbix administrators",
                    "roleid": "2",
                    "user_groups": [
                        {
                            "usrgrpid": "7"
                        },
                        {
                            "usrgrpid": "8"
                        }
                    ]
                }
            ]
        },
        {
            "userdirectoryid": "3",
            "idp_type": "1",
            "name": "LDAP API server #1",
            "provision_status": "0",
            "description": "",
            "host": "ldap://local.ldap",
            "port": "389",
            "base_dn": "ou=Users,dc=example,dc=org",
            "search_attribute": "uid",
            "bind_dn": "cn=ldap_search,dc=example,dc=org",
```

```
            "start_tls": "1",
            "search_filter": "",
            "group_basedn": "",
            "group_name": "",
            "group_member": "",
            "group_filter": "",
            "group_membership": "",
            "user_username": "",
            "user_lastname": "",
            "user_ref_attr": "",
            "provision_media": [],
            "provision_groups": []
        }
    ],
    "id": 1
}
```


参考

- 用户组

来源

CUserDirectory::get() in *ui/include/classes/api/services/CUserDirectory.php*.

用户组

此类用于处理用户组。

对象参考：

- 用户组
- 权限
- 基于标签的权限

可用方法：

- usergroup.create - 创建新的用户组
- usergroup.delete - 删除用户组
- usergroup.get - 检索用户组
- usergroup.update - 更新用户组

用户组对象

以下对象与 `usergroup` 接口相关。

用户组

用户组对象具有以下属性。

Property	Type	Description
usrgrpid	ID	用户组的 ID。 属性行为： - 只读 - 更新操作中为必需
name	string	用户组的名称。 属性行为： - 创建操作中为必需
debug_mode	integer	是否启用调试模式。 可能的值： 0 - （默认）禁用； 1 - 启用。

Property	Type	Description
gui_access	integer	组中用户的前端身份验证方法。 可能的值： 0 - （默认）使用系统默认身份验证方法； 1 - 使用内部身份验证； 2 - 使用 LDAP 身份验证； 3 - 禁止访问前端。
mfa_status	integer	是否为组中的用户启用 MFA。 可能的值： 0 - 禁用（对于所有已配置的 MFA 方法）； 1 - 启用（对于所有已配置的 MFA 方法）。
mfaid	ID	组中用户使用的MFA 方法。 属性行为： - 对于超级管理员类型的用户支持 - 如果身份验证对象的 `mfa_status` 设置为 “已启用”，则支持
proxy_mode	integer	处理对未包含在 proxy 组中的 proxy 的访问的模式。 可能的值： 0 - 拒绝列表； 1 - （默认）允许列表。 属性行为： - 对于超级管理员类型的用户支持。
proxy_group_mode	integer	处理对 proxy 组（及其 proxy）的访问的模式。 可能的值： 0 - 拒绝列表； 1 - （默认）允许列表。 属性行为： - 对于超级管理员类型的用户支持。
users_status	integer	是否启用用户组。 对于已取消配置的用户，无法启用用户组。 可能的值： 0 - （默认）启用； 1 - 禁用。
userdirectoryid	ID	用于身份验证的用户目录的 ID。 属性行为： - 对于超级管理员类型的用户支持 - 如果 `gui_access` 设置为 “使用系统默认身份验证方法” 或 “使用 LDAP 身份验证”，则支持

权限

权限对象具有以下属性。

属性	类型	描述
id	ID	要添加权限的主机组或模板组的 ID。 属性行为: - 必填

属性	类型	描述
permission	integer	主机组或模板组的访问级别。 可能的值： 0 - 拒绝访问； 2 - 只读访问； 3 - 读写访问。 属性行为: - 对于创建操作为 必填

基于标签的权限

基于标签的权限对象具有以下属性。

属性	Type	说明
groupid	ID	要添加权限的主机组 ID。 Property behavior: - *required*
tag	string	标签名称。
value	string	标签值。

创建

描述

`object usergroup.create(object/array userGroups)`

此方法允许您创建新的用户组。

Note:
此方法仅适用于 *Super admin* 用户类型。在用户角色设置中，可以撤销调用此方法的权限。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的用户组。

除标准用户组属性外，该方法还接受以下参数。

参数	类型	描述
hostgroup_rights	object/array	要分配给用户组的主机组权限。
proxies	object/array	要关联到用户组的 proxy。 proxy 只能定义 `proxyid` 属性。
proxy_groups	object/array	要关联到用户组的 proxy 组。 proxy 组只能定义 `proxy_groupid` 属性。
templategroup_rights	object/array	要分配给用户组的模板组权限。
tag_filters	array	要分配给用户组的基于标签的权限。
users	object/array	要添加到用户组的用户。 用户只能定义 `userid` 属性。

返回值

`(object)` 返回一个对象，其中在 `usrgrpids` 属性下包含已创建用户组的 ID。返回的 ID 顺序与传入的用户组顺序一致。

示例

创建一个用户组

创建一个用户组，禁止其访问主机组"2"，并向其添加一个用户。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.create",
    "params": {
        "name": "Operation managers",
        "hostgroup_rights": {
            "id": "2",
            "permission": 0
        },
        "users": [
            {
                "userid": "12"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20"
        ]
    },
    "id": 1
}
```

参考

- 权限

源代码

位于 *ui/include/classes/api/services/CUserGroup.php* 中的 CUserGroup::create()。

删除

描述

`object usergroup.delete(array userGroupIds)`

此方法允许你删除用户组。

Attention:
已停用 用户组（在Authentication 中为 `disabled_usrgrpid` 指定的用户组）不能被删除。

Note:
此方法仅适用于 超级管理员用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(array)` 需要删除的用户组 ID。

返回值

`(object)` 返回一个带有 `usrgrpids` 属性（其中包含被删除用户组 ID）的对象。

示例

批量删除用户组

删除两个用户组

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.delete",
    "params": [
        "20",
        "21"
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20",
            "21"
        ]
    },
    "id": 1
}
```

来源

CUserGroup::delete() in *ui/include/classes/api/services/CUserGroup.php*.

更新

描述

`object usergroup.update(object/array userGroups)`

此方法允许您更新现有的用户组。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的用户组属性。

必须为每个用户组定义 usrgrpid 属性，所有其他属性均为可选属性。仅更新传递的属性，其他属性保持不变。

除标准用户组属性外，该方法还接受以下参数。

参数	类型	描述
hostgroup_rights	object/array	用于替换当前分配给用户组的主机组权限。
proxies	object/array	要关联到用户组的 proxy。 proxy 只能定义 proxyid 属性。
proxy_groups	object/array	要关联到用户组的 proxy 组。 proxy 组只能定义 proxy_groupid 属性。
templategroup_rights	object/array	用于替换当前分配给用户组的模板组权限。
tag_filters	array	用于替换当前分配给用户组的基于标签的权限。
users	object/array	用于替换当前分配给用户组的用户。 用户只能定义 userid 属性。

返回值

(object) 返回一个带有 usrgrpids 属性（其中包含被更新用户组 ID）的对象。

示例

启用用户组并更新权限

启用一个用户组，并为其提供对主机组 “2” 和 “4” 的读写权限。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.update",
    "params": {
        "usrgrpid": "17",
        "users_status": "0",
        "hostgroup_rights": [
            {
                "id": "2",
                "permission": 3
            },
            {
                "id": "4",
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17"
        ]
    },
    "id": 1
}
```

参考

- 权限

来源

CUserGroup::update() in *ui/include/classes/api/services/CUserGroup.php*.

获取

描述

`integer/array usergroup.get(object parameters)`

该方法允许您根据给定参数检索用户组。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
mfaids	ID/array	仅返回使用给定 MFA 方法的用户组。 参数行为： - 对 *Super admin* 类型的用户支持

参数	类型	描述
mfa_status	integer	仅返回具有给定 MFA 状态的用户组。 有关支持的状态列表，请参阅用户组页面。
proxyids	ID/array	仅返回包含给定 proxy 的用户组。 参数行为： - 对 *Super admin* 类型的用户支持。
proxy_groupids	ID/array	仅返回包含给定 proxy 组的用户组。 参数行为： - 对 *Super admin* 类型的用户支持。
selectProxies	ID/array	返回包含用户组 proxy 的 `proxies` 属性。 参数行为： - 对 *Super admin* 类型的用户支持。
selectProxyGroups	ID/array	返回包含用户组 proxy 组的 `proxy_groups` 属性。 参数行为： - 对 *Super admin* 类型的用户支持。
status	integer	仅返回具有给定状态的用户组。 有关支持的状态列表，请参阅用户组页面。
userids	ID/array	仅返回包含给定用户的用户组。
usrgrpids	ID/array	仅返回具有给定 ID 的用户组。
selectTagFilters	query	在`tag_filters` 属性中返回基于标签的用户组权限。
selectUsers	query	在`users` 属性中返回用户组中的用户。 有关基于用户类型的限制，请参阅`user.get`。
selectHostGroupRights	query	在 `hostgroup_rights` 属性中返回用户组的主机组权限。 有关主机组访问级别的列表，请参阅用户组页面。
selectTemplateGroupRights	query	在 `templategroup_rights` 属性中返回用户组的模板组权限。 有关模板组访问级别的列表，请参阅用户组页面。
limitSelects	integer	限制子查询返回的记录数。
output	query	要返回的用户组对象属性。 *Admin* 和 *User* 类型的用户只能获取以下属性：`usrgrpid`、`name`、`gui_access`、`users_status`、`debug_mode`、`mfa_status`。 默认值：`extend`。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`usrgrpid`、`name`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回结果二选一:

- 一个对象数组;
- 检索对象的计数（使用 `countOutput` 参数时）。

示例

检索已启用的用户组

检索所有已启用的用户组。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.get",
    "params": {
        "output": "extend",
        "status": 0
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "usrgrpid": "7",
            "name": "Zabbix administrators",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1",
            "userdirectoryid": "0",
            "mfa_status": "0",
            "mfaid": "0"
        },
        {
            "usrgrpid": "8",
            "name": "Guests",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0",
            "mfa_status": "0",
            "mfaid": "0"
        },
        {
            "usrgrpid": "11",
            "name": "Enabled debug mode",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1",
            "userdirectoryid": "0",
            "mfa_status": "0",
            "mfaid": "0"
        },
        {
            "usrgrpid": "12",
            "name": "No access to the frontend",
            "gui_access": "2",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0",
            "mfa_status": "0",
            "mfaid": "0"
        },
        {
            "usrgrpid": "14",
            "name": "Read only",
```

```
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0",
            "mfa_status": "0",
            "mfaid": "0"
        },
        {
            "usrgrpid": "18",
            "name": "Deny",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0",
            "mfa_status": "0",
            "mfaid": "0"
        }
    ],
    "id": 1
}
```


参考

- 用户

来源

CUserGroup::get() in *ui/include/classes/api/services/CUserGroup.php*.

监控项

此类旨在用于处理监控项。

对象引用：

- 监控项
  - HTTP 标头
  - HTTP 查询字段
- 监控项标签
- 监控项预处理

可用方法：

- item.create - 创建新的监控项
- item.delete - 删除监控项
- item.get - 检索监控项
- item.update - 更新监控项

监控项对象

以下对象直接与 `item` API 相关。

监控项

Note:
无法通过 Zabbix API 直接创建、更新或删除 Web 监控项。

监控项对象具有以下属性。

属性	类型	描述
itemid	ID	监控项的 ID。 属性行为： - 只读 - 更新操作中为必需
delay	string	监控项的更新间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或者可选地接受一个或多个自定义间隔，所有值以分号分隔。自定义间隔可以混合使用灵活间隔和调度间隔。 接受用户宏。如果使用用户宏，该值必须是单个宏。不支持多个宏，也不支持宏与文本混合使用。灵活间隔可以写成由正斜杠分隔的两个宏（例如 {$FLEX_INTERVAL}/{$FLEX_PERIOD}）。 示例： 1h;wd1-5h9-18;{$Macro1}/1-7,00:00-24:00;0/6-7,12:00-24:00;{$Macr 属性行为： - 当 type 设置为 “Zabbix agent”(0)、“Simple check”(3)、“Zabbix internal”(5)、“External check”(10)、“Database monitor”(11)、“IPMI agent”(12)、“SSH agent”(13)、“TELNET agent”(14)、“Calculated”(15)、“JMX agent”(16)、“HTTP agent”(19)、“SNMP agent”(20)、“Script”(21)、“Browser”(22) 或 “Telemetry query”(24) 时为必需；或者当 type 设置为 “Zabbix agent (active)”(7) 且 key_ 不包含 “mqtt.get” 时为必需
hostid	ID	监控项所属主机或模板的 ID。 属性行为： - 常量 - 创建操作中为必需
interfaceid	ID	监控项主机接口的 ID。 属性行为： - 当监控项属于主机且 type 设置为 “Zabbix agent”、“IPMI agent”、“JMX agent”、“SNMP trap” 或 “SNMP agent” 时为必需 - 当监控项属于主机且 type 设置为 “Simple check”、“External check”、“SSH agent”、“TELNET agent” 或 “HTTP agent” 时支持 - 对于发现的对象为只读
key_	string	监控项键值。 属性行为： - 创建操作中为必需 - 对于继承的对象或发现的对象为只读
name	string	监控项名称。 支持用户宏。 属性行为： - 创建操作中为必需 - 对于继承的对象或发现的对象为只读
name_resolved	string	解析用户宏后的监控项名称。 属性行为： - 只读

属性	类型	描述
type	integer	监控项类型。 可能的值： 0 - Zabbix agent； 2 - Zabbix trapper； 3 - Simple check； 5 - Zabbix internal； 7 - Zabbix agent (active)； 9 - Web item； 10 - External check； 11 - Database monitor； 12 - IPMI agent； 13 - SSH agent； 14 - TELNET agent； 15 - Calculated； 16 - JMX agent； 17 - SNMP trap； 18 - Dependent item； 19 - HTTP agent； 20 - SNMP agent； 21 - Script； 22 - Browser； 24 - Telemetry query。 属性行为： - 创建操作中为必需 - 对于继承的对象或发现的对象为只读
url	string	URL 字符串。 支持用户宏、{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}。 属性行为： - 当 type 设置为 “HTTP agent” 时为必需 - 对于继承的对象或发现的对象为只读
value_type	integer	监控项的信息类型。 可能的值： 0 - 数值（浮点）； 1 - 字符； 2 - 日志； 3 - 数值（无符号）； 4 - 文本； 5 - 二进制； 6 - JSON。 属性行为： - 创建操作中为必需 - 对于继承的对象或发现的对象为只读
allow_traps	integer	允许以类似 trapper 监控项的方式填充值。 0 - （默认）不接受传入数据； 1 - 接受传入数据。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于发现的对象为只读

属性	类型	描述
authtype	integer	身份验证方法。 当 type 设置为 “SSH agent” 时的可能值： 0 - （默认）密码； 1 - 公钥。 当 type 设置为 “HTTP agent” 时的可能值： 0 - （默认）无； 1 - basic； 2 - NTLM； 3 - Kerberos； 4 - Digest。 属性行为： - 当 type 设置为 “SSH agent” 或 “HTTP agent” 时支持 - 对于继承的对象（当 type 设置为 “HTTP agent” 时）或发现的对象为只读
description	string	监控项描述。 属性行为： - 对于发现的对象为只读
error	string	更新监控项值时发生问题的错误文本。 属性行为： - 只读
flags	integer	监控项的来源。 可能的值： 0 - 普通监控项； 4 - 从原型转换而来的监控项。 属性行为： - 只读
follow_redirects	integer	轮询数据时跟随响应重定向。 可能的值： 0 - 不跟随重定向； 1 - （默认）跟随重定向。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
headers	array	执行 HTTP 请求时发送的标头数组。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
history	string	历史数据应保留多长时间的时间单位。 也接受用户宏。 默认值：31d。 属性行为： - 对于发现的对象为只读
http_proxy	string	HTTP(S) proxy 连接字符串。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读

属性	类型	描述
inventory_link	integer	由监控项填充的主机资产清单字段 ID。 支持的主机资产清单字段及其 ID，请参阅主机资产清单页面。 默认值：0。 属性行为： - 当 `value_type` 设置为 “numeric float”、“character”、“numeric unsigned” 或 “text” 时支持 - 对于发现的对象为只读
ipmi_sensor	string	IPMI 传感器。 属性行为： - 当 `type` 设置为 “IPMI agent” 且 `key_` 未设置为 “ipmi.get” 时为必需 - 当 `type` 设置为 “IPMI agent” 时支持 - 对于继承的对象或发现的对象为只读
jmx_endpoint	string	JMX agent 自定义连接字符串。 默认值： service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi 属性行为： - 当 `type` 设置为 “JMX agent” 时支持 - 对于发现的对象为只读
lastclock	timestamp	监控项值上次更新的时间。 默认情况下，仅显示最近 24 小时内的值。您可以在“Administration” > “General”菜单部分中更改 *Max history display period* 参数的值，以延长此时间段。 属性行为： - 只读
lastns	integer	监控项值上次更新时的纳秒数。 默认情况下，仅显示最近 24 小时内的值。您可以在“Administration” > “General”菜单部分中更改 *Max history display period* 参数的值，以延长此时间段。 属性行为： - 只读
lastvalue	string	监控项的最新值。 默认情况下，仅显示最近 24 小时内的值。您可以在“Administration” > “General”菜单部分中更改 *Max history display period* 参数的值，以延长此时间段。 属性行为： - 只读
logtimefmt	string	日志条目中的时间格式。 属性行为： - 当 `value_type` 设置为 “log” 时支持 - 对于继承的对象或发现的对象为只读
master_itemid	ID	主监控项的 ID。 属性行为： - 当 `type` 设置为 “Dependent item” 时为必需 - 对于继承的对象或发现的对象为只读

属性	类型	描述
output_format	integer	是否应将响应转换为 JSON。 0 - （默认）存储原始数据； 1 - 转换为 JSON。
params	string	属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读 取决于监控项类型的其他参数： - SSH agent 和 TELNET agent 监控项要执行的脚本； - 数据库监控项的 SQL 查询； - 计算监控项的公式； - Script 和 Browser 监控项的脚本。
parameters	object/array	属性行为： - 当 type 设置为 “Database monitor”、“SSH agent”、“TELNET agent”、“Calculated”、“Script” 或 “Browser” 时为必需 - 对于继承的对象（当 type 设置为 “Script” 或 “Browser” 时）或发现的对象为只读 当 type 设置为 “Script” 或 “Browser” 时的其他参数。包含具有 name 和 value 属性的对象数组，其中 name 必须唯一。
password	string	属性行为： - 当 type 设置为 “Script” 或 “Browser” 时支持 - 对于继承的对象或发现的对象为只读 身份验证密码。
post_type	integer	属性行为： - 当 type 设置为 “JMX agent” 且设置了 username 时为必需 - 当 type 设置为 “Simple check”、“SSH agent”、“TELNET agent”、“Database monitor” 或 “HTTP agent” 时支持 - 对于继承的对象（当 type 设置为 “HTTP agent” 时）或发现的对象为只读 posts 属性中存储的 POST 数据正文类型。 可能的值： 0 - （默认）原始数据； 2 - JSON 数据； 3 - XML 数据。
posts	string	属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读 HTTP(S) 请求正文数据。
prevvalue	string	属性行为： - 当 type 设置为 “HTTP agent” 且 post_type 设置为 “JSON data” 或 “XML data” 时为必需 - 当 type 设置为 “HTTP agent” 且 post_type 设置为 “Raw data” 时支持 - 对于继承的对象或发现的对象为只读 监控项的上一个值。 默认情况下，仅显示最近 24 小时内的值。您可以在“Administration” > “General”菜单部分中更改 *Max history display period* 参数的值，以延长此时间段。 属性行为： - 只读

属性	类型	描述
privatekey	string	私钥文件名称。 属性行为： - 当 type 设置为 “SSH agent” 且 authtype 设置为 “public key” 时为必需 - 对于发现的对象为只读
publickey	string	公钥文件名称。 属性行为： - 当 type 设置为 “SSH agent” 且 authtype 设置为 “public key” 时为必需 - 对于发现的对象为只读
query_fields	array	执行 HTTP 请求时发送的查询字段数组。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
request_method	integer	请求方法类型。 可能的值： 0 - （默认）GET； 1 - POST； 2 - PUT； 3 - HEAD。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
retrieve_mode	integer	应存储响应的哪一部分。 当 request_method 设置为 “GET”、“POST” 或 “PUT” 时的可能值： 0 - （默认）正文； 1 - 标头； 2 - 同时存储正文和标头。 当 request_method 设置为 “HEAD” 时的可能值： 1 - 标头。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
snmp_oid	string	SNMP OID。 属性行为： - 当 type 设置为 “SNMP agent” 时为必需 - 对于继承的对象或发现的对象为只读
ssl_cert_file	string	SSL 公钥文件路径。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
ssl_key_file	string	SSL 私钥文件路径。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
ssl_key_password	string	SSL 私钥文件密码。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读

属性	类型	描述
state	integer	监控项状态。 可能的值： 0 - （默认）正常； 1 - 不支持。 属性行为： - 只读
status	integer	监控项状态。 可能的值： 0 - （默认）已启用的监控项； 1 - 已禁用的监控项。
status_codes	string	以逗号分隔的所需 HTTP 状态码范围。 也支持将用户宏作为逗号分隔列表的一部分。 示例：200,200-{$M},{$M},200-400 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
templateid	ID	父模板监控项的 ID。 提示：使用 hostid 属性指定监控项所属的模板。 属性行为： - 只读
timeout	string	监控项数据轮询请求超时。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 300s、1m）。也接受用户宏。 可能的值范围：1-600s。 默认值：”” - 使用 proxy/全局设置。 属性行为： - 当 type 设置为 “Zabbix agent”(0)、“Simple check”(3) 且 key_ 不以 “vmware.” 和 “icmpping” 开头、“Zabbix agent (active)”(7)、“External check”(10)、“Database monitor”(11)、“SSH agent”(13)、“TELNET agent”(14)、“HTTP agent”(19)、“SNMP agent”(20) 且 snmp_oid 以 “walk[” 或 “get[” 开头、“Script”(21) 或 “Browser”(22) 时支持 - 对于继承的对象和发现的对象为只读； - 如果应用默认值且监控项属于由不可访问的 proxy 或 proxy 组监控的主机，则使用全局超时。
trapper_hosts	string	允许的主机。 属性行为： - 对于发现的对象为只读 - 当 type 设置为 “Zabbix trapper”，或者当 type 设置为 “HTTP agent” 且 allow_traps 设置为 “Allow incoming data” 时支持。 如果省略该属性，则会自动将其设置为 {$TRAPPER.ALLOWED_HOSTS} 用户宏（默认值）。默认情况下，{$TRAPPER.ALLOWED_HOSTS} 为 127.0.0.1,::1。 如果该属性设置为空字符串，则将拒绝来自所有主机的传入连接。
trends	string	趋势数据应保留多长时间的时间单位。 也接受用户宏。 默认值：365d。 属性行为： - 当 value_type 设置为 “numeric float” 或 “numeric unsigned” 时支持 - 对于发现的对象为只读

属性	类型	描述
units	string	值单位。 属性行为： - 当 `value_type` 设置为 “numeric float” 或 “numeric unsigned” 时支持 - 对于继承的对象或发现的对象为只读
username	string	身份验证用户名。 属性行为： - 当 `type` 设置为 “SSH agent”、“TELNET agent”，或者当 `type` 设置为 “JMX agent” 且设置了 `password` 时为必需 - 当 `type` 设置为 “Simple check”、“Database monitor” 或 “HTTP agent” 时支持 - 对于继承的对象（当 `type` 设置为 “HTTP agent” 时）或发现的对象为只读
uuid	string	通用唯一标识符，用于将导入的监控项链接到已存在的监控项。如果未提供，则自动生成。 属性行为： - 当监控项属于模板时支持
valuemapid	ID	关联值映射的 ID。 属性行为： - 当 `value_type` 设置为 “numeric float”、“character” 或 “numeric unsigned” 时支持 - 对于继承的对象或发现的对象为只读
verify_host	integer	是否验证连接的主机名是否与主机证书中的名称匹配。 可能的值： 0 - （默认）不验证； 1 - 验证。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
verify_peer	integer	是否验证主机证书是否真实有效。 可能的值： 0 - （默认）不验证； 1 - 验证。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对于继承的对象或发现的对象为只读
query	object	Telemetry query 配置。定义监控项从遥测数据源查询的内容、筛选数据的方式以及聚合结果的方式。 属性行为： - 当 `type` 设置为 “Telemetry query” 时为必需 - 对于继承的对象或发现的对象为只读
time_shift	string	监控项查询的时间范围结束于当前时间之前多久。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m、2h、1d）。也接受用户宏。 可能的值范围：0-86400s。 默认值：15s。 属性行为： - 当 `type` 设置为 “Telemetry query” 时支持

属性	类型	描述
lookback_limit	string	当没有可继续使用的上一次查询（例如监控项首次运行时），或上一次查询早于 `delay`（例如由于 Zabbix 服务器停止运行）时，监控项向当前时间之前追溯多长时间来查询数据。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m、2h、1d）。也接受用户宏。 可能的值范围：1-259200s。 默认值：10m。 属性行为： - 当 `type` 设置为 “Telemetry query” 时支持
granularity	string	查询的完整时间范围被拆分成的时间片段长度。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m、2h、1d）。也接受用户宏。 可能的值范围：1-86400s。 默认值：15s。 属性行为： - 当 `type` 设置为 “Telemetry query” 时支持

HTTP 标头

标头对象具有以下属性：

属性	类型	说明
name	string	HTTP 标头名称。 属性行为： - 必填
value	string	标头值。 属性行为： - 必填

HTTP 查询字段

查询字段对象定义了用于指定 URL 参数的名称和值。它具有以下属性：

属性	类型	描述
name	string	参数名称。 属性行为： - 必填
value	string	参数值。 属性行为： - 必填

遥测查询

遥测查询对象具有以下属性。

属性	类型	描述
signal_type	integer	监控项查询的 Signal 类型。 可能的值： 0 - Traces； 1 - Metrics； 2 - Logs。 属性行为： - 必需
metric_point_type	integer	监控项查询的 Metric point 类型。 可能的值： 0 - Sum； 1 - Gauge； 2 - Histogram； 3 - Exponential histogram。 属性行为： - 当 `signal_type` 设置为”Metrics” 时必需
columns	array	监控项查询的遥测数据列数组。可以为空。 属性行为： - 必需
aggregated_columns	array	监控项查询的遥测数据聚合列数组，以及如何根据列记录计算值。 属性行为： - 必需
filter	object	查询中列记录的遥测查询过滤器。 属性行为： - 必需

遥测数据列

遥测数据列对象具有以下属性。

Property	Type	Description
column	string	监控项查询的遥测数据列名称。 请参阅支持的遥测数据列。 对于多个单值列，每个 `column` 值必须唯一。对于包含键值对的多个列，每个 `column` 和 `attribute_key` 对必须唯一。
attribute_key	string	监控项在包含键值对的列中查询的键名称。 对于包含键值对的多个列，每个 `column` 和 `attribute_key` 对必须唯一。 属性行为： - 如果 `column` 设置为包含键值对的列，则为必需

遥测数据聚合列

遥测数据聚合列对象具有以下属性。

属性	类型	描述
column	string	监控项查询的遥测数据列名称。 请参阅支持的遥测数据列。 对于多个列，每个 `column` 值必须唯一。 如果将 `function` 设置为”Count”，则不支持此属性。 属性行为： - 如果将 `function` 设置为”Min”、”Max”、”Average”、”Sum”或”Percentile”，则为必需
function	integer	监控项应用于列记录的函数。 可能的值： 1 - Min； 2 - Max； 3 - Average； 4 - Count； 5 - Sum； 8 - Percentile。 属性行为： - 必需
parameters	array	监控项根据列记录计算的百分位数。 可能的值：0 到 100 之间的数字，以字符串形式给出。 属性行为： - 如果将 `function` 设置为”Percentile”，则为必需
alias	string	计算值的唯一名称，用于在监控项值中标识该值。 每个 `alias` 值必须唯一。 属性行为： - 必需

遥测数据过滤器

遥测数据过滤器对象具有以下属性。

属性	类型	描述
conditions	array	遥测数据过滤器条件，遥测数据列中的记录必须满足这些条件。 属性行为： - 必需
evaltype	integer	过滤器条件的评估方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 属性行为： - 必需

属性	类型	描述
eval_formula	string	用于评估过滤器条件的生成表达式。该表达式包含通过其 `formulaid` 引用特定过滤器条件的 ID。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 属性行为： - 只读
formula	string	用户定义的表达式，用于评估使用自定义表达式的过滤器条件。该表达式必须包含通过其 `formulaid` 引用特定过滤器条件的 ID。表达式中使用的 ID 必须与过滤器条件中定义的 ID 完全匹配：不能有任何条件未使用或被省略。 属性行为： - 当 `evaltype` 设置为 “自定义表达式” 时为必需

遥测数据过滤条件

遥测数据过滤条件对象具有以下属性。

属性	类型	描述
column	string	过滤条件所应用的遥测数据列的名称。 请参阅支持的遥测数据列。 属性行为： - 必需
attribute_key	string	在包含键值对的列中，条件所应用的键的名称。 属性行为： - 如果 `column` 设置为包含键值对的列，则为必需
operator	integer	过滤条件运算符。 单值列的可能值： 0 - 等于； 1 - 不等于； 2 - 包含； 3 - 不包含。 包含键值对的列的可能值： 0 - 等于； 1 - 不等于； 12 - 存在。 属性行为： - 必需
value	string	用于比较的值。 属性行为： - 如果 `operator` 设置为 “等于”、“不等于”、“包含” 或 “不包含”，则为必需
formulaid	string	用于从自定义表达式引用条件的任意唯一 ID。只能包含大写字母。修改过滤条件时，ID 必须由用户定义；但之后请求这些条件时，ID 将重新生成。 属性行为： - 如果遥测查询过滤器对象的 `evaltype` 设置为 “自定义表达式”，则为必需

监控项标签

监控项标签对象具有以下属性。

属性	Type	描述
tag	string	监控项标签名称。 属性行为： - 必填
value	string	监控项标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板； 1 - 主机。 属性行为： - 只读。
objectid	ID	标签继承自的对象的 ID。 属性行为： - 只读。

监控项预处理

监控项预处理对象具有以下属性。

属性	类型	描述
type	integer	预处理选项类型。 可能的值： 1 - 自定义乘数； 2 - 右侧修剪； 3 - 左侧修剪； 4 - 修剪； 5 - 正则表达式； 6 - 布尔值转十进制； 7 - 八进制转十进制； 8 - 十六进制转十进制； 9 - 简单变化； 10 - 每秒变化； 11 - XML XPath； 12 - JSONPath； 13 - 范围检查； 14 - 匹配正则表达式； 15 - 不匹配正则表达式； 16 - 检查 JSON 中的错误； 17 - 检查 XML 中的错误； 18 - 使用正则表达式检查错误； 19 - 丢弃未更改的值； 20 - 使用心跳丢弃未更改的值； 21 - JavaScript； 22 - Prometheus 模式； 23 - Prometheus 转 JSON； 24 - CSV 转 JSON； 25 - 替换； 26 - 检查不受支持的值； 27 - XML 转 JSON； 28 - SNMP walk 值； 29 - SNMP walk 转 JSON； 30 - SNMP get 值。 属性行为： - 必需
params	string	预处理选项使用的其他参数。 多个参数使用换行符 (\n) 分隔。 如果 `type` 设置为 “检查不受支持的值”，参数遵循 `<scope>[\n<pattern>]` 语法，其中 *pattern* 是正则表达式，*scope* 为以下值之一： -1 - 匹配任意错误； 0 - 检查错误消息是否匹配 *pattern*； 1 - 检查错误消息是否不匹配 *pattern*。 属性行为： - 当 `type` 设置为 “自定义乘数”(1)、“右侧修剪”(2)、“左侧修剪”(3)、“修剪”(4)、“正则表达式”(5)、“XML XPath”(11)、“JSONPath”(12)、“范围检查”(13)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“使用正则表达式检查错误”(18)、“使用心跳丢弃未更改的值”(20)、“JavaScript”(21)、“Prometheus 模式”(22)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“替换”(25)、“检查不受支持的值”(26)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时必需

属性	类型	描述
error_handler	integer	预处理步骤失败时使用的操作类型。 可能的值： 0 - 由 Zabbix 服务器设置错误消息； 1 - 丢弃值； 2 - 设置自定义值； 3 - 设置自定义错误消息。 当 type 设置为 “检查不受支持的值” 时的可能值： 1 - 丢弃值； 2 - 设置自定义值； 3 - 设置自定义错误消息。 属性行为： - 当 type 设置为 “自定义乘数”(1)、“正则表达式”(5)、“布尔值转十进制”(6)、“八进制转十进制”(7)、“十六进制转十进制”(8)、“简单变化”(9)、“每秒变化”(10)、“XML XPath”(11)、“JSONPath”(12)、“范围检查”(13)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“使用正则表达式检查错误”(18)、“Prometheus 模式”(22)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“检查不受支持的值”(26)、“XML 转 JSON”(27)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时必需
error_handler_params	string	错误处理程序参数。 属性行为： - 当 error_handler 设置为 “设置自定义值” 或 “设置自定义错误消息” 时必需

以下参数和错误处理程序适用于每种预处理类型。

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
1	自定义乘数	数字 ^1, 6^			0, 1, 2, 3
2	右侧修剪	字符列表[2]			
3	左侧修剪	字符列表[2]			
4	修剪	字符列表[2]			
5	正则表达式	模式[3]	输出[2]		0, 1, 2, 3
6	布尔值转十进制				0, 1, 2, 3
7	八进制转十进制				0, 1, 2, 3
8	十六进制转十进制				0, 1, 2, 3
9	简单变化				0, 1, 2, 3
10	每秒变化				0, 1, 2, 3
11	XML XPath	路径[4]			0, 1, 2, 3

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
12	JSONPath	路径[4]			0, 1, 2, 3
13	范围检查	最小值 ^1, 6^	最大值 ^1, 6^		0, 1, 2, 3
14	匹配正则表达式	模式[3]			0, 1, 2, 3
15	不匹配正则表达式	模式[3]			0, 1, 2, 3
16	检查 JSON 中的错误	路径[4]			0, 1, 2, 3
17	检查 XML 中的错误	路径[4]			0, 1, 2, 3
18	使用正则表达式检查错误	模式[3]	输出[2]		0, 1, 2, 3
19	丢弃未更改的值				
20	使用心跳丢弃未更改的值	秒 ^5, 6^			
21	JavaScript	脚本[2]			
22	Prometheus 模式	模式 ^6, 7^	`value`、`label`、`function`	输出 ^8, 9^	0, 1, 2, 3
23	Prometheus 转 JSON	模式 ^6, 7^			0, 1, 2, 3
24	CSV 转 JSON	字符[2]	字符[2]	0,1	0, 1, 2, 3
25	替换	搜索字符串[2]	替换字符串[2]		
26	检查不受支持的值	范围[1]	模式 ^3, 6^		1, 2, 3
27	XML 转 JSON				0, 1, 2, 3

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
28	SNMP walk 值	OID[2]	格式： 0 - 不变 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数		0, 1, 2, 3
29	SNMP walk 转 JSON[10]	字段名称[2]	OID 前缀[2]	格式： 0 - 不变 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数	0, 1, 2, 3
30	SNMP get 值	格式： 1 - 从 Hex-STRING 转换为 UTF-8 2 - 从 Hex-STRING 转换为 MAC 3 - 从 BITS 转换为整数			0, 1, 2, 3

[1] 浮点数（整数会隐式转换为浮点值）
 [2] 字符串
 [3] 正则表达式
 [4] JSONPath 或 XML XPath
 [5] 正整数（支持时间后缀，例如 30s、1m、2h、1d）
 [6] 用户宏
 [7] 遵循以下语法的 Prometheus 模式：`<metric name>{<label name>="<label value>", ...} == <value>`。Prometheus 模式的每个组成部分（指标、标签名称、标签值和指标值）都可以是用户宏。
 [8] 遵循以下语法的 Prometheus 输出：如果选择 `label` 作为第二个参数，则为 `<label name>`（可以是用户宏）。
 [9] 如果选择 `function` 作为第二个参数，则为以下聚合函数之一：`sum`、`min`、`max`、`avg`、`count`。
 [10] 支持多条“字段名称、OID 前缀、格式记录”，记录之间使用换行符分隔。

创建

描述

`object item.create(object/array items)`

此方法允许您创建新的监控项。

Note:
无法通过 Zabbix API 创建 Web 监控项。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的监控项。

除了标准监控项属性外，该方法还接受以下参数。

参数	Type	说明
preprocessing	array	监控项预处理选项。

参数	Type	说明
tags	array	监控项标签。

返回值

(object) 返回一个对象，其中在 itemids 属性下包含已创建监控项的 ID。返回的 ID 顺序与传入的监控项顺序一致。

示例

创建监控项

创建一个数值型 Zabbix agent 监控项，并使用 2 个监控项标签来监控 ID 为"30074" 的主机上的可用磁盘空间。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on /home/joe/",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "30074",
        "type": 0,
        "value_type": 3,
        "interfaceid": "30084",
        "tags": [
            {
                "tag": "component",
                "value": "storage"
            },
            {
                "tag": "equipment",
                "value": "workstation"
            }
        ],
        "delay": "30s"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24758"
        ]
    },
    "id": 1
}
```

创建主机资产监控项

创建 Zabbix agent 监控项，用于增添主机的 "OS" 资产字段。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "uname",
        "key_": "system.uname",
        "hostid": "30021",
        "type": 0,
        "interfaceid": "30007",
```

```
        "value_type": 1,
        "delay": "10s",
        "inventory_link": 5
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 1
}
```

使用预处理创建监控项

使用自定义乘数创建一个监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Device uptime",
        "key_": "sysUpTime",
        "hostid": "10084",
        "type": 20,
        "snmp_oid": "SNMPv2-MIB::sysUpTime.0",
        "value_type": 1,
        "delay": "60s",
        "interfaceid": "83",
        "preprocessing": [
            {
                "type": 1,
                "params": "0.01",
                "error_handler": 1,
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44210"
        ]
    },
    "id": 1
}
```

创建依赖监控项

为 ID 为 24759 的主监控项创建一个依赖监控项。
只允许依赖同一主机上的监控项，因此主监控项和依赖监控项应具有相同的 hostid。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
      "hostid": "30074",
      "name": "Dependent test item",
      "key_": "dependent.item",
      "type": 18,
      "master_itemid": "24759",
      "value_type": 2
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

创建 HTTP agent 监控项

创建 JSON 响应预处理式的 POST 请求方法监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "url":"http://127.0.0.1/http.php",
        "query_fields": [
            {
                "name": "mode",
                "value": "json"
            },
            {
                "name": "min",
                "value": "10"
            },
            {
                "name": "max",
                "value": "100"
            }
        ],
        "interfaceid": "1",
        "type": 19,
        "hostid": "10254",
        "delay": "5s",
        "key_": "json",
        "name": "HTTP agent example JSON",
        "value_type": 0,
        "output_format": 1,
        "preprocessing": [
            {
                "type": 12,
                "params": "$.random",
                "error_handler": 0,
                "error_handler_params": ""
```

```
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

创建脚本监控项

创建用于简单数据收集的脚本监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.item",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\nreturn request.post(\"https://postman-echo.com/post\"
        "parameters": [
            {
                "name": "host",
                "value": "{HOST.CONN}"
            }
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

来源

CItem::create() in *ui/include/classes/api/services/CItem.php*.

删除

描述

`object item.delete(array itemIds)`

此方法允许您删除监控项。

> **Note:**
> 无法通过 Zabbix API 删除 Web 监控项。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用此方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 删除目标监控项 ID。

返回值

(object) 返回一个对象，该对象包含 itemids 属性下已删除项目的 ID。

示例

删除多个监控项

删除两个监控项。
 如果删除了主监控项，则依赖监控项和监控项原型也会被自动删除。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.delete",
    "params": [
        "22982",
        "22986"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22982",
            "22986"
        ]
    },
    "id": 1
}
```

来源

CItem::delete() in *ui/include/classes/api/services/CItem.php*.

更新

描述

`object item.update(object/array items)`

此方法允许您更新现有监控项。

> **Note:**
> 无法通过 Zabbix API 更新 Web 监控项。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的监控项属性。

必须为每个监控项定义 itemid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准监控项属性之外，该方法还接受以下参数。

参数	类型	描述
preprocessing	array	用于替换当前预处理选项的监控项预处理选项。 参数行为: - 对继承对象或发现的对象为 只读
tags	array	监控项标签。 参数行为: - 对发现的对象为 只读

返回值

(object) 返回一个对象，该对象包含 itemids 属性下更新的监控项的 ID。

示例

启用监控项

启用监控项，且将其状态设置为 “0” 。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 1
}
```

更新依赖监控项

更新依赖监控项名称和主监控项 ID。仅允许依赖于同一主机，因此主监控项和依赖监控项应具有相同的主机 ID。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "name": "Dependent item updated name",
        "master_itemid": "25562",
        "itemid": "189019"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189019"
        ]
    },
    "id": 1
}
```

更新 HTTP 客户端监控项

启用监控项状态值捕获。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "allow_traps": 1
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ]
    },
    "id": 1
}
```

更新监控项预处理

更新一个应用了 “在范围内” 预处理规则的监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "preprocessing": [
            {
                "type": 13,
                "params": "\n100",
                "error_handler": 1,
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
```

```
            "23856"
        ]
    },
    "id": 1
}
```

更新脚本监控项

使用不同的脚本更新脚本监控项，并删除之前脚本使用的不必要参数。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "params": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

源码

ui/include/classes/api/services/CItem.php - CItem::update()

获取

描述

`integer/array item.get(object parameters)`

该方法允许您根据给定参数检索监控项。

> **Note:**
> 此方法对任何类型的用户都可用。在用户角色设置中可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
itemids	ID/array	仅返回具有给定 ID 的监控项。
groupids	ID/array	仅返回属于给定组中主机的监控项。
templateids	ID/array	仅返回属于给定模板的监控项。
hostids	ID/array	仅返回属于给定主机的监控项。
proxyids	ID/array	仅返回由给定 proxies 监控的监控项。
interfaceids	ID/array	仅返回使用给定主机接口的监控项。
graphids	ID/array	仅返回在给定图形中使用的监控项。
triggerids	ID/array	仅返回在给定触发器中使用的监控项。
webitems	flag	在结果中包含 web 监控项。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的监控项。

参数	类型	描述
inheritedTags	boolean	返回在主机/模板/链接模板中也具有给定 tags 的监控项。 可能的值： true - 模板/主机/链接模板也必须具有给定标签； false - *(默认)* 忽略来自模板/主机/链接模板的标签。
templated	boolean	如果设置为 true，则仅返回属于模板的监控项。
monitored	boolean	如果设置为 true，则仅返回属于受监控主机的已启用监控项。
group	string	仅返回属于具有给定名称的组的监控项。
host	string	仅返回属于具有给定名称的主机的监控项。
evaltype	integer	标签求值方法。 可能的值： 0 - *(默认)* And/Or； 2 - Or。
tags	array	仅返回具有给定标签的监控项。 格式：[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]。 空数组将返回所有监控项。 可能的operator 值： 0 - *(默认)* Contains； 1 - Equals； 2 - Does not contain； 3 - Does not equal； 4 - Exists； 5 - Does not exist。
with_triggers	boolean	如果设置为 true，则仅返回在触发器中使用的监控项。
selectHosts	query	返回一个hosts 属性，其中包含该监控项所属主机的数组。
selectInterfaces	query	返回一个interfaces 属性，其中包含该监控项使用的主机接口数组。
selectTriggers	query	返回一个triggers 属性，其中包含使用该监控项的触发器。 支持 count。
selectGraphs	query	返回一个graphs 属性，其中包含包含该监控项的图形。 支持 count。
selectDiscoveryData	query	返回一个 discoveryData 属性，其中包含监控项发现对象数据。监控项发现对象将已发现的监控项链接到发现该监控项所基于的监控项原型。 它具有以下属性： parent_itemid - (string) 创建该监控项所基于的监控项原型 ID； key_ - (string) 监控项原型的键； status - (int) 监控项发现状态： 0 - *(默认)* 监控项已发现， 1 - 监控项不再被发现； ts_delete - (timestamp) 不再被发现的监控项将被删除的时间； ts_disable - (timestamp) 不再被发现的监控项将被禁用的时间； disable_source - (int) 指示监控项是由 LLD 规则禁用还是手动禁用： 0 - *(默认)* 自动禁用， 1 - 由 LLD 规则禁用。
selectDiscoveryRule	query	返回一个discoveryRule 属性，其中包含创建该监控项的 LLD 规则。
selectInheritedTags	query	返回一个inheritedTags 属性，其中包含位于模板/主机/链接模板上的标签。
selectPreprocessing	query	返回一个preprocessing 属性，其中包含监控项预处理选项。
selectTags	query	在tags 属性中返回监控项标签。
selectValueMap	query	返回一个valuemap 属性，其中包含监控项值映射。

参数	类型	描述
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 `text` 数据类型 的属性。 支持附加属性： `host` - 该监控项所属主机的技术名称。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： `selectGraphs` - 结果将按 `name` 排序； `selectTriggers` - 结果将按 `description` 排序。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`itemid`、`name`、`key_`、`delay`、`history`、`trends`、`type`、`status`。
countOutput	boolean	这些参数在参考说明中描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectItemDiscovery	query	返回一个 `itemDiscovery` 属性，其中包含监控项发现对象。监控项发现对象将该监控项链接到创建它所基于的监控项原型。 此查询已弃用，请改用 `selectDiscoveryData`。

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 `countOutput` 参数，则返回已检索对象的数量。

示例

按键查找监控项

检索指定主机 ID 中被触发器使用、且监控项键中包含单词"system.cpu" 的所有监控项，并按名称对结果进行排序。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10084",
        "with_triggers": true,
        "search": {
            "key_": "system.cpu"
        },
        "sortfield": "name"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "42269",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "CPU utilization",
            "key_": "system.cpu.util",
            "delay": "0",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "%",
            "formula": "",
            "logtimefmt": "",
            "templateid": "42267",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "CPU utilization in %.",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "42264",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "name_resolved": "CPU utilization",
            "parameters": [],
```

```
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42259",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Load average (15m avg)",
            "key_": "system.cpu.load[all,avg15]",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "42219",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
```

```
            "name_resolved": "Load average (15m avg)",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42249",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Load average (1m avg)",
            "key_": "system.cpu.load[all,avg1]",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "42209",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
```

```
            "state": "0",
            "error": "",
            "name_resolved": "Load average (1m avg)",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42257",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Load average (5m avg)",
            "key_": "system.cpu.load[all,avg5]",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "42217",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
```

```
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "name_resolved": "Load average (5m avg)",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42260",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Number of CPUs",
            "key_": "system.cpu.num",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "42220",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
```

```
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "name_resolved": "Number of CPUs",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```

按键查找依赖监控项

从 ID 为 “10116” 的主机中检索所有键中包含单词 “apache” 的依赖监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10116",
        "search": {
            "key_": "apache"
        },
        "filter": {
            "type": 18
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25550",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Days",
            "key_": "apache.status.uptime.days",
            "delay": "0",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
```

```
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "flags": "0",
        "interfaceid": "0",
        "description": "",
        "inventory_link": "0",
        "lifetime": "7d",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "25545",
        "timeout": "",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "0",
        "output_format": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0",
        "uuid": "",
        "lifetime_type": "0",
        "enabled_lifetime_type": "2",
        "enabled_lifetime": "0",
        "state": "0",
        "error": "",
        "name_resolved": "Days",
        "parameters": [],
        "lastclock": "0",
        "lastns": "0",
        "lastvalue": "0",
        "prevvalue": "0"
    },
    {
        "itemid": "25555",
        "type": "18",
        "snmp_oid": "",
        "hostid": "10116",
        "name": "Hours",
        "key_": "apache.status.uptime.hours",
        "delay": "0",
        "history": "90d",
        "trends": "365d",
        "status": "0",
        "value_type": "3",
        "trapper_hosts": "",
        "units": "",
        "formula": "",
        "logtimefmt": "",
        "templateid": "0",
        "valuemapid": "0",
        "params": "",
```

```
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "25545",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "name_resolved": "Hours",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```


查找 HTTP agent 监控项

查找指定主机 ID 中 post body 类型为 XML 的 HTTP agent 监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "hostids": "10255",
        "filter": {
            "type": 19,
            "post_type": 3
        }
```

```
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28252",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10255",
            "name": "template item",
            "key_": "ti",
            "delay": "30s",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "localhost",
            "query_fields": [
                {
                    "name": "mode",
                    "value": "xml"
                }
            ],
            "posts": "<body>\r\n<![CDATA[{$MACRO}<foo></bar>]]>\r\n</body>",
            "status_codes": "200",
            "follow_redirects": "0",
            "post_type": "3",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "1",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
```

```
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "state": "0",
            "error": "",
            "name_resolved": "template item",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "",
            "prevvalue": ""
        }
    ],
    "id": 1
}
```


检索预处理规则监控项

检索 ID 为 “10254” 的主机所有监控项及其预处理规则。

请求：


```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": ["itemid", "name", "key_"],
        "selectPreprocessing": "extend",
        "hostids": "10254"
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": {
        "itemid": "23865",
        "name": "HTTP agent example JSON",
        "key_": "json",
        "preprocessing": [
            {
                "type": "12",
                "params": "$.random",
                "error_handler": "1",
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```


另请参阅

- 发现规则
- 图形
- 主机
- 主机接口
- 触发器

来源

CItem::get() in *ui/include/classes/api/services/CItem.php*.

监控项原型

此类用于处理监控项原型。

对象引用：

- 监控项原型
  - HTTP 标头
  - HTTP 查询字段
- 监控项原型标签
- 监控项原型预处理

可用方法：

- itemprototype.create - 创建新的监控项原型
- itemprototype.delete - 删除监控项原型
- itemprototype.get - 检索监控项原型
- itemprototype.update - 更新监控项原型

监控项原型对象

以下对象与 监控项原型 API 直接相关。

监控项原型

监控项原型对象具有以下属性。

属性	类型	描述
itemid	ID	监控项原型的 ID。 属性行为： - 只读 - 更新操作时为必需
delay	string	监控项原型的更新间隔。 接受秒数（例如 1、60、300）、带有时间后缀的值（例如 30s、1m、2h、1d），或者可选地接受一个或多个自定义间隔，所有值以分号分隔。自定义间隔可以混合使用灵活间隔和调度间隔。 接受用户宏和 LLD 宏。如果使用宏，该值必须是单个宏。不支持多个宏或宏与文本混合使用。灵活间隔可以写成由斜杠分隔的两个宏（例如 {$FLEX_INTERVAL}/{$FLEX_PERIOD}）。 示例： 1h;wd1-5h9-18;{$Macro1}/1-7,00:00-24:00;0/6-7,12:00-24:00;{$Macr 属性行为： - 当 type 设置为 “Zabbix agent”(0)、“Simple check”(3)、“Zabbix internal”(5)、“External check”(10)、“Database monitor”(11)、“IPMI agent”(12)、“SSH agent”(13)、“TELNET agent”(14)、“Calculated”(15)、“JMX agent”(16)、“HTTP agent”(19)、“SNMP agent”(20)、“Script”(21)、“Browser”(22) 或 “Telemetry query”(24) 时为必需；或者当 type 设置为 “Zabbix agent (active)”(7) 且 key_ 不包含 “mqtt.get” 时为必需
hostid	ID	监控项原型所属主机的 ID。 属性行为： - 常量 - 创建操作时为必需

属性	类型	描述
interfaceid	ID	监控项原型所属主机接口的 ID。 属性行为： - 当监控项原型属于主机且 type 设置为 “Zabbix agent”、“IPMI agent”、“JMX agent”、“SNMP trap” 或 “SNMP agent” 时为必需 - 当监控项原型属于主机且 type 设置为 “Simple check”、“External check”、“SSH agent”、“TELNET agent” 或 “HTTP agent” 时支持
key_	string	监控项原型键。 属性行为： - 创建操作时为必需 - 对继承对象为只读
name	string	监控项原型的名称。 支持用户宏。 属性行为： - 创建操作时为必需 - 对继承对象为只读
type	integer	监控项原型的类型。 可能的值： 0 - Zabbix agent； 2 - Zabbix trapper； 3 - Simple check； 5 - Zabbix internal； 7 - Zabbix agent (active)； 10 - External check； 11 - Database monitor； 12 - IPMI agent； 13 - SSH agent； 14 - TELNET agent； 15 - Calculated； 16 - JMX agent； 17 - SNMP trap； 18 - Dependent item； 19 - HTTP agent； 20 - SNMP agent； 21 - Script； 22 - Browser； 24 - Telemetry query。 属性行为： - 创建操作时为必需 - 对继承对象为只读
url	string	URL 字符串。 支持 LLD 宏、用户宏、{HOST.IP}、{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.NAME}、{HOST.PORT}、{ITEM.ID}、{ITEM.KEY}。 属性行为： - 当 type 设置为 “HTTP agent” 时为必需 - 对继承对象为只读

属性	类型	描述
value_type	integer	监控项原型的信息类型。 可能的值： 0 - 数值（浮点）； 1 - 字符； 2 - 日志； 3 - 数值（无符号）； 4 - 文本； 5 - 二进制； 6 - JSON。 属性行为： - 创建操作时为必需 - 对继承对象为只读
allow_traps	integer	允许以类似 trapper 监控项的方式填充值。 0 - *(默认)* 不接受传入数据； 1 - 接受传入数据。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持
authtype	integer	身份验证方法。 当 `type` 设置为 “SSH agent” 时的可能值： 0 - *(默认)* 密码； 1 - 公钥。 当 `type` 设置为 “HTTP agent” 时的可能值： 0 - *(默认)* 无； 1 - basic； 2 - NTLM； 3 - Kerberos； 4 - Digest。 属性行为： - 当 `type` 设置为 “SSH agent” 或 “HTTP agent” 时支持 - 对继承对象为只读（当 `type` 设置为 “HTTP agent” 时）
description	string	监控项原型的描述。
flags	integer	监控项原型的来源。 可能的值： 2 - 监控项原型； 6 - 已发现的监控项原型 属性行为： - 只读
follow_redirects	integer	轮询数据时跟随响应重定向。 可能的值： 0 - 不跟随重定向； 1 - *(默认)* 跟随重定向。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
headers	array	执行 HTTP 请求时发送的标头数组。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读

属性	类型	描述
history	string	历史数据应保留多长时间的时间单位。 也接受用户宏和 LLD 宏。 默认值：31d。
http_proxy	string	HTTP(S) proxy 连接字符串。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
ipmi_sensor	string	IPMI 传感器。 属性行为： - 当 `type` 设置为 “IPMI agent” 且 `key_` 未设置为 “ipmi.get” 时为必需 - 当 `type` 设置为 “IPMI agent” 时支持 - 对继承对象为只读
jmx_endpoint	string	JMX agent 自定义连接字符串。 默认值： service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi 属性行为： - 当 `type` 设置为 “JMX agent” 时支持
logtimefmt	string	日志条目中的时间格式。 属性行为： - 当 `value_type` 设置为 “log” 时支持 - 对继承对象为只读
master_itemid	ID	主监控项的 ID。 属性行为： - 当 `type` 设置为 “Dependent item” 时为必需 - 对继承对象为只读
output_format	integer	是否将响应转换为 JSON。 可能的值： 0 - *(默认)* 存储原始数据； 1 - 转换为 JSON。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
params	string	取决于监控项原型类型的其他参数： - SSH agent 和 TELNET agent 监控项原型要执行的脚本； - 数据库监控监控项原型的 SQL 查询； - 计算监控项原型的公式； - Script 和 Browser 监控项原型的脚本。 属性行为： - 当 `type` 设置为 “Database monitor”、“SSH agent”、“TELNET agent”、“Calculated”、“Script” 或 “Browser” 时为必需 - 对继承对象为只读（当 `type` 设置为 “Script” 或 “Browser” 时）
parameters	object/array	当 `type` 设置为 “Script” 或 “Browser” 时的其他参数。由包含 `name` 和 `value` 属性的对象组成的数组，其中 `name` 必须唯一。 属性行为： - 当 `type` 设置为 “Script” 或 “Browser” 时支持 - 对继承对象为只读

属性	类型	描述
password	string	身份验证密码。 属性行为： - 当 type 设置为 “JMX agent” 且设置了 username 时为必需 - 当 type 设置为 “Simple check”、“SSH agent”、“TELNET agent”、“Database monitor” 或 “HTTP agent” 时支持 - 对继承对象为只读（当 type 设置为 “HTTP agent” 时）
post_type	integer	posts 属性中存储的 POST 数据正文类型。 可能的值： 0 - *(默认)* 原始数据。 2 - JSON 数据。 3 - XML 数据。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对继承对象为只读
posts	string	HTTP(S) 请求正文数据。 属性行为： - 当 type 设置为 “HTTP agent” 且 post_type 设置为 “JSON data” 或 “XML data” 时为必需 - 当 type 设置为 “HTTP agent” 且 post_type 设置为 “Raw data” 时支持 - 对继承对象为只读
privatekey	string	私钥文件的名称。 属性行为： - 当 type 设置为 “SSH agent” 且 authtype 设置为 “public key” 时为必需
publickey	string	公钥文件的名称。 属性行为： - 当 type 设置为 “SSH agent” 且 authtype 设置为 “public key” 时为必需
query_fields	array	执行 HTTP 请求时发送的查询字段数组。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对继承对象为只读
request_method	integer	请求方法类型。 可能的值： 0 - *(默认)* GET； 1 - POST； 2 - PUT； 3 - HEAD。 属性行为： - 当 type 设置为 “HTTP agent” 时支持 - 对继承对象为只读

属性	类型	描述
retrieve_mode	integer	应存储响应的哪一部分。 当 `request_method` 设置为 “GET”、“POST” 或 “PUT” 时的可能值： 0 - *(默认)* 正文； 1 - 标头； 2 - 同时存储正文和标头。 当 `request_method` 设置为 “HEAD” 时的可能值： 1 - 标头。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
snmp_oid	string	SNMP OID。 属性行为： - 当 `type` 设置为 “SNMP agent” 时为必需 - 对继承对象为只读
ssl_cert_file	string	公共 SSL 密钥文件路径。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
ssl_key_file	string	私有 SSL 密钥文件路径。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
ssl_key_password	string	SSL 密钥文件的密码。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
status	integer	监控项原型的状态。 可能的值： 0 - *(默认)* 已启用的监控项原型； 1 - 已禁用的监控项原型； 3 - 不支持的监控项原型。
status_codes	string	以逗号分隔的所需 HTTP 状态码范围。 也支持在逗号分隔的列表中使用用户宏或 LLD 宏。 示例：200,200-{$M},{$M},200-400 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
templateid	ID	父模板监控项原型的 ID。 属性行为： - 只读

属性	类型	描述
timeout	string	监控项数据轮询请求超时时间。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m）。也接受用户宏和 LLD 宏。 可能的值范围：1-600s。 默认值：”” - 使用 proxy/全局设置。 属性行为： - 当 type 设置为 “Zabbix agent”(0)、“Simple check”(3) 且 key_ 不以 “vmware.” 和 “icmpping” 开头、“Zabbix agent (active)”(7)、“External check”(10)、“Database monitor”(11)、“SSH agent”(13)、“TELNET agent”(14)、“HTTP agent”(19)、“SNMP agent”(20) 且 snmp_oid 以 “walk[” 或 “get[” 开头、“Script”(21) 或 “Browser”(22) 时支持 - 对继承对象为只读
trapper_hosts	string	允许的主机。 属性行为： - 当 type 设置为 “Zabbix trapper” 时支持，或者当 type 设置为 “HTTP agent” 且 allow_traps 设置为 “Allow incoming data” 时支持。 如果省略该属性，则会自动将其设置为 {$TRAPPER.ALLOWED_HOSTS} 用户宏（默认值）。默认情况下，{$TRAPPER.ALLOWED_HOSTS} 为 127.0.0.1,::1。 如果该属性设置为空字符串，则将拒绝来自所有主机的传入连接。
trends	string	趋势数据应保留多长时间的时间单位。 也接受用户宏和 LLD 宏。 默认值：365d。 属性行为： - 当 value_type 设置为 “numeric float” 或 “numeric unsigned” 时支持
units	string	值单位。 属性行为： - 当 value_type 设置为 “numeric float” 或 “numeric unsigned” 时支持 - 对继承对象为只读
username	string	身份验证用户名。 属性行为： - 当 type 设置为 “SSH agent” 或 “TELNET agent” 时为必需，或者当 type 设置为 “JMX agent” 且设置了 password 时为必需 - 当 type 设置为 “Simple check”、“Database monitor” 或 “HTTP agent” 时支持 - 对继承对象为只读（当 type 设置为 “HTTP agent” 时）
uuid	string	通用唯一标识符，用于将导入的监控项原型链接到已存在的监控项原型。如果未提供，则自动生成。 属性行为： - 当监控项原型属于模板时支持
valuemapid	ID	关联值映射的 ID。 属性行为： - 当 value_type 设置为 “numeric float”、“character” 或 “numeric unsigned” 时支持 - 对继承对象为只读

属性	类型	描述
verify_host	integer	是否验证连接的主机名是否与主机证书中的主机名匹配。 可能的值： 0 - *(默认)* 不验证； 1 - 验证。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
verify_peer	integer	是否验证主机证书是否真实有效。 可能的值： 0 - *(默认)* 不验证； 1 - 验证。 属性行为： - 当 `type` 设置为 “HTTP agent” 时支持 - 对继承对象为只读
discover	integer	监控项原型发现状态。 可能的值： 0 - *(默认)* 将发现新监控项； 1 - 不发现新监控项，并将现有监控项标记为丢失。
query	object	Telemetry query 配置。定义监控项从遥测数据源查询哪些内容、如何筛选数据以及如何聚合结果。 属性行为： - 当 `type` 设置为 “Telemetry query” 时为必需 - 对继承对象或已发现对象为只读
time_shift	string	监控项查询的时间范围结束于当前时间之前多久。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m、2h、1d）。也接受用户宏和 LLD 宏。 可能的值范围：0-86400s。 默认值：15s。 属性行为： - 当 `type` 设置为 “Telemetry query” 时支持
lookback_limit	string	当没有可继续使用的上一次查询（例如监控项首次运行时），或上一次查询早于 `delay`（例如 Zabbix 服务器停止运行时），监控项查询数据时最多回溯到当前时间之前多久。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m、2h、1d）。也接受用户宏和 LLD 宏。 可能的值范围：1-259200s。 默认值：10m。 属性行为： - 当 `type` 设置为 “Telemetry query” 时支持

属性	类型	描述
granularity	string	查询的完整时间范围被拆分成的每个时间片段的长度。 接受秒数（例如 1、60、300）或带有时间后缀的值（例如 30s、1m、2h、1d）。也接受用户宏和 LLD 宏。 可能的值范围：1-86400s。 默认值：15s。 属性行为： - 当 type 设置为 “Telemetry query” 时支持

HTTP 标头

header 对象具有以下属性：

属性	类型	说明
name	string	HTTP 标头名称。 属性行为: - 必填
value	string	标头值。 属性行为: - 必填

HTTP 查询字段

查询字段对象定义了用于指定 URL 参数的名称和值。它具有以下属性：

属性	类型	描述
name	string	参数名称。 属性行为： - 必填
value	string	参数值。 属性行为： - 必填

Telemetry 查询

Telemetry 查询对象具有以下属性。

Property	Type	Description
signal_type	integer	监控项查询的 Signal 类型。 可能的值： 0 - Traces； 1 - Metrics； 2 - Logs。 属性行为： - 必需

Property	Type	Description
metric_point_type	integer	监控项查询的 Metric point 类型。 可能的值： 0 - Sum； 1 - Gauge； 2 - Histogram； 3 - Exponential histogram。 属性行为： - 当 `signal_type` 设置为”Metrics” 时必需
columns	array	监控项查询的Telemetry 数据列数组。可以为空。 属性行为： - 必需
aggregated_columns	array	监控项查询的Telemetry 数据聚合列数组，以及如何根据列记录计算值。 属性行为： - 必需
filter	object	查询中列记录的Telemetry 查询过滤器。 属性行为： - 必需

遥测数据列

遥测数据列对象具有以下属性。

Property	Type	Description
column	string	监控项查询的遥测数据列名称。 请参阅支持的遥测数据列。 对于多个单值列，每个 `column` 值必须唯一。对于多个存储键值对的列，每个 `column` 和 `attribute_key` 对必须唯一。
attribute_key	string	监控项在存储键值对的列中查询的键名称。 对于多个存储键值对的列，每个 `column` 和 `attribute_key` 对必须唯一。 属性行为： - 如果 `column` 设置为存储键值对的列，则为必需

遥测数据聚合列

遥测数据聚合列对象具有以下属性。

Property	Type	Description
column	string	监控项查询的遥测数据列名称。 请参阅支持的遥测数据列。 对于多个列，每个 `column` 值必须唯一。 如果将 `function` 设置为”Count”，则不支持此属性。 属性行为： - 如果将 `function` 设置为”Min”、”Max”、”Average”、”Sum” 或”Percentile”，则为必需

Property	Type	Description
function	integer	监控项应用于列记录的函数。 可能的值： 1 - Min； 2 - Max； 3 - Average； 4 - Count； 5 - Sum； 8 - Percentile。 属性行为： - 必需
parameters	array	监控项根据列记录计算的百分位数。 可能的值：0 到 100 之间的数字，以字符串形式给出。 属性行为： - 如果将 `function` 设置为"Percentile"，则为必需
alias	string	计算值的唯一名称，用于在监控项值中标识该值。 每个 `alias` 值必须唯一。 属性行为： - 必需

遥测数据过滤器

遥测数据过滤器对象具有以下属性。

Property	Type	Description
conditions	array	遥测数据过滤器条件，遥测数据列中的记录必须满足这些条件。 属性行为： - 必需
evaltype	integer	过滤条件的评估方法。 可能的值： 0 - And/Or； 1 - And； 2 - Or； 3 - 自定义表达式。 属性行为： - 必需
eval_formula	string	用于评估过滤条件的生成表达式。该表达式包含通过其 `formulaid` 引用特定过滤条件的 ID。对于使用自定义表达式的过滤器，`eval_formula` 的值等于 `formula` 的值。 属性行为： - 只读
formula	string	用于评估使用自定义表达式的过滤器条件的用户定义表达式。该表达式必须包含通过其 `formulaid` 引用特定过滤条件的 ID。表达式中使用的 ID 必须与过滤器条件中定义的 ID 完全匹配：不能有任何条件未使用或被省略。 属性行为： - 当 `evaltype` 设置为 "自定义表达式" 时必需

遥测数据筛选条件

遥测数据筛选条件对象具有以下属性。

Property	Type	Description
column	string	筛选条件所应用的遥测数据列的名称。 请参阅支持的遥测数据列。 属性行为： - 必需
attribute_key	string	在包含键值对的列中，条件所应用的键的名称。 属性行为： - 如果 `column` 设置为包含键值对的列，则为必需
operator	integer	筛选条件运算符。 单值列的可能值： 0 - 等于； 1 - 不等于； 2 - 包含； 3 - 不包含。 包含键值对的列的可能值： 0 - 等于； 1 - 不等于； 12 - 存在。 属性行为： - 必需
value	string	用于比较的值。 属性行为： - 如果 `operator` 设置为 “等于”、“不等于”、“包含” 或 “不包含”，则为必需
formulaid	string	用于从自定义表达式中引用条件的任意唯一 ID。只能包含大写字母。修改筛选条件时，必须由用户定义该 ID；但之后请求这些条件时，系统会重新生成该 ID。 属性行为： - 如果遥测查询筛选器对象的 `evaltype` 设置为 “自定义表达式”，则为必需

监控项原型标签

监控项原型标签对象具有以下属性。

属性	Type	描述
tag	string	监控项原型标签名称。 属性行为： - 必填
value	string	监控项原型标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板； 1 - 主机。 属性行为： - 只读。
objectid	ID	标签继承自的对象 ID。 属性行为： - 只读。

监控项原型预处理

监控项原型预处理对象具有以下属性。

属性	类型	描述
type	integer	预处理选项类型。 可能的值： 1 - 自定义倍数； 2 - 右侧修剪； 3 - 左侧修剪； 4 - 修剪； 5 - 正则表达式； 6 - 布尔值转十进制； 7 - 八进制转十进制； 8 - 十六进制转十进制； 9 - 简单变化； 10 - 每秒变化； 11 - XML XPath； 12 - JSONPath； 13 - 范围检查； 14 - 匹配正则表达式； 15 - 不匹配正则表达式； 16 - 检查 JSON 中的错误； 17 - 检查 XML 中的错误； 18 - 使用正则表达式检查错误； 19 - 丢弃未更改的值； 20 - 使用心跳丢弃未更改的值； 21 - JavaScript； 22 - Prometheus 模式； 23 - Prometheus 转 JSON； 24 - CSV 转 JSON； 25 - 替换； 26 - 检查不支持； 27 - XML 转 JSON； 28 - SNMP walk 值； 29 - SNMP walk 转 JSON； 30 - SNMP get 值。 属性行为： - 必需
params	string	预处理选项使用的其他参数。 多个参数使用换行符 (\n) 分隔。 如果 `type` 设置为 “检查不支持”，参数遵循 `<scope>[\n<pattern>]` 语法，其中 *pattern* 是正则表达式，*scope* 为以下值之一： -1 - 匹配任意错误； 0 - 检查错误消息是否匹配 *pattern*； 1 - 检查错误消息是否不匹配 *pattern*。 属性行为： - 当 `type` 设置为 “自定义倍数”(1)、“右侧修剪”(2)、“左侧修剪”(3)、“修剪”(4)、“正则表达式”(5)、“XML XPath”(11)、“JSONPath”(12)、“范围检查”(13)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“使用正则表达式检查错误”(18)、“使用心跳丢弃未更改的值”(20)、“JavaScript”(21)、“Prometheus 模式”(22)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“替换”(25)、“检查不支持”(26)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时必需

属性	类型	描述
error_handler	integer	预处理步骤失败时使用的操作类型。 可能的值： 0 - 由 Zabbix 服务器设置错误消息； 1 - 丢弃值； 2 - 设置自定义值； 3 - 设置自定义错误消息。 当 type 设置为 “检查不支持” 时的可能值： 1 - 丢弃值； 2 - 设置自定义值； 3 - 设置自定义错误消息。 属性行为： - 当 type 设置为 “自定义倍数”(1)、“正则表达式”(5)、“布尔值转十进制”(6)、“八进制转十进制”(7)、“十六进制转十进制”(8)、“简单变化”(9)、“每秒变化”(10)、“XML XPath”(11)、“JSONPath”(12)、“范围检查”(13)、“匹配正则表达式”(14)、“不匹配正则表达式”(15)、“检查 JSON 中的错误”(16)、“检查 XML 中的错误”(17)、“使用正则表达式检查错误”(18)、“Prometheus 模式”(22)、“Prometheus 转 JSON”(23)、“CSV 转 JSON”(24)、“检查不支持”(26)、“XML 转 JSON”(27)、“SNMP walk 值”(28)、“SNMP walk 转 JSON”(29) 或 “SNMP get 值”(30) 时必需
error_handler_params	string	错误处理程序参数。 属性行为： - 当 error_handler 设置为 “设置自定义值” 或 “设置自定义错误消息” 时必需

以下参数和错误处理程序适用于每种预处理类型。

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
1	自定义倍数	number^1, 6^			0, 1, 2, 3
2	右侧修剪	字符列表[2]			
3	左侧修剪	字符列表[2]			
4	修剪	字符列表[2]			
5	正则表达式	pattern[3]	output[2]		0, 1, 2, 3
6	布尔值转十进制				0, 1, 2, 3
7	八进制转十进制				0, 1, 2, 3
8	十六进制转十进制				0, 1, 2, 3
9	简单变化				0, 1, 2, 3
10	每秒变化				0, 1, 2, 3
11	XML XPath	path[4]			0, 1, 2, 3

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
12	JSONPath	path[4]			0, 1, 2, 3
13	范围检查	min^1, 6^	max^1, 6^		0, 1, 2, 3
14	匹配正则表达式	pattern[3]			0, 1, 2, 3
15	不匹配正则表达式	pattern[3]			0, 1, 2, 3
16	检查 JSON 中的错误	path[4]			0, 1, 2, 3
17	检查 XML 中的错误	path[4]			0, 1, 2, 3
18	使用正则表达式检查错误	pattern[3]	output[2]		0, 1, 2, 3
19	丢弃未更改的值				
20	使用心跳丢弃未更改的值	seconds^5, 6^			
21	JavaScript	script[2]			
22	Prometheus 模式	pattern^6, 7^	`value`、`label`、`function`	output^8, 9^	0, 1, 2, 3
23	Prometheus 转 JSON	pattern^6, 7^			0, 1, 2, 3
24	CSV 转 JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	替换	search string[2]	replacement[2]		
26	检查不支持	scope[1]	pattern^3, 6^		1, 2, 3
27	XML 转 JSON				0, 1, 2, 3
28	SNMP walk 值	OID[2]	格式： 0 - 不变更 1 - 将 Hex-STRING 转换为 UTF-8 2 - 将 Hex-STRING 转换为 MAC 3 - 将 BITS 转换为整数		0, 1, 2, 3

预处理类型	名称	参数 1	参数 2	参数 3	支持的错误处理程序
29	SNMP walk 转 JSON[10]	字段名称[2]	OID 前缀[2]	格式： 0 - 不变更 1 - 将 Hex-STRING 转换为 UTF-8 2 - 将 Hex-STRING 转换为 MAC 3 - 将 BITS 转换为整数	0, 1, 2, 3
30	SNMP get 值	格式： 1 - 将 Hex-STRING 转换为 UTF-8 2 - 将 Hex-STRING 转换为 MAC 3 - 将 BITS 转换为整数			0, 1, 2, 3

[1] 整数或浮点数
 [2] 字符串
 [3] 正则表达式
 [4] JSONPath 或 XML XPath
 [5] 正整数（支持时间后缀，例如 30s、1m、2h、1d）
 [6] 用户宏、LLD 宏
 [7] 遵循以下语法的 Prometheus 模式：`<metric name>{<label name>="<label value>", ...} == <value>`。每个 Prometheus 模式组件（指标、标签名称、标签值和指标值）都可以是用户宏或 LLD 宏。
 [8] 当第二个参数选择 `label` 时，Prometheus 输出遵循以下语法：`<label name>`（可以是用户宏或 LLD 宏）。
 [9] 当第二个参数选择 `function` 时，可以使用以下聚合函数之一：`sum`、`min`、`max`、`avg`、`count`。
 [10] 支持多条 “字段名称、OID 前缀、格式” 记录，记录之间使用换行符分隔。

创建

描述

`object itemprototype.create(object/array itemPrototypes)`

此方法允许您创建新的监控项原型。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles 。

参数

`(object/array)` 要创建的监控项原型。

除标准监控项原型属性外，该方法还接受以下参数。

参数	类型	描述
ruleid	ID	该监控项所属的LLD 规则 的 ID。 参数行为: - 必需
preprocessing	array	监控项原型预处理 选项。
tags	array	监控项原型标签。

返回值

返回一个对象 （`object`），该对象包含在 `itemids` 属性下创建的监控项原型的 IDs。返回的 IDs 的顺序与传递的项目原型的顺序相匹配。

案例

创建监控项原型

创建监控项原型来发现文件系统上的可用磁盘空间。发现的 Zabbix agent 监控项每 30 秒更新一次的数字。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Free disk space on {#FSNAME}",
        "key_": "vfs.fs.size[{#FSNAME},free]",
        "hostid": "10197",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "interfaceid": "112",
        "delay": "30s"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27666"
        ]
    },
    "id": 1
}
```

创建监控项原型预处理

先创建一个监控项，第二步在预处理中使用每秒更改和自定义乘数。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Incoming network traffic on {#IFNAME}",
        "key_": "net.if.in[{#IFNAME}]",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
        "units": "bps",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": 10,
                "params": "",
                "error_handler": 0,
                "error_handler_params": ""
            },
            {
                "type": 1,
                "params": "8",
                "error_handler": 2,
                "error_handler_params": "10"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

创建依赖监控项原型

创建依赖与 ID 44211 监控项原型的监控项原型。只允许依赖一些主机 (模板/发现规则)，因此主要监控项和依赖监控项应该具有相同的 hostid 和 ruleid。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
      "hostid": "10001",
      "ruleid": "27665",
      "name": "Dependent test item prototype",
      "key_": "dependent.prototype",
      "type": 18,
      "master_itemid": "44211",
      "value_type": 3
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44212"
        ]
    },
    "id": 1
}
```

创建 HTTP 客户端监控项原型

使用用户宏、查询字段和自定义头部创建带有 URL 的监控项原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "type": "19",
        "hostid": "10254",
        "ruleid": "28256",
        "interfaceid": "2",
        "name": "api item prototype example",
        "key_": "api_http_item",
        "value_type": 3,
        "url": "{$URL_PROTOTYPE}",
        "query_fields": [
            {
                "name": "min",
                "value": "10"
```

```
            },
            {
                "name": "max",
                "value" "100"
            }
        ],
        "headers": [
            {
                "name": "X-Source",
                "value": "api"
            }
        ],
        "delay": "35"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

创建监控项原型脚本

使用脚本监控项原型创建一个简单的数据集合。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.itemprototype",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\nreturn request.post(\"https://postman-echo.com/post\"
        "parameters": [
            {
                "name": "host",
                "value": "{HOST.CONN}"
            }
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
```

```
    "id": 1
}
```

来源

CItemPrototype::create() in *ui/include/classes/api/services/CItemPrototype.php*.

删除

描述

`object itemprototype.delete(array itemPrototypeIds)`

此方法允许您删除监控项原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

删除可通过监控项原型的 ID（array）。

返回值

（object）返回一个对象，该对象包含 prototypeids 属性下已删除监控项原型的 ID。

示例

删除多个监控项原型

删除两个监控项原型。
 如果删除主监控项或监控项原型，从属监控项原型会被自动移除。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.delete",
    "params": [
        "27352",
        "27356"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "prototypeids": [
            "27352",
            "27356"
        ]
    },
    "id": 1
}
```

来源

CItemPrototype::delete() in *ui/include/classes/api/services/CItemPrototype.php*.

更新

描述

`object itemprototype.update(object/array itemPrototypes)`

此方法允许你更新现有的监控项原型。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的监控项原型属性。

必须为每个监控项原型定义 itemid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除标准监控项原型属性外，该方法还接受以下参数。

参数	类型	描述
preprocessing	array	用于替换当前预处理选项的监控项原型预处理选项。 参数行为: - 对继承对象为只读
tags	array	监控项原型标签。

返回值

(object) 返回一个对象，其中包含 itemids 属性下更新的监控项原型的 ID。

示例

修改监控项原型接口

修改自动发现监控项的主机接口。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "27428",
        "interfaceid": "132"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27428"
        ]
    },
    "id": 1
}
```

更新依赖监控项原型

使用新主监控项原型 ID 更新依赖监控项原型。只允许依赖于同一主机（模板/发现规则）监控项原型，因此主监控项原型和依赖监控项原型应具有相同的 hostid 和 ruleid。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "master_itemid": "25570",
        "itemid": "189030"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189030"
        ]
    },
    "id": 1
}
```

更新 HTTP 代理监控项原型

修改查询字段和移除所有自定义的头部。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid":"28305",
        "query_fields": [
            {
                "name": "random",
                "value": "qwertyuiopasdfghjklzxcvbnm"
            }
        ],
        "headers": []
    }
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

更新监控项原型预处理选项

使用监控项原型预处理规则“自定义乘数”更新监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": 1,
                "params": "4",
                "error_handler": 2,
                "error_handler_params": "5"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

更新监控项原型脚本

使用不同的脚本更新监控项原型脚本，删除先前脚本使用的不必要参数。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

来源

CItemPrototype::update() in *ui/include/classes/api/services/CItemPrototype.php*.

获取

描述

`integer/array itemprototype.get(object parameters)`

该方法允许您根据给定参数检索监控项原型。

> **Note:**
> 此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
discoveryids	ID/array	仅返回属于给定 LLD 规则的监控项原型。
graphids	ID/array	仅返回在给定图形原型中使用的监控项原型。
hostids	ID/array	仅返回属于给定主机的监控项原型。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的监控项原型。
itemids	ID/array	仅返回具有给定 ID 的监控项原型。

参数	类型	说明
monitored	boolean	如果设置为 `true`，则仅返回属于已监控主机的已启用监控项原型。
templated	boolean	如果设置为 `true`，则仅返回属于模板的监控项原型。
templateids	ID/array	仅返回属于给定模板的监控项原型。
triggerids	ID/array	仅返回在给定触发器原型中使用的监控项原型。
selectDiscoveryData	query	返回 `discoveryData` 属性，其中包含监控项原型发现对象数据。监控项原型发现对象将已发现的监控项原型链接到发现它的监控项原型。 它具有以下属性： `parent_itemid` - (`string`) 创建该监控项原型所依据的监控项原型 ID； `key_` - (`string`) 监控项原型的键值； `status` - (`int`) 监控项原型发现状态： 0 - *(默认)* 监控项原型已发现， 1 - 监控项原型不再被发现； `ts_delete` - (`timestamp`) 不再被发现的监控项原型将被删除的时间； `ts_disable` - (`timestamp`) 不再被发现的监控项原型将被禁用的时间； `disable_source` - (`int`) 指示监控项原型是由 LLD 规则还是手动禁用： 0 - *(默认)* 自动禁用， 1 - 由 LLD 规则禁用。
selectDiscoveryRule	query	返回`discoveryRule` 属性，其中包含该监控项原型所属的低级别发现规则。
selectDiscoveryRulePrototype	query	返回`discoveryRulePrototype` 属性，其中包含该监控项原型所属的父 LLD 规则原型。
selectGraphs	query	返回`graphs` 属性，其中包含使用该监控项原型的图形原型。 支持 `count`。
selectHosts	query	返回`hosts` 属性，其中包含该监控项原型所属主机的数组。
selectInheritedTags	query	返回`inheritedTags` 属性，其中包含模板/主机/已链接模板上的标签。
selectTags	query	在`tags` 属性中返回监控项原型标签。
selectTriggers	query	返回`triggers` 属性，其中包含使用该监控项原型的触发器原型。 支持 `count`。
selectPreprocessing	query	返回`preprocessing` 属性，其中包含监控项原型预处理选项。
selectValueMap	query	返回`valuemap` 属性，其中包含监控项原型值映射。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 `text` 数据类型 的属性。 支持附加属性： `host` - 该监控项原型所属主机的技术名称。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： `selectGraphs` - 结果将按 `name` 排序； `selectTriggers` - 结果将按 `description` 排序。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`itemid`、`name`、`key_`、`delay`、`type`、`status`、`history`、`trends`、`discovered`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回其中之一：

- 一个对象数组；
- 如果使用 countOutput 参数，返回被检索对象的数量。

示例

从 LLD 规则中检索监控项原型

检索指定 LLD 规则 ID 的所有监控项原型。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23077",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10079",
            "name": "Incoming network traffic on en0",
            "key_": "net.if.in[en0]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "bps",
            "formula": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "2",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
```

```
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "parameters": []
        },
        {
            "itemid": "10010",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10001",
            "name": "Processor load (1 min average per core)",
            "key_": "system.cpu.load[percpu,avg1]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "2",
            "interfaceid": "0",
            "description": "The processor load is calculated as system CPU load divided by number of CPU c
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
```

```
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "parameters": []
        }
    ],
    "id": 1
}
```

查找依赖监控项

为指定的监控项 ID 查找一个依赖监控项。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "output": "extend",
        "filter": {
            "type": 18,
            "master_itemid": "25545"
        },
        "limit": "1"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25547",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Seconds",
            "key_": "apache.status.uptime.seconds",
            "delay": "0",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
```

```
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "25545",
            "timeout": "",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "parameters": []
        }
    ],
    "id": 1
}
```


查找 HTTP agent 监控项原型

查找指定主机 ID 中请求方法为 HEAD 的 HTTP agent 监控项原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "hostids": "10254",
        "filter": {
            "type": 19,
            "request_method": 3
        }
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28257",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10254",
            "name": "discovered",
            "key_": "item[{#INAME}]",
            "delay": "{#IUPDATE}",
            "history": "90d",
            "trends": "30d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "logtimefmt": "",
            "templateid": "28255",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "2",
            "interfaceid": "2",
            "description": "",
            "inventory_link": "0",
            "lifetime": "7d",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "",
            "url": "{#IURL}",
            "query_fields": [],
            "posts": "",
            "status_codes": "",
            "follow_redirects": "0",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "lifetime_type": "0",
            "enabled_lifetime_type": "2",
            "enabled_lifetime": "0",
            "parameters": []
        }
    ],
```

```
    "id": 1
}
```

另请参阅

- 主机
- 图形原型
- 触发器原型

来源

CItemPrototype::get() in *ui/include/classes/api/services/CItemPrototype.php*.

维护

此类用于处理维护。

对象引用：

- 维护
- 时间段
- 问题标签

可用方法：

- maintenance.create - 创建新的维护
- maintenance.delete - 删除维护
- maintenance.get - 检索维护
- maintenance.update - 更新维护

维护期对象

如下对象与维护期 API 关联。

维护

维护对象具有以下属性。

Property	Type	Description
maintenanceid	ID	维护的 ID。 Property behavior: - *read-only* - *required* for update operations
name	string	维护名称。 Property behavior: - *required* for create operations
active_since	timestamp	维护开始生效的时间（包含）。 给定值将向下舍入到分钟。 Property behavior: - *required* for create operations
active_till	timestamp	维护停止生效的时间（不包含）。 给定值将向下舍入到分钟。 Property behavior: - *required* for create operations
description	string	维护说明。
maintenance_type	integer	维护类型。 可能的值： 0 - *(default)* with data collection; 1 - without data collection.

Property	Type	Description
tags_evaltype	integer	问题标签的评估方法。 可能的值： 0 - *(default)* And/Or; 2 - Or.

时间段

时间段对象用于定义维护生效的时间段。它具有以下属性。

属性	类型	描述
period	integer	维护时间段的持续时间，单位为秒。 给定值将向下取整到分钟。 可能的取值范围：300 - 86399940。 默认值：3600。
timeperiod_type	integer	时间段类型。 可能的值： 0 - *(默认)* 仅一次； 2 - 每日； 3 - 每周； 4 - 每月。
start_date	timestamp	维护时间段生效的日期。 给定值将向下取整到分钟。 默认值：当前日期。 属性行为： - 当 `timeperiod_type` 设置为 “仅一次” 时，支持
start_time	integer	维护开始的每日时间，单位为秒。 给定值将向下取整到分钟。 默认值：0。 属性行为： - 当 `timeperiod_type` 设置为 “每日”、“每周” 或 “每月” 时，支持
every	integer	对于每日和每周时间段，`every` 定义维护生效的天或周间隔。 当 `timeperiod_type` 设置为 “每日” 或 “每周” 时，默认值：1。 对于设置了 `day` 的每月时间段，`every` 属性定义维护生效的每月第几天。 当 `timeperiod_type` 设置为 “每月” 且设置了 `day` 时，默认值：1。 对于设置了 `dayofweek` 的每月时间段，`every` 属性定义维护生效的每月第几周。 当 `timeperiod_type` 设置为 “每月” 且设置了 `dayofweek` 时，可能的值： 1 - *(默认)* 第一周； 2 - 第二周； 3 - 第三周； 4 - 第四周； 5 - 最后一周。 属性行为： - 当 `timeperiod_type` 设置为 “每日”、“每周” 或 “每月” 时，支持

属性	类型	描述
dayofweek	integer	维护生效的星期几。 可能的位图值： 1 - 星期一； 2 - 星期二； 4 - 星期三； 8 - 星期四； 16 - 星期五； 32 - 星期六； 64 - 星期日。 这是一个位掩码字段；可接受任意可能位图值之和（例如，21 表示星期一、星期三和星期五）。 属性行为： - 当 `timeperiod_type` 设置为 “每周” 时，或当 `timeperiod_type` 设置为 “每月” 且未设置 `day` 时，必需
day	integer	维护生效的每月日期。 属性行为： - 当 `timeperiod_type` 设置为 “每月” 且未设置 `dayofweek` 时，必需
month	integer	维护生效的月份。 可能的位图值： 1 - 一月； 2 - 二月； 4 - 三月； 8 - 四月； 16 - 五月； 32 - 六月； 64 - 七月； 128 - 八月； 256 - 九月； 512 - 十月； 1024 - 十一月； 2048 - 十二月。 这是一个位掩码字段；可接受任意可能位图值之和（例如，585 表示一月、四月、七月和十月）。 属性行为： - 当 `timeperiod_type` 设置为 “每月” 时，必需

事件名称

事件名称对象用于定义维护生效时必须抑制哪些问题。仅当Maintenance 对象 的 `maintenance_type` 设置为 “with data collection” 时，才能指定事件名称。它具有以下属性。

Property	Type	Description
operator	integer	条件运算符。 可能的值： 2 - （默认）包含； 3 - 不包含。
value	string	事件名称值。 属性行为： - 必需

问题标签

问题标签对象用于定义维护生效时必须抑制哪些问题。仅当维护对象的 `maintenance_type` 设置为 “带数据采集” 时，才能指定标签。它具有以下属性。

Property	Type	Description
tag	string	问题标签名称。 属性行为： - 必需
operator	integer	条件运算符。 可能的值： 0 - 等于； 1 - 不等于； 2 - （默认）包含； 3 - 不包含。
value	string	问题标签值。

创建

描述

`object maintenance.create(object/array maintenances)`

此方法允许您创建新的维护。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的维护。

除标准维护属性外，该方法还接受以下参数。

参数	类型	描述
groups	object/array	要置于维护状态的主机群组。 主机群组只能定义 `groupid` 属性。 参数行为： - 如果未设置 `hosts`，则为必需
hosts	object/array	要置于维护状态的主机。 主机只能定义 `hostid` 属性。 参数行为： - 如果未设置 `groups`，则为必需
triggers	object/array	要置于维护状态的触发器。 触发器只能定义 `triggerid` 属性。 参数行为： - 如果未设置 `groups` 和 `hosts`，则为必需 - 当 maintenance_type=0（收集数据）时支持
event_names	object/array	用于定义要置于维护状态的问题的事件名称。 参数行为： - 当 maintenance_type=0（收集数据）时支持
tags	object/array	用于定义要置于维护状态的问题的问题标签。 参数行为： - 当维护对象的 `maintenance_type` 设置为 “收集数据” 时支持

参数	类型	描述
timeperiods	object/array	维护时间段。 参数行为： - 必需

返回值

(object) 在 maintenanceids 属性中返回一个包含所有已被创建的维护模式的对象的 ID。返回的 IDs 的排序与传递的维护模式的 IDs 顺序一致。

案例

创建维护

创建一个维护，启用主机组（ID 为"20"）的数据采集，并设置问题标签 **service:mysqld** 和 **error**。
维护时间必须从 2026 年 3 月 17 日持续到 2027 年 3 月 17 日，每周日 18:00 生效，持续一小时。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Sunday maintenance",
        "active_since": 1773720240,
        "active_till": 1805256240,
        "tags_evaltype": 0,
        "groups": [
            {"groupid": "20"}
        ],
        "timeperiods": [
            {
                "period": 3600,
                "timeperiod_type": 3,
                "start_time": 64800,
                "every": 1,
                "dayofweek": 64
            }
        ],
        "tags": [
            {
                "tag": "service",
                "operator": "0",
                "value": "mysqld"
            },
            {
                "tag": "error",
                "operator": "2",
                "value": ""
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
```

```
    "id": 1
}
```

创建临时维护

为 ID 为”4” 的主机组、ID 为”12345” 的触发器以及事件名称 **Printer: ink low** 创建一个持续一小时且收集数据的临时维护。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Ad-hoc maintenance",
        "active_since": 1792057740,
        "active_till": 1792061340,
        "tags_evaltype": 0,
        "groups": [
            {"groupid": "4"}
        ],
        "triggers": [
            {"triggerid": "12345"}
        ],
        "timeperiods": [
            {
                "period": 3600,
                "timeperiod_type": 0,
                "start_date": 1792057740
            }
        ],
        "event_names": [
            {
                "operator": "2",
                "value": "Printer: ink low"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "4"
        ]
    },
    "id": 1
}
```

另请参阅

- 主机
- 主机组
- 触发器
- 事件名称
- 问题标签
- 时间段

来源

CMaintenance::create() in *ui/include/classes/api/services/CMaintenance.php*.

删除

描述

`object maintenance.delete(array maintenanceIds)`

此方法允许您删除维护周期。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的维护周期的 IDs。

返回值

(object) 在 maintenanceids 属性下返回包含已被删除的维护周期的 ID 对象。

案例

删除多个维护周期

删除两个维护周期。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.delete",
    "params": [
        "3",
        "1"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3",
            "1"
        ]
    },
    "id": 1
}
```

来源

CMaintenance::delete() in *frontends/php/include/classes/api/services/CMaintenance.php*.

更新

描述

`object maintenance.update(object/array maintenances)`

此方法允许您更新现有的维护。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的维护属性。

必须为每个维护定义 maintenanceid 属性，其他所有属性均为可选属性。仅更新传递的属性，其他属性保持不变。

除标准维护属性外，该方法还接受以下参数。

参数	类型	描述
groups	object/array	用于替换当前主机组的主机组。 主机组只能定义 groupid 属性。 参数行为： - 如果未设置 hosts，则为必需
hosts	object/array	用于替换当前主机的主机。 主机只能定义 hostid 属性。 参数行为： - 如果未设置 groups，则为必需
triggers	object/array	用于替换当前触发器的触发器。 触发器只能定义 triggerid 属性。 参数行为： - 如果未设置 groups 和 hosts，则为必需 - 如果 maintenance_type=0（收集数据），则为支持
event_names	object/array	用于替换当前事件名称的事件名称。 参数行为： - 如果 maintenance_type=0（收集数据），则为支持
tags	object/array	用于替换当前标签的问题标签。 参数行为： - 如果Maintenance 对象的 maintenance_type 设置为 “收集数据”，则为支持
timeperiods	object/array	用于替换当前时间段的维护时间段。

返回值

(object) 在 maintenanceids 属性中返回一个包含已被更新的维护模式的 IDs 的对象。

案例

指定不同的主机

将当前分配维护的主机替换为两台不同的主机。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.update",
    "params": {
        "maintenanceid": "3",
        "hosts": [
            {"hostid": "10085"},
            {"hostid": "10084"}
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
```

```
            ]
        },
        "id": 1
}
```

另请参阅

- 时间段

来源

CMaintenance::update() in *ui/include/classes/api/services/CMaintenance.php*.

获取

描述

`integer/array maintenance.get(object parameters)`

该方法允许您根据给定参数检索维护。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
groupids	ID/array	仅返回分配给指定主机组的维护。
hostids	ID/array	仅返回分配给指定主机的维护。
triggerids	ID/array	仅返回分配给指定触发器的维护。
maintenanceids	ID/array	仅返回具有指定 ID 的维护。
selectHostGroups	query	返回一个`hostgroups` 属性，其中包含分配给该维护的主机组。
selectHosts	query	返回一个`hosts` 属性，其中包含分配给该维护的主机。
selectTriggers	query	返回一个`triggers` 属性，其中包含分配给该维护的触发器。
selectEventNames	query	返回一个`event_names` 属性，其中包含分配给该维护的事件名称。
selectTags	query	返回一个`tags` 属性，其中包含该维护的问题标签。
selectTimeperiods	query	返回一个`timeperiods` 属性，其中包含该维护的时间段。
sortfield	string/array	按指定属性对结果进行排序。 可能的值：`maintenanceid`、`name`、`maintenance_type`、`active_since`、`active_till`。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下之一：

- 对象数组；
- 如果使用了 `countOutput` 参数，则返回检索到的对象数量。

案例

获取维护期

获取所有已配置的维护期、已分配维护范围的数据（主机组、主机、触发器、事件名称、标签）以及定义的时间段。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.get",
    "params": {
        "output": "extend",
        "selectHostGroups": "extend",
        "selectHosts": "extend",
        "selectTriggers": "extend",
        "selectEventNames": "extend",
        "selectTags": "extend",
        "selectTimeperiods": "extend"
    },
    "id": 1
}
```

响应：

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "maintenanceid": "1",
      "name": " 常规数据库维护",
      "maintenance_type": "0",
      "description": "",
      "active_since": "1791936000",
      "active_till": "2107555200",
      "tags_evaltype": "0",
      "hostgroups": [
        {
          "groupid": "20",
          "name": " 数据库",
          "flags": "0",
          "uuid": "748ad4d098d447d492bb935c907f652f"
        }
      ],
      "triggers": [],
      "event_names": [],
      "hosts": [],
      "tags": [],
      "timeperiods": [
        {
          "timeperiod_type": "4",
          "every": "1",
          "month": "4095",
          "dayofweek": "0",
          "day": "1",
          "start_time": "64800",
          "period": "3600",
          "start_date": "0"
        }
      ]
    },
    {
      "maintenanceid": "2",
      "name": " 临时维护：打印机：墨水耗尽",
      "maintenance_type": "0",
      "description": "",
```

```
      "active_since": "1792057740",
      "active_till": "1792061340",
      "tags_evaltype": "0",
      "hostgroups": [
        {
          "groupid": "4",
          "name": "Zabbix 服务器",
          "flags": "0",
          "uuid": "6f6799aa69e844b4b3918f779f2abf08"
        }
      ],
      "triggers": [],
      "event_names": [
        {
          "operator": "2",
          "value": " 打印机：墨水耗尽"
        }
      ],
      "hosts": [],
      "tags": [],
      "timeperiods": [
        {
          "timeperiod_type": "0",
          "every": "1",
          "month": "0",
          "dayofweek": "0",
          "day": "0",
          "start_time": "0",
          "period": "3600",
          "start_date": "1792057740"
        }
      ]
    }
  ],
  "id": 1
}
```


另请参阅

- 主机
- 主机组
- 触发器
- 事件名称
- 问题标签
- 时间段

源代码

位于 *ui/include/classes/api/services/CMaintenance.php* 中的 CMaintenance::get()。

脚本

此类用于处理脚本。

对象引用：

- Script
  - webhook 参数
- Debug
  - 日志条目

可用方法：

- script.create - 创建新脚本
- script.delete - 删除脚本
- script.execute - 运行脚本

- script.get - 检索脚本
- script.getscriptsbyevents - 检索事件的脚本
- script.getscriptsbyhosts - 检索主机的脚本
- script.update - 更新脚本

脚本对象

以下对象与 脚本 API 直接相关。

脚本

脚本对象具有以下属性。

属性	类型	描述
scriptid	ID	脚本的 ID。 属性行为： - 只读 - 更新操作时为必需
name	string	脚本名称。 属性行为： - 创建操作时为必需
type	integer	脚本类型。 如果 scope 设置为”action operation”，可能的值为： 0 - 脚本； 1 - IPMI； 2 - SSH； 3 - TELNET； 5 - webhook。 如果 scope 设置为”manual host action” 或”manual event action”，可能的值为： 6 - URL。
command	string	要运行的命令。 属性行为： - 创建操作时为必需 - 如果 type 设置为”Script”、”IPMI”、”SSH”、”TELNET” 或”Webhook”，则为必需
scope	integer	脚本范围。 可能的值： 1 - action operation； 2 - manual host action； 4 - manual event action。 属性行为： - 创建操作时为必需
execute_on	integer	运行脚本的位置。 可能的值： 0 - 在 Zabbix agent 上运行； 1 - 在 Zabbix 服务器上运行。仅当 Zabbix 服务器启用了全局脚本执行时才支持； 2 - （默认）在 Zabbix 服务器或 proxy 上运行。 属性行为： - 当 type 设置为”Script” 时支持

属性	类型	描述
menu_path	string	由斜杠分隔的文件夹。单击主机或事件时，这些文件夹会在前端中构成类似菜单的导航。 属性行为： - 当 scope 设置为”manual host action” 或”manual event action” 时支持
authtype	integer	SSH 脚本类型使用的身份验证方法。 可能的值： 0 - 密码； 1 - 公钥。
username	string	属性行为： - 当 type 设置为”SSH” 时支持 用于身份验证的用户名。
password	string	属性行为： - 当 type 设置为”SSH” 或”TELNET” 时为必需 使用密码身份验证的 SSH 脚本和 TELNET 脚本所使用的密码。
publickey	string	属性行为： - 当 type 设置为”SSH” 且 authtype 设置为”password”，或 type 设置为”TELNET” 时支持 使用公钥身份验证的 SSH 脚本所使用的公钥文件名称。
privatekey	string	属性行为： - 当 type 设置为”SSH” 且 authtype 设置为”public key” 时为必需 使用公钥身份验证的 SSH 脚本所使用的私钥文件名称。
port	string	属性行为： - 当 type 设置为”SSH” 且 authtype 设置为”public key” 时为必需 SSH 和 TELNET 脚本使用的端口号。
groupid	ID	属性行为： - 当 type 设置为”SSH” 或”TELNET” 时支持 脚本可以在其上运行的主机组的 ID。 如果设置为”0”，则所有主机组都可以使用该脚本。 默认值：0。
usrgrpid	ID	允许运行脚本的用户组的 ID。 如果设置为”0”，则所有用户组都可以使用该脚本。 默认值：0。
host_access	integer	属性行为： - 当 scope 设置为”manual host action” 或”manual event action” 时支持 运行脚本所需的主机权限。 可能的值： 2 - （默认）读取； 3 - 写入。
confirmation	string	属性行为： - 当 scope 设置为”manual host action” 或”manual event action” 时支持 确认弹窗文本。 尝试从 Zabbix 前端运行脚本时，将显示该弹窗。 属性行为： - 当 scope 设置为”manual host action” 或”manual event action” 时支持

属性	类型	描述
timeout	string	webhook 脚本执行超时时间，单位为秒。支持时间后缀（例如 30s、1m）。 可能的值：1-60s。 默认值：30s。
parameters	array	属性行为： - 当 type 设置为”Webhook” 时为必需 webhook 输入参数数组。
description	string	属性行为： - 当 type 设置为”Webhook” 时支持 脚本描述。
url	string	用户定义的 URL。 属性行为： - 当 type 设置为”URL” 时为必需
new_window	integer	在新窗口中打开 URL。 可能的值： 0 - 否； 1 - （默认）是。 属性行为： - 当 type 设置为”URL” 时支持
manualinput	integer	指示脚本是否接受用户提供的输入。 可能的值： 0 - （默认）禁用； 1 - 启用； 属性行为： - 当 scope 设置为”manual host action” 或”manual event action” 时支持
manualinput_prompt	string	手动输入提示文本。 属性行为： - 当 manualinput 设置为”Enabled” 时为必需
manualinput_validator	string	用于验证用户提供的输入的字符串字段。该字符串由正则表达式或以逗号分隔的一组值组成。 属性行为： - 当 manualinput 设置为”Enabled” 时为必需
manualinput_validator_type	integer	确定所需的用户输入类型。 可能的值： 0 - （默认）字符串。表示将 manualinput_validator 视为正则表达式； 1 - 列表。表示将 manualinput_validator 视为以逗号分隔的可能输入值列表。 属性行为： - 当 manualinput 设置为”Enabled” 时支持
manualinput_default_value	string	用于自动填充用户输入的默认值。 属性行为： - 当 manualinput_validator_type 设置为”String” 时支持

webhook 参数

调用 webhook 脚本时传递的参数具有以下属性。

属性	Type	说明
name	string	参数名称。 Property behavior: - *required*
value	string	参数值。支持宏。

调试

已执行的 webhook 脚本的调试信息。debug 对象具有以下属性。

属性	Type	说明
logs	array	日志条目数组。
ms	string	脚本执行持续时间，单位为毫秒。

日志条目

日志条目对象具有以下属性。

属性	类型	描述
level	integer	日志级别。
ms	string	从脚本运行开始到添加日志条目为止所经过的时间（以毫秒为单位）。
message	string	日志消息。

创建

描述

`object script.create(object/array scripts)`

此方法允许您创建新的脚本。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

（对象/数组）要创建的脚本。

此方法接受具有标准脚本属性的脚本。

返回值

（对象）返回一个 `scriptids` 属性包含被创建脚本 ID 的对象。返回的 ID 顺序与传入脚本的顺序一致。

示例

创建 webhook 脚本

创建一个 webhook 脚本，用于向外部服务发送 HTTP 请求。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Webhook script",
        "command": "try {\n var request = new HttpRequest(),\n response,\n data;\n\n request.addHeader('Co
        "scope": 1,
        "type": 5,
        "timeout": "40s",
        "parameters": [
            {
                "name": "token",
```

```
                "value": "{$WEBHOOK.TOKEN}"
            },
            {
                "name": "host",
                "value": "{HOST.HOST}"
            },
            {
                "name": "v",
                "value": "2.2"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3"
        ]
    },
    "id": 1
}
```

创建 SSH 脚本

创建一个 SSH 脚本，通过公钥完成身份验证从而可以在主机运行，并且带有上下文菜单。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "SSH script",
        "command": "my script command",
        "type": 2,
        "authtype": 1,
        "username": "John",
        "publickey": "pub.key",
        "privatekey": "priv.key",
        "password": "secret",
        "port": "12345",
        "scope": 2,
        "menu_path": "All scripts/SSH",
        "usrgrpid": "7",
        "groupid": "4"
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "5"
        ]
    },
    "id": 1
}
```

创建定制脚本

创建一个重启服务器的定制脚本。该脚本会要求主机的写权限，并且在前端运行之前会显示配置信息。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Reboot server",
        "command": "reboot server {MANUALINPUT}",
        "type": 0,
        "scope": 2,
        "confirmation": "Are you sure you would like to reboot the server {MANUALINPUT}?",
        "manualinput": 1,
        "manualinput_prompt": "Which server you want to reboot?",
        "manualinput_validator": "[1-9]",
        "manualinput_validator_type": 0,
        "manualinput_default_value": "1"
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "4"
        ]
    },
    "id": 1
}
```

创建一个 URL 类型的脚本

为主为作用域创建一个 URL 类型的脚本，该脚本会要求主机的写权限，并且在前端运行之前会显示配置信息。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "URL script",
        "type": 6,
        "scope": 2,
        "url": "http://zabbix/ui/zabbix.php?action=host.edit&hostid={HOST.ID}",
        "confirmation": "Edit host {HOST.NAME}?",
        "new_window": 0
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "56"
        ]
    },
    "id": 1
}
```

创建一个手动输入的 URL 类型脚本

为在新窗口中打开并具有手动输入的事件作用域创建 URL 类型脚本。请求:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "script.create",
    "params": {
        "name": "URL script with manual input",
        "type": 6,
        "scope": 4,
        "url": "http://zabbix/ui/zabbix.php?action={MANUALINPUT}",
        "new_window": 1,
        "manualinput": 1,
        "manualinput_prompt": "Select a page to open:",
        "manualinput_validator": "dashboard.view,script.list,actionlog.list",
        "manualinput_validator_type": 1
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "57"
        ]
    },
    "id": 1
}
```

源代码

CScript::create() 在 *ui/include/classes/api/services/CScript.php*。

删除

描述

`object script.delete(array scriptIds)`

此方法允许您删除脚本。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

（数组）要删除的脚本 ID。

返回值

（对象）返回一个 `scriptids` 属性包含被删除脚本 ID 的对象。

示例

删除多个脚本

删除两个脚本请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.delete",
    "params": [
        "3",
        "4"
    ],
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3",
            "4"
        ]
    },
    "id": 1
}
```

源代码

CScript::delete() 在 *frontends/php/include/classes/api/services/CScript.php*。

执行

描述

`object script.execute(object parameters)`

此方法允许您在主机或事件上运行脚本。URL 类型的脚本除外；这类脚本不可执行。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 包含要运行的脚本 ID，以及主机 ID 或事件 ID 和 manualinput 值的参数。

参数	类型	描述
scriptid	ID	要运行的脚本的 ID。 参数行为: - 必需
hostid	ID	要在其上运行脚本的主机 ID。 参数行为: - 如果未设置 `eventid`，则为 必需
eventid	ID	要在其上运行脚本的事件 ID。 参数行为: - 如果未设置 `hostid`，则为 必需
manualinput	string	用户提供的用于运行脚本的值，替换 {MANUALINPUT} 宏。

返回值

`(object)` 返回脚本执行结果。

属性	Type	描述
response	string	脚本是否成功运行。 可能的值 - `success`。
value	string	脚本输出。
debug	object	如果执行的是 webhook 脚本，则包含一个debug 对象。对于其他脚本类型，则包含空对象。

示例

运行 webhook 脚本

运行一个向外部服务发送 HTTP 请求的 webhook 脚本。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
```

```
        "scriptid": "4",
        "hostid": "30079"
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "{\"status\":\"sent\",\"timestamp\":\"1611235391\"}",
        "debug": {
            "logs": [
                {
                    "level": 3,
                    "ms": 480,
                    "message": "[Webhook Script] HTTP status: 200."
                }
            ],
            "ms": 495
        }
    },
    "id": 1
}
```

运行自定义脚本

在主机上运行一个 “ping” 脚本。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "1",
        "hostid": "30079"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes from 127.0.0.1: icmp_req=1 tt
        "debug": []
    },
    "id": 1
}
```

使用手动输入运行自定义脚本

在主机上运行一个 “ping” 脚本，命令为 "ping -c {MANUALINPUT} {HOST.CONN}; case $? in [01]) true;; *) false;; esac"。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "7",
        "hostid": "30079",
```

```
        "manualinput": "2"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes from 127.0.0.1: icmp_seq=1 tt
        "debug": []
    },
    "id": 1
}
```

源代码

CScript::execute() 在 *ui/include/classes/api/services/CScript.php*。

更新

描述

`object script.update(object/array scripts)`

此方法允许您更新现有脚本。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 待更新的脚本属性

所有脚本必须定义 `scriptid` 属性，其他所有属性都是可选的。只有被传递的属性会被更新，其他所有属性保持不变。例外是 `type` 属性会从 5（Webhook）变成其他：`parameters` 属性会被清除。

返回值

(object) 返回一个 `scriptids` 属性包含被更新脚本 ID 的对象。

示例

修改脚本命令

把脚本命令修改为 “/bin/ping -c 10 {HOST.CONN} 2>&1”。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.update",
    "params": {
        "scriptid": "1",
        "command": "/bin/ping -c 10 {HOST.CONN} 2>&1"
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "1"
        ]
    },
```

```
    "id": 1
}
```

更改脚本命令并添加手动输入

改变脚本命令"/bin/ping -c {MANUALINPUT} {HOST.CONN} 2>&1". 请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.update",
    "params": {
      "scriptid": "1",
      "command": "/bin/ping -c {MANUALINPUT} {HOST.CONN} 2>&1",
      "manualinput": "1",
      "manualinput_prompt": "Specify the number of ICMP packets to send with the ping command",
      "manualinput_validator": "^(?:[1-9]|10)$",
      "manualinput_validator_type": "0",
      "manualinput_default_value": "10"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "1"
        ]
    },
    "id": 1
}
```

源代码

CScript::update() 在 *frontends/php/include/classes/api/services/CScript.php*。

根据事件获取脚本

描述

`object script.getscriptsbyevents(object parameters)`

此方法允许您检索给定事件上的所有可用脚本；如果提供了脚本 ID，则可检索特定脚本。当提供 manualinput 时，它会将 {MANUALINPUT} 宏替换为指定的值。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 该方法接受对象或对象数组，包含以下参数。

参数	类型	说明
eventid	ID	要返回其脚本的事件 ID。 必须唯一。 参数行为： - 必填
scriptid	ID	要返回的脚本 ID。
manualinput	string	用户提供的 {MANUALINPUT} 宏值。

返回值

(object) 返回一个对象，该对象的属性和数组为事件 ID 可用的脚本作为值。如果提供了脚本 ID，则关联的值是一个包含特定脚本的数组。

该方法将自动展开 “确认” 文本中的宏，手动输入提示 text 和 `url`。

如果提供了 manualinput 参数，{MANUALPINPUT} 宏将解析为指定值。

示例

按事件 ID 检索脚本

检索事件 ID 为”632” and ”614” 的全部脚本请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyevents",
    "params": [
      {
         "eventid":  "632"
      },
      {
         "eventid":  "614"
      }
    ],
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "632": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "4",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "manualinput": "0",
                "manualinput_prompt": "",
                "manualinput_validator_type": "0",
                "manualinput_validator": "",
                "manualinput_default_value": "",
                "parameters": []
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
```

```
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "4",
            "name": "Open Zabbix page",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Are you sure you want to open page *UNKNOWN*?",
            "type": "6",
            "execute_on": "2",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "http://localhost/ui/zabbix.php?action=*UNKNOWN*",
            "new_window": "1",
            "manualinput": "1",
            "manualinput_prompt": "Zabbix page to open:",
            "manualinput_validator_type": "1",
            "manualinput_validator": "dashboard.view,discovery.view",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
```

```
                "timeout": "30s",
                "scope": "4",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "manualinput": "0",
                "manualinput_prompt": "",
                "manualinput_validator_type": "0",
                "manualinput_validator": "",
                "manualinput_default_value": "",
                "parameters": []
            }
        ],
        "614": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "4",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "manualinput": "0",
                "manualinput_prompt": "",
                "manualinput_validator_type": "1",
                "manualinput_validator": "",
                "manualinput_default_value": "",
                "parameters": []
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "4",
```

```
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "4",
            "name": "Open Zabbix page",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Are you sure you want to open page *UNKNOWN*?",
            "type": "6",
            "execute_on": "2",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "http://localhost/ui/zabbix.php?action=*UNKNOWN*",
            "new_window": "1",
            "manualinput": "1",
            "manualinput_prompt": "Zabbix page to open:",
            "manualinput_validator_type": "1",
            "manualinput_validator": "dashboard.view,discovery.view",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
```

```
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "manualinput": "0",
                "manualinput_prompt": "",
                "manualinput_validator_type": "0",
                "manualinput_validator": "",
                "manualinput_default_value": "",
                "parameters": []
            }
        ]
    },
    "id": 1
}
```

检索具有 manualinput 值的特定脚本。

检索具有手动输入值 “dashboard.view” 的事件 “632” 的 ID 为 “4” 的脚本。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyevents",
    "params": [
      {
        "eventid":  "632",
        "scriptid": "4",
        "manualinput": "dashboard.view"
      }
  ],
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
      "632": [
        {
          "scriptid": "4",
          "name": "Open Zabbix page",
          "command": "",
          "host_access": "2",
          "usrgrpid": "0",
          "groupid": "0",
          "description": "",
          "confirmation": "Are you sure you want to open page dashboard.view?",
          "type": "6",
          "execute_on": "2",
          "timeout": "30s",
          "scope": "4",
          "port": "",
          "authtype": "0",
          "username": "",
          "password": "",
          "publickey": "",
          "privatekey": "",
          "menu_path": "",
          "url": "http://localhost/ui/zabbix.php?action=dashboard.view",
          "new_window": "1",
          "manualinput": "1",
          "manualinput_prompt": "Zabbix page to open:",
          "manualinput_validator_type": "1",
```

```
            "manualinput_validator": "dashboard.view,discovery.view",
            "manualinput_default_value": "",
            "parameters": []
          }
        ]
    },
    "id": 1
}
```

来源

CScript::getScriptsByEvents() in *ui/include/classes/api/services/CScript.php*.

获取脚本

描述

`integer/array script.get(object parameters)`

该方法允许您根据给定参数检索脚本。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
groupids	ID/array	仅返回可在给定主机组上运行的脚本。
hostids	ID/array	仅返回可在给定主机上运行的脚本。
scriptids	ID/array	仅返回具有给定 ID 的脚本。
usrgrpids	ID/array	仅返回可由给定用户组中的用户运行的脚本。
selectHostGroups	query	返回`hostgroups` 属性，其中包含该脚本可运行的主机组。
selectHosts	query	返回`hosts` 属性，其中包含该脚本可运行的主机。
selectActions	query	返回`actions` 属性，其中包含与该脚本关联的动作。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`scriptid`、`name`。
countOutput	boolean	这些参数在参考注释中进行了说明。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(整型/数组) 返回其中之一：- 一组对象; - 如果用到 `countOutput` 参数，被检索到的对象数,。

示例

检索所有脚本

检索所有配置的脚本。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.get",
```

```
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "scriptid": "1",
            "name": "Ping",
            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator": "",
            "manualinput_validator_type": "0",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
```

```
            "manualinput_prompt": "",
            "manualinput_validator": "",
            "manualinput_validator_type": "0",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "3",
            "name": "Detect operating system",
            "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator": "",
            "manualinput_validator_type": "0",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "4",
            "name": "Webhook",
            "command": "try {\n var request = new HttpRequest(),\n response,\n data;\n\n request.addHeader
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "5",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator": "",
            "manualinput_validator_type": "0",
            "manualinput_default_value": "",
```

```
            "parameters": [
                {
                    "name": "token",
                    "value": "{$WEBHOOK.TOKEN}"
                },
                {
                    "name": "host",
                    "value": "{HOST.HOST}"
                },
                {
                    "name": "v",
                    "value": "2.2"
                }
            ]
        },
        {
            "scriptid": "5",
            "name": "URL",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Go to {HOST.NAME}?",
            "type": "6",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "http://zabbix/ui/zabbix.php?action=latest.view&hostids[]={HOST.ID}",
            "new_window": "0",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator": "",
            "manualinput_validator_type": "0",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "6",
            "name": "URL with user input",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Open zabbix page {MANUALINPUT}?",
            "type": "6",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
```

```
            "privatekey": "",
            "menu_path": "",
            "url": "http://zabbix/ui/zabbix.php?action={MANUALINPUT}",
            "new_window": "0",
            "manualinput": "1",
            "manualinput_prompt": "Select a page to open:",
            "manualinput_validator": "dashboard.view,script.list,actionlog.list",
            "manualinput_validator_type": "1",
            "parameters": []
        }
    ],
    "id": 1
}
```

另请参见

- 主机
- 主机组

源代码

CScript::get() 在 *frontends/php/include/classes/api/services/CScript.php*.

通过主机获取脚本

描述

`object script.getscriptsbyhosts(object parameters)`

此方法允许您检索给定主机上的所有可用脚本；如果提供了脚本 ID，则可检索特定脚本。当提供 manualinput 时，它会将 {MANUALINPUT} 宏替换为指定的值。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 该方法接受对象或对象数组，包含以下参数。

参数	类型	描述
hostid	ID	要返回其脚本的主机 ID。 必须唯一。 参数行为: - 必填
scriptid	ID	要返回的脚本 ID。
manualinput	string	用户提供的 {MANUALINPUT} 宏值。

返回值

(object) 返回一个对象，其中以主机 ID 作为属性，可用脚本数组作为值。如果提供了脚本 ID，则对应的值为包含该特定脚本的数组。

Note:
该方法会自动展开 confirmation 文本、manualinput prompt 文本和 url 中的宏。
如果提供了 manualinput 参数，则 {MANUALINPUT} 宏将被解析为指定的值。

示例

通过主机 ID 检索脚本

检索主机 “30079” 和 “30073” 上可用的所有脚本。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyhosts",
    "params": [
```

```
        {
            "hostid":  "30079"
        },
        {
            "hostid":  "30073"
        }
    ],
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
        "30079": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "manualinput": "0",
                "manualinput_prompt": "",
                "manualinput_validator_type": "0",
                "manualinput_validator": "",
                "manualinput_default_value": "",
                "parameters": []
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
```

```
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "4",
            "name": "Open Zabbix page",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Are you sure you want to open page *UNKNOWN*?",
            "type": "6",
            "execute_on": "2",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "http://localhost/ui/zabbix.php?action=*UNKNOWN*",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "Zabbix page to open:",
            "manualinput_validator_type": "0",
            "manualinput_validator": "dashboard.view,discovery.view",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
```

```
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
        }
    ],
    "30073": [
        {
            "scriptid": "3",
            "name": "Detect operating system",
            "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "1",
            "name": "Ping",
            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
```

```
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "4",
            "name": "Open Zabbix page",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Are you sure you want to open page *UNKNOWN*?",
            "type": "6",
            "execute_on": "2",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "http://localhost/ui/zabbix.php?action=*UNKNOWN*",
            "new_window": "1",
            "manualinput": "1",
            "manualinput_prompt": "Zabbix page to open:",
            "manualinput_validator_type": "1",
            "manualinput_validator": "dashboard.view,discovery.view",
            "manualinput_default_value": "",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "manualinput": "0",
            "manualinput_prompt": "",
            "manualinput_validator_type": "0",
            "manualinput_validator": "",
            "manualinput_default_value": "",
            "parameters": []
```

```
            }
        ]
    },
    "id": 1
}
```

检索具有手工输入值的特定脚本。

检索主机 “30079” 上 ID 为 “4”、手动输入值为 “dashboard.view” 的脚本。请求:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyhosts",
    "params": [
      {
        "hostid":  "30079",
        "scriptid": "4",
        "manualinput": "dashboard.view"
      }
  ],
    "id": 1
}
```

响应

```
{
    "jsonrpc": "2.0",
    "result": {
      "30079": [
        {
          "scriptid": "4",
          "name": "Open Zabbix page",
          "command": "",
          "host_access": "2",
          "usrgrpid": "0",
          "groupid": "0",
          "description": "",
          "confirmation": "Are you sure you want to open page dashboard.view?",
          "type": "6",
          "execute_on": "2",
          "timeout": "30s",
          "scope": "2",
          "port": "",
          "authtype": "0",
          "username": "",
          "password": "",
          "publickey": "",
          "privatekey": "",
          "menu_path": "",
          "url": "http://localhost/ui/zabbix.php?action=dashboard.view",
          "new_window": "1",
          "manualinput": "1",
          "manualinput_prompt": "Zabbix page to open:",
          "manualinput_validator_type": "1",
          "manualinput_validator": "dashboard.view,discovery.view",
          "manualinput_default_value": "",
          "parameters": []
        }
      ]
    },
    "id": 1
}
```

源代码

CScript::getScriptsByHosts() 在 *frontends/php/include/classes/api/services/CScript.php*。

自动注册

此类设计用于处理自动注册。

对象引用：

- 自动注册

可用的方法：

- autoregistration.get - 检索自动注册信息
- autoregistration.update - 更新自动注册信息

自动注册对象

以下对象与 `autoregistration` API 直接相关。

自动注册

自动注册对象具有以下属性。

属性	类型	描述
tls_accept	integer	自动注册允许的传入连接类型。 可能的值： 1 - 允许未加密连接； 2 - 允许使用 PSK 的 TLS； 3 - 同时允许未加密连接和使用 PSK 的 TLS 连接。
tls_psk_identity	string	PSK 标识；必须且只能与一个 PSK 配对使用（在autoregistration、hosts 和proxies 中）。 请勿在 PSK 标识中包含敏感信息，因为它会通过未加密的网络发送，以告知接收方应使用哪个 PSK。 属性行为： - 仅写
tls_psk	string	预共享密钥（PSK）；必须至少为 32 个十六进制数字。 属性行为： - 仅写

更新

描述

`object autoregistration.update(object autoregistration)`

此方法允许您更新现有的自动注册。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object)` 要更新的自动注册属性。

返回值

`(boolean )` 成功更新后返回布尔值 true 作为结果。

示例

请求:

```
{
    "jsonrpc": "2.0",
    "method": "autoregistration.update",
    "params": {
        "tls_accept": "3",
        "tls_psk_identity": "PSK 001",
        "tls_psk": "11111595725ac58dd977beef14b97461a7c1045b9a1c923453302c5473193478"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

源码位置

CAutoregistration::update() in *ui/include/classes/api/services/CAutoregistration.php* 文件中。

获取

描述

`object autoregistration.get(object parameters)`

该方法允许你根据给定参数检索自动注册对象。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 用于定义所需输出的参数。

该方法仅支持一个参数。

参数	Type	描述
output	query	该参数在参考注释中进行了说明。

返回值

(object) 返回自动注册对象。

示例

请求:

```
{
   "jsonrpc": "2.0",
    "method": "autoregistration.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tls_accept": "3"
    },
```

```
    "id": 1
}
```

源码位置

CAutoregistration::get() in *ui/include/classes/api/services/CAutoregistration.php* 文件中。

角色

此类用于处理用户角色。

对象引用：

- 角色
- 角色规则
  - UI 元素
  - 服务
  - 服务标签
  - 模块
  - 操作

可用方法：

- role.create - 创建新的用户角色
- role.delete - 删除用户角色
- role.get - 获取用户角色
- role.update - 更新用户角色

角色对象
以下内容将详细介绍有关 角色 API 的相关功能。

角色

角色对象具有以下属性：

属性	类型	描述
roleid	ID	角色的 ID。 属性行为： - 只读 - 更新操作时为必填项
name	string	角色名称。 属性行为： - 创建操作时为必填项
type	integer	用户类型。 可能的值： 1 - (默认) 用户； 2 - 管理员； 3 - 超级管理员。 属性行为： - 创建操作时为必填项
readonly	integer	角色是否为只读。 可能的值： 0 - (默认) 否； 1 - 是。 属性行为： - 只读

角色规则

角色规则对象具有以下属性：

属性	类型	描述
ui	array	UI 元素对象数组。
ui.default_access	integer	是否启用对新 UI 元素的访问权限。 可能的值： 0 - 禁用； 1 - *(默认)* 启用。
services.read.mode	integer	对服务的只读访问权限。 可能的值： 0 - 仅对 `services.read.list` 指定或与 `services.read.tag` 属性匹配的服务具有只读访问权限； 1 - *(默认)* 对所有服务具有只读访问权限。
services.read.list	array	服务对象数组。 指定的服务（包括子服务）将被授予用户角色只读访问权限。只读访问权限不会覆盖对服务的读写访问权限。 属性行为： - 当 `services.read.mode` 设置为"0" 时支持
services.read.tag	object	服务标签对象数组。 与标签匹配的服务（包括子服务）将被授予用户角色只读访问权限。只读访问权限不会覆盖对服务的读写访问权限。 属性行为： - 当 `services.read.mode` 设置为"0" 时支持
services.write.mode	integer	对服务的读写访问权限。 可能的值： 0 - *(默认)* 仅对 `services.write.list` 指定或与 `services.write.tag` 属性匹配的服务具有读写访问权限； 1 - 对所有服务具有读写访问权限。
services.write.list	array	服务对象数组。 指定的服务（包括子服务）将被授予用户角色读写访问权限。读写访问权限将覆盖对服务的只读访问权限。 属性行为： - 当 `services.write.mode` 设置为"0" 时支持
services.write.tag	object	服务标签对象数组。 与标签匹配的服务（包括子服务）将被授予用户角色读写访问权限。读写访问权限将覆盖对服务的只读访问权限。 属性行为： - 当 `services.write.mode` 设置为"0" 时支持
modules	array	模块对象数组。
modules.default_access	integer	是否启用对新模块的访问权限。 可能的值： 0 - 禁用； 1 - *(默认)* 启用。
api.access	integer	是否启用 API 访问权限。 可能的值： 0 - *(默认)* 禁用； 1 - 启用。

属性	类型	描述
api.mode	integer	处理 `api` 属性中列出的 API 方法的模式。 可能的值： 0 - *(默认)* 拒绝列表； 1 - 允许列表。
api	array	API 方法数组。
actions	array	操作对象数组。
actions.default_access	integer	是否启用对新操作的访问权限。 可能的值： 0 - 禁用； 1 - *(默认)* 启用。
devices.access	integer	是否启用对用户自有移动设备的访问权限，以便与Zabbix 移动应用配合使用。 可能的值： 0 - （*User* 和 *Admin* 的默认值）禁用； 1 - （*Super admin* 的默认值）启用。 属性行为： - 仅当`EnableMobileDevices` 启用时支持
devices.actions	array	设备操作对象数组。 属性行为： - 当 `devices.access` 设置为"1" 时支持
devices.actions.default_access	integer	是否启用对新设备操作的访问权限。 可能的值： 0 - （*User* 和 *Admin* 的默认值）禁用； 1 - （*Super admin* 的默认值）启用。 属性行为： - 仅当`EnableMobileDevices` 启用时支持

UI 元素

UI 元素对象具有以下属性：

Property	Type	Description
name	string	UI 元素的名称。 如果Role 对象的 type 设置为”User”、”Admin” 或”Super admin”，可能的值包括： monitoring.dashboard - 监控 > 仪表板； monitoring.problems - 监控 > 问题； monitoring.hosts - 监控 > 主机； monitoring.latest_data - 监控 > 最新数据； monitoring.maps - 监控 > 拓扑图； services.services - 服务 > 服务； services.sla_report - 服务 > *SLA* 报告； inventory.overview - 资产清单 > 概览； inventory.hosts - 资产清单 > 主机； reports.availability_report - 报表 > 可用性报告； reports.top_triggers - 报表 > 触发器 *Top 100*。 如果Role 对象的 type 设置为”Admin” 或”Super admin”，可能的值包括： monitoring.discovery - 监控 > 发现； services.sla - 服务 > *SLA*； reports.scheduled_reports - 报表 > 计划报表； reports.notifications - 报表 > 通知； configuration.template_groups - 数据采集 > 模板组； configuration.host_groups - 数据采集 > 主机组； configuration.templates - 数据采集 > 模板； configuration.hosts - 数据采集 > 主机； configuration.maintenance - 数据采集 > 维护； configuration.discovery - 数据采集 > 发现； configuration.trigger_actions - 告警 > 动作 > 触发器动作； configuration.service_actions - 告警 > 动作 > 服务动作； configuration.discovery_actions - 告警 > 动作 > 发现动作； configuration.autoregistration_actions - 告警 > 动作 > 自动注册动作； configuration.internal_actions - 告警 > 动作 > 内部动作。 如果Role 对象的 type 设置为”Super admin”，可能的值包括： reports.system_info - 报表 > 系统信息； reports.audit - 报表 > 审计日志； reports.action_log - 报表 > 动作日志； configuration.event_correlation - 数据采集 > 事件关联； administration.media_types - 告警 > 媒介类型； administration.scripts - 告警 > 脚本； administration.user_groups - 用户 > 用户组； administration.user_roles - 用户 > 用户角色； administration.users - 用户 > 用户； administration.api_tokens - 用户 > *API* 令牌； administration.authentication - 用户 > 身份验证； administration.linked_devices - 用户 > 设备； administration.general - 管理 > 常规； administration.audit_log - 管理 > 审计日志； administration.housekeeping - 管理 > 数据保留； administration.proxy_groups - 管理 > *proxy* 组； administration.proxies - 管理 > *proxy*； administration.macros - 管理 > 宏； administration.queue - 管理 > 队列。 属性行为： - *必需*

Property	Type	Description
status	integer	UI 元素的访问权限是否已启用。 可能的值： 0 - 已禁用； 1 - (默认) 已启用。

服务

属性	类型	描述
serviceid	ID	服务的 ID。 属性行为： - 必需

服务标签

Property	Type	Description
tag	string	标签名称。 如果指定为空字符串，则服务标签将不会用于服务匹配。 属性行为： - 必需
value	string	标签值。 如果未指定值或指定为空字符串，则仅标签名称将用于服务匹配。

模块

模块对象具有以下属性：

属性	类型	描述
moduleid	ID	模块的 ID。 属性行为： - 必需
status	integer	是否启用对该模块的访问。 可能的值： 0 - 已禁用； 1 - (默认) 已启用。

操作

操作对象具有以下属性：

属性	类型	描述
name	string	操作的名称。 如果Role 对象的 type 设置为”User”、”Admin” 或”Super admin”，可能的值包括： edit_dashboards - 创建和编辑仪表板； edit_maps - 创建和编辑地图； add_problem_comments - 添加问题评论； change_severity - 更改问题严重性； acknowledge_problems - 确认问题； suppress_problems - 抑制问题； close_problems - 关闭问题； execute_scripts - 执行脚本； manage_api_tokens - 管理 API 令牌； change_problem_ranking - 将问题排名从原因更改为症状，反之亦然； edit_own_media - 允许用户创建和编辑自己的媒介。 如果Role 对象的 type 设置为”Admin” 或”Super admin”，可能的值包括： edit_maintenance - 创建和编辑维护； manage_scheduled_reports - 管理计划报表， manage_sla - 管理 SLA， select_server_for_monitoring - 允许用户创建由服务器监控的主机，以及由服务器执行的网络发现规则监控的主机。 如果Role 对象的 type 设置为”User” 或”Admin”，可能的值包括： invoke_execute_now - 允许仅对主机具有读取权限的用户执行监控项检查。 如果Role 对象的 type 设置为”Super admin”，可能的值包括： edit_user_media - 允许用户为其他用户创建和编辑媒介。 属性行为： - 必需
status	integer	是否启用执行该操作的访问权限。 可能的值： 0 - 已禁用； 1 - (默认) 已启用。

设备操作

设备操作对象具有以下属性：

Property	Type	Description
name	string	设备操作的名称。 如果Role 对象的 type 设置为”User”、”Admin” 或”Super admin”，可能的值： manage_own - 允许用户添加或移除自己的移动设备。 如果Role 对象的 type 设置为”Super admin”，可能的值： manage_user - 允许用户为所有用户添加或移除移动设备。 属性行为： - *required*
status	integer	是否启用执行该操作的访问权限。 可能的值： 0 - 已禁用； 1 - （默认）已启用。

创建

描述

`object role.create(object/array roles)`

此方法允许您创建新角色。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(`object/array`) 要创建的角色。

除了标准角色属性之外，该方法还接受以下参数。

参数	Type	描述
rules	array	要为该角色创建的角色规则。

返回值

根据 `roleids` 特性，(`object`) 会返回一个对象，包含创建的角色用户的 ID 号。返回的 ID 号顺序对应查询的角色用户次序。

示例

创建角色

创建一个类型为 “User” 的角色，并禁止其访问两个 UI 元素。请求:

```
{
    "jsonrpc": "2.0",
    "method": "role.create",
    "params": {
        "name": "Operator",
        "type": "1",
        "rules": {
            "ui": [
                {
                    "name": "monitoring.hosts",
                    "status": "0"
                },
                {
                    "name": "monitoring.maps",
                    "status": "0"
                }
            ]
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

另请参阅

- 角色规则

- UI 元素
- 模块
- 操作

来源

CRole::create() in *ui/include/classes/api/services/CRole.php*.

删除

描述

`object role.delete(array roleids)`

此方法允许您删除角色。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(array) 要删除的角色用户 ID

返回值

根据 `roleids` 特性，(object) 会返回一个对象，包含删除的角色用户的 ID 号。

示例

删除多个用户角色

删除两个用户角色。请求:

```
{
    "jsonrpc": "2.0",
    "method": "role.delete",
    "params": [
        "4",
        "5"
    ],
    "id": 1
}
```

Response: 响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

参考来源

CRole::delete() in *ui/include/classes/api/services/CRole.php*.

更新

描述

`object role.update(object/array roles)`

此方法允许您更新现有角色。

Note:
此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的角色属性。

必须为每个角色定义 roleid 属性，所有其他属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准角色属性之外，该方法还接受以下参数。

参数	Type	说明
rules	array	要为该角色更新的访问rules。

返回值

根据 roleids 的特性，(object) 返回一个包含已更新职责用户 ID 的对象

示例

关闭运行脚本功能

更新 ID 号为 5 的角色，关闭其运行脚本的功能。请求:

```
{
    "jsonrpc": "2.0",
    "method": "role.update",
    "params": [
        {
            "roleid": "5",
            "rules": {
                "actions": [
                    {
                        "name": "execute_scripts",
                        "status": "0"
                    }
                ]
            }
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

限制对 API 的访问

更新 ID 号为 5 的角色，拒绝任何” 创建”, ” 更新” 或者” 删除” 方式。请求:

```
{
    "jsonrpc": "2.0",
    "method": "role.update",
    "params": [
        {
            "roleid": "5",
            "rules": {
                "api.access": "1",
```

```
                "api.mode": "0",
                "api": ["*.create", "*.update", "*.delete"]
            }
        }
    ],
    "id": 1
}
```

Response: 响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

参考来源

CRole::update() in *ui/include/classes/api/services/CRole.php*.

获取

描述

`integer/array role.get(object parameters)`

该方法允许您根据给定参数检索角色。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
roleids	ID/array	仅返回具有指定 ID 的角色。
current	boolean	如果设置为 true，则仅返回当前用户的角色。
selectRules	query	返回包含角色规则的rules 属性。
selectUsers	query	返回包含已分配该角色的用户的users 属性。 有关基于用户类型的限制，请参阅user.get。
sortfield	string/array	按指定属性对结果进行排序。 可能的值：roleid、name。
countOutput	boolean	这些参数在参考说明中进行了介绍。
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回下列两种之一：

-- 一组对象；- 在 countOutput 参数应用的情况下，返回检索对象的数量。

示例

获取角色数据

获取 “Super admin role” 角色的数据及其访问规则。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "role.get",
    "params": {
        "output": "extend",
        "selectRules": "extend",
        "roleids": "3"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "roleid": "3",
            "name": "Super admin role",
            "type": "3",
            "readonly": "1",
            "rules": {
                "ui": [
                    {
                        "name": "monitoring.dashboard",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.problems",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.hosts",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.latest_data",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.maps",
                        "status": "1"
                    },
                    {
                        "name": "services.services",
                        "status": "1"
                    },
                    {
                        "name": "services.sla_report",
                        "status": "1"
                    },
                    {
                        "name": "inventory.overview",
```

```
            "status": "1"
        },
        {
            "name": "inventory.hosts",
            "status": "1"
        },
        {
            "name": "reports.availability_report",
            "status": "1"
        },
        {
            "name": "reports.top_triggers",
            "status": "1"
        },
        {
            "name": "monitoring.discovery",
            "status": "1"
        },
        {
            "name": "services.sla",
            "status": "1"
        },
        {
            "name": "reports.scheduled_reports",
            "status": "1"
        },
        {
            "name": "reports.notifications",
            "status": "1"
        },
        {
            "name": "configuration.template_groups",
            "status": "1"
        },
        {
            "name": "configuration.host_groups",
            "status": "1"
        },
        {
            "name": "configuration.templates",
            "status": "1"
        },
        {
            "name": "configuration.hosts",
            "status": "1"
        },
        {
            "name": "configuration.maintenance",
            "status": "1"
        },
        {
            "name": "configuration.discovery",
            "status": "1"
        },
        {
            "name": "configuration.trigger_actions",
            "status": "1"
        },
        {
            "name": "configuration.service_actions",
            "status": "1"
        },
```

```
                {
                    "name": "configuration.discovery_actions",
                    "status": "1"
                },
                {
                    "name": "configuration.autoregistration_actions",
                    "status": "1"
                },
                {
                    "name": "configuration.internal_actions",
                    "status": "1"
                },
                {
                    "name": "reports.system_info",
                    "status": "1"
                },
                {
                    "name": "reports.audit",
                    "status": "1"
                },
                {
                    "name": "reports.action_log",
                    "status": "1"
                },
                {
                    "name": "configuration.event_correlation",
                    "status": "1"
                },
                {
                    "name": "administration.media_types",
                    "status": "1"
                },
                {
                    "name": "administration.scripts",
                    "status": "1"
                },
                {
                    "name": "administration.user_groups",
                    "status": "1"
                },
                {
                    "name": "administration.user_roles",
                    "status": "1"
                },
                {
                    "name": "administration.users",
                    "status": "1"
                },
                {
                    "name": "administration.api_tokens",
                    "status": "1"
                },
                {
                    "name": "administration.authentication",
                    "status": "1"
                },
                {
                    "name": "administration.general",
                    "status": "1"
                },
                {
                    "name": "administration.audit_log",
```

```
                "status": "1"
            },
            {
                "name": "administration.housekeeping",
                "status": "1"
            },
            {
                "name": "administration.proxy_groups",
                "status": "1"
            },
            {
                "name": "administration.proxies",
                "status": "1"
            },
            {
                "name": "administration.macros",
                "status": "1"
            },
            {
                "name": "administration.queue",
                "status": "1"
            }
        ],
        "ui.default_access": "1",
        "services.read.mode": "1",
        "services.read.list": [],
        "services.read.tag": {
            "tag": "",
            "value": ""
        },
        "services.write.mode": "1",
        "services.write.list": [],
        "services.write.tag": {
            "tag": "",
            "value": ""
        },
        "modules": [
            {
                "moduleid": 1,
                "status": "1"
            },
            {
                "moduleid": 2,
                "status": "1"
            },
            {
                "moduleid": 3,
                "status": "1"
            },
            {
                "moduleid": 4,
                "status": "1"
            },
            {
                "moduleid": 5,
                "status": "1"
            },
            {
                "moduleid": 6,
                "status": "1"
            },
            {
```

```
            "moduleid": 7,
            "status": "1"
        },
        {
            "moduleid": 8,
            "status": "1"
        },
        {
            "moduleid": 9,
            "status": "1"
        },
        {
            "moduleid": 10,
            "status": "1"
        },
        {
            "moduleid": 11,
            "status": "1"
        },
        {
            "moduleid": 12,
            "status": "1"
        },
        {
            "moduleid": 13,
            "status": "1"
        },
        {
            "moduleid": 14,
            "status": "1"
        },
        {
            "moduleid": 15,
            "status": "1"
        },
        {
            "moduleid": 16,
            "status": "1"
        },
        {
            "moduleid": 17,
            "status": "1"
        },
        {
            "moduleid": 18,
            "status": "1"
        },
        {
            "moduleid": 19,
            "status": "1"
        },
        {
            "moduleid": 20,
            "status": "1"
        },
        {
            "moduleid": 21,
            "status": "1"
        },
        {
            "moduleid": 22,
            "status": "1"
```

```
            },
            {
                "moduleid": 23,
                "status": "1"
            },
            {
                "moduleid": 24,
                "status": "1"
            },
            {
                "moduleid": 25,
                "status": "1"
            },
            {
                "moduleid": 26,
                "status": "1"
            },
            {
                "moduleid": 27,
                "status": "1"
            },
            {
                "moduleid": 28,
                "status": "1"
            },
            {
                "moduleid": 29,
                "status": "1"
            },
            {
                "moduleid": 30,
                "status": "1"
            },
            {
                "moduleid": 31,
                "status": "1"
            },
            {
                "moduleid": 32,
                "status": "1"
            },
            {
                "moduleid": 33,
                "status": "1"
            }
        ],
        "modules.default_access": "1",
        "api.access": "1",
        "api.mode": "0",
        "api": [],
        "actions": [
            {
                "name": "edit_dashboards",
                "status": "1"
            },
            {
                "name": "edit_maps",
                "status": "1"
            },
            {
                "name": "acknowledge_problems",
                "status": "1"
```

```
                    },
                    {
                        "name": "suppress_problems",
                        "status": "1"
                    },
                    {
                        "name": "close_problems",
                        "status": "1"
                    },
                    {
                        "name": "change_severity",
                        "status": "1"
                    },
                    {
                        "name": "add_problem_comments",
                        "status": "1"
                    },
                    {
                        "name": "execute_scripts",
                        "status": "1"
                    },
                    {
                        "name": "manage_api_tokens",
                        "status": "1"
                    },
                    {
                        "name": "edit_maintenance",
                        "status": "1"
                    },
                    {
                        "name": "manage_scheduled_reports",
                        "status": "1"
                    },
                    {
                        "name": "manage_sla",
                        "status": "1"
                    },
                    {
                        "name": "invoke_execute_now",
                        "status": "1"
                    },
                    {
                        "name": "change_problem_ranking",
                        "status": "1"
                    },
                    {
                        "name": "edit_own_media",
                        "status": "1"
                    },
                    {
                        "name": "edit_user_media",
                        "status": "1"
                    }
                ],
                "actions.default_access": "1"
            }
        }
    ],
    "id": 1
}
```


另请参阅

- 角色规则
- 用户

参考来源

CRole::get() in *ui/include/classes/api/services/CRole.php*.

触发器

此类用于处理触发器。

对象参考：

- Trigger
- Trigger tag

可用方法：

- trigger.create - 创建新的触发器
- trigger.delete - 删除触发器
- trigger.get - 检索触发器
- trigger.update - 更新触发器

触发器对象

以下对象与 `trigger`API 直接相关.

触发器

触发器对象具有以下属性。

Property	Type	Description
triggerid	ID	触发器的 ID。 属性行为： - 只读 - 更新操作时 必填
description	string	触发器名称。 属性行为： - 创建操作时 必填
expression	string	触发器表达式。 `value_type` 设置为 5（二进制）或 6（JSON）的监控项不能在触发器表达式中使用。 属性行为： - 创建操作时 必填
event_name	string	由触发器生成的事件名称。
opdata	string	运行数据。
comments	string	触发器的附加描述。
error	string	如果在更新触发器状态时出现任何问题，则为错误文本。 属性行为： - 只读
flags	integer	触发器的来源。 可能的值： 0 - （默认）普通触发器； 4 - 由原型转换而来的触发器。 属性行为： - 只读

Property	Type	Description
lastchange	timestamp	触发器上次更改其状态的时间。 属性行为： - 只读
priority	integer	触发器的严重性。 可能的值： 0 - （默认）未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 高； 5 - 灾难。
state	integer	触发器表达式的状态。 可能的值： 0 - （默认）触发器状态为最新； 1 - 当前触发器状态未知。 属性行为： - 只读
status	integer	触发器是启用还是禁用。 可能的值： 0 - （默认）启用； 1 - 禁用。
templateid	ID	父模板触发器的 ID。 属性行为： - 只读
type	integer	触发器是否可以生成多个问题事件。 可能的值： 0 - （默认）不生成多个事件； 1 - 生成多个事件。
url	string	与触发器关联的 URL。
url_name	string	与触发器关联的 URL 标签。
value	integer	触发器是否处于 OK 或问题状态。 可能的值： 0 - （默认） OK； 1 - 问题。 属性行为： - 只读
recovery_mode	integer	OK 事件生成模式。 可能的值： 0 - （默认）表达式； 1 - 恢复表达式； 2 - 无。
recovery_expression	string	触发器恢复表达式。 `value_type` 设置为 5（二进制）或 6（JSON）的监控项不能在触发器恢复表达式中使用。
correlation_mode	integer	OK 事件关闭方式。 可能的值： 0 - （默认）所有问题； 1 - 当标签值匹配时关闭所有问题。
correlation_tag	string	用于匹配的标签。

Property	Type	Description
manual_close	integer	允许手动关闭。 可能的值： 0 - （默认）否； 1 - 是。
uuid	string	通用唯一标识符，用于将导入的触发器链接到已存在的触发器。如果未提供，则自动生成。 属性行为： - 如果触发器属于模板，则 支持

触发器标签

触发器标签对象具有以下属性。

Property	Type	Description
tag	string	触发器标签名称。 属性行为： - 必填
value	string	触发器标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板； 1 - 主机； 3 - 监控项。 属性行为： - 只读。
objectid	ID	标签继承自的对象的 ID。 属性行为： - 只读。
automatic	integer	触发器标签的类型： 可能的值： 0 - *(默认)* 由用户创建的标签； 1 - 由 LLD 创建的标签；属性行为： - 只读。

创建

描述

`object trigger.create(object/array triggers)`

此方法允许您创建新的触发器。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的触发器。

除了标准触发器属性之外，该方法还接受以下参数。

参数	类型	说明
dependencies	array	该触发器所依赖的触发器。 这些触发器必须仅定义 triggerid 属性。
tags	array	触发器标签。

返回值

(object) 返回一个对象，该对象包含在 triggerids 属性下创建的触发器 ID，返回的 ID 顺序与传递的触发器的顺序相匹配.

示例

创建触发器

创建两个触发器，每个触发器依赖于另一个触发器。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": [
        {
            "description": "{HOST.NAME} 上的处理器负载过高",
            "expression": "last(/Linux server/system.cpu.load[percpu,avg1])>5",
            "dependencies": [
                {
                    "triggerid": "17367"
                }
            ]
        },
        {
            "description": " 服务已停止",
            "expression": "length(last(/Linux server/log[/var/log/system,Service .* has stopped]))<>0",
            "dependencies": [
                {
                    "triggerid": "17368"
                }
            ],
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 1
}
```

源码

CTrigger::create() in *ui/include/classes/api/services/CTrigger.php*.

删除

描述

`object trigger.delete(array triggerIds)`

此方法允许您删除触发器。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的触发器 ID 列表.

返回值

(object) 返回一个对象，该对象包含在 `triggerids` 属性下已删除触发器 ID.

示例

删除多个触发器

删除两个触发器。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "trigger.delete",
    "params": [
        "12002",
        "12003"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

源码

CTrigger::delete() in *ui/include/classes/api/services/CTrigger.php*.

更新

描述

`object trigger.update(object/array triggers)`

此方法允许您更新现有的触发器。

Note:
此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的触发器属性。

必须为每个触发器定义 triggerid 属性，其他所有属性均为可选。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准触发器属性之外，该方法还接受以下参数。

Parameter	Type	Description
dependencies	array	该触发器所依赖的触发器。 这些触发器必须仅定义 triggerid 属性。
tags	array	触发器标签。

返回值

(object) 返回一个对象，其中包含 triggerids 属性下已更新的触发器的 ID.

示例

启用触发器

启用一个触发器，即将其状态设置为 “0”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

替换触发器标签

替换触发器的标签。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "tags": [
            {
                "tag": "service",
                "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
            },
            {
```

```
                "tag": "error",
                "value": ""
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

替换依赖项

替换触发器的依赖项.

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "22713",
        "dependencies": [
            {
                "triggerid": "22712"
            },
            {
                "triggerid": "22772"
            }
        ]
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "22713"
        ]
    },
    "id": 1
}
```

源码

CTrigger::update() in *ui/include/classes/api/services/CTrigger.php*.

获取

描述

`integer/array trigger.get(object parameters)`

该方法允许您根据给定参数检索触发器。

Note:
此方法对任何类型的用户都可用。在用户角色设置中可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
triggerids	ID/array	仅返回具有给定 ID 的触发器。
groupids	ID/array	仅返回属于给定主机组或模板组中的主机或模板的触发器。
templateids	ID/array	仅返回属于给定模板的触发器。
hostids	ID/array	仅返回属于给定主机的触发器。
itemids	ID/array	仅返回包含给定监控项的触发器。
maintenanceids	ID/array	仅返回受给定维护影响的触发器。
functions	string/array	仅返回使用给定函数的触发器。 有关支持的函数列表，请参阅支持的函数页面。
group	string	仅返回属于具有给定名称的主机组或模板组中的主机或模板的触发器。
host	string	仅返回属于具有给定技术名称的主机的触发器。
inherited	boolean	如果设置为 true，则仅返回从模板继承的触发器。
templated	boolean	如果设置为 true，则仅返回属于模板的触发器。
dependent	boolean	如果设置为 true，则仅返回具有依赖关系的触发器。如果设置为 false，则仅返回不具有依赖关系的触发器。
monitored	flag	仅返回属于受监控主机且仅包含启用监控项的已启用触发器。
active	flag	仅返回属于受监控主机的已启用触发器。
maintenance	boolean	如果设置为 true，则仅返回属于处于维护状态的主机的已启用触发器。
withUnacknowledgedEvents	flag	仅返回具有未确认事件的触发器。
withAcknowledgedEvents	flag	仅返回所有事件均已确认的触发器。
withLastEventUnacknowledged	flag	仅返回最后一个事件未确认的触发器。
skipDependent	flag	跳过处于问题状态且依赖于其他触发器的触发器。请注意，如果其他触发器已禁用、具有已禁用的监控项或属于已禁用的监控项主机，则会忽略这些触发器。
lastChangeSince	timestamp	仅返回在给定时间之后更改过状态的触发器。
lastChangeTill	timestamp	仅返回在给定时间之前更改过状态的触发器。
only_true	flag	仅返回最近处于问题状态的触发器（在ok_period 内）。
min_severity	integer	仅返回严重性大于或等于给定严重性的触发器。
evaltype	integer	标签评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。
tags	array	仅返回具有给定标签的触发器。 格式：[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]。 空数组返回所有触发器。 可能的运算符值： 0 - （默认）包含； 1 - 等于； 2 - 不包含； 3 - 不等于； 4 - 存在； 5 - 不存在。
expandComment	flag	展开触发器描述中的宏。
expandDescription	flag	展开触发器名称中的宏。
expandExpression	flag	展开触发器表达式中的函数和宏。
selectHostGroups	query	在hostgroups 属性中返回触发器所属的主机组。
selectHosts	query	在hosts 属性中返回触发器所属的主机。
selectItems	query	在items 属性中返回触发器包含的监控项。

参数	类型	描述
selectFunctions	query	在 functions 属性中返回触发器使用的函数。 函数对象表示触发器表达式中使用的函数，并具有以下属性： functionid - (ID) 函数的 ID； itemid - (ID) 函数中使用的监控项的 ID； function - (string) 函数名称； parameter - (string) 传递给函数的参数。返回字符串中的查询参数会替换为 $ 符号。
selectDependencies	query	在 dependencies 属性中返回该触发器所依赖的触发器。
selectDiscoveryData	query	返回包含触发器发现对象数据的 discoveryData 属性。触发器发现对象将已发现的触发器链接到发现它的触发器原型。 它具有以下属性： parent_triggerid - (ID) 创建该触发器所依据的触发器原型的 ID； status - (int) 触发器发现状态： 0 - （默认）已发现触发器， 1 - 不再发现触发器； ts_delete - (timestamp) 不再发现的触发器将被删除的时间； ts_disable - (timestamp) 不再发现的触发器将被禁用的时间； disable_source - (int) 指示触发器是由 LLD 规则还是手动禁用： 0 - （默认）自动禁用， 1 - 由 LLD 规则禁用。
selectDiscoveryRule	query	在 discoveryRule 属性中返回创建该触发器的低级别发现规则。
selectLastEvent	query	在lastEvent 属性中返回最近的重要触发器事件。
selectTags	query	在tags 属性中返回触发器标签。
inheritedTags	boolean	返回在模板/主机/链接模板中也具有给定 tags 的触发器。 可能的值： true - 模板/主机/链接模板也必须具有给定标签； false - （默认）忽略模板/主机/链接模板中的标签。
selectInheritedTags	query	返回inheritedTags 属性，其中包含从模板、链接模板和主机继承的标签，以及从触发器或恢复表达式中引用的监控项继承的标签。
selectTemplateGroups	query	在templategroups 属性中返回触发器所属的模板组。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名称，值为要进行匹配的单个值或值数组。 不支持 text 数据类型的属性。 支持以下附加属性： host - 触发器所属主机的技术名称； hostid - 触发器所属主机的 ID。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectHosts - 结果将按 host 排序。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：triggerid、description、status、priority、lastchange、hostname。
countOutput	boolean	这些参数在参考说明中进行了描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

参数	类型	描述
selectTriggerDiscovery	query	在 triggerDiscovery 属性中返回触发器发现对象。触发器发现对象将触发器链接到创建该触发器所依据的触发器原型。 此查询已弃用，请改用 selectDiscoveryData。

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回已检索对象的数量。

示例

通过触发器 ID 检索数据

检索触发器 “14062” 中使用的所有数据和函数。

请求:

```
{
"jsonrpc": "2.0",
"method": "trigger.get",
"params": {
"triggerids": "14062",
"output": "extend",
"selectFunctions": "extend"
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": [
{
"triggerid": "14062",
"expression": "{13513><10m",
"description": "{HOST.NAME} 已重启 (正常运行时间 < 10m)",
"url": "",
"status": "0",
"value": "0",
"priority": "2",
"lastchange": "0",
"comments": " 主机正常运行时间小于 10 分钟”,
“错误”: “”,
“templateid”: “10016”,
“类型”: “0”,
“状态”: “0”,
“标志”: “0”,
“恢复模式”: “0”,
“恢复表达”: “”,
“关联模式”: “0”,
“关联标签”: “”,
“manual_close”: “0”,
“opdata”: “”,
“事件名称”: “”,
“uuid”: “”,
“url_name”: “”,
“函数”: [
{
“函数 id”: “13513”,
“itemid”: “24350”,
“triggerid”: “14062”,
```

```
“参数”: “$”,
“函数”: “last”
}
]
}
],
“id”: 1
}
```

检索问题状态的触发器

检索问题状态的所有触发器的 ID、名称和严重性，并按严重性降序排列。

请求:

```
{
"jsonrpc": "2.0",
"method": "trigger.get",
"params": {
"output": [
"triggerid",
"description",
"priority"
],
"filter": {
"value": 1
},
"sortfield": "priority",
"sortorder": "DESC"
},
"id": 1
}
```

响应:

```
{
"jsonrpc": "2.0",
"result": [
{
"triggerid": "13907",
"description": "Zabbix 自我监控进程 < 100% 繁忙",
"priority": "4"
},
{
"triggerid": "13824",
"description": "Zabbix discoverer 进程超过 75% 繁忙”,
“优先级”:“3”
}
],
“id”: 1
}
```

检索带有标签的特定触发器

检索带有标签的特定触发器。

请求 :

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
```

```
        "triggerids": [
            "17578"
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17370",
            "description": "Service status",
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}",
                    "automatic": "1"
                },
                {
                    "tag": "error",
                    "value": "",
                    "automatic": "1"
                }
            ]
        }
    ],
    "id": 1
}
```

另请参阅

- 发现规则
- 监控项
- 主机
- 主机组
- 模板组

源码

CTrigger::get() in *ui/include/classes/api/services/CTrigger.php*.

触发器原型

此类用于处理触发器原型。

对象参考：

- 触发器原型
- 触发器原型标签

可用方法：

- triggerprototype.create - 创建新的触发器原型
- triggerprototype.delete - 删除触发器原型
- triggerprototype.get - 检索触发器原型
- triggerprototype.update - 更新触发器原型

触发器原型对象

以下对象与 `triggerprototype` API 直接相关。

触发器原型

触发器原型对象具有以下属性。

属性	类型	描述
triggerid	ID	触发器原型的 ID。 属性行为： - 只读 - 更新操作时为 必需
description	string	触发器原型的名称。 属性行为： - 创建操作时为 必需
expression	string	触发器表达式。 必须至少包含一个监控项原型。 不能在触发器表达式中使用 `value_type` 设置为 5（二进制）或 6（JSON）的监控项原型。 属性行为： - 创建操作时为 必需
event_name	string	由触发器生成的事件名称。
opdata	string	运行数据。
comments	string	触发器原型的附加注释。
priority	integer	触发器原型的严重性。 可能的值： 0 - *(默认)* 未分类； 1 - 信息； 2 - 警告； 3 - 一般严重； 4 - 高； 5 - 灾难。
status	integer	触发器原型是启用还是禁用。 可能的值： 0 - *(默认)* 启用； 1 - 禁用。
flags	integer	触发器原型的来源。 可能的值： 2 - 触发器原型； 6 - 发现的触发器原型。 属性行为： - 只读
templateid	ID	父模板触发器原型的 ID。 属性行为： - 只读
type	integer	触发器原型是否可以生成多个问题事件。 可能的值： 0 - *(默认)* 不生成多个事件； 1 - 生成多个事件。
url	string	与触发器原型关联的 URL。
url_name	string	与触发器原型关联的 URL 标签。
recovery_mode	integer	OK 事件生成模式。 可能的值： 0 - *(默认)* 表达式； 1 - 恢复表达式； 2 - 无。

属性	类型	描述
recovery_expression	string	触发器恢复表达式。 必须至少包含一个监控项原型。 不能在触发器恢复表达式中使用 `value_type` 设置为 5（二进制）或 6（JSON）的监控项原型。
correlation_mode	integer	OK 事件关闭方式。 可能的值： 0 - *(默认)* 所有问题； 1 - 当标签值匹配时关闭所有问题。
correlation_tag	string	用于匹配的标签。
manual_close	integer	允许手动关闭。 可能的值： 0 - *(默认)* 否； 1 - 是。
discover	integer	触发器原型发现状态。 可能的值： 0 - *(默认)* 将发现新的触发器； 1 - 将不会发现新的触发器，现有触发器将被标记为丢失。
uuid	string	通用唯一标识符，用于将导入的触发器原型链接到已存在的触发器原型。如果未提供，将自动生成。 属性行为： - 如果触发器原型属于模板，则 受支持

触发器原型标签

触发器原型标签对象具有以下属性。

属性	类型	描述
tag	string	触发器原型标签名称。 属性行为： - 必填
value	string	触发器原型标签值。 属性行为： - 只读。
object	integer	标签继承自的对象类型。 可能的值： 0 - 模板； 1 - 主机； 3 - 监控项； 4 - 监控项原型。 属性行为： - 只读。
objectid	ID	标签继承自的对象的 ID。 属性行为： - 只读。

创建

描述

`object triggerprototype.create(object/array triggerPrototypes)`

此方法允许您创建新的触发器原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要创建的触发器原型。

除了标准触发器原型属性之外，该方法还接受以下参数。

参数	Type	描述
dependencies	array	该触发器原型所依赖的触发器和触发器原型。 这些触发器必须仅定义 triggerid 属性。
tags	array	触发器原型标签。

返回值

(object) 返回一个对象，该对象包含在 triggerids 属性中已创建触发器原型的 ID，返回 ID 的顺序与传递触发器原型的顺序相匹配.

示例

创建触发器原型

创建一个触发器原型，用于检测文件系统的可用磁盘空间是否低于 20%。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.create",
    "params": {
        "description": " 卷 {#FSNAME} 的可用磁盘空间低于 20%",
        "expression": "last(/Zabbix server/vfs.fs.size[{#FSNAME},pfree])<20",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17372"
        ]
    },
    "id": 1
}
```

源码

CTriggerPrototype::create() in *ui/include/classes/api/services/CTriggerPrototype.php*.

删除

描述

`object triggerprototype.delete(array triggerPrototypeIds)`

此方法允许您删除触发器原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的触发器原型的 ID.

返回值

(object) 返回一个对象，该对象包含在 triggerids 属性中已删除触发器原型的 ID.

示例

删除多个触发器原型

删除两个触发器原型。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.delete",
    "params": [
        "12002",
        "12003"
    ],
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

源码

CTriggerPrototype::delete() in *ui/include/classes/api/services/CTriggerPrototype.php*.

更新

描述

`object triggerprototype.update(object/array triggerPrototypes)`

此方法允许您更新现有的触发器原型。

> **Note:**
> 此方法仅适用于 *Admin* 和 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object/array) 要更新的触发器原型属性。

必须为每个触发器原型定义 triggerid 属性，所有其他属性都是可选的。只有传入的属性会被更新，其他所有属性将保持不变。

除了标准触发器原型属性之外，该方法还接受以下参数。

参数	类型	描述
dependencies	array	触发器原型所依赖的触发器和触发器原型。 这些触发器必须只定义 triggerid 属性。
tags	array	触发器原型标签。

返回值

(object) 返回一个对象，该对象是包含在 triggerids 属性中已更新触发器原型的 ID.

示例

启用触发器原型

启用触发器原型，即将其状态设置为 “0”。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

替换触发器原型标签

替换一个触发器原型的标签。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "17373",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17373"
        ]
    },
    "id": 1
}
```

源码

CTriggerPrototype::update() in *ui/include/classes/api/services/CTriggerPrototype.php*.

获取

描述

`integer/array triggerprototype.get(object parameters)`

该方法允许您根据给定参数检索触发器原型。

Note:
此方法对任何类型的用户都可用。在用户角色设置中，可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
active	flag	仅返回属于受监控主机的已启用触发器原型。
discoveryids	ID/array	仅返回属于给定 LLD 规则的触发器原型。
functions	string/array	仅返回使用给定函数的触发器。 支持的函数列表请参见支持的函数页面。
group	string	仅返回属于名称与给定名称匹配的主机组或模板组中的主机或模板的触发器原型。
groupids	ID/array	仅返回属于给定主机组或模板组中的主机或模板的触发器原型。
host	string	仅返回属于名称与给定名称匹配的主机的触发器原型。
hostids	ID/array	仅返回属于给定主机的触发器原型。
inherited	boolean	如果设置为 `true`，则仅返回从模板继承的触发器原型。
maintenance	boolean	如果设置为 `true`，则仅返回属于维护中主机的已启用触发器原型。
min_severity	integer	仅返回严重性大于或等于给定严重性的触发器原型。
monitored	flag	仅返回属于受监控主机且仅包含已启用监控项的已启用触发器原型。
templated	boolean	如果设置为 `true`，则仅返回属于模板的触发器原型。
templateids	ID/array	仅返回属于给定模板的触发器原型。
triggerids	ID/array	仅返回具有给定 ID 的触发器原型。
expandExpression	flag	展开触发器表达式中的函数和宏。
selectDependencies	query	在 `dependencies` 属性中返回该触发器原型所依赖的触发器原型和触发器。

参数	类型	说明
selectDiscoveryData	query	返回一个 discoveryData 属性，其中包含触发器原型发现对象数据。触发器原型发现对象将已发现的触发器原型链接到发现它的触发器原型。 它具有以下属性： parent_triggerid - (ID) 创建该触发器原型所基于的触发器原型 ID； status - (int) 触发器原型发现状态： 0 - *(默认)* 触发器原型已发现， 1 - 触发器原型不再被发现； ts_delete - (timestamp) 不再被发现的触发器原型将被删除的时间； ts_disable - (timestamp) 不再被发现的触发器原型将被禁用的时间； disable_source - (int) 指示触发器原型是由 LLD 规则还是手动禁用： 0 - *(默认)* 自动禁用， 1 - 由 LLD 规则禁用。
selectDiscoveryRule	query	在 discoveryRule 属性中返回该触发器原型所属的LLD 规则。
selectDiscoveryRulePrototype	query	返回一个discoveryRulePrototype 属性，其中包含该触发器原型所属的父 LLD 规则原型。
selectFunctions	query	在 functions 属性中返回触发器原型中使用的函数。 函数对象表示触发器表达式中使用的函数，并具有以下属性： functionid - (ID) 函数的 ID； itemid - (ID) 函数中使用的监控项 ID； function - (string) 函数名称； parameter - (string) 传递给函数的参数。返回的字符串中，查询参数会被替换为 $ 符号。
selectHostGroups	query	在hostgroups 属性中返回该触发器原型所属的主机组。
selectHosts	query	在hosts 属性中返回该触发器原型所属的主机。
selectInheritedTags	query	返回一个inheritedTags 属性，其中包含从模板、已链接模板和主机继承的标签，以及从触发器表达式或恢复表达式中引用的监控项原型继承的标签。
selectItems	query	在items 属性中返回该触发器原型使用的监控项和监控项原型。
selectTags	query	在tags 属性中返回触发器原型标签。
selectTemplateGroups	query	在templategroups 属性中返回该触发器原型所属的模板组。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值为单个值或用于匹配的值数组。 不支持 text 数据类型 的属性。 支持以下附加属性： host - 该触发器原型所属主机的技术名称； hostid - 该触发器原型所属主机的 ID。
limitSelects	integer	限制子选择返回的记录数。 适用于以下子选择： selectHosts - 结果将按 host 排序。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：triggerid、description、status、priority、discovered。
countOutput	boolean	这些参数在参考说明中描述。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

从 LLD 规则中检索触发器原型

检索某个 LLD 规则中的所有触发器原型及其函数。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": "extend",
        "selectFunctions": "extend",
        "discoveryids": "22450"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13272",
            "expression": "{12598}<20",
            "description": " 卷 {#FSNAME} 上的空闲 inode 少于 20%",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "",
            "templateid": "0",
            "type": "0",
            "state": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "discover": "0",
            "event_name": "",
            "uuid": "6ce467d05e8745409a177799bed34bb3",
            "url_name": "",
            "functions": [
                {
                    "functionid": "12598",
                    "itemid": "22454",
                    "triggerid": "13272",
                    "parameter": "$",
                    "function": "last"
                }
            ]
        },
        {
```

```
            "triggerid": "13266",
            "expression": "{13500}<20",
            "description": " 卷 {#FSNAME} 上的可用磁盘空间少于 20%",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "",
            "templateid": "0",
            "type": "0",
            "state": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "discover": "0",
            "event_name": "",
            "uuid": "74a1fc62bfe24b7eabe4e244c70dc384",
            "url_name": "",
            "functions": [
                {
                    "functionid": "13500",
                    "itemid": "22686",
                    "triggerid": "13266",
                    "parameter": "$",
                    "function": "last"
                }
            ]
        }
    ],
    "id": 1
}
```

检索带有标签的特定触发器原型

请求：

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17373"
        ]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17373",
```

```
            "description": " 卷 {#FSNAME} 上的可用磁盘空间低于 20%",
            "tags": [
                {
                    "tag": "volume",
                    "value": "{#FSNAME}"
                },
                {
                    "tag": "type",
                    "value": "{#FSTYPE}"
                }
            ]
        }
    ],
    "id": 1
}
```

另请参见

- 发现规则
- 监控项
- 主机
- 主机组
- 模板组

来源

ui/include/classes/api/services/CTriggerPrototype.php 中的 CTriggerPrototype::get()。

认证

这个类被设计用于处理身份验证设置。

对象引用：

- 认证

可用的方法：

- authentication.get - 检索身份认证设置
- authentication.update - 更新身份认证设置

认证对象

以下对象与 `authentication` API 直接相关。

身份验证

身份验证对象具有以下属性。

属性	类型	描述
authentication_type	integer	默认身份验证方式。 可能的值： 0 - （默认）内部； 1 - LDAP。
http_auth_enabled	integer	HTTP 身份验证。 可能的值： 0 - （默认）已禁用； 1 - 已启用。 属性行为： - 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['http_auth_enabled']` 设置为 `true`，则支持。

属性	类型	描述
http_login_form	integer	默认登录表单。 可能的值： 0 - （默认） Zabbix 登录表单； 1 - HTTP 登录表单。 属性行为： - 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['http_auth_enabled']` 设置为 `true`，则支持。
http_strip_domains	string	要移除的域名。 属性行为： - 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['http_auth_enabled']` 设置为 `true`，则支持。
http_case_sensitive	integer	HTTP 区分大小写的登录。 可能的值： 0 - 关闭； 1 - （默认）开启。 属性行为： - 如果在前端配置文件中将 `$ZBX_FEATURE_FLAGS['http_auth_enabled']` 设置为 `true`，则支持。
ldap_auth_enabled	integer	LDAP 身份验证。 可能的值： 0 - （默认）已禁用； 1 - 已启用。
ldap_case_sensitive	integer	LDAP 区分大小写的登录。 可能的值： 0 - 关闭； 1 - （默认）开启。
ldap_userdirectoryid	ID	LDAP 身份验证的默认用户目录 ID。 用于将 `gui_access` 设置为 LDAP 或系统默认的用户组。 属性行为： - 如果将 `ldap_auth_enabled` 设置为 “已启用”，则必需
saml_auth_enabled	integer	SAML 身份验证。 可能的值： 0 - （默认）已禁用； 1 - 已启用。
saml_case_sensitive	integer	SAML 区分大小写的登录。 可能的值： 0 - 关闭； 1 - （默认）开启。
passwd_min_length	integer	密码最小长度要求。 可能的值范围为 1 至 70。 默认值：8。

属性	类型	描述
passwd_check_rules	integer	密码检查规则。 可能的位图值： 0 - 检查密码长度； 1 - 检查密码是否使用大小写拉丁字母； 2 - 检查密码是否使用数字； 4 - 检查密码是否使用特殊字符； 8 - （默认）检查密码是否不在常用密码列表中，并且不包含单词 “Zabbix” 或用户的名字、姓氏或用户名的变体。 这是一个位掩码字段；可以使用任意可能位图值的总和（例如，使用 15 检查所有规则）。
ldap_jit_status	integer	LDAP 配置的状态。 可能的值： 0 - 对已配置的 LDAP IdP 禁用； 1 - 对已配置的 LDAP IdP 启用。
saml_jit_status	integer	SAML 配置的状态。 可能的值： 0 - 对已配置的 SAML IdP 禁用； 1 - 对已配置的 SAML IdP 启用。
jit_provision_interval	string	已登录用户的 JIT 配置请求之间的时间间隔。 接受秒数（例如 3600），或带有时间后缀的值（例如 3600s、60m、1h、1d、1M、1y），包括对月和年的支持。最小值：1h。 默认值：1h。
disabled_usrgrpid	ID	仅适用于 LDAP 配置。 要将取消配置的用户分配到的用户组 ID。 该用户组必须处于禁用状态，并且在配置后无法启用或删除。
mfa_status	integer	属性行为： - 如果将 `ldap_jit_status` 设置为 “对已配置的 LDAP IdP 启用”，或将 `saml_jit_status` 设置为 “对已配置的 SAML IdP 启用”，则必需 多重身份验证。 可能的值： 0 - 已禁用（对所有已配置的 MFA 方法）； 1 - 已启用（对所有已配置的 MFA 方法）。
mfaid	ID	为启用 MFA 的用户组设置的默认MFA 方法。 属性行为： - 如果将 `mfa_status` 设置为 “已启用”，则必需

更新

描述

`object authentication.update(object authentication)`

此方法允许您更新现有的身份验证设置。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

`(object)` 要更新的认证属性。

返回值

(array) 返回一个包含已更新参数名称的数组。

示例

请求:

```
{
    "jsonrpc": "2.0",
    "method": "authentication.update",
    "params": {
        "http_auth_enabled": 1,
        "http_case_sensitive": 0,
        "http_login_form": 1
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        "http_auth_enabled",
        "http_case_sensitive",
        "http_login_form"
    ],
    "id": 1
}
```

源码位置

CAuthentication::update() 在 *ui/include/classes/api/services/CAuthentication.php* 文件中。

获取

描述

`object authentication.get(object parameters)`

该方法允许你根据给定参数检索认证对象。

> **Note:**
> 此方法仅对 *Super admin* 用户类型可用。在用户角色设置中可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

(object) 用于定义所需输出的参数。

该方法仅支持一个参数。

参数	Type	描述
output	query	此参数在参考注释中进行了说明。

返回值

(object) 返回认证对象。

示例

请求:

```
{
    "jsonrpc": "2.0",
    "method": "authentication.get",
    "params": {
        "output": "extend"
    },
```

```
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "authentication_type": "0",
        "http_auth_enabled": "0",
        "http_login_form": "0",
        "http_strip_domains": "",
        "http_case_sensitive": "1",
        "ldap_auth_enabled": "0",
        "ldap_case_sensitive": "1",
        "ldap_userdirectoryid": "0",
        "saml_auth_enabled": "0",
        "saml_case_sensitive": "0",
        "passwd_min_length": "8",
        "passwd_check_rules": "15",
        "jit_provision_interval": "1h",
        "saml_jit_status": "0",
        "ldap_jit_status": "0",
        "disabled_usrgrpid": "9",
        "mfa_status": "0",
        "mfaid": "0"
    },
    "id": 1
}
```

源码位置

CAuthentication::get() 在 *ui/include/classes/api/services/CAuthentication.php* 文件中。

设备

此类用于与移动设备配合使用，以支持Zabbix 移动应用。

对象引用：

- 设备

可用方法：

- device.get - 获取设备
- device.offboard - 移除（退出管理）设备

设备对象

设备对象表示用于Zabbix 移动应用 的移动设备。

Property	Type	Description
deviceid	ID	设备 ID（内部 ID，例如 1、2）。 Property behavior： - 只读
userid	ID	设备用户的 ID。 Property behavior： - 常量
uuid	string	设备的全局唯一标识符（UUIDv7）（外部标识符，例如 `019f930e-0ea2-7c1f-9dcc-0fbb60c40664`）。 Property behavior： - 只读

Property	Type	Description
name	string	设备名称。 Property behavior： - 只读
status	integer	设备状态。 可能的值： 0 - 新建 1 - 活动 2 - 未分配 Property behavior： - 只读
activated_at	timestamp	设备在移动应用中添加的日期和时间；如果尚未添加，则为 0。 Property behavior： - 只读
lastaccess	timestamp	设备上次在移动应用中完成身份验证的日期和时间；如果尚未完成身份验证，则为 0。 Property behavior： - 只读

device.get

此方法允许您检索用于Zabbix 移动应用 的移动设备。

如果启用了`EnableMobileDevices`，并且用户具有设备访问权限，则 *User*、*Admin* 和 *Super admin* 用户可以使用此方法。可以在用户角色配置中撤销对 API 方法的访问权限。

User 和 *Admin* 用户只能检索自己的设备。

方法签名：`integer/array device.get(object parameters)`

方法源代码：`ui/include/classes/api/services/CDevice.php` 中的 `CDevice::get()`

参数

Parameter	Type	Description
deviceids	ID/array	仅返回具有给定设备 ID 的设备。
userids	ID/array	仅返回属于具有给定用户 ID 的用户的设备。
roleids	ID/array	仅返回属于具有给定用户角色 ID 的用户的设备。
usrgrpids	ID/array	仅返回属于具有给定用户组 ID 的用户的设备。
sortfield	string/array	按给定属性对结果进行排序。 可能的值：`deviceid`、`name`。
countOutput	boolean	这些参数在参考说明中进行了描述。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

返回对象数组，或返回所检索对象的数量（如果使用了 `countOutput`）。

示例

获取设备

获取属于用户 ID 1、2、3 的用户设备。
 请注意，用户 2（访客）没有设备，并且设备 3 尚未在 Zabbix 移动应用中添加或完成身份验证。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "device.get",
    "params": {
        "output": "extend",
        "userids": [1, 2, 3]
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "deviceid": "1",
            "userid": "1",
            "uuid": "019f930e-0ea2-7c1f-9dcc-0fbb60c40664",
            "name": "Google Pixel 10",
            "status": "1",
            "activated_at": "1784533122",
            "lastaccess": "1784879933"
        },
        {
            "deviceid": "2",
            "userid": "1",
            "uuid": "019d00fb-9e87-7ad5-bb4e-978467254aff",
            "name": "Apple iPhone 15 Pro",
            "status": "1",
            "activated_at": "1784533225",
            "lastaccess": "0"
        },
        {
            "deviceid": "3",
            "userid": "3",
            "uuid": "019d00fb-9e87-7ad5-bb4e-6ed88a2297ee",
            "name": "",
            "status": "0",
            "activated_at": "0",
            "lastaccess": "0"
        }
    ],
    "id": 1
}
```

device.offboard

此方法允许移除（取消注册）用于Zabbix 移动应用 的移动设备。

如果启用了EnableMobileDevices，且用户拥有设备访问权限，则 *User*、*Admin* 和 *Super admin* 用户可以使用此方法。可以在用户角色配置中撤销对 API 方法的访问权限。

User 和 *Admin* 用户只能移除自己的设备。

方法签名：`object device.offboard(object parameters)`

方法源代码：`ui/include/classes/api/services/CDevice.php` 中的 `CDevice::offboard()`

参数

Parameter	Type	Description
uuid	string	设备通用唯一标识符（UUIDv7）。与 deviceid 匹配。
force	boolean	设置为 true 可忽略 “未找到设备” 错误并强制移除设备。 默认值：false

返回值

返回一个包含已移除设备 UUID 的对象。

示例

移除设备

移除具有指定 UUID 的设备，并忽略 “未找到设备” 错误。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "device.offboard",
    "params": {
        "uuid": "019d0106-5eef-7154-a159-bb29e35d38e5",
        "force": true
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "uuid": "019d0106-5eef-7154-a159-bb29e35d38e5"
    },
    "id": 1
}
```

设置
此类为进行常用管理设置而设计。参考对象: - Settings设置

可用办法 - settings.get - retrieve settings- 检索设置 - settings.update - update settings- 更新设置

设置对象

以下对象都是与 settings 直接相关的 API。

设置

设置对象具有以下属性。

Property	Type	Description
default_lang	string	默认系统语言。 默认值：en_US。
default_timezone	string	默认系统时区。 默认值：system - 系统默认值。 支持的时区完整列表请参阅 PHP 文档。

Property	Type	Description
default_theme	string	默认主题。 可能的值： `blue-theme` - （默认）蓝色； `blue-classic-theme` - 蓝色（经典）； `dark-theme` - 深色； `dark-blue-theme` - 深蓝色； `dark-classic-theme` - 深色（经典）； `hc-light` - 高对比度浅色； `hc-dark` - 高对比度深色。
search_limit	integer	搜索和筛选结果的限制。 默认值：1000。
max_overview_table_size	integer	Data overview 和 Trigger overview 仪表板小组件中显示的最大列数和行数。 默认值：50。
max_in_table	integer	表格单元格中显示的元素最大数量。 默认值：50。
server_check_interval	integer	Zabbix 服务器停止运行时显示警告。 可能的值： 0 - 不显示警告； 10 - （默认）显示警告。
work_period	string	工作时间。 默认值：1-5,09:00-18:00。
show_technical_errors	integer	向非超级管理员用户，以及不属于启用调试模式的用户组的用户显示技术错误（PHP/SQL）。 可能的值： 0 - （默认）不显示技术错误； 1 - 显示技术错误。
history_period	string	Latest data、Web 和 Data overview 仪表板小组件中显示历史数据的最大时间段。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d）。 默认值：24h。
period_default	string	时间筛选器的默认时间段。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d、1M、1y），包括对月份和年份的支持。 默认值：1h。
max_period	string	时间筛选器的最大时间段。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h、1d、1M、1y），包括对月份和年份的支持。 默认值：2y。
severity_color_0	string	“未分类” 严重性级别的颜色，以十六进制颜色代码表示。 默认值：97AAB3。
severity_color_1	string	“信息” 严重性级别的颜色，以十六进制颜色代码表示。 默认值：7499FF。
severity_color_2	string	“警告” 严重性级别的颜色，以十六进制颜色代码表示。 默认值：FFC859。
severity_color_3	string	“一般” 严重性级别的颜色，以十六进制颜色代码表示。 默认值：FFA059。

Property	Type	Description
severity_color_4	string	“高” 严重性级别的颜色，以十六进制颜色代码表示。 默认值：E97659。
severity_color_5	string	“灾难” 严重性级别的颜色，以十六进制颜色代码表示。 默认值：E45959。
severity_name_0	string	“未分类” 严重性级别的名称。 默认值：Not classified。
severity_name_1	string	“信息” 严重性级别的名称。 默认值：Information。
severity_name_2	string	“警告” 严重性级别的名称。 默认值：Warning。
severity_name_3	string	“一般” 严重性级别的名称。 默认值：Average。
severity_name_4	string	“高” 严重性级别的名称。 默认值：High。
severity_name_5	string	“灾难” 严重性级别的名称。 默认值：Disaster。
custom_color	integer	使用自定义事件状态颜色。 可能的值： 0 - （默认）不使用自定义事件状态颜色； 1 - 使用自定义事件状态颜色。
ok_period	string	显示 OK 触发器的时间段。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h）。 默认值：5m。
blink_period	string	触发器状态发生变化时的闪烁时间段。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m、2h）。 默认值：2m。
problem_unack_color	string	未确认 PROBLEM 事件的颜色，以十六进制颜色代码表示。 默认值：CC0000。
problem_ack_color	string	已确认 PROBLEM 事件的颜色，以十六进制颜色代码表示。 默认值：CC0000。
ok_unack_color	string	未确认 RESOLVED 事件的颜色，以十六进制颜色代码表示。 默认值：009900。
ok_ack_color	string	已确认 RESOLVED 事件的颜色，以十六进制颜色代码表示。 默认值：009900。
problem_unack_style	integer	未确认 PROBLEM 事件的闪烁。 可能的值： 0 - 不显示闪烁； 1 - （默认）显示闪烁。
problem_ack_style	integer	已确认 PROBLEM 事件的闪烁。 可能的值： 0 - 不显示闪烁； 1 - （默认）显示闪烁。

Property	Type	Description
ok_unack_style	integer	未确认 RESOLVED 事件的闪烁。 可能的值： 0 - 不显示闪烁； 1 - （默认）显示闪烁。
ok_ack_style	integer	已确认 RESOLVED 事件的闪烁。 可能的值： 0 - 不显示闪烁； 1 - （默认）显示闪烁。
url	string	前端 URL。
discovery_groupid	ID	自动将发现的主机放置到的主机组 ID。
default_inventory_mode	integer	默认主机资产清单模式。 可能的值： -1 - （默认）禁用； 0 - 手动； 1 - 自动。
alert_usrgrpid	ID	将数据库停止运行警报消息发送到的用户组 ID。 如果设置为 “0”，则不会发送警报消息。
snmptrap_logging	integer	记录不匹配的 SNMP trap。 可能的值： 0 - 不记录不匹配的 SNMP trap； 1 - （默认）记录不匹配的 SNMP trap。
login_attempts	integer	登录表单被阻止前允许的失败登录尝试次数。 默认值：5。
login_block	string	如果失败登录尝试次数超过 login_attempts 字段中定义的值，登录表单被阻止的时间间隔。 接受秒数（例如 1、60），或带有时间后缀的值（例如 30s、1m）。 默认值：30s。
validate_uri_schemes	integer	验证 URI 方案。 可能的值： 0 - 不验证； 1 - （默认）验证。
uri_valid_schemes	string	有效的 URI 方案。 默认值：http,https,ftp,file,mailto,tel,ssh。
x_frame_options	string	X-Frame-Options HTTP 标头。 默认值：SAMEORIGIN。
iframe_sandboxing_enabled	integer	使用 iframe 沙箱。 可能的值： 0 - 不使用； 1 - （默认）使用。
iframe_sandboxing_exceptions	string	iframe 沙箱例外。
connect_timeout	string	与 Zabbix 服务器建立连接的超时时间。 可能的值范围：1-30s。 默认值：3s。 属性行为： *- required*

Property	Type	Description
socket_timeout	string	网络默认超时时间。 可能的值范围：1-300s。 默认值：3s。 属性行为： - *required*
media_type_test_timeout	string	媒介类型测试的网络超时时间。 可能的值范围：1-300s。 默认值：65s。 属性行为： - *required*
item_test_timeout	string	监控项测试的网络超时时间。 可能的值范围：1-600s。 默认值：60s。 属性行为： - *required*
script_timeout	string	脚本执行的网络超时时间。 可能的值范围：1-300s。 默认值：60s。 属性行为： - *required*
report_test_timeout	string	计划报表测试的网络超时时间。 可能的值范围：1-300s。 默认值：60s。 属性行为： - *required*
device_link_timeout	string	添加移动设备以便与Zabbix 移动应用配合使用时的超时时间；适用于 Zabbix API 与桥接适配器之间的通信。 可能的值范围：1-300s。 默认值：60s。 属性行为： - *required*
auditlog_enabled	integer	是否启用审计日志记录。 可能的值： 0 - 禁用； 1 - （默认）启用。
auditlog_mode	integer	是否启用服务器（System 用户）执行的低级别发现、网络发现和自动注册活动的审计日志记录。 可能的值： 0 - 禁用； 1 - （默认）启用。

Property	Type	Description
ha_failover_delay	string	故障转移延迟时间（秒）。 默认值：1m。 属性行为： - *read-only*
geomaps_tile_provider	string	地理地图图块提供程序。 可能的值： `OpenStreetMap.Mapnik` - （默认） OpenStreetMap Mapnik； `OpenTopoMap` - OpenTopoMap； `Stamen.TonerLite` - Stamen Toner Lite； `Stamen.Terrain` - Stamen Terrain； `USGS.USTopo` - USGS US Topo； `USGS.USImagery` - USGS US Imagery。 支持使用空字符串指定 `geomaps_tile_url`、`geomaps_max_zoom` 和 `geomaps_attribution` 的自定义值。
geomaps_tile_url	string	地理地图图块 URL。 属性行为： - 当 `geomaps_tile_provider` 设置为空字符串时 *supported*
geomaps_max_zoom	integer	地理地图最大缩放级别。 可能的值：0-30。 属性行为： - 当 `geomaps_tile_provider` 设置为空字符串时 *supported*
geomaps_attribution	string	地理地图归属文本。 属性行为： - 当 `geomaps_tile_provider` 设置为空字符串时 *supported*
vault_provider	integer	Vault 提供程序。 可能的值： 0 - （默认） HashiCorp Vault； 1 - CyberArk Vault。
proxy_secrets_provider	integer	通过以下方式解析 secret 宏值： 可能的值： 0 - （默认）仅服务器； 1 - 服务器和 proxy 独立解析。
timeout_zabbix_agent	string	处理时间不超过 `timeout_*` 秒。 接受秒数（例如 1、60、300），或带有时间后缀的值（例如 30s、1m）。也接受用户宏。 可能的值范围：1-600s。 默认值：3s。 `timeout_browser` 的默认值：60s。 属性行为： - *required*
timeout_simple_check timeout_snmp_agent timeout_external_check timeout_db_monitor timeout_http_agent timeout_ssh_agent timeout_telnet_agent timeout_script		

Property	Type	Description
timeout_browser		
timeout_telemetry_query		
apm_global_db	object	全局数据源配置。

全局数据源

全局数据源对象具有以下属性：

Property	Type	Description
status	integer	全局数据源状态。 可能的值： 0 - （默认）未配置； 1 - 已配置。
url	string	用于连接数据源的 URL。 属性行为： - 如果用户将 `status` 更改为 1（已配置），则必需。
authentication_type	integer	身份验证类型。 可能的值： 0 - （默认）使用用户名和密码； 1 - 不使用身份验证。 属性行为： - 如果用户将 `status` 更改为 1（已配置），则必需。
username	string	用于登录数据源的用户名。 属性行为： - 当 `status` 设置为 1（已配置）且 `authentication_type` 设置为 0（用户名和密码）时支持。 - 如果用户将 `authentication_type` 更改为 0（用户名和密码），则必需。
password	string	用于登录数据源的密码。 此参数的值可以为空。 属性行为： - 当 `status` 设置为 1（已配置）且 `authentication_type` 设置为 0（用户名和密码）时支持。
db	string	数据库名称。 此参数的值可以为空。 属性行为： - 当 `status` 设置为 1（已配置）时支持。
ssl_verify_peer	integer	验证服务器证书。 可能的值： 0 - （默认）禁用； 1 - 启用。 属性行为： - 当 `status` 设置为 1（已配置）且 `url` 使用 `https://` 方案时支持。

Property	Type	Description
ssl_verify_host	integer	验证主机。 可能的值： 0 - 禁用； 1 - （默认）启用。 属性行为： - 当 `status` 设置为 1（已配置）且 `url` 使用 `https://` 方案时支持。

更新

描述

`object settings.update(object settings)`

此方法允许您更新现有的通用设置。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

（对象）设置待更新的属性。

返回值

（数组）返回被更新参数名字的数组。

示例

请求:

```
{
    "jsonrpc": "2.0",
    "method": "settings.update",
    "params": {
        "login_attempts": "1",
        "login_block": "1m"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        "login_attempts",
        "login_block"
    ],
    "id": 1
}
```

来源

CSettings::update() in *ui/include/classes/api/services/CSettings.php*.

获取

描述

`object settings.get(object parameters)`

该方法允许你根据给定参数检索 settings 对象。

Note:
此方法对任何类型的用户都可用。调用该方法的权限可以在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(object) 定义所需输出的参数。

该方法仅支持一个参数。

参数	Type	说明
output	query	该参数在参考说明中进行了描述。

返回值

(对象) 返回设置对象。

示例

请求:

```
{
    "jsonrpc": "2.0",
    "method": "settings.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": {
        "alert_usrgrpid": "7",
        "apm_global_db": {
            "status": 1,
            "url": "https://localhost:8443",
            "authentication_type": 0,
            "username": "zabbix",
            "db": "",
            "ssl_verify_peer": 1,
            "ssl_verify_host": 1
        },
        "auditlog_enabled": "1",
        "auditlog_mode": "1",
        "blink_period": "2m",
        "connect_timeout": "3s",
        "custom_color": "0",
        "default_inventory_mode": "-1",
        "default_lang": "en_US",
        "default_theme": "blue-theme",
        "default_timezone": "system",
        "device_link_timeout": "60s",
        "discovery_groupid": "5",
        "geomaps_attribution": "",
        "geomaps_max_zoom": "0",
        "geomaps_tile_provider": "OpenStreetMap.Mapnik",
        "geomaps_tile_url": "",
        "ha_failover_delay": "1m",
        "history_period": "24h",
        "iframe_sandboxing_enabled": "1",
        "iframe_sandboxing_exceptions": "",
        "item_test_timeout": "60s",
        "login_attempts": "5",
```

```
        "login_block": "30s",
        "max_in_table": "50",
        "max_overview_table_size": "50",
        "max_period": "2y",
        "media_type_test_timeout": "65s",
        "ok_ack_color": "009900",
        "ok_ack_style": "1",
        "ok_period": "5m",
        "ok_unack_color": "009900",
        "ok_unack_style": "1",
        "period_default": "1h",
        "problem_ack_color": "CC0000",
        "problem_ack_style": "1",
        "problem_unack_color": "CC0000",
        "problem_unack_style": "1",
        "proxy_secrets_provider": "0",
        "report_test_timeout": "60s",
        "script_timeout": "60s",
        "search_limit": "1000",
        "server_check_interval": "10",
        "serverid": "019f4707-1a3e-70e6-aca2-286a2b696358",
        "severity_color_0": "97AAB3",
        "severity_color_1": "7499FF",
        "severity_color_2": "FFC859",
        "severity_color_3": "FFA059",
        "severity_color_4": "E97659",
        "severity_color_5": "E45959",
        "severity_name_0": " 未分类",
        "severity_name_1": " 信息",
        "severity_name_2": " 警告",
        "severity_name_3": " 一般",
        "severity_name_4": " 严重",
        "severity_name_5": " 灾难",
        "show_technical_errors": "0",
        "snmptrap_logging": "1",
        "socket_timeout": "3s",
        "timeout_browser": "60s",
        "timeout_db_monitor": "3s",
        "timeout_external_check": "3s",
        "timeout_http_agent": "3s",
        "timeout_script": "3s",
        "timeout_simple_check": "3s",
        "timeout_snmp_agent": "3s",
        "timeout_ssh_agent": "3s",
        "timeout_telemetry_query": "3s",
        "timeout_telnet_agent": "3s",
        "timeout_zabbix_agent": "3s",
        "uri_valid_schemes": "http,https,ftp,file,mailto,tel,ssh",
        "url": "",
        "validate_uri_schemes": "1",
        "vault_provider": "0",
        "work_period": "1-5,09:00-18:00",
        "x_frame_options": "SAMEORIGIN"
    },
    "id": 1
}
```


来源

CSettings::get() in *ui/include/classes/api/services/CSettings.php*.

趋势

此类用于处理趋势数据。

对象引用：

- 浮点型趋势
- 整型趋势

可用方法：

- trend.get - 检索趋势

趋势对象

以下对象与 `trend` API 直接相关.

> **Note:**
> 趋势对象根据监控项类型的信息而有所不同，它们由 Zabbix Server 创建，不能通过 API 进行修改.

浮点趋势

浮点趋势对象具有以下属性。

属性	类型	描述
clock	timestamp	计算该值所在小时的时间戳。例如，时间戳 “04:00:00” 表示为 “04:00:00-04:59:59” 时间段计算的值。
itemid	ID	相关监控项的 ID。
num	integer	该小时内可用值的数量。
value_min	float	每小时最小值。
value_avg	float	每小时平均值。
value_max	float	每小时最大值。

整数趋势

整数趋势对象具有以下属性。

属性	类型	描述
clock	timestamp	计算该值的小时的时间戳。例如，时间戳 “04:00:00” 表示为 “04:00:00-04:59:59” 这一时间段计算的值。
itemid	ID	相关监控项的 ID。
num	integer	该小时可用值的数量。
value_min	integer	每小时最小值。
value_avg	integer	每小时平均值。
value_max	integer	每小时最大值。

获取

描述

`integer/array trend.get(object parameters)`

该方法允许您根据给定参数检索趋势数据。

> **Note:**
> 此方法对任何类型的用户都可用。
> 可在用户角色设置中撤销调用该方法的权限。
> 有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
itemids	ID/array	仅返回具有给定监控项 ID 的趋势数据。
time_from	timestamp	仅返回在给定时间之后或该时间点采集的值。
time_till	timestamp	仅返回在给定时间之前或该时间点采集的值。
countOutput	boolean	统计检索到的对象数量。
limit	integer	限制检索到的对象数量。
output	query	设置要返回的Trend 对象 属性。

返回值

(integer/array) 返回两者其中任一:

- 一个对象数组
- 如果已经使用了 countOutput 参数，则统计对象的数量.

示例

获取监控项趋势数据

请求:

```
{
    "jsonrpc": "2.0",
    "method": "trend.get",
    "params": {
        "output": [
            "itemid",
            "clock",
            "num",
            "value_min",
            "value_avg",
            "value_max"
        ],
        "itemids": [
            "23715"
        ],
        "limit": "1"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23715",
            "clock": "1446199200",
            "num": "60",
            "value_min": "0.165",
            "value_avg": "0.2168",
            "value_max": "0.35"
        }
    ],
    "id": 1
}
```

源码

CTrend::get() in *ui/include/classes/api/services/CTrend.php*.

连接器

此类用于处理连接器。

对象参考：

- 连接器
- 标签过滤器

可用方法：

- connector.create - 创建新的连接器
- connector.delete - 删除连接器
- connector.get - 检索连接器
- connector.update - 更新连接器

连接器对象

以下对象与 `connector` API 直接相关。

连接器

连接器对象具有以下属性。

属性	类型	描述
connectorid	ID	连接器的 ID。 属性行为： - 只读 - 更新操作中为必需
name	string	连接器的名称。 属性行为： - 创建操作中为必需
url	string	端点 URL，即接收方的 URL。 支持用户宏。 属性行为： - 创建操作中为必需
protocol	integer	通信协议。 可能的值： 0 - （默认） Zabbix Streaming Protocol v1.0。
data_type	integer	数据类型。 可能的值： 0 - （默认）监控项值； 1 - 事件。
item_value_type	integer	要发送的监控项值类型之和。 可能的值： 1 - 数值（浮点）； 2 - 字符； 4 - 日志； 8 - 数值（无符号）； 16 - 文本； 32 - 二进制； 64 - JSON。 默认值：31 - 所有监控项类型（二进制和 JSON 除外）。 属性行为： - 当 `data_type` 设置为 “监控项值” 时支持。
max_records	integer	一条消息中可以发送的事件或监控项的最大数量。 可能的值：0-2147483647（32 位有符号整数的最大值）。 默认值：0 - 不限。

属性	类型	描述
max_senders	integer	为此连接器运行的发送进程数量。 可能的值：1-100。 默认值：1。
max_attempts	integer	尝试次数。 可能的值：1-5。 默认值：1。
attempt_interval	string	重试尝试之间的间隔。 接受秒数。 可能的值：0s-10s。 默认值：5s。 属性行为： - 当 `max_attempts` 大于 1 时支持。
timeout	string	超时时间。 支持时间后缀（例如 30s、1m）。 支持用户宏。 可能的值：1s-60s。 默认值：5s。
http_proxy	string	HTTP(S) proxy 连接字符串，格式为 *[protocol]://[username[:password]@]proxy.example.com[:port]*。 支持用户宏。
authtype	integer	HTTP 身份验证方法。 可能的值： 0 - （默认）无； 1 - Basic； 2 - NTLM； 3 - Kerberos； 4 - Digest； 5 - Bearer。
username	string	用户名。 支持用户宏。 属性行为： - 当 `authtype` 设置为 “Basic”、“NTLM”、“Kerberos” 或 “Digest” 时支持
password	string	密码。 支持用户宏。 属性行为： - 当 `authtype` 设置为 “Basic”、“NTLM”、“Kerberos” 或 “Digest” 时支持
token	string	Bearer 令牌。 支持用户宏。 属性行为： - 当 `authtype` 设置为 “Bearer” 时为必需
verify_peer	integer	是否验证主机的证书是否真实有效。 可能的值： 0 - 不验证； 1 - （默认）验证。

属性	类型	描述
verify_host	integer	是否验证连接的主机名是否与主机证书中的名称匹配。 可能的值： 0 - 不验证； 1 - （默认）验证。
ssl_cert_file	string	公共 SSL 密钥文件路径。 支持用户宏。
ssl_key_file	string	私有 SSL 密钥文件路径。 支持用户宏。
ssl_key_password	string	SSL 密钥文件的密码。 支持用户宏。
description	text	连接器的描述。
status	integer	连接器是否启用。 可能的值： 0 - 已禁用； 1 - （默认）已启用。
tags_evaltype	integer	标签评估方法。 可能的值： 0 - （默认） And/Or； 2 - Or。

标签过滤器

标签过滤器允许您仅导出匹配的监控项值或事件。如果未设置，则会导出所有内容。标签过滤器对象具有以下属性。

Property	Type	Description
tag	string	标签名称。 Property behavior: - *required*
operator	integer	条件operator。 可能的值： 0 - *(default)* 等于； 1 - 不等于； 2 - 包含； 3 - 不包含； 12 - 存在； 13 - 不存在。
value	string	标签值。

创建

描述

`object connector.create(object/array connectors)`

此方法允许您创建新的 connector 对象。

Note:
此方法仅适用于 *Super admin* 用户类型。可在用户角色设置中撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object/array)` 要创建的连接器对象。

除了标准连接器属性外，该方法还接受以下参数。

参数	Type	描述
tags	array	连接器的标签过滤器。

返回值

(object) 返回一个对象，其中包含 connectorids 属性下创建的连接器的 ID。返回的 ID 的顺序与传递的连接器的顺序相匹配。

示例

创建连接器

创建一个带有标签过滤的连接器以导出触发器事件。HTTP 身份验证将使用 Bearer 令牌进行。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "connector.create",
    "params": [
        {
            "name": "Export of events",
            "data_type": 1,
            "url": "{$DATA_EXPORT_URL}",
            "authtype": 5,
            "token": "{$DATA_EXPORT_BEARER_TOKEN}",
            "tags": [
                {
                    "tag": "service",
                    "operator": 0,
                    "value": "mysqld"
                }
            ]
        }
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorid": [
            "3"
        ]
    },
    "id": 1
}
```

源码位置

CConnector::create() 在 *ui/include/classes/api/services/CConnector.php 文件中。

删除

描述

`object connector.delete(array connectorids)`

此方法允许您删除 connector 条目。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见User roles。

参数

(array) 要删除的连接器 ID。

返回值

(object) 返回一个对象，该对象在 connectorids 属性下包含已删除连接器的 ID。

示例

删除多个连接器

删除两个连接器。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "connector.delete",
    "params": [
        3,
        5
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorids": [
            "3",
            "5"
        ]
    },
    "id": 1
}
```

源码位置

CConnector::delete() 在 *ui/include/classes/api/services/CConnector.php 文件中。

更新

描述

object connector.update(object/array connectors)

此方法允许您更新现有的 connector。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。调用该方法的权限可在用户角色设置中撤销。有关更多信息，请参见用户角色。

参数

(object/array) 要更新的连接器属性。

必须为每个连接器定义 connectorid 属性，其他所有属性均为可选。仅会更新已传递的属性，其他所有属性将保持不变。

除了标准连接器属性外，该方法还接受以下参数。

参数	类型	描述
tags	array	用于替换当前标签过滤器的连接器标签过滤器。

返回值

(object) 返回一个对象，该对象在 connectorids 属性下包含已更新的连接器的 ID。

示例

更改 HTTP 认证类型

将 ID 为”3” 的连接器的 HTTP 认证类型更改为 Bearer。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "connector.update",
    "params": {
        "connectorid": 3,
        "authtype": 5,
        "token": "{$DATA_EXPORT_BEARER_TOKEN}"
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorids": [
            "3"
        ]
    },
    "id": 1
}
```

更新标签过滤

更改 ID 为”5” 的连接器标签。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "connector.update",
    "params": [
        {
            "connectorid": 5,
            "tags_evaltype": 2,
            "tags": [
                {
                    "tag": "service",
                    "operator": 0,
                    "value": "mysqld"
                },
                {
                    "tag": "error",
                    "operator": 12,
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorids": [
            "5"
        ]
    },
    "id": 1
}
```

源码位置

CConnector::update() 在 *ui/include/classes/api/services/CConnector.php 文件中。

获取

描述

`integer/array connector.get(object parameters)`

该方法允许您根据给定参数检索 connector 对象。

> **Note:**
> 此方法仅适用于 *Super admin* 用户类型。可在用户角色设置中撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 用于定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
connectorids	ID/array	仅返回具有给定 ID 的连接器。
selectTags	query	返回带有连接器标签过滤器 的 `tags` 属性。 支持 `count`。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`connectorid`、`name`、`data_type`、`status`。
countOutput	boolean	这些参数在参考注释中进行了说明。
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

`(integer/array)` 返回以下之一：

- 对象数组；
- 如果使用了 `countOutput` 参数，则返回检索到的对象数量。

示例

获取所有连接器

检索有关所有连接器及其属性的所有数据。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "connector.get",
    "params": {
        "output": "extend",
        "selectTags": ["tag", "operator", "value"],
        "preservekeys": true
    },
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "connectorid": "1",
            "name": "Export of item values",
            "protocol": "0",
            "data_type": "0",
            "url": "{$DATA_EXPORT_VALUES_URL}",
            "item_value_type": "31",
            "authtype": "4",
            "username": "{$DATA_EXPORT_VALUES_USERNAME}",
            "password": "{$DATA_EXPORT_VALUES_PASSWORD}",
            "token": "",
            "max_records": "0",
            "max_senders": "4",
            "max_attempts": "2",
            "attempt_interval": "10s",
            "timeout": "10s",
            "http_proxy": "{$DATA_EXPORT_VALUES_PROXY}",
            "verify_peer": "1",
            "verify_host": "1",
            "ssl_cert_file": "{$DATA_EXPORT_VALUES_SSL_CERT_FILE}",
            "ssl_key_file": "{$DATA_EXPORT_VALUES_SSL_KEY_FILE}",
            "ssl_key_password": "",
            "description": "",
            "status": "1",
            "tags_evaltype": "0",
            "tags": [
                {
                    "tag": "component",
                    "operator": "0",
                    "value": "memory"
                }
            ]
        },
        {
            "connectorid": "2",
            "name": "Export of events",
            "protocol": "0",
            "data_type": "1",
            "url": "{$DATA_EXPORT_EVENTS_URL}",
            "item_value_type": "31",
            "authtype": "5",
            "username": "",
            "password": "",
            "token": "{$DATA_EXPORT_EVENTS_BEARER_TOKEN}",
            "max_records": "0",
            "max_senders": "2",
            "max_attempts": "1",
            "attempt_interval": "5s",
            "timeout": "5s",
            "http_proxy": "",
            "verify_peer": "1",
            "verify_host": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "description": "",
            "status": "1",
            "tags_evaltype": "0",
```

```
            "tags": [
                {
                    "tag": "scope",
                    "operator": "0",
                    "value": "performance"
                }
            ]
        }
    ],
    "id": 1
}
```

源码位置

CConnector:get() 在 *ui/include/classes/api/services/CConnector.php 文件中。

配置

此类用于导出和导入 Zabbix 配置数据。

可用的方法：

- configuration.export - 导出配置数据
- configuration.import - 导入配置数据
- configuration.importcompare - 导入配置之前比较与当前配置差异

对比导入

描述

`array configuration.importcompare(object parameters)`

此方法允许您将导入文件与当前系统元素进行比较，并显示如果导入该文件将会发生哪些更改。

> **Note:**
> 此方法对任何类型的用户都可用。可在用户角色设置中撤销调用此方法的权限。有关更多信息，请参见用户角色。

参数

`(object)` 包含可导入的可能数据以及数据应如何处理的规则的参数。

Parameter	Type	Description
format	string	序列化字符串的格式。 可能的值： `yaml` - YAML； `xml` - XML； `json` - JSON。 参数行为： - *required*
source	string	包含配置数据的序列化字符串。 参数行为： - *required*
rules	object	用于比较新对象和现有对象的规则。 *Admin* 和 *User* 类型用户只能将新对象与其具有 *read-only* 或 *read-write* permission 的现有对象进行比较。 `rules` 参数的详细说明见下表。 参数行为： - *required*

Parameter	Type	Description
returnMissingObjects	boolean	响应对象是否应包含缺失对象信息的规则；默认值：`false`。

Attention:
如果未给出任何规则，则没有内容可更新，结果将为空。

Note:
仅会对主机组和模板进行比较。触发器和图形仅会针对已导入的模板进行比较，其他内容都将被视为“新建”。

`rules` 对象支持以下参数。

Parameter	Type	Description
dashboards	object	导入仪表板的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的仪表板；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true` 且系统中存在某个可导入的仪表板，则会对其进行完整更新。页面和小部件将被新内容完全覆盖；默认值：`false`。
discoveryRules	object	导入 LLD 规则的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的 LLD 规则；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的 LLD 规则；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，数据库中将删除导入数据中不存在的 LLD 规则；默认值：`false`。
global_regexes	object	导入全局正则表达式的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的全局正则表达式；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的全局正则表达式，并完全覆盖其子表达式；默认值：`false`。
graphs	object	导入图形的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的图形；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的图形；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，数据库中将删除导入数据中不存在的图形；默认值：`false`。
host_groups	object	导入主机组的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的主机组；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的主机组；默认值：`false`。

Parameter	Type	Description
httptests	object	导入 web 场景的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的 web 场景；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的 web 场景；默认值：false； deleteMissing - (boolean) 如果设为 true，数据库中将删除导入数据中不存在的 web 场景；默认值：false。
images	object	导入图像的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的图像；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的图像；默认值：false。 此参数不会对输出产生任何影响。仅为与 configuration.import 保持一致而允许使用。
items	object	导入监控项的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的监控项；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的监控项；默认值：false； deleteMissing - (boolean) 如果设为 true，数据库中将删除导入数据中不存在的监控项；默认值：false。
maps	object	导入地图的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的地图；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的地图；默认值：false。 此参数不会对输出产生任何影响。仅为与 configuration.import 保持一致而允许使用。
mediaTypes	object	导入媒体类型的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的媒体类型；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的媒体类型；默认值：false。 此参数不会对输出产生任何影响。仅为与 configuration.import 保持一致而允许使用。
templateDashboards	object	导入模板仪表板的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的模板仪表板；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的模板仪表板；默认值：false； deleteMissing - (boolean) 如果设为 true，数据库中将删除导入数据中不存在的模板仪表板；默认值：false。

Parameter	Type	Description
template_groups	object	导入模板组的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的模板组；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的模板组；默认值：false。
templateLinkage	object	导入模板链接的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将把未链接到正在导入的主机或模板、但存在于导入数据中的模板进行链接；默认值：false； deleteMissing - (boolean) 如果设为 true，将取消与正在导入的主机或模板已建立链接、但不在导入数据中的模板的链接，同时不会移除从这些取消链接的模板继承的实体（监控项、触发器等）；默认值：false。
templates	object	导入模板的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的模板；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的模板；默认值：false。
triggers	object	导入触发器的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的触发器；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的触发器；默认值：false； deleteMissing - (boolean) 如果设为 true，数据库中将删除导入数据中不存在的触发器；默认值：false。
valueMaps	object	导入主机或模板值映射的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的值映射；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的值映射；默认值：false； deleteMissing - (boolean) 如果设为 true，数据库中将删除导入数据中不存在的值映射；默认值：false。

返回值

(array) 返回一个数组，其中包含将进行更改的配置。

示例

比较模板的导入

将 XML 字符串中包含的模板与当前系统元素进行比较，并显示如果导入此模板将会发生哪些更改。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "configuration.importcompare",
    "params": {
        "format": "xml",
        "rules": {
            "discoveryRules": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
```

```
            "graphs": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "host_groups": {
                "createMissing": true,
                "updateExisting": true
            },
            "template_groups": {
                "createMissing": true,
                "updateExisting": true
            },
            "httptests": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "templateLinkage": {
                "createMissing": true,
                "deleteMissing": true
            },
            "templates": {
                "createMissing": true,
                "updateExisting": true
            },
            "templateDashboards": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "triggers": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>8.0</version><templ
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": {
        "templates": {
            "updated": [
                {
                    "before": {
                        "uuid": "5aef0444a82a4d8cb7a95dc4c0c85330",
                        "template": "New template",
```

```
            "name": "New template",
            "groups": [
                {
                    "name": "Templates"
                }
            ]
        },
        "after": {
            "uuid": "5aef0444a82a4d8cb7a95dc4c0c85330",
            "template": "New template",
            "name": "New template",
            "groups": [
                {
                    "name": "Templates"
                }
            ]
        },
        "items": {
            "added": [
                {
                    "after": {
                        "uuid": "648006da5971424ead0c47ddbbf1ea2e",
                        "name": "CPU utilization",
                        "key": "system.cpu.util",
                        "value_type": "FLOAT",
                        "units": "%"
                    },
                    "triggers": {
                        "added": [
                            {
                                "after": {
                                    "uuid": "736225012c534ec480c2a66a91322ce0",
                                    "expression": "avg(/New template/system.cpu.util,3m)>70",
                                    "name": "CPU utilization too high on 'New host' for 3 minu
                                    "priority": "WARNING"
                                }
                            }
                        ]
                    }
                }
            ],
            "removed": [
                {
                    "before": {
                        "uuid": "6805d4c39a624a8bab2cc8ab63df1ab3",
                        "name": "CPU load",
                        "key": "system.cpu.load",
                        "value_type": "FLOAT"
                    },
                    "triggers": {
                        "removed": [
                            {
                                "before": {
                                    "uuid": "ab4c2526c2bc42e48a633082255ebcb3",
                                    "expression": "avg(/New template/system.cpu.load,3m)>2",
                                    "name": "CPU load too high on 'New host' for 3 minutes",
                                    "priority": "WARNING"
                                }
                            }
                        ]
                    }
                }
```

```
                        ],
                        "updated": [
                            {
                                "before": {
                                    "uuid": "7f1e6f1e48aa4a128e5b6a958a5d11c3",
                                    "name": "Zabbix agent ping",
                                    "key": "agent.ping"
                                },
                                "after": {
                                    "uuid": "7f1e6f1e48aa4a128e5b6a958a5d11c3",
                                    "name": "Zabbix agent ping",
                                    "key": "agent.ping",
                                    "delay": "3m"
                                }
                            }
                        ]
                    }
                }
            ]
        }
    },
    "id": 1
}
```


源码位置

CConfiguration::importcompare() 在 *ui/include/classes/api/services/CConfiguration.php* 文件中。

导入

描述

`boolean configuration.import(object parameters)`

此方法允许您从序列化字符串中导入配置数据。

Note:
此方法对任何类型的用户都可用。在用户角色设置中可以撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 包含要导入的数据以及数据应如何处理的规则的参数。

Parameter	Type	Description
format	string	序列化字符串的格式。 可能的值： `yaml` - YAML； `xml` - XML； `json` - JSON。 参数行为： - *required*
source	string	包含配置数据的序列化字符串。 参数行为： - *required*

Parameter	Type	Description
rules	object	关于如何导入新对象和现有对象的规则。 *Admin* 类型用户只能导入其具有 *read-write* 权限的对象，以及地图。例如，只有当用户的用户组对导入的主机将所属的主机组具有权限时，才能导入主机及其实体（监控项、触发器、图形等）。*Admin* 类型用户不能导入图像和媒体类型。 `rules` 参数的详细说明见下表。 参数行为： - *required*
returnMissingObjects	boolean	指定响应对象是否应包含缺失对象的信息；默认值：`false`。

Attention:
如果未提供任何规则，则不会更新配置。

`rules` 对象支持以下参数。

Parameter	Type	Description
dashboards	object	关于如何导入仪表板的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的仪表板；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，且系统中存在某个可导入的仪表板，则会对其进行完整更新。页面和小部件将被新内容完全覆盖；默认值：`false`。
discoveryRules	object	关于如何导入 LLD 规则的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的 LLD 规则；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的 LLD 规则；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，将从数据库中删除导入数据中不存在的 LLD 规则；默认值：`false`。
global_regexes	object	关于如何导入全局正则表达式的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的全局正则表达式；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的全局正则表达式，其子表达式将被完全覆盖；默认值：`false`。
graphs	object	关于如何导入图形的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的图形；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的图形；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，将从数据库中删除导入数据中不存在的图形；默认值：`false`。
host_groups	object	关于如何导入主机组的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的主机组；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的主机组；默认值：`false`。

Parameter	Type	Description
hosts	object	关于如何导入主机的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的主机；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的主机；默认值：`false`。
httptests	object	关于如何导入 web 场景的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的 web 场景；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的 web 场景；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，将从数据库中删除导入数据中不存在的 web 场景；默认值：`false`。
images	object	关于如何导入图像的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的图像；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的图像；默认值：`false`。
items	object	关于如何导入监控项的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的监控项；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的监控项；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，将从数据库中删除导入数据中不存在的监控项；默认值：`false`。
maps	object	关于如何导入地图的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的地图；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的地图；默认值：`false`。
mediaTypes	object	关于如何导入媒体类型的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的媒体类型；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的媒体类型；默认值：`false`。
templateDashboards	object	关于如何导入模板仪表板的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的模板仪表板；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的模板仪表板；默认值：`false`； `deleteMissing` - (`boolean`) 如果设为 `true`，将从数据库中删除导入数据中不存在的模板仪表板；默认值：`false`。
template_groups	object	关于如何导入模板组的规则。 支持的参数： `createMissing` - (`boolean`) 如果设为 `true`，将创建新的模板组；默认值：`false`； `updateExisting` - (`boolean`) 如果设为 `true`，将更新现有的模板组；默认值：`false`。

Parameter	Type	Description
templateLinkage	object	关于如何导入模板链接的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将链接那些未与正在导入的主机或模板关联、但存在于导入数据中的模板；默认值：false； deleteMissing - (boolean) 如果设为 true，将取消那些已与正在导入的主机或模板关联、但不在导入数据中的模板的链接，而不会移除从这些取消链接的模板继承的实体（监控项、触发器等）；默认值：false。
templates	object	关于如何导入模板的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的模板；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的模板；默认值：false。
triggers	object	关于如何导入触发器的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的触发器；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的触发器；默认值：false； deleteMissing - (boolean) 如果设为 true，将从数据库中删除导入数据中不存在的触发器；默认值：false。
valueMaps	object	关于如何导入主机或模板值映射的规则。 支持的参数： createMissing - (boolean) 如果设为 true，将创建新的值映射；默认值：false； updateExisting - (boolean) 如果设为 true，将更新现有的值映射；默认值：false； deleteMissing - (boolean) 如果设为 true，将从数据库中删除导入数据中不存在的值映射；默认值：false。

返回值

(boolean) 如果导入成功，则返回 true。

示例

导入模板

导入 XML 字符串中包含的模板配置。如果 XML 字符串中缺少任何监控项或触发器，它们将从数据库中删除，其他所有内容将保持不变。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "templates": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "triggers": {
                "createMissing": true,
                "updateExisting": true,
```

```
                "deleteMissing": true
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>8.0</version><templ
    },
    "id": 1
}
```


响应：


```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```


源码位置

CConfiguration::import() 在 *ui/include/classes/api/services/CConfiguration.php* 文件中。

导出

描述

`string configuration.export(object parameters)`

此方法允许您将配置数据导出为序列化字符串。

Note:
此方法对任何类型的用户都可用。可在用户角色设置中撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 定义要导出的对象以及所使用格式的参数。

Parameter	Type	Description
format	string	数据必须以何种格式导出。 可能的值： `yaml` - YAML； `xml` - XML； `json` - JSON； `raw` - 未处理的 PHP 数组。 Parameter behavior： - *required*
prettyprint	boolean	通过添加缩进使输出更易于阅读。 可能的值： `true` - 添加缩进； `false` - *(default)* 不添加缩进。

Parameter	Type	Description
options	object	要导出的对象。 options 对象具有以下参数： host_groups - (array) 要导出的主机组 ID； hosts - (array) 要导出的主机 ID； images - (array) 要导出的图像 ID； maps - (array) 要导出的地图 ID； mediaTypes - (array) 要导出的媒体类型 ID； template_groups - (array) 要导出的模板组 ID； templates - (array) 要导出的模板 ID； dashboards - (array) 要导出的仪表板 ID； global_regexes - (array) 要导出的全局正则表达式 ID。 *Admin* 和 *User* 类型用户只能导出其具有 *read-only* 或 *read-write* permission 的对象，以及图像，但不能导出媒体类型。 Parameter behavior： - *required*.

返回值

(string) 返回一个包含请求配置的数据序列化字符串。

示例

导出模板

将模板 “10571” 的配置导出为 XML 字符串。

请求：

```
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "templates": [
                "10571"
            ]
        },
        "format": "xml"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>8.0</version><template_
    "id": 1
}
```

源码位置

CConfiguration::export() 在 *ui/include/classes/api/services/CConfiguration.php* 文件中。

问题

此类用于处理问题。

对象引用：

- 问题
  - 媒体类型 URL
- 问题标签

可用方法：

- problem.get - 检索问题

问题对象

下列对象直接关联到 问题 API。

问题

Note:
问题由 Zabbix 服务器创建，无法通过 API 修改。

问题对象具有以下属性。

属性	Type	描述
eventid	ID	问题事件的 ID。
source	integer	问题事件的类型。 可能的值： 0 - 由触发器创建的事件； 3 - 内部事件； 4 - 在服务状态更新时创建的事件。
object	integer	与问题事件相关的对象类型。 如果 `source` 设置为 “由触发器创建的事件”，可能的值为： 0 - 触发器。 如果 `source` 设置为 “内部事件”，可能的值为： 0 - 触发器； 4 - 监控项； 5 - LLD 规则。 如果 `source` 设置为 “在服务状态更新时创建的事件”，可能的值为： 6 - 服务。
objectid	ID	相关对象的 ID。
clock	timestamp	创建问题事件的时间。
ns	integer	创建问题事件时的纳秒值。
r_eventid	ID	恢复事件的 ID。
r_clock	timestamp	创建恢复事件的时间。
r_ns	integer	创建恢复事件时的纳秒值。
cause_eventid	ID	原因事件的 ID。
correlationid	ID	如果此事件是由全局关联规则恢复的，则为关联规则的 ID。
userid	ID	关闭该问题的用户 ID（如果该问题是手动关闭的）。
name	string	已恢复问题的名称。
acknowledged	integer	问题的确认状态。 可能的值： 0 - 未确认； 1 - 已确认。
severity	integer	问题当前的严重性。 可能的值： 0 - 未分类； 1 - 信息； 2 - 警告； 3 - 一般； 4 - 高； 5 - 灾难。

属性	Type	描述
suppressed	integer	问题是否被抑制。 可能的值： 0 - 问题处于正常状态； 1 - 问题已被抑制。
opdata	string	带有已展开宏的运行数据。
urls	array	活动的媒介类型 URL。

媒介类型 URL

媒介类型 URL 对象具有以下属性。

属性	类型	描述
name	string	媒介类型中定义的 URL 名称。
url	string	媒介类型中定义的 URL 值。

结果将仅包含启用了事件菜单条目的活动媒介类型的条目。属性中使用的宏将被展开，但如果其中一个属性包含未展开的宏，则这两个属性都将从结果中排除。有关支持的宏，请参见支持的宏。

问题标签

问题标签对象具有以下属性。

属性	Type	描述
tag	string	问题标签名称。
value	string	问题标签值。

获取

描述

`integer/array problem.get(object parameters)`

该方法允许你根据给定参数检索问题。

此方法用于检索未解决的问题。如果指定，还可以额外检索最近已解决的问题。用于定义 “最近” 时长的周期在 *Administration* > *General* 中进行设置。在该周期之前已解决的问题不会保留在问题表中。要检索更早之前已解决的问题，请使用`event.get` 方法。

Attention:
如果已删除实体对应的问题尚未被 housekeeper 清理，此方法可能会返回这些问题。

Note:
任何类型的用户都可以使用此方法。可在用户角色设置中撤销调用该方法的权限。有关更多信息，请参见User roles。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

Parameter	Type	Description
eventids	ID/array	仅返回具有给定 ID 的问题。
groupids	ID/array	仅返回由属于给定主机组的对象创建的问题。
hostids	ID/array	仅返回由属于给定主机的对象创建的问题。
objectids	ID/array	仅返回由给定对象创建的问题。

Parameter	Type	Description
source	integer	仅返回具有给定类型的问题。 有关支持的事件类型列表，请参见problem event object page。 默认值：0 - 由触发器创建的问题。
object	integer	仅返回由给定类型对象创建的问题。 有关支持的对象类型列表，请参见problem event object page。 默认值：0 - 触发器。
acknowledged	boolean	如果设置为 `true`，则仅返回已确认的问题。
action	integer	仅返回已执行给定event update actions 的问题。对于多个操作，请使用任何可接受的位图值之和作为位掩码（例如，34 表示确认并抑制事件）。
action_userids	ID/array	仅返回由给定 ID 的用户执行问题事件更新操作的问题。
suppressed	boolean	如果设置为 `true`，则仅返回已抑制的问题。
symptom	boolean	如果设置为 `true`，则仅返回症状问题事件。
severities	integer/array	仅返回具有给定事件严重性的的问题。仅当 object 为 trigger 时适用。
evaltype	integer	标签evaluation method。 可能的值： 0 - *(默认)* And/Or； 2 - Or。
tags	array	仅返回具有给定标签的问题。 格式：`[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`。 空数组返回所有问题。 可能的operator 值： 0 - *(默认)* Contains； 1 - Equals； 2 - Does not contain； 3 - Does not equal； 4 - Exists； 5 - Does not exist。
recent	boolean	如果设置为 `true`，则返回最近的问题，包括在`ok_period` 内已解决的问题以及所有未解决的问题。 如果设置为 `false`，则仅返回未解决的问题。
eventid_from	string	仅返回 ID 大于或等于给定 ID 的问题。
eventid_till	string	仅返回 ID 小于或等于给定 ID 的问题。
time_from	timestamp	仅返回在给定时间之后或该时间创建的问题。
time_till	timestamp	仅返回在给定时间之前或该时间创建的问题。
selectAcknowledges	query	返回带有问题更新的 `acknowledges` 属性。问题更新按时间倒序排列。 问题更新对象具有以下属性： `acknowledgeid` - (ID) 更新的 ID； `userid` - (ID) 更新该事件的用户 ID； `eventid` - (ID) 被更新事件的 ID； `clock` - (timestamp) 事件被更新的时间； `message` - (string) 消息文本； `action` - (integer) 更新操作类型（参见`event.acknowledge`）； `old_severity` - (integer) 此更新操作之前的事件严重性； `new_severity` - (integer) 此更新操作之后的事件严重性； `suppress_until` - (timestamp) 事件将被抑制到的时间； `taskid` - (ID) 如果当前事件正在进行等级变更，则为任务 ID； `maintenanceid` - (ID) 维护的 ID。 支持 `count`。
selectTags	query	返回带有问题标签的`tags` 属性。输出格式：`[{"tag": "<tag>", "value": "<value>"}, ...]`。

Parameter	Type	Description
selectSuppressionData	query	返回 suppression_data 属性，其中包含活动维护和手动抑制列表： maintenanceid - (ID) 维护的 ID； userid - (ID) 抑制该问题的用户 ID； suppress_until - (integer) 问题被抑制到的时间。
filter	object	仅返回与给定过滤器完全匹配的结果。 接受一个对象，其中键为属性名，值可以是单个值或用于匹配的值数组。 不支持 text data type 的属性。
sortfield	string/array	按给定属性对结果排序。 可能的值：eventid。
countOutput	boolean	这些参数在reference commentary 中有说明。
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

返回值

(integer/array) 返回以下之一：

- 对象数组；
- 如果使用了 countOutput 参数，则返回检索到的对象数量。

示例

获取触发器问题事件

从触发器 “15112” 检索最近的事件。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "problem.get",
    "params": {
        "output": "extend",
        "selectAcknowledges": "extend",
        "selectTags": "extend",
        "selectSuppressionData": "extend",
        "objectids": "15112",
        "recent": true,
        "sortfield": ["eventid"],
        "sortorder": "DESC"
    },
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "1245463",
            "source": "0",
            "object": "0",
            "objectid": "15112",
```

```
            "clock": "1472457242",
            "ns": "209442442",
            "r_eventid": "1245468",
            "r_clock": "1472457285",
            "r_ns": "125644870",
            "correlationid": "0",
            "userid": "1",
            "name": "Zabbix agent on localhost is unreachable for 5 minutes",
            "acknowledged": "1",
            "severity": "3",
            "cause_eventid": "0",
            "suppressed": "1",
            "acknowledges": [
                {
                    "acknowledgeid": "14443",
                    "userid": "1",
                    "eventid": "1245463",
                    "clock": "1472457281",
                    "message": "problem solved",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0",
                    "suppress_until": "1472511600",
                    "taskid": "0",
                    "maintenanceid": "0"
                }
            ],
            "opdata": "",
            "suppression_data": [
                {
                    "maintenanceid": "15",
                    "suppress_until": "1472511600",
                    "userid": "0"
                }
            ],
            "tags": [
                {
                    "tag": "test-tag",
                    "value": "test-value"
                }
            ],
            "urls": []
        }
    ],
    "id": 1
}
```


获取指定用户已确认的问题

获取 ID=10 的用户已确认的问题

请求:


```
{
    "jsonrpc": "2.0",
    "method": "problem.get",
    "params": {
        "output": "extend",
        "action": 2,
        "action_userids": [10],
        "selectAcknowledges": ["userid", "action"],
        "sortfield": ["eventid"],
        "sortorder": "DESC"
    },
```

```
    "id": 1
}
```

响应：

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "1248566",
            "source": "0",
            "object": "0",
            "objectid": "15142",
            "clock": "1472457242",
            "ns": "209442442",
            "r_eventid": "1245468",
            "r_clock": "1472457285",
            "r_ns": "125644870",
            "correlationid": "0",
            "userid": "10",
            "name": "Zabbix agent on localhost is unreachable for 5 minutes",
            "acknowledged": "1",
            "severity": "3",
            "cause_eventid": "0",
            "suppressed": "0",
            "acknowledges": [
                {
                    "acknowledgeid": "9",
                    "userid": "10",
                    "action": "2"
                }
            ],
            "opdata": "",
            "urls": []
        }
    ],
    "id": 1
}
```

另请参阅

- 告警
- 监控项
- 主机
- LLD 规则
- 触发器

来源

ui/include/classes/api/services/CProblem.php 中的 CEvent::get()。

高可用性节点

此类旨在与属于高可用性集群一部分的服务器节点或独立服务器实例配合使用。

对象引用：

- 高可用性节点

可用方法：

- hanode.get - 检索节点

高可用节点对象

以下对象与操作 Zabbix servers 的高可用集群有关。

高可用性节点

Note:
节点由 Zabbix 服务器创建，且不能通过 API 修改。

高可用性节点对象具有以下属性。

Property	Type	Description
ha_nodeid	ID	节点的 ID。
name	string	分配给节点的名称，使用 `zabbix_server.conf` 中的 HANodeName 配置项。对于以独立模式运行的服务器，此值为空。
address	string	节点连接来源的 IP 或 DNS 名称。
port	integer	节点运行所在的端口。
lastaccess	integer	心跳时间，即来自节点的最后更新时间。UTC 时间戳。
status	integer	节点状态。 可能的值： 0 - 待机； 1 - 手动停止； 2 - 不可用； 3 - 活动。

获取

描述

`integer/array hanode.get(object parameters)`

该方法允许您根据给定参数检索高可用集群节点列表。

Note:
此方法仅适用于 超级管理员用户类型。有关更多信息，请参见用户角色。

参数

`(object)` 定义所需输出的参数。

该方法支持以下参数。

参数	类型	说明
ha_nodeids	ID/array	仅返回具有给定节点 ID 的节点。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称，值可以是单个值或用于匹配的值数组。 支持的属性：`name`、`address`、`status`。
sortfield	string/array	按给定属性对结果进行排序。 可选值：`name`、`lastaccess`、`status`。
countOutput	flag	这些参数在参考注释中进行了说明。
limit	integer	
output	query	
preservekeys	boolean	
sortorder	string/array	

返回值

返回 `(integer/array)` 其中之一：

- 一个对象的数组;
- 如果使用了 `countOutput` 参数，则为检索到的对象的数量。

示例

获取一个按状态排序的节点列表

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "preservekeys": true,
        "sortfield": "status",
        "sortorder": "DESC"
    },
    "id": 1
}
```

响应:

```
{
  "jsonrpc": "2.0",
  "result": {
    "ckuo7i1nw000h0sajj3l3hh8u": {
      "ha_nodeid": "ckuo7i1nw000h0sajj3l3hh8u",
      "name": "node-active",
      "address": "192.168.1.13",
      "port": "10051",
      "lastaccess": "1635335704",
      "status": "3"
    },
    "ckuo7i1nw000e0sajwfttc1mp": {
      "ha_nodeid": "ckuo7i1nw000e0sajwfttc1mp",
      "name": "node6",
      "address": "192.168.1.10",
      "port": "10053",
      "lastaccess": "1635332902",
      "status": "2"
    },
    "ckuo7i1nv000c0sajz85xcrtt": {
      "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
      "name": "node4",
      "address": "192.168.1.8",
      "port": "10052",
      "lastaccess": "1635334214",
      "status": "1"
    },
    "ckuo7i1nv000a0saj1fcdkeu4": {
      "ha_nodeid": "ckuo7i1nv000a0saj1fcdkeu4",
      "name": "node2",
      "address": "192.168.1.6",
      "port": "10051",
      "lastaccess": "1635335705",
      "status": "0"
    }
  },
  "id": 1
}
```

通过节点 ID 获取特定节点的列表

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "ha_nodeids": ["ckuo7i1nw000e0sajwfttc1mp", "ckuo7i1nv000c0sajz85xcrtt"]
```

```
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
            "name": "node4",
            "address": "192.168.1.8",
            "port": "10052",
            "lastaccess": "1635334214",
            "status": "1"
        },
        {
            "ha_nodeid": "ckuo7i1nw000e0sajwfttc1mp",
            "name": "node6",
            "address": "192.168.1.10",
            "port": "10053",
            "lastaccess": "1635332902",
            "status": "2"
        }
    ],
    "id": 1
}
```

获取停用节点的列表

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "output": ["ha_nodeid", "address", "port"],
        "filter": {
            "status": 1
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "ha_nodeid": "ckuo7i1nw000g0sajjsjre7e3",
            "address": "192.168.1.12",
            "port": "10051"
        },
        {
            "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
            "address": "192.168.1.8",
            "port": "10052"
        },
        {
            "ha_nodeid": "ckuo7i1nv000d0sajd95y1b6x",
            "address": "192.168.1.9",
            "port": "10053"
        }
```

```
    ],
    "id": 1
}
```

获取备用节点的数量

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "countOutput": true,
        "filter": {
            "status": 0
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "3",
    "id": 1
}
```

检查指定 IP 地址的节点的状态

请求:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "output": ["name", "status"],
        "filter": {
            "address": ["192.168.1.7", "192.168.1.13"]
        }
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "name": "node3",
            "status": "0"
        },
        {
            "name": "node-active",
            "status": "3"
        }
    ],
    "id": 1
}
```

源码

CHaNode::get() in *ui/include/classes/api/services/CHaNode.php*.

注释　方法签名

每个 API 方法页面都包含方法签名。

示例：`object action.create(object/array actions)`

签名	描述
`object`	方法返回的数据类型。
`action.create`	方法的完整名称。
`object/array`	方法接受的数据类型。
`actions`	方法接受内容的简要说明。

数据类型

Zabbix API 支持以下数据类型作为输入：

类型	描述
ID	用于引用实体的唯一标识符。
boolean	布尔值（`true` 或 `false`）。
flag	如果传递的值不等于 `null`，则被视为 `true`；否则，该值被视为 `false`。
integer	整数。
float	浮点数。
string	文本字符串。
text	更长的文本字符串。
timestamp	Unix 时间戳。
array	有序值序列（普通数组）。
object	关联数组。
query	定义要返回的数据的值。该值可以定义为属性名称的数组（仅返回特定属性），或作为以下预定义值之一： `extend` - 返回所有对象属性； `count` - 返回检索到的记录数，仅由某些子查询支持。

Attention:
Zabbix API 返回结果为字符串或数组。

属性行为

某些对象属性会用简短标签标记，以描述其行为。使用的标签如下：

- 只读 - 属性值会自动设置，用户无法定义或更改，即使在某些特定条件下也是如此（例如，继承对象或发现对象的 只读属性）；
- 只写 - 属性值可以设置，但之后无法访问；
- 常量 - 属性值可在创建对象时设置，但之后无法更改；
- 支持 - 不要求设置属性值，但在某些特定条件下允许设置（例如，当 `type` 设置为”Simple check”、”External check”、”SSH agent”、”TELNET agent” 或”HTTP agent” 时为 支持）；但请注意，无论条件如何，支持属性仍可能被设置为其默认值；
- 必需 - 对于所有操作（get 操作除外）或在某些特定条件下，必须设置属性值（例如，创建操作时为 必需；当 `operationtype` 设置为”global script” 且未设置 `opcommand_hst` 时为 必需）。

Note:
对于更新操作，属性在更新操作期间被设置时，即视为 “已设置”。

未标记这些标签的属性为可选。

参数行为

部分操作参数带有简短标签，用于说明该参数在操作中的行为。使用以下标签：

- 只读 - 参数值会自动设置，用户无法定义或更改，即使在某些特定条件下也是如此（例如，继承对象或发现对象的 只读参数）；
- 只写 - 参数值可以设置，但之后无法访问；

- 支持 - 不要求必须设置参数值，但在某些特定条件下允许设置（例如，当 Proxy 对象的 operating_mode 设置为 “passive proxy” 时，参数为 支持）；但请注意，无论条件如何，支持参数仍可能被设置为其默认值；
- 必填 - 必须设置参数值。

未标记标签的参数为可选。

预留 **ID** 值 **“0”** 保留的 ID 值 “0” 可用于过滤元素和删除引用对象。- 例如，要从主机中删除 proxy 的代理，应将 proxyid 设置为 0（“proxyid”：“0”）- 或用于过滤由 zabbix server 监控的主机，proxyids 设置为 0（“proxyids”：“0”）。

通用**”get”** 方法参数 以下参数受所有 get 方法支持：

参数	类型	说明
countOutput	boolean	返回结果中的记录数，而不是实际数据。
editable	boolean	如果设置为 true，则仅返回用户具有写权限的对象。 默认值：false。
excludeSearch	boolean	返回与 search 参数中给定条件不匹配的结果。
filter	object	仅返回与给定过滤条件完全匹配的结果。 接受一个对象，其中键为属性名称（例如，host.get 中的主机对象属性、item.get 中的监控项对象属性等），值可以是单个值或用于匹配的值数组。 不支持 text 数据类型 的属性。 请注意，某些方法对此参数具有特定功能，详见相应的方法页面（例如，host.get 中的 filter 参数还支持主机接口属性）。
limit	integer	限制返回的记录数。
output	query	要返回的对象属性。 请注意，对象 ID（即 hostid、itemid 等）始终包含在响应中，即使未在 output 参数中指定也是如此。 默认值：extend。
preservekeys	boolean	在结果数组中使用 ID 作为键。
search	object	返回与给定模式匹配的结果（不区分大小写）。 接受一个对象，其中键为属性名称（例如，host.get 中的主机对象属性、item.get 中的监控项对象属性等），值为要搜索的字符串。如果未提供其他选项，将执行 LIKE "%…%" 搜索。 仅支持 string 和 text 数据类型 的属性。 请注意，某些方法对此参数具有特定功能，详见相应的方法页面（例如，host.get 中的 search 参数还支持主机接口属性）。
searchByAny	boolean	如果设置为 true，则返回匹配 filter 或 search 参数中任一条件的结果，而不是匹配全部条件。 默认值：false。
searchWildcardsEnabled	boolean	如果设置为 true，则允许在 search 参数中使用”*” 作为通配符。 默认值：false。
sortfield	string/array	按给定属性对结果进行排序。可用于排序的属性列表请参见具体 API get 方法的说明。在排序前不会展开宏。 如果未指定值，则返回的数据不排序。
sortorder	string/array	排序顺序。如果传递的是数组，则每个值将与 sortfield 参数中给定的相应属性对应。 可能的值： ASC - (默认) 升序； DESC - 降序。
startSearch	boolean	search 参数将比较字段的开头，即改为执行 LIKE "…%" 搜索。 如果 searchWildcardsEnabled 设置为 true，则忽略此参数。

实体来源标志　Get 方法会为与低级发现相关的实体（LLD 规则/LLD 规则原型、监控项/监控项原型等）返回一个 flags 属性。此属性可用于标识实体是否已被发现，因为对已发现实体的编辑受到限制。

flags 属性会根据以下这些值的组合（“+” 运算）返回结果：

Value	Description
0	基础实体（监控项、触发器、图形、主机）
1	低级发现规则
2	任意原型（监控项原型、触发器原型、LLD 规则原型等）
4	已发现实体（已发现的监控项、触发器、图形、主机、LLD 规则）

flags 属性返回的组合值可能为：

Value	Combination of	Description
0	0	普通实体（监控项、触发器、图形、主机）。
2	2	实体原型（监控项原型、触发器原型等）。
6	2+4	已发现的监控项、触发器、图形、主机（由原型转换而来）。
1	1	低级发现规则。
3	1+2	低级发现规则原型。
5	1+4	已发现的低级发现规则（由原型转换而来）。
7	1+2+4	已发现的低级发现规则原型。

示例　用户权限检查

用户是否有权写入以”MySQL” 或”Linux” 开头的主机？

请求：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
        "search": {
            "host": ["MySQL", "Linux"]
        },
        "editable": true,
        "startSearch": true,
        "searchByAny": true
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "0",
    "id": 1
}
```

Note:
返回结果是”0” 意思是没有主机具有” 读/写” 权限。

不匹配统计

计算名称不包含子字符串”ubuntu” 的主机数量

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
```

```
        "countOutput": true,
        "search": {
            "host": "ubuntu"
        },
        "excludeSearch": true
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": "44",
    "id": 1
}
```

使用通配符搜索主机

查找名称包含单词”server” 且主机接口的端口号为”10050” 或”10071” 的主机。按主机名降序排序，并限制只输出 5 个主机。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "50003",
            "host": "WebServer-Tomcat02",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
```

```
        },
        {
            "hostid": "50004",
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "99032",
            "host": "MySQL server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    ],
    "id": 1
}
```

使用”preservekeys” 和通配符搜索主机

如果您在前面的请求中添加参数 “preservekeys”，则结果将作为关联数组返回，其中键是对象的 id。

请求:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5,
        "preservekeys": true
    },
    "id": 1
}
```

响应:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "50003": {
            "hostid": "50003",
            "host": "WebServer-Tomcat02",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50005": {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50004": {
            "hostid": "50004",
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "99032": {
            "hostid": "99032",
            "host": "MySQL server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        "99061": {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    },
    "id": 1
}
```


附录 **2.** 从 **7.4** 到 **8.0** 的变更

重大变更 history

ZBXNEXT-10069 为history.get 方法新增了参数 maxValueSize。如果您正在请求具有二进制值（history=5）的监控项数据，请将 maxValueSize 设置为 null；否则，history.get 将返回被截断为 64KiB 的监控项值。

主机

ZBXNEXT-9185 为主机 对象的 monitored_by 属性添加了行为描述。ZBXNEXT-10387 host.massupdate：已移除该方法。请改用 host.update。

主机群组

ZBXNEXT-10387 hostgroup.massupdate：已移除该方法。请改用 hostgroup.update、hostgroup.massadd 或 hostgroup.massremove

hostinterface

ZBXNEXT-10814 hostinterface.create、hostinterface.update、hostinterface.delete、hostinterface.massadd 和 hostinterface.massremove 方法已从 hostinterface API 中移除。请改用主机 对象的方法。

角色

ZBXNEXT-9185 为角色action 对象的 name 属性新增了 select_server_for_monitoring 值。

模板

ZBXNEXT-10387 template.massupdate：已移除该方法。请改用 template.update、template.massadd 或 template_massremove。

模板组

ZBXNEXT-10387 templategroup.massupdate：已移除该方法。请改用 templategroup.update、templategroup.massadd 或 templategroup.massremove。

用户组

ZBXNEXT-9185 向usergroup.create 和usergroup.update 方法添加了新参数 proxies 和 proxy_groups。
 向usergroup.get 方法添加了新参数 proxyids、proxy_groupids、selectProxies 和 selectProxyGroups。
 向用户组对象添加了新属性 proxy_mode 和 proxy_group_mode。

其他更改和错误修复 action

如果事件源为发现/自动注册，则在 conditiontype action 过滤参数中新增值 29（proxy 组）。

apmlog

ZBXNEXT-10083 新增了APM 日志 API。

apmmetric

ZBXNEXT-10083 新增了APM metric API。

apmspan

ZBXNEXT-10083 新增了APM span API。

apmtrace

ZBXNEXT-10083 新增APM trace API。

配置

ZBXNEXT-5419 为configuration.importcompare 方法新增了 returnMissingObjects 参数。
 ZBXNEXT-5419 为rules 对象（configuration.importcompare 方法）新增了 dashboards 参数。
 ZBXNEXT-5419 为configuration.import 方法新增了 returnMissingObjects 参数。
 ZBXNEXT-5419 为rules 对象（configuration.import 方法）新增了 dashboards 参数。ZBXNEXT-10390 为configuration.export 方法的 options 对象新增了 global_regexes 参数。ZBXNEXT-10390 为configuration.importcompare 和configuration.import 方法的 rules 对象新增了 global_regexes 参数。

连接器

ZBXNEXT-10069 已向connector 对象新增 item_value_type 值（64 - JSON）。

仪表板

ZBXNEXT-4769 为graph 小组件新增了仪表板小组件字段 show_hostnames。
 ZBXNEXT-9546 为svggraph 和piechart 小组件新增了仪表板小组件字段 ds.0.item_tags_evaltype、ds.0.item_tags.0.tag、ds.0.item_tags.0.operator 和 ds.0.item_tags.0.value。
 ZBXNEXT-10158 为topitems 小组件新增了仪表板小组件字段 columns.0.aggregate_columns、columns.0.combined_column_name 和 columns.0.column_aggregate_function。
 ZBXNEXT-9424、ZBXNEXT-4769 新增了仪表板小组件类型scatterplot。
 ZBXNEXT-8160 为geomap 小组件新增了仪表板小组件字段 clustering_mode 和 clustering_zoom_level。
 ZBXNEXT-943 为graph 小组件新增了仪表板小组件字段 ds.0.invert_values。
 ZBXNEXT-10657 为honeycomb 小组件的 Color 和 Background color 参数新增了 “dark-blue-theme” 主题的默认值。

drule

ZBXNEXT-9185 更新了发现规则对象的 timeout 属性行为。

设备

ZBXNEXT-10357 新增了Device API，包含 device.get 和 device.offboard 方法。

事件

ZBXNEXT-1150 在event.get 方法的 selectAcknowledges 参数中新增了一个 maintenanceid 属性。

历史记录

ZBXNEXT-10069 已向history.get 方法添加新的 history 值（6 - JSON）。
 ZBXNEXT-10069 已添加新的 JSON history 对象。

主机原型

ZBXNEXT-9546 向hostprototype.get 方法新增了参数 selectInheritedTags。

httptest

ZBXNEXT-9546 已向httptest.get 方法添加了新参数 inheritedTags 和 selectInheritedTags。

监控项

ZBXNEXT-10249 向监控项 对象添加了新的 type 值（24 - Telemetry query）。
 ZBXNEXT-10249 向监控项 对象添加了新的属性 query、time_shift、lookback_limit 和 granularity。
 ZBXNEXT-9185 更新了监控项 对象的 timeout 属性行为。
 ZBXNEXT-10069 向监控项 对象添加了新的 value_type 值（6 - JSON）。
 ZBXNEXT-9546 向item.get 方法添加了新的参数 inheritedTags 和 selectInheritedTags。

监控项原型

ZBXNEXT-10249 向监控项原型对象添加了新的 type 值（24 - Telemetry query）。
 ZBXNEXT-10249 向监控项原型对象添加了新的属性 query、time_shift、lookback_limit 和 granularity。
 ZBXNEXT-10069 向itemprototype对象添加了新的 value_type 值（6 - JSON）。
 ZBXNEXT-9546 向itemprototype.get方法添加了新的参数 selectInheritedTags。

维护

ZBXNEXT-8339 为maintenance.create 和maintenance.update 方法添加了 triggers 和 event_names 属性。problem_tags 属性新增了 “不等于” 和 “不包含” 运算符。
 ZBXNEXT-8339 为maintenance.get 方法添加了 triggerids、selectTriggers 和 selectEventNames 参数。

mediatype

ZBXNEXT-10357 为media type 对象新增了 type 值（5 - Push）。

问题

ZBXNEXT-1150 在problem.get 方法的 selectAcknowledges 参数中新增了一个属性 maintenanceid。

proxy

ZBXNEXT-10084 在proxy.get 方法中新增参数 selectApm。
 ZBXNEXT-10084 在proxy.create 和proxy.update 方法中新增参数 apm。
 ZBXNEXT-10084 新增应用性能监控 (APM) 对象。
 ZBXNEXT-10249 在proxy 对象中新增属性 timeout_telemetry_query。

regexp

ZBXNEXT-10390 为regular expression 对象新增了一个 description 属性。

角色

ZBXNEXT-10469 更改了角色规则对象中 api.access 属性的默认值。默认值现在为 0（禁用）。

serverinfo

ZBXNEXT-10357 新增了服务器信息 API，其方法为 serverinfo.get。

设置

ZBXNEXT-10249 向settings 对象添加了新属性 timeout_telemetry_query。
 ZBXNEXT-10680 向settings 对象添加了新属性 apm_global_db。
 ZBXNEXT-10357 向settings 对象添加了新属性 device_link_timeout。
 ZBXNEXT-10657 向settings 对象的 default_theme 属性添加了 dark-blue-theme 值。

模板

ZBXNEXT-9546 向template.get 方法新增了参数 inheritedTags 和 selectInheritedTags。

templatedashboard

ZBXNEXT-9424 新增了模板仪表板小部件类型scatterplot。

触发器

ZBXNEXT-9546 为trigger.get 方法新增了参数 inheritedTags 和 selectInheritedTags。
 ZBXNEXT-8339 为trigger.get 方法新增了属性 maintenanceids。

triggerprototype

ZBXNEXT-9546 向triggerprototype.get 方法新增了参数 selectInheritedTags。

用户

ZBXNEXT-10657 为user 对象的 theme 属性添加了 dark-blue-theme 值。
 ZBXNEXT-8339 为user.create、user.get 和user.update 方法添加了新属性 default_maintenance_period。

用户目录

ZBXNEXT-9834 向用户目录对象添加了 idp_certificate、sp_private_key、sp_certificate、idp_certificate_hash、sp_private_key_hash 和 sp_certificate_hash 属性。
 DEV-5022 更改了用户目录对象中 sign_assertions、sign_authn_requests、sign_messages、sign_logout_requests 和 sign_logout_responses 属性的默认值。默认值现在为 1（启用）。

附录 **3. 8.0** 中的变更

8.0.1 此版本尚未发布。

扩展

概述 尽管 Zabbix 提供了多种功能，但总可以扩展添加更多功能。扩展是一种方便的方式，可以在不更改其源代码的情况下修改和增强 Zabbix 的监控功能。

您可以通过使用内置扩展选项（捕获器监控项、用户参数等）或使用或创建自定义扩展（可加载模块、插件等）来扩展 Zabbix 功能。

本节概述了用于扩展 Zabbix 的所有选项。

使用自定义命令进行数据收集 Trapper 监控项

Trapper 监控项 是接受传入数据而不是主动查询数据的监控项。Trapper 监控项适用于向 Zabbix 服务器或 proxy 发送特定数据，例如在长时间运行的用户脚本场景中发送周期性的可用性和性能数据。可以使用Zabbix sender 工具或 Zabbix sender 协议 向 Zabbix 服务器或 proxy 发送数据。也可以使用history.push API 方法向 Zabbix 服务器发送数据。

外部检查

外部检查 是通过运行可执行文件（例如 shell 脚本或二进制文件）来执行检查的监控项。

外部检查由 Zabbix server 或 proxy（当主机通过 proxy 监控时）执行，不需要在被监控的主机上运行 agent。

用户参数

用户参数 是用户定义的命令（与用户定义的键相关联），执行后可以从运行 Zabbix agent 的主机检索所需的数据。用户参数对于配置 Zabbix 中未预定义的 agent 或 agent2 监控项很有用。

system.run[] Zabbix agent 监控项

system.run[] Zabbix agent 监控项 是用于执行用户定义命令的监控项(与预定义键 system.run[] 关联，例如 system.run[myscript.sh]）该命令可在运行 Zabbix agent 的主机上执行。

Note:
system.run[] 监控项默认处于禁用状态，如需使用，必须在 Zabbix agent 或 Zabbix agent 2 配置文件中启用（允许）并进行定义（AllowKey 或AllowKeyRegexp 配置参数）。

Attention:
外部检查、用户参数和 Zabbix agent 监控项等监控项中的用户定义命令 system.run[] 均由用于运行 Zabbix 组件的 OS 用户执行。要执行这些命令，此用户必须具有必要的权限。

HTTP agent 监控项

HTTP agent 监控项用于通过 HTTP/HTTPS 执行数据请求。HTTP agent 监控项适用于向 HTTP 端点发送请求，以从 *Elasticsearch* 和 *OpenWeatherMap* 等服务中检索数据，也可用于检查 Zabbix API 的状态或 Apache、NGINX web 服务器的状态等。HTTP agent 监控项（启用捕获功能后）还可以充当trapper 监控项。

脚本监控项

脚本监控项是用于执行用户定义的 JavaScript 代码的项，该代码通过 HTTP/HTTPS 检索数据。当 HTTP 代理项提供的功能不够时，脚本项非常有用。例如，在需要多个步骤或复杂逻辑的苛刻数据收集场景中，可以配置脚本项以进行 HTTP 调用，然后处理收到的数据，然后将转换后的值传递给第二个 HTTP 调

> **Note:**
> Zabbix server 和 proxy 支持 HTTP agent 和脚本监控项，并且不需要在被监控的主机上运行 agent。

高级扩展　可加载模块

可加载模块, 用 C 语言编写, 是一种多功能且注重性能的选项，用于扩展 UNIX 平台上 Zabbix 组件 (server, proxy, agent) 的功能。加载模块基本上是 Zabbix 守护程序使用的共享库，并在启动时加载。该库应包含某些函数，以便 Zabbix 进程可以检测到该文件确实是它可以加载和使用的模块。

可加载模块具有许多优点，包括能够添加新指标或实现任何其他逻辑 (例如，Zabbix 历史数据导出), 出色的性能以及开发、使用和共享其提供的功能的选项。它有助于无故障维护，并有助于更轻松地独立于 Zabbix 核心代码库提供新功能。

可加载模块在复杂的监控设置中特别有用。当监控嵌入式系统时，如果有大量监控参数或逻辑复杂或启动时间较长的繁重脚本，用户参数、`system.run[]` Zabbix 代理项和外部检查等扩展将对性能产生影响。可加载模块提供了一种在不牺牲性能的情况下扩展 Zabbix 功能的方法。

插件

插件扩展了 Zabbix agent 2 的监控能力。
它们使用 Go 编程语言编写，为可加载模块（使用 C 编写）提供了一种替代方案。

您也可以为 Zabbix agent 2 开发自己的插件（独立数据采集器）。请参见开发者中心。

自定义警报　Webhooks

Webhook是一种 Zabbix媒体类型，它提供了一种将 Zabbix 警报功能扩展到外部软件（例如帮助台系统、聊天或 Messenger ）的选项。与脚本项类似，Webhook 可用于使用自定义 JavaScript 代码进行 HTTP 调用，例如，将通知推送到不同的平台（例如 Microsoft Teams、Discord 和 Jira）。它还可以返回一些数据（例如，有关创建的帮助台工单的数据），然后这些数据会显示在 Zabbix 中

现有 webhook 可在 Zabbix Git 代码库中找到。有关自定义 webhook 开发，请参阅Webhook 开发指南.。

警报脚本

警报脚本是一种 Zabbix媒体类型，它提供了创建替代方法（脚本）来处理 Zabbix 警报的选项。如果您对 Zabbix 中现有的发送警报的媒体类型不满意，警报脚本会很有用

前端自定义　自定义主题

可以使用自定义主题来更改 Zabbix 前端的外观。请参阅有关创建和应用您自己的主题的说明。

前端模块

前端模块提供了通过添加第三方模块或开发您自己的模块来扩展 Zabbix 前端功能的选项。使用前端模块，您可以添加新的菜单项、它们各自的视图、操作等。

全局脚本　全局脚本是用户定义的一组命令，可根据配置的范围和用户权限在监控目标上执行（通过 shell (/bin/sh) 解释器）。可以配置全局脚本以执行以下操作：

- 动作操作
- 手动主机动作
- 手动事件动作

全局脚本在许多情况下都很有用。例如，如果配置为操作或手动主机操作，则可以使用全局脚本自动或手动执行远程命令，例如重新启动应用程序（Web 服务器、中间件、CRM 等）或释放磁盘空间（删除旧文件、清理/tmp 目录等）。或者，另一个示例，如果配置为手动事件操作，则可以使用全局脚本来管理外部系统中的问题单。

全局脚本可以由 Zabbix server、proxy 或 agent 程序执行。

> **Attention:**
> 用户定义的命令由运行 Zabbix 组件的 OS 用户执行。要执行这些命令，该用户必须具有必要的权限。

Zabbix API Zabbix API是基于 HTTP 的 API，是 Zabbix 前端的一部分。使用 Zabbix API，您可以执行以下任何操作：

- Automate routine tasks.
- 以编程方式检索和修改 Zabbix 的配置。
- 导入和导出 Zabbix 配置。
- 访问 Zabbix 历史和趋势数据。
- 配置应用程序以与 Zabbix 配合使用。
- 将 Zabbix 与第三方软件集成。
- 自动执行日常任务。

see Zabbix API Method reference. Zabbix API 包含多种方法，这些方法名义上被分组到单独的 API 中。每种方法执行特定任务。有关可用方法以及 Zabbix API 提供的功能的概述，请参阅 Zabbix API方法参考。

可加载模块

概述

可加载模块为扩展 Zabbix 功能提供了一种注重性能的方案。

您可以通过多种方式扩展 Zabbix 功能，例如使用用户参数、外部检查以及system.run Zabbix agent 监控项。这些方式都非常有效，但有一个主要缺点，即 fork()。Zabbix 每次处理用户指标时都必须 fork 一个新进程，这对性能不利。通常这不是什么大问题，但在监控嵌入式系统、监控参数数量庞大，或者使用具有复杂逻辑或较长启动时间的重量级脚本时，这可能会成为一个严重问题。

支持可加载模块后，可以在不牺牲性能的情况下扩展 Zabbix agent、服务器和 proxy。

可加载模块本质上是由 Zabbix 守护进程使用并在启动时加载的共享库。该库应包含特定函数，以便 Zabbix 进程能够识别该文件确实是它可以加载并使用的模块。

可加载模块具有多项优势。卓越的性能以及实现任意逻辑的能力都非常重要，但也许最重要的优势是能够开发、使用和共享 Zabbix 模块。这有助于实现无障碍维护，并帮助更轻松地交付新功能，而且无需依赖 Zabbix 核心代码库。

模块许可和二进制形式分发受 AGPL-3.0 许可证约束（模块在运行时与 Zabbix 链接，并使用 Zabbix 头文件；自 Zabbix 7.0 起，整个 Zabbix 代码均采用 AGPL-3.0 许可证）。Zabbix 不保证二进制兼容性。

模块 API 的稳定性在一个 Zabbix LTS（长期支持）发布周期内得到保证。Zabbix API 的稳定性不作保证（从技术上讲，模块可以调用 Zabbix 内部函数，但不能保证此类模块能够正常工作）。

模块 API

为了使共享库被视为 Zabbix 模块，它应实现并导出若干函数。
当前 Zabbix 模块 API 中有六个函数，其中只有一个是必需的，另外五个是可选的。

必需接口

唯一必需的函数是 zbx_module_api_version()：

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

此函数应返回此模块实现的 API 版本，并且为了使模块能够被加载，该版本必须与 Zabbix 支持的模块 API 版本匹配。Zabbix 支持的模块 API 版本为 ZBX_MODULE_API_VERSION。因此，此函数应返回此常量。为此目的使用的旧常量 ZBX_MODULE_API_VERSION_ONE 现在已定义为等于 ZBX_MODULE_API_VERSION，以保持源代码兼容性，但不建议使用它。

可选接口

可选函数包括 zbx_module_init()、zbx_module_item_list()、zbx_module_item_timeout()、zbx_module_history_write_cbs() 和 zbx_module_uninit()：

```
int zbx_module_init(void);
```

此函数应执行模块所需的初始化（如果有）。如果成功，应返回 ZBX_MODULE_OK。否则，应返回 ZBX_MODULE_FAIL。在后一种情况下，Zabbix 将不会启动。

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

此函数应返回模块支持的监控项列表。每个监控项都在 ZBX_METRIC 结构中定义，详情请参见下文。列表以一个 ZBX_METRIC 结构结束，该结构的"key" 字段为 NULL。

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

如果模块导出了 zbx_module_item_list()，则 Zabbix 会使用此函数来指定 Zabbix 配置文件中的超时设置，模块实现的监控项检查应遵循该设置。这里的"timeout" 参数以秒为单位。

```
ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void);
```

此函数应返回 Zabbix 服务器用于导出不同数据类型历史数据的回调函数。回调函数作为 ZBX_HISTORY_WRITE_CBS 结构的字段提供；如果模块不关心某种类型的历史数据，则相应字段可以为 NULL。

```
int zbx_module_uninit(void);
```

此函数应执行必要的反初始化操作（如果有），例如释放已分配的资源、关闭文件描述符等。

除 zbx_module_uninit() 外，所有函数都会在模块加载时于 Zabbix 启动过程中调用一次；zbx_module_uninit() 则会在 Zabbix 关闭、模块卸载时调用一次。

定义监控项

每个监控项都在一个 ZBX_METRIC 结构中定义：

```
typedef struct
{
    char        *key;
    unsigned    flags;
    int     (*function)();
    char        *test_param;
}
ZBX_METRIC;
```

这里，**key** 是监控项键（例如 "dummy.random"），**flags** 要么是 CF_HAVEPARAMS，要么是 0（取决于该监控项是否接受参数），**function** 是实现该监控项的 C 函数（例如 zbx_module_dummy_random），而 **test_param** 是在 Zabbix agent 使用 -p 标志启动时要使用的参数列表（例如 "1,1000"，可以为 NULL）。定义示例可能如下所示：

```
static ZBX_METRIC keys[] =
{
    { "dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random, "1,1000" },
    { NULL }
}
```

实现监控项的每个函数都应接受两个指针参数，第一个参数类型为 AGENT_REQUEST，第二个参数类型为 AGENT_RESULT：

```
int zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    ...

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}
```

如果成功获取到监控项值，这些函数应返回 SYSINFO_RET_OK。否则，应返回 SYSINFO_RET_FAIL。有关如何从 AGENT_REQUEST 获取信息以及如何在 AGENT_RESULT 中设置信息的详细说明，请参见下面的示例"dummy" 模块。

提供历史导出回调

> **Attention:**
> 通过模块进行的历史导出不再受 Zabbix proxy 支持。

模块可以指定按类型导出历史数据的函数：数值型（浮点）、数值型（无符号）、字符型、文本和日志：

```
typedef struct
{
    void    (*history_float_cb)(const ZBX_HISTORY_FLOAT *history, int history_num);
    void    (*history_integer_cb)(const ZBX_HISTORY_INTEGER *history, int history_num);
    void    (*history_string_cb)(const ZBX_HISTORY_STRING *history, int history_num);
    void    (*history_text_cb)(const ZBX_HISTORY_TEXT *history, int history_num);
    void    (*history_log_cb)(const ZBX_HISTORY_LOG *history, int history_num);
}
ZBX_HISTORY_WRITE_CBS;
```

每个函数都应将 history 数组和 history_num 元素数量作为参数。

根据要导出的历史数据类型，history 分别是以下结构体的数组：

```
typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    double      value;
}
ZBX_HISTORY_FLOAT;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    zbx_uint64_t    value;
}
ZBX_HISTORY_INTEGER;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
}
ZBX_HISTORY_STRING;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
}
ZBX_HISTORY_TEXT;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
    const char  *source;
    int     timestamp;
    int     logeventid;
    int     severity;
}
ZBX_HISTORY_LOG;
```

回调将由 Zabbix 服务器的 history syncer 进程在历史同步过程结束时使用，即在数据写入 Zabbix 数据库并保存到值缓存之后。

Attention:
如果历史导出模块发生内部错误，建议将模块设计为：在恢复之前不要阻塞整个监控，而是丢弃数据，并允许 Zabbix 服务器继续运行。

构建模块

目前模块应在 Zabbix 源码树内构建，因为模块 API 依赖于在 Zabbix 头文件中定义的一些数据结构。

对于可加载模块来说，最重要的头文件是 `include/module.h`，它定义了这些数据结构。帮助 `include/module.h` 正常工作的其他必要系统头文件是 `stdlib.h` 和 `stdint.h`。

有了这些信息，模块的构建就已经准备就绪。模块应包含 `stdlib.h`、`stdint.h` 和 `module.h`，并且构建脚本应确保这些文件位于包含路径中。有关详细信息，请参见下面的示例 “dummy” 模块。

另一个有用的头文件是 include/zbxcommon.h，它定义了 zabbix_log() 函数，可用于日志记录和调试。

配置参数

Zabbix agent、服务器和 proxy 支持使用两个参数处理模块：

- LoadModulePath - 可加载模块所在位置的完整路径
- LoadModule - 启动时要加载的模块。

模块必须位于 LoadModulePath 指定的目录中，或者模块名称前必须包含路径。如果前置路径是绝对路径（以 / 开头），则会忽略 LoadModulePath。允许使用多个 LoadModule 参数。

例如，要扩展 Zabbix agent，可以添加以下参数：

```
LoadModulePath=/usr/local/lib/zabbix/agent/
LoadModule=mariadb.so
LoadModule=apache.so
LoadModule=kernel.so
LoadModule=/usr/local/lib/zabbix/dummy.so
```

agent 启动时，会从 /usr/local/lib/zabbix/agent 目录加载 mariadb.so、apache.so 和 kernel.so 模块，而 dummy.so 将从 /usr/local/lib/zabbix 加载。如果模块缺失、权限不正确，或者共享库不是 Zabbix 模块，agent 将无法启动。

前端配置

Zabbix agent、服务器和 proxy 支持可加载模块。因此，Zabbix 前端中的监控项类型取决于模块加载的位置。如果模块加载到 agent 中，则监控项类型应为"Zabbix agent" 或"Zabbix agent (active)"。如果模块加载到服务器或 proxy 中，则监控项类型应为"Simple check"。

通过 Zabbix 模块进行历史数据导出不需要任何前端配置。如果模块已被服务器成功加载，并提供 zbx_module_history_write_cbs() 函数，且该函数返回至少一个非 NULL 的回调函数，那么历史数据导出将自动启用。

Dummy 模块

Zabbix 包含一个用 C 语言编写的示例模块。
该模块位于 src/modules/dummy 下：

```
alex@alex:~trunk/src/modules/dummy$ ls -l
-rw-rw-r-- 1 alex alex 9019 Apr 24 17:54 dummy.c
-rw-rw-r-- 1 alex alex   67 Apr 24 17:54 Makefile
-rw-rw-r-- 1 alex alex  245 Apr 24 17:54 README
```

该模块文档完善，可作为您自行开发模块的模板。

在按照上文说明在 Zabbix 源码树根目录运行 ./configure 之后，只需执行 make 即可构建 dummy.so。

```
/*
** Zabbix
** Copyright (C) 2001-2020 Zabbix SIA
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
**/

###include <stdlib.h>
###include <string.h>
###include <time.h>
###include <stdint.h>

###include "module.h"
```

```c
/* 该变量保存用于监控项处理的超时设置 */
static int  item_timeout = 0;

/* 模块应将内部函数定义为 static，并使用不同于 Zabbix 内部 */
/* 符号（zbx_*）和可加载模块 API 函数（zbx_module_*）的命名模式，以避免冲突 */
static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);

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

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_api_version                                           *
 *                                                                            *
 * Purpose: returns version number of the module interface                    *
 *                                                                            *
 * Return value: ZBX_MODULE_API_VERSION - version of module.h module is       *
 *               compiled with, in order to load module successfully Zabbix   *
 *               MUST be compiled with the same version of this header file   *
 *                                                                            *
 ******************************************************************************/
int zbx_module_api_version(void)
{
    return ZBX_MODULE_API_VERSION;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_item_timeout                                          *
 *                                                                            *
 * Purpose: set timeout value for processing of items                         *
 *                                                                            *
 * Parameters: timeout - timeout in seconds, 0 - no timeout set               *
 *                                                                            *
 ******************************************************************************/
void    zbx_module_item_timeout(int timeout)
{
    item_timeout = timeout;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_item_list                                             *
 *                                                                            *
 * Purpose: returns list of item keys supported by the module                 *
 *                                                                            *
 * Return value: list of item keys                                            *
 *                                                                            *
 ******************************************************************************/
ZBX_METRIC  *zbx_module_item_list(void)
{
    return keys;
}
```

```c
static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    SET_UI64_RESULT(result, 1);

    return SYSINFO_RET_OK;
}

static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char    *param;

    if (1 != request->nparam)
    {
        /* 设置可选错误消息 */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param = get_rparam(request, 0);

    SET_STR_RESULT(result, strdup(param));

    return SYSINFO_RET_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: dummy_random                                                     *
 *                                                                            *
 * Purpose: a main entry point for processing of an item                      *
 *                                                                            *
 * Parameters: request - structure that contains item key and parameters      *
 *              request->key - item key without parameters                    *
 *              request->nparam - number of parameters                        *
 *              request->params[N-1] - pointers to item key parameters        *
 *              request->types[N-1] - item key parameters types:              *
 *                  REQUEST_PARAMETER_TYPE_UNDEFINED (key parameter is empty) *
 *                  REQUEST_PARAMETER_TYPE_ARRAY (array)                      *
 *                  REQUEST_PARAMETER_TYPE_STRING (quoted or unquoted string) *
 *                                                                            *
 *             result - structure that will contain result                    *
 *                                                                            *
 * Return value: SYSINFO_RET_FAIL - function failed, item will be marked      *
 *                                  as not supported by zabbix                *
 *               SYSINFO_RET_OK - success                                     *
 *                                                                            *
 * Comment: get_rparam(request, N-1) can be used to get a pointer to the Nth  *
 *          parameter starting from 0 (first parameter). Make sure it exists  *
 *          by checking value of request->nparam.                             *
 *          In the same manner get_rparam_type(request, N-1) can be used to   *
 *          get a parameter type.                                             *
 *                                                                            *
 ******************************************************************************/
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char    *param1, *param2;
    int from, to;

    if (2 != request->nparam)
    {
        /* 设置可选错误消息 */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
```

```
        return SYSINFO_RET_FAIL;
    }

    param1 = get_rparam(request, 0);
    param2 = get_rparam(request, 1);

    /* 为简化起见，这里不对参数和类型进行严格校验 */
    from = atoi(param1);
    to = atoi(param2);

    if (from > to)
    {
        SET_MSG_RESULT(result, strdup("Invalid range specified."));
        return SYSINFO_RET_FAIL;
    }

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_init                                                  *
 *                                                                            *
 * Purpose: the function is called on agent startup                           *
 *          It should be used to call any initialization routines             *
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - module initialization failed               *
 *                                                                            *
 * Comment: the module won't be loaded in case of ZBX_MODULE_FAIL             *
 *                                                                            *
 ******************************************************************************/
int zbx_module_init(void)
{
    /* dummy.random 的初始化 */
    srand(time(NULL));

    return ZBX_MODULE_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_uninit                                                *
 *                                                                            *
 * Purpose: the function is called on agent shutdown                          *
 *          It should be used to cleanup used resources if there are any      *
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - function failed                            *
 *                                                                            *
 ******************************************************************************/
int zbx_module_uninit(void)
{
    return ZBX_MODULE_OK;
}

/******************************************************************************
 *                                                                            *
 * Functions: dummy_history_float_cb                                          *
 *            dummy_history_integer_cb                                        *
```

```
 *             dummy_history_string_cb                                        *
 *             dummy_history_text_cb                                          *
 *             dummy_history_log_cb                                           *
 *                                                                            *
 * Purpose: callback functions for storing historical data of types float,    *
 *          integer, string, text and log respectively in external storage    *
 *                                                                            *
 * Parameters: history     - array of historical data                         *
 *             history_num - number of elements in history array              *
 *                                                                            *
 ******************************************************************************/
static void dummy_history_float_cb(const ZBX_HISTORY_FLOAT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* 对 history[i].itemid、history[i].clock、history[i].ns、history[i].value 等进行处理 */
    }
}

static void dummy_history_integer_cb(const ZBX_HISTORY_INTEGER *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* 对 history[i].itemid、history[i].clock、history[i].ns、history[i].value 等进行处理 */
    }
}

static void dummy_history_string_cb(const ZBX_HISTORY_STRING *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* 对 history[i].itemid、history[i].clock、history[i].ns、history[i].value 等进行处理 */
    }
}

static void dummy_history_text_cb(const ZBX_HISTORY_TEXT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* 对 history[i].itemid、history[i].clock、history[i].ns、history[i].value 等进行处理 */
    }
}

static void dummy_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* 对 history[i].itemid、history[i].clock、history[i].ns、history[i].value 等进行处理 */
    }
}

/******************************************************************************
```

```
 *                                                                            *
 * Function: zbx_module_history_write_cbs                                     *
 *                                                                            *
 * Purpose: returns a set of module functions Zabbix will call to export      *
 *          different types of historical data                                *
 *                                                                            *
 * Return value: structure with callback function pointers (can be NULL if    *
 *               module is not interested in data of certain types)           *
 *                                                                            *
 ******************************************************************************/
ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void)
{
    static ZBX_HISTORY_WRITE_CBS    dummy_callbacks =
    {
        dummy_history_float_cb,
        dummy_history_integer_cb,
        dummy_history_string_cb,
        dummy_history_text_cb,
        dummy_history_log_cb,
    };

    return dummy_callbacks;
}
```

该模块导出三个新监控项：

- dummy.ping - 始终返回 1
- dummy.echo[param1] - 原样返回第一个参数，例如 dummy.echo[ABC] 将返回 ABC
- dummy.random[param1, param2] - 返回位于 param1-param2 范围内的随机数，例如 dummy.random[1,1000000]

限制

可加载模块的支持仅在 Unix 平台上实现。这意味着它不适用于 Windows agent。

在某些情况下，模块可能需要从 zabbix_agentd.conf 中读取与模块相关的配置参数。目前不支持这样做。如果你需要模块使用某些配置参数，可能应该实现对模块专用配置文件的解析。

前端模块

概述

可以通过添加第三方模块或开发自己的模块来增强 Zabbix 前端功能，而无需修改 Zabbix 的源代码。

请注意，模块代码将以与 Zabbix 源代码相同的权限运行。这意味着：

- 第三方模块可能有害。您必须信任所安装的模块。
- 第三方模块代码中的错误可能会导致前端崩溃。如果发生这种情况，只需从前端中移除该模块代码即可。只要重新加载 Zabbix 前端，您就会看到一条提示，说明某些模块缺失。转到模块管理（位于 *Administration* > *General* > *Modules*），然后再次单击 *Scan directory*，以从数据库中删除不存在的模块。

安装

请始终阅读特定模块的安装手册。建议逐个安装新模块，以便更容易发现故障。

在安装模块之前：

- 确保您已从可信来源下载该模块。安装有害代码可能会导致数据丢失等后果。
- 同一模块的不同版本（相同 ID）可以并行安装，但一次只能启用一个版本。

安装模块的步骤：

- 将模块解压到 Zabbix 前端安装目录中 modules 目录下的独立文件夹中（例如，zabbix/ui/modules）。
- 确保模块文件夹至少包含 manifest.json 文件。
- 导航到模块管理，然后点击 扫描目录按钮。
- 新模块将出现在列表中，并显示其版本、作者、描述和状态。
- 通过点击其状态来启用模块。

故障排除：

问题	解决方案
模块未出现在列表中	如果您的模块没有出现在列表中，请确保 `manifest.json` 存在于 `modules/your-module/` 中。如果不存在，您可能将模块解压到了错误的目录。如果存在，则该模块可能与您的 Zabbix 版本不兼容。另外，请检查 web 服务器用户对模块文件夹及其所有子目录至少具有读取和搜索权限（`r-x`），并对其中所有文件具有读取权限（`r--`）。
前端崩溃	该模块代码与当前 Zabbix 版本或服务器配置不兼容。请删除模块文件并重新加载前端。您将看到某些模块缺失的提示。请前往模块管理，再次点击 扫描目录，以从数据库中移除不存在的模块。
出现关于相同命名空间、*ID* 或操作的错误消息	新模块尝试注册一个已被其他已启用模块注册的命名空间、ID 或操作。请在启用新模块之前，先禁用冲突的模块（错误消息中提到的模块）。
出现技术性错误消息	请将错误报告给该模块的开发者。

开发模块

有关开发自定义模块的信息，请参阅开发者中心。

附录

概述　本节包含参考资料、配置参数、技术规范和操作指南，用于为主手册提供支持。

基本参考

- 安装与设置 - 数据库创建、服务器和 agent 的安装与配置。
- 进程配置 - Zabbix 服务器、proxy、agent 等的配置参数。
- 监控项 - 监控项键值参考和支持的数据类型。
- 宏 - 内置宏和用户宏的语法与用法。

安装和设置

本节涵盖数据库创建、安全配置、特定平台的设置步骤以及功能配置。

请使用侧边栏查找适用于您的环境的步骤。

数据库创建

概述

在安装 Zabbix 服务器或 proxy 期间，必须创建 Zabbix 数据库。

本节提供创建 Zabbix 数据库的说明。每种受支持的数据库都有一套单独的说明。

> **Note:**
> 要通过创建权限最小化的数据库角色/用户来提高数据库安全性，请参阅各受支持数据库的数据库创建最佳实践：

> - MySQL/MariaDB
> - PostgreSQL/TimescaleDB
>
> 有关配置安全的 TLS 连接，请参阅到数据库的安全连接。

UTF-8 是 Zabbix 唯一支持的编码。已知它可以正常工作且不存在安全缺陷。用户应注意，使用其他某些编码时已知存在安全问题。如需切换到 UTF-8，请参阅修复 Zabbix 数据库字符集和排序规则。另请参阅使用 utf8mb4 排序规则进行过滤的限制。

> **Note:**
> 如果从 Zabbix Git repository 安装，则在继续执行后续步骤之前，需要运行以下命令：

 `make dbschema`

MySQL/MariaDB

Zabbix 服务器/proxy 要与 MySQL/MariaDB 数据库正常配合工作，支持使用带有 utf8mb4_bin 排序规则的 utf8mb4 字符集。
 为避免将来出现问题，强烈建议在新安装和现有安装中都使用带有 utf8mb4_bin 排序规则的 utf8mb4 字符集。要将使用 utf8mb3 字符集和 utf8mb3_bin 排序规则的数据库转换为使用 utf8mb4 字符集和 utf8mb4_bin 排序规则的数据库，请按照修复 Zabbix 数据库字符集和排序规则中的说明操作。

在导入架构期间，需要创建确定性触发器。在 MySQL 和 MariaDB 上，如果启用了二进制日志记录、没有超级用户权限，并且 MySQL 配置文件中未设置 `log_bin_trust_function_creators = 1`，则需要设置 `GLOBAL log_bin_trust_function_creators = 1`。

如果您通过软件包或容器安装 Zabbix，请参阅相应的说明。

如果您从源代码安装 Zabbix：

- 创建并配置数据库和用户。出于安全原因，请为 Zabbix 数据库用户使用强度高且唯一的密码。

```
mysql -uroot -p<password>

mysql> create database zabbix character set utf8mb4 collate utf8mb4_bin;
mysql> create user 'zabbix'@'localhost' identified by '<password>';
mysql> grant all privileges on zabbix.* to 'zabbix'@'localhost';
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
mysql> quit;
```

- 将数据导入数据库，并将 utf8mb4 字符集设置为默认字符集（假设您位于 Zabbix 源代码的根目录中）。对于 Zabbix proxy 数据库，只应导入 `schema.sql`（不导入 images.sql 和 data.sql）。

```
cd database/mysql
mysql -uzabbix -p<password> zabbix < schema.sql
#### 如果要为 Zabbix proxy 创建数据库，请在此处停止
mysql -uzabbix -p<password> zabbix < images.sql
mysql -uzabbix -p<password> --default-character-set=utf8mb4 zabbix < data.sql
```

成功导入架构后，可以禁用 `log_bin_trust_function_creators`：

```
mysql -uroot -p<password>

mysql> SET GLOBAL log_bin_trust_function_creators = 0;
mysql> quit;
```

PostgreSQL

您需要拥有具有创建数据库对象权限的数据库用户。

如果您是从 packages 或containers 安装 Zabbix，请参阅相应的说明。

如果您是从源代码安装 Zabbix：

- 创建数据库用户。

以下 shell 命令将创建用户 `zabbix`。在提示时指定密码，并重复输入密码（注意，系统可能会先要求您输入 sudo 密码）：

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

- 创建数据库。

以下 shell 命令将使用之前创建的用户作为所有者（`-O zabbix`）创建数据库 `zabbix`（最后一个参数）。

```
sudo -u postgres createdb -O zabbix -E Unicode -T template0 zabbix
```

- 导入初始架构和数据（假设您位于 Zabbix 源代码的根目录中）。对于 Zabbix proxy 数据库，只应导入 `schema.sql`（不导入 `images.sql` 或 `data.sql`）。

```
cd database/postgresql
cat schema.sql | sudo -u zabbix psql zabbix
#### 如果您正在为 Zabbix proxy 创建数据库，请在此处停止
cat images.sql | sudo -u zabbix psql zabbix
cat data.sql | sudo -u zabbix psql zabbix
```

> **Attention:**
> 以上命令仅作为示例，在大多数 GNU/Linux 安装中都可以正常工作。您可以根据系统/数据库的配置使用不同的命令，例如：

 `psql -U <username>`

 如果您在设置数据库时遇到任何问题，请咨询您的数据库管理员。

TimescaleDB

有关创建和配置 TimescaleDB 的说明，请参见单独的章节。

Elasticsearch 和 ClickHouse

Zabbix 也可以将历史数据存储在Elasticsearch 或ClickHouse 中。有关设置说明，请参阅链接页面。

SQLite

仅 **Zabbix proxy** 支持使用 SQLite！

如果数据库不存在，将自动创建该数据库。

返回安装章节。

修复 **Zabbix** 数据库字符集和排序规则

MySQL/MariaDB

从历史上看，MySQL 及其衍生版本将'utf8' 用作 utf8mb3 的别名 - 这是 MySQL 自己对标准 UTF8 的 3 字节实现，而标准 UTF8 是 4 字节。从 MySQL 8.0.28 和 MariaDB 10.6.1 开始，'utf8mb3' 字符集已被弃用，并且在某个时间点其支持将被移除，而'utf8' 将成为'utf8mb4' 的引用。自 Zabbix 6.0 起，已支持'utf8mb4'。为避免将来出现问题，强烈建议使用'utf8mb4'。切换到'utf8mb4' 的另一个优点是支持补充 Unicode 字符。

Warning:
由于 Zabbix 6.0 之前的版本无法识别 utf8mb4，请务必先将 Zabbix 服务器和 DB schema 升级到 6.0.x 或更高版本，然后再执行 utf8mb4 转换。

1. 检查数据库字符集和排序规则。

例如：

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| latin2                   | latin2 _general_ci   |
+--------------------------+----------------------+
```

或者：

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

如上所示，这里的字符集不是'utf8mb4'，排序规则也不是'utf8mb4_bin'，因此需要进行修正。

2. 停止 Zabbix。

3. 创建数据库备份副本！

4. 在数据库级别修正字符集和排序规则：

```
alter database <your DB name> character set utf8mb4 collate utf8mb4_bin;
```

修正后的值：

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8mb4                  | utf8mb4_bin          |
+--------------------------+----------------------+
```

5. 加载脚本 以修正表级和列级的字符集与排序规则：

```
mysql <your DB name> < utf8mb4_convert.sql
```

6. 执行脚本：

```
                SET @ZABBIX_DATABASE = '<your DB name>';
如果是 MariaDB → set innodb_strict_mode = OFF;
                CALL zbx_convert_utf8();
如果是 MariaDB → set innodb_strict_mode = ON;
                drop procedure zbx_convert_utf8;
```

请注意，'utf8mb4' 预计会占用略多一些磁盘空间。

7. 如果没有错误 - 您可以考虑使用修正后的数据库创建一个数据库备份副本。

8. 启动 Zabbix。

升级数据库以使用主键

概述

本节提供了在现有安装中手动将表升级为主键的说明。

升级为主键可优化数据的索引和访问方式，从而可能加快查询速度并节省空间。它还可改进集群部署中的数据管理和同步，有助于扩展，并确保即使某些服务器发生故障，系统仍能保持可靠。

> **Attention:**
> 本页提供的说明面向高级用户，可能需要根据您的具体配置进行调整。升级为主键可能耗时且占用大量资源。请确保有足够的可用磁盘空间；根据数据库大小和存储的数据量，该过程可能最多需要当前历史表所用空间的 2.5 倍。

自 Zabbix 6.0 起，新安装中的所有表都使用主键。

不会自动将数据库升级为主键；不过，在将 Zabbix 服务器升级到 6.0 或更高版本后，可以手动升级现有安装。

> **Attention:**
> 自 Zabbix 7.0 起，将表升级为主键时，也会将表升级为使用双精度数据类型。

 如果您使用的是 Zabbix 7.0（或更高版本），表已经使用双精度。不过，仍然可以按照本页说明将表升级为主键，而不会影响已经使用双精度的表。

 如果您使用的是 Zabbix 6.4（或更早版本），建议先将表升级为双精度。有关更多信息，请参阅 Zabbix 7.0 文档中的 升级到扩展范围的数值。

可用说明适用于：

- MariaDB/MySQL
- PostgreSQL
- TimescaleDB

重要说明

要执行数据库升级：

1. 停止 Zabbix 服务器。

强烈建议在升级期间停止 Zabbix 服务器。但是，如果确有必要，也可以在服务器运行时执行升级（仅适用于 MySQL、MariaDB，以及未使用 TimescaleDB 的 PostgreSQL）。

2. 备份您的数据库。
3. 安装与您的 Zabbix 版本兼容的最新 zabbix-sql-scripts 软件包（例如，对于 RHEL：`dnf install zabbix-sql-scripts`）。
4. 运行适用于您的数据库的脚本。
5. 启动 Zabbix 服务器。

> **Warning:**
> 仅对服务器数据库运行这些脚本。proxy 不会从此次升级中受益。

如果数据库使用了分区，请联系数据库管理员或 Zabbix 技术支持以获取帮助。

成功升级到主键后，可以删除 CSV 文件。

可选地，可将 Zabbix 前端切换到维护模式。

MariaDB/MySQL

导出和导入必须在 tmux/screen 中执行，以确保会话不会断开。

另请参见：重要说明

使用 mysqlsh 的 MySQL 8.4+

此方法可用于正在运行的 Zabbix 服务器，但建议在升级期间停止服务器。
必须已安装 MySQL Shell（*mysqlsh*）installed，并且能够连接到数据库。

- 以 root 身份（推荐）或任何具有 FILE 权限的用户登录 MySQL 控制台。
- 启用 local_infile 变量后启动 MySQL。
- 通过运行 `history_upgrade_prepare.sql` 重命名旧表并创建新表。

```
mysql -uzabbix -p<password> zabbix < /usr/share/zabbix/sql-scripts/mysql/option-patches/history_upgrade_pr
```

- 导出并导入数据。

通过 mysqlsh 连接。如果使用 socket 连接，可能需要指定路径。

```
sudo mysqlsh -uroot -S /run/mysqld/mysqld.sock --no-password -Dzabbix
```

使用以下命令切换到 JavaScript 模式：

```
\js
```

然后运行下面的代码（可根据需要更改 CSVPATH）：

```
CSVPATH="/var/lib/mysql-files";

util.exportTable("history_old", CSVPATH + "/history.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history.csv", {"dialect": "csv", "table": "history" });

util.exportTable("history_uint_old", CSVPATH + "/history_uint.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_uint.csv", {"dialect": "csv", "table": "history_uint" });

util.exportTable("history_str_old", CSVPATH + "/history_str.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_str.csv", {"dialect": "csv", "table": "history_str" });

util.exportTable("history_log_old", CSVPATH + "/history_log.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_log.csv", {"dialect": "csv", "table": "history_log" });

util.exportTable("history_text_old", CSVPATH + "/history_text.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_text.csv", {"dialect": "csv", "table": "history_text" });
```

如果收到 “JavaScript is not supported” 消息，说明您的 MySQL Shell 安装不支持 JS。
在这种情况下，请安装 Oracle 官方的 MySQL Shell package（或从源代码构建），以启用 JavaScript 模式。

- 按照迁移后说明 删除旧表。

MariaDB/MySQL 8.4+ 且不使用 mysqlsh

此升级方法耗时更长，仅在无法使用 *mysqlsh* 执行升级时才应使用。

表升级

- 以 root 身份（推荐）或任何具有 FILE 权限的用户登录 MySQL 控制台。
- 如果在 Zabbix 服务器运行时执行迁移，请在启用 local_infile 变量的情况下启动 MySQL。
- 通过运行 `history_upgrade_prepare.sql` 重命名旧表并创建新表：

```
mysql -uzabbix -p<password> zabbix < /usr/share/zabbix/sql-scripts/mysql/option-patches/history_upgrade_pr
```

在服务器停止时迁移

在迁移数据之前，必须禁用 `max_execution_time`（在 MySQL 中）或 `max_statement_time`（在 MariaDB 中），以避免迁移期间超时。

对于 MySQL：

```
SET @@max_execution_time=0;
```

对于 MariaDB：

```
SET @@max_statement_time=0;
```

```
INSERT IGNORE INTO history SELECT * FROM history_old;
INSERT IGNORE INTO history_uint SELECT * FROM history_uint_old;
INSERT IGNORE INTO history_str SELECT * FROM history_str_old;
```

```
INSERT IGNORE INTO history_log SELECT * FROM history_log_old;
INSERT IGNORE INTO history_text SELECT * FROM history_text_old;
```

请按照迁移后说明删除旧表。

在服务器运行时迁移

检查启用了哪些导入/导出路径：

```
mysql> SELECT @@secure_file_priv;
+-----------------------+
| @@secure_file_priv    |
+-----------------------+
| /var/lib/mysql-files/ |
+-----------------------+
```

如果 *secure_file_priv* 的值是一个目录路径，则导入/导出将针对该目录中的文件执行。
在这种情况下，请相应地修改查询中的文件路径，或者在升级期间将 *secure_file_priv* 的值设置为空字符串。

如果 *secure_file_priv* 的值为空，则可以从任意位置执行导入/导出。

如果 *secure_file_priv* 的值为 NULL，请将其设置为包含已导出表数据的路径（上例中的'/var/lib/mysql-files/'）。

有关更多信息，请参阅MySQL 文档或MariaDB 文档。

在导出数据之前，必须禁用 max_execution_time（在 MySQL 中）或 max_statement_time（在 MariaDB 中），以避免导出期间超时。

对于 MySQL：

```
SET @@max_execution_time=0;
```

对于 MariaDB：

```
SET @@max_statement_time=0;
```

```
SELECT * INTO OUTFILE '/var/lib/mysql-files/history.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES TER
LOAD DATA INFILE '/var/lib/mysql-files/history.csv' IGNORE INTO TABLE history FIELDS TERMINATED BY ',' ESC

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_uint.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINE
LOAD DATA INFILE '/var/lib/mysql-files/history_uint.csv' IGNORE INTO TABLE history_uint FIELDS TERMINATED

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_str.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES
LOAD DATA INFILE '/var/lib/mysql-files/history_str.csv' IGNORE INTO TABLE history_str FIELDS TERMINATED BY

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_log.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES
LOAD DATA INFILE '/var/lib/mysql-files/history_log.csv' IGNORE INTO TABLE history_log FIELDS TERMINATED BY

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_text.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINE
LOAD DATA INFILE '/var/lib/mysql-files/history_text.csv' IGNORE INTO TABLE history_text FIELDS TERMINATED
```

请按照迁移后说明删除旧表。

PostgreSQL

导出和导入必须在 tmux/screen 中执行，以确保会话不会断开。对于安装了 TimescaleDB 的环境，请跳过本节，继续查看PostgreSQL + TimescaleDB。

另请参见：重要说明

表升级

- 使用 history_upgrade_prepare.sql 重命名表：

```
sudo -u zabbix psql zabbix < /usr/share/zabbix/sql-scripts/postgresql/option-patches/history_upgrade_prepa
```

在服务器停止时进行迁移

- 导出当前历史数据，将其导入临时表，然后在忽略重复项的情况下将数据插入新表：

```
INSERT INTO history SELECT * FROM history_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_uint SELECT * FROM history_uint_old ON CONFLICT (itemid,clock,ns) DO NOTHING;
```

```
INSERT INTO history_str SELECT * FROM history_str_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_log SELECT * FROM history_log_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_text SELECT * FROM history_text_old ON CONFLICT (itemid,clock,ns) DO NOTHING;
```

有关提升 INSERT 性能的提示，请参见：PostgreSQL: Bulk Loading Huge Amounts of Data、Checkpoint Distance and Amount of WAL。

- 按照迁移后说明 删除旧表。

使用运行中的服务器进行迁移

- 导出当前历史数据，将其导入临时表，然后在忽略重复项的情况下将数据插入新表中：

```
\copy history_old TO '/tmp/history.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    DOUBLE PRECISION DEFAULT '0.0000'          NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history FROM '/tmp/history.csv' DELIMITER ',' CSV
INSERT INTO history SELECT * FROM temp_history ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy history_uint_old TO '/tmp/history_uint.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_uint (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    numeric(20)     DEFAULT '0'               NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_uint FROM '/tmp/history_uint.csv' DELIMITER ',' CSV
INSERT INTO history_uint SELECT * FROM temp_history_uint ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy history_str_old TO '/tmp/history_str.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_str (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    varchar(255)    DEFAULT ''                NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_str FROM '/tmp/history_str.csv' DELIMITER ',' CSV
INSERT INTO history_str (itemid,clock,value,ns) SELECT * FROM temp_history_str ON CONFLICT (itemid,clock,n

\copy history_log_old TO '/tmp/history_log.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_log (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    timestamp                integer         DEFAULT '0'               NOT NULL,
    source                   varchar(64)     DEFAULT ''                NOT NULL,
    severity                 integer         DEFAULT '0'               NOT NULL,
    value                    text            DEFAULT ''                NOT NULL,
    logeventid               integer         DEFAULT '0'               NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_log FROM '/tmp/history_log.csv' DELIMITER ',' CSV
INSERT INTO history_log SELECT * FROM temp_history_log ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy history_text_old TO '/tmp/history_text.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_text (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    text            DEFAULT ''                NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
```

```
);
\copy temp_history_text FROM '/tmp/history_text.csv' DELIMITER ',' CSV
INSERT INTO history_text SELECT * FROM temp_history_text ON CONFLICT (itemid,clock,ns) DO NOTHING;
```

- 按照迁移后说明删除旧表。

PostgreSQL + TimescaleDB

导出和导入必须在 tmux/screen 中执行，以确保会话不会断开。
升级期间 Zabbix 服务器应处于关闭状态。

另请参见：重要说明

- 使用 history_upgrade_prepare.sql 重命名表。
  - 如果已启用压缩(默认安装中已启用)，请从 /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patche
    运行脚本：

    ```
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/history_
    ```

  - 如果已禁用压缩，请从 /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compr
    运行脚本：

    ```
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/histo
    ```

- 根据压缩设置运行 TimescaleDB hypertable 迁移脚本：
  - 如果已启用压缩(默认安装中已启用)，请从 /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patche
    运行脚本：

    ```
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/history_
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/history_
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/history_
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/history_
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/history_
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/with-compression/trends_u
    ```

  - 如果已禁用压缩，请从 /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compr
    运行脚本：

    ```
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/histo
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/histo
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/histo
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/histo
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/histo
    cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/option-patches/without-compression/trend
    ```

另请参见：提示，以提升 INSERT 性能。

- 按照迁移后说明删除旧表。

迁移后

对于所有数据库，迁移完成后，请执行以下操作：

- 验证一切是否按预期运行。
- 删除旧表：

```
DROP TABLE history_old;
DROP TABLE history_uint_old;
DROP TABLE history_str_old;
DROP TABLE history_log_old;
DROP TABLE history_text_old;
```

- 对于 TimescaleDB，还需删除以下旧表：

```
DROP TABLE trends_old;
```

另请参阅

- 准备用于分区的审计日志表

为分区准备 **auditlog** 表

概述

某些数据库（例如 MySQL）要求分区列成为表的唯一约束的一部分。因此，auditlog 要按时间对表进行分区，必须将主键从更改 auditid 为复合键 auditid + clock。

本节提供更改 auditlog 表主键的指导说明。

> **Attention:**
> 本页提供的说明适用于高级用户。请注意，这些说明可能需要根据您的具体配置进行调整。更改主键也可能与将来的升级补丁不兼容，因此可能需要手动处理将来的升级。更改主键可能

 是一项资源密集型操作，根据表大小需要花费大量时间 auditlog。建议在更改时停止 Zabbix Server 并将 Zabbix 前端切换到 [维护模式 manual/web_interface/maintenance_mode)。但是，如果绝对必要，有一种方法可以在不停机的情况下更改主键（见下文）。

对 auditlog 表进行分区可以改进例如大型部署中的清理工作。

尽管 Zabbix 清理 目前无法利用分区表（TimescaleDB 除外），但您可以禁用 Zabbix 清理，并使用脚本删除分区。

从 Zabbix 7.0 开始，TimescaleDB 的 auditlog 表已转换为 hypertable，这使得清理进程能够按 chunk 删除数据。要将现有的 auditlog 表升级为 hypertable，请参见升级 TimescaleDB schema。

MySQL

重建索引的重要说明

在 ALTER TABLE 操作期间，MySQL 会自动为主键重建索引。
不过，强烈建议同时使用 OPTIMIZE TABLE 语句手动重建索引，以确保数据库性能达到最佳。

重建索引时，临时所需的额外磁盘空间可能与表本身占用的空间一样多。
要获取数据和索引的当前大小，可以执行以下语句：

```
ANALYZE TABLE auditlog;
SHOW TABLE STATUS LIKE 'auditlog';
```

如果可用磁盘空间是一个问题，请按照无需停机更改主键中的说明进行操作。
还可以考虑以下其他选项：

- 增加 MySQL 参数 sort_buffer_size 可能有助于在手动重建索引时减少磁盘空间使用。
  但是，修改此变量可能会影响数据库的整体内存使用。
- 考虑通过删除可能不必要的数据来释放空间。
- 在执行 housekeeper 之前，考虑降低 数据存储周期housekeeper 参数。

修改主键并停机

1. 删除当前 auditlog 表的主键，并添加新的主键。

```
ALTER TABLE auditlog DROP PRIMARY KEY, ADD PRIMARY KEY (auditid, clock);
```

2. 重建索引（可选，但强烈建议，参见重建索引的重要说明）。

```
OPTIMIZE TABLE auditlog;
```

无停机更改主键

这里介绍手动更改主键的方法。
另外，你也可以使用 Percona 的 pt-online-schema-change 工具包。
该工具包会自动执行以下操作，同时尽量减少更改 auditlog 表所占用的空间。

1. 创建一个带有新主键的新表，并创建索引。

```
CREATE TABLE `auditlog_new` (
  `auditid`             varchar(25)                         NOT NULL,
  `userid`              bigint unsigned                     NULL,
  `username`            varchar(100)    DEFAULT ''          NOT NULL,
  `clock`               integer         DEFAULT '0'         NOT NULL,
  `ip`                  varchar(39)     DEFAULT ''          NOT NULL,
  `action`              integer         DEFAULT '0'         NOT NULL,
  `resourcetype`        integer         DEFAULT '0'         NOT NULL,
  `resourceid`          bigint unsigned                     NULL,
  `resource_cuid`       varchar(25)                         NULL,
  `resourcename`        varchar(255)    DEFAULT ''          NOT NULL,
  `recordsetid`         varchar(25)                         NOT NULL,
  `details`             longtext                            NOT NULL,
```

```
  PRIMARY KEY (auditid,clock)
) ENGINE=InnoDB;
CREATE INDEX `auditlog_1` ON `auditlog_new` (`userid`,`clock`);
CREATE INDEX `auditlog_2` ON `auditlog_new` (`clock`);
CREATE INDEX `auditlog_3` ON `auditlog_new` (`resourcetype`,`resourceid`);
```

2. 交换表。

```
RENAME TABLE auditlog TO auditlog_old, auditlog_new TO auditlog;
```

3. 将旧表中的数据复制到新表。

```
INSERT INTO auditlog SELECT * FROM auditlog_old;
```

可以分批执行（根据需要使用多个带有 WHERE clock 子句的 INSERT INTO 语句），以避免过度占用资源。

4. 删除旧表。

```
DROP TABLE auditlog_old;
```

PostgreSQL

重建索引的重要说明

在 ALTER TABLE 操作期间，PostgreSQL 会自动重建主键的索引。
不过，强烈建议同时使用 REINDEX TABLE CONCURRENTLY 语句手动重建索引，以确保数据库性能达到最佳。

重建索引时，临时所需的磁盘空间最多可能达到当前索引占用空间的三倍。
要获取索引的当前大小，可以执行以下查询：

```
SELECT pg_size_pretty(pg_indexes_size('auditlog'));
```

如果可用磁盘空间是一个问题，请按照无需停机更改主键中的说明进行操作。
还可以使用以下其他选项：

- 增加 PostgreSQL 参数 maintenance_work_mem 可能有助于在手动重建索引时减少磁盘空间使用。
  但是，修改此变量可能会影响数据库的整体内存使用。
- 如果你有另一块磁盘或一个可用空间更多的表空间，可以考虑更改索引重建时的临时存储位置。
  你可以设置 PostgreSQL 参数 temp_tablespaces 来为临时对象指定不同的表空间。
- 考虑通过删除可能不必要的数据来释放空间。
- 在执行 housekeeper 之前，考虑降低 *Data storage period* housekeeper 参数。

修改主键并停机

1. 删除当前 auditlog 表的主键，并添加新的主键。

```
ALTER TABLE auditlog DROP CONSTRAINT auditlog_pkey;
ALTER TABLE auditlog ADD PRIMARY KEY (auditid,clock);
```

2. 重建索引（可选，但强烈建议，参见重建索引的重要说明）。

```
REINDEX TABLE CONCURRENTLY auditlog;
```

无停机更改主键

这里介绍了手动更改主键的方法。
另外，也可以考虑使用 pg_repack 扩展来创建新表、复制数据并交换表。

1. 创建一个具有新主键的新表并创建索引。

```
CREATE TABLE auditlog_new (
  auditid             varchar(25)                          NOT NULL,
  userid              bigint                               NULL,
  username            varchar(100)    DEFAULT ''           NOT NULL,
  clock               integer         DEFAULT '0'          NOT NULL,
  ip                  varchar(39)     DEFAULT ''           NOT NULL,
  action              integer         DEFAULT '0'          NOT NULL,
  resourcetype        integer         DEFAULT '0'          NOT NULL,
  resourceid          bigint                               NULL,
  resource_cuid       varchar(25)                          NULL,
  resourcename        varchar(255)    DEFAULT ''           NOT NULL,
  recordsetid         varchar(25)                          NOT NULL,
```

```
  details               text            DEFAULT ''                NOT NULL,
  PRIMARY KEY (auditid,clock)
);
CREATE INDEX auditlog_new_1 ON auditlog_new (userid,clock);
CREATE INDEX auditlog_new_2 ON auditlog_new (clock);
CREATE INDEX auditlog_new_3 ON auditlog_new (resourcetype,resourceid);
```

2. 交换表。

```
ALTER TABLE auditlog RENAME TO auditlog_old;
ALTER TABLE auditlog_new RENAME TO auditlog;
```

3. 将数据从旧表复制到新表。

```
INSERT INTO auditlog SELECT * FROM auditlog_old;
```

可以分批执行（根据需要使用多个带有 WHERE clock 子句的 INSERT INTO 语句），以避免过度占用资源。

4. 删除旧表。

```
DROP TABLE auditlog_old;
```

另请参阅

- 数据库升级到主键

TimescaleDB 设置

概述

Zabbix 支持 TimescaleDB，这是一种基于 PostgreSQL 的数据库解决方案，可自动按时间将数据分区为多个块，从而在大规模场景下提供更快的性能。

> **Warning:**
> 目前，Zabbix proxy 不支持 TimescaleDB。

本页中的说明可用于以下场景：

- 创建 TimescaleDB 数据库，或将现有 PostgreSQL 表迁移到 TimescaleDB（请参见配置）。
- 在升级 Zabbix 时升级现有 TimescaleDB 数据库架构（请参见升级 TimescaleDB 架构）。

配置

前提条件：数据库服务器上已安装受支持版本的 TimescaleDB 扩展。安装说明请参见 TimescaleDB documentation。

> **Warning:**
> 在安装 TimescaleDB 之前，请先从官方 PostgreSQL 仓库安装受支持的 PostgreSQL 版本。

通过执行以下命令为指定数据库启用 TimescaleDB 扩展：

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

运行此命令需要数据库管理员权限。

> **Note:**
> 如果您使用的数据库 schema 不是 “public”，则需要在上述命令中添加 SCHEMA 子句。例如：
 echo "CREATE EXTENSION IF NOT EXISTS timescaledb SCHEMA yourschema CASCADE;" | sudo -u postgres psql zabbix

然后运行 postgresql/timescaledb/schema.sql 脚本。对于全新安装，必须在使用初始 schema/数据创建常规 PostgreSQL 数据库之后运行该脚本（请参见数据库创建）。

```
cat /usr/share/zabbix/sql-scripts/postgresql/timescaledb/schema.sql | sudo -u zabbix psql zabbix
```

> **Attention:**
> 在 TimescaleDB 2.9.0 及更高版本上运行 schema.sql 脚本时，如果出现提示未遵循最佳实践的警告消息，请忽略。无论是否出现此警告，配置都将成功完成。

现有历史数据、趋势数据和审计日志数据的迁移可能需要较长时间。在迁移期间，Zabbix 服务器和前端必须停止运行。

`schema.sql` 脚本会设置以下 housekeeping 参数：

- 覆盖监控项历史数据保留期
- 覆盖监控项趋势数据保留期

为了对历史数据和趋势数据使用分区 housekeeping，必须同时启用这两个选项。也可以分别单独启用覆盖，例如仅针对历史数据或仅针对趋势数据。

对于 PostgreSQL 和 TimescaleDB，`postgresql/timescaledb/schema.sql` 脚本还会设置另外两个参数：

- 启用压缩
- 压缩 7 天前的记录

要使 housekeeper 成功删除压缩数据，必须同时启用 覆盖监控项历史数据保留期和 覆盖监控项趋势数据保留期选项。

如果未启用覆盖且表中存在压缩块，housekeeper 将不会删除这些表中的数据，并且会在*Housekeeping* 和*System information* 部分显示有关配置不正确的警告。

所有这些参数都可以在安装完成后于 *Administration* > *Housekeeping* 中更改。

Note:
您可能需要运行 TimescaleDB 提供的 timescaledb-tune 工具，以优化 `postgresql.conf` 中的 PostgreSQL 配置参数。

升级 TimescaleDB schema

当将 Zabbix 升级到包含新的 TimescaleDB hypertable 的版本时，Zabbix 服务器不会自动配置这些 hypertable（例如，从 Zabbix 6.4 升级到 8.0 时，由于 7.0.0 和 7.0.2 版本引入了新的 hypertable）。

要配置新的 TimescaleDB hypertable，请按以下步骤操作：

1. 启动 Zabbix 服务器；这将升级现有数据库。
2. 检查服务器日志文件，确认数据库升级已完成；完成后，停止 Zabbix 服务器。
   请注意，如果服务器尝试为不是 hypertable 的表启用压缩，则会记录一条警告。
3. 运行 `postgresql/timescaledb/schema.sql` 脚本；这将配置新的 TimescaleDB hypertable。
   请注意，自 Zabbix 7.0.0 起，该脚本的位置和名称已从 `postgresql/timescaledb.sql` 更改为 `postgresql/timescaledb/schema.`

Attention:
在 TimescaleDB 2.9.0 及更高版本上运行 `schema.sql` 脚本时，如果出现提示未遵循最佳实践的警告消息，请忽略。无论是否出现此警告，配置都将成功完成。

TimescaleDB 压缩

所有属于 TimescaleDB hypertable 的 Zabbix 表都支持原生 TimescaleDB 压缩。在升级或迁移到 TimescaleDB 期间，大型表的初始压缩可能会花费较长时间。

请注意，压缩仅在”timescale” Timescale Community license 下受支持，在”apache” Apache 2.0 license 下不受支持。如果 Zabbix 检测到压缩不受支持，会在 Zabbix 服务器日志中写入警告信息，并且用户无法在前端启用压缩。

Note:
建议用户在使用压缩之前，先熟悉 TimescaleDB documentation 中关于压缩的内容。

请注意，压缩会带来某些限制，具体如下：

- 不允许对已压缩的 chunk 进行修改（插入、删除、更新）
- 不允许对已压缩表进行架构更改。

可以在 Zabbix 前端的 *Administration* > *Housekeeping* 部分中的 *History, trends and audit log compression* 区块中更改压缩设置。

Parameter	Default	Comments
Enable compression	Enabled	勾选或取消勾选该复选框不会立即启用/停用压缩。由于压缩由 Housekeeper 处理，变更将在最多 2 倍 HousekeepingFrequency 小时后生效（在zabbix_server.conf 中设置） 禁用压缩后，落入压缩周期内的新 chunk 将不会被压缩。不过，所有之前已压缩的数据仍将保持压缩状态。要解压之前已压缩的 chunk，请按照 TimescaleDB documentation 中的说明进行操作。
Compress records older than	7d	从较旧版本、且支持 TimescaleDB 的 Zabbix 升级时，默认不会启用压缩。此参数不能小于 7 天。 由于已压缩 chunk 的不可变性，所有晚到的数据（例如由 proxy 延迟传递的数据）如果早于此值，将会被丢弃。

为了获得更好的趋势更新性能，您可能需要将 `trends` 和 `trends_uint` 表的"chunk_time_interval" 从 30 天降低到 7 天或更短，具体取决于使用趋势的监控项数量。此设置的目的是遵循 TimescaleDB 的最佳实践，并确保 chunk 大小保持在系统可用资源范围内。

Elasticsearch 设置

Zabbix 可以将历史数据存储在 Elasticsearch 中，作为关系型数据库的替代方案。

Attention:
Elasticsearch 的支持目前仍处于实验阶段。

本指南介绍受支持版本 的 Elasticsearch 配置。如果您使用的是其他版本，某些功能可能无法按预期工作。

Elasticsearch 可以存储以下值类型：

监控项值类型	数据库表	Elasticsearch 类型
数值（无符号）	history_uint	uint
数值（浮点）	history	dbl
字符	history_str	str
日志	history_log	log
文本	history_text	text
二进制	history_bin	不受 Zabbix 支持
JSON	history_json	json

Note:
Elasticsearch 不接受 JSON 数组。JSON 值必须是单个对象，或一组对象。

重要说明

- Elasticsearch 需要 libcurl。详情请参见requirements。
- housekeeper 不会删除 Elasticsearch 中的数据。如需控制数据保留时长，请参见 Index Lifecycle Management。
- Zabbix 不会在 Elasticsearch 中计算或存储趋势数据。请考虑延长history storage period 以保留更早的数据。
- 使用 Elasticsearch 时，从数据库检索值的范围查询会受到数据存储周期时间戳的限制。
- Zabbix proxy 不支持 Elasticsearch。

如果尚未安装 Elasticsearch，请在继续之前参考 official installation guide。

配置 Elasticsearch

要将历史数据存储到 Elasticsearch 中，您需要：

- 为要存储的每种值类型创建一个 索引——这是 Elasticsearch 存储数据的位置，类似于关系型数据库中的表。
- 为每个索引定义一个 映射——它定义了数据的结构，类似于表结构。
- 设置一个 ingest pipeline，以便在存储前处理值（JSON 值和基于日期的索引必须使用）。

Elasticsearch 可以将数据存储在每种值类型对应的单个索引中，也可以分布在多个基于日期的索引中。
下面将介绍这两种方法。

将历史数据存储在单个索引中

在这种方法中，给定值类型的所有历史数据都会写入单个索引（例如 `uint` 或 `text`）。

要为 *Numeric (unsigned)* 值类型创建索引，请向您的 Elasticsearch 实例发送以下请求（URL 中使用 `/uint`）：

```
curl -X PUT \
 http://localhost:9200/uint \
 -H 'content-type:application/json' \
 -d '{
     "settings": {
      "index": {
         "number_of_replicas": 1,
         "number_of_shards": 5
      }
   },
   "mappings": {
      "properties": {
         "itemid": { "type": "long" },
         "clock": { "format": "epoch_second", "type": "date" },
         "value": { "type": "long" }
      }
   }
}'
```

Elasticsearch 将返回确认信息，表明索引已创建：

```
{"acknowledged": true, "shards_acknowledged": true, "index": "uint"}
```

对于您希望存储在 Elasticsearch 中的每种其他值类型，都必须发送类似的请求。

> **Note:**
> 所有值类型的映射均可在 Zabbix 源代码仓库 中找到。

例如，要为 *Text* 值类型创建索引：

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

JSON 值类型

与其他值类型不同，JSON 值在存储前需要额外处理。

下面的索引对已解析值和原始值使用了单独的字段，因此需要使用 ingest pipeline 将每个值解析为 JSON 并存储到正确的字段中。

要为 *JSON* 值类型创建索引，请向您的 Elasticsearch 实例发送以下请求（URL 中包含 `/json`）。

```
curl -X PUT \
 http://localhost:9200/json \
 -H 'content-type:application/json' \
 -d '{
   "settings": {
      "number_of_shards": 5,
      "number_of_replicas": 1
   },
   "mappings": {
      "dynamic": false,
      "properties": {
         "itemid": { "type": "long" },
         "clock": { "type": "date", "format": "epoch_second" },
         "ns": { "type": "long" },
         "value_parsed": { "type": "flattened" },
         "value_raw": { "type": "keyword", "ignore_above": 1000000 }
    }
  }
}'
```

然后，创建 ingest pipeline：

```
curl -X PUT \
 http://localhost:9200/_ingest/pipeline/json \
 -H 'content-type:application/json' \
 -d '{
   "processors": [
      {
         "json": {
            "field": "value",
            "target_field": "value_parsed",
            "ignore_failure": true
         }
      },
      {
         "set": {
            "if": "ctx.value_parsed == null",
            "field": "value_raw",
            "value": "{{{ value }}}"
         }
      }
   ],
   "on_failure": [
      {
         "set": {
            "field": "value_raw",
            "value": "{{{ value }}}"
         }
      }
   ]
}'
```

Elasticsearch 将返回一条确认信息，表明 ingest pipeline 已创建：

```
{"acknowledged": true}
```

将历史数据存储在按日期划分的索引中

Elasticsearch 不必将所有历史数据都写入单个索引（例如 `uint`），而是可以将这些数据分布到多个按日期划分的索引中（例如 `uint-2026-01-01`、`uint-2026-01-02`）。这样可以更轻松地随时间管理数据量和保留策略。

要启用此功能，您需要：

- 为要存储的每种值类型创建一个 索引模板——这会告诉 Elasticsearch 在自动创建新的按日期划分的索引时应应用哪种映射。
- 为每种值类型创建一个 ingest pipeline——它会处理每个传入的值，并将其路由到正确的按日期划分的索引。

索引模板

要为 text 索引创建模板，请发送包含以下详细信息的请求：

- 在 Elasticsearch 实例的 URL 中使用 _template/text_template。
- 在 "index_patterns" 字段中使用 "text*" 以匹配索引名称。
- 为 text 值类型使用映射（请参见 Zabbix 源代码仓库 中的映射）。

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

json 索引的模板：

```
curl -X PUT \
 http://localhost:9200/_template/json_template \
 -H 'content-type:application/json' \
 -d '{
   "index_patterns": [ "json*" ],
   "settings": {
      "number_of_shards": 5,
      "number_of_replicas": 1
   },
   "mappings": {
      "dynamic": false,
      "properties": {
         "itemid": { "type": "long" },
         "clock": { "type": "date", "format": "epoch_second" },
         "ns": { "type": "long" },
         "value_parsed": { "type": "flattened" },
         "value_raw": { "type": "keyword", "ignore_above": 1000000 }
      }
   }
}'
```

Ingest pipeline

要为 text 索引创建 ingest pipeline：

- 在 Elasticsearch 实例的 URL 中使用 _ingest/pipeline/text-pipeline。
- 包含一个 date_index_name 处理器，根据时间戳将每个值路由到正确的基于日期的索引。

```
curl -X PUT \
 http://localhost:9200/_ingest/pipeline/text-pipeline \
 -H 'content-type:application/json' \
```

```
 -d '{
   "description": "daily text index naming",
   "processors": [
      {
         "date_index_name": {
            "field": "clock",
            "date_formats": ["UNIX"],
            "index_name_prefix": "text-",
            "date_rounding": "d"
         }
      }
   ]
}'
```

对于 json 索引，该 pipeline 还需要先解析 JSON 值，然后再将其路由到正确的索引：

```
curl -X PUT \
 http://localhost:9200/_ingest/pipeline/json-pipeline \
 -H 'content-type:application/json' \
 -d '{
   "description": "daily json index naming"
   "processors": [
      {
         "json": {
            "field": "value",
            "target_field": "value_parsed",
            "ignore_failure": true
         }
      },
      {
         "script": {
            "source": "if (ctx.value_parsed == null || !(ctx.value_parsed instanceof Map)) { ctx.value_raw
         }
      },
      {
         "date_index_name": {
            "field": "clock",
            "date_formats": [ "UNIX" ],
            "index_name_prefix": "json-",
            "date_rounding": "d"
         }
      }
   ]
}'
```

配置 Zabbix 服务器

在 Zabbix 服务器配置文件 (zabbix_server.conf) 中，设置HistoryProvider 参数。

例如，要将 *Character*、*Log*、*Text* 和 *JSON* 类型的值存储到 Elasticsearch 中（同时将 *Numeric* 值保留在数据库中）：

```
HistoryProvider=elasticsearch;value_types="str,log,text,json",url=http://localhost:9200
```

如果你使用基于日期的索引，请在该参数中添加 date_index=1：

```
HistoryProvider=elasticsearch;value_types="str,log,text,json",url=http://localhost:9200,date_index=1
```

完成更改后，重启 Zabbix 服务器：

```
systemctl restart zabbix-server
```

配置 Zabbix 前端

在 Zabbix 前端配置文件（zabbix.conf.php）中，将$HISTORY_PROVIDERS 变量设置为与服务器配置匹配：

```
$HISTORY_PROVIDERS[] = [
  'types' => ['str','log','text','json'],
  'provider' => 'elasticsearch',
```

```
  'url' => 'http://localhost:9200'
];
```

故障排查

以下步骤可能有助于你排查 Elasticsearch 配置中的问题：

1. 检查 Elasticsearch 或 Zabbix 服务器日志中是否有错误。
2. 要识别慢查询，请在 Zabbix 服务器配置参数HistoryProvider 中使用 log_slow_queries 选项。
3. 确认 Elasticsearch 允许来自 Zabbix 服务器和 Zabbix 前端的访问。
4. 确认已启用 auto_create_index：

```
curl -X GET \
 "http://localhost:9200/_cluster/settings?include_defaults=true&filter_path=**.auto_create_index"

#### {"defaults": {"action": {"auto_create_index": "false"} } }
```

要启用它，请发送以下请求：

```
curl -X PUT \
 http://localhost:9200/_cluster/settings \
 -H 'content-type:application/json' \
 -d '{
   "persistent": {
      "action.auto_create_index": "true"
   }
}'

#### {"acknowledged": true, "persistent": {"action": {"auto_create_index": "true"} }, "transient": {} }
```

5. 通过向各自的 URL 发送 GET 请求，确认映射、模板和 ingest pipeline 是否正确：

```
curl -X GET http://localhost:9200/json
curl -X GET http://localhost:9200/_template/json*
curl -X GET http://localhost:9200/_ingest/pipeline/json*
```

你可以将收到的响应与 Elasticsearch API 文档中的预期响应进行比较。

6. 检查是否有任何 shards 处于失败状态；重启 Elasticsearch 可能会解决此问题。
7. 查询 Elasticsearch，查看 Zabbix 收集的数据是否已存储，例如：

```
curl 'http://localhost:9200/json/_search' \
 -H 'Content-Type: application/json' \
 -d '{
   "query": {
     "term": {
       "itemid": 42269
     }
   }
 }'
```

8. 如果你需要重置 Elasticsearch 配置并重新开始，可以删除所有索引、模板和 ingest pipeline：

```
curl -X DELETE "http://localhost:9200/_all"
curl -X DELETE "http://localhost:9200/_template/*"
curl -X DELETE "http://localhost:9200/_ingest/pipeline/*"
```

ClickHouse 设置

Zabbix 可以将历史数据存储在 ClickHouse 中，作为关系数据库的替代方案。

> **Note:**
> ClickHouse 还可以用作应用性能监控（APM）的遥测数据源；有关详细信息，请参阅 管理 > 数据源。

本指南介绍受支持版本 的 ClickHouse 设置。如果您使用的是其他版本，某些功能可能无法按预期工作。

ClickHouse 可以存储以下值类型：

监控项值类型	数据库表	ClickHouse 类型
数值（无符号）	history_uint	uint
数值（浮点）	history	dbl
字符	history_str	str
日志	history_log	log
文本	history_text	text
二进制	history_bin	Zabbix 不支持
JSON	history_json	json

Note:
ClickHouse 不接受 JSON 数组。JSON 值必须是单个对象或一组对象。此外，ClickHouse 对带有 NULL 的 JSON 键的处理方式与对缺失键的处理方式相同。

重要说明

- housekeeper 不会删除 ClickHouse 中的数据。如需控制数据的保留时长，请在 ClickHouse 中配置 ClickHouse 数据存储期限。
- Zabbix 不会在 ClickHouse 中计算或存储趋势数据。如需保留较早的数据，请考虑延长历史数据存储期限。
- 如果要将历史数据从现有的 Zabbix 数据库（MySQL 或 PostgreSQL）迁移到 ClickHouse，请参阅 ClickHouse 架构和历史数据迁移脚本。
- ClickHouse 不支持作为 Zabbix proxy 的历史数据存储，但在启用 APM 的情况下，可以用作 Zabbix proxy 的遥测数据源。
- 如果您使用 ClickHouse 集群，请参阅配置 ClickHouse 集群。

配置 ClickHouse

您需要创建并配置 Zabbix 数据库和用户，并导入数据库结构。

本指南提供了有关 ClickHouse 的 Docker 或 package 安装的说明。

Docker

1. 在运行 ClickHouse 容器时，创建并配置 Zabbix 数据库和用户：

```
sudo docker run -d \
  --name clickhouse \
  -e CLICKHOUSE_DB=zabbix \
  -e CLICKHOUSE_USER=zabbix \
  -e CLICKHOUSE_DEFAULT_ACCESS_MANAGEMENT=1 \
  -e CLICKHOUSE_PASSWORD=<password> \
  -p 8123:8123/tcp \
  -p 9000:9000/tcp \
  --cap-add=SYS_NICE \
  --cap-add=NET_ADMIN \
  --cap-add=IPC_LOCK \
  --ulimit nofile=262144:262144 \
  -v clickhouse_data:/var/lib/clickhouse \
  clickhouse/clickhouse-server:26.4
```

2. 确认 ClickHouse 正常工作，并且可以连接到它：

```
sudo docker exec -it clickhouse \
  clickhouse-client \
  --query "SELECT version()"

#### 26.4.4.38
```

3. 使用 Zabbix 目录中的 history_all.sh 脚本导入数据库模式：

```
./usr/share/zabbix/sql-scripts/clickhouse/history_all.sh \
  --user zabbix \
  --password <password> \
  --db zabbix \
  --server http://localhost:8123
```

Note:
ClickHouse 的数据存储期限（Time-To-Live，TTL）默认是 31 天。如需更改，请在导入数据库模式时使用 `--ttl` 选项（例如，`--ttl 604800` 表示 7 天），或者稍后再配置。

4. 验证表是否已创建：

```
sudo docker exec -it clickhouse \
  clickhouse-client \
  --user zabbix \
  --password <password> \
  --query "SHOW TABLES FROM zabbix"

#### history
#### history_json
#### history_log
#### history_str
#### history_text
#### history_uint
```

软件包

1. 启动 ClickHouse：

```
sudo service clickhouse-server start
```

2. 创建并配置 Zabbix 数据库和用户：

```
clickhouse-client --user default

localhost :) CREATE DATABASE IF NOT EXISTS zabbix
localhost :) CREATE USER IF NOT EXISTS zabbix IDENTIFIED WITH sha256_password BY '<password>'
localhost :) GRANT CREATE, ALTER, DROP, INSERT, SELECT, UPDATE, OPTIMIZE ON zabbix.* TO zabbix
localhost :) quit
```

3. 使用 Zabbix 目录中的 history_all.sh 脚本导入数据库模式：

```
./usr/share/zabbix/sql-scripts/clickhouse/history_all.sh \
  --user zabbix \
  --password <password> \
  --db zabbix \
  --server http://localhost:8123
```

Note:
ClickHouse 的数据存储期限（Time-To-Live，TTL）默认是 31 天。如需更改，请在导入数据库模式时使用 `--ttl` 选项（例如，`--ttl 604800` 表示 7 天），或者稍后再进行配置。

4. 验证表是否已创建：

```
clickhouse-client \
  --user zabbix \
  --password <password> \
  --query "SHOW TABLES FROM zabbix"

#### history
#### history_json
#### history_log
#### history_str
#### history_text
#### history_uint
```

配置 Zabbix 服务器

在 Zabbix 服务器配置文件（zabbix_server.conf）中，设置HistoryProvider 参数。

例如，要将所有支持的值类型存储到 ClickHouse 中：

```
HistoryProvider=clickhouse;value_types="uint,dbl,str,log,text,json",url=http://localhost:8123,db=zabbix,us
```

完成更改后，重启 Zabbix 服务器：

```
systemctl restart zabbix-server
```

配置 Zabbix 前端

在 Zabbix 前端配置文件（zabbix.conf.php）中，将$HISTORY_PROVIDERS 变量设置为与服务器配置匹配：

```
$HISTORY_PROVIDERS[] = [
  'types' => ['uint','dbl','str','log','text','json'],
  'provider' => 'clickhouse',
  'url' => 'http://localhost:8123',
  'db' => 'zabbix',
  'username' => 'zabbix',
  'password' => '<password>'
];
```

其他配置

以下步骤是可选的。基础设置不需要这些步骤。

配置 ClickHouse 数据保留期

ClickHouse 的数据保留期（Time-To-Live，简称 TTL）默认是 31 天。如需更改，请运行以下命令。

下面的示例使用 Docker。如果你是通过软件包安装的 ClickHouse，请直接在 ClickHouse 客户端中运行这些查询。

1. 修改表（将 history_json 和 3600 替换为你需要的值）：

```
sudo docker exec -it clickhouse \
  clickhouse-client \
  --user zabbix \
  --password <password> \
  --query "ALTER TABLE zabbix.history_json MODIFY TTL clock_ns + toIntervalSecond(3600)"
```

2. 立即应用更改：

```
sudo docker exec -it clickhouse \
  clickhouse-client \
  --user zabbix \
  --password <password> \
  --query "OPTIMIZE TABLE zabbix.history_json MATERIALIZE TTL"
```

3. 验证数据保留期是否已更改：

```
sudo docker exec -it clickhouse \
  clickhouse-client \
  --user zabbix \
  --password <password> \
  --query "SHOW CREATE TABLE zabbix.history_json"

#### CREATE TABLE zabbix.history_json
#### (
####     `itemid` UInt64,
####     `clock_ns` DateTime64(9),
####     `value` JSON,
####     `value_str` String
#### )
#### ENGINE = MergeTree
#### PARTITION BY toDate(clock_ns)
#### PRIMARY KEY (itemid, clock_ns)
#### ORDER BY (itemid, clock_ns)
#### TTL clock_ns + toIntervalSecond(3600)
#### SETTINGS index_granularity = 8192
```

4. 重启 Zabbix 服务器，以在 *Administration* > *Housekeeping* 中刷新数据保留期：

```
systemctl restart zabbix-server
```

配置 ClickHouse 集群

本节提供带复制和分片的 ClickHouse 集群配置步骤。

下面的步骤使用 Docker 以及 ClickHouse 文档中的 ClickHouse `cluster_2S_2R`（运行 ClickHouse 26.4.4.38 和 ClickHouse Keeper）。

在开始之前，建议在 Zabbix 的 ClickHouse 集群前放置一个负载均衡器。Zabbix 服务器和前端配置只支持单个 ClickHouse URL，因此它只能指向一个节点。如果该节点宕机，负载均衡器会通过同一个 URL 将 Zabbix 重定向到健康节点。

1. 创建 Zabbix 数据库：

```
sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "CREATE DATABASE IF NOT EXISTS zabbix ON CLUSTER cluster_2S_2R ENGINE = Replicated('/clickhouse/
```

Attention:
ON CLUSTER 子句只会作用于你运行命令时属于该集群的节点。如果你之后向集群中添加节点，请再次运行此命令。

2. 创建并配置 Zabbix 用户：

```
sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "CREATE USER IF NOT EXISTS zabbix ON CLUSTER cluster_2S_2R IDENTIFIED WITH sha256_password BY '<

sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "GRANT ON CLUSTER cluster_2S_2R CREATE, ALTER, DROP, INSERT, SELECT, UPDATE, OPTIMIZE ON zabbix.
```

3. 使用 Zabbix 目录中的 `history_all.sh` 脚本导入数据库模式：

```
./usr/share/zabbix/sql-scripts/clickhouse/history_all.sh \
  --user zabbix \
  --password <password> \
  --db zabbix \
  --server http://localhost:8123 \
  --engine "ReplicatedMergeTree()"
```

Note:
ClickHouse 的数据存储期限（Time-To-Live，TTL）默认是 31 天。如需更改，可在导入数据库模式时使用 `--ttl` 选项（例如，`--ttl 604800` 表示 7 天），或者稍后再进行配置。

Note:
如果你的 ClickHouse 集群需要显式的副本路径（例如，ClickHouse 服务器配置未为复制表设置默认路径），请分别运行 模式脚本（history_schema.sh、history_uint_schema.sh 等），并为每个表设置不同的路径：

```
./usr/share/zabbix/sql-scripts/clickhouse/history_schema.sh \
  --user zabbix \
  --password <password> \
  --db zabbix \
  --server http://localhost:8123 \
  --engine "ReplicatedMergeTree('/clickhouse/tables/{shard}/history', '{replica}')"
```

有关 ClickHouse 复制的更多详细信息，请参见 ClickHouse 文档中的复制数据和Replicated* 表引擎。

4. 为每种表类型配置分片。重命名这些表，并创建一个使用原始名称的 Distributed 表，指向重命名后的表。这个 Distributed 表才是 Zabbix 实际读写的对象；该表会自动将数据分散到各个分片。例如，对于 history 表：

```
sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "RENAME TABLE zabbix.history TO zabbix.history_local"
```

```
sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "CREATE TABLE zabbix.history AS zabbix.history_local ENGINE = Distributed('cluster_2S_2R', 'zabb
```

Note:
有关 ClickHouse 分片的更多详细信息，请参见 ClickHouse 文档中的 Distributed table engine。

5. 配置 Zabbix 服务器。

6. 配置 Zabbix 前端。

7. 验证 Zabbix 数据是否已分布到各个分片。由于 Zabbix 服务器会持续插入新数据，zabbix.history_local 的行数在重复检查之间可能会发生变化。在每个节点上运行此命令：

```
sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "SELECT count(*) FROM zabbix.history_local"

#### 2446 (clickhouse-01)
#### 2471 (clickhouse-02)
#### 2494 (clickhouse-03)
```

8. 检查复制是否同步（active_replicas 应与 total_replicas 一致）：

```
sudo docker exec -it clickhouse-01 \
  clickhouse-client \
  --user default \
  --query "SELECT database, table, is_leader, total_replicas, active_replicas FROM system.replicas WHERE d

#### zabbix  history_json_local  1 2 2
#### zabbix  history_local       1 2 2
#### zabbix  history_log_local   1 2 2
#### zabbix  history_str_local   1 2 2
#### zabbix  history_text_local  1 2 2
#### zabbix  history_uint_local  1 2 2
```

故障排除

以下步骤可能有助于你排查 ClickHouse 配置中的问题：

1. 检查 ClickHouse 或 Zabbix 服务器日志中的错误。
2. 要识别慢查询，请在HistoryProvider Zabbix 服务器配置参数中使用 log_slow_queries 选项。
3. 验证 ClickHouse 是否允许来自 Zabbix 服务器和 Zabbix 前端的访问。
4. 查询 ClickHouse，查看 Zabbix 收集的数据是否已存储，例如：

```
sudo docker exec -it clickhouse \
  clickhouse-client \
  --user zabbix \
  --password <password> \
  --query "SELECT * FROM zabbix.history_uint WHERE itemid = 42269"
```

针对不同发行版的 **Zabbix NGINX** 设置说明

RHEL

在 RHEL 8、9 和 10 上，NGINX 已包含在 AppStream 中，无需额外的软件仓库。只需启用并安装：

```
dnf module list nginx
dnf -y install nginx
```

在 RHEL 7 上，可以通过 CentOS 7 Extras 软件仓库（如果使用 CentOS）安装 NGINX：

```
yum -y install nginx
```

如果您希望使用最新的上游构建版本，请在 /etc/yum.repos.d/nginx.repo 中配置官方 NGINX 软件仓库：

```
cat > /etc/yum.repos.d/nginx.repo << 'EOF'
[nginx-stable]
name=nginx stable repo
baseurl=https://nginx.org/packages/rhel/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
EOF
dnf -y install nginx
```

SLES 15

在 SUSE Linux Enterprise Server 15 中，您需要配置 php-fpm（配置文件的路径可能会因服务包而略有不同）：

```
cp /etc/php8/fpm/php-fpm.conf{.default,}
cp /etc/php8/fpm/php-fpm.d/www.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php8/fpm/php-fpm.d/www.conf
```

使用 **Microsoft Entra ID** 配置 **SAML**

概述

本节提供了相关指南，说明如何使用 SAML 2.0 身份验证，通过 Microsoft Entra ID（原 Microsoft Azure Active Directory）为 Zabbix 配置单点登录和用户预配。

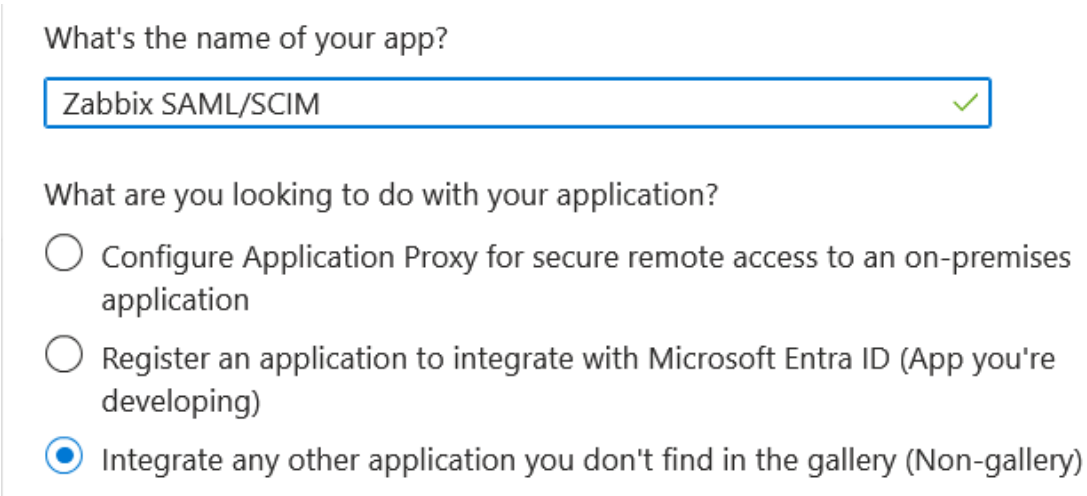
Microsoft Entra ID 配置

创建应用程序

1. 登录 Microsoft Entra ID 中的 Microsoft Entra 管理中心。出于测试目的，您可以在 Microsoft Entra ID 中创建一个免费试用帐户。

2. 在 Microsoft Entra 管理中心中，选择 *Applications* > *Enterprise applications* > *New application* > *Create your own application*。

3. 添加应用名称，并选择 *Integrate any other application...* 选项。然后，单击 *Create*。

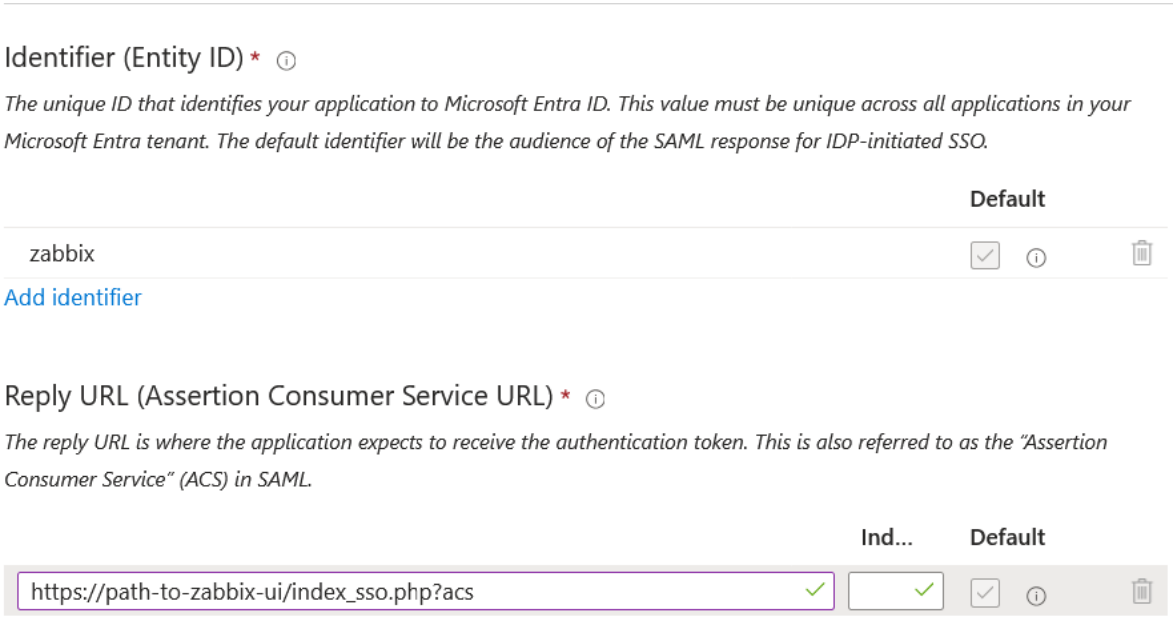


设置单点登录

1. 在应用页面中，转到 *Set up single sign on*，然后单击 *Get started*。接着选择 *SAML*。

2. 编辑 *Basic SAML Configuration*：

- 在 *Identifier (Entity ID)* 中设置一个用于在 Microsoft Entra ID 中标识应用的唯一名称，例如 zabbix；
- 在 *Reply URL (Assertion Consumer Service URL)* 中设置 Zabbix 单点登录端点：https://<path-to-zabbix-ui>/index_sso.php?ac

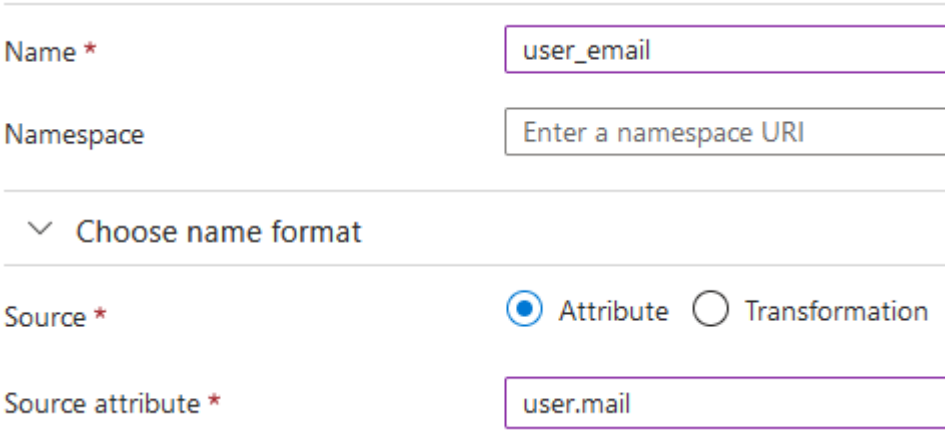


请注意，必须使用 `https`。要将其与 Zabbix 配合使用，请将以下行添加到前端配置文件中：

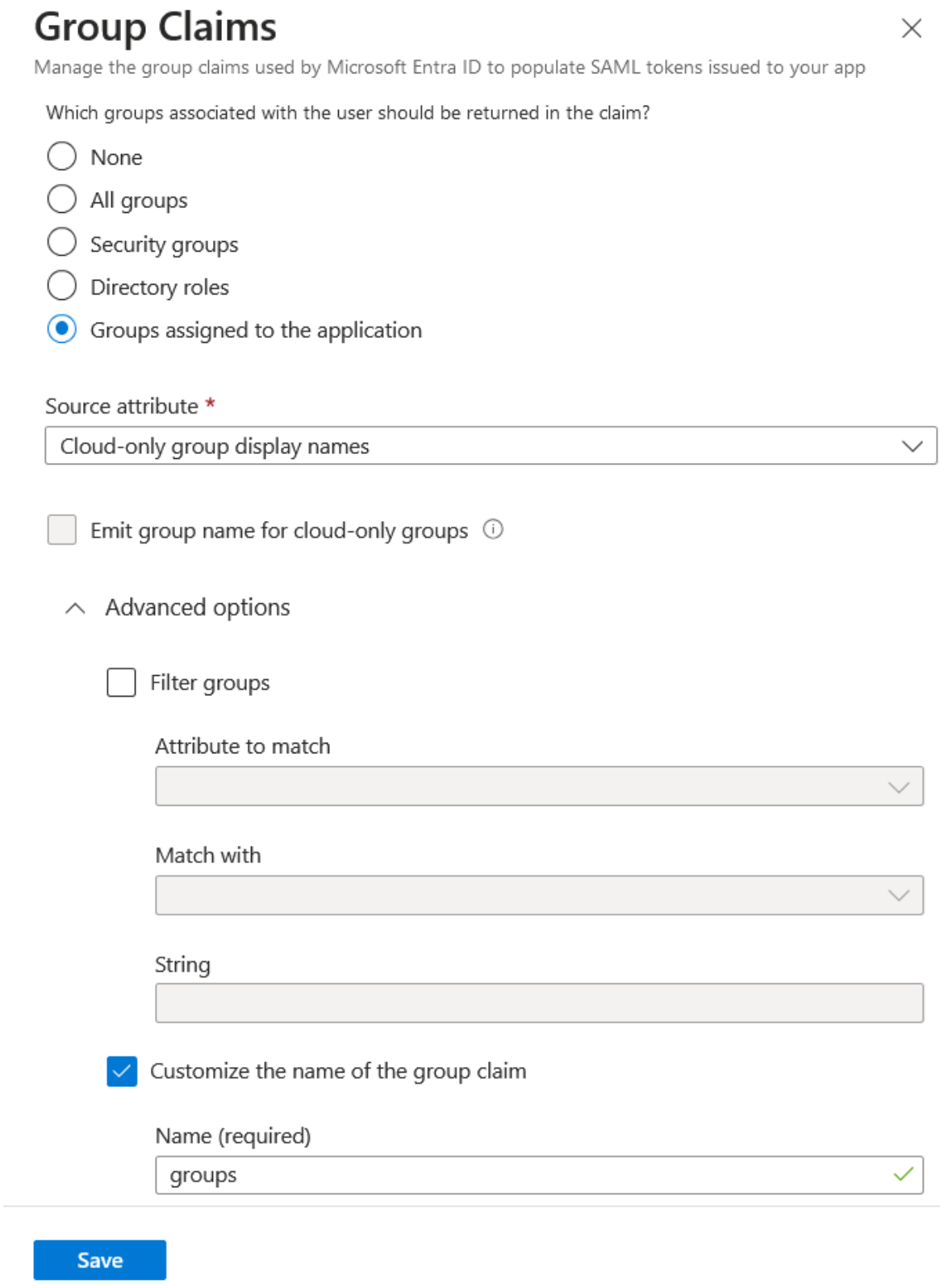
```
$SSO['SETTINGS'] = ['use_proxy_headers' => true];
```

3. 编辑 *Attributes & Claims*。您必须添加要传递给 Zabbix 的所有属性（user_name、user_lastname、user_email、user_mobile、groups）。属性名称可以任意设置。可以使用不同的属性名称，但必须与 Zabbix SAML 设置中的相应字段值匹配。

- 单击 *Add new claim* 以添加属性：



- 单击 *Add a group claim* 以添加用于向 Zabbix 传递组信息的属性：



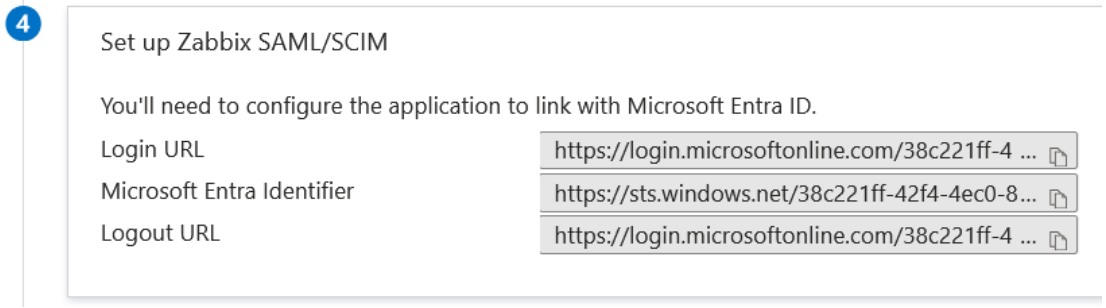
在此声明中，选定的 *Source attribute* 传递给 Zabbix 的必须是组名称（而不是组 ID）。否则，JIT 用户配置将无法正常工作。

4. 在 *SAML Certificates* 中下载 Entra ID 提供的 Base64 证书，并将其放置到 Zabbix 前端安装目录中的 `conf/certs` 目录（具体位置可能因操作系统或 Zabbix 安装来源而异）。

运行以下命令为其设置 644 权限：

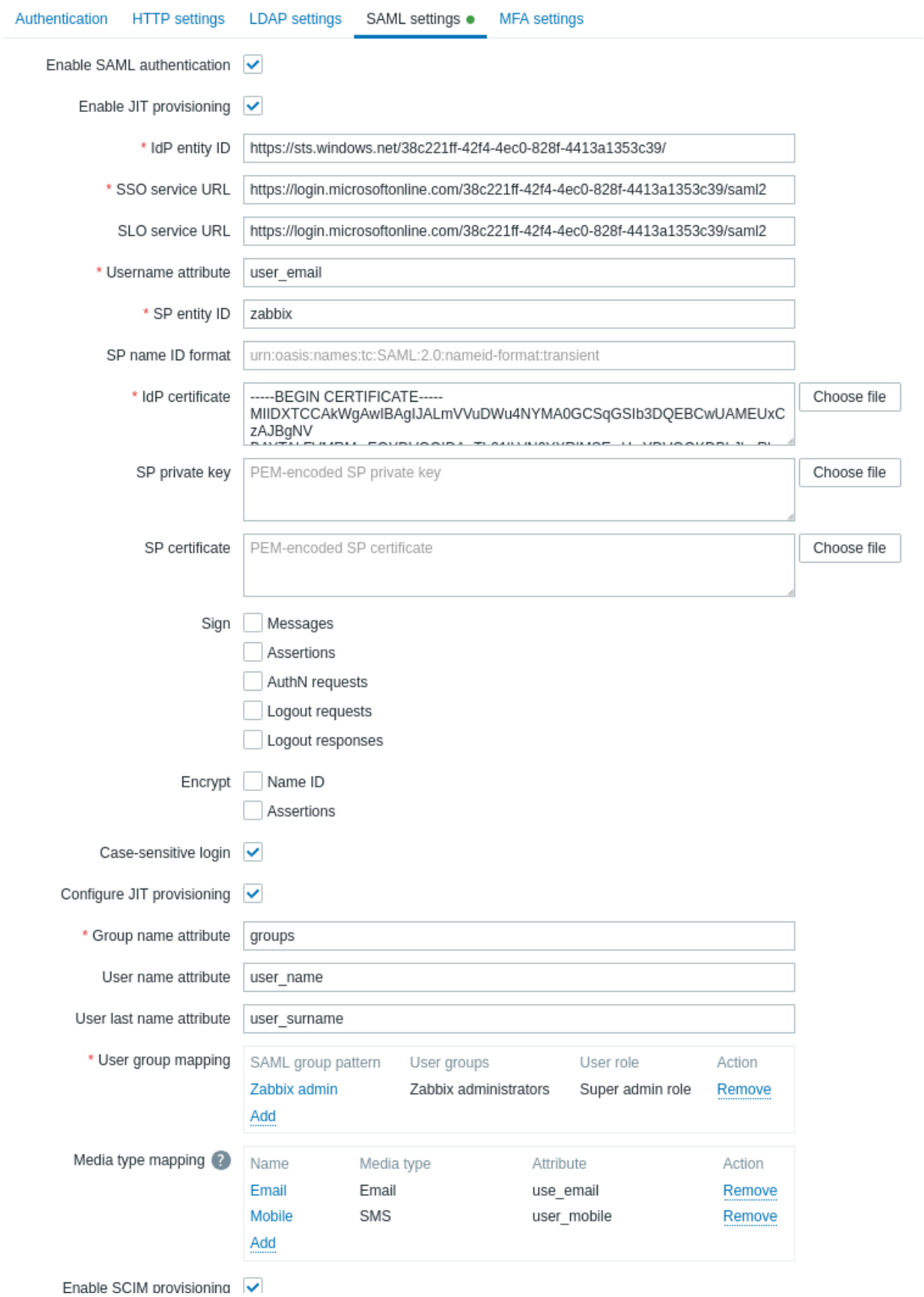
```
chmod 644 entra.cer
```

5. 使用 Entra ID 中 *Set up <your app name>* 下的值配置 Zabbix SAML 身份验证（请参阅下一节）：



Zabbix 配置

1. 在 Zabbix 中，转到SAML 设置，并根据 Entra ID 配置填写配置选项：



Zabbix 字段	Entra ID 中的设置字段	示例值
IdP 实体 *ID*	Microsoft Entra 标识符	
SSO 服务 *URL*	登录 URL	
SLO 服务 *URL*	注销 URL	
SP 实体 *ID*	标识符（实体 ID）	
用户名属性	自定义属性（声明）	`user_email`
组名称属性	自定义属性（声明）	`groups`
用户姓名属性	自定义属性（声明）	`user_name`

Zabbix 字段	Entra ID 中的设置字段	示例值
用户姓氏属性	自定义属性（声明）	`user_lastname`

2. 添加 Entra ID 提供的 Base64 证书。

如果在前端配置文件（`zabbix.conf.php`）中设置了 `$SSO['CERT_STORAGE'] = 'database'`，则可以在 SAML 配置期间将证书文本粘贴到前端，或上传证书文件——文件系统中无需存储证书文件。

如果在配置文件中设置了 `$SSO['CERT_STORAGE'] = 'file'`，则必须将证书存储在文件系统中（默认位于 `ui/conf/certs`，或位于配置文件中配置的路径），并且前端不会将证书存储在数据库中。请注意，如果未设置 `$SSO['CERT_STORAGE']` 或该设置已被注释，则默认使用文件存储，并从 `ui/conf/certs` 读取证书。

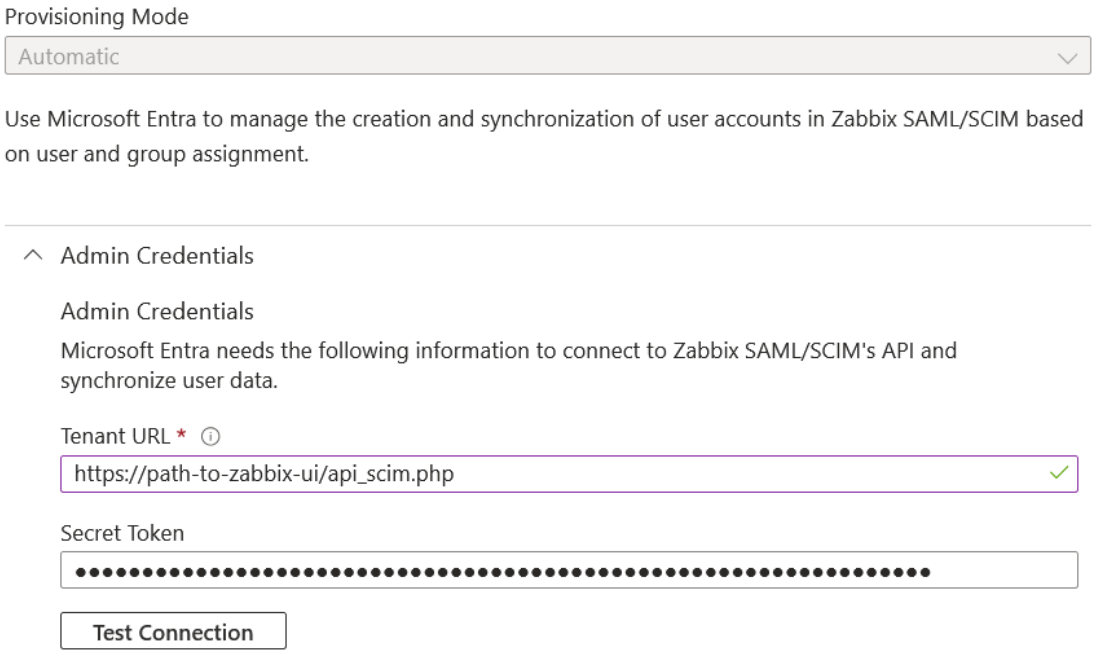
此外，还必须配置用户组映射。媒体映射为可选配置。

3. 单击 更新按钮保存这些设置。

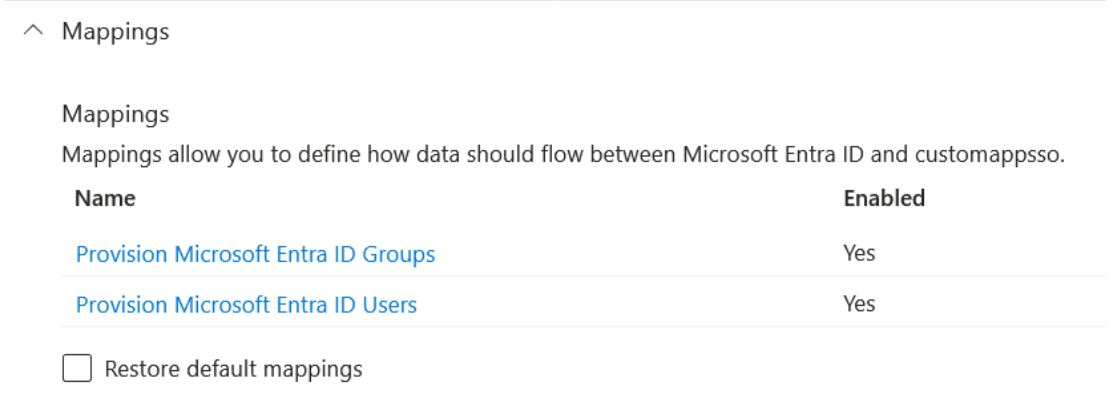
SCIM 用户预配

1. 在你的 Entra ID 应用页面中，从主菜单打开 Provisioning 页面。
点击 *Get started*，然后选择 Automatic provisioning mode：

- 在 *Tenant URL* 中，设置以下值：`https://<path-to-zabbix-ui>/api_scim.php`
- 在 *Secret token* 中，输入具有 Super admin 权限的 Zabbix API token。
- 点击 *Test connection*，查看连接是否已建立。

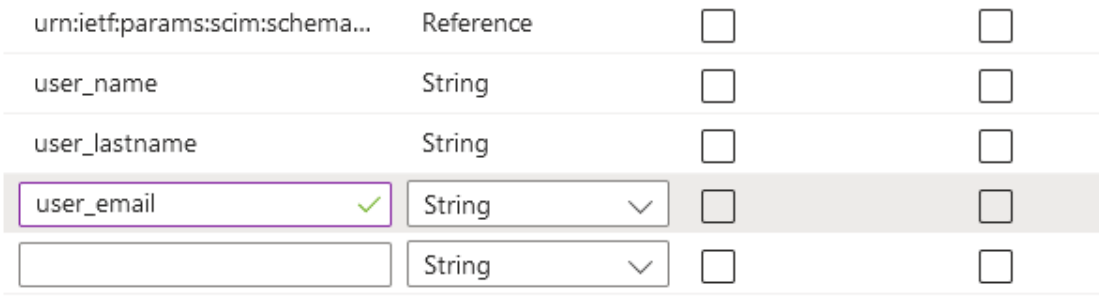


2. 现在你可以添加所有将通过 SCIM 传递给 Zabbix 的属性。
为此，点击 *Mappings*，然后点击 *Provision Microsoft Entra ID Users*。



在 Attribute Mapping 列表底部，启用 *Show advanced options*，然后点击 *Edit attribute list for customappsso*。

在属性列表底部，添加你自己的属性，类型为'String'：



保存该列表。

3. 现在你可以为已添加的属性添加映射。
在 Attribute Mapping 列表底部，点击 *Add New Mapping*，并按如下所示创建映射：



当所有映射都添加完成后，保存映射列表。



4. 作为将用户预配到 Zabbix 的前提条件，你必须先在 Entra ID 中配置用户和组。

为此，转到 *Microsoft Entra admin center*，然后在相应的 Users 和 Groups 页面中添加用户/组。

5. 当用户和组已在 Entra ID 中创建后，你可以转到应用的 *Users and groups* 菜单，并将它们添加到该应用中。

6. 转到应用的 *Provisioning* 菜单，然后点击 *Start provisioning*，即可将用户预配到 Zabbix。

请注意，Entra ID 中的 Users PATCH 请求不支持媒体中的更改。

身份验证请求签名

可以将 Entra ID 配置为验证已签名身份验证请求的签名。

要实现此功能，请创建公钥/私钥：

```
openssl req -x509 -newkey rsa:4096 -keyout /usr/share/zabbix/conf/certs/request-sign.key -out /usr/share/z
```

分配权限：

```
chown apache /usr/share/zabbix/conf/certs/request-sign.key
chmod 400 /usr/share/zabbix/conf/certs/request-sign.key
```

通过添加以下内容，更新 Zabbix 前端配置：

```
$SSO['SP_KEY'] = 'conf/certs/request-sign.key';
$SSO['SP_CERT'] = 'conf/certs/request-sign.crt';
```

故障排除

当用户尝试通过 SAML 登录 Zabbix，且已经登录 MicroSoft Edge 配置文件时，使用 MicroSoft Edge 浏览器可能会出现身份验证问题。出现此类问题时，用户可能可以在 MicroSoft Edge 的隐私模式下登录 Zabbix。

为避免此情况下的身份验证问题，可能需要在 Zabbix 前端配置文件中将 `requestedAuthnContext` 设置为 `false`。

```
$SSO['SETTINGS'] = [
    'security' => [
        'requestedAuthnContext' => false
    ]
];
```

使用 **Okta** 配置 **SAML**

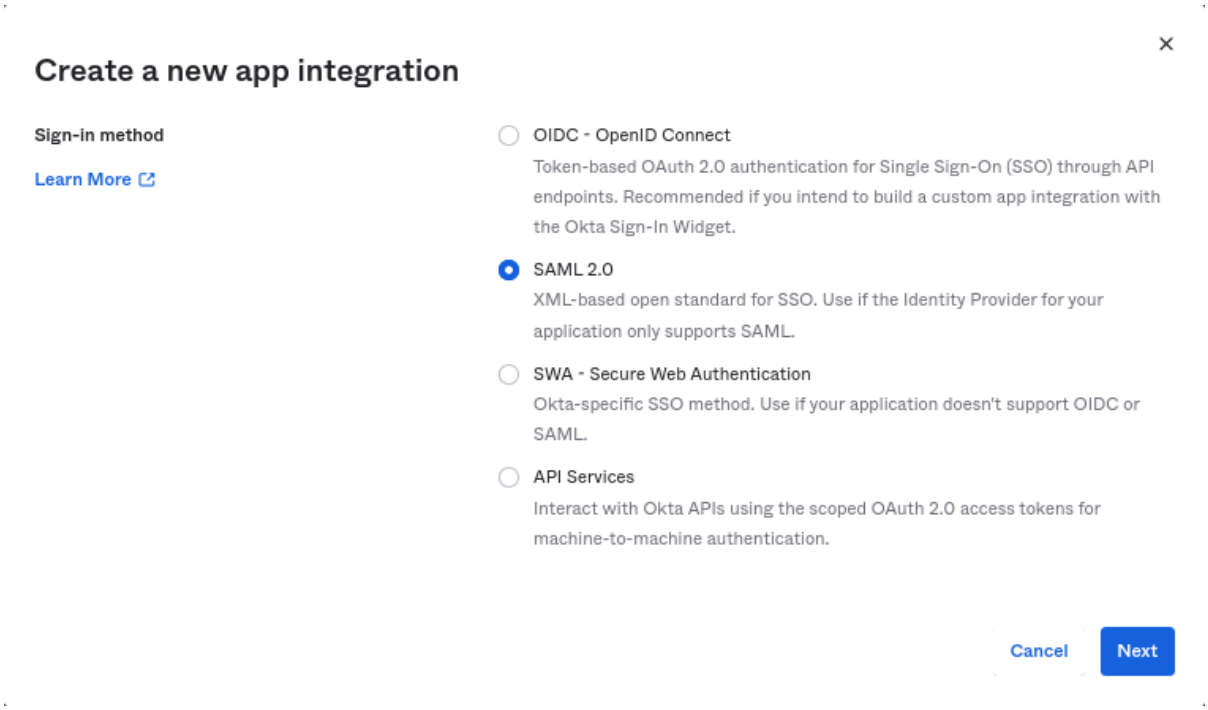
本节提供有关配置 Okta 的指南，以便为 Zabbix 启用 SAML 2.0 身份验证和用户预配。

Okta 配置

1. 访问 https://developer.okta.com/signup/ 并注册/登录您的账户。

2. 在 Okta web 界面中，导航到 *Applications* > *Applications*。

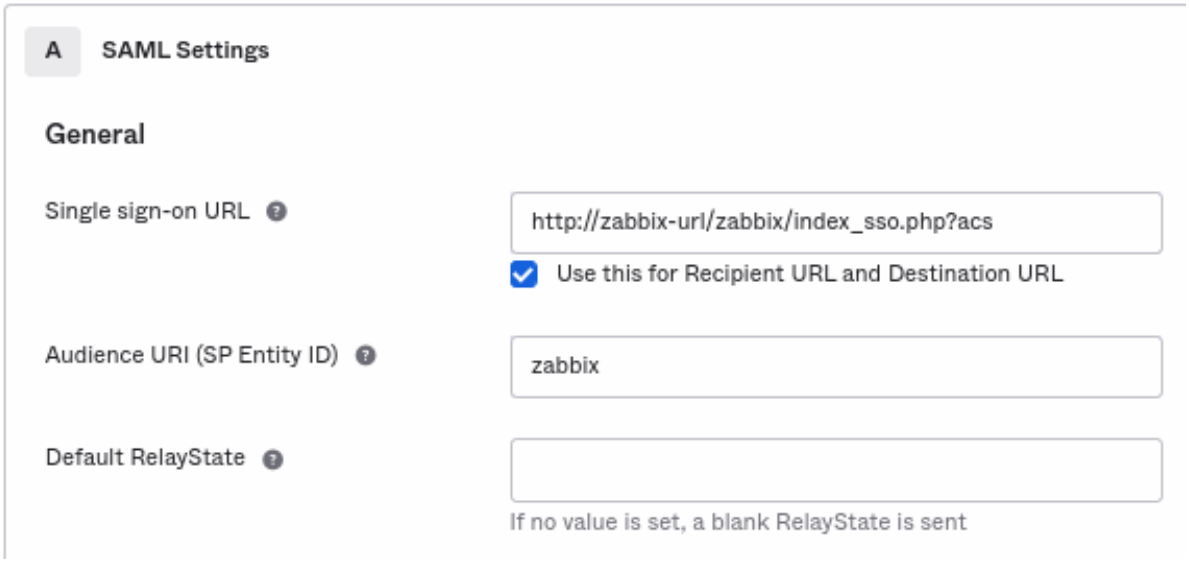
3. 点击 *Create App Integration*。



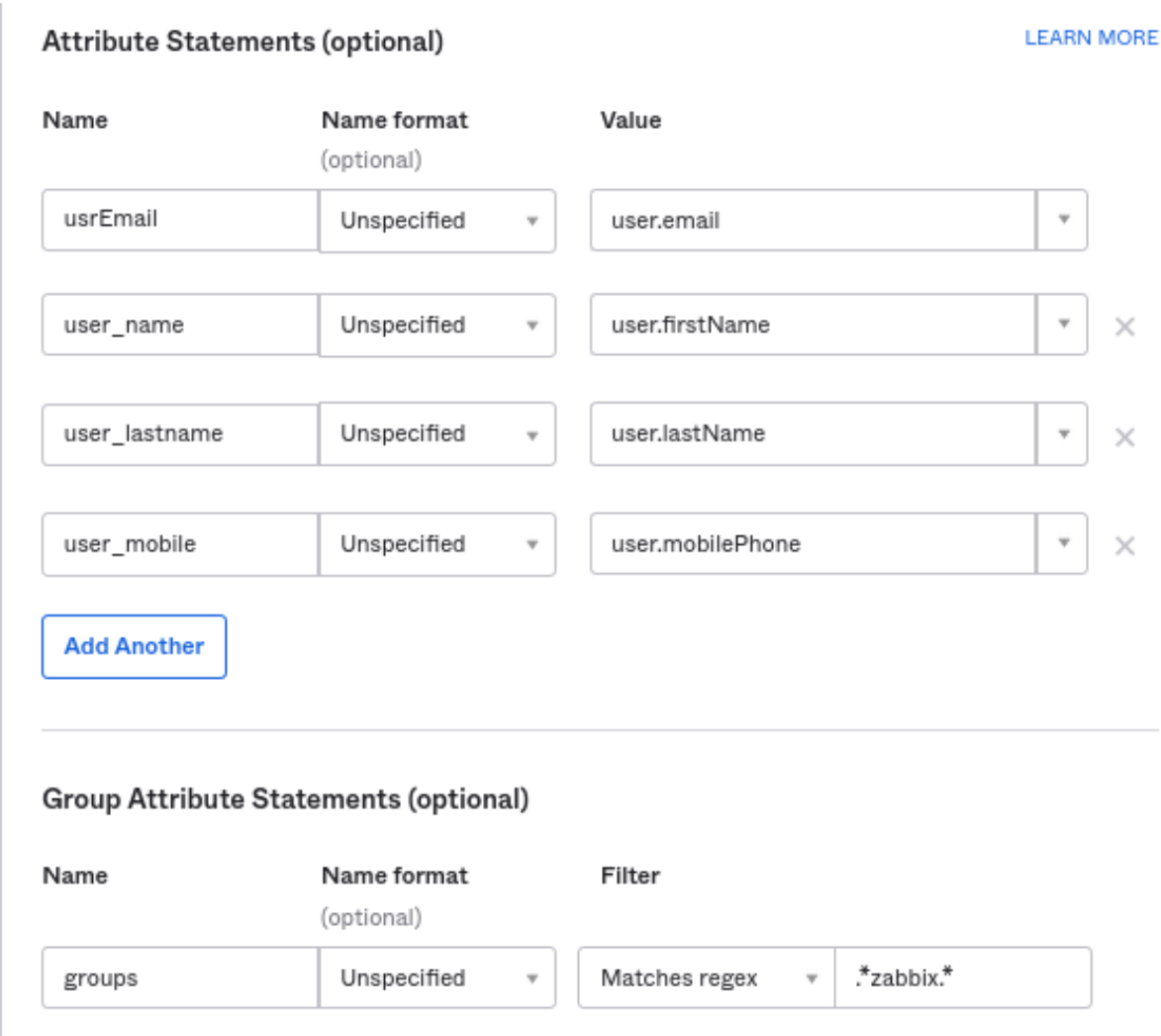
将登录方法选择为”SAML 2.0”，然后点击 *Next*。

4. 在常规设置中，填写应用名称并点击 *Next*。

5. 在 SAML 配置中，输入下面提供的值，然后点击 *Next*。



- 在 **General** 中添加：
  - *Single sign-on URL*: `http://<your-zabbix-url>/zabbix/index_sso.php?acs<br>` 请注意这里使用的是"http"，而不是"https"，这样请求中的 `acs` 参数不会被截断。还应勾选 *Use this for Recipient URL and Destination URL* 复选框。
  - *Audience URI (SP Entity ID)*: `zabbix<br>` 请注意，此值将在 SAML 断言中用作唯一的服务提供商标识符（如果不匹配，操作将被拒绝）。此字段中可以指定 URL 或任意字符串数据。
  - *Default RelayState*:
 将此字段留空；如果需要自定义重定向，可以在 Zabbix 的 *Users* > *Users* 设置中添加。
  - 其他字段可根据您的偏好填写。
- 在 **Attribute Statements/Group Attribute Statements** 中添加：



这些属性语句会被插入到与 Zabbix 共享的 SAML 断言中。

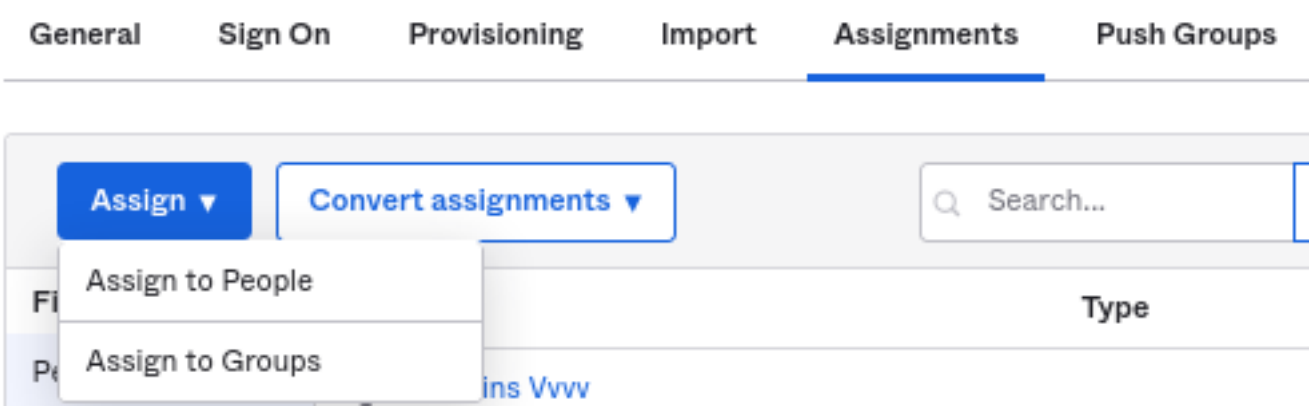
此处使用的属性名称仅为任意示例。您可以使用不同的属性名称，但必须确保它们与 Zabbix SAML 设置中相应字段的值匹配。

如果您想在 不使用 JIT 用户预配的情况下为 Zabbix 配置 SAML 登录，那么只需要 email 属性。

Note:
如果计划使用加密连接，请生成私钥和公钥加密证书，然后将公钥证书上传到 Okta。当 *Assertion Encryption* 设置为”Encrypted” 时，会显示证书上传表单（点击 *Show Advanced Settings* 可找到此参数）。

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

7. 进入新创建应用的”Assignments” 选项卡，点击 *Assign* 按钮，然后在下拉菜单中选择 *Assign to People*。



8. 在弹出的窗口中，将应用分配给将使用 SAML 2.0 通过 Zabbix 进行身份验证的用户，然后点击 *Save and go back*。

9. 下载 IdP 证书。通过运行以下命令将其权限设置为 644：

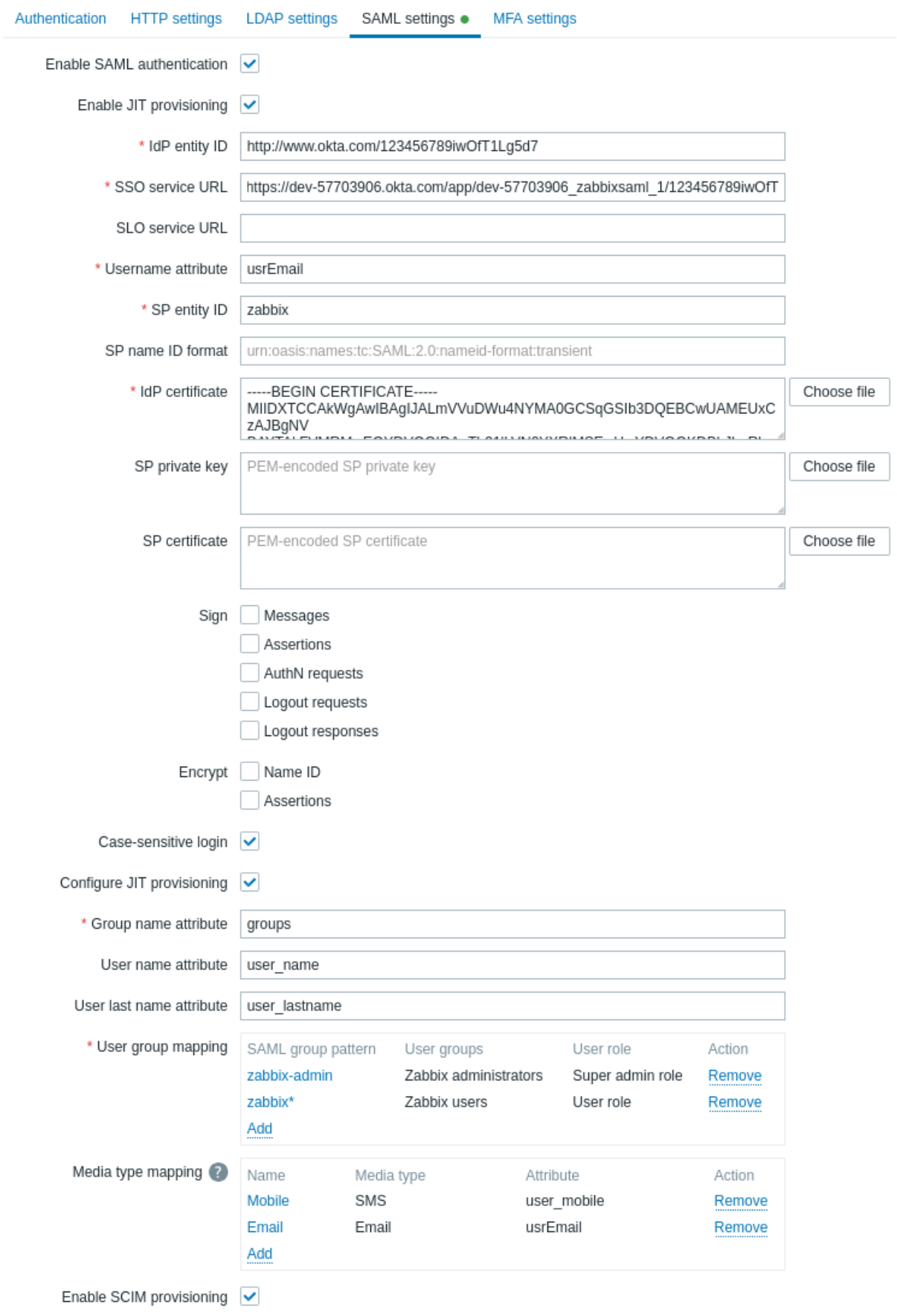
```
chmod 644 idp.crt
```

10. 进入”Sign On” 选项卡并点击 *View Setup Instructions* 按钮。

设置说明将在新选项卡中打开；在配置 Zabbix 时请保持此选项卡打开。

Zabbix 配置

1. 在 Zabbix 中，转到SAML 设置，并根据 Okta 的设置说明填写配置选项：



Zabbix 字段	Okta 中的设置字段	示例值
IdP 实体 *ID*	Identity Provider Issuer	
SSO 服务 *URL*	Identity Provider Single Sign-On URL	
用户名属性	Attribute name	`usrEmail`
SP 实体 *ID*	Audience URI	`zabbix`
组名称属性	Attribute name	`groups`

Zabbix 字段	Okta 中的设置字段	示例值
用户名称属性	Attribute name	`user_name`
用户姓氏属性	Attribute name	`user_lastname`

还需要配置用户组和媒体映射。

2. 添加 Okta SAML 设置说明中提供的 Base64 证书。

如果在前端配置文件（`zabbix.conf.php`）中设置了 `$SSO['CERT_STORAGE'] = 'database'`，则可以在 SAML 配置期间将证书文本粘贴到前端，或上传证书文件，无需在文件系统中存储文件。

如果在配置文件中设置了 `$SSO['CERT_STORAGE'] = 'file'`，则必须在文件系统中提供证书（默认位于 `ui/conf/certs`，或位于配置文件中配置的路径），并且前端不会将证书存储在数据库中。请注意，如果未设置 `$SSO['CERT_STORAGE']` 或该设置已被注释，则默认使用文件存储，并从 `ui/conf/certs` 读取证书。

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

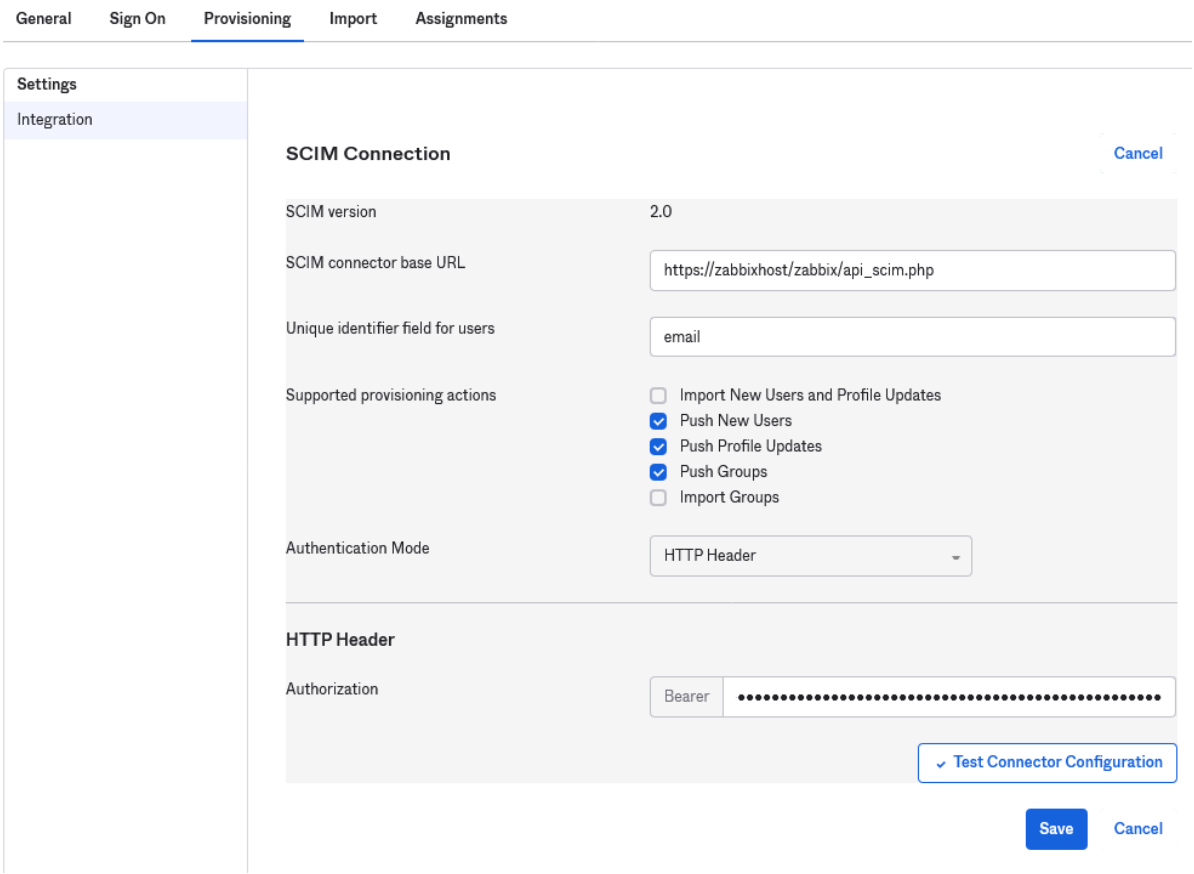
4. 单击 *Update* 按钮保存这些设置。

SCIM 配置

1. 要启用 SCIM 配置，请在 Okta 中进入应用的 *General* > *App Settings*。

勾选 *Enable SCIM provisioning* 复选框。
完成后，会出现新的 *Provisioning* 选项卡。

2. 进入 *Provisioning* 选项卡，设置 SCIM 连接：- 在 *SCIM connector base URL* 中，指定 Zabbix 前端的路径，并在末尾附加 `api_scim.php`，即：
 `https://<your-zabbix-url>/zabbix/api_scim.php` - *Unique identifier field for users*：`email` - *Authentication mode*：`HTTP header` - 在 *Authorization* 中，输入具有 Super admin 权限的有效 API token。



Attention:
如果遇到身份验证问题，请参阅转发 Authorization 标头。

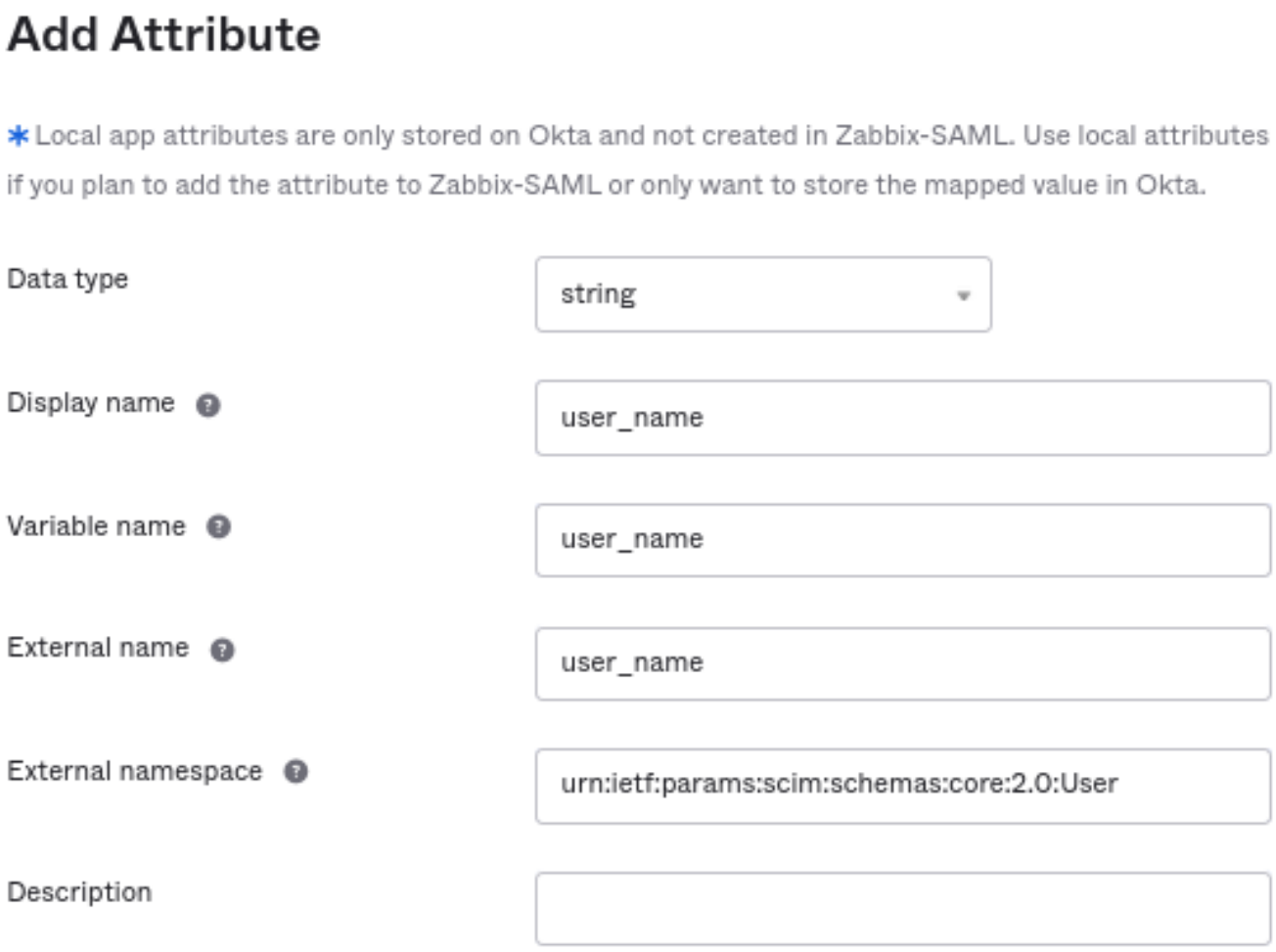
3. 点击 *Test Connector Configuration* 测试连接。
如果一切正常，将显示成功消息。

4. 在 *Provisioning* > *To App* 中，确保勾选以下复选框：- Create Users - Update User Attributes - Deactivate Users

这样可以确保这些请求类型会发送到 Zabbix。

5. 确保 SAML 中定义的所有属性也已在 SCIM 中定义。
在 *Provisioning* > *To App* 中，点击 *Go to Profile Editor*，即可访问应用的配置文件编辑器。

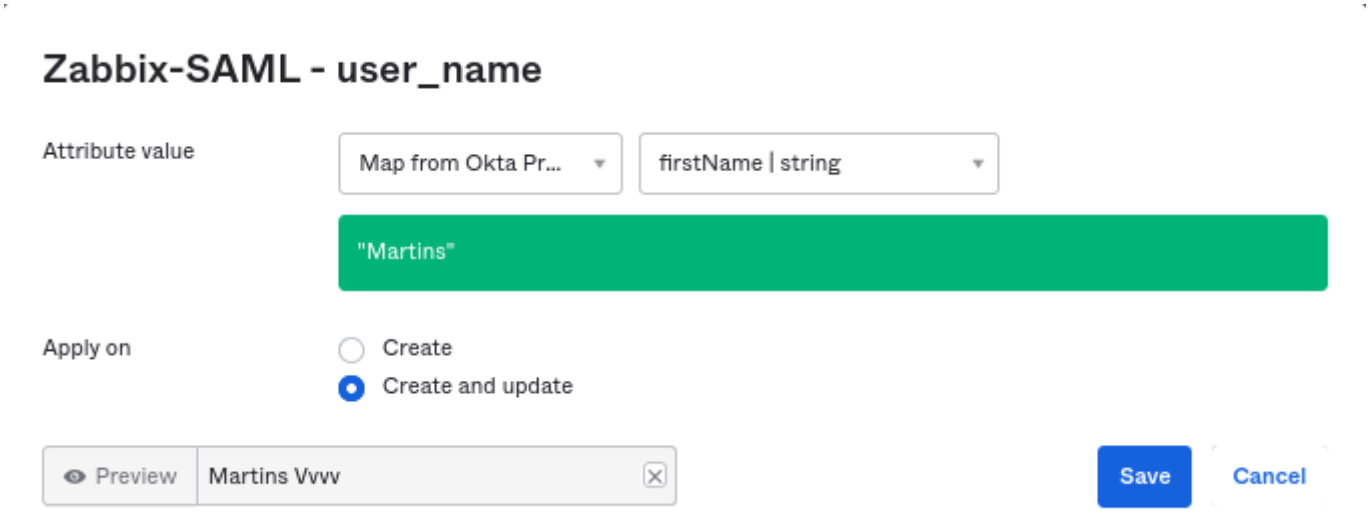
点击 *Add Attribute*。
为 *Display name*、*Variable name* 和 *External name* 填写 SAML 属性名称，例如 `user_name`。



External namespace 应与用户架构相同：`urn:ietf:params:scim:schemas:core:2.0:User`

6. 进入应用的 *Provisioning* > *To App* > *Attribute Mappings*。
点击底部的 *Show Unmapped Attributes*。
此时会显示新添加的属性。

7. 映射每个添加的属性。



8. 在”Assignments” 选项卡中添加用户。

需要先在 *Directory* > *People* 中添加这些用户。
所有这些分配都会作为请求发送到 Zabbix。

9. 在"Push Groups" 选项卡中添加组。
Zabbix SAML 设置中的用户组映射模式必须与此处指定的组匹配。
如果不匹配，则无法在 Zabbix 中创建用户。

每次发生更改时，都会发送有关组成员的信息。

使用 **OneLogin** 进行 **SAML** 设置

概述

本节提供使用 SAML 2.0 身份验证从 OneLogin 配置单点登录和 Zabbix 用户配置的指南。

OneLogin 配置

创建应用

1. 登录到您的 OneLogin 账户。为了进行测试，您可以在 OneLogin 中创建一个免费的开发者账户。

2. 在 OneLogin 前端中，导航到 *Applications* > *Applications*。

3. 单击"Add App" 并搜索合适的应用。本页中的指南基于 *SCIM Provisioner with SAML (SCIM v2 Enterprise, full SAML)* 应用示例。

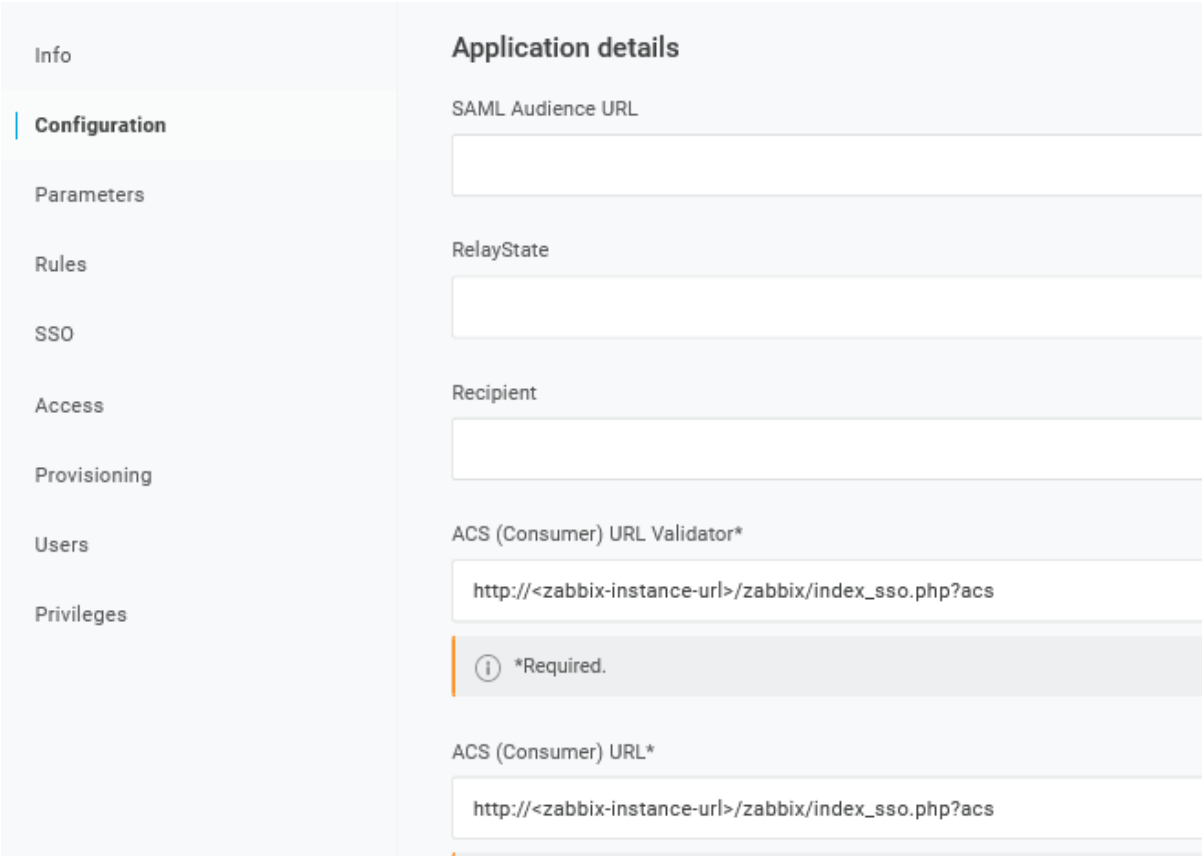
4. 首先，您可能需要自定义应用的显示名称。您也可能需要添加图标和应用详细信息。然后，单击 *Save*。

设置 SSO/SCIM 配置

1. 在 *Configuration* > *Application details* 中，将 Zabbix 单点登录端点 `http://<zabbix-instance-url>/zabbix/index_sso.php?acs` 设置为以下字段的值：

- *ACS (Consumer) URL Validator*
- *ACS (Consumer) URL*

请注意使用的是"http"，而不是"https"，这样请求中的 `acs` 参数才不会被截掉。



也可以使用"https"。要将其与 Zabbix 配合使用，请将以下行添加到前端配置文件（`zabbix.conf.php`）中：

```
$SSO['SETTINGS'] = ['use_proxy_headers' => true];
```

其他选项保留默认值。

2. 在 *Configuration* > *API connection* 中，设置以下值：

- *SCIM Base URL*：`https://<zabbix-instance-url>/zabbix/api_scim.php`
- *SCIM JSON Template*：应包含您希望通过 SCIM 传递给 Zabbix 的所有自定义属性，例如 `user_name`、`user_lastname`、`user_email` 和 `user_mobile`：

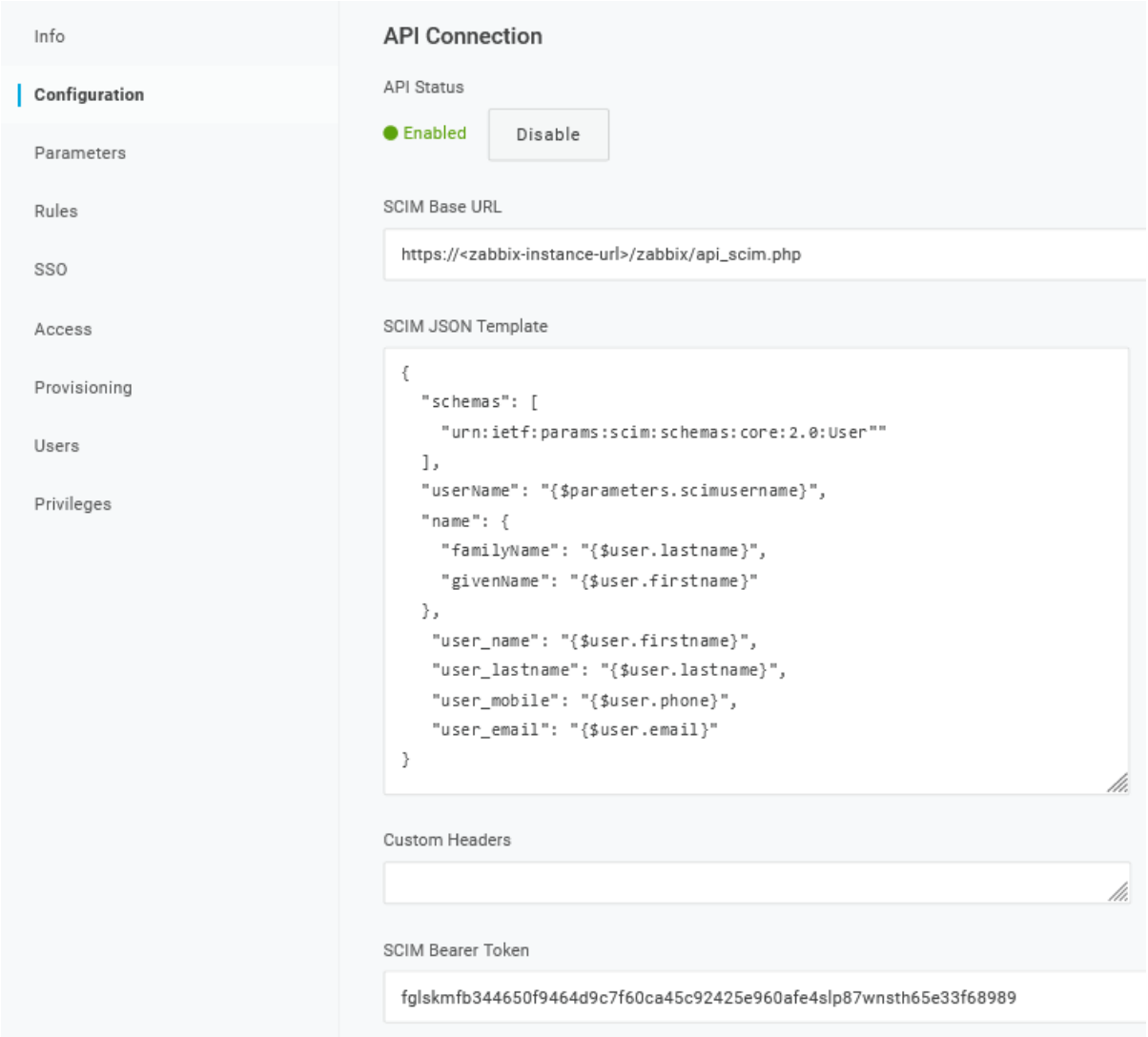
```
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
  ],
  "userName": "{$parameters.scimusername}",
  "name": {
    "familyName": "{$user.lastname}",
    "givenName": "{$user.firstname}"
  },
   "user_name": "{$user.firstname}",
   "user_lastname": "{$user.lastname}",
   "user_mobile": "{$user.phone}",
   "user_email": "{$user.email}"
}
```

属性名称可以任意指定。可以使用不同的属性名称，但必须与 Zabbix SAML 设置中相应字段的值匹配。

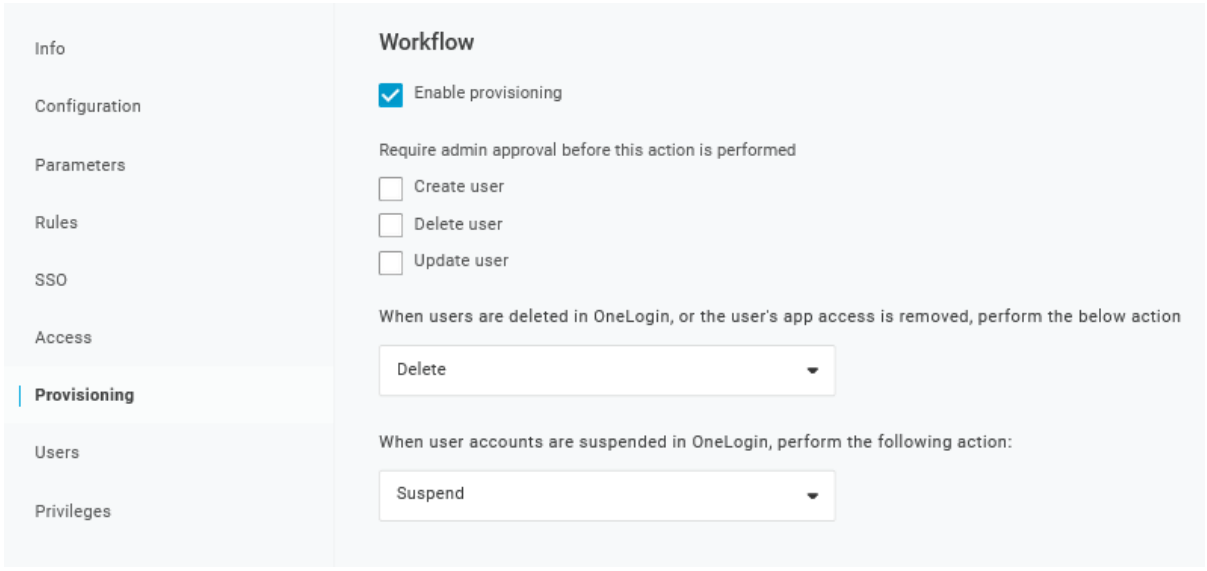
请注意，要使用户配置正常工作，OneLogin 需要在响应中接收包含 `givenName` 和 `familyName` 的 `name` 属性，即使服务提供商未要求该属性。因此，必须在应用程序配置部分的架构中指定该属性。

- *SCIM Bearer Token*：输入具有 Super admin 权限的 Zabbix API token。

单击 *Enable* 以激活连接。

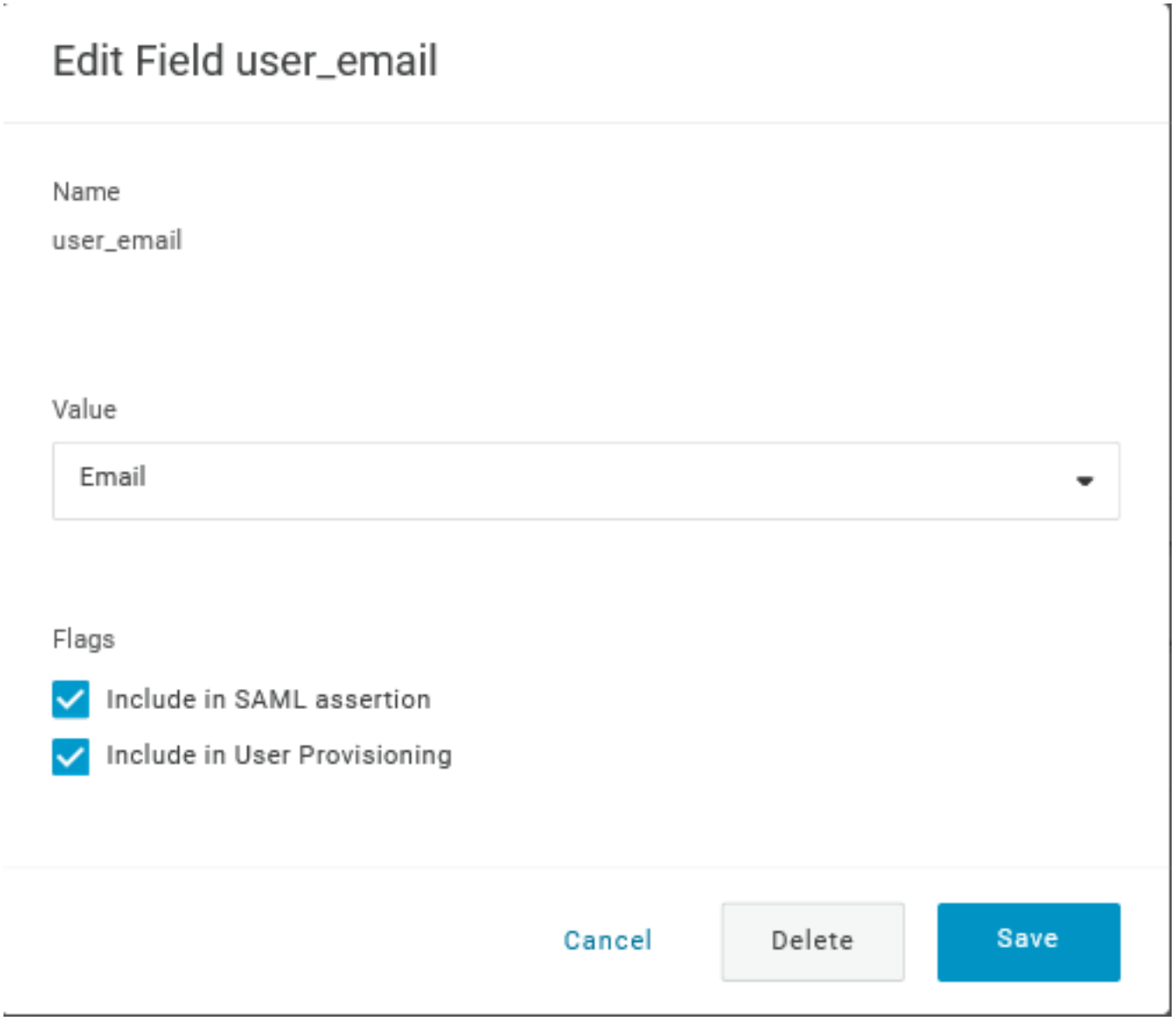


3. 在 *Provisioning* 页面中，启用 Provisioning 选项：



4. *Parameters* 页面包含默认参数列表：

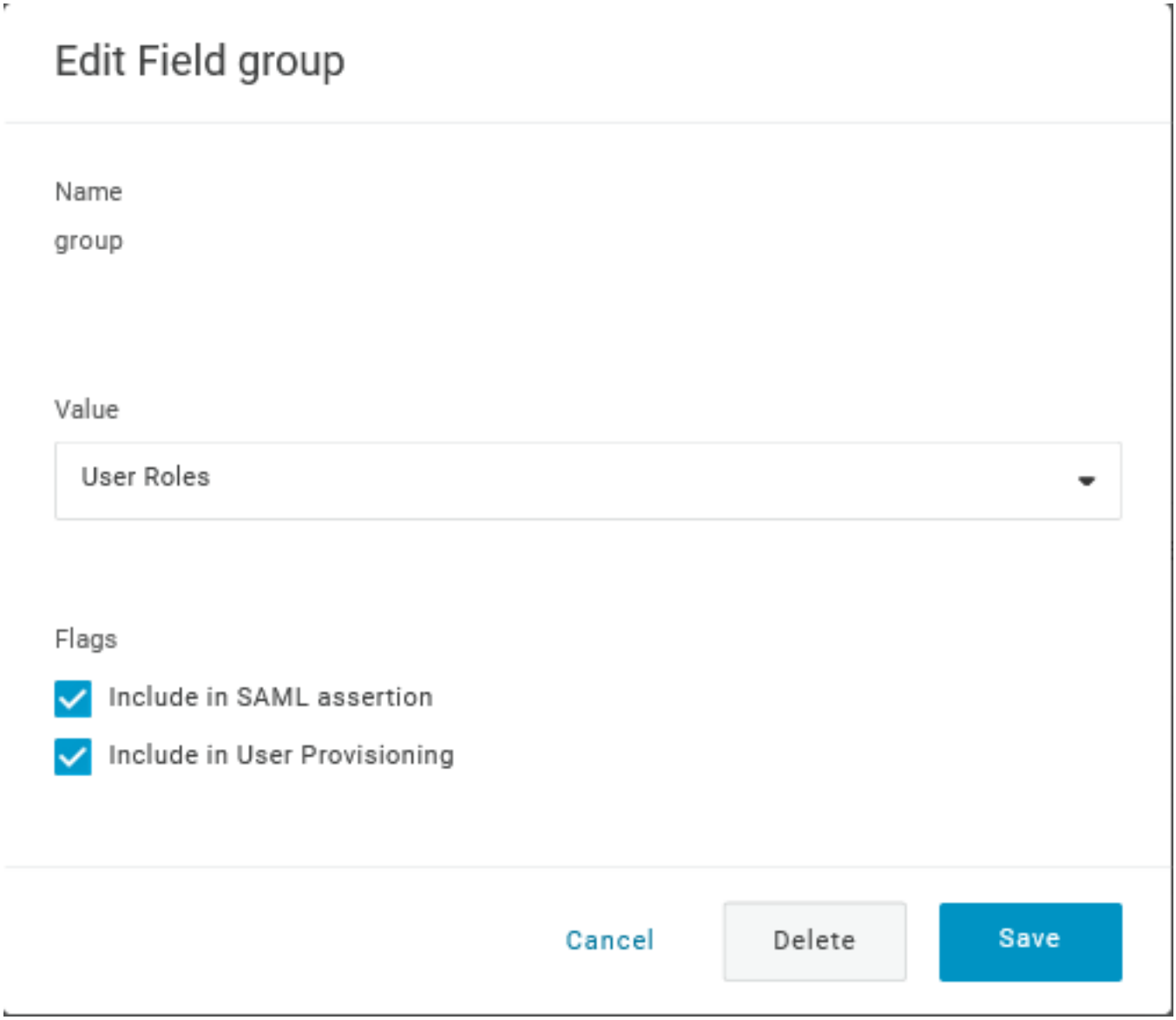
- 确保'scimusername' 与 OneLogin 中的用户登录值匹配（例如电子邮件地址）；
- 为'Groups' 参数勾选 *Include in User Provisioning* 选项；
- 单击"+" 创建 SAML 断言和用户配置所需的自定义参数，例如 `user_name`、`user_lastname`、`user_email` 和 `user_mobile`：



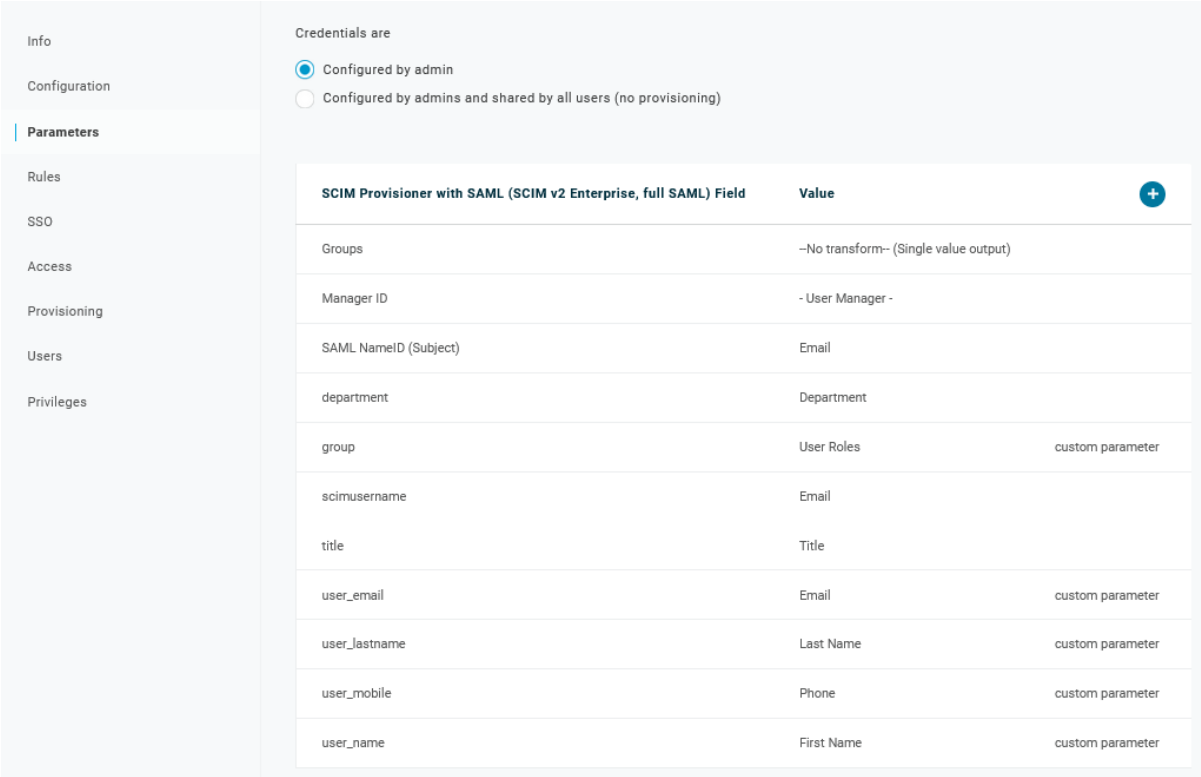
添加参数时，请确保同时勾选 *Include in SAML assertion* 和 *Include in User Provisioning* 选项。

- 添加与 OneLogin 中用户角色匹配的'group' 参数。用户角色将作为字符串传递，并使用分号；分隔。OneLogin 用户角色将用于在

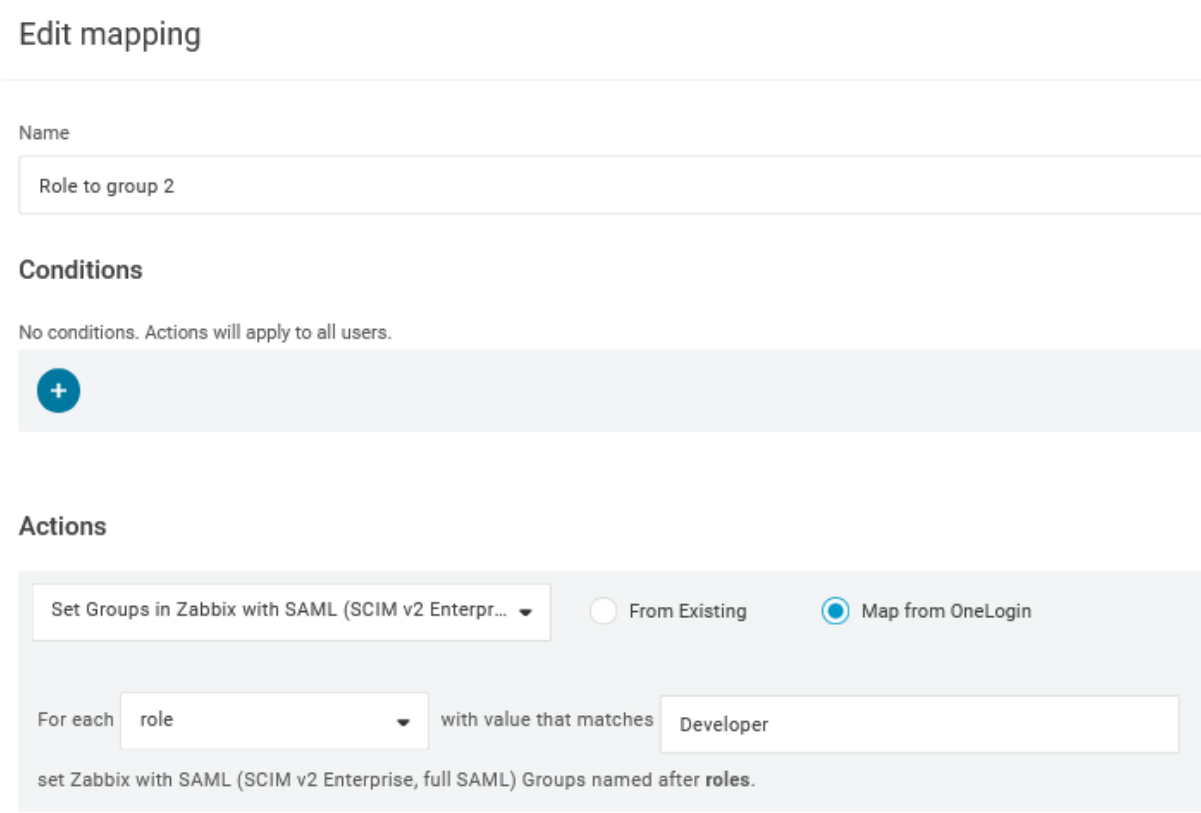
Zabbix 中创建用户组：



验证参数列表：



5. 在 *Rules* 页面中，为默认的 Groups 参数创建用户角色映射。



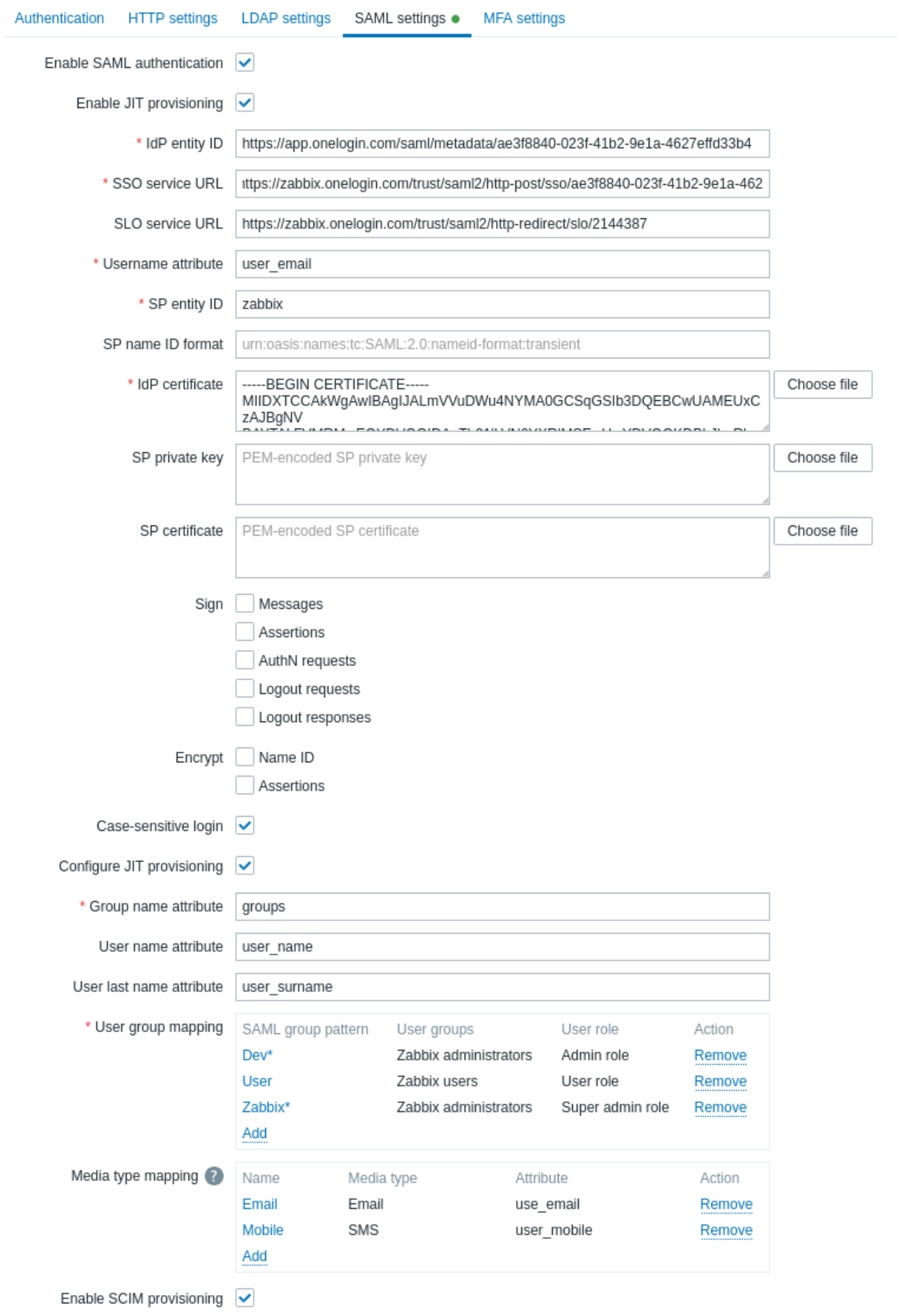
您可以使用正则表达式将特定角色作为用户组传递。角色名称不应包含 ；，因为 OneLogin 在发送包含多个角色的属性时会将其用作分隔符。

6. 下载 IdP 证书。运行以下命令将其权限设置为 644：

```
chmod 644 idp.crt
```

Zabbix 配置

1. 在 Zabbix 中，转到SAML 设置，并根据 OneLogin 配置填写配置选项：



Zabbix 字段	OneLogin 中的设置字段	示例值
IdP 实体 *ID*	Issuer URL （请参见 OneLogin 中应用的 SSO 选项卡）	
SSO 服务 *URL*	SAML 2.0 Endpoint (HTTP) （请参见 OneLogin 中应用的 SSO 选项卡）	

Zabbix 字段	OneLogin 中的设置字段	示例值
SLO 服务 URL	SLO Endpoint (HTTP) （请参见 OneLogin 中应用的 SSO 选项卡）	
用户名属性	Custom parameter	`user_email`
组名称属性	Custom parameter	`group`
用户名字属性	Custom parameter	`user_name`
用户姓氏属性	Custom parameter	`user_lastname`

此外，还必须配置用户组映射。媒体映射是可选的。点击 更新以保存这些设置。

2. 添加 OneLogin 提供的 Base64 证书。

如果在前端配置文件（`zabbix.conf.php`）中设置了 `$SSO['CERT_STORAGE'] = 'database'`，则可以在 SAML 配置期间将证书文本粘贴到前端，或上传证书文件，无需在文件系统中存储文件。

如果在配置文件中设置了 `$SSO['CERT_STORAGE'] = 'file'`，则必须使证书在文件系统中可用（默认位于 `ui/conf/certs`，或位于配置文件中配置的路径），并且前端不会将证书存储在数据库中。请注意，如果未设置 `$SSO['CERT_STORAGE']` 或该设置已被注释掉，则默认使用文件存储，并从 `ui/conf/certs` 读取证书。

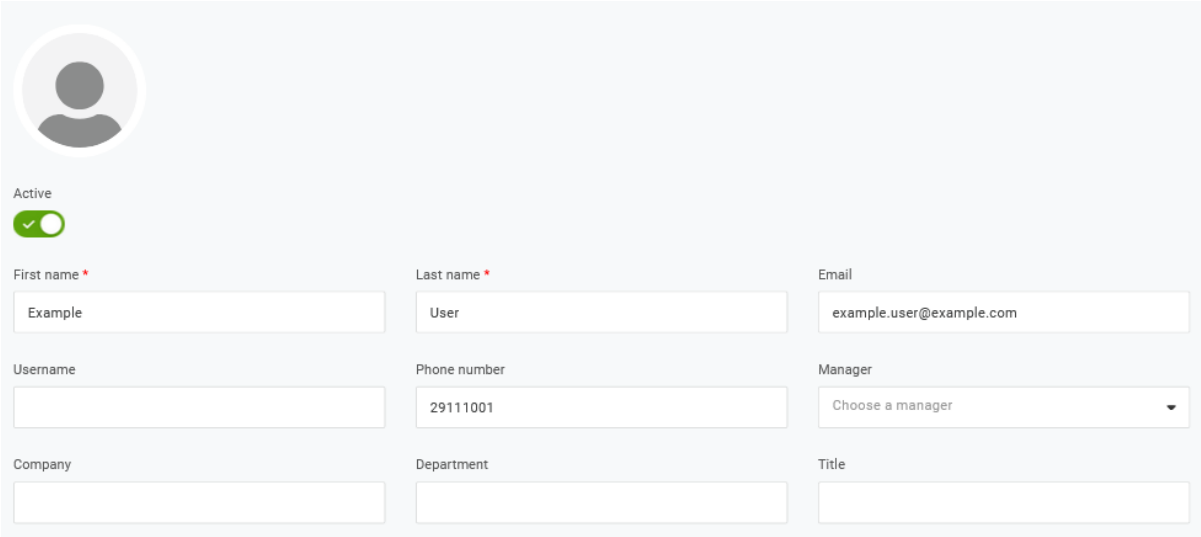
在 OneLogin 中，选择 *Applications* > *SSO*，然后在当前证书下点击 *View details*，即可访问证书下载。

3. 点击 更新按钮以保存这些设置。

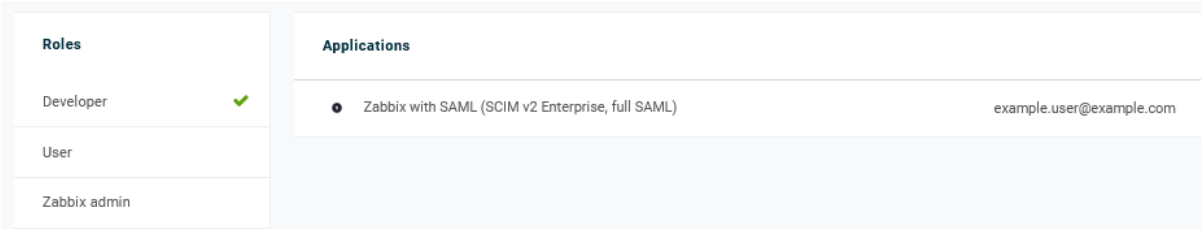
SCIM 用户预配

启用用户预配后，现在可以在 OneLogin 中添加/更新用户及其角色，并将其立即预配到 Zabbix。

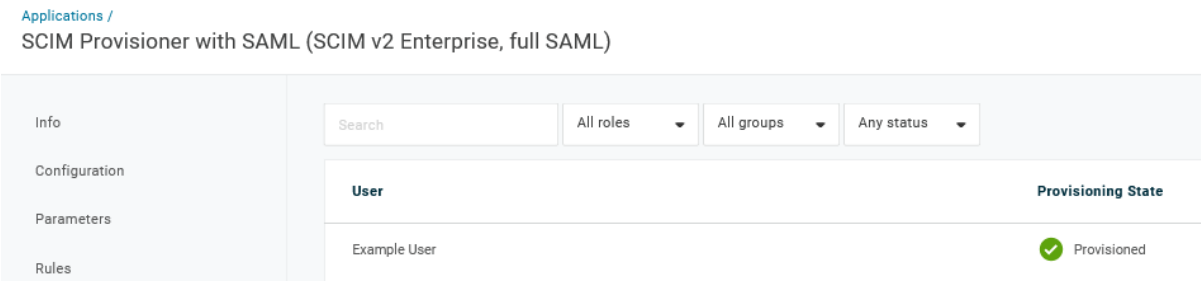
例如，您可以创建一个新用户：



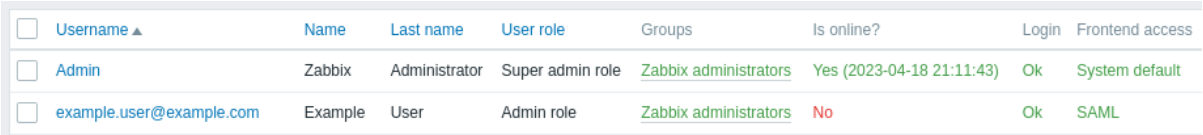
将其添加到用户角色以及将为该用户执行预配的应用程序中：



保存用户后，它将被预配到 Zabbix。在 *Application* > *Users* 中，您可以检查当前应用程序用户的预配状态：



如果预配成功，该用户将显示在 Zabbix 用户列表中。



Username ▲	Name	Last name	User role	Groups	Is online?	Login	Frontend access
Admin	Zabbix	Administrator	Super admin role	Zabbix administrators	Yes (2023-04-18 21:11:43)	Ok	System default
example.user@example.com	Example	User	Admin role	Zabbix administrators	No	Ok	SAML

设置计划报告

概述

本节提供有关安装 Zabbix Web 服务和配置 Zabbix 以启用计划报告 生成的说明。

安装

应安装新的Zabbix Web 服务 进程和Google Chrome 浏览器，以启用计划报表的生成功能。
Web 服务可以安装在安装了 Zabbix 服务器的同一台机器上，也可以安装在不同的机器上。
Google Chrome 浏览器应安装在安装了 Web 服务的同一台机器上。

官方 zabbix-web-service 软件包可从 Zabbix 仓库 获取。
Google Chrome 浏览器不包含在这些软件包中，必须单独安装。

要从源代码编译 Zabbix web service，请参见安装 Zabbix web service。

安装完成后，在安装了 web service 的机器上运行 `zabbix_web_service`：

```
zabbix_web_service
```

配置

为确保所有相关元素之间能够正常通信，请确保正确配置服务器配置和前端配置参数。

Zabbix 服务器

需要更新 Zabbix 服务器配置文件中的以下参数：`WebServiceURL` 和`StartReportWriters`。

WebServiceURL

此参数是启用与 Web 服务通信所必需的。
URL 格式应为 `http[s]://host:port/report`。

- 默认情况下，Web 服务监听端口 10053。
  可以在Web 服务配置文件 中指定其他端口。
- 必须指定 `/report` 路径（该路径已硬编码，无法更改）。

示例：

```
WebServiceURL=http://localhost:10053/report
```

> **Attention:**
> 强烈建议使用证书在 Zabbix 服务器与 Zabbix Web 服务之间设置加密。
> 默认情况下，Zabbix 服务器与 Zabbix Web 服务之间传输的数据未加密，可能导致未经授权的访问。

StartReportWriters

此参数决定应启动的报表写入器进程数量。
如果未设置或等于 0，则会禁用报表生成。
根据所需报表的数量和频率，可以启用 1 到 100 个报表写入器进程。

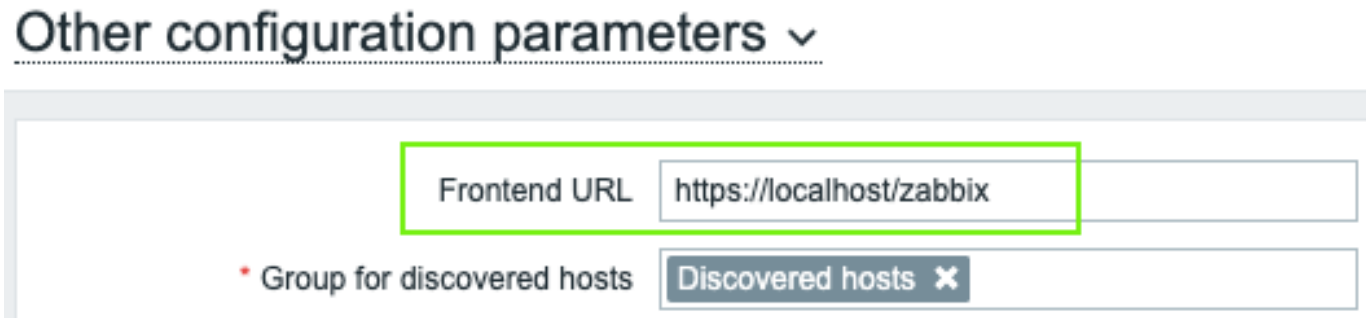
示例：

```
StartReportWriters=3
```

Zabbix 前端

应设置 *Frontend URL* 参数，以启用 Zabbix 前端与 Zabbix Web 服务之间的通信：

- 进入前端菜单部分中的 *Administration* > *General* > *Other*
- 在 *Frontend URL* 参数中指定 Zabbix Web 界面的完整 URL



Note:
设置过程完成后，您可以配置并发送一个测试报告，以确保一切正常工作。

监控项

本节包含有关监控项的参考信息和技术说明，包括特定监控项键的参数详情、平台相关细节以及特殊监控项配置的说明。

请使用侧边栏查找与您的使用场景相关的信息。

vm.memory.size 参数

概述

本节提供vm.memory.size[<mode>] agent 监控项的一些参数详细信息。

参数

以下参数可用于此监控项：

- **active** - 当前正在使用或刚刚使用过的内存，因此它位于 RAM 中
- **anon** - 不与文件关联的内存（无法从文件重新读取）
- **available** - 可用内存，根据平台不同采用不同的计算方式（见下表）
- **buffers** - 用于文件系统元数据等内容的缓存
- **cached** - 用于各种内容的缓存
- **exec** - 可执行代码，通常来自（程序）文件
- **file** - 最近访问文件内容的缓存
- **free** - 可立即提供给任何请求内存实体的内存
- **inactive** - 被标记为未使用的内存
- **pavailable** - “available” 内存占 “total” 的百分比（计算方式为 `available/total*100`）
- **pinned** - 与 “wired” 相同
- **pused** - “used” 内存占 “total” 的百分比（计算方式为 `used/total*100`）
- **shared** - 可被多个进程同时访问的内存
- **slab** - 内核为缓存其自身使用的数据结构而使用的内存总量
- **total** - 可用的物理内存总量
- **used** - 已使用内存，根据平台不同采用不同的计算方式（见下表）
- **wired** - 被标记为始终驻留在 RAM 中的内存。它永远不会被移到磁盘。

Warning:
其中一些参数是平台特定的，在您的平台上可能不可用。有关详细信息，请参见Zabbix agent items。

available 和 **used** 的平台特定计算方式：

Platform	**”available”**	**”used”**
AIX	free + cached	实际正在使用的内存
FreeBSD	inactive + cached + free	active + wired + cached
HP UX	free	total - free
Linux<3.14	free + buffers + cached	total - free

Platform	”available”	”used”
Linux 3.14+ (也回移植到了 RHEL 7 上的 3.10)	/proc/meminfo，详情请参见 Linux 内核文档中的 “MemAvailable”。 请注意，由于并非所有页缓存都可以释放，且计算中使用了低水位线，因此 free + buffers + cached 不再等于 “available”。	total - free
NetBSD	inactive + execpages + file + free	total - free
OpenBSD	inactive + free + cached	active + wired
OSX	inactive + free	active + wired
Solaris	free	total - free
Win32	free	total - free

Attention:
vm.memory.size[used] 与 *vm.memory.size[available]* 的总和不一定等于 total。例如，在 FreeBSD 上：
 * Active、inactive、wired、cached 内存被视为已使用，因为它们存储了一些有用的信息。
 * 同时，inactive、cached、free 内存也被视为可用，因为这些类型的内存可以立即分配给请求更多内存的进程。
因此，inactive 内存会同时被视为已使用和可用。正因为如此，*vm.memory.size[used]* 监控项仅用于提供信息，而 *vm.memory.size[available]* 则设计用于触发器。

参见

1.关于不同操作系统内存计算的详细信息

Windows agent 监控项的最低权限级别

概述

使用 agent 监控系统时，一种最佳实践是从安装 agent 的主机上获取指标。要使用最小权限原则，有必要确定哪些指标是从 agent 那里获得的。

本文档中的表格允许您选择最低权限，保证 Zabbix agent 的正确运行。

如果选择了其他用户才能使 agent 工作，而不是”LocalSystem”，则要使 agent 作为 Windows 服务运行，新用户必须具有 “本地策略 → 用户权限分配” 中的” 作为服务登录” 权限分配” 以及创建、写入和删除 Zabbix agent 日志文件的权利。必须将 Active Directory 用户添加到性能监视器用户组。

Note:
基于 agent 的权限处理问题上，需要给出” 技术上可接受的最低要求” 的权限组，并且事先为监控对象提供权限。

Windows 上支持的常用 agent 监控项

监控项键值	用户组	
	推荐	最低技术上可接受（功能受限）
agent.hostname	Guests	Guests
agent.ping	Guests	Guests
agent.variant	Guests	Guests
agent.version	Guests	Guests
log	Administrators	Guests
log.count	Administrators	Guests
logrt	Administrators	Guests
logrt.count	Administrators	Guests
net.dns	Guests	Guests
net.dns.perf	Guests	Guests
net.dns.record	Guests	Guests
net.if.discovery	Guests	Guests
net.if.in	Guests	Guests
net.if.out	Guests	Guests
net.if.total	Guests	Guests
net.tcp.listen	Guests	Guests
net.tcp.port	Guests	Guests
net.tcp.service	Guests	Guests

监控项键值	用户组	
net.tcp.service.perf	Guests	Guests
net.udp.service	Guests	Guests
net.udp.service.perf	Guests	Guests
proc.num	Administrators	Guests
system.cpu.discovery	Performance Monitor Users	Performance Monitor Users
system.cpu.load	Performance Monitor Users	Performance Monitor Users
system.cpu.num	Guests	Guests
system.cpu.util	Performance Monitor Users	Performance Monitor Users
system.hostname	Guests	Guests
system.localtime	Guests	Guests
system.run	Administrators	Guests
system.sw.arch	Guests	Guests
system.swap.size	Guests	Guests
system.uname	Guests	Guests
system.uptime	Performance Monitor Users	Performance Monitor Users
vfs.dir.count	Administrators	Guests
vfs.dir.get	Administrators	Guests
vfs.dir.size	Administrators	Guests
vfs.file.cksum	Administrators	Guests
vfs.file.contents	Administrators	Guests
vfs.file.exists	Administrators	Guests
vfs.file.md5sum	Administrators	Guests
vfs.file.regexp	Administrators	Guests
vfs.file.regmatch	Administrators	Guests
vfs.file.size	Administrators	Guests
vfs.file.time	Administrators	Guests
vfs.fs.discovery	Administrators	Guests
vfs.fs.size	Administrators	Guests
vfs.fs.get	Administrators	Guests
vm.memory.size	Guests	Guests
web.page.get	Guests	Guests
web.page.perf	Guests	Guests
web.page.regexp	Guests	Guests
zabbix.stats	Guests	Guests

Windows 特定的监控项键

监控项键	用户组	
	推荐	最低技术可接受（功能受限）
eventlog	Event Log Readers	Guests
net.list	Guests	Guests
perf_counter	Performance Monitor Users	Performance Monitor Users
proc_info	Administrators	Guests
service.discovery	Guests	Guests
service.info	Guests	Guests
services	Guests	Guests
wmi.get	Administrators	Guests
vm.vmemory.size	Guests	Guests

返回值的编码

服务器期望每个返回的文本值都采用 UTF8 编码。这适用于任何类型的检查：Zabbix agent、SSH、Telnet 等。

不同的被监控系统/设备和检查可能会在返回值中包含非 ASCII 字符。对于这种情况，几乎所有可能的 **zabbix** key 都包含一个额外的监控项 key 参数 - **<encoding>**。这个 key 参数是可选的，但如果返回值不是 UTF8 编码且包含非 ASCII 字符，则应指定该参数。否则，结果可能会出乎意料且不可预测。

下面将说明在这种情况下不同数据库后端的行为。

MySQL

如果某个值在非 UTF8 编码中包含非 ASCII 字符，则在数据库存储该值时，此字符及其后续内容将被丢弃。不会向 zabbix_server.log 写入任何警告消息。
 至少适用于 MySQL 5.1.61 版本。

PostgreSQL

如果某个值在非 UTF8 编码中包含非 ASCII 字符，这将导致 SQL 查询失败（PGRES_FATAL_ERROR:ERROR invalid byte sequence for encoding），并且数据不会被存储。相应的警告信息将写入 zabbix_server.log。
 至少适用于 PostgreSQL 9.1.3 版本。

大文件支持

大文件支持，通常缩写为 LFS，是指在 32 位操作系统上处理大于 2 GB 的文件的能力。对大文件的支持至少会影响日志文件监控以及所有vfs.file.* 监控项。

大文件支持取决于 Zabbix 编译时系统所具备的能力，但在 32 位 Solaris 上会被完全禁用，因为它与 procfs 和 swapctl 不兼容。

传感器

概述

每个传感器芯片在 sysfs /sys/devices 树中都有其各自的目录。要查找所有传感器芯片，更方便的方法是沿着 /sys/class/hwmon/hwmon* 中的设备符号链接进行查找，其中 * 是一个实际数字（0、1、2、.....）。

传感器读数位于虚拟设备的 /sys/class/hwmon/hwmon*/ 目录中，或非虚拟设备的 /sys/class/hwmon/hwmon*/device 目录中。位于 hwmon* 或 hwmon*/device 目录中的名为 name 的文件包含芯片名称，该名称对应于传感器芯片所使用的内核驱动名称。

每个文件中只有一个传感器读数值。上述任一目录中包含传感器读数的文件，其通用命名方案为：<type><number>_<item>，其中

- **type** - 对于传感器芯片，可以是"in"（电压）、"temp"（温度）、"fan"（风扇）等。
- **item** - "input"（测量值）、"max"（高阈值）、"min"（低阈值）等。
- **number** - 始终用于可能出现多次的元素（通常从 1 开始，但电压从 0 开始）。如果文件不对应特定元素，则其名称会比较简单，不带编号。

有关主机上可用传感器的信息，可以使用 **sensors-detect** 和 **sensors** 工具获取（lm-sensors package）。**Sensors-detect** 有助于确定可用传感器所需的模块。加载模块后，可以使用 **sensors** 程序显示所有传感器芯片的读数。该程序使用的传感器读数标签可能与通用命名方案（<type><number>_<item>）不同：

- 如果存在名为 <type><number>_label 的文件，则将使用该文件中的标签，而不是 <type><number><item> 名称；
- 如果不存在 <type><number>_label 文件，则程序会在 /etc/sensors.conf（也可能是 /etc/sensors3.conf，或其他位置）中查找名称替换规则。

这种标签机制使用户能够确定所使用的硬件类型。如果既没有 <type><number>_label 文件，配置文件中也没有标签，则可以通过 name 属性（hwmon*/device/name）确定硬件类型。zabbix_agent 可接受的实际传感器名称，可以通过使用 -u 参数运行 **sensors** 程序来获取（sensors -u）。

在 **sensors** 程序中，可用传感器按总线类型分组显示（ISA adapter、PCI adapter、SPI adapter、Virtual device、ACPI interface、HID adapter）。

获取传感器标识符

由 sensors 命令打印的传感器标签并不总是可以直接使用，因为不同芯片供应商的标签命名方式各不相同。例如，sensors 的输出可能包含以下几行：

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

其中，只有一个显示名称与内部标识符直接对应：

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

尝试使用其他显示标签（如 Vcore 或 +12V）将无法工作：

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

要查找 Zabbix 可接受的内部传感器标识符，请运行 sensors -u。在输出中，可能会看到以下内容：

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

因此，Vcore 标签对应标识符 in1，而 +12V 对应 in4。根据内核 hwmon sysfs 规范，这些是芯片引脚上的电压，通常来说可能需要进行缩放。

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

Zabbix 不仅可以获取电压（in），还可以获取电流（curr）、温度（temp）和风扇转速（fan）读数。

兼容性

旧版发行版

传感器读数从 /proc/sys/dev/sensors 目录获取：

- **device** - 设备名称（如果使用 <mode>，则为正则表达式）
- **sensor** - 传感器名称（如果使用 <mode>，则为正则表达式）
- **mode** - 可能的值：avg、max、min（如果省略此参数，则 device 和 sensor 按字面值处理）

示例键值：sensor[w83781d-i2c-0-2d,temp1]。

当前发行版

传感器读数从 /sys/class/hwmon 目录获取：

- **device** - 设备名称（非正则表达式）。设备名称可以是设备的实际名称（例如 0000:00:18.3），也可以是使用 sensors 程序获取的名称（例如 k8temp-pci-00c3）。使用哪种名称由用户自行决定。
- **sensor** - 传感器名称（非正则表达式）。
- **mode** - 可选值：avg、max、min（如果省略此参数，则 device 和 sensor 按原样处理）。

键值示例：

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

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

proc.mem 监控项中的 **memtype** 参数说明

概述

memtype 参数支持 Linux、AIX、FreeBSD 和 Solaris 平台。

所有这些平台均支持 memtype 的三个常用值：pmem、rss 和 vsize。此外，某些平台还支持特定于平台的 memtype 值。

AIX

请参阅下表，了解 AIX 上 memtype 参数支持的值。

支持的值	描述	procentry64 结构中的来源	尝试与其兼容
vsize [1]	虚拟内存大小	pi_size	
pmem	实际内存百分比	pi_prm	ps -o pmem
rss	常驻集大小	pi_trss + pi_drss	ps -o rssize
size	进程大小（代码 + 数据）	pi_dvm	ps gvw 的 SIZE 列
dsize	数据大小	pi_dsize	
tsize	文本（代码）大小	pi_tsize	ps gvw 的 TSIZ 列
sdsize	共享库中的数据大小	pi_sdsize	

支持的值	描述	procentry64 结构中的来源	尝试与其兼容
drss	数据常驻集大小	pi_drss	
trss	文本常驻集大小	pi_trss	

AIX 注意事项：

1. 在 AIX 上为 proc.mem[] 监控项键选择参数时，请尽量指定精确的进程选择条件。否则，可能会将不需要的进程计入 proc.mem[] 结果。

示例：

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

此示例表明，仅指定命令行（用于匹配的正则表达式）参数会导致 Zabbix agent 将自身计入统计结果，这可能不是您想要的结果。

2. 不要使用 ps -ef 浏览进程——它只显示非内核进程。请使用 ps -Af 查看 Zabbix agent 能够看到的所有进程。
3. 下面以 topasrec 为例，了解 Zabbix agent 如何通过 proc.mem[] 选择进程。

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

proc.mem[] 的参数如下：

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

其中：

- 第一个条件是进程名称（参数 <name>）。在我们的示例中，Zabbix agent 将其识别为 topasrec。要匹配该进程，您需要指定 topasrec，或者将其留空。
- 第二个条件是用户名（参数 <user>）。要匹配该进程，您需要指定 root，或者将其留空。
- 进程选择中使用的第三个条件是参数 <cmdline>。Zabbix agent 将其值识别为 /usr/bin/topasrec -L -s 300 -R 1 -r 6 -o /var/perf/daily/ -ypersistent=1 -O type=bin -ystart_time=04:08:54,Mar16,2023。要匹配该进程，您需要指定一个与此字符串匹配的正则表达式，或者将其留空。

参数 <mode> 和 <memtype> 会在应用上述三个条件后使用。

FreeBSD

请参阅下表，了解 FreeBSD 上 memtype 参数支持的值。

支持的值	描述	kinfo_proc 结构中的来源	尝试与以下命令兼容
vsize	虚拟内存大小	kp_eproc.e_vm.vm_map.size 或 ki_size	ps -o vsz
pmem	实际内存百分比	根据 rss 计算	ps -o pmem
rss	常驻集大小	kp_eproc.e_vm.vm_rssize 或 ki_rssize	ps -o rss
size [1]	进程大小（代码 + 数据 + 堆栈）	tsize + dsize + ssize	
tsize	文本（代码）大小	kp_eproc.e_vm.vm_tsize 或 ki_tsize	ps -o tsiz
dsize	数据大小	kp_eproc.e_vm.vm_dsize 或 ki_dsize	ps -o dsiz
ssize	堆栈大小	kp_eproc.e_vm.vm_ssize 或 ki_ssize	ps -o ssiz

Linux

请参阅下表，了解 Linux 上 memtype 参数支持的值。

支持的值	描述	/proc/<pid>/status 文件中的来源
vsize [1]	虚拟内存大小	VmSize
pmem	实际内存百分比	(VmRSS/total_memory) * 100
rss	常驻集大小	VmRSS
data	数据段大小	VmData
exe	代码段大小	VmExe
hwm	常驻集峰值大小	VmHWM
lck	锁定内存大小	VmLck
lib	共享库大小	VmLib
peak	虚拟内存峰值大小	VmPeak
pin	固定页面大小	VmPin
pte	页表项大小	VmPTE

支持的值	描述	/proc/<pid>/status 文件中的来源
size	进程代码段 + 数据段 + 栈段的大小	VmExe + VmData + VmStk
stk	栈段大小	VmStk
swap	使用的交换空间大小	VmSwap

Linux 注意事项：

1. 较旧的 Linux 内核不支持所有 memtype 值。例如，Linux 2.4 内核不支持 hwm、pin、peak、pte 和 swap 值。
2. 我们注意到，使用 proc.mem[...,...,...,...,data] 对 Zabbix agent active check 进程进行自我监控时，显示的值比 agent 的 /proc/<pid>/status 文件中 VmData 行报告的值大 4 kB。在自我测量时，agent 的数据段会增加 4 kB，随后恢复到之前的大小。

Solaris

请参见下表中 Solaris 支持的 memtype 参数值。

支持的值	描述	psinfo 结构中的来源	尝试兼容的命令
vsize [1]	进程映像大小	pr_size	ps -o vsz
pmem	实际内存百分比	pr_pctmem	ps -o pmem
rss	常驻集大小 该值可能被低估（请参见 man ps 中对 rss 的描述）。	pr_rssize	ps -o rss

脚注

[1] 默认值.

关于在 **proc.mem** 和 **proc.num** 监控项中选择进程的说明

修改其命令行的进程

有些程序会通过修改其命令行来显示当前活动状态。用户可以通过运行 ps 和 top 命令查看这些活动。这类程序的示例包括 *PostgreSQL*、*Sendmail*、*Zabbix*。

下面来看一个 Linux 中的示例。假设我们想监控若干个 Zabbix agent 进程。

ps 命令将相关进程显示为：

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

按名称和用户选择进程即可得到结果：

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

现在我们将 zabbix_agentd 可执行文件重命名为 zabbix_agentd_30，然后重启它。

此时 ps 显示为：

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

现在按名称和用户选择进程会得到错误结果：

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

为什么只是将可执行文件重命名为一个更长的名称，就会导致结果出现如此大的差异？

Zabbix agent 首先会检查进程名。它会打开 /proc/<pid>/status 文件并检查其中的 Name 行。在本例中，Name 行如下：

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

status 文件中的进程名会被截断为 15 个字符。

使用 ps 命令也可以看到类似结果：

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

显然，这与我们的 proc.num[] 的 name 参数值 zabbix_agentd_30 并不相等。在无法匹配 status 文件中的进程名后，Zabbix agent 会转而检查 /proc/<pid>/cmdline 文件。

agent 看到的 “cmdline” 文件内容，可以通过运行以下命令来说明：

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

在本例中，/proc/<pid>/cmdline 文件包含不可见、不可打印的空字节，它们在 *C* 语言中用于终止字符串。在本示例中，这些空字节显示为 “<NUL>”。

Zabbix agent 会检查主进程的 “cmdline”，并取出 zabbix_agentd_30，这与我们的 name 参数值 zabbix_agentd_30 匹配。因此，主进程会被监控项 proc.num[zabbix_agentd_30,zabbix] 统计到。

在检查下一个进程时，agent 会从 cmdline 文件中取出 zabbix_agentd_30: collector [idle 1 sec]，这不符合我们的 name 参数 zabbix_agentd_30。因此，只有未修改其命令行的主进程会被统计到。其他 agent 进程由于修改了其命令行而被忽略。

这个示例表明，在这种情况下，name 参数不能用于 proc.mem[] 和 proc.num[] 中选择进程。

> **Note:**
> 对于 proc.get[] 监控项，当 Zabbix agent 通过 “cmdline” 检查进程名时，它只会使用从最后一个斜杠开始、直到第一个空格或冒号为止的那部分名称。只有当从 cmdline 文件获取的进程名开头与 status 文件中截短后的进程名完全匹配时，才会使用从 cmdline 文件获取的进程名。该算法对于过滤器中的进程名和 JSON 输出中的进程名都是相同的。

使用带有适当正则表达式的 cmdline 参数可以得到正确结果：

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

在使用 proc.get[]、proc.mem[] 和 proc.num[] 监控项监控会修改其命令行的程序时，请务必小心。

在将 name 和 cmdline 参数用于 proc.get[]、proc.mem[] 和 proc.num[] 监控项之前，您可能希望先使用 proc.num[] 监控项和 ps 命令测试这些参数。

Linux 内核线程

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

让我们以内核线程为例：

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

可以用进程 name 参数来选择：

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

但使用进程 cmdline 参数就会不起作用：

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

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

proc.mem[] 和 proc.num[] 监控项中的线程计数

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

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

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

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

但是如果用户进程和内核线程名字相同会发生什么呢？可能会是这样:

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

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

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

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

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

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

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

本页详细介绍了针对 service 参数中指定的各种服务的 net.tcp.service 和 net.udp.service 检查的实现。

监控项 net.tcp.service 参数

ftp

创建 TCP 连接并期望响应的前 4 个字符为”220”，然后发送”QUIT\r\n”。如果未指定，则使用默认端口 21。

http

创建一个 TCP 连接而不期望和发送任何东西。如果未指定，则使用默认端口 80。

https

使用（并且仅适用于）libcurl，不验证证书的真实性，不验证 SSL 证书中的主机名，仅获取响应标头（HEAD 请求）。如果未指定，则使用默认端口 443。

imap

创建 TCP 连接并期望响应的前 4 个字符为”* OK”，然后发送”a1 LOGOUT\r\n”。如果未指定，则使用默认端口 143。

ldap

打开与 LDAP 服务器的连接并执行 LDAP 搜索操作，过滤器设置为 (objectClass=*)。期望成功检索第一个条目的第一个属性。如果未指定，则使用默认端口 389。

nntp

创建 TCP 连接并期望响应的前 3 个字符为”200” or ”201”，然后发送”QUIT\r\n”。如果未指定，则使用默认端口 119。

pop

创建 TCP 连接并期望响应的前 3 个字符为”+OK”，然后发送”QUIT\r\n”。如果未指定，则使用默认端口 110。

smtp

创建一个 TCP 连接并期望响应的前 3 个字符为”220”，后跟空格、行尾或破折号。包含破折号的行属于多行响应，响应将被重新读取直到收到没有破折号的行。然后发送”QUIT\r\n”。如果未指定，则使用默认端口 25。

ssh

创建 TCP 连接。如果连接已经建立，双方交换一个标识字符串（SSH-major.minor-XXXX），其中 major 和 minor 是协议版本，XXXX 是一个字符串。Zabbix 检查是否找到与规范匹配的字符串，然后在不匹配时发回字符串”SSH-major.minor-zabbix_agent\r\n” 或”0\n” 。如果未指定，则使用默认端口 22。

tcp

创建一个 TCP 连接而不期望和发送任何东西。与其他检查不同，需要指定端口参数。

telnet

创建一个 TCP 连接并期望登录提示（末尾为’:’ ）。如果未指定，则使用默认端口 23。

监控项 net.udp.service 参数

ntp

在 UDP 上发送一个 SNTP 包，并根据 RFC 4330，第五部分 需要验证响应。如果未指定，则使用默认端口 123。

proc.get 参数

概述

监控项**proc.get**[<name>,<user>,<cmdline>,<mode>] 支持 Linux、Windows、FreeBSD、OpenBSD 和 NetBSD。

监控项返回的进程参数列表会因操作系统以及 “mode” 参数值的不同而有所变化。

Linux

在 Linux 上，对于每种模式，都会返回以下进程参数：

mode=process	mode=thread	mode=summary
pid：PID	pid：PID	name：进程名称
ppid：父进程 PID	ppid：父进程 PID	processes：进程数
name：进程名称	name：进程名称	vsize：虚拟内存大小
cmdline：带参数的命令	user：进程运行所使用的用户（真实用户）	pmem：实际内存百分比
user：进程运行所使用的用户（真实用户）	group：进程运行所使用的组（真实组）	rss：常驻集大小
group：进程运行所使用的组（真实组）	uid：用户 ID	data：数据段大小
uid：用户 ID	gid：进程运行所使用的组的 ID	exe：代码段大小
gid：进程运行所使用的组的 ID	tid：线程 ID	lib：共享库大小
vsize：虚拟内存大小	tname：线程名称	lck：锁定内存大小
pmem：实际内存百分比	cputime_user：用户模式下消耗的 CPU 总时间（值以时钟滴答数表示——除以 `getconf CLK_TCK` 或 `sysconf(_SC_CLK_TCK)` 可获得秒数）	pin：固定页面大小
rss：常驻集大小	cputime_system：系统模式下消耗的 CPU 总时间（值以时钟滴答数表示——除以 `getconf CLK_TCK` 或 `sysconf(_SC_CLK_TCK)` 可获得秒数）	pte：页表项大小

mode=process	mode=thread	mode=summary
data：数据段大小	state：线程状态	size：进程代码段、数据段和堆栈段的总大小
exe：代码段大小	ctx_switches：上下文切换次数	stk：堆栈段大小
hwm：常驻集大小峰值	page_faults：页面错误数	swap：使用的交换空间大小
lck：锁定内存大小		cputime_user：用户模式下消耗的 CPU 总时间（值以时钟滴答数表示——除以 `getconf CLK_TCK` 或 `sysconf(_SC_CLK_TCK)` 可获得秒数）
lib：共享库大小		cputime_system：系统模式下消耗的 CPU 总时间（值以时钟滴答数表示——除以 `getconf CLK_TCK` 或 `sysconf(_SC_CLK_TCK)` 可获得秒数）
peak：虚拟内存大小峰值		ctx_switches：上下文切换次数
pin：固定页面大小		threads：线程数
pte：页表项大小		page_faults：页面错误数
size：进程代码段、数据段和堆栈段的总大小		pss：按比例计算的集大小内存
stk：堆栈段大小		
swap：使用的交换空间大小		
cputime_user：用户模式下消耗的 CPU 总时间（值以时钟滴答数表示——除以 `getconf CLK_TCK` 或 `sysconf(_SC_CLK_TCK)` 可获得秒数）		
cputime_system：系统模式下消耗的 CPU 总时间（值以时钟滴答数表示——除以 `getconf CLK_TCK` 或 `sysconf(_SC_CLK_TCK)` 可获得秒数）		
state：进程状态（从 procfs 透明获取，长格式）		
ctx_switches：上下文切换次数		
threads：线程数		
page_faults：页面错误数		
pss：按比例计算的集大小内存		

基于 BSD 的操作系统

在 FreeBSD、OpenBSD 和 NetBSD 上，每种模式都会返回以下进程参数：

mode=process	mode=thread	mode=summary
pid: PID	pid: PID	name: 进程名称
ppid: 父 PID	ppid: 父 PID	processes: 进程数量
jid: jail 的 ID（仅 FreeBSD）	jid: jail 的 ID（仅 FreeBSD）	vsize: 虚拟内存大小
jname: jail 名称（仅 FreeBSD）	jname: jail 名称（仅 FreeBSD）	pmem: 实际内存百分比（仅 FreeBSD）
name: 进程名称	name: 进程名称	rss: 常驻集大小
cmdline: 带参数的命令	user: 进程运行所属的用户（真实）	size: 进程大小（代码 + 数据 + 栈）
user: 进程运行所属的用户（真实）	group: 进程运行所属的组（真实）	tsize: 文本（代码）大小
group: 进程运行所属的组（真实）	uid: 用户 ID	dsize: 数据大小
uid: 用户 ID	gid: 进程运行所属组的 ID	ssize: 栈大小
gid: 进程运行所属组的 ID	tid: 线程 ID	cputime_user: 在用户模式下消耗的 CPU 总时间（以秒为单位报告）
vsize: 虚拟内存大小	tname: 线程名称	cputime_system: 在系统模式下消耗的 CPU 总时间（以秒为单位报告）
pmem: 实际内存百分比（仅 FreeBSD）	cputime_user: 在用户模式下消耗的 CPU 总时间（以秒为单位报告）	ctx_switches: 上下文切换次数
rss: 常驻集大小	cputime_system: 在系统模式下消耗的 CPU 总时间（以秒为单位报告）	threads: 线程数量（NetBSD 不支持）

mode=process	mode=thread	mode=summary
size: 进程大小（代码 + 数据 + 栈） tsize: 文本（代码）大小 dsize: 数据大小 ssize: 栈大小 cputime_user: 在用户模式下消耗的 CPU 总时间（以秒为单位报告） cputime_system: 在系统模式下消耗的 CPU 总时间（以秒为单位报告） state: 进程状态（磁盘休眠/运行中/休眠中/跟踪停止/僵尸/其他） ctx_switches: 上下文切换次数 threads: 线程数量（NetBSD 不支持） page_faults: 缺页次数 fds: 文件描述符数量（仅 OpenBSD） swap: 已使用的交换空间大小 io_read_op: 系统必须执行输入的次数 io_write_op: 系统必须执行输出的次数	state: 线程状态 ctx_switches: 上下文切换次数 io_read_op: 系统必须执行输入的次数 io_write_op: 系统必须执行输出的次数	stk: 栈段大小 page_faults: 缺页次数 fds: 文件描述符数量（仅 OpenBSD） swap: 已使用的交换空间大小 io_read_op: 系统必须执行输入的次数 io_write_op: 系统必须执行输出的次数

Windows

在 Windows 上，每种模式都会返回以下进程参数：

mode=process	mode=thread	mode=summary
pid: PID ppid: 父 PID name: 进程名称 user: 进程运行所属用户 sid: 用户 SID vmsize: 虚拟内存大小 wkset: 进程工作集大小 cputime_user: 在用户模式下消耗的 CPU 总时间（以毫秒为单位报告） cputime_system: 在系统模式下消耗的 CPU 总时间（以毫秒为单位报告） threads: 线程数量 page_faults: 缺页次数 handles: 句柄数量 io_read_b: 读取的 IO 字节数 io_write_b: 写入的 IO 字节数 io_read_op: IO 读取操作次数 io_write_op: IO 写入操作次数 io_other_b: 除读写操作外传输的 IO 字节数 io_other_op: 除读写操作外的 IO 操作次数	pid: PID ppid: 父 PID name: 进程名称 user: 进程运行所属用户 sid: 用户 SID tid: 线程 ID	name: 进程名称 processes: 进程数量 vmsize: 虚拟内存大小 wkset: 进程工作集大小 cputime_user: 在用户模式下消耗的 CPU 总 cputime_system: 在系统模式下消耗的 CPU threads: 线程数量 page_faults: 缺页次数 handles: 句柄数量 io_read_b: 读取的 IO 字节数 io_write_b: 写入的 IO 字节数 io_read_op: IO 读取操作次数 io_write_op: IO 写入操作次数 io_other_b: 除读写操作外传输的 IO 字节数 io_other_op: 除读写操作外的 IO 操作次数

不可达/不可用主机接口设置

概述

当 agent 检查（Zabbix、SNMP、IPMI、JMX）失败且主机接口变得不可达时，若干配置参数定义了 Zabbix 服务器应如何行为。

不可达接口

当 Zabbix、SNMP、IPMI 或 JMX agent 的检查失败（网络错误、超时）后，主机接口会被视为不可达。自 Zabbix 6.2.0 起，主动 Zabbix agent 检查也会影响接口可用性。当主动检查变为不可用时，它们会计入 agent 接口的整体可用性状态。

从接口变为不可达的那一刻起，**UnreachableDelay** 定义了使用某个监控项（包括 LLD 规则）重新检查该接口的频率。
这些重新检查由不可达轮询器执行（对于 IPMI 检查，则由 IPMI 轮询器执行）。
默认情况下，连续可达性检查之间的间隔为 15 秒。

> **Attention:**
> 异步轮询器执行的检查不会转移到不可达轮询器。

在 Zabbix 服务器日志中，不可达状态会通过如下消息体现：

```
Zabbix agent item "system.cpu.load[percpu,avg1]" on host "New host" failed: first network error, wait for
Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait f
```

日志消息会指定失败的具体监控项及其类型（Zabbix agent）。

Note:
Timeout 参数也会影响接口在不可达期间多早被重新检查。如果 Timeout 设置为 20 秒，UnreachableDelay 设置为 30 秒，那么下一次检查将在首次尝试后的 50 秒进行。

UnreachablePeriod 参数定义了不可达期间的总持续时间。默认 UnreachablePeriod 为 45 秒。
该值应比 UnreachableDelay 大几倍，以确保在接口被标记为不可用之前会被多次重新检查。

一个内部监控项zabbix[host,active_agent,available] 可用于监控不可达场景下主动检查的可用性。

将主机接口状态切换回可用

当不可达时期结束时，再次轮询接口，降低导致主机接口无法访问的监控项的优先级状态。如果不可达接口再次出现，则监控自动恢复正常：

```
resuming Zabbix agent checks on host "New host": connection restored
```

Note:
一旦接口可用，主机不会立即轮询所有监控项，有两个原因：- 它可能会使主机过载。- 主机接口恢复时间并不总是与监控项计划轮询时间匹配。
因此，在主机接口可用后，监控项不会立即被轮询，但他们将被重新安排到下一次轮询。

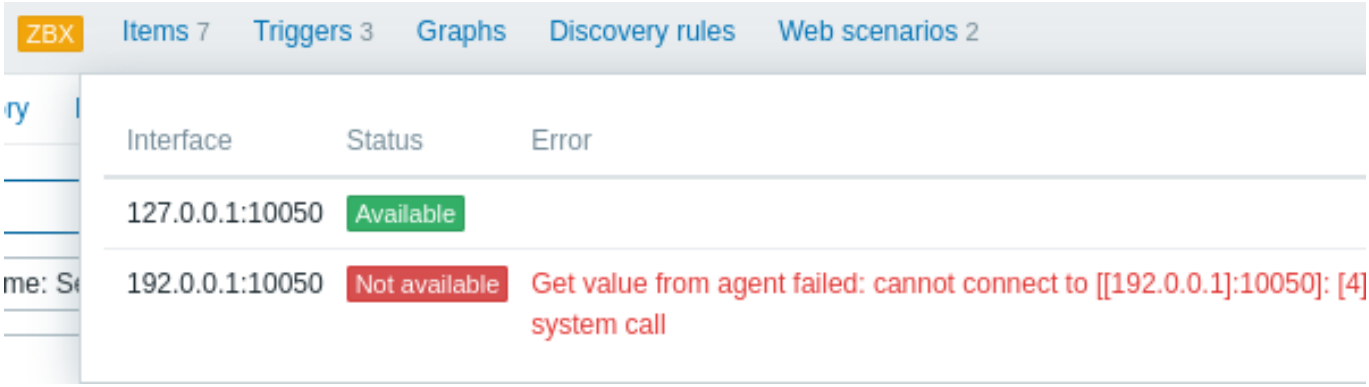
不可用接口

在 UnreachablePeriod 结束后，如果接口仍未重新出现，则该接口会被视为不可用。

在服务器日志中，会显示如下消息：

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

并且在前端 中，主机可用性图标会从绿色/灰色变为黄色/红色（当鼠标悬停在主机可用性图标上时，会显示提示框，其中可查看不可用接口的详细信息）：



UnavailableDelay 参数定义了在接口不可用期间，检查接口的频率。

默认值为 60 秒（因此在这种情况下，上述日志消息中的 “temporarily disabling” 表示禁用检查 1 分钟）。

当与该接口的连接恢复后，监控也会自动恢复正常：

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

Zabbix 统计信息的远程监控

概述

可以将 Zabbix 服务器和 proxy 的一些内部指标通过另一台 Zabbix 实例或第三方工具远程访问。这对于支持人员/服务提供商远程监控其客户的 Zabbix 服务器/proxy 很有用；或者在 Zabbix 不是主要监控工具的组织中，也可以通过第三方系统在统一监控架构中监控 Zabbix 的内部指标。

Zabbix 内部统计信息会暴露给在新的’StatsAllowedIP’ server/proxy 参数中配置的一组地址。只有来自这些地址的请求才会被接受。

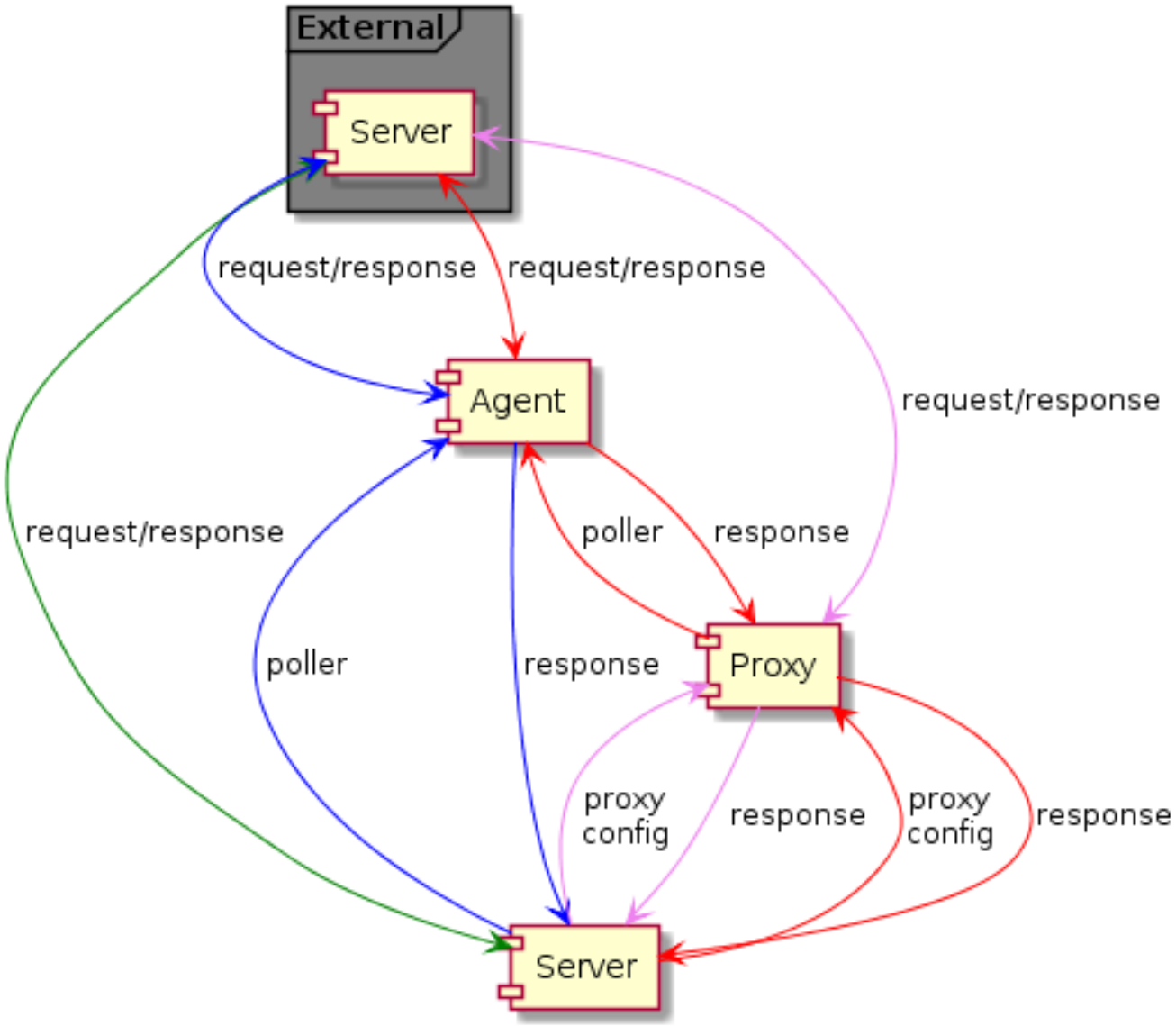
监控项

要在另一个 Zabbix 实例上配置内部统计信息的查询，可以使用两项：

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

另请参见：内部监控项，Zabbix agent 监控项

下图根据上下文说明了这两个项的用法。



- - Server → external Zabbix instance (zabbix[stats,<ip>,<port>])
- - Server → proxy → external Zabbix instance (zabbix[stats,<ip>,<port>])
- - Server → agent → external Zabbix instance (zabbix.stats[<ip>,<port>])
- - Server → proxy → agent → external Zabbix instance (zabbix.stats[<ip>,<port>])

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

暴露的指标

统计监控项会批量收集统计信息并返回 JSON，作为依赖监控项获取其数据的基础。以下内部指标会由这两个监控项中的任意一个返回：

- zabbix[boottime]
- zabbix[hosts]
- zabbix[items]
- zabbix[items_unsupported]
- zabbix[preprocessing]（仅服务器）
- zabbix[preprocessing_queue]（仅服务器）
- zabbix[process,<type>,<mode>,<state>]（仅基于进程类型的统计信息）

- zabbix[rcache,<cache>,<mode>]
- zabbix[requiredperformance]
- zabbix[triggers]（仅服务器）
- zabbix[uptime]
- zabbix[vcache,buffer,<mode>]（仅服务器）
- zabbix[vcache,cache,<parameter>]
- zabbix[version]
- zabbix[vmware,buffer,<mode>]
- zabbix[wcache,<cache>,<mode>]（“trends” 缓存类型仅服务器）

模板

可用于从外部实例远程监控 Zabbix server 或 proxy 内部指标的模板：

- Remote Zabbix server health
- Remote Zabbix proxy health

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

Trapper 进程

从外部 Zabbix 实例接收内部指标请求由 trapper 进程处理；该进程会验证请求、收集指标、创建 JSON 数据缓冲区，并将准备好的 JSON 发回，例如，从服务器返回：

```
{
  "response": "success",
  "data": {
    "boottime": N,
    "uptime": N,
    "hosts": N,
    "items": N,
    "items_unsupported": N,
    "preprocessing": {
        "queued": {
          "count": N,
          "size": N
        },
        "direct": {
          "count": N,
          "size": N
        },
        "queue": N
    },
    "preprocessing_queue": N,
    "process": {
      "alert manager": {
        "busy": {
          "avg": N,
          "max": N,
          "min": N
        },
        "idle": {
          "avg": N,
          "max": N,
          "min": N
        },
        "count": N
      },
...
    },
    "queue": N,
    "rcache": {
      "total": N,
      "free": N,
      "pfree": N,
      "used": N,
```

```
      "pused": N
    },
    "requiredperformance": N,
    "triggers": N,
    "uptime": N,
    "vcache": {
      "buffer": {
        "total": N,
        "free": N,
        "pfree": N,
        "used": N,
        "pused": N
      },
      "cache": {
        "requests": N,
        "hits": N,
        "misses": N,
        "mode": N
      }
    },
    "vmware": {
      "total": N,
      "free": N,
      "pfree": N,
      "used": N,
      "pused": N
    },
    "version": "N",
    "wcache": {
      "values": {
        "all": N,
        "float": N,
        "uint": N,
        "str": N,
        "log": N,
        "text": N,
        "not supported": N
      },
      "history": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      },
      "index": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      },
      "trend": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      }
    }
  }
}
```

内部队列监控项

另外还有两个监控项，专门用于远程查询另一个 Zabbix 实例的内部队列统计信息：

- `zabbix[stats,<ip>,<port>,queue,<from>,<to>]` 内部监控项 - 用于直接查询远程 Zabbix 服务器/proxy 的内部队列
- `zabbix.stats[<ip>,<port>,queue,<from>,<to>]` agent 监控项 - 用于基于 agent 查询远程 Zabbix 服务器/proxy 的内部队列

另请参阅：内部监控项、Zabbix agent 监控项

使用 **Zabbix** 配置 **Kerberos**

概述

Kerberos 身份验证可用于 Zabbix 中的 web 监测和 HTTP 监控项。

本页介绍了一个示例：在 Debian/Ubuntu 上为 Zabbix 服务器配置 Kerberos，使其能够使用 Zabbix 进程的 Kerberos principal 对 `www.example.com` 执行 web 监测。

配置

1. 安装 KDC 和客户端工具：

```
sudo apt update
sudo apt install krb5-kdc krb5-admin-server krb5-user
```

在软件包设置过程中回答提示，例如：

```
默认 Kerberos 版本 5 realm：EXAMPLE.COM
realm 的 Kerberos 服务器：localhost（或您的 FQDN）
Kerberos realm 的管理服务器：localhost（或您的 FQDN）
```

2. 映射易记的主机名（可选，用于本地测试）。

如果没有 DNS，请编辑 /etc/hosts，并为 DC 和 Web 服务器添加条目：

```
sudo vi /etc/hosts
```

可以添加如下示例行：

```
192.168.1.100  dc01.example.com dc01
```

3. 配置 Kerberos 客户端和 KDC realm：

```
sudo vi /etc/krb5.conf
```

示例设置：

```
[libdefaults]
    default_realm = EXAMPLE.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    rdns = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true

[realms]
    EXAMPLE.COM = {
        kdc = dc01.example.com
        admin_server = dc01.example.com
    }

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

如果计划使用 `.localdomain` 或其他非公共名称，请添加明确的 domain→realm 映射，以确保主机名 →realm 映射正常工作。
此处不匹配会导致 `Server not found in Kerberos database` 错误。

4. 初始化 Kerberos 数据库（一次性操作，在 KDC 主机上执行）。
出现提示时设置安全的主密码：

```
sudo krb5_newrealm
```

5. 使用客户端将要使用的确切主机名创建 HTTP/host.fqdn@REALM principal；建议使用小写(例如 HTTP/web.example.com@EXAMPLE.COM)。大小写或名称不匹配会导致 Server not found in Kerberos database。

```
sudo kadmin.local
```

在 kadmin.local 中：

```
addprinc kerb-admin@EXAMPLE.COM      # 管理 principal
addprinc -randkey HTTP/dc01.example.com@EXAMPLE.COM
ktadd -k /etc/apache2/http.keytab HTTP/dc01.example.com@EXAMPLE.COM
quit
```

将 keytab 移动到 Web 主机（如果是同一台计算机，则保留在本机），并设置 Apache 可使用的权限：

```
chown www-data:www-data /etc/apache2/http.keytab
chmod 600 /etc/apache2/http.keytab
#### 验证
sudo -u www-data -k /etc/apache2/http.keytab
```

6. 安装并启用 Apache GSSAPI 模块：

```
sudo apt install libapache2-mod-auth-gssapi
sudo a2enmod auth_gssapi
sudo a2enmod headers
sudo systemctl restart apache2
```

> **Attention:**
> 并非所有 mod_auth_gssapi 版本都支持每个 Gssapi* 指令。
> 如果 Apache 因 Invalid command 'GssapiCredStore' 失败，请删除不受支持的指令或升级模块。

7. 配置 VirtualHost（根据您的 Zabbix UI 调整 DocumentRoot / 路径）：

```
sudo vi /etc/apache2/sites-available/zabbix.conf
```

在 zabbix.conf 中：

```
<VirtualHost *:80>
    ServerName dc01.example.com
    DocumentRoot /usr/share/zabbix/ui
    <Directory /usr/share/zabbix/ui>
        Options FollowSymLinks
        AllowOverride None
        Require all granted
        AuthType GSSAPI
        AuthName "Kerberos Login"
        GssapiCredStore keytab:/etc/apache2/http.keytab
        GssapiLocalName On
        Require valid-user
    </Directory>
    RequestHeader set X-Remote-User %{REMOTE_USER}s env=REMOTE_USER
    RequestHeader unset Authorization
</VirtualHost>
```

重启 Apache：

```
sudo systemctl restart apache2
```

8. 启用/启动 KDC 服务并验证监听端口（在 KDC 主机上执行）：

```
sudo systemctl enable --now krb5-kdc krb5-admin-server
ss -tnlp | grep :80    # 或：sudo netstat -tnlp | grep :80
```

9. 获取用于测试的 TGT（以将使用该票证的用户身份运行）。

应在票证列表中看到 krbtgt/EXAMPLE.COM@EXAMPLE.COM。
请以需要该票证的同一 OS 用户身份运行 kinit（例如，用于 Web 检查的 zabbix，或用于交互式浏览器 SSO 测试的

www-data/Apache)。
发放给其他 OS 用户的票证不会显示，除非调整 KRB5CCNAME 和权限。

```
kinit kerb-admin@EXAMPLE.COM
klist
```

10. 使用 curl 测试 SPNEGO 交换（从具有有效 TGT 的客户端执行）。
返回 200 OK（或重定向到应用）表示 SPNEGO 成功：

```
curl -v --negotiate -u : http://dc01.example.com/
```

11. 可选：如果 Zabbix UI 应接受经过 HTTP 身份验证的登录，请在 Zabbix 前端中启用 HTTP 身份验证。

在前端配置文件（zabbix.conf.php）中设置：

```
$ZBX_FEATURE_FLAGS['http_auth_enabled'] = true;
```

然后在 Web 界面中配置 HTTP 身份验证。

12. 浏览器配置(以 Firefox 为例)：将 network.negotiate-auth.trusted-uris 设置为执行 Negotiate 的主机(dc01.example.com)，以便浏览器自动发送 Kerberos 令牌。

在 about:config 中：

```
network.negotiate-auth.trusted-uris = dc01.example.com
```

现在访问 http://dc01.example.com 应可直接登录 Zabbix，而无需填写表单。

13. 保持密钥/票证有效。
Kerberos 票证的默认生命周期约为 10 小时。
添加 cron/systemd 定时器以避免过期：

```
#### 用于 Web 服务
kinit -kt /etc/apache2/http.keytab HTTP/dc01.example.com@EXAMPLE.COM
#### 用于监控用户
kinit -kt /var/lib/zabbix/kerb.keytab kerb-admin@EXAMPLE.COM
```

14. 日常维护检查：

- klist -k /etc/apache2/http.keytab - 验证 keytab 中是否存在服务 principal。
- sudo tail -f /var/log/apache2/error.log - 监视 GSSAPI 错误(gss_acquire_cred[_from]() failed to get server creds 表示 keytab/权限问题或缺少 principal)。
- curl --negotiate 返回 401/403 通常表示 principal 错误、没有票证、主机标头不匹配或文件系统权限问题；请检查日志和 /etc/krb5.conf 中的 domain 映射。

安全性和文件权限说明

Keytab 文件必须仅对需要使用它们的账户可读。
权限示例：zabbix 用户的 keytab 文件可设置为由 zabbix:zabbix 所有且权限为 0400；Apache 的 keytab 文件可设置为由 root:www-data 所有且权限为 0440。

避免在主机上存储长期有效的明文密码。
在可能的情况下，请使用 keytab 或已加入域的计算机主体。

运行会设置 KRB5CCNAME 或复制 keytab 的测试或脚本后，请再次检查所有权和权限——Web 服务器拒绝凭据通常是由文件权限问题导致的。

modbus.get 参数

概述

下表显示了 modbus.get 监控项 参数的详细信息。

参数

Parameter	Description	Defaults	Example
endpoint	端点的协议和地址，定义为 `protocol://connection_string` 可能的协议值：*rtu*、*ascii*（仅 Agent 2）、*tcp* 连接字符串格式： 使用 *tcp* 时 - `address:port` 使用串行线路时：*rtu*、*ascii* - `port_name:speed:params` 其中 'speed' - 1200、9600 等 'params' - 数据位（5、6、7 或 8）、奇偶校验（n、e 或 o，分别表示无校验/偶校验/奇校验）、停止位（1 或 2）	protocol: 无 *rtu/ascii* 协议： port_name: 无 speed: 115200 params: 8n1 *tcp* 协议： address: 无 port: 502	tcp://192.168.6.1:511 tcp://192.168.6.2 tcp://[::1]:511 tcp://::1 tcp://localhost:511 tcp://localhost rtu://COM1:9600:8n ascii://COM2:1200:7o2 rtu://ttyS0:9600 ascii://ttyS1
slave id	目标设备的 Modbus 地址（1 到 247），请参见 MODBUS Messaging Implementation Guide（第 23 页） tcp 设备（非 GW）将忽略该字段	serial: 1 tcp: 255 (0xFF)	2
function	留空或填写受支持的功能值： 1 - 读取线圈（Read Coil）， 2 - 读取离散输入（Read Discrete Input）， 3 - 读取保持寄存器（Read Holding Registers）， 4 - 读取输入寄存器（Read Input Registers）	空	3
address	第一个寄存器、线圈或输入的地址。 如果'function' 为空，则'address' 应位于以下范围内： 线圈 - 00001 - 09999 离散输入 - 10001 - 19999 输入寄存器 - 30001 - 39999 保持寄存器 - 40001 - 49999 如果'function' 不为空，则'address' 字段范围为 0 到 65535，并且将不作修改直接使用（PDU）	空 function: 00001 非空 function: 0	9999
count	将从设备读取的连续'type' 数量，其中： 对于 Coil 或 Discrete input，'type' = 1 bit 对于其他情况：(count*sizeof(type))/2 = 实际读取的寄存器数量 如果'offset' 不为 0，则该值将加到 "实际数量" 上 "实际数量" 的可接受范围为 1:65535	1	2
type	数据类型： 对于 Read Coil 和 Read Discrete Input - *bit* 对于 Read Holding Registers 和 Read Input Registers： *int8* - 8bit *uint8* - 8bit（无符号） *int16* - 16bit *uint16* - 16bit（无符号） *int32* - 32bit *uint32* - 32bit（无符号） *float* - 32bit *uint64* - 64bit（无符号） *double* - 64bit	bit uint16	uint64

Parameter	Description	Defaults	Example
endianness	字节序类型： *be* - 大端序 *le* - 小端序 *mbe* - 中间大端序 *mle* - 中间小端序 限制： 对于 1 bit - be 对于 8 bits - be、le 对于 16 bits - be、le	be	le
offset	从'address' 开始计算、其结果将被丢弃的寄存器数量。 每个寄存器的大小为 16bit（用于支持不支持随机读取访问的设备）。	0	4

为 **VMware** 创建自定义性能计数器名称

概述

VMware 性能计数器路径采用 `group/counter[rollup]` 格式，其中：

- `group` - 性能计数器组，例如 *cpu*
- `counter` - 性能计数器名称，例如 *usagemhz*
- `rollup` - 性能计数器汇总类型，例如 *average*

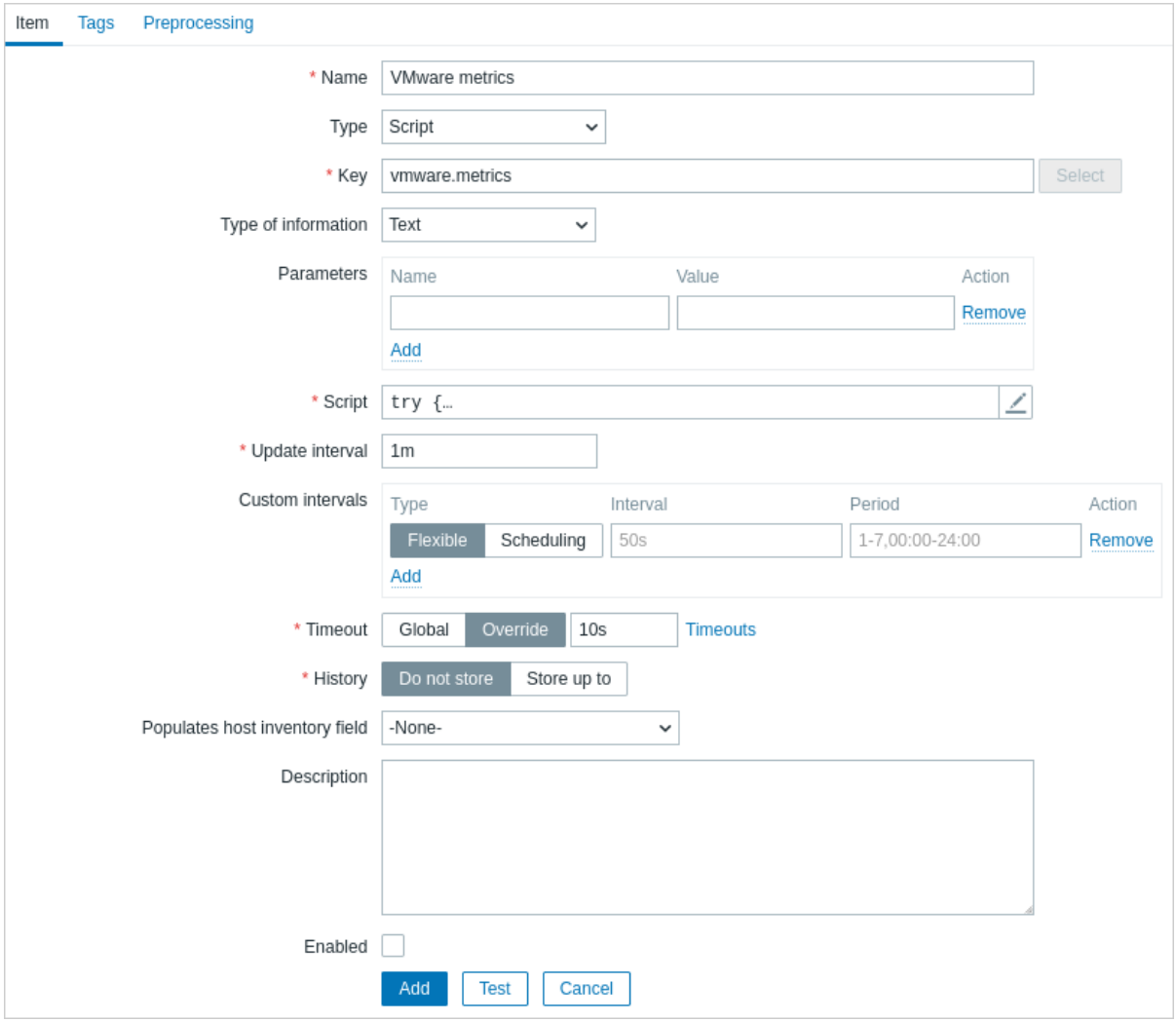
因此，上述示例对应的计数器路径为：`cpu/usagemhz[average]`

性能计数器组说明、计数器名称和汇总类型可在 VMware 文档 中找到。

可以通过在 Zabbix 中使用脚本监控项来获取内部名称，并创建自定义性能计数器名称。

配置

1. 在主 VMware 主机上创建一个已禁用的 Script 监控项（其中存在 **eventlog[]** 监控项），并使用以下参数：



- 名称: VMware metrics
- 类型: Script
- 键值: vmware.metrics
- 信息类型: Text
- 脚本: 复制并粘贴下面提供的脚本
- 超时: 10
- 历史: 不保存
- 启用: 未勾选

脚本

```
try {
    Zabbix.log(4, 'vmware metrics script');

    var result, resp,
    req = new HttpRequest();
    req.addHeader('Content-Type: application/xml');
    req.addHeader('SOAPAction: "urn:vim25/6.0"');

    login = '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:vi
    <soapenv:Header/>\
    <soapenv:Body>\
        <urn:Login>\
            <urn:_this type="SessionManager">SessionManager</urn:_this>\
            <urn:userName>{$VMWARE.USERNAME}</urn:userName>\
            <urn:password>{$VMWARE.PASSWORD}</urn:password>\
        </urn:Login>\
    </soapenv:Body>\
</soapenv:Envelope>'
    resp = req.post("{$VMWARE.URL}", login);
    if (req.getStatus() != 200) {
```

```
        throw 'Response code: '+req.getStatus();
    }

    query = '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:vi
<soapenv:Header/>\
    <soapenv:Body>\
        <urn:RetrieveProperties>\
            <urn:_this type="PropertyCollector">propertyCollector</urn:_this>\
            <urn:specSet>\
                <urn:propSet>\
                   <urn:type>PerformanceManager</urn:type>\
                   <urn:pathSet>perfCounter</urn:pathSet>\
                </urn:propSet>\
                <urn:objectSet>\
                   <urn:obj type="PerformanceManager">PerfMgr</urn:obj>\
                </urn:objectSet>\
            </urn:specSet>\
        </urn:RetrieveProperties>\
    </soapenv:Body>\
</soapenv:Envelope>'
    resp = req.post("{$VMWARE.URL}", query);
    if (req.getStatus() != 200) {
        throw 'Response code: '+req.getStatus();
    }
    Zabbix.log(4, 'vmware metrics=' + resp);
    result = resp;

    logout = '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:v
    <soapenv:Header/>\
    <soapenv:Body>\
        <urn:Logout>\
            <urn:_this type="SessionManager">SessionManager</urn:_this>\
        </urn:Logout>\
    </soapenv:Body>\
</soapenv:Envelope>'

    resp = req.post("{$VMWARE.URL}",logout);
    if (req.getStatus() != 200) {
        throw 'Response code: '+req.getStatus();
    }

} catch (error) {
    Zabbix.log(4, 'vmware call failed : '+error);
    result = {};
}

return result;
```

配置好监控项后，点击 *Test* 按钮，然后点击 *Get value*。



将收到的 XML 复制到任意 XML 格式化工具中，并找到所需的指标。

一个指标的 XML 示例：

```
<PerfCounterInfo xsi:type="PerfCounterInfo">
    <key>6</key>
```

```
    <nameInfo>
        <label>Usage in MHz</label>
        <summary>CPU usage in megahertz during the interval</summary>
        <key>usagemhz</key>
    </nameInfo>
    <groupInfo>
        <label>CPU</label>
        <summary>CPU</summary>
        <key>cpu</key>
    </groupInfo>
    <unitInfo>
        <label>MHz</label>
        <summary>Megahertz</summary>
        <key>megaHertz</key>
    </unitInfo>
    <rollupType>average</rollupType>
    <statsType>rate</statsType>
    <level>1</level>
    <perDeviceLevel>3</perDeviceLevel>
</PerfCounterInfo>
```

使用 XPath 从收到的 XML 中提取计数器路径。以上示例中，XPath 为：

field	xPath	value
group	//groupInfo[../key=6]/key	cpu
counter	//nameInfo[../key=6]/key	usagemhz
rollup	//rollupType[../key=6]	average

在这种情况下，得到的性能计数器路径为：`cpu/usagemhz[average]`

system.sw.packages.get 的返回值

概述

本节提供 Zabbix agent 监控项`system.sw.packages.get` 返回值的详细信息。

详细信息

此监控项的输出是一个对象数组，其中每个对象都包含以下属性：

- **name** - 软件包名称
- **manager** - 报告此数据的软件包管理器（rpm、dpkg、pacman、pkgtool 或 portage）
- **version** - 软件包版本
- **size** - 未压缩的软件包大小（以字节为单位）（如果不可用，则设为 0）
- **arch** - 软件包架构
- **buildtime** - 包含 2 个条目的对象：
  - **timestamp** - 软件包构建时的 UNIX 时间戳（如果不可用，则设为 0）
  - **value** - 软件包构建时的人类可读日期和时间（如果不可用，则设为空字符串）
- **installtime** - 包含 2 个条目的对象：
  - **timestamp** - 软件包安装时的 UNIX 时间戳（如果不可用，则设为 0）
  - **value** - 软件包安装时的人类可读日期和时间（如果不可用，则设为空字符串）

示例：

```
[
    {
        "name": "util-linux-core",
        "manager": "rpm",
        "version": "2.37.4-3.el9",
        "size": 1296335,
        "arch": "x86_64",
        "buildtime": {
            "timestamp" : 1653552239,
            "value" : "Sep 20 01:39:40 2021 UTC"
```

```
        },
        "installtime": {
            "timestamp" : 1660780885,
            "value" : "Aug 18 00:01:25 2022 UTC"
        }
    },
    {
        "name": "xfonts-base",
        "manager": "dpkg",
        "version": "1:1.0.5",
        "size": 7337984,
        "arch": "all",
        "buildtime": {
            "timestamp": 0,
            "value": ""
        },
        "installtime": {
            "timestamp": 0,
            "value": ""
        }
    }
]
```

net.dns.get 的返回值

概述

本部分提供 Zabbix agent 2 监控项net.dns.get 返回值的详细信息。

详情

此监控项的输出是一个对象，其中包含基于监控项键中提供的参数的 DNS 记录信息。

例如，net.dns.get[,example.com] 监控项可能返回如下被拒绝查询的 JSON：

```
{
    "flags": [
        "RA"
    ],
    "query_time": "0.019030",
    "question_section": [
        {
            "qclass": "IN",
            "qname": "example.com.",
            "qtype": "SOA"
        }
    ],
    "response_code": "REFUSED",
    "zbx_error_code": 0
}
```

通过指定 DNS 服务器的 IP 地址，net.dns.get[192.0.2.1,example.com] 监控项可能返回如下 JSON：

```
{
    "answer_section": [
        {
            "class": "IN",
            "name": "example.com.",
            "rdata": {
                "expire": 1209600,
                "mbox": "noc.dns.example.org.",
                "minttl": 3600,
                "ns": "ns.example.org.",
                "refresh": 7200,
                "retry": 3600,
```

```
                "serial": 2022091378
            },
            "rdlength": 44,
            "ttl": 1205,
            "type": "SOA"
        }
    ],
    "flags": [
        "RA"
    ],
    "query_time": "0.029556",
    "question_section": [
        {
            "qclass": "IN",
            "qname": "example.com.",
            "qtype": "SOA"
        }
    ],
    "response_code": "NOERROR",
    "zbx_error_code": 0
}
```


如果存在连接问题，net.dns.get[192.0.2.1,example.com] 监控项可能返回错误：


```
{
    "zbx_error_code": -1,
    "zbx_error_msg": "Communication error: read udp 192.0.2.1:12345->192.0.2.1:53: i/o timeout"
}
```


可能出现以下类型的错误代码：

Scenario	"zbx_error_code"	"zbx_error_msg"
没有错误，且已接收并解析 DNS 响应。	0	
DNS 已停止运行。	-1	"Communication error"
JSON 解析过程中发生错误	-2	"Received unexpected response"

添加额外参数后，net.dns.get[192.0.2.1,example.com,ANY,5,5,tcp,"cdflag,rdflag,dnssec,nsid,edns0,aaflag,adflag" 监控项可能返回如下 JSON：


```
{
    "additional_section": [
        {
            "extended_rcode": 32768,
            "name": ".",
            "rdata": {
                "options": [
                {
                    "code": 0,
                    "nsid": "67 70 64 6e 73 2d 6c 70 70"
                }
                ]
        },
            "rdlength": 13,
            "type": "OPT",
            "udp_payload": 512
        }
    ],
    "answer_section": [
        {
            "class": "IN",
            "name": "example.com.",
            "rdata": {
                "a": "192.0.2.1"
```

```
            },
            "rdlength": 4,
            "ttl": 19308,
            "type": "A"
        },
        {
            "class": "IN",
            "name": "example.com.",
            "rdata": {
                "algorithm": 13,
                "expiration": 1704715951,
                "inception": 1702910624,
                "key_tag": 21021,
                "labels": 2,
                "orig_ttl": 86400,
                "signature": "HVBOBcJJQy0S08J3f8kviPj8UkEUj7wmyiMyQqPSWgQIY9SCEJ5plq6KuxJmtAek1txZWXDo+6tp
                "signer_name": "example.com.",
                "type_covered": "A"
            },
            "rdlength": 95,
            "ttl": 19308,
            "type": "RRSIG"
        }
    ],
    "flags": [
        "RD",
        "RA",
        "AD",
        "CD"
    ],
    "query_time": "0.058221",
    "question_section": [
        {
            "qclass": "IN",
            "qname": "example.com.",
            "qtype": "ANY"
        }
    ],
    "response_code": "NOERROR",
    "zbx_error_code": 0
}
```


参见

对于 DNS 记录的更多信息，查阅:

- 域名 - 实施和规范
- 域名系统（DNS）参数

net.if.get 的返回值

概述

本节提供net.if.get 监控项的返回值详细信息。
该监控项在 Linux 上受 Zabbix agent 和 Zabbix agent 2 支持。

详细信息

此监控项返回名称与给定正则表达式匹配的网络接口的详细信息。输出采用 JSON 对象结构，包含 config 和 values 数组。

输出包含以下属性：

- **name** - 接口名称，从 /proc/net/dev 读取；
- **ifalias** - 接口别名，从 /sys/class/net/<name>/ifalias 读取；
- **mac** - 接口 MAC 地址，从 /sys/class/net/<name>/address 读取；
- **type** - 接口类型（physical、virtual、loopback）；

- **speed** - 链路速度，单位为 Mbits/sec，仅适用于以太网，从 /sys/class/net/<name>/speed 读取；
- **duplex** - 接口最近或当前的双工值（half、full），仅适用于以太网，从 /sys/class/net/<name>/duplex 读取；
- **administrative_state** - 配置状态（down、up），从 /sys/class/net/<name>/flags 读取；
- **operational_state** - 接口 RFC2863 运行状态（unknown、notpresent、down、lowerlayerdown、testing、dormant、up），从 sys/class/net/<name>/operstate 读取；
- **carrier** - 物理链路是否存在（0 - 物理链路关闭，1 - 物理链路开启），从 /sys/class/net/<name>/carrier 读取；
- **carrier_changes** - 链路状态从 up 变为 down 以及从 down 变为 up 的总次数，从 /sys/class/net/<name>/carrier_changes 读取；
- **carrier_up_count** - 链路变为 up 的次数，从 /sys/class/net/<name>/carrier_up_count 读取；
- **carrier_down_count** - 链路变为 down 的次数，从 /sys/class/net/<name>/carrier_down_count 读取；
- **in** - 入站流量，从 /proc/net/dev 读取。包含接口计数器：
  - **bytes** - 正常字节数；
  - **packets** - 正常数据包数；
  - **errors** - 错误数据包总数；
  - **dropped** - 已接收但未处理的数据包数 + 主机遗漏的数据包数；
  - **overruns** - 溢出事件数；
  - **frame** - 帧错误数据包数；
  - **compressed** - 正确接收的压缩数据包数；
  - **multicast** - 已接收的多播数据包数。
- **out** - 出站流量，从 /proc/net/dev 读取。包含接口计数器：
  - **bytes** - 正常字节数；
  - **packets** - 正常数据包数；
  - **errors** - 错误数据包总数；
  - **dropped** - 已接收但未处理的数据包数 + 主机遗漏的数据包数；
  - **overruns** - 溢出事件数；
  - **collisions** - 数据包传输期间的冲突次数；
  - **carrier** - 由于传输期间载波丢失而导致的帧传输错误次数；
  - **compressed** - 已发送的压缩数据包数。

不可用的指标不会包含在 JSON 输出中。在以下示例中，”lo” 接口未配置别名，因此不包含 ifalias 属性。

示例：


```
{
    "config": [
        {
            "name": "lo",
            "mac": "00:00:00:00:00:00",
            "type": "loopback",
            "administrative_state": "up",
            "operational_state": "unknown",
            "carrier": 1
        },
        {
            "name": "enp3s0",
            "ifalias": "Wired Ethernet connection",
            "mac": "8c:16:45:e3:1d:58",
            "type": "physical",
            "duplex": "unknown",
            "administrative_state": "up",
            "operational_state": "down",
            "carrier": 0
        },
        {
            "name": "wlp5s0",
            "ifalias": "Wi-Fi network adapter",
            "mac": "34:e1:2d:72:67:c6",
            "type": "physical",
            "administrative_state": "up",
            "operational_state": "up",
            "carrier": 1
        }
    ],
    "values": [
```

```
        {
            "name": "lo",
            "mac": "00:00:00:00:00:00",
            "carrier": 1,
            "carrier_changes": 0,
            "carrier_up_count": 0,
            "carrier_down_count": 0,
            "in": {
                "bytes": 1040047,
                "packets": 10784,
                "errors": 0,
                "dropped": 0,
                "overruns": 0,
                "frame": 0,
                "compressed": 0,
                "multicast": 0
            },
            "out": {
                "bytes": 1040047,
                "packets": 10784,
                "errors": 0,
                "dropped": 0,
                "overruns": 0,
                "collisions": 0,
                "carrier": 0,
                "compressed": 0
            }
        },
        {
            "name": "enp3s0",
            "ifalias": "Wired Ethernet connection",
            "mac": "8c:16:45:e3:1d:58",
            "carrier": 0,
            "carrier_changes": 1,
            "carrier_up_count": 0,
            "carrier_down_count": 1,
            "in": {
                "bytes": 0,
                "packets": 0,
                "errors": 0,
                "dropped": 0,
                "overruns": 0,
                "frame": 0,
                "compressed": 0,
                "multicast": 0
            },
            "out": {
                "bytes": 0,
                "packets": 0,
                "errors": 0,
                "dropped": 0,
                "overruns": 0,
                "collisions": 0,
                "carrier": 0,
                "compressed": 0
            }
        },
        {
            "name": "wlp5s0",
            "ifalias": "Wi-Fi network adapter",
            "mac": "34:e1:2d:72:67:c6",
            "carrier": 1,
```

```
            "carrier_changes": 6,
            "carrier_up_count": 3,
            "carrier_down_count": 3,
            "in": {
                "bytes": 165537298,
                "packets": 216182,
                "errors": 0,
                "dropped": 291,
                "overruns": 0,
                "frame": 0,
                "compressed": 0,
                "multicast": 0
            },
            "out": {
                "bytes": 252574347,
                "packets": 246607,
                "errors": 0,
                "dropped": 0,
                "overruns": 0,
                "collisions": 0,
                "carrier": 0,
                "compressed": 0
            }
        }
    ]
}
```


另请参阅

有关网络设备 sysfs ABI 的更多信息，请参见：

- Linux 内核文档

vfs.dev.get 的返回值

概述

本节提供`vfs.dev.get`监控项的返回值详情。该监控项在 Linux 上由 Zabbix agent 和 Zabbix agent 2 支持，在 AIX 上由 Zabbix agent 支持。

详情

此监控项返回名称与给定正则表达式匹配的块设备的详细信息。

输出结构为一个 JSON 对象，包含 `config`，以及根据 <mode> 参数不同而包含的 `values` 数组。

`config` 数组包含所选模式可用的属性。`values` 数组仅在提供块设备统计信息的模式下返回。

不可用的指标会从 JSON 输出中排除。

以下各节描述了每种模式返回的属性。

磁盘模式

`vfs.dev.get[disks,.*]` 返回块设备配置数据。

属性：

- **name** - 块设备名称；
- **devid** - 块设备标识符，例如基于 WWN 的设备 ID 或内核生成的设备 ID；
- **type** - 块设备类型（disk、partition、rom）；
- **path** - 块设备节点路径；
- **model** - 设备型号；
- **serial** - 设备序列号；
- **wwn** - 设备的全球唯一名称（WWN）；
- **size_bytes** - 设备大小，单位为字节；
- **logical_block_size** - 逻辑块大小，单位为字节；
- **physical_block_size** - 物理块大小，单位为字节。

Linux 示例：


```
{
  "config": [
    {
      "name": "sda",
      "devid": "ata-ST2000LM015-2E8174_ZDZ3YHZY",
      "type": "disk",
      "path": "/dev/sda",
      "model": " ST2000LM015-2E81",
      "wwn": "naa.5000c500b2791423",
      "size_bytes": 2000398934016,
      "logical_block_size": 512,
      "physical_block_size": 4096
    },
    {
      "name": "sdb",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0",
      "type": "disk",
      "path": "/dev/sdb",
      "model": "DataTraveler 3.0",
      "size_bytes": 61991813632,
      "logical_block_size": 512,
      "physical_block_size": 512
    },
    {
      "name": "nvme0n1",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633",
      "type": "disk",
      "path": "/dev/nvme0n1",
      "model": "SAMSUNG MZVLW256HEHP-000L7",
      "serial": "S35ENX1K753633",
      "wwn": "eui.002538b781b7dc55",
      "size_bytes": 256060514304,
      "logical_block_size": 512,
      "physical_block_size": 512
    }
  ]
}
```


AIX 示例：


```
{
  "config": [
    {
      "name": "cd0",
      "type": "disk",
      "path": "/dev/cd0",
      "size_bytes": 0,
      "logical_block_size": 512
    },
    {
      "name": "hdisk1",
      "type": "disk",
      "devid": "48333321360050768OC818022F000000000015EA404214503IBMfcp05VDASD03AIXvscsi",
      "path": "/dev/hdisk1",
      "size_bytes": 51136000000,
      "logical_block_size": 512
    },
    {
      "name": "hdisk0",
      "type": "disk",
      "devid": "48333321360050768OC818022F000000000015EA304214503IBMfcp05VDASD03AIXvscsi",
      "path": "/dev/hdisk0",
```

```
      "size_bytes": 0,
      "logical_block_size": 512
    }
  ]
}
```

设备模式

`vfs.dev.get[devices,.*]` 返回块设备配置数据，包括分区。

属性：

- **name** - 块设备名称。
- **devid** - 块设备标识符。
- **type** - 块设备类型（disk、partition、rom）。
- **partitions** - 设备上的分区。包含分区名称及其字节大小。

Linux 示例：

```
{
  "config": [
    {
      "name": "sda",
      "devid": "ata-ST2000LM015-2E8174_ZDZ3YHZY",
      "type": "disk",
      "partitions": {
        "sda1": 16759808,
        "sda2": 2000381018112
      }
    },
    {
      "name": "sdb",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0",
      "type": "disk",
      "partitions": {
        "sdb1": 27631878144,
        "sdb2": 34359768576,
        "sdb3": 32256
      }
    },
    {
      "name": "nvme0n1",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633",
      "type": "disk",
      "partitions": {
        "nvme0n1p1": 209715200,
        "nvme0n1p2": 16777216,
        "nvme0n1p3": 255060869120,
        "nvme0n1p4": 770703360
      }
    }
  ]
}
```

disk_stats 模式

`vfs.dev.get[disk_stats,.*]` 返回块设备配置数据和磁盘统计信息。

属性：

- **name** - 块设备名称。
- **devid** - 块设备标识符。
- **type** - 块设备类型（disk、partition、rom）。
- **size_bytes** - 设备大小，单位为字节。
- **logical_block_size** - 逻辑块大小，单位为字节。
- **stats** - 块设备 I/O 统计信息。包含：
  - **reads_completed** - 已完成的读取操作数量；

- **writes_completed** - 已完成的写入操作数量；
- **bytes_read** - 已读取的字节数；
- **bytes_written** - 已写入的字节数；
- **io_time_ms** - 执行 I/O 的总耗时，单位为毫秒。

Linux 示例


```
{
  "config": [
    {
      "name": "sda",
      "devid": "ata-ST2000LM015-2E8174_ZDZ3YHZY",
      "type": "disk",
      "size_bytes": 61991813632
    },
    {
      "name": "sdb",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0",
      "type": "disk",
      "size_bytes": 61991813632
    },
    {
      "name": "nvme0n1",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633",
      "type": "disk",
      "size_bytes": 256060514304
    }
  ],
  "values": [
    {
      "name": "sda",
      "stats": {
        "reads_completed": 871,
        "writes_completed": 0,
        "bytes_read": 13045760,
        "bytes_written": 0,
        "io_time_ms": 1256
      }
    },
    {
      "name": "sdb",
      "stats": {
        "reads_completed": 51017,
        "writes_completed": 10682,
        "bytes_read": 10514627,
        "bytes_written": 1530472,
        "io_time_ms": 284988
      }
    },
    {
      "name": "nvme0n1",
      "stats": {
        "reads_completed": 1259,
        "writes_completed": 7,
        "bytes_read": 49698,
        "bytes_written": 6,
        "io_time_ms": 243
      }
    }
  ]
}
```


AIX 示例：

```
{
  "config": [
    {
      "name": "cd0",
      "type": "disk",
      "path": "/dev/cd0",
      "size_bytes": 0,
      "logical_block_size": 2048
    },
    {
      "name": "hdisk1",
      "type": "disk",
      "devid": "48333321360050768OC818022F000000000015EA404214503IBMfcp05VDASD03AIXvscsi",
      "path": "/dev/hdisk1",
      "size_bytes": 51136000000,
      "logical_block_size": 512
    },
    {
      "name": "hdisk0",
      "type": "disk",
      "devid": "48333321360050768OC818022F000000000015EA304214503IBMfcp05VDASD03AIXvscsi",
      "path": "/dev/hdisk0",
      "size_bytes": 0,
      "logical_block_size": 512
    }
  ],
  "values": [
    {
      "name": "cd0",
      "stats": {
        "reads_completed": 21,
        "writes_completed": 0,
        "bytes_read": 75776,
        "bytes_written": 0,
        "io_time_ms": 0
      }
    },
    {
      "name": "hdisk1",
      "stats": {
        "reads_completed": 35888,
        "writes_completed": 1165147,
        "bytes_read": 593699840,
        "bytes_written": 10009646592,
        "io_time_ms": 69
      }
    },
    {
      "name": "hdisk0",
      "stats": {
        "reads_completed": 3,
        "writes_completed": 0,
        "bytes_read": 12800,
        "bytes_written": 0,
        "io_time_ms": 0
      }
    }
  ]
}
```


device_stats 模式

`vfs.dev.get[device_stats,.*]` 返回块设备配置数据和设备统计信息。

属性：

- **name** - 块设备名称。
- **devid** - 块设备标识符。
- **type** - 块设备类型（`disk`、`partition`、`rom`）。
- **size_bytes** - 设备大小，单位为字节。
- **stats** - 块设备 I/O 统计信息。包含：
  - **reads_completed** - 已完成的读取操作次数；
  - **writes_completed** - 已完成的写入操作次数；
  - **bytes_read** - 已读取的字节数；
  - **bytes_written** - 已写入的字节数；
  - **io_time_ms** - 执行 I/O 所花费的总时间，单位为毫秒。

Linux 示例：


```
{
  "config": [
    {
      "name": "sda",
      "devid": "ata-ST2000LM015-2E8174_ZDZ3YHZY",
      "type": "disk",
      "size_bytes": 61991813632
    },
    {
      "name": "sda1",
      "type": "partition",
      "devid": "ata-ST2000LM015-2E8174_ZDZ3YHZY-part1",
      "size_bytes": 1234567890
    },
    {
      "name": "sda2",
      "type": "partition",
      "devid": "ata-ST2000LM015-2E8174_ZDZ3YHZY-part2",
      "size_bytes": 1234567890
    },
    {
      "name": "sdb",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0",
      "type": "disk",
      "size_bytes": 61991813632
    },
    {
      "name": "sdb1",
      "type": "partition",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0-part1",
      "size_bytes": 1234567890
    },
    {
      "name": "sdb2",
      "type": "partition",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0-part2",
      "size_bytes": 1234567890
    },
    {
      "name": "sdb3",
      "type": "partition",
      "devid": "usb-Kingston_DataTraveler_3.0_E0D55E6C382C15A158991CBD-0:0-part3",
      "size_bytes": 1234567890
    },
    {
      "name": "nvme0n1",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633",
      "type": "disk",
      "size_bytes": 256060514304
```

```
    },
    {
      "name": "nvme0n1p1",
      "type": "partition",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633-part1",
      "size_bytes": 1234567890
    },
    {
      "name": "nvme0n1p2",
      "type": "partition",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633-part2",
      "size_bytes": 1234567890
    },
    {
      "name": "nvme0n1p3",
      "type": "partition",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633-part3",
      "size_bytes": 1234567890
    },
    {
      "name": "nvme0n1p4",
      "type": "partition",
      "devid": "nvme-SAMSUNG_MZVLW256HEHP-000L7_S35ENX1K753633-part4",
      "size_bytes": 1234567890
    }
  ],
  "values": [
    {
      "name": "sda",
      "stats": {
        "reads_completed": 871,
        "writes_completed": 0,
        "bytes_read": 13045760,
        "bytes_written": 0,
        "io_time_ms": 1256
      }
    },
    {
      "name": "sda1",
      "stats": {
        "reads_completed": 123,
        "writes_completed": 123,
        "bytes_read": 123,
        "bytes_written": 123,
        "io_time_ms": 123
      }
    },
    {
      "name": "sda2",
      "stats": {
        "reads_completed": 123,
        "writes_completed": 123,
        "bytes_read": 123,
        "bytes_written": 123,
        "io_time_ms": 123
      }
    },
    {
      "name": "sdb",
      "stats": {
        "reads_completed": 51017,
        "writes_completed": 10682,
```

```
      "bytes_read": 10514627,
      "bytes_written": 1530472,
      "io_time_ms": 284988
    }
  },
  {
    "name": "sdb1",
    "stats": {
      "reads_completed": 123,
      "writes_completed": 123,
      "bytes_read": 123,
      "bytes_written": 123,
      "io_time_ms": 123
    }
  },
  {
    "name": "sdb2",
    "stats": {
      "reads_completed": 123,
      "writes_completed": 123,
      "bytes_read": 123,
      "bytes_written": 123,
      "io_time_ms": 123
    }
  },
  {
    "name": "sdb3",
    "stats": {
      "reads_completed": 123,
      "writes_completed": 123,
      "bytes_read": 123,
      "bytes_written": 123,
      "io_time_ms": 123
    }
  },
  {
    "name": "nvme0n1",
    "stats": {
      "reads_completed": 1259,
      "writes_completed": 7,
      "bytes_read": 49698,
      "bytes_written": 6,
      "io_time_ms": 243
    }
  },
  {
    "name": "nvme0n1p1",
    "stats": {
      "reads_completed": 123,
      "writes_completed": 123,
      "bytes_read": 123,
      "bytes_written": 123,
      "io_time_ms": 123
    }
  },
  {
    "name": "nvme0n1p2",
    "stats": {
      "reads_completed": 123,
      "writes_completed": 123,
      "bytes_read": 123,
      "bytes_written": 123,
```

```
        "io_time_ms": 123
      }
    },
    {
      "name": "nvme0n1p3",
      "stats": {
        "reads_completed": 123,
        "writes_completed": 123,
        "bytes_read": 123,
        "bytes_written": 123,
        "io_time_ms": 123
      }
    },
    {
      "name": "nvme0n1p4",
      "stats": {
        "reads_completed": 123,
        "writes_completed": 123,
        "bytes_read": 123,
        "bytes_written": 123,
        "io_time_ms": 123
      }
    }
  ]
}
```

关于 **Windows** 上 **system.cpu.util** 监控项的说明

system.cpu.util 监控项提供 CPU 利用率百分比。

当在 Windows 上的 Zabbix agent 中启动一个采集进程时，会为 N 个逻辑处理器（线程）的 N 个监控项初始化一个缓冲区。系统会使用 Windows 性能计数器定期更新缓冲区中的值。当请求 `system.cpu.util[n]` 时，将返回这些值，其中 0 <= n < N。

缓冲区中还会额外使用一个监控项用于 `system.cpu.util[total]`。

Zabbix 会自动选择不同的性能计数器，以同时支持 NUMA 系统以及不支持处理器组的旧版 Windows。

如果逻辑处理器（线程）数量小于或等于 64，且处理器组数量等于 1，则：

```
N is logical processor (thread) count.

\Processor(_Total)\% Processor Time
\Processor(0)\% Processor Time
\Processor(1)\% Processor Time
\Processor(2)\% Processor Time
...
\Processor(N-1)\% Processor Time
```

否则：

```
G is processor group count and N is logical processor (thread) count in group.

\Processor Information(_Total)\% Processor Time
\Processor Information(0,0)\% Processor Time
\Processor Information(0,1)\% Processor Time
\Processor Information(0,2)\% Processor Time
...
\Processor Information(0,N-1)\% Processor Time
...
\Processor Information(G-1,0)\% Processor Time
\Processor Information(G-1,1)\% Processor Time
\Processor Information(G-1,2)\% Processor Time
...
\Processor Information(G-1,N-1)\% Processor Time
```

大型 **JSON** 值支持

为确保使用 JSON 数据类型（1MiB 或更大）时能够可靠地处理大值，请根据需要检查并应用以下配置更改：

1. 如果您使用的是 MySQL 或 MariaDB，请在 `/etc/mysql/my.cnf` 中增大允许的最大数据包大小（在某些系统上默认可能为 1MB）：

```
[mysqld]
max_allowed_packet=128M

[mysqldump]
max_allowed_packet=1G
```

2. 根据需要调整其他 MySQL/MariaDB 服务器系统变量和 InnoDB 系统变量。例如：

```
innodb_io_capacity=1900
connect_timeout=600
wait_timeout=57600
interactive_timeout=57600
net_read_timeout=7200
net_write_timeout=7200
net-buffer-length=32704
```

3. 将 Zabbix 服务器配置文件中的`HistoryCacheSize` 增加到 2G。

4. 如果您使用的是 Zabbix proxy，请在 proxy 配置文件中将`HistoryCacheSize` 增加到 2G，并将`ProxyMemoryBufferSize` 增加到 2G。

5. 如果您使用的是 Zabbix agent，请增加 Zabbix 服务器/proxy 与 agent 之间的通信超时时间（例如，监控项超时 或 Zabbix 服务器/proxy 和agent 配置文件中的 `Timeout` 参数值）。否则，它们之间的消息可能无法完成传输，您可能会看到诸如 *"message length does not match expected length"* 的错误。此外，还应根据需要检查其他超时设置，例如脚本超时 或监控项测试 超时。

支持的函数

请参见触发器表达式和函数。

宏

可以使用开箱即用的支持的宏 和按位置支持的用户宏。

位置支持的宏

概述

按位置支持的宏表包含在特定位置支持的所有宏（内置宏、表达式宏和用户可定义宏）的参考信息。

如果某个相关组中的所有宏都受支持，则只显示组名。单击每个宏名称或组名称可查看宏的完整详细信息。

宏详情列表包含按应用领域分组的所有内置宏的详细列表。请注意，如需自定义宏值（例如，截短或提取特定子字符串），可以使用宏函数。

位置支持的宏

位置	支持的宏	备注
Action		
时间段条件、默认操作步骤持续时间、步骤持续时间	用户可定义 宏（仅全局）	单个宏必须填满整个字段；不支持多个宏/与文本混用

位置		支持的宏	备注
通知和命令			
	基于触发器	Action 宏 日期/时间 宏 {ESC.HISTORY} 事件 宏、原因/症状事件 宏 事件更新宏：{EVENT.UPDATE.HISTORY}、 {EVENT.UPDATE.STATUS} 表达式宏：{?EXPRESSION} {FUNCTION.VALUE} 主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、 {HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、 {HOST.PORT} 目标主机 宏（仅命令） 主机清单 宏 监控项 宏（除{ITEM.STATE}、{ITEM.STATE.ERROR} 外） proxy 宏 触发器宏：{TRIGGER.DESCRIPTION}、{TRIGGER.EVENTS.*}、 {TRIGGER.EXPRESSION}、{TRIGGER.EXPRESSION.EXPLAIN}、 {TRIGGER.EXPRESSION.RECOVERY}、{TRIGGER.ID}、 {TRIGGER.HOSTGROUP.NAME}、{TRIGGER.NAME}、 {TRIGGER.NAME.ORIG}、{TRIGGER.NSEVERITY}、 {TRIGGER.SEVERITY}、{TRIGGER.STATUS}、 {TRIGGER.TEMPLATE.NAME}、{TRIGGER.URL}、 {TRIGGER.URL.NAME}、{TRIGGER.VALUE} 用户可定义 宏	另见：在消息中使用宏
	问题更新	Action 宏 日期/时间 宏 {ESC.HISTORY} 事件 宏、原因/症状事件 宏、恢复事件 宏 事件更新 宏：全部（除{EVENT.UPDATE.NSEVERITY}、 {EVENT.UPDATE.SEVERITY} 外） 表达式宏：{?EXPRESSION} 函数 宏 主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、 {HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、 {HOST.PORT} 目标主机 宏（仅命令） 主机清单 宏 监控项 宏（除{ITEM.STATE}、{ITEM.STATE.ERROR} 外） proxy 宏 触发器宏：{TRIGGER.DESCRIPTION}、{TRIGGER.EVENTS.*}、 {TRIGGER.EXPRESSION}、{TRIGGER.EXPRESSION.EXPLAIN}、 {TRIGGER.EXPRESSION.RECOVERY}、{TRIGGER.ID}、 {TRIGGER.HOSTGROUP.NAME}、{TRIGGER.NAME}、 {TRIGGER.NAME.ORIG}、{TRIGGER.NSEVERITY}、 {TRIGGER.SEVERITY}、{TRIGGER.STATUS}、 {TRIGGER.TEMPLATE.NAME}、{TRIGGER.URL}、 {TRIGGER.URL.NAME}、{TRIGGER.VALUE} {USER.FULLNAME} 用户可定义 宏	
	问题恢复	恢复事件 宏 事件 宏 {FUNCTION.RECOVERY.VALUE} {TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}	

位置		支持的宏	备注
	发现	Action 宏 日期/时间 宏 发现 宏 事件宏：{EVENT.AGE}、{EVENT.DATE}、{EVENT.ID}、 {EVENT.OBJECT}、{EVENT.SOURCE}、{EVENT.TIME}、 {EVENT.TIMESTAMP} 目标主机 宏（仅命令） proxy 宏	
	自动注册	Action 宏 日期/时间 宏 事件宏：{EVENT.AGE}、{EVENT.DATE}、{EVENT.ID}、 {EVENT.OBJECT}、{EVENT.SOURCE}、{EVENT.TIME}、 {EVENT.TIMESTAMP} 主机宏：{HOST.HOST}、{HOST.IP}、{HOST.PORT}、 {HOST.METADATA} 目标主机 宏（仅命令） proxy 宏	
	基于服务	Action 宏 日期/时间 宏 {ESC.HISTORY} 事件宏：全部，除{EVENT.ACK.STATUS}、{EVENT.OPDATA} 外 服务 宏 用户可定义 宏	
	服务恢复	事件 宏（除{EVENT.ACK.STATUS}、{EVENT.OPDATA} 外）、恢复事件 宏	
	服务更新	Action 宏 日期/时间 宏 {ESC.HISTORY} 事件 宏（除{EVENT.ACK.STATUS}、{EVENT.OPDATA} 外） 事件更新宏：{EVENT.UPDATE.DATE}、 {EVENT.UPDATE.NSEVERITY}、{EVENT.UPDATE.SEVERITY}、 {EVENT.UPDATE.STATUS}、{EVENT.UPDATE.TIME}、 {EVENT.UPDATE.TIMESTAMP} 服务 宏 用户可定义 宏	
内部通知		Action 宏 日期/时间 宏 {ESC.HISTORY} 事件 宏（除{EVENT.NSEVERITY}、{EVENT.SEVERITY} 外）、恢复事件 宏 主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、 {HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 监控项宏：{ITEM.DESCRIPTION}、{ITEM.DESCRIPTION.ORIG}、 {ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG}、{ITEM.NAME}、 {ITEM.NAME.ORIG}、{ITEM.VALUETYPE} proxy 宏	
	基于监控项	监控项宏：{ITEM.STATE}、{ITEM.STATE.ERROR}	
	基于 *LLD* 规则	低级别发现通知 宏	
	基于触发器	触发器宏：{TRIGGER.DESCRIPTION}、{TRIGGER.EXPRESSION}、 {TRIGGER.EXPRESSION.RECOVERY}、 {TRIGGER.HOSTGROUP.NAME}、{TRIGGER.ID}、 {TRIGGER.NAME}、{TRIGGER.NAME.ORIG}、 {TRIGGER.NSEVERITY}、{TRIGGER.SEVERITY}、 {TRIGGER.STATE}、{TRIGGER.STATE.ERROR}、 {TRIGGER.TEMPLATE.NAME}、{TRIGGER.URL}、 {TRIGGER.URL.NAME} 用户可定义 宏	

位置		支持的宏	备注
告警脚本参数		告警 宏 用户可定义 宏（仅全局）	
Connector			
	URL、用户名、密码、*Bearer token*、*HTTP proxy*、*SSL* 证书、*SSL* 密钥文件、*SSL* 密钥密码字段	用户可定义 宏（仅全局）	
	Timeout 字段	用户可定义 宏（仅全局）	单个宏必须填满整个字段；不支持多个宏/与文本混用。
图形名称		表达式宏：{?EXPRESSION}	此位置的该宏仅支持一个以秒为参数的 **avg**、**last**、**max**、**min** 函数（可使用时间后缀）。 可在该宏中使用 {HOST.HOST<1-9>} 宏作为主机（另见带索引的宏）。 用法示例： {?avg(/{HOST.HOST}/item.key,1h)} 不支持的用法： {?last(/host/item1)/last(/host/item2)} {?last(/host/item1)*10} {?count(/host/item1,5m)}
主机、主机原型			
	接口 *IP/DNS*	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME} 用户可定义 宏 [1]	[1] 对于 IP 地址，单个宏必须填满整个字段；不支持多个宏/与文本混用
	接口端口	用户可定义 宏	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	SNMP community *SNMPv3* 上下文名称、安全名称、认证口令、隐私口令	用户可定义 宏	
	IPMI 用户名、密码		
监控项、监控项原型、**LLD** 规则			
	名称	用户可定义 宏	LLD 规则名称中不支持。
	描述	用户可定义 宏	

位置		支持的宏	备注
监控项类型	键值参数	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 用户可定义 宏	支持的 {HOST.*} 宏将解析为为监控项选择的接口。当用于没有接口的监控项时，它们将按优先级顺序解析为主机的 Zabbix agent、SNMP、JMX 或 IPMI 接口；如果主机没有任何接口，则解析为'UNKNOWN'。 Zabbix trapper 监控项的监控项键值中不支持宏。
	更新间隔、自定义间隔、*Timeout*（适用于受支持的监控项类型）、最多保留（历史/趋势）字段	用户可定义 宏	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	删除丢失的资源、禁用丢失的资源字段	用户可定义 宏	这些字段仅支持 LLD 规则。 单个宏必须填满整个字段；不支持多个宏/与文本混用。
	预处理步骤参数（包括自定义脚本）、 自定义错误处理参数 (设置值为和 设置错误为字段)	用户可定义 宏	
	过滤正则表达式	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 用户可定义 宏	这些字段仅支持 LLD 规则。
	覆盖过滤正则表达式	用户可定义 宏	这些字段仅支持 LLD 规则。
	覆盖操作：更新间隔、历史数据保留期、趋势数据保留期（适用于监控项原型）	用户可定义 宏	这些字段仅支持 LLD 规则。 单个宏必须填满整个字段；不支持多个宏/与文本混用。
	Browser 监控项 参数名称和值	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 监控项宏：{ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG} 用户可定义 宏	
	Script 监控项 参数名称和值		
	Calculated/aggregate 监控项公式 (表达式常量、函数参数、监控项键值参数，(仅聚合监控项）过滤条件（主机组名称和标签名称）)	用户可定义 宏	
	Database monitor 监控项 用户名、密码、*SQL* 查询字段	用户可定义 宏	

位置		支持的宏	备注
	HTTP agent 监控项 *URL*、查询字段、请求体、标头（名称和值）、*SSL* 证书文件、*SSL* 密钥文件字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 监控项宏：{ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG} 用户可定义 宏	包含秘密宏的 URL 将无法工作，因为其中的宏会被解析为"******"
	HTTP agent 监控项 *HTTP Proxy* 字段	监控项宏：{ITEM.ID}、{ITEM.KEY}、{ITEM.KEY.ORIG} 用户可定义 宏	
	HTTP agent 监控项所需状态码、*HTTP* 认证用户名/密码、*SSL* 密钥密码字段	用户可定义 宏	
	HTTP agent 监控项允许的主机字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME} 用户可定义 宏	
	JMX agent 监控项 *Endpoint* 字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.PORT} 用户可定义 宏	
	Trapper 监控项允许的主机字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP} 用户可定义 宏	
	SNMP agent 监控项 *SNMP OID* 字段	用户可定义 宏	
	SSH and Telnet 监控项 脚本	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 用户可定义 宏	
	SSH agent 监控项用户名、公钥文件、私钥文件、密码字段	用户可定义 宏	
	Telnet agent 监控项用户名、密码字段	用户可定义 宏	
地图			
	URL、*URL* 名称	主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} {HOSTGROUP.ID} 主机清单 宏 地图 宏 {TRIGGER.ID}	
	元素标签	表达式宏：{?EXPRESSION} [1] 主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} {HOSTGROUP.ID} 主机清单 宏 地图 宏 触发器宏：{TRIGGER.EVENTS.*}、{TRIGGER.ID}、{TRIGGER.PROBLEM.EVENTS.*}、{TRIGGERS.ACK}、{TRIGGERS.PROBLEM.ACK}、{TRIGGERS.PROBLEM.UNACK}、{TRIGGERS.UNACK}	[1] 此位置的该宏仅支持一个以秒为参数的 **avg**、**last**、**max**、**min** 函数（可使用时间后缀）。 可在该宏中使用 {HOST.HOST<1-9>} 宏作为主机。 用法示例： {?avg(/{HOST.HOST}/item.key,1h)} 不支持的用法： {?last(/host/item1)/last(/host/item2)} {?last(/host/item1)*10} {?count(/host/item1,5m)}
	链接标签	表达式宏：{?EXPRESSION} [1]	
	形状标签	表达式宏：{?EXPRESSION} [1]	
	地图形状中的文本字段	{MAP.NAME}	
网络发现			

位置		支持的宏	备注
	更新间隔字段	用户可定义 宏（仅全局）	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	SNMP community、*SNMP OID* 字段 *SNMPv3* 上下文名称、安全名称、认证口令、隐私口令字段	用户可定义 宏（仅全局）	
proxy			
	活动 *agent* 的地址 > 端口（如果 proxy 属于某个组） 对于被动 proxy 模式：接口 地址和 端口字段 覆盖模式下的 监控项类型超时字段	用户可定义 宏（仅全局）	单个宏必须填满整个字段；不支持多个宏/与文本混用。
proxy 组			
	故障转移周期、*proxy* 最小数量字段	用户可定义 宏（仅全局）	单个宏必须填满整个字段；不支持多个宏/与文本混用。
脚本			
	命令、确认文本 (类型：script，手动 主机操作)	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.ID} [1]、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 [2] {MANUALINPUT} 用户名 宏 用户可定义 宏	[1] 仅确认文本 [2] 适用于 Zabbix 服务器和 Zabbix proxy
	命令 (类型：script，手动 事件操作)	日期/时间 宏 事件 宏、原因/症状事件 宏、恢复事件 宏（如果已发生恢复） 事件更新宏：{EVENT.UPDATE.HISTORY}、{EVENT.UPDATE.STATUS} {FUNCTION.VALUE} 主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 监控项 宏（除{ITEM.STATE}、{ITEM.STATE.ERROR} 外） {MANUALINPUT} proxy 宏 触发器宏：{TRIGGER.DESCRIPTION}、{TRIGGER.EVENTS.*}、{TRIGGER.EXPRESSION}、{TRIGGER.EXPRESSION.EXPLAIN}、{TRIGGER.EXPRESSION.RECOVERY}、{TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}、{TRIGGER.HOSTGROUP.NAME}、{TRIGGER.ID}、{TRIGGER.NAME}、{TRIGGER.NAME.ORIG}、{TRIGGER.NSEVERITY}、{TRIGGER.SEVERITY}、{TRIGGER.STATUS}、{TRIGGER.TEMPLATE.NAME}、{TRIGGER.URL}、{TRIGGER.URL.NAME}、{TRIGGER.VALUE} 用户名 宏 用户可定义 宏	
	确认文本 (类型：script，手动 事件操作)	事件宏：{EVENT.ID}、{EVENT.NAME}、{EVENT.NSEVERITY}、{EVENT.SEVERITY}、{EVENT.STATUS}、{EVENT.VALUE} 主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 {MANUALINPUT} 用户名 宏 用户可定义 宏	

位置		支持的宏	备注
	URL、确认文本 (类型：URL，手动 主机/事件操作)		
标签名称和值触发器		主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 监控项宏：{ITEM.LASTVALUE*}、{ITEM.LOG.*}、{ITEM.VALUE*} 触发器宏：{TRIGGER.ID}（仅触发器标签值） 用户可定义 宏	仅在事件生成过程中解析标签名称和值宏。
	名称	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 监控项宏：{ITEM.LASTVALUE*}、{ITEM.LOG.*}、{ITEM.VALUE*} 位置宏/引用：$1...$9 用户可定义 宏	
	事件名称	日期/时间宏：{TIME}、{TIMESTAMP} 表达式宏：{?EXPRESSION} 函数 宏 主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 监控项宏：{ITEM.LASTVALUE*}、{ITEM.LOG.*}、{ITEM.VALUE*} 触发器宏：{TRIGGER.EXPRESSION.EXPLAIN}	
	运维数据	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 监控项宏：{ITEM.LASTVALUE*}、{ITEM.LOG.*}、{ITEM.VALUE*} 用户可定义 宏	
	表达式	{TRIGGER.VALUE} 用户可定义 宏 [1]	[1] 仅用于常量和函数参数；不支持秘密宏
	匹配标签、菜单项名称、菜单项 *URL* 字段	用户可定义 宏	包含秘密宏的 URL 将无法工作，因为其中的宏会被解析为”******”。
	描述	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 监控项宏：{ITEM.LASTVALUE*}、{ITEM.LOG.*}、{ITEM.VALUE*} 用户可定义 宏	
	URL	{EVENT.ID} 主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 监控项宏：{ITEM.LASTVALUE*}、{ITEM.LOG.*}、{ITEM.VALUE*} {TRIGGER.ID}	
用户参数命令		位置宏/引用：$1...$9	
Web 场景			
	名称、*Agent*、*HTTP proxy* 字段	用户可定义 宏	
	更新间隔字段	用户可定义 宏	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	变量、标头、*SSL* 证书、*SSL* 密钥文件字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 用户可定义 宏	
	用户、密码、*SSL* 密钥密码字段	用户可定义 宏	

位置		支持的宏	备注
	Web 场景步骤 名称、变量（仅值）字段	用户可定义 宏	
	Web 场景步骤 *Timeout*、所需状态码字段	用户可定义 宏	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	Web 场景步骤 *URL*、标头（名称和值）、所需字符串字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 用户可定义 宏	
	Web 场景步骤 *Post* 字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.IP}、{HOST.NAME}、{HOST.PORT}	
webhook 媒体类型			
	参数值	告警 宏 所有在基于触发器的问题通知中支持的宏（见上文）	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	JavaScript 脚本、*JavaScript* 脚本参数名 菜单项名称、菜单项 *URL*	用户可定义 宏（仅全局） {EVENT.TAGS.<tag name>}	
小组件			
	Gauge 小组件 描述字段	主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 监控项 宏（除{ITEM.STATE.ERROR} 外） 用户可定义 宏（仅全局）	
	Honeycomb 小组件 主/副标签字段 监控项值小组件 描述字段		
	Top hosts 小组件 文本数据列	主机宏：{HOST.CONN}、{HOST.DESCRIPTION}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 主机清单 宏 用户可定义 宏（仅全局）	
	URL（动态）小组件 *URL* 字段	主机宏：{HOST.CONN}、{HOST.DNS}、{HOST.HOST}、{HOST.ID}、{HOST.IP}、{HOST.NAME}、{HOST.PORT} 用户可定义 宏	
自定义文本列			
	监控 > 问题	主机 宏（除{HOST.METADATA} 外） 主机清单 宏 监控项 宏（除{ITEM.STATE.ERROR} 外） 触发器宏：{TRIGGER.ID} 事件宏：{EVENT.ID} 用户可定义 宏	
	监控 > 主机	主机 宏（除{HOST.METADATA} 外） 主机清单 宏 用户可定义 宏	

位置		支持的宏	备注
其他位置	监控 > 最新数据	主机 宏（除{HOST.METADATA} 外） 主机清单 宏 监控项 宏（除{ITEM.STATE.ERROR} 外） 用户可定义 宏	
	数据采集 > 模板	主机宏：{HOST.DESCRIPTION}、{HOST.HOST}、{HOST.ID}、{HOST.NAME}	
	数据采集 > 主机	主机 宏（除{HOST.METADATA} 外） 主机清单 宏 用户可定义 宏	
	工作时间字段	用户可定义 宏（仅全局）	单个宏必须填满整个字段；不支持多个宏/与文本混用。
	Timeouts for item types 字段（全局设置）		
	用户媒介启用时字段	用户可定义 宏（仅全局）	
	Email 媒体类型用户名、密码字段	用户可定义 宏（仅全局）	推荐使用秘密宏。
	计划报告主题、消息字段	日期/时间宏：{TIME}、{TIMESTAMP}	

宏详细信息

此列表包含按应用领域分组的每个内置宏的详细信息：

- 动作宏
- 告警宏
- 日期/时间宏
- 发现宏
- 事件宏
  - 事件更新宏
  - 原因/症状事件宏
  - 恢复事件宏
- 升级宏
- 函数宏
- 主机宏
  - 目标主机宏
- 主机组宏
- 主机清单宏
- 监控项宏
- 低级别发现通知宏
- 拓扑图宏
- Proxy 宏
- 脚本宏
- 服务宏
- 触发器宏
- 用户名宏

动作宏

{ACTION.ID}

已触发动作的数字 ID。

{ACTION.NAME}

已触发动作的名称。

告警宏

{ALERT.MESSAGE}

操作配置中的默认消息值。

{ALERT.SENDTO}

用户介质配置中的 *Send to* 值。

{ALERT.SUBJECT}

操作配置中的默认主题值。

日期和时间宏

{DATE}

当前日期，格式为 `yyyy.mm.dd`。

{TIME}

当前时间，格式为 `hh:mm:ss`。

{TIMESTAMP}

当前时间的 UNIX 时间戳格式。

发现宏

{DISCOVERY.DEVICE.IPADDRESS}

已发现设备的 IP 地址。
 始终可用，不依赖于是否已添加主机。

{DISCOVERY.DEVICE.DNS}

已发现设备的 DNS 名称。
 始终可用，不依赖于是否已添加主机。

{DISCOVERY.DEVICE.STATUS}

已发现设备的状态（UP/DOWN）。

{DISCOVERY.DEVICE.UPTIME}

特定设备自上次发现状态变更以来的时间。
 例如：1h 29m 01s。
 对于状态为 DOWN 的设备，这是其宕机持续时间。

{DISCOVERY.RULE.NAME}

发现设备/服务存在或不存在的发现规则名称。

{DISCOVERY.SERVICE.NAME}

已发现服务的名称。
 例如：HTTP。

{DISCOVERY.SERVICE.PORT}

已发现服务的端口。
 例如：80。

{DISCOVERY.SERVICE.STATUS}

已发现服务的状态（UP/DOWN）。

{DISCOVERY.SERVICE.UPTIME}

特定服务自上次发现状态变更以来的时间。
 例如：1h 29m 01s。
 对于状态为 DOWN 的服务，这是其停机时长。

升级宏

{ESC.HISTORY}

先前已发送通知的日志，包括其升级步骤和状态（已发送、进行中或 失败）。

事件宏

{EVENT.ACK.STATUS}

事件的确认状态（是/否）。

{EVENT.AGE}

触发操作的事件的持续时间，精确到秒。
 在升级消息中很有用。例如：9m 13s。

{EVENT.DATE}

触发动作的事件日期，格式为 `yyyy.mm.dd`。
 例如：2025.04.14。

{EVENT.DURATION}

事件的持续时间（问题事件与恢复事件之间的时间差），精确到秒。
 适用于问题恢复消息。

{EVENT.ID}

触发动作的事件的数字 ID。

{EVENT.NAME}

触发操作的事件名称。

{EVENT.NSEVERITY}

事件严重性的数值。
 可能的值：*0* - 未分类，*1* - 信息，*2* - 警告，*3* - 一般严重，*4* - 高严重，*5* - 灾难。

{EVENT.OBJECT}

事件对象的数值。
 可能的值：*0* - 触发器，*1* - 已发现主机，*2* - 已发现服务，*3* - 自动注册，*4* - 监控项，*5* - 低级别发现规则。

{EVENT.OPDATA}

问题对应的底层触发器的运行数据。

{EVENT.SEVERITY}

事件严重性的名称。

{EVENT.SOURCE}

事件来源的数值。
 可能的值：*0* - 触发器，*1* - 发现，*2* - 自动注册，*3* - 内部，*4* - 服务。

{EVENT.STATUS}

触发动作的事件的文字状态。

{EVENT.TAGS}

事件标签的逗号分隔列表。
 如果不存在标签，则展开为空字符串。

{EVENT.TAGSJSON}

包含事件标签对象的 JSON 数组。
 如果不存在标签，则展开为空数组。

{EVENT.TAGS.<tag name>}

由标签名称引用的事件标签值。
 包含非字母数字字符的标签名称（包括非英语多字节 UTF 字符）必须用双引号括起来。带引号的标签名称中的引号和反斜杠必须用反斜杠转义。

{EVENT.TIME}

触发动作的事件时间，格式为 `hh:mm:ss`。
 例如：12:57:53。

{EVENT.TIMESTAMP}

触发动作的事件的 UNIX 时间戳。

{EVENT.VALUE}

触发动作的事件的数值。
 可能的值：*1* - 故障，*0* - 恢复中。

原因/症状事件宏

{EVENT.CAUSE.ACK.STATUS}

原因事件的确认状态（是/否）。
 在症状事件的上下文中使用。

{EVENT.CAUSE.AGE}

原因事件的持续时间，精确到秒。
 在升级消息中很有用。
 用于症状事件的上下文中。

{EVENT.CAUSE.DATE}

原因事件的日期，格式为 `yyyy.mm.dd`。
 用于症状事件的上下文中。

{EVENT.CAUSE.DURATION}

原因事件的持续时间（问题事件与恢复事件之间的时间差），精确到秒。
 在问题恢复消息中很有用。
 用于症状事件的上下文中。

{EVENT.CAUSE.ID}

原因事件的数字 ID。
 在症状事件的上下文中使用。

{EVENT.CAUSE.NAME}

原因事件的名称。
 在症状事件的上下文中使用。

{EVENT.CAUSE.NSEVERITY}

原因事件严重性的数值。
 可能的值：*0* - 未分类，*1* - 信息，*2* - 警告，*3* - 一般严重，*4* - 高，*5* - 灾难。
 用于症状事件的上下文中。

{EVENT.CAUSE.OBJECT}

原因事件对象的数值。
 可能的值：*0* - 触发器，*1* - 已发现主机，*2* - 已发现服务，*3* - 自动注册，*4* - 监控项，*5* - 低级别发现规则。
 用于症状事件的上下文中。

{EVENT.CAUSE.OPDATA}

原因问题的底层触发器的运行数据。
 用于症状事件的上下文中。

{EVENT.CAUSE.SEVERITY}

原因事件严重性的名称。
 可能的值：未分类、信息、警告、一般严重、高严重、灾难。
 用于症状事件的上下文中。

{EVENT.CAUSE.SOURCE}

原因事件来源的数值。
 可能的值：*0* - 触发器，*1* - 发现，*2* - 自动注册，*3* - 内部。
 用于症状事件的上下文中。

{EVENT.CAUSE.STATUS}

原因事件的文字值。
 用于症状事件的上下文中。

{EVENT.CAUSE.TAGS}

原因事件标签的逗号分隔列表。
 如果不存在标签，则展开为空字符串。
 用于症状事件的上下文中。

{EVENT.CAUSE.TAGSJSON}

一个 JSON 数组，包含原因事件标签对象。
 如果不存在标签，则展开为空数组。
 用于症状事件的上下文中。

{EVENT.CAUSE.TAGS.<tag name>}

由标签名称引用的原因事件标签值。
 包含非字母数字字符的标签名称（包括非英语多字节 UTF 字符）必须用双引号括起来。带引号的标签名称中的引号和反斜杠必须使用反斜杠进行转义。
 用于症状事件的上下文中。

{EVENT.CAUSE.TIME}

原因事件的时间，格式为 `hh:mm:ss`。
 用于症状事件的上下文中。

{EVENT.CAUSE.TIMESTAMP}

原因事件的 UNIX 时间戳。
 在症状事件的上下文中使用。

{EVENT.CAUSE.UPDATE.HISTORY}

原因问题更新日志（确认等）。
 用于症状事件的上下文中。

{EVENT.CAUSE.VALUE}

原因事件的数值。
 可能的值：*1* - 问题，*0* - 恢复中。
 在症状事件的上下文中使用。

{EVENT.SYMPTOMS}

症状事件列表。包括以下详细信息：主机名、事件名称、严重性、持续时间、服务标签和值。
 此宏用于原因事件的上下文中，并返回有关症状事件的信息。

恢复事件宏

{EVENT.RECOVERY.DATE}

恢复事件的日期，格式为 `yyyy.mm.dd`。

{EVENT.RECOVERY.ID}

恢复事件的数字 ID。

{EVENT.RECOVERY.NAME}

恢复事件的名称。

{EVENT.RECOVERY.STATUS}

恢复事件的文字值。

{EVENT.RECOVERY.TAGS}

恢复事件标签的逗号分隔列表。如果不存在标签，则展开为空字符串。

{EVENT.RECOVERY.TAGSJSON}

包含恢复事件标签对象的 JSON 数组。如果不存在标签，则展开为空数组。

{EVENT.RECOVERY.TIME}

恢复事件的时间，格式为 `hh:mm:ss`。

{EVENT.RECOVERY.TIMESTAMP}

恢复事件的 UNIX 时间戳。

{EVENT.RECOVERY.VALUE}

恢复事件的数值。

事件更新宏

{EVENT.UPDATE.ACTION}

在问题更新期间执行的操作的人类可读名称。
 可解析为以下值：*acknowledged*、*unacknowledged*、*commented*、*changed severity from (original severity) to (updated severity)* 和 *closed*（取决于一次更新中执行了多少个操作）。

{EVENT.UPDATE.ACTIONJSON}

一个 JSON 数组，包含在问题更新期间执行的操作详情。
 可能的 JSON 属性值：
- true（用于 `acknowledge`、`unacknowledge`、`close`、`unsuppress`、`cause` 和 `symptom` 属性）；
- <message string>（用于 `message` 属性）；
- 时间戳（用于 `suppress_until` 和 `timestamp` 属性），或 0（如果无限期抑制，则用于 `suppress_until` 属性）；
- 0、1、2、3、4、5（用于 `old` 和 `new` 严重性属性）。

 例如：`{"acknowledge":true,"message":"Monthly maintenance.","severity":{"old":2,"new":1},"suppress_until":1730851199,"timestamp":1730822048}`。

{EVENT.UPDATE.DATE}

事件更新（确认等）的日期，格式为 `yyyy.mm.dd`。

{EVENT.UPDATE.HISTORY}

问题更新（确认等）的日志。

{EVENT.UPDATE.MESSAGE}

问题更新消息。

{EVENT.UPDATE.NSEVERITY}

问题更新操作期间设置的新事件严重性的数值。

{EVENT.UPDATE.SEVERITY}

在问题更新操作期间设置的新事件严重性的名称。

{EVENT.UPDATE.STATUS}

问题更新状态的数值。
 可能的值：*0* - 因问题/恢复事件而调用 webhook，*1* - 更新操作。

{EVENT.UPDATE.TIME}

事件更新（确认等）的时间，格式为 `hh:mm:ss`。

{EVENT.UPDATE.TIMESTAMP}

事件更新（确认等）的 UNIX 时间戳。

函数宏

{FUNCTION.VALUE}

事件发生时，触发器表达式中第 N 个基于监控项的函数的值。
 仅统计以 `/host/key` 作为第一个参数的函数。

 此宏可与数字索引一起使用，例如 {FUNCTION.VALUE<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等函数。请参见indexed macros。

{FUNCTION.RECOVERY.VALUE}

事件发生时，恢复表达式中第 N 个基于监控项的函数的值。*
 仅统计第一个参数为 `/host/key` 的函数。

 此宏可与数字索引一起使用，例如 {FUNCTION.RECOVERY.VALUE<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等函数。请参见indexed macros。

主机宏

{HOST.CONN}

主机 IP 地址或 DNS 名称，具体取决于主机设置。
 在全局脚本、接口 IP/DNS 字段和 web 场景中，该宏将解析为主 agent 接口。如果未定义 agent 接口，则将使用主 SNMP 接口。如果也未定义 SNMP 接口，则将使用主 JMX 接口。如果 JMX 接口也未定义，则将使用主 IPMI 接口。如果主机没有任何接口，则该宏解析为'UNKNOWN'。

 该宏可与数字索引一起使用，例如 {HOST.CONN<1-9>}，用于在触发器表达式中指向第一台、第二台、第三台等主机。请参见indexed macros。

{HOST.DESCRIPTION}

主机描述。

 此宏可与数字索引一起使用，如 {HOST.DESCRIPTION<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{HOST.DNS}

主机 DNS 名称。
 在全局脚本、接口 IP/DNS 字段和 web 场景中，该宏将解析为主 agent 接口。如果未定义 agent 接口，则将使用主 SNMP 接口。如果也未定义 SNMP 接口，则将使用主 JMX 接口。如果未定义 JMX 接口，则将使用主 IPMI 接口。如果主机没有任何接口，则该宏解析为'UNKNOWN'。

 该宏可与数字索引一起使用，如 {HOST.DNS<1-9>}，以在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{HOST.HOST}

技术主机名称。
 如果该宏作为history function 的第一个参数中的占位符使用，则不支持Macro functions，例如 `last(/{HOST.HOST}/{ITEM`

 该宏可与数字索引一起使用，格式为 {HOST.HOST<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{HOST.ID}

主机 ID。

 此宏可与数字索引一起使用，如 {HOST.ID<1-9>}，以在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{HOST.IP}

主机 IP 地址。
 在全局脚本、接口 IP/DNS 字段和 web 场景中，该宏将解析为主 agent 接口。如果未定义 agent 接口，则将使用主 SNMP 接口。如果也未定义 SNMP 接口，则将使用主 JMX 接口。如果 JMX 接口也未定义，则将使用主 IPMI 接口。如果主机没有任何接口，则该宏解析为'UNKNOWN'。

 该宏可与数字索引一起使用，格式为 {HOST.IP<1-9>}，用于在触发器表达式中指向第一台、第二台、第三台等主机。请参见indexed macros。

{HOST.METADATA}

主机元数据。

 仅用于活动 agent 自动注册。

{HOST.NAME}

可见的主机名称。

 此宏可与数字索引一起使用，格式为 {HOST.NAME<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{HOST.PORT}

主机（agent）端口。
 在全局脚本、接口 IP/DNS 字段和 web 场景中，该宏将解析为主 agent 接口。如果未定义 agent 接口，则将使用主 SNMP 接口。如果也未定义 SNMP 接口，则将使用主 JMX 接口。如果 JMX 接口也未定义，则将使用主 IPMI 接口。如果主机没有任何接口，则该宏解析为'UNKNOWN'。

 此宏可与数字索引一起使用，如 {HOST.PORT<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

目标主机宏

{HOST.TARGET.CONN}

目标主机的 IP 地址或 DNS 名称。

{HOST.TARGET.DNS}

目标主机的 DNS 名称。

{HOST.TARGET.HOST}

目标主机的技术名称。

{HOST.TARGET.IP}

目标主机的 IP 地址。

{HOST.TARGET.NAME}

目标主机的可见名称。

主机组宏

{HOSTGROUP.ID}

主机组 ID。

主机资产记录宏

{INVENTORY.ALIAS}

主机清单中的 *Alias* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.ALIAS<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.ASSET.TAG}

主机清单中的 资产标签字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.ASSET.TAG<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.CHASSIS}

主机库存中的 机箱字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.CHASSIS<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.CONTACT}

主机清单中的 *Contact* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.CONTACT<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.CONTRACT.NUMBER}

主机清单中的 合同编号字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.CONTRACT.NUMBER<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.DEPLOYMENT.STATUS}

主机清单中的 部署状态字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.DEPLOYMENT.STATUS<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HARDWARE}

主机清单中的 *Hardware* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HARDWARE<1-9>}，以在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HARDWARE.FULL}

主机库存中的 硬件（完整详情）字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HARDWARE.FULL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HOST.NETMASK}

主机清单中的 主机子网掩码字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HOST.NETMASK<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HOST.NETWORKS}

主机库存中的 *Host networks* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HOST.NETWORKS<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HOST.ROUTER}

主机清单中的 *Host router* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HOST.ROUTER<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HW.ARCH}

主机清单中的 硬件架构字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HW.ARCH<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HW.DATE.DECOMM}

主机清单中的 硬件退役日期字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HW.DATE.DECOMM<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HW.DATE.EXPIRY}

主机清单中的 硬件维护到期日期字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HW.DATE.EXPIRY<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HW.DATE.INSTALL}

主机清单中的 硬件安装日期字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HW.DATE.INSTALL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.HW.DATE.PURCHASE}

主机清单中的 硬件购买日期字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.HW.DATE.PURCHASE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.INSTALLER.NAME}

主机清单中的 安装程序名称字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.INSTALLER.NAME<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.LOCATION}

主机清单中的 *Location* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.LOCATION<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.LOCATION.LAT}

主机清单中的 位置纬度字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.LOCATION.LAT<1-9>}，用于在触发器表达式中指向第一台、第二台、第三台等主机。请参见indexed macros。

{INVENTORY.LOCATION.LON}

主机清单中的 位置经度字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.LOCATION.LON<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.MACADDRESS.A}

主机库存中的 *MAC* 地址 *A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.MACADDRESS.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.MACADDRESS.B}

主机库存中的 *MAC* 地址 *B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.MACADDRESS.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.MODEL}

主机清单中的 *Model* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.MODEL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.NAME}

主机清单中的 *Name* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.NAME<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.NOTES}

主机清单中的 *Notes* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.NOTES<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.OOB.IP}

主机清单中的 *OOB IP* 地址字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.OOB.IP<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.OOB.NETMASK}

主机清单中的 *OOB* 子网掩码字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.OOB.NETMASK<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.OOB.ROUTER}

主机清单中的 *OOB* 路由器字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.OOB.ROUTER<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.OS}

主机清单中的 *OS* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.OS<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.OS.FULL}

主机清单中的 操作系统（完整详情）字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.OS.FULL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.OS.SHORT}

主机清单中的 *OS (Short)* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.OS.SHORT<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.PRIMARY.CELL}

主机清单中的 主要 *POC* 单元字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.CELL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.PRIMARY.EMAIL}

主机清单中的 主要 *POC* 电子邮件字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.EMAIL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见索引宏。

{INVENTORY.POC.PRIMARY.NAME}

主机清单中的 主要 *POC* 名称字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.NAME<1-9>}，以在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.PRIMARY.NOTES}

主机清单中的 主要联系人备注字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.NOTES<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.PRIMARY.PHONE.A}

主机库存中的 *Primary POC phone A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.PHONE.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.PRIMARY.PHONE.B}

主机清单中的 *Primary POC phone B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.PHONE.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.PRIMARY.SCREEN}

主机清单中的 主要 *POC* 屏幕名称字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.PRIMARY.SCREEN<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.CELL}

主机清单中的 *Secondary POC cell* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.CELL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.EMAIL}

主机库存中的 *Secondary POC email* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.EMAIL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.NAME}

主机清单中的 *Secondary POC name* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.NAME<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.NOTES}

主机清单中的 *Secondary POC notes* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.NOTES<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.PHONE.A}

主机清单中的 *Secondary POC phone A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.PHONE.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.PHONE.B}

主机库存中的 *Secondary POC phone B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.PHONE.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.POC.SECONDARY.SCREEN}

主机清单中的 *Secondary POC screen name* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.POC.SECONDARY.SCREEN<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SERIALNO.A}

主机库存中的 序列号 *A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SERIALNO.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SERIALNO.B}

主机清单中的 *Serial number B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SERIALNO.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.ADDRESS.A}

主机清单中的 *Site address A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.ADDRESS.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.ADDRESS.B}

主机清单中的 *Site address B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.ADDRESS.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.ADDRESS.C}

主机清单中的 *Site address C* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.ADDRESS.C<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.CITY}

主机库存中的 *Site city* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.CITY<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.COUNTRY}

主机清单中的 站点国家/地区字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.COUNTRY<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.NOTES}

主机清单中的 站点备注字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.NOTES<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.RACK}

主机清单中的 站点机架位置字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.RACK<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.STATE}

主机库存中的 *Site state/province* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.STATE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SITE.ZIP}

主机清单中的 *Site ZIP/postal* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SITE.ZIP<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE}

主机库存中的 *Software* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE.APP.A}

主机清单中的 软件应用 *A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE.APP.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE.APP.B}

主机清单中的 *Software application B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE.APP.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE.APP.C}

主机清单中的 *Software application C* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE.APP.C<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE.APP.D}

主机清单中的 *Software application D* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE.APP.D<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE.APP.E}

主机清单中的 *Software application E* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE.APP.E<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.SOFTWARE.FULL}

主机清单中的 软件（完整详情）字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.SOFTWARE.FULL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.TAG}

主机库存中的 *Tag* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.TAG<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.TYPE}

主机清单中的 *Type* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.TYPE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.TYPE.FULL}

主机库存中的 类型（完整详情）字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.TYPE.FULL<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.URL.A}

主机清单中的 *URL A* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.URL.A<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.URL.B}

主机清单中的 *URL B* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.URL.B<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.URL.C}

主机清单中的 *URL C* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.URL.C<1-9>}，以在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

{INVENTORY.VENDOR}

主机清单中的 *Vendor* 字段。

 此宏可与数字索引一起使用，例如 {INVENTORY.VENDOR<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等主机。请参见indexed macros。

监控项宏

{ITEM.DESCRIPTION}

导致通知的触发器表达式中第 N 个监控项的描述。

 此宏可与数字索引一起使用，格式为 {ITEM.DESCRIPTION<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等主机。请参见indexed macros。

{ITEM.DESCRIPTION.ORIG}

导致通知的触发器表达式中第 N 个监控项的原始描述(宏未解析)。

 此宏可与数字索引一起使用，例如 {ITEM.DESCRIPTION.ORIG<1-9>}，以指向触发器表达式中的第一个、第二个、第三个等主机。请参见indexed macros。

{ITEM.ID}

导致通知的触发器表达式中第 N 个监控项的数字 ID。

 此宏可与数字索引一起使用，如 {ITEM.ID<1-9>}，以指向触发器表达式中的第一个、第二个、第三个等主机。请参见indexed macros。

{ITEM.KEY}

导致通知的触发器表达式中第 N 个监控项的键值。
 如果此宏作为历史函数 第一个参数中的占位符使用，则不支持宏函数，例如 `last(/{HOST.HOST}/{ITEM.KEY})`。

 此宏可与数字索引一起使用，格式为 {ITEM.KEY<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等主机。请参见indexed macros。

{ITEM.KEY.ORIG}

导致通知的触发器表达式中第 N 个监控项的原始 key（未解析宏）。

 此宏可与数字索引一起使用，如 {ITEM.KEY.ORIG<1-9>}，以指向触发器表达式中的第一个、第二个、第三个等主机。请参见indexed macros。

{ITEM.LASTVALUE}

导致通知的触发器表达式中第 N 个监控项的最新值。
 如果最新历史值是在 *Max history display period* 时间之前采集的，则在前端中会解析为 *UNKNOWN*（在*Administration > General* 菜单部分中设置）。
 当在问题名称中使用时，在查看问题事件时，该宏不会解析为最新的监控项值；相反，它会保留问题发生时的监控项值。
 当用于通知时，在某些情况下，该宏在触发

器触发的那一刻可能不会解析为最新的监控项值。例如，如果某个监控项快速接收到两个值"A" 和"B"，并且触发器针对"A" 触发，那么由于轻微的处理延迟，通知中可能会将"B" 显示为最新值 - 也就是在触发器触发与通知创建之间，最新的监控项值发生了变化。为避免这种情况，您可以使用 {ITEM.VALUE} 宏，它会解析为触发器触发时刻的值，从而确保通知中使用正确的值。
 它是 `last(/{HOST.HOST}/{ITEM.KEY})` 的别名。
 文本/日志监控项的解析值在以下位置会被前端截断为 20 个字符：
- Operational data;
- Trigger description;
- Trigger URLs;
- Trigger URL labels;
- Description of the item value widget.
 要解析为完整值，您可以使用macro functions，因为服务器不会截断任何值。例如：`{{ITEM.LASTVALUE}.regsub("(.*)", \1)}`。

 此宏可与数字索引一起使用，例如 {ITEM.LASTVALUE<1-9>}，以指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LASTVALUE.AGE}

最新监控项值采集与宏求值之间经过的时间。
 以人类可读的格式显示（例如 1m 45s）。

 此宏可与数字索引一起使用，例如 {ITEM.LASTVALUE.AGE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LASTVALUE.DATE}

收集最新监控项值的日期。
 以 YYYYMMDD 格式显示。
 在触发器名称（位于 *Monitoring* > *Problems* 列表中）、事件名称以及标签名称和值中，使用服务器时区。在其他所有情况下，使用用户时区。

 此宏可与数字索引一起使用，例如 {ITEM.LASTVALUE.DATE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LASTVALUE.TIME}

收集最新监控项值的时间。
 以 HHMMSS 格式显示。
 在触发器名称（位于 *Monitoring* > *Problems* 列表中）、事件名称以及标签名称和值中，使用服务器时区。在其他所有情况下，使用用户时区。

 此宏可与数字索引一起使用，例如 {ITEM.LASTVALUE.TIME<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LASTVALUE.TIMESTAMP}

收集最新监控项值时的 UNIX 时间戳。

 此宏可与数字索引一起使用，例如 {ITEM.LASTVALUE.TIMESTAMP<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.AGE}

日志事件的年龄。
 精确到秒。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.AGE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.DATE}

日志条目写入日志时的日期。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.DATE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.EVENTID}

事件日志中事件的 ID。
 仅适用于 Windows 事件日志监控。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.EVENTID<1-9>}，以指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.NSEVERITY}

事件日志中事件的数值严重性。
 仅适用于 Windows 事件日志监控。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.NSEVERITY<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.SEVERITY}

事件日志中事件的文字严重性。
 仅适用于 Windows 事件日志监控。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.SEVERITY<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.SOURCE}

事件日志中事件的来源。
 仅适用于 Windows 事件日志监控。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.SOURCE<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.TIME}

日志条目写入日志的时间。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.TIME<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.LOG.TIMESTAMP}

写入日志条目时的 UNIX 时间戳。

 此宏可与数字索引一起使用，例如 {ITEM.LOG.TIMESTAMP<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.NAME}

已解析所有宏后的监控项名称。

 此宏可与数字索引一起使用，例如 {ITEM.NAME<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.NAME.ORIG}

监控项的原始名称（未解析宏）。

 此宏可与数字索引一起使用，例如 {ITEM.NAME.ORIG<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.STATE}

导致通知的触发器表达式中第 N 个监控项的最新状态。
 可能的值：*Not supported*、*Normal*。

 此宏可与数字索引一起使用，例如 {ITEM.STATE<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.STATE.ERROR}

监控项变为不支持时，说明其原因的详细错误消息。
 如果监控项进入不支持状态后又立即恢复为支持状态，则错误字段可能为空。

{ITEM.VALUE}

解析为以下两种之一：

- 如果在触发器状态变化的上下文中使用，例如在显示事件或发送通知时，则解析为触发器表达式中第 N 个监控项的历史（事件发生时）值；
- 如果在不涉及触发器状态变化的上下文中使用，例如在弹出选择窗口中显示触发器列表时，则解析为触发器表达式中第 N 个监控项的最新值。在这种情况下，其作用与 {ITEM.LASTVALUE} 相同。

在第一种情况下，如果历史值已经被删除或从未存储过，则会解析为 *UNKNOWN*。
 在第二种情况下，并且仅在前端中，如果最新历史值的采集时间早于 *Max history display period* 时间，则会解析为 *UNKNOWN*（该时间在*Administration > General* 菜单部分中设置）。
 文本/日志监控项的解析值在前端的以下位置会被截断为 20 个字符：
- Operational data;
- Trigger description;
- Trigger URLs;
- Trigger URL labels;
- Description of the item value widget.
 若要解析为完整值，可以使用macro functions，因为服务器不会截断任何值。例如：`{{ITEM.VALUE}.regsub("(.*)", \1)}`。

 此宏可与数字索引一起使用，例如 {ITEM.VALUE<1-9>}，以指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.VALUE.AGE}

监控项值采集与宏求值之间经过的时间。
 以人类可读的格式显示（例如 1m 45s）。

 此宏可与数字索引一起使用，例如 {ITEM.VALUE.AGE<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.VALUE.DATE}

收集监控项值的日期。
 以 `YYYY.MM.DD` 格式显示。
 在触发器名称（位于 *Monitoring* > *Problems* 列表中）、事件名称以及标签名称和值中，使用服务器时区。在其他所有情况下，使用用户时区。

 此宏可与数字索引一起使用，例如 {ITEM.VALUE.DATE<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.VALUE.TIME}

采集监控项值的时间。
 以 `HH:MM:SS` 格式显示。
 在触发器名称（位于 *Monitoring* > *Problems* 列表中）、事件名称以及标签名称和值中，使用服务器时区。在其他所有情况下，使用用户时区。

 此宏可与数字索引一起使用，例如 {ITEM.VALUE.TIME<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.VALUE.TIMESTAMP}

采集监控项值时的 UNIX 时间戳。

 此宏可与数字索引一起使用，例如 {ITEM.VALUE.TIMESTAMP<1-9>}，用于在触发器表达式中指向第一个、第二个、第三个等监控项。请参见indexed macros。

{ITEM.VALUETYPE}

导致通知的触发器表达式中第 N 个监控项的值类型。
 可能的值：*0* - 数值浮点型，*1* - 字符型，*2* - 日志，*3* - 数值无符号型，*4* - 文本。

 此宏可与数字索引一起使用，例如 {ITEM.VALUETYPE<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。请参见indexed macros。

低级发现通知宏

{LLDRULE.DESCRIPTION}

导致通知的低级别发现规则的描述。

{LLDRULE.DESCRIPTION.ORIG}

导致通知的低级别发现规则的原始描述（其中的宏未解析）。

{LLDRULE.ID}

导致通知的低级别发现规则的数字 ID。

{LLDRULE.KEY}

导致通知的低级别发现规则的键值。

{LLDRULE.KEY.ORIG}

导致通知的低级别发现规则的原始键值（宏未解析）。

{LLDRULE.NAME}

导致通知的低级别发现规则的名称。

{LLDRULE.NAME.ORIG}

导致通知的低级别发现规则的原始名称（宏未解析）。

{LLDRULE.STATE}

低级别发现规则的最新状态。
 可能的值：*Not supported*、*Normal*。

 此宏可与数字索引一起使用，例如 {LLDRULE.STATE<1-9>}，用于指向触发器表达式中的第一项、第二项、第三项等监控项。请参见indexed macros。

{LLDRULE.STATE.ERROR}

包含低级发现规则变为不支持原因详细信息的错误消息。
 如果低级发现规则进入不支持状态后又立即恢复为受支持状态，则错误字段可能为空。

地图宏

{MAP.ID}

网络拓扑图 ID。

{MAP.NAME}

网络地图名称。

Proxy 宏

{PROXY.DESCRIPTION}

proxy 描述。
 可解析为以下任一项：

- 触发器表达式中第 N 个监控项所属的 proxy（在基于触发器的通知中）。此处可以使用indexed 宏；
- 执行发现的 proxy（在发现通知中）。此处使用 {PROXY.DESCRIPTION}，不要加索引；
- 活动 agent 注册到的 proxy（在自动注册通知中）。此处使用 {PROXY.DESCRIPTION}，不要加索引。

此宏可与数字索引一起使用，例如 {PROXY.DESCRIPTION<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。另请参见indexed macros。

{PROXY.NAME}

proxy 名称。
 可解析为以下任一项：

- 触发器表达式中第 N 个监控项对应的 proxy（在基于触发器的通知中）。此处可以使用带索引 宏；
- 执行发现的 proxy（在发现通知中）。此处使用 {PROXY.NAME}，不要加索引；
- 活动 agent 注册到的 proxy（在自动注册通知中）。此处使用 {PROXY.NAME}，不要加索引。

此宏可与数字索引一起使用，例如 {PROXY.NAME<1-9>}，用于指向触发器表达式中的第一个、第二个、第三个等监控项。另请参见带索引宏。

脚本宏

{MANUALINPUT}

用户在执行脚本时指定的手动输入值。

服务宏

{SERVICE.DESCRIPTION}

已解析宏的服务描述。

{SERVICE.ID}

触发操作的服务的数字 ID。

{SERVICE.NAME}

解析宏后的服务名称。

{SERVICE.ROOTCAUSE}

导致服务失败的触发器问题事件列表，按严重性和主机名称排序。
 包括以下详细信息：主机名称、事件名称、严重性、持续时间、服务标签和值。

{SERVICE.TAGS}

以逗号分隔的服务事件标签列表。
 服务事件标签可在服务配置的标签部分中定义。如果不存在标签，则展开为空字符串。

{SERVICE.TAGSJSON}

包含服务事件标签对象的 JSON 数组。
 服务事件标签可在服务配置的标签部分中定义。如果不存在标签，则展开为空数组。

{SERVICE.TAGS.<tag name>}

由标签名称引用的服务事件标签值。
 服务事件标签可在服务配置的标签部分中定义。
 包含非字母数字字符的标签名称（包括非英语多字节 UTF 字符）必须用双引号括起来。带引号的标签名称中的引号和反斜杠必须使用反斜杠进行转义。

触发器宏

{TRIGGER.DESCRIPTION}

触发器描述。
 如果在通知文本中使用 {TRIGGER.DESCRIPTION}，则会展开触发器描述中支持的所有宏。

{TRIGGER.EVENTS.ACK}

在拓扑图中，表示拓扑图元素的已确认事件数量；在通知中，表示生成当前事件的触发器的已确认事件数量。

{TRIGGER.EVENTS.PROBLEM.ACK}

所有触发器中已确认的问题事件数量，不考虑其状态。

{TRIGGER.EVENTS.PROBLEM.UNACK}

所有触发器中未确认的问题事件数量，不考虑其状态。

{TRIGGER.EVENTS.UNACK}

地图中某个地图元素的未确认事件数量，或通知中生成当前事件的触发器的未确认事件数量。

{TRIGGER.EXPRESSION}

触发器表达式。

{TRIGGER.EXPRESSION.EXPLAIN}

部分求值的触发器表达式。
 基于监控项的函数会在事件生成时进行计算并替换为结果，而所有其他函数则按表达式中的原样显示。可用于调试触发器表达式。

{TRIGGER.EXPRESSION.RECOVERY}

如果触发器配置 中的 *OK event generation* 设置为 “Recovery expression”，则返回触发器恢复表达式；否则返回空字符串。

{TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}

部分求值的触发器恢复表达式。
 基于监控项的函数会在事件生成时进行求值并替换为结果，而所有其他函数则按表达式中的原样显示。可用于调试触发器恢复表达式。

{TRIGGER.HOSTGROUP.NAME}

定义该触发器的主机组列表，按 SQL 查询排序，并以逗号加空格分隔。

{TRIGGER.ID}

触发此操作的数字触发器 ID。

{TRIGGER.NAME}

已解析宏的触发器名称。
 请注意，可在动作中使用 {EVENT.NAME} 来显示已解析宏的已触发事件/问题名称。

{TRIGGER.NAME.ORIG}

原始触发器名称（宏未解析）。

{TRIGGER.NSEVERITY}

触发器严重性的数值。
 可能的值：*0* - 未分类，*1* - 信息，*2* - 警告，*3* - 一般严重，*4* - 高，*5* - 灾难。

{TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}

处于问题状态的触发器中，已确认的问题事件数量。

{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}

处于问题状态的触发器中未确认的问题事件数量。

{TRIGGER.SEVERITY}

触发器严重性名称。
 可在 *Administration* > *General* > *Trigger displaying options* 中定义。

{TRIGGER.STATE}

触发器表达式的最新状态。
 可能的值：*Unknown*、*Normal*。

{TRIGGER.STATE.ERROR}

触发器变为不支持时，说明其原因的详细错误消息。
 如果触发器进入不支持状态后又立即恢复为支持状态，则错误字段可能为空。

{TRIGGER.STATUS}

执行操作步骤时的触发器值。
 可能的值：*Problem*、*OK*。

{TRIGGER.TEMPLATE.NAME}

按 SQL 查询排序、以逗号加空格分隔的模板列表，其中定义了该触发器；如果该触发器是在主机中定义的，则为 *UNKNOWN*。

{TRIGGER.URL}

触发器 URL。

{TRIGGER.URL.NAME}

触发器 URL 的标签。

{TRIGGER.VALUE}

当前的触发器数值。
 可能的值：*0* - 触发器处于正常状态，*1* - 触发器处于问题状态。

{TRIGGERS.ACK}

地图元素的已确认触发器数量，忽略触发器状态。
 如果某个触发器的所有问题事件都已确认，则该触发器被视为已确认。

{TRIGGERS.PROBLEM.ACK}

地图元素中已确认的问题触发器数量。
 如果某个触发器的所有问题事件都已确认，则该触发器被视为已确认。

{TRIGGERS.PROBLEM.UNACK}

地图元素中未确认问题触发器的数量。
 如果某个触发器至少有一个问题事件未被确认，则该触发器被视为未确认。

{TRIGGERS.UNACK}

地图元素中未确认触发器的数量，忽略触发器状态。
 如果某个触发器至少有一个问题事件未被确认，则该触发器被视为未确认。

用户名宏

{USER.FULLNAME}

添加了事件确认或启动了脚本的用户的姓名、姓氏和用户名。

{USER.NAME}

启动脚本的用户名称。

{USER.SURNAME}

启动该脚本的用户的姓氏。

{USER.USERNAME}

启动该脚本的用户的用户名。

索引宏

索引宏语法 **{MACRO<1-9>}** 只能引用触发器 表达式字段中的第 N 个监控项、函数或主机：

- **{HOST.IP1}, {HOST.IP2}, {HOST.IP3}** 分别解析为触发器表达式中第一个、第二个和第三个主机的 IP 地址（如果存在）。
- **{ITEM.VALUE1}, {ITEM.VALUE2}, {ITEM.VALUE3}** 分别解析为事件发生时触发器表达式中第一个、第二个和第三个监控项的值（如果存在）。
- **{FUNCTION.VALUE1}, {FUNCTION.VALUE2}, {FUNCTION.VALUE3}** 分别解析为事件发生时第一个、第二个和第三个基于监控项的函数的值（如果存在）。

在触发器的上下文中，索引宏始终引用触发器配置中的 表达式字段，而不是 恢复表达式。例如，在恢复事件中，**{ITEM.VALUE2}** 将解析为恢复时问题表达式中第二个监控项的值。

{HOST.HOST<1-9>} 宏也支持在 **graph names** 中的 `{?func(/host/key,param)}` 表达式宏内使用。例如，graph name 中的 `{?func(/{HOST.HOST2}/key,param)}` 将解析为图形中第二个监控项所属的主机。

Warning:
除这里提到的情况外，索引宏不会在任何其他上下文中解析。对于其他上下文，请使用不带索引的宏（**{HOST.HOST}**, **{HOST.IP}** 等）。

位置支持的用户宏

概述

本章节包含支持用户可定义 宏的位置列表。

Note:
对于 动作，网络发现，*Proxy* 和此页的 其他位置部分下列出的所有位置，仅支持全局级别的用户宏。在上述位置，主机级和模板级宏将不会被解析。

Note:
若要自定义宏值（例如，缩短或提取特定的子字符串），你可以使用宏函数。

动作

在动作中，用户宏可用于以下地方：

位置		多个宏/混合文本[1]
基于触发器的通知和命令		支持
基于触发器的内部通知		支持
问题更新通知		支持
基于服务的通知和命令		支持
服务更新通知		支持
时间周期条件		不支持
操作		
	默认操作步骤持续时间	不支持
	步骤持续时间	不支持

主机/主机原型

在主机 和主机原型 配置中，用户宏可用于以下字段：

位置		多个宏/与文本混合[1]
接口 IP/DNS		仅 DNS
接口端口		否
SNMP v1, v2		
	SNMP community	是
SNMP v3		
	上下文名称	是
	安全名称	是
	认证口令	是
	隐私口令	是
IPMI		
	用户名	是
	密码	是
标签[2]		
	标签名称	是
	标签值	是

监控项 / 监控项原型

在监控项或监控项原型配置中，用户宏可用于以下字段：

位置		多
监控项名称		是
监控项键参数		是
更新间隔		否
自定义间隔		否
超时（适用于支持的监控项类型）		否
保留时间（用于历史数据和趋势数据）		否
描述		是
计算/聚合监控项		
	公式（表达式常量和函数参数；监控项键参数；(仅限聚合监控项）筛选条件（主机组名称和标签名称)）	是
数据库监控		
	用户名	是
	密码	是
	SQL 查询	是
HTTP agent		
	URL[3]	是
	查询字段	是
	请求正文	是
	标头（名称和值）	是
	所需状态码	是
	HTTP proxy	是
	HTTP 身份验证用户名	是
	HTTP 身份验证密码	是
	SSI 证书文件	是
	SSI 密钥文件	是
	SSI 密钥密码	是
	允许的主机	是
JMX agent		
	JMX 端点	是
脚本监控项		
	参数名称和值	是
浏览器监控项		
	参数名称和值	是
Telemetry 查询监控项		
	时间偏移	否
	回溯限制	否
	粒度	否
SNMP agent		
	SNMP OID	是
SSH agent		
	用户名	是
	公钥文件	是
	私钥文件	是
	密码	是
	脚本	是
TELNET agent		
	用户名	是
	密码	是
	脚本	是
Zabbix trapper		
	允许的主机	是
标签[2]		
	标签名称	是
	标签值	是
预处理步骤		
	参数（包括自定义脚本）	是
	自定义错误处理参数（设置值为和设置错误为字段）	是

低级发现

在低级发现规则中，用户宏可用于以下字段：

位置		多个宏/与文本混合[1]
键参数		是
更新间隔		否
自定义间隔		否
超时（适用于支持的监控项类型）		否
删除丢失的资源		否
禁用丢失的资源		否
描述		是
SNMP agent		
	SNMP OID	是
SSH agent		
	用户名	是
	公钥文件	是
	私钥文件	是
	密码	是
	脚本	是
TELNET agent		
	用户名	是
	密码	是
	脚本	是
Zabbix trapper		
	允许的主机	是
Database monitor		
	用户名	是
	密码	是
	SQL 查询	是
JMX agent		
	JMX 端点	是
HTTP agent		
	URL[3]	是
	查询字段	是
	请求体	是
	标头（名称和值）	是
	所需状态码	是
	HTTP 认证用户名	是
	HTTP 认证密码	是
过滤器		
	正则表达式	是
覆盖		
	过滤器：正则表达式	是
	操作：更新间隔（用于监控项原型）	否
	操作：历史数据存储期（用于监控项原型）	否
	操作：趋势数据存储期（用于监控项原型）	否

网络发现

在网络发现规则中，用户宏可用于以下地方：

位置		多个宏/混合文本[1]
更新间隔		不支持
SNMP v1, v2		
	SNMP 团体名	支持
	SNMP OID	支持
SNMP v3		
	上下文名称	支持
	安全名称	支持
	身份验证密码	支持
	隐私密码	支持
	SNMP OID	支持

proxy

在proxy 配置中，用户宏可用于以下字段：

Location	Multiple macros/mix with text[1]
Address for active agents > *Port*（如果 proxy 属于某个组）	no
Interface address and port（用于被动 proxy）	no
Timeouts for item types	no

proxy 组

在proxy group 配置中，用户宏可用于以下字段：

位置	多个宏/与文本混合[1]
故障切换周期	否
proxy 最小数量	否

模板

在模板 配置中，用户宏可用于以下地方：

位置		多个宏/混合文本[1]
标签[2]		
	标签名称	支持
	标签值	支持

触发器

在触发器 配置中，用户宏可用于以下地方：

位置		多个宏/混合文本[1]
名称		支持
操作数据		支持
表达式 (仅在常量和函数参数中; 不支持加密的宏)。		支持
匹配标签		支持
菜单项名称		支持
菜单项 URL[3]		支持
描述		支持
标签[2]		
	标签名称	支持
	标签值	支持

Web 场景

在Web 场景配置中，用户宏可用于以下地方：

位置		多个宏/混合文本[1]
名称		支持
更新间隔		不支持
代理		支持
HTTP 代理		支持
变量 (只允许值)		支持
请求头部 (名称和值)		支持
步骤		
	名称	支持
	URL[3]	支持
	变量 (只允许值)	支持
	请求头部 (名称和值)	支持
	超时	不支持
	返回字符串	支持

位置		多个宏/混合文本[1]
	返回状态码	不支持
认证		
	用户	支持
	密码	支持
	SSL 证书	支持
	SSL 密钥文件	支持
	SSL 密钥密码	支持
标签		
	标签名称	支持
	标签值	支持

其他位置

除了此处列出的位置信息外，用户宏还可用于以下字段：

位置	多个宏/与文本混用[1]
全局脚本(URL、脚本、SSH、Telnet、IPMI)，包括确认文本	yes
webhook	
JavaScript 脚本	no
JavaScript 脚本参数名	no
JavaScript 脚本参数值	yes
仪表板	
Top hosts 仪表板小部件中的 *Text* 数据类型列	yes
Item value 和 *Gauge* 仪表板小部件中的 *Description* 参数	yes
Honeycomb 仪表板小部件中主/副标签的 *Text* 参数	yes
URL 仪表板小部件中的 *URL*[3] 参数	yes
监控>问题、主机、最新数据	
Custom text 列	yes

位置		多个宏/与文本混用[1]
数据采集 > 模板、主机	*Custom text* 列	yes
告警 > 媒体类型 > 消息模板	主题	yes
	消息	yes
告警 > 媒体类型 > 脚本	脚本参数	yes
告警 > 媒体类型 > 媒体类型	*Email* 媒体类型的 *Username* 和 *Password* 字段（当 *Authentication* 设置为”Username and password” 时；建议使用secret macros）	yes
用户 > 用户 > 媒体	激活时	yes

位置		多个宏/与文本混用[1]
管理>通用>*GUI*	工作时间	no
管理>通用>超时	监控项类型的超时	no
管理>通用>连接器	URL	yes
	用户名	yes
	密码	yes
	Bearer token	yes
	超时	no
	HTTP proxy	yes
	SSL 证书文件	yes
	SSL 密钥文件	yes
	SSL 密钥密码	yes

有关 Zabbix 支持的所有宏的完整列表，请参见supported macros。

脚注

[1] 如果某个位置不支持字段中包含多个宏或宏与文本混合，则必须使用单个宏填满整个字段。

[2] 标签名称和值中使用的宏仅在事件生成过程中解析。

[3] 包含secret macro的 URL 将无法正常工作，因为其中的宏会被解析为”******”。

[4] Zabbix trapper 监控项的监控项键中不支持宏。

版本兼容性

支持的 agent

为与 Zabbix 8.0 兼容，Zabbix agent 不得低于 1.6 版本，也不得高于 8.0 版本。

您可能需要检查旧版 agent 的配置，因为某些参数已发生变化，例如 3.0 之前版本中与日志记录相关的参数。

要充分利用最新功能、监控指标、改进的性能以及更低的内存使用量，请使用最新的受支持 agent。

Windows XP 注意事项

- 在 32 位 Windows XP 上，请勿使用高于 6.0.x 的 Zabbix agent；
- 在 Windows XP/Server 2003 上，请勿使用高于 Zabbix 4.0.x 的 agent 模板。较新的模板使用英文性能计数器，而 Windows Vista/Server 2008 之前的系统不支持这些计数器。

支持的 agent 2

自 4.4 版本起，较旧的 Zabbix agent 2 与 Zabbix 8.0 兼容；Zabbix agent 2 的版本不得高于 8.0。

请注意，使用 4.4 和 5.0 版本的 Zabbix agent 2 时，刷新不受支持监控项所使用的默认间隔为 10 分钟。

为了充分利用最新功能、指标、性能改进以及更低的内存占用，请使用最新的受支持 agent 2。

支持的 Zabbix proxy

为了与 Zabbix 8.0 完全兼容，proxy 必须与服务器的主版本一致。
只有 Zabbix 8.0.x proxy 与 Zabbix 8.0.x 服务器完全兼容。

过时的 proxy 会得到部分支持：它们仍然可以收集数据并执行脚本，但无法接收配置更新，例如新的监控项。

相对于 Zabbix 服务器，proxy 可以是：

- 当前（proxy 和服务器具有相同的主版本）；
- 过时（proxy 版本早于服务器版本，但仍被部分支持）；
- 不受支持（proxy 版本早于服务器上一个 LTS 发布版本，或 proxy 版本高于服务器主版本）。

示例：

服务器版本	当前 proxy 版本	过时 proxy 版本	不受支持 proxy 版本
6.4	6.4	6.0、6.2	低于 6.0；高于 6.4
7.0	7.0	6.0、6.2、6.4	低于 6.0；高于 7.0
7.2	7.2	7.0	低于 7.0；高于 7.2
7.4	7.4	7.0	低于 7.0；高于 7.4
8.0	8.0	7.0、7.2、7.4	低于 7.0；高于 8.0

proxy 支持的功能：

proxy 版本	数据更新	配置更新	任务
当前	是	是	是
过时	是	否	远程命令（例如 shell 脚本）； 立即监控项值检查（即 立即执行）； 注意：不支持预处理使用真实值的测试。
不支持	否	否	否

使用不兼容的 Zabbix 守护进程版本时，会记录警告信息。

支持的 XML 文件

Zabbix 8.0 支持导入版本不早于 1.8 的 XML 文件。

Attention:
在 XML 导出格式中，触发器依赖关系仅按名称存储。如果存在多个同名触发器（例如，具有不同的严重性和表达式），并且它们之间定义了依赖关系，则无法导入它们。必须先从 XML 文件中手动删除此类依赖关系，并在导入后重新添加。

时间段语法

概述

要设置时间段，必须使用以下格式：

`d-d,hh:mm-hh:mm`

其中各符号含义如下：

Symbol	Description
d	星期几：1 - 星期一，2 - 星期二，...，7 - 星期日
hh	小时：00-24
mm	分钟：00-59

你可以使用分号 (;) 分隔符指定多个时间段：

`d-d,hh:mm-hh:mm;d-d,hh:mm-hh:mm...`

将时间段留空等同于 1-7,00:00-24:00，这是默认值。

> **Attention:**
> 时间段的上限不包含在内。因此，如果你指定 09:00-18:00，则该时间段内包含的最后一秒是 17:59:59。

示例

工作时间。周一至周五 9:00 到 18:00：

`1-5,09:00-18:00`

工作时间加上周末。周一至周五 9:00 到 18:00 和周六、周日 10:00 到 16:00：

`1-5,09:00-18:00;6-7,10:00-16:00`

单位符号

概述

处理诸如 86400、104857600 或 1000000 这样的大数值可能比较困难，也容易导致错误。为简化配置并提高可读性，Zabbix 支持单位符号（后缀），其作用是作为数值乘数。

不带后缀的触发器表达式：

```
last(/host/system.uptime)<86400
avg(/host/system.cpu.load,600s)<10
last(/host/vm.memory.size[available])<20971520
```

带后缀的触发器表达式：

```
last(/host/system.uptime)<1d
avg(/host/system.cpu.load,10m)<10
last(/host/vm.memory.size[available])<20M
```

后缀还可以简化其他实体（如监控项、部件等）的配置，并帮助以人类可读的格式显示监控项值。

> **Note:**
> 如需确认某个实体的配置字段是否支持后缀，请始终查看正在配置的相应实体页面。

时间后缀

Zabbix 前端在实体配置中支持以下时间后缀：
 s - 秒 *(*使用时，其作用与原始值相同*)*；
 m - 分钟；
 h - 小时；
 d - 天；
 w - 周；
 M - 月 （仅限趋势函数）；
 y - 年 （仅限趋势函数）。

时间后缀仅支持整数。例如，支持 1h，但不支持 1,5h 或 1.5h；请改用 90m。

内存大小后缀

Zabbix 支持以下内存大小后缀：
 K - 千字节
 M - 兆字节
 G - 千兆字节
 T - 太字节

监控项值后缀

后缀也可用于以人类可读的格式显示数值型监控项值。

要启用此功能，请在配置监控项时，在 *Units* 字段中使用以下后缀之一：
 B - 字节；
 Bps - 每秒字节数；
 s - 秒，最多显示三个最大的非零时间单位；
 uptime - 以 hh:mm:ss 或 N days, hh:mm:ss 格式显示的已运行时间；
 unixtime - Unix 时间戳，格式为 yyyy.mm.dd hh:mm:ss。

这些后缀在解释和显示时还适用以下附加规则：

- 对于 B 和 Bps，Zabbix 使用二进制换算（1K = 1024B），遵循 JEDEC 标准。
- 对于其他单位（如 Hz、W 等），Zabbix 使用十进制换算（1K = 1000）。
- 对于 s（秒）：
  - 格式包含 yyy mmm ddd hhh mmm sss ms；仅显示最多三个最大的非零时间单位（例如，1M 10d 4h）。
  - 如果某个单位为零且位于两个非零单位之间，则会被省略（例如，显示 10d 56m，而不是 10d 0h 56m）。

当使用 *Units* 时，以下倍率后缀会自动应用于监控项值：
 K, **M**, **G**, **T** - 千、兆、吉、太；
 P, **E**, **Z**, **Y** - 拍、艾、泽、尧 *(这些仅在前端中应用)*。

要防止单位转换，请使用 ! 前缀（例如，!B 或!s）。

以下示例展示了接收到的监控项值如何根据指定单位进行转换：

```
1 B → 1 B
1024 B → 1 KB
1536 B → 1.5 KB
881764 B → 881.76 KB
881764 !B → 881764 B

0.0000155 s → 0.016ms
3470400 s → 1M 10d 4h
2606400 s → 1M 4h
2592000 s → 1M
2592001 s → 1M
2592001 !s → 2592001 s

17764 uptime → 04:56:04
86400 uptime → 1 day, 00:00:00
881764 uptime → 10 days, 04:56:04
32417764 uptime → 375 days, 04:56:04
32417764 !uptime → 32417764 uptime

881764 unixtime → 1970-01-11 04:56:04 AM

17764 Hz → 17.76 KHz
86400 Hz → 86.4 KHz
881764 Hz → 881.76 KHz
32417764 Hz → 32.42 MHz

0 ! → 0
0 !! → 0 !
```

Note:
在 Zabbix 4.0 之前，存在一个硬编码的单位停用列表，包含 ms、rpm、RPM、%。该停用列表现已弃用，因此，防止转换此类单位的正确方式是使用 !ms、!rpm、!RPM、!%。

命令执行

Zabbix 为外部检查、用户参数、system.run 监控项、自定义告警脚本、远程命令和全局脚本使用通用功能。

执行步骤

Note:
默认情况下，Zabbix 中的所有脚本都是使用 *sh* shell 执行的，并且无法修改默认 shell。要使用不同的 shell，您可以采用一种解决方法：创建一个脚本文件并在命令执行期间调用该脚本。

命令/脚本在 Unix 和 Windows 平台上的执行方式类似：

1. Zabbix（父进程）创建一个用于通信的管道
2. Zabbix 将管道设置为待创建的子进程的输出
3. Zabbix 创建子进程（运行命令/脚本）
4. 为子进程创建新的进程组（在 Unix 中）或作业（在 Windows 中）
5. Zabbix 从管道中读取，直到发生超时或没有数据写入另一端（所有句柄/文件描述符都已关闭）。请注意，子进程可以创建更多进程，并在退出或关闭句柄/文件描述符之前退出。
6. 如果没有发生超时，Zabbix 会一直等待，直到初始子进程退出或发生超时
7. 如果初始子进程退出且未发生超时，Zabbix 会检查初始子进程的退出代码，并将其与 0 进行比较（非零值被视为执行失败，仅适用于在 Zabbix server 和 Zabbix proxy 上执行的自定义告警脚本、远程命令和用户脚本）
8. 此时，如果所有操作都已完成，整个过程树（即过程组或作业）终止

Attention:
Zabbix 中，初始子进程已退出，并且没有其他进程仍保持输出句柄/文件描述符打开状态时，则判断命令/脚本已经完成了处理。处理完成后，所有创建的进程都会被终止。

命令中的所有双引号和反斜杠都使用反斜杠进行转义，并且命令用双引号括起来。

退出码检查

退出码按以下条件进行检查：

- 仅适用于在 Zabbix 服务器和 Zabbix proxy 上执行的自定义告警脚本、远程命令和用户脚本。
- 任何不等于 0 的退出码都视为执行失败。
- 失败执行的标准错误和标准输出内容会被收集，并可在前端中查看（显示执行结果的位置）。
- 通过在agent/proxy 上启用 LogRemoteCommands 参数，可以为在 Zabbix agent/proxy 上执行的远程命令创建额外的日志条目。

失败的命令/脚本在前端中可能显示的消息和日志条目：

- 失败执行的标准错误和标准输出内容（如果有）。
- "Process exited with code: N."（对于空输出且退出码不等于 0 的情况）。
- "Process killed by signal: N."（对于被信号终止的进程，仅限 Linux）。
- "Process terminated unexpectedly."（对于因未知原因终止的进程）。

另请参阅

- 外部检查
- 用户参数
- system.run 监控项
- 自定义告警脚本
- 远程命令
- 全局脚本

转义示例

概述

本页提供了在各种上下文中使用正则表达式时正确使用转义的示例。

Note:
使用触发器表达式构造函数时，会自动添加正则表达式中的正确转义。

示例

带上下文的用户宏

正则表达式：`\.+\"[a-z]+<br>` 带上下文的用户宏：`{$MACRO:regex:"\.+\\"[a-z]+"}`

注意：

- 反斜杠不需要转义；
- 引号需要转义。

监控项键参数中的宏函数

正则表达式：`.+:(\d+)$<br>` 监控项键：`net.tcp.service[tcp,,"{{$ENDPOINT}.regsub(\".+:(\d+)$\",\1)}"]`

注意：

- `regsub` 宏函数中的正则表达式使用双引号括起来（因为包含右括号）；
- 正则表达式两侧的引号需要转义（因为整个第三个监控项参数使用双引号括起来）；
- 第三个监控项键参数使用双引号括起来，因为它包含逗号。

LLD 宏函数

正则表达式：`\.+\"([a-z]+)<br>` LLD 宏：`{{#MACRO}.iregsub("\.+\\"([a-z]+)", \1)}`

注意：

- 反斜杠不需要转义；
- 引号需要转义。

用户宏上下文中的 **LLD** 宏函数

正则表达式：`\.+\"([a-z]+)<br>` LLD 宏：`{{#MACRO}.iregsub("\.+\\"([a-z]+)", \1)}<br>` 带上下文的用户宏：`{$MACRO:"{{#MACRO}.iregsub(\"\.+\\\"[a-z]+\", \1)}"}`

注意：

- LLD 的反斜杠转义方式不变；
- 将 LLD 宏插入到用户宏上下文时，需要将其放入字符串中：

1. 在宏表达式两侧添加引号；
2. 引号需要转义；总共会新增 3 个反斜杠。

函数的字符串参数（任意）

以 concat 为例。

字符串内容：`\.+\"[a-z]+<br>` 表达式：`concat("abc", "\\.\\\"[a-z]+")`

注意：

- 字符串参数要求同时对反斜杠和引号进行转义。

函数字符串参数中的 **LLD** 宏函数

正则表达式：`\.+\"([a-z]+)<br>` LLD 宏：`{{#MACRO}.iregsub("\.+\\"([a-z]+)", \1)}<br>` 表达式：`concat("abc, "{{#MACRO}.iregsub(\"\\.+\\\\\"([a-z]+)\", \\1)}")`

注意：

- 字符串参数要求同时对反斜杠和引号进行转义；
- 由于宏只有在字符串去引号后才会被解析，因此还需要再增加一层转义；

函数字符串参数中的带上下文用户宏

正则表达式：`\.+\"[a-z]+<br>` 带上下文的用户宏：`{$MACRO:regex:"\.+\\"[a-z]+"}<br>` 表达式：`concat("abc, "{$MACRO:regex:\"\\.+\\\\\"[a-z]+\"}")`

注意：

- 与前一个示例相同，这里也需要额外增加一层转义；
- 反斜杠和引号仅针对最外层转义进行转义（因为它是一个字符串参数）。

函数中的用户宏上下文中的 **LLD** 宏函数

正则表达式：`\.+\"([a-z])+<br>` LLD 宏：`{{#MACRO}.iregsub("\.+\\"([a-z]+)", \1)}<br>` 带上下文的用户宏：`{$MACRO:"{{#MACRO}.iregsub(\"\.+\\\"([a-z]+)\", \1)}"}<br>` 表达式：`concat("abc, "{$MACRO:\"{{#MACRO}.iregsub\\1)}\"}")`

注意这里有三层转义：

1. 对于 LLD 宏函数，不对反斜杠进行转义；
2. 对于带上下文的用户宏，不对反斜杠进行转义；
3. 对于函数的字符串参数，需要对反斜杠进行转义。

仅位于字符串中的带上下文用户宏

正则表达式：`\.+\"[a-z]+<br>` 带上下文的用户宏：`{$MACRO:regex:"\.+\\"[a-z]+"}<br>` 位于某个表达式的字符串中，例如：`func(arg1, arg2, arg3)="{$MACRO:regex:\"\\.+\\\\\"[a-z]+\"}"`

注意：

- 字符串同样要求对反斜杠进行转义；
- 字符串同样要求对引号进行转义；
- 这同样是一个包含 2 层转义的情况：

1. 对用户宏上下文进行转义，不对反斜杠进行转义；
2. 由于它是一个带反斜杠转义的字符串，因此还需要再进行一层转义。

什么是 **Zabbix**

Zabbix 由 Alexei Vladishev 创建，目前由 Zabbix SIA 积极开发和提供支持。

Zabbix 是一款企业级开源分布式监控解决方案。

Zabbix 是一款用于监控网络众多参数，以及服务器、虚拟机、应用程序、服务、数据库、网站、云等的运行状况和完整性的软件。Zabbix 使用灵活的通知机制，允许用户为几乎任何事件配置基于电子邮件的告警。这使得用户能够快速响应服务器问题。Zabbix 基于存储的数据提供出色的报表和数据可视化功能。因此，Zabbix 非常适合用于容量规划。

Zabbix 同时支持轮询和捕获。所有 Zabbix 报表和统计信息以及配置参数，均可通过基于 Web 的前端访问。基于 Web 的前端确保您可以从任何位置评估网络状态和服务器运行状况。经过正确配置后，Zabbix 可以在 IT 基础设施监控中发挥重要作用。对于只有几台服务器的小型组织和拥有大量服务器的大型企业而言，情况同样如此。

Zabbix 可免费使用。Zabbix 根据 AGPL-3.0 许可证编写和分发。这意味着其源代码可自由分发，并向公众开放。

商业支持由 Zabbix 公司及其遍布全球的合作伙伴提供。

详细了解Zabbix 功能。

Zabbix 的用户　世界各地不同规模的许多组织都将 Zabbix 作为主要的监控平台。

架构　Zabbix 由多个主要软件组件组成。它们的职责如下所述。

服务器

Zabbix server 是 agent 报告可用性、完整性信息和统计数据的中心组件。服务器是存储所有配置、统计和运行数据的中央存储库。

数据库存储

所有配置信息以及 Zabbix 收集的数据都存储在数据库中。

Web 界面

为了能够从任何地方、任何平台轻松访问 Zabbix，系统提供了基于 Web 的界面。该界面是 Zabbix 服务器的一部分，通常（但不一定）运行在与服务器相同的物理机器上。

proxy

Zabbix proxy 可以代表 Zabbix 服务器收集性能和可用性数据。proxy 是 Zabbix 部署中的可选组件；不过，它对于分担单个 Zabbix 服务器的负载可能非常有帮助。

Agent

Zabbix agent 部署在监控目标上，用于主动监控本地资源和应用程序，并将收集到的数据报告给 Zabbix 服务器。自 Zabbix 4.4 起，提供两种类型的 agent：Zabbix agent（轻量级，支持多种平台，使用 C 编写）和Zabbix agent 2（灵活性更高，可通过插件轻松扩展，使用 Go 编写）。

数据流　此外，退一步来看 Zabbix 中整体的数据流也很重要。为了创建一个用于收集数据的监控项，您必须先创建一个主机。从 Zabbix 的另一端来看，要创建触发器，您必须先有一个监控项。要创建动作，您必须先有一个触发器。因此，如果您希望在 *Server X* 的 CPU 负载过高时收到告警，您必须先为 *Server X* 创建一个主机条目，然后创建一个用于监控其 CPU 的监控项，再创建一个在 CPU 过高时触发的触发器，最后创建一个向您发送电子邮件的动作。虽然这看起来步骤很多，但借助模板，这其实并不复杂。不过，正因为采用了这种设计，才可以创建非常灵活的配置。

特性　Zabbix 是一个高度集成的网络监控解决方案，在一个软件包中提供了多种功能。

数据采集

- 可用性和性能检查
- 支持 SNMP（包括 trapping 和 polling）、IPMI、JMX、VMware 监控
- 自定义检查
- 按自定义间隔采集所需数据
- 由服务器/proxy 和 agent 执行

灵活的阈值定义

- 您可以定义非常灵活的问题阈值，称为触发器，它们引用后端数据库中的值

高度可配置的告警

- 通知发送可按升级计划、接收者、媒介类型进行自定义
- 可使用宏变量使通知更有意义且更有帮助
- 自动操作包括远程命令

实时绘图

- 监控项会使用内置绘图功能立即生成图表

Web 监控能力

- Zabbix 可以沿着网站上模拟的鼠标点击路径进行跟踪，并检查功能和响应时间

丰富的可视化选项

- 能够创建自定义图表，将多个监控项组合到单一视图中
- 网络地图
- 仪表板式概览中的幻灯片放映
- 报告
- 被监控资源的高层（业务）视图

历史数据存储

- 数据存储在数据库中
- 可配置的历史记录
- 内置的 housekeeping 过程

易于配置

- 将被监控设备作为主机添加
- 一旦进入数据库，主机就会被纳入监控
- 将模板应用到被监控设备

模板的使用

- 在模板中对检查进行分组
- 模板可以继承其他模板

网络发现

- 自动发现网络设备
- agent 自动注册
- 发现文件系统、网络接口和 SNMP OID

快速的 **web** 前端

- 基于 PHP 的前端
- 可从任何地方访问
- 您可以通过点击进行操作
- 审计日志

Zabbix API

- Zabbix API 为 Zabbix 提供可编程接口，用于批量操作、第三方软件集成及其他用途。

权限系统

- 安全的用户身份验证
- 某些用户可限制访问特定视图

功能完整且易于扩展的 **agent**

- 部署在监控目标上
- 可部署在 Linux 和 Windows 上

二进制守护进程

- 使用 C 语言编写，具有高性能和较小的内存占用
- 易于移植

适用于复杂环境

- 通过使用 Zabbix proxy 轻松实现远程监控

术语表　本节介绍 Zabbix 中一些常用术语的含义。

主机

- 任何物理或虚拟设备、应用程序、服务，或其他在逻辑上相关的受监控参数集合。

主机组

- 主机的逻辑分组。为不同用户组分配主机访问权限时会使用主机组。

监控项

- 希望从主机接收的某一特定数据，即一种数据指标。

值预处理

- 对接收到的指标值进行转换，然后再将其保存到数据库中。

触发器

- 定义问题阈值并用于“评估”监控项中接收的数据的逻辑表达式。

当接收到的数据高于阈值时，触发器会从 正常状态变为 问题状态。当接收到的数据低于阈值时，触发器会保持或返回 正常状态。

模板

- 一组可应用于一个或多个主机的实体（监控项、触发器、图形、低级别发现规则、*Web* 场景）。

模板用于加快在主机上部署监控任务的速度，同时也便于对监控任务进行批量更改。模板直接链接到单个主机。

模板组

- 模板的逻辑分组。为不同用户组分配模板访问权限时会使用模板组。

事件

- 值得关注的某件事情的一次发生，例如触发器状态发生变化，或执行发现/*agent* 自动注册。

事件标签

- 为事件预先定义的标记。它可用于事件关联、细化权限等。

事件关联

- 一种灵活且精确地将问题与其解决方案关联起来的方法。

例如，可以定义由一个触发器报告的问题由另一个触发器解决，而后者甚至可以使用不同的数据采集方法。

问题

- 处于“问题”状态的触发器。

问题更新

- *Zabbix* 提供的问题管理选项，例如添加评论、确认、更改严重性或手动关闭问题。

动作

- 预先定义的事件响应方式。

动作由操作（例如发送通知）和条件（何时执行操作）组成。

升级

- 在动作中执行操作的自定义场景；发送通知/执行远程命令的序列。

宏

- 根据上下文解析为特定值的变量占位符。

媒介

- 传递通知的方式；传递渠道。

通知

- 通过所选媒介渠道向用户发送的有关某个事件的消息。

远程命令

- 在满足某些条件时，在受监控主机上自动执行的预定义命令。

***Web* 场景**

- 用于检查网站可用性的一个或多个 *HTTP* 请求。

前端

- *Zabbix* 提供的 *Web* 界面。

仪表板

- 前端中可自定义的部分，用于以称为小组件的可视化单元显示重要信息的摘要和可视化内容。

小组件

- 显示特定类型和来源信息的可视化单元（摘要、地图、图形、时钟等），用于仪表板中。

Zabbix API

- *Zabbix API* 允许您使用 *JSON RPC* 协议创建、更新和获取 *Zabbix* 对象（例如主机、监控项、图形等），或执行其他自定义任务。

***Zabbix* 服务器**

- *Zabbix* 软件的中央进程，负责执行监控、与 *Zabbix proxy* 和 *agent* 交互、计算触发器、发送通知；也是数据的中央存储库。

Zabbix proxy

- 可代表 *Zabbix* 服务器收集数据的进程，从而分担服务器的一部分处理负载。

Zabbix agent

- 部署在监控目标上的进程，用于主动监控本地资源和应用程序。

Zabbix agent 2

- 新一代 *Zabbix agent*，用于主动监控本地资源和应用程序，并支持使用自定义插件进行监控。

> **Attention:**
> 由于 Zabbix agent 2 与 Zabbix agent 共享许多功能，因此如果功能行为相同，文档中的 “Zabbix agent” 同时指 Zabbix agent 和 Zabbix agent 2。只有在功能存在差异时，才会特别指明 Zabbix agent 2。

加密

- 支持使用传输层安全性（*TLS*）协议对 *Zabbix* 组件（服务器、*proxy*、*agent*、*zabbix_sender* 和 *zabbix_get* 工具）之间的通信进行加密。

***agent* 自动注册**

- *Zabbix agent* 自动注册为主机并开始监控的过程。

网络发现

- 自动发现网络设备。

低级别发现

- 自动发现特定设备上的低级别实体（例如文件系统、网络接口等）。

低级别发现规则

- 用于自动发现设备上低级别实体的一组定义。

监控项原型

- 包含作为变量的特定参数、可用于低级别发现的指标。低级别发现完成后，变量会自动替换为实际发现的参数，并且该指标会自动开始收集数据。

触发器原型

- 包含作为变量的特定参数、可用于低级别发现的触发器。低级别发现完成后，变量会自动替换为实际发现的参数，并且触发器会自动开始评估数据。

低级别发现中还使用了其他一些 Zabbix 实体的原型，包括图形原型、主机原型和主机组原型。

Zabbix Cloud

Zabbix Cloud

了解 Zabbix Cloud 中组织和节点的可用配置选项及访问控制。

请注意，Zabbix Cloud 节点当前运行 **Zabbix 7.0**。
 有关 Zabbix 7.0 的详细信息，请参阅官方文档。

设置和管理 **Zabbix Cloud**

开始使用 **Zabbix Cloud**　通过引导式说明，帮助您快速启动并运行 Zabbix Cloud 实例，包括初始配置以及连接受监控设备。

节点配置　了解如何设置您的 Zabbix Cloud 节点，重点关注访问、数据处理和资源设置。

添加用户　了解如何在 Zabbix Cloud 组织及其节点中为用户添加并分配角色。

了解 **Zabbix Cloud**

Zabbix Cloud 与本地部署对比　比较这两种部署方式在管理、可扩展性和维护方面的差异，并了解哪一种能够最无缝地支持您的工作流程。

探索 **Zabbix Cloud**　了解 Zabbix Cloud 概览，包括其主要功能、优势以及订阅所包含的内容。有关设置和使用的常见问题，请参阅常见问题解答（FAQ）。

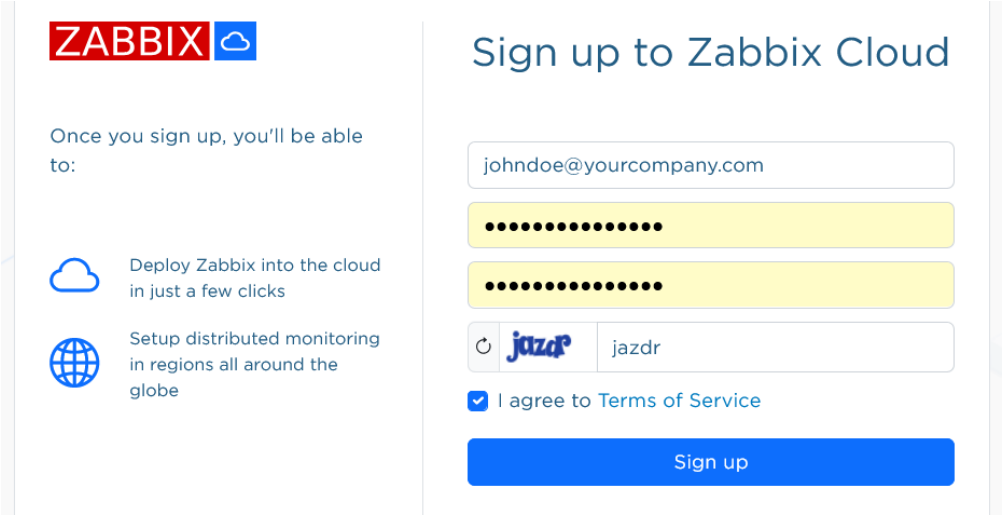
在云中部署 **Zabbix**

要在云端启动 Zabbix，请按照以下流程操作：

1. 在 cloud.zabbix.com 登录或注册。
2. 创建云节点。
3. 开始在云端使用 Zabbix。

登录或注册　您可以使用现有的 Google、GitHub 或 Microsoft 账户在 cloud.zabbix.com 登录。

或者，前往注册页面进行注册。



注册后，您将在电子邮件中收到新账户的确认信息。

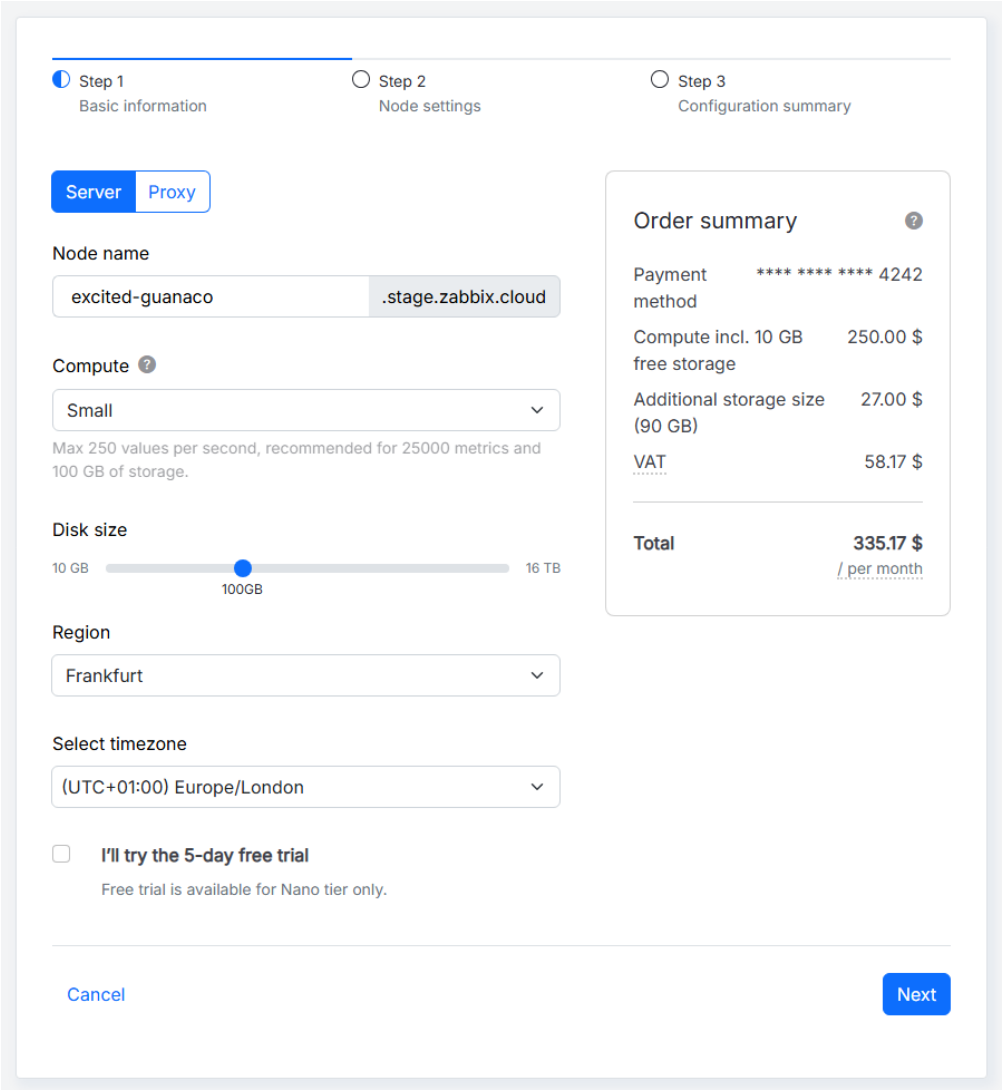
前往 cloud.zabbix.com 登录。登录时，系统会要求您输入一次性密码。该密码将通过电子邮件发送给您。

创建云节点　云节点是部署在 Zabbix Cloud 中的 Zabbix 实例。它可以部署为服务器节点或 proxy 节点。

登录后，您会看到默认情况下账户中已存在一个 *Private* 组织。您可以稍后重命名默认组织。

点击 *Create New Node*。系统将打开一个包含三个步骤的节点配置表单。

在第一步中，您将选择基本信息。



Parameter	Description
Server/Proxy	选择部署 Zabbix 服务器实例还是 Zabbix proxy 实例。
Node name	节点名称将以 `<your-name>.zabbix.cloud` 的格式构成唯一子域名，用于访问节点的服务。
Compute	选择节点的计算能力。 要确定所需的资源级别，请参阅定价详情。如果需要更高的计算能力，您可以随时升级。创建非试用节点前需要设置计费方式。请通过 *Billing* 菜单添加计费信息。 您可以使用 *Nano* 层级节点进行为期 5 天的免费试用。试用期间不会向您收费，但试用结束后，如果未将节点转换为付费节点，系统将删除该节点。
Disk size	滑块默认设置为根据层级推荐的最小值。 如果您存储大量历史数据或趋势数据，请增大该值。请注意，磁盘越大，每月费用越高，并且您无法自行缩小磁盘；如需缩小磁盘，必须联系支持团队。
Region	选择节点的部署区域。
Select timezone	选择 Zabbix 服务器的时区。此时区由 Zabbix 服务器使用，与维护时区和用户配置文件时区相互独立。

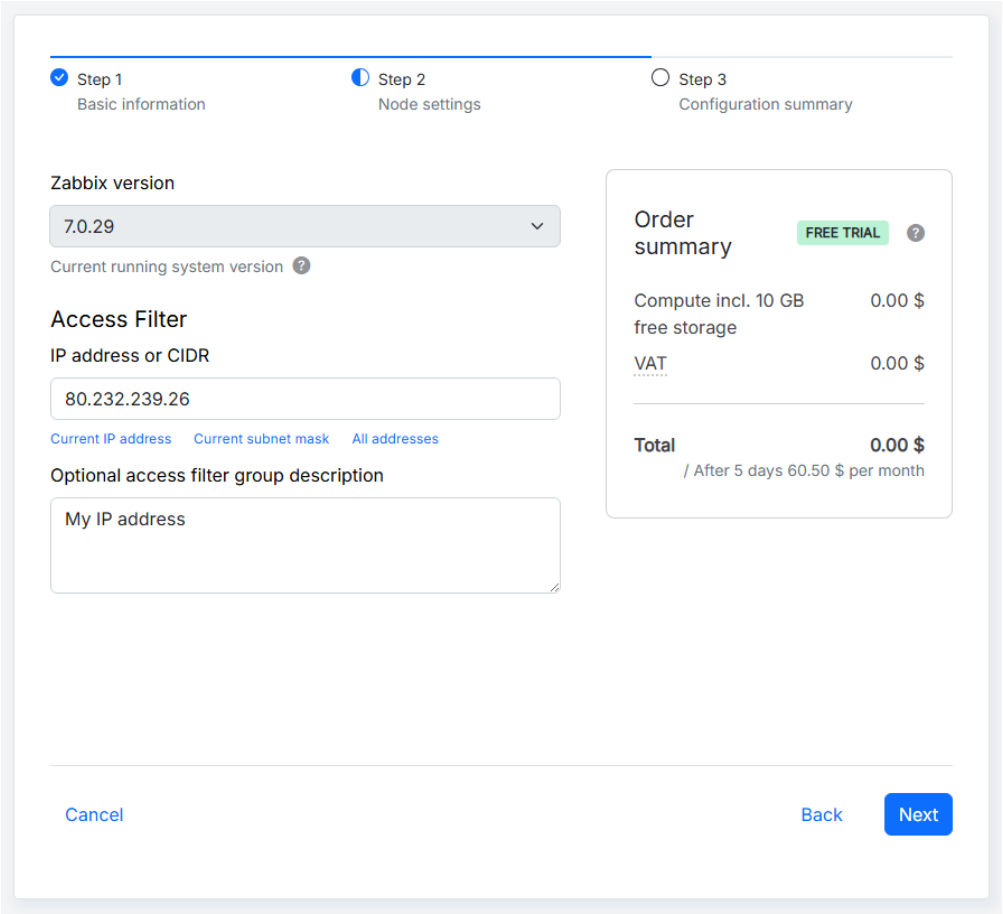
完成配置后，点击 *Next*。

在第二步中，您将选择节点设置。

第二步中的配置选项取决于您创建的是服务器节点还是 proxy 节点。

配置服务器节点

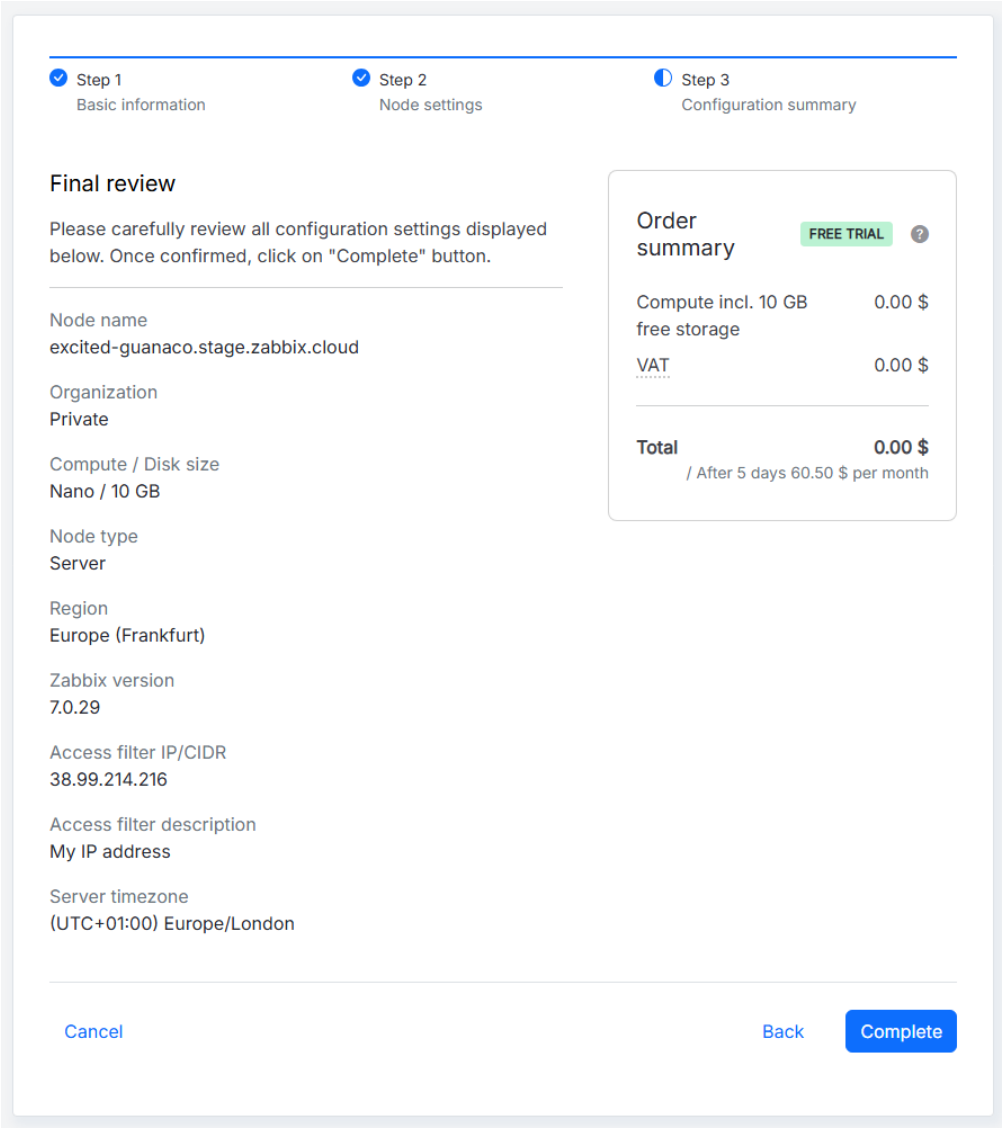
创建服务器节点时，可以使用以下参数：



Parameter	Description
Zabbix version	选择 Zabbix 版本。Zabbix Cloud 支持最新的 LTS 版本，并会自动应用次要版本更新。
IP address or CIDR	指定节点的初始访问过滤器。手动输入 IP 地址或 CIDR，或者选择相关选项自动填写该字段。对于服务器节点，此过滤器适用于与服务器、API 和前端的连接。稍后可以在访问过滤器选项卡中更改该过滤器。
Optional access filter group description	为访问过滤器组添加可选描述，该组可以包含多个 IP 地址或 CIDR。

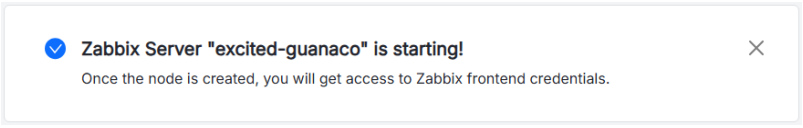
Warning:
创建节点时，*IP address or CIDR* 字段会预先填入用于访问 Zabbix Cloud 的 IP 地址。您可以更改此值，以指定其他初始访问过滤器。创建节点后，您可以在 *Access filters* 选项卡中添加、编辑或删除访问过滤器。

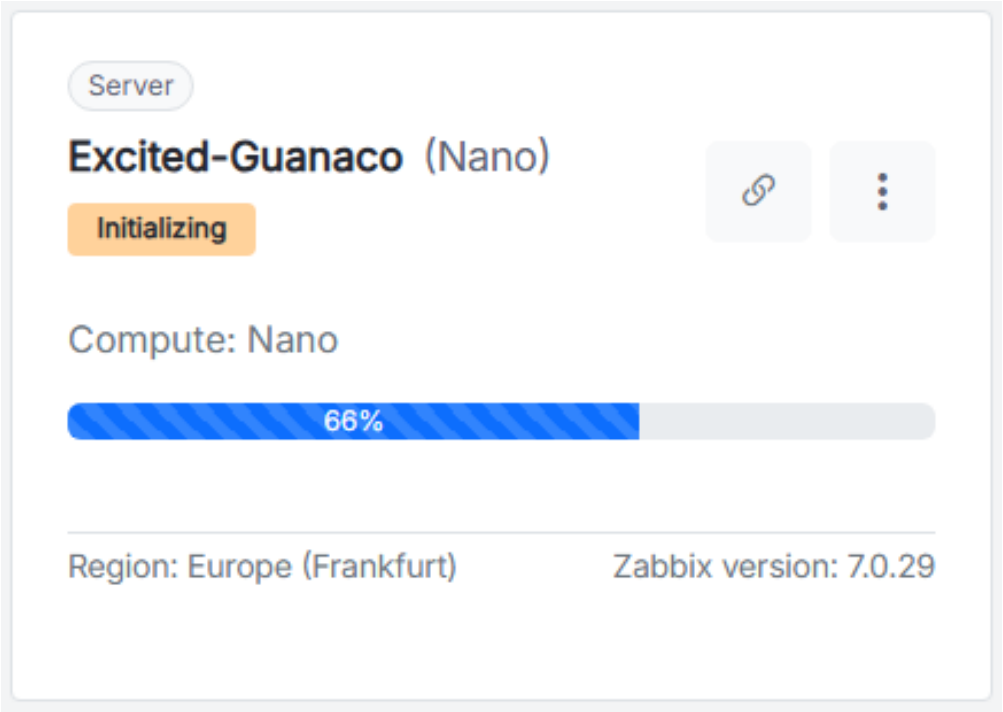
第三步显示节点的配置摘要。



查看配置设置并确认所有内容正确后，对于试用节点，点击 *Complete*；对于付费节点，点击 *Proceed to payment*。否则，点击 *Back* 按钮返回上一步，或点击 *Cancel* 从头开始。

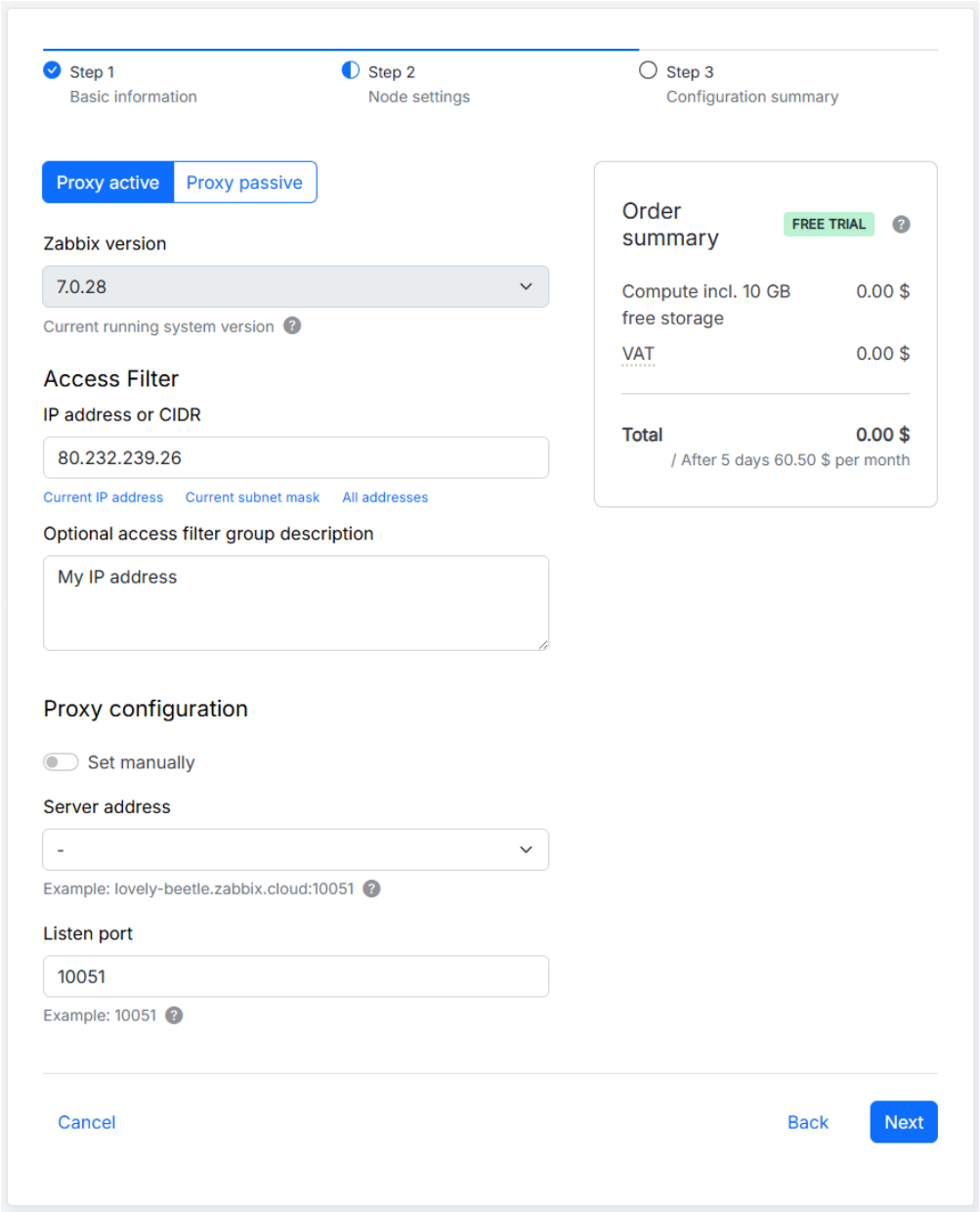
节点初始化期间，系统会显示相应消息，并且该节点已经出现在 *Organizations* 下。





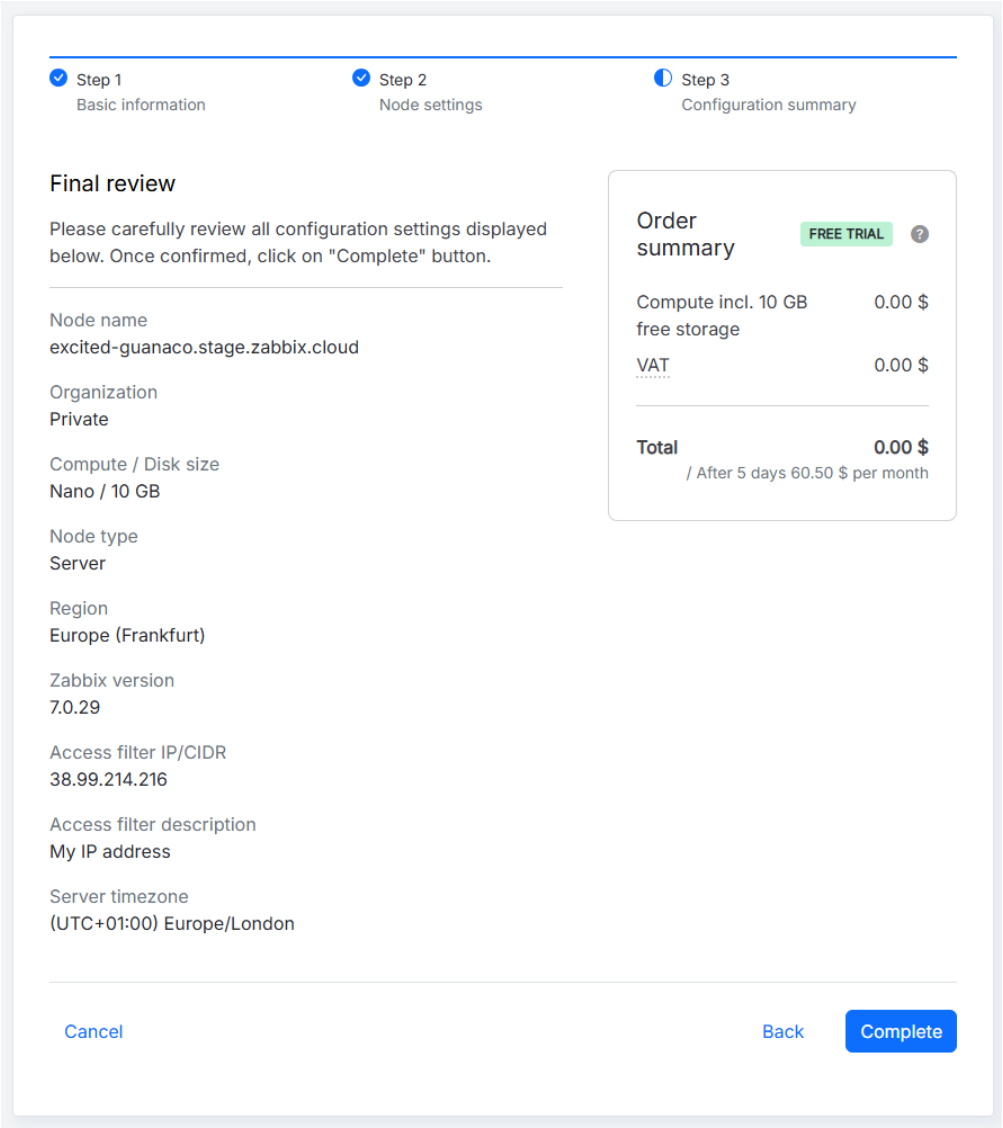
配置 proxy 节点

创建 proxy 节点时，可以使用以下参数：



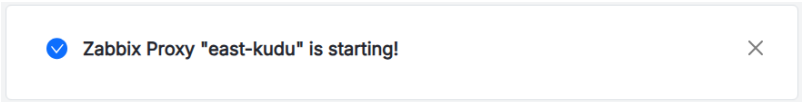
Parameter	Description
Proxy active/Proxy passive	选择proxy 模式。在主动模式下，proxy 向 Zabbix 服务器发起连接；在被动模式下，Zabbix 服务器向 proxy 发起连接。
Zabbix version	选择 Zabbix 版本。Zabbix Cloud 支持最新的 LTS 版本，并会自动应用次要版本更新。
IP address or CIDR	指定 proxy 的初始入站 IP 访问过滤器。手动输入 IP 地址或 CIDR，或者选择相关选项自动填写该字段。该过滤器定义允许到达 proxy 的网络请求，并且稍后可以在访问过滤器选项卡中更改。
Optional access filter group description	为访问过滤器组添加可选描述，该组可以包含多个 IP 地址或 CIDR。
Set manually	选择指定 Zabbix 服务器的方式。禁用时，从可用服务器节点列表中选择一个服务器节点。启用时，手动指定服务器地址。
Server address	指定 Zabbix 服务器地址。禁用 *Set manually* 时，从列表中选择一个服务器节点。启用时，最多指定 10 个服务器地址。对于主动 proxy，请指定 Zabbix 服务器的 IP 地址或 DNS 名称。对于被动 proxy，请指定 Zabbix 服务器的 IP 地址、DNS 名称或 CIDR。访问过滤器需要单独配置；指定服务器地址时不会自动更新访问过滤器。
Listen port	指定 proxy 监听传入连接的端口。默认端口为 10051。创建节点后无法更改监听端口。

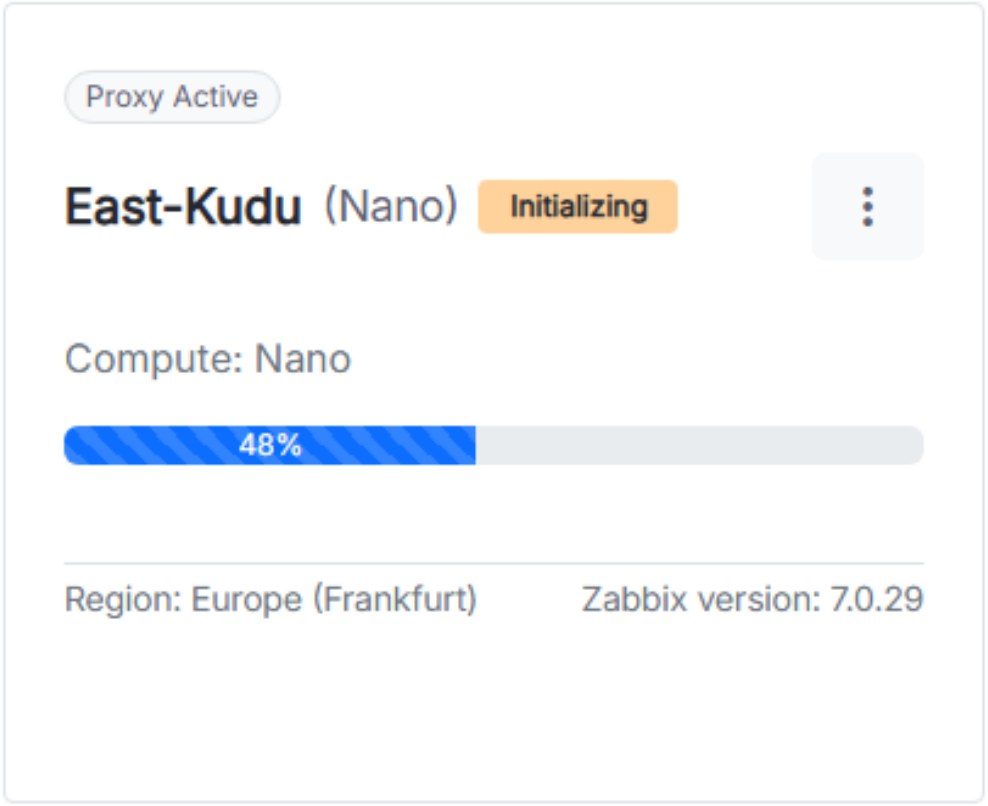
在第三步中，您将看到节点的配置摘要。



查看配置设置并确认所有内容正确后，对于试用节点，点击 *Complete*；对于付费节点，点击 *Proceed to payment*。否则，点击 *Back* 按钮返回上一步，或点击 *Cancel* 从头开始。

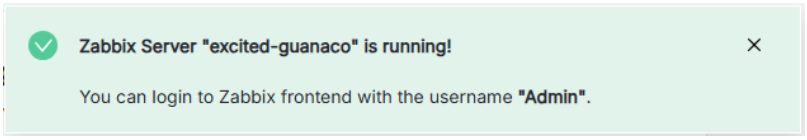
节点初始化期间，系统会显示相应消息，并且该节点已经出现在 *Organizations* 下。





要配置 Zabbix Cloud proxy，请参阅设置 Zabbix Cloud proxy。

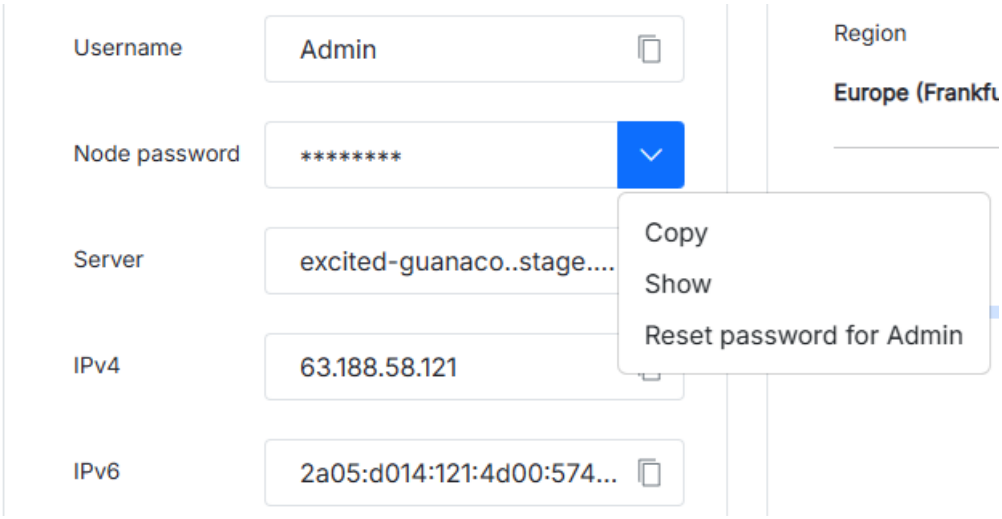
开始使用 **Zabbix**



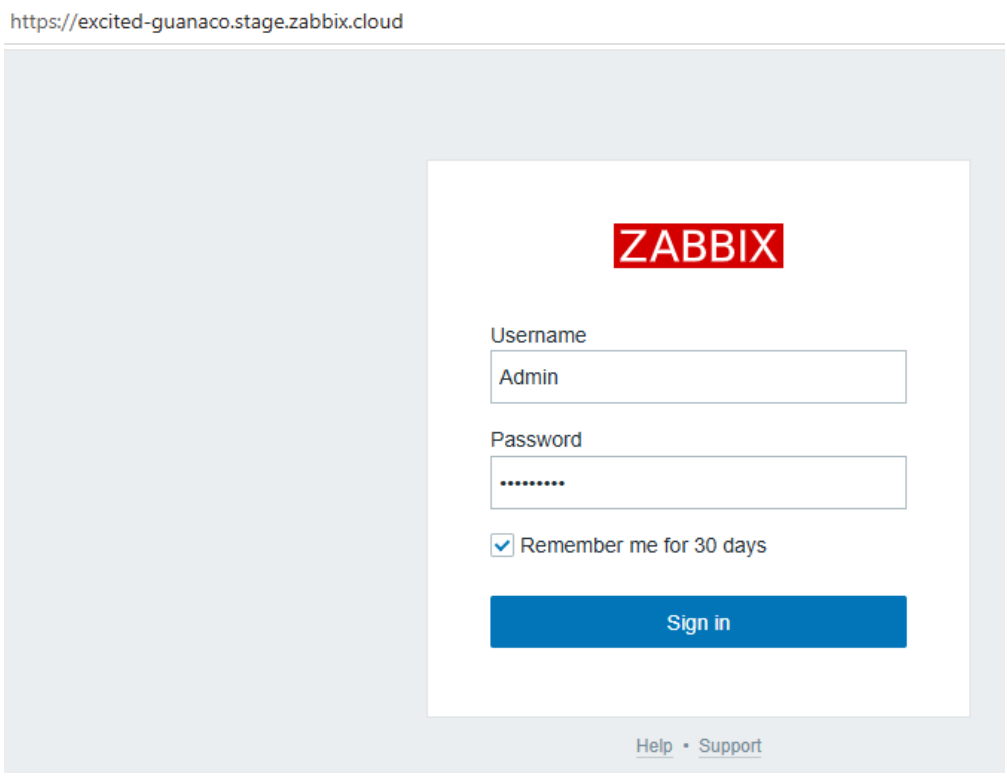
当服务器节点状态从 *Initializing* 变为 *Running* 后，您可以登录 Zabbix 前端。

要获取初始前端密码，请点击已创建的节点。

您将进入节点的设置（概览、访问过滤器、加密、备份等）。在 **Connection info** 中，找到 *Node password* 字段，然后从下拉菜单中选择 *Reset password for Admin*。密码重置后，点击同一下拉菜单并选择 *Copy*。



点击前端 URL，打开 Zabbix 前端（`excited-guanaco.zabbix.cloud`）。



使用以下凭据登录：

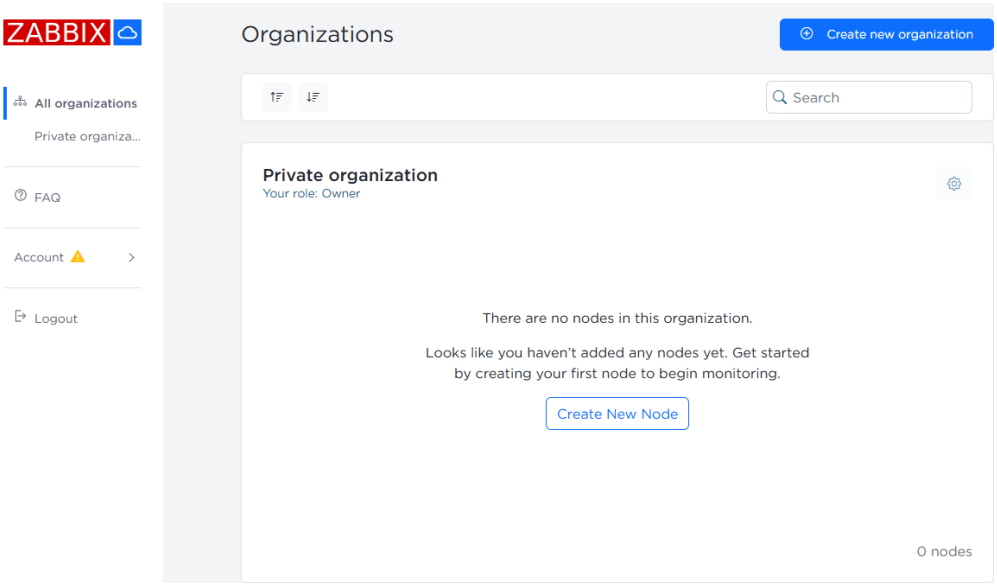
- *Username* - `Admin`
- *Password* - 提供复制的密码

要开始监控，您需要定义访问过滤器 - 允许连接到 Zabbix Cloud 服务器的 IP 地址。至少必须定义一个地址。

组织管理　组织是一个或多个节点的容器。组织还可以帮助您构建资源结构并委派职责，例如：

- 节点管理
- 计费

登录 Zabbix Cloud 后，将显示所有组织页面。默认情况下，系统会为您的账户创建一个私有组织。

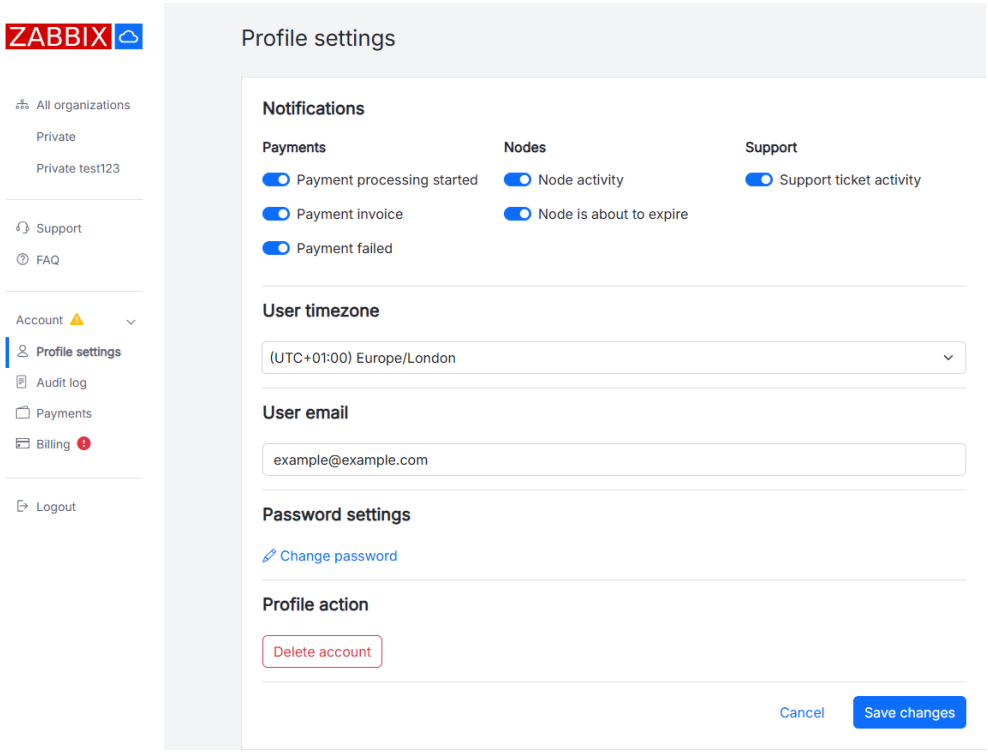


作为此组织的所有者，您可以管理用户、管理员、节点以及组织名称。点击  设置按钮以访问管理选项。

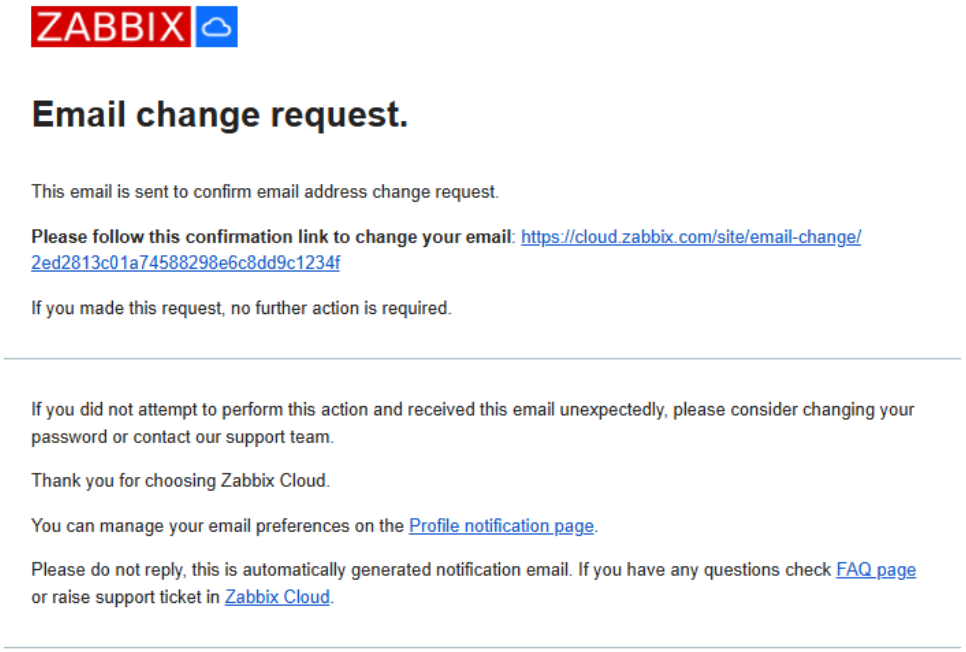
账户管理　更改账户电子邮件地址

要更改账户电子邮件地址，请导航至 账户 > 配置文件设置。

在 用户电子邮件字段中输入新的电子邮件地址，然后点击 保存更改。



系统会向新的电子邮件地址发送确认消息。请点击消息中的链接，以确认电子邮件地址更改。



确认链接会打开 配置文件设置页面，并显示确认消息；用户电子邮件字段中也会显示更新后的电子邮件地址。

确认更改后，系统会向新的和之前的电子邮件地址发送确认更改的邮件。发送到之前地址的消息中会包含之前和新的电子邮件地址。

节点配置

概述

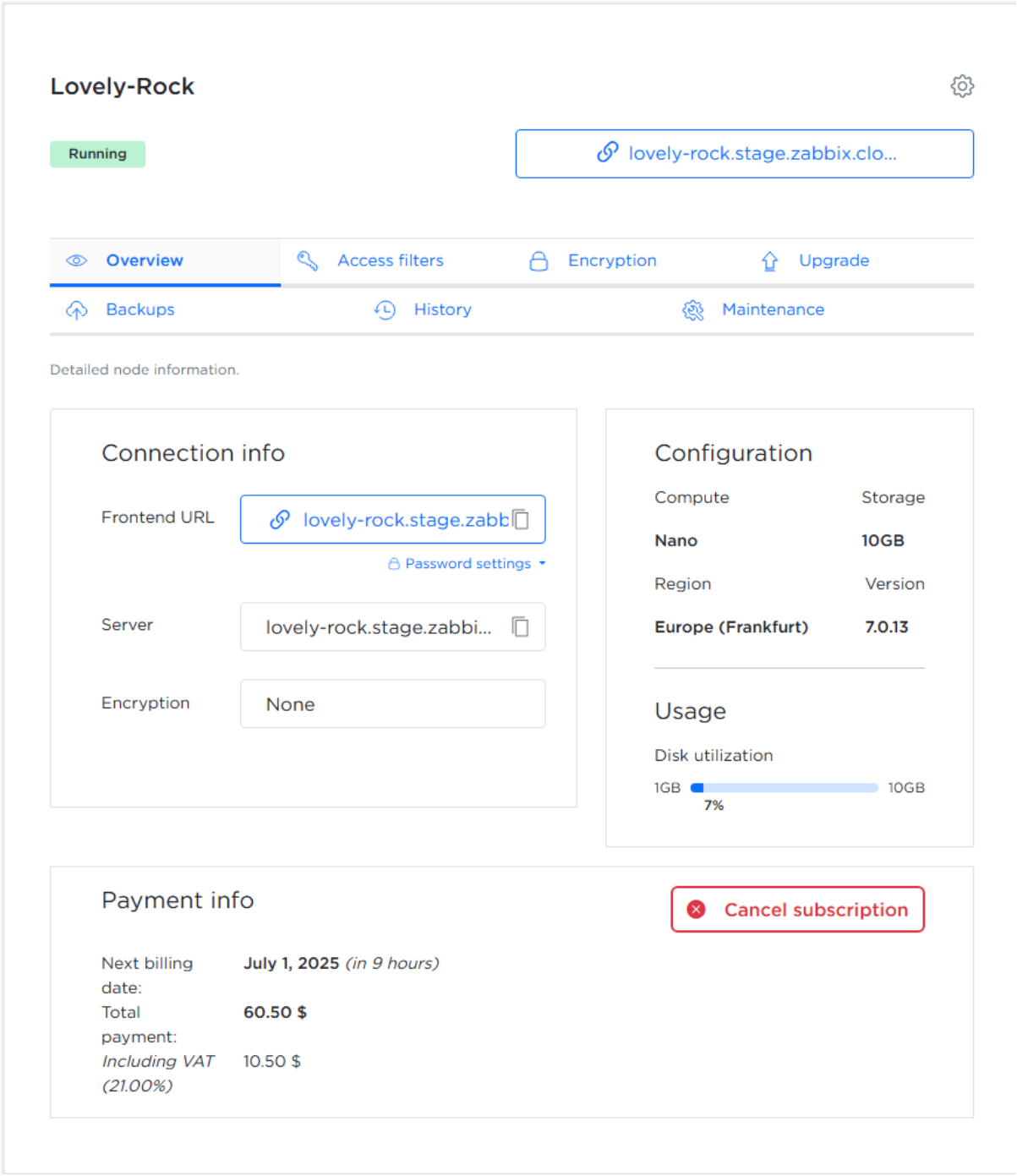
云节点是部署在云中的 Zabbix 组件。服务器节点包括 Zabbix 服务器、数据库和前端，而 proxy 节点包括 Zabbix proxy 和数据库。云节点具有专用的计算资源（CPU、内存）和磁盘存储空间。

用户可以创建一个或多个云节点。云节点可以分组到组织中（按结构、区域等分组），从而构建分布式监控环境。

云节点和组织通过 Zabbix Cloud 用户界面创建。要访问节点配置页面，请从所有组织、您的组织菜单中点击节点，或点击节点启动后显示的节点配置页面链接。

在节点配置视图中，每个选项卡都可用于管理节点的特定方面：

- 概述 - 查看节点详细信息、打开其前端以及管理节点用户。
- 访问过滤器 - 按 IP 地址或子网控制访问。
- 加密 - 通过上传证书、密钥和可选的 CRL 配置 TLS。
- 备份 - 创建、恢复和删除节点快照；查看下一次计划备份。
- 历史记录 - 设置审计日志、历史数据和趋势数据的保留期限，并使用预设单位和每天 3 次的更新限制。
- 维护 - 按时区、日期和小时定义每周维护时段。
- 升级 - 扩展计算/存储资源，或管理试用/付费状态。

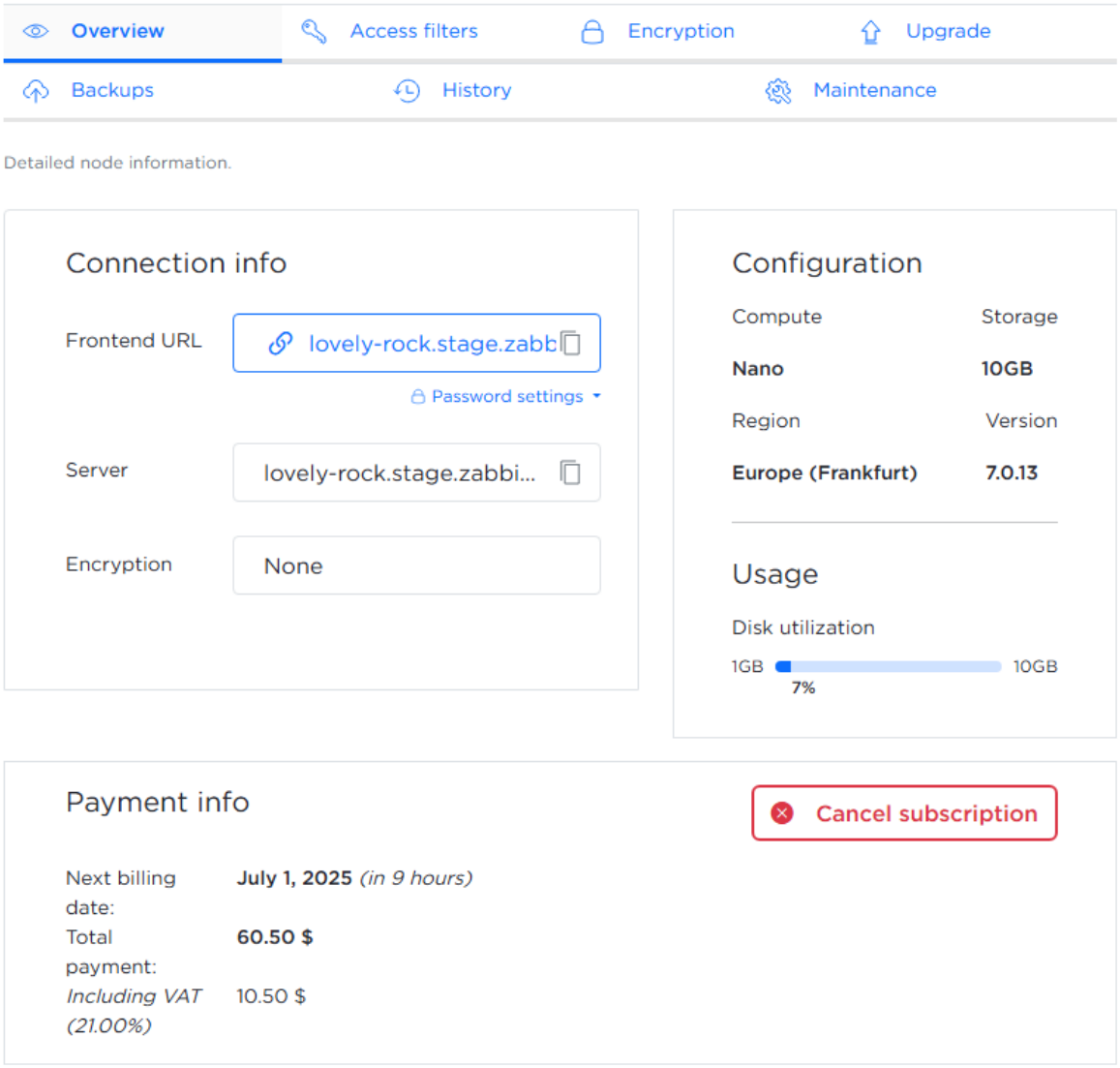


Note:
根据节点类型（服务器或 proxy）、查看页面的用户类型以及节点状态（试用/非试用），某些选项卡可能不可用。

概览选项卡

概览选项卡提供有关节点的详细信息。显示的信息取决于节点是服务器还是 proxy。

服务器节点　服务器节点的概览选项卡包含以下部分：



连接信息：

- 可点击的前端 URL，并提供复制到剪贴板选项
- 用户名
- 节点密码及其设置
- 服务器主机名，并提供复制到剪贴板选项
- IPv4 地址，并提供复制到剪贴板选项
- IPv6 地址，并提供复制到剪贴板选项
- 当前加密方法

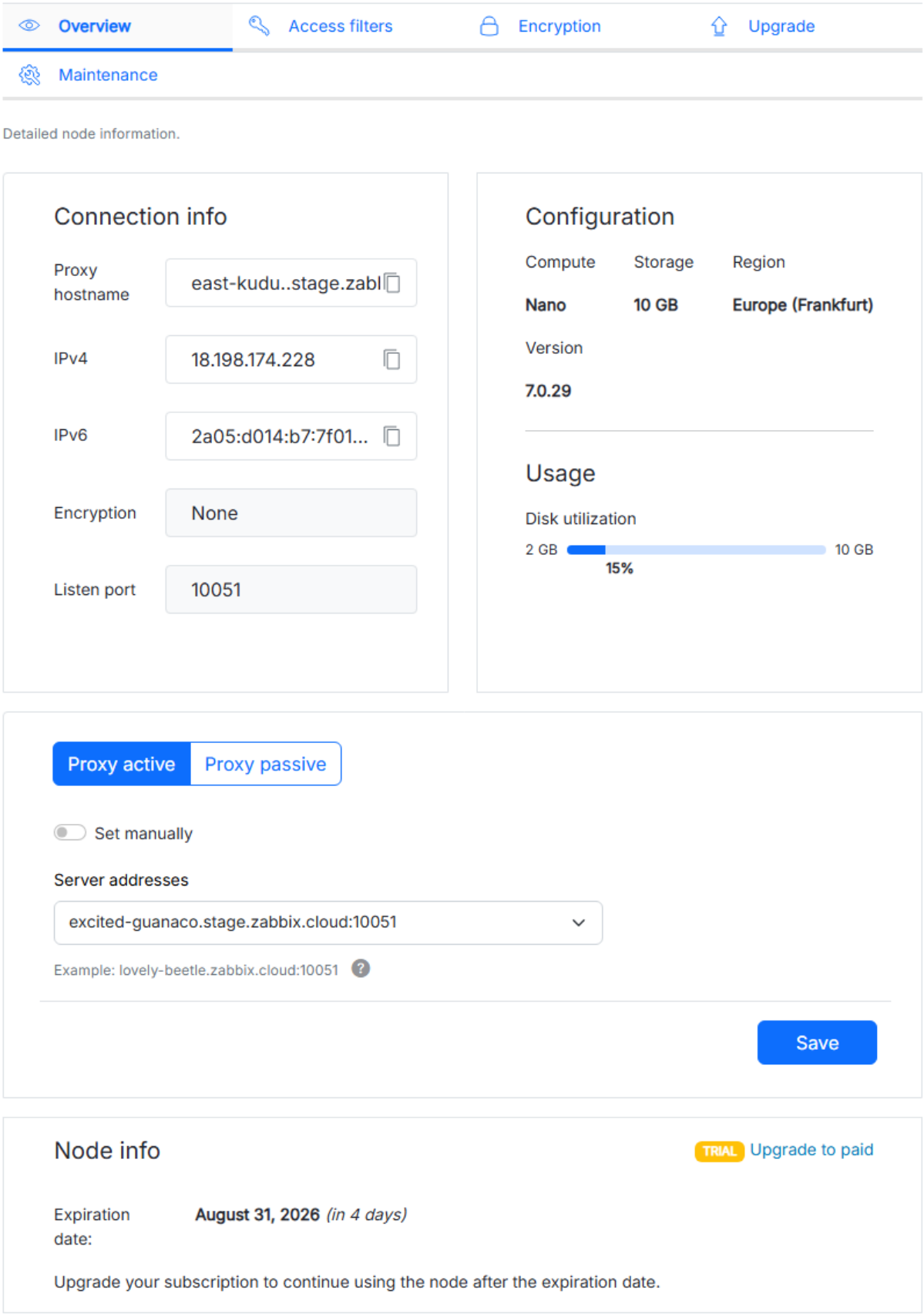
配置和使用情况：

- 计算层级、最大存储空间、区域
- Zabbix 组件版本
- 磁盘使用率

节点信息/付款信息：

- 到期日期（适用于试用节点和预付费节点）
- 节点完成初始化后，根据节点类型显示相应的上下文按钮：- 升级为付费版 - 适用于处于免费试用期的节点 - 取消订阅 - 适用于具有有效订阅的付费节点 - 重新订阅付费版 - 在订阅已取消时显示 - 对于预付费节点，不显示操作按钮。预付费节点有效期较长，且无法取消

Proxy 节点　Proxy 节点的概览选项卡包含以下部分：



连接信息：

- Proxy 主机名，支持复制到剪贴板
- IPv4 地址，支持复制到剪贴板
- IPv6 地址，支持复制到剪贴板
- 当前加密方法
- 当前监听端口

配置和使用情况：

- 计算层级、最大存储空间、区域
- Zabbix 组件版本
- 磁盘使用率

Proxy 模式：

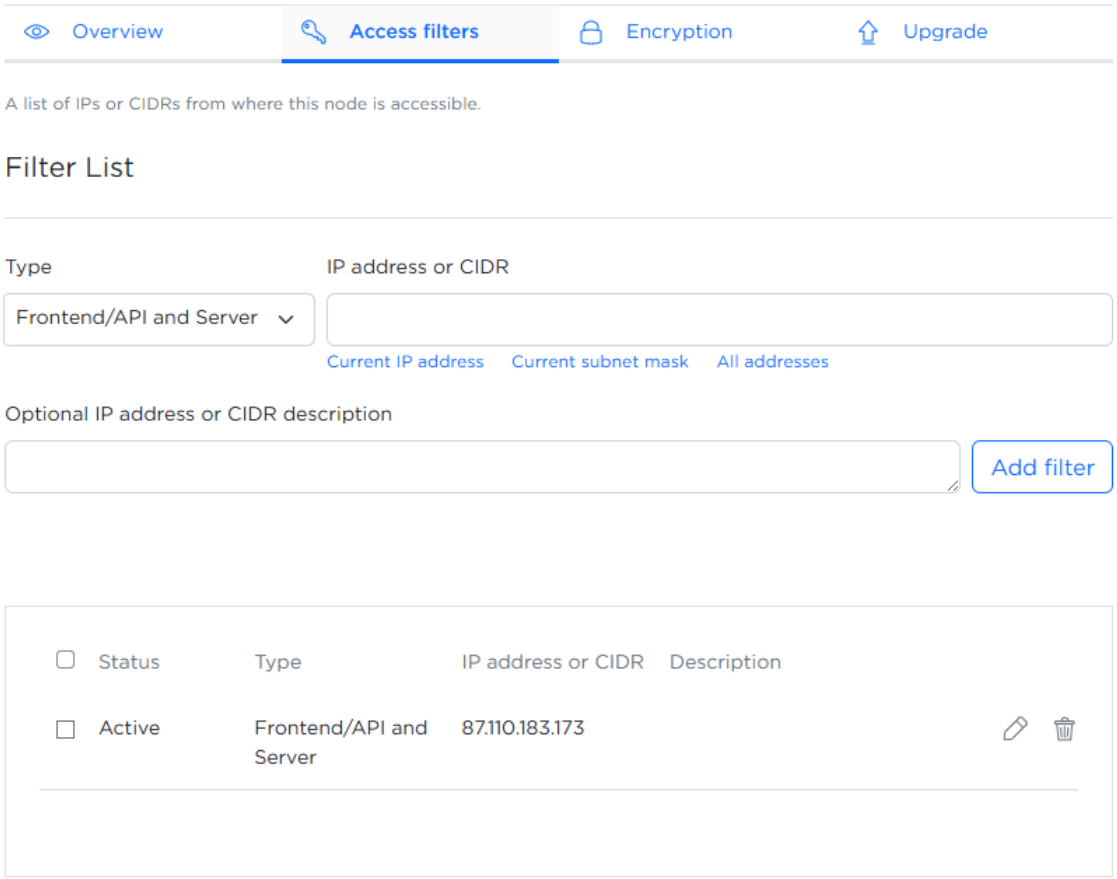
- 在主动和被动 proxy 模式之间切换
- 从服务器地址下拉列表中选择服务器地址，或启用手动设置以手动输入服务器地址

节点信息/付款信息：

- 到期日期（适用于试用节点和预付费节点）
- 节点完成初始化后，根据节点类型显示相应的按钮：- 升级为付费版 - 适用于处于免费试用期的节点 - 取消订阅 - 适用于具有有效订阅的付费节点 - 重新订阅付费版 - 在订阅已取消时显示 - 对于预付费节点，不显示操作按钮；此类节点有效期较长且无法取消订阅

访问过滤器选项卡

访问过滤器选项卡可让您按 IP 地址或子网限制对 Zabbix 节点的访问。



要配置访问过滤器，请填写以下字段：

- 类型 - 对于服务器节点，使用下拉菜单选择要限制的 Zabbix 组件：仅前端/API、仅服务器，或两者。对于 proxy 节点，仅提供入站 *IP*。它定义了允许到达 proxy 的传入网络请求。
- *IP* 地址或 *CIDR* - 手动输入 IP 地址或 CIDR，或选择相关选项自动填写该字段。
- 可选的 *IP* 地址或 *CIDR* 描述 - 为每个 IP 或子网添加可选描述。

填写完字段后，单击添加过滤器。添加的过滤器会显示在下方的表格中，其中包括状态、类型、IP 地址或子网以及描述。

Warning:
在节点初始化期间，系统会自动为您使用的 IP 地址创建访问过滤器。如果您之后从其他 IP 或网络进行连接，则必须在此处添加该地址，否则将被阻止。

您可以：

-  - 编辑过滤器。

-  - 删除过滤器，或使用复选框删除多个过滤器。

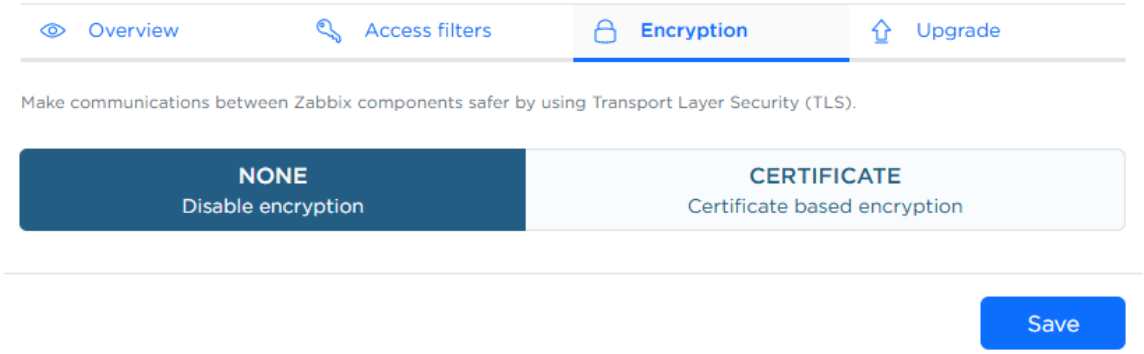
加密选项卡

加密选项卡用于通过上传所需的证书文件，为 Zabbix 组件之间的通信配置 TLS 加密。

要使用证书启用安全通信，请选择 *CERTIFICATE*，然后从计算机上传以下文件：

- 根 *CA* 证书 - 用于验证对等证书的受信任 CA 证书。
  点击 *Choose file*，选择包含受信任根 CA 证书的文件（通常为 `.crt` 或 `.pem` 文件，采用 PEM 编码）。
- 证书链 - 表示您自己的证书的信任链。
  点击 *Choose file*，选择包含您的证书及任何中间 CA 证书的 PEM 编码文件（例如 `.crt` 或 `.pem`）。
- 私钥 - 与您的证书对应的私钥。
  点击 *Choose file*，选择采用 PEM 编码的 `.key` 文件。
- 已吊销的证书（可选）- 如果您维护证书吊销列表（CRL），请点击 *Choose file* 并选择 CRL 文件，以指定不再受信任的证书。

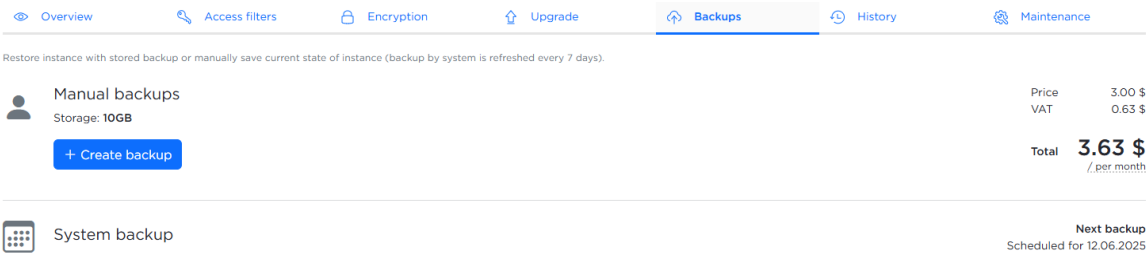
选择文件后，点击 *Save*。
这些文件将被上传、验证并应用，以配置 Zabbix 组件之间的 TLS 加密。



备份选项卡

备份选项卡用于查看、创建、恢复和删除 Zabbix 节点的备份。

您可以随时从任意已保存的备份恢复实例，或创建新的手动备份。该选项卡还会显示下一次计划备份的日期。



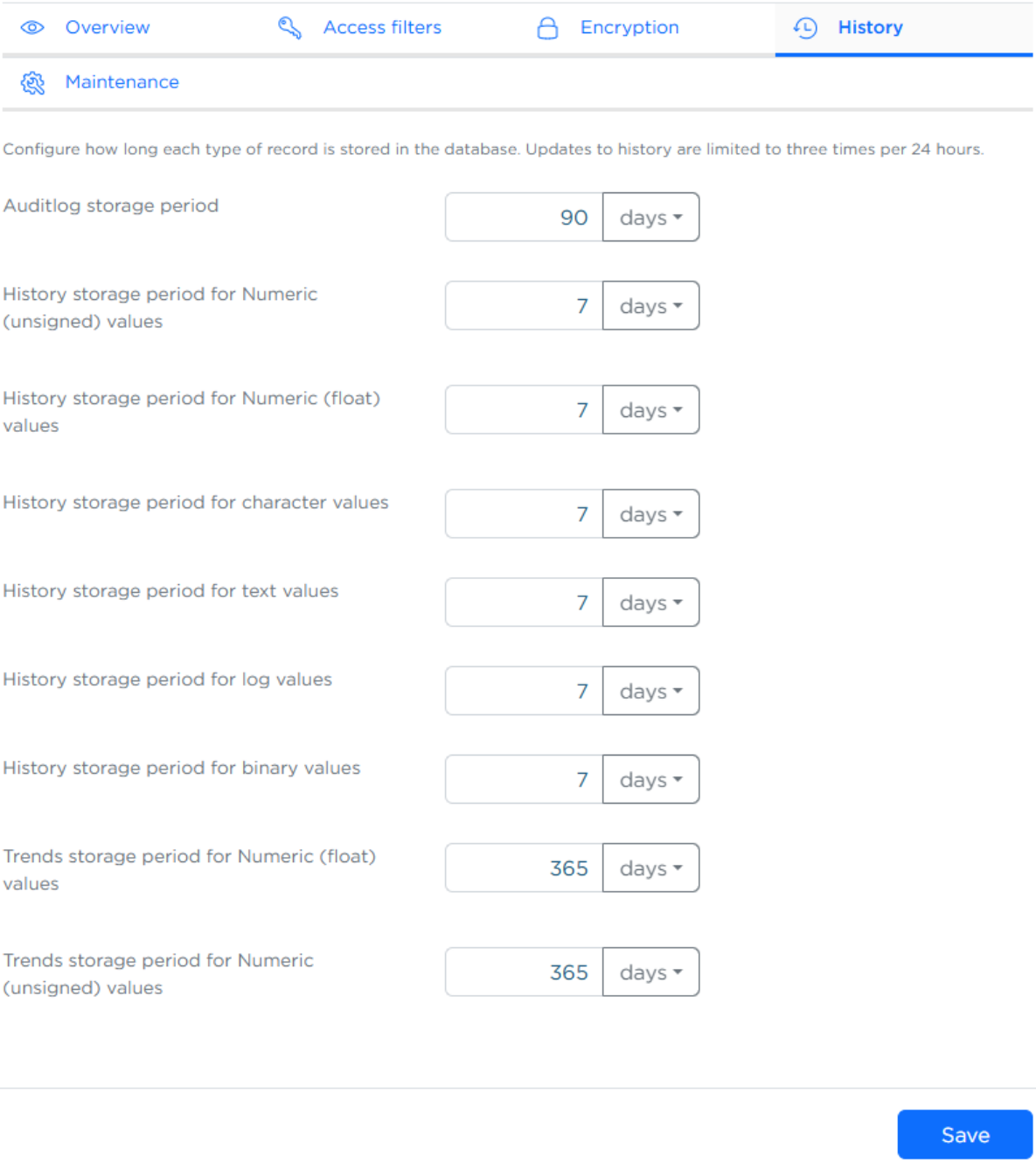
创建备份：

1. 单击创建备份按钮。
2. 在弹出窗口中输入可选的描述。
3. 单击创建进行确认。

Warning:
额外备份将根据存储使用量按月收费。

历史记录选项卡

历史记录选项卡用于配置不同类型历史数据的存储时长。



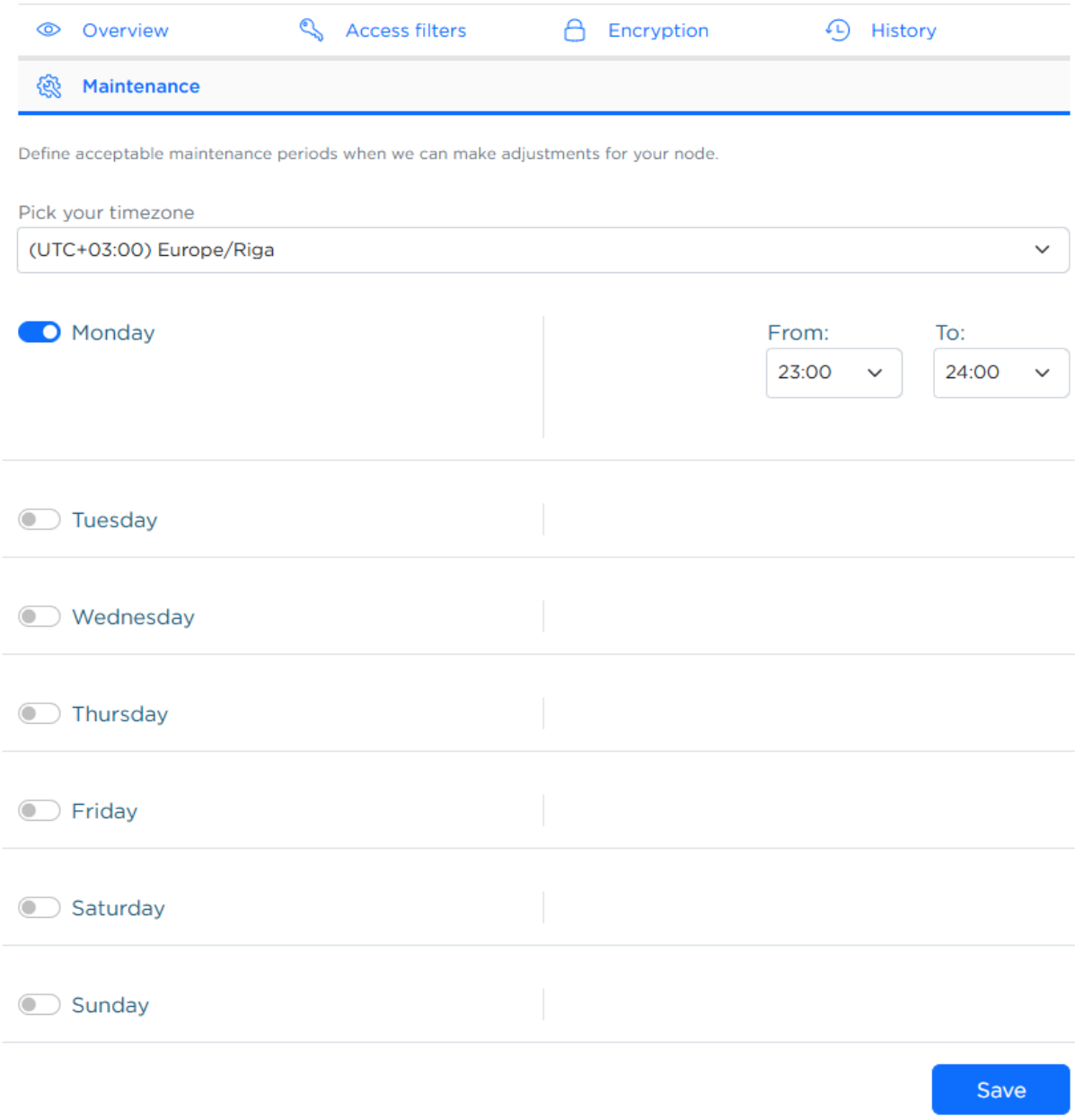
您可以为每种数据类型（例如数值、文本和日志）设置存储期限，并选择单位（天、月或年）。每 24 小时最多可以更改三次。

更新数值后，单击保存以应用更改。

Warning:
较长的保留期限可能会增加磁盘使用量，并影响性能。建议根据实际数据和分析需求配置保留期限。
 Zabbix Cloud 使用数据库分区管理数据存储，而不是传统的 Zabbix housekeeper。这可以实现更快的性能和更可预测的保留处理。

维护选项卡

维护选项卡用于定义可接受的维护时段，以便在您的节点上执行更新或调整。

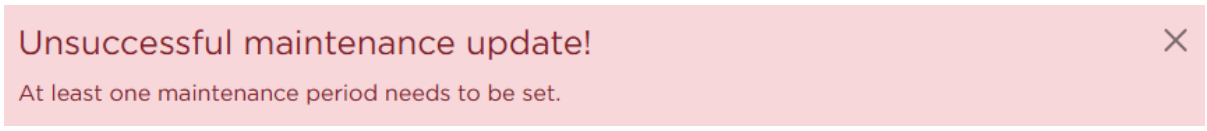


您必须：

- 从下拉列表中选择您的时区。
- 至少选择一天和一个维护小时窗口。

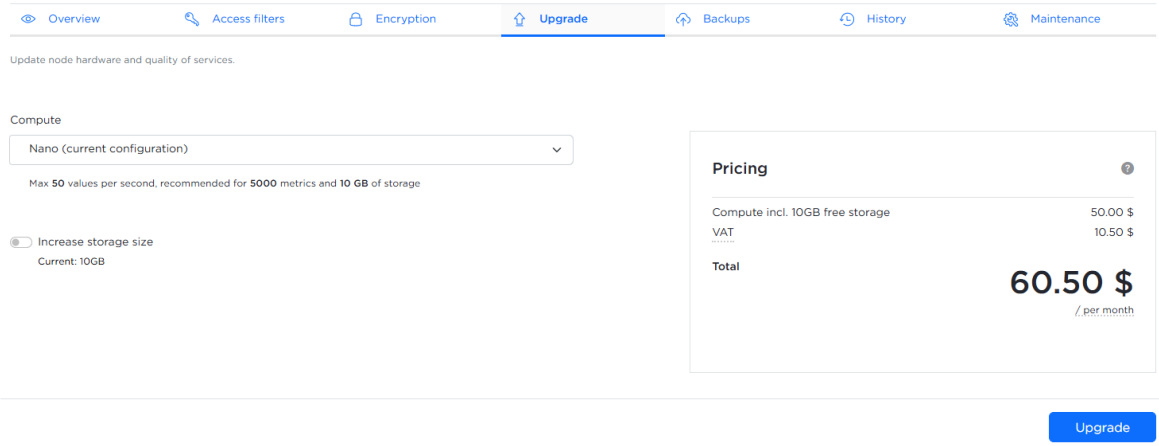
点击保存以应用更改。

如果未选择日期或时间窗口，将显示错误。



升级选项卡

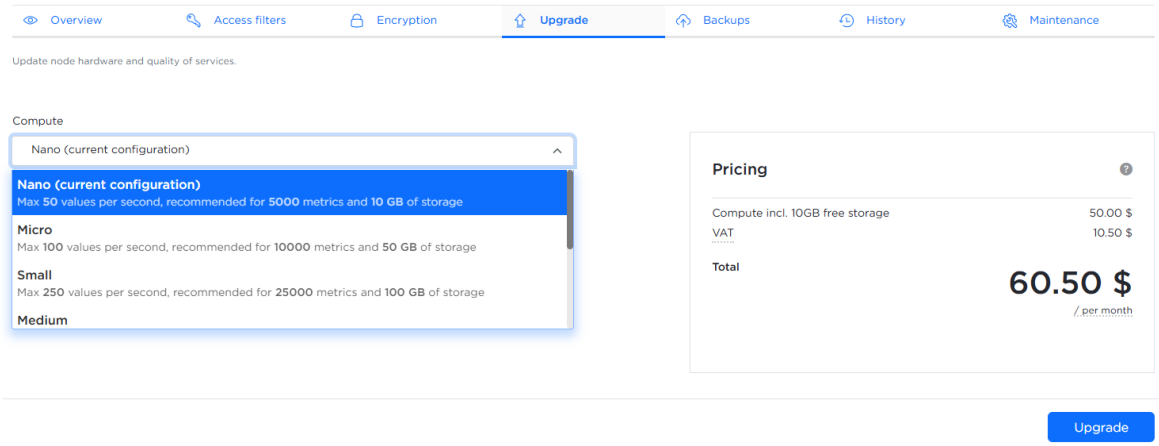
升级选项卡用于扩展节点资源或管理订阅。



您可以提高计算层级或增加磁盘大小。请注意，磁盘大小每 24 小时只能增加一次，且无法通过 UI 减少（如需缩减磁盘，请联系 Zabbix 支持）。免费试用节点可以转换为付费订阅。

提高计算层级：

1. 点击 计算下拉菜单。



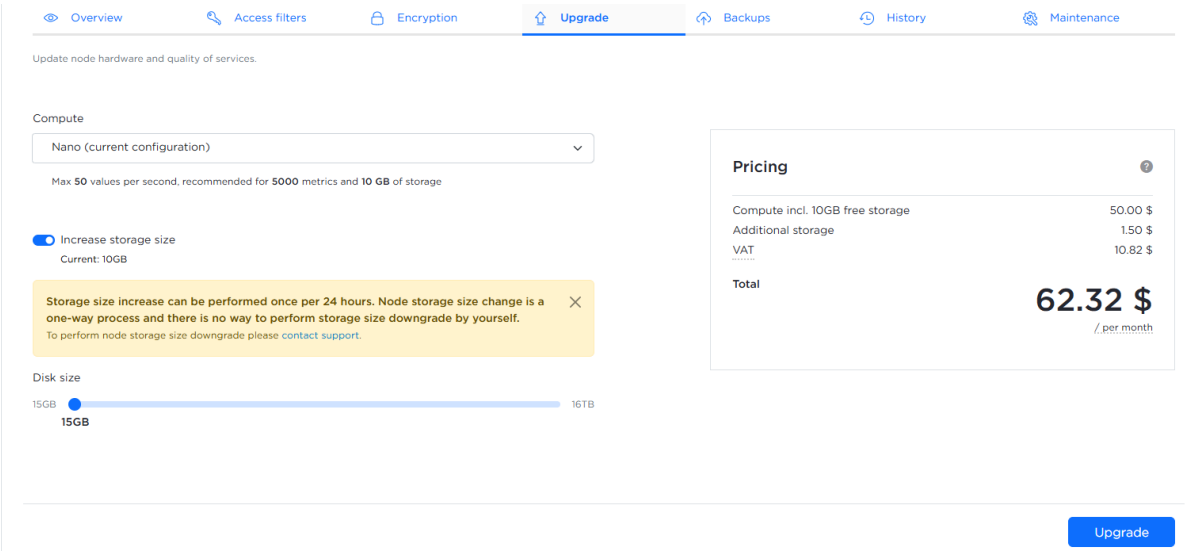
2. 选择所需的配置层级。

3. 点击 升级按钮。

增加磁盘大小：

1. 点击 增加存储大小单选按钮。

2. 将 磁盘大小滑块调整到所需值。



3. 点击 升级。

> **Note:**
> 确认更改前，定价表中会显示更新后的每月总价格。

添加用户

简介

以下示例说明了如何将用户添加到您的组织和节点中。

> **Note:**
> 复制、清除和重置前端密码只能由所有者执行。

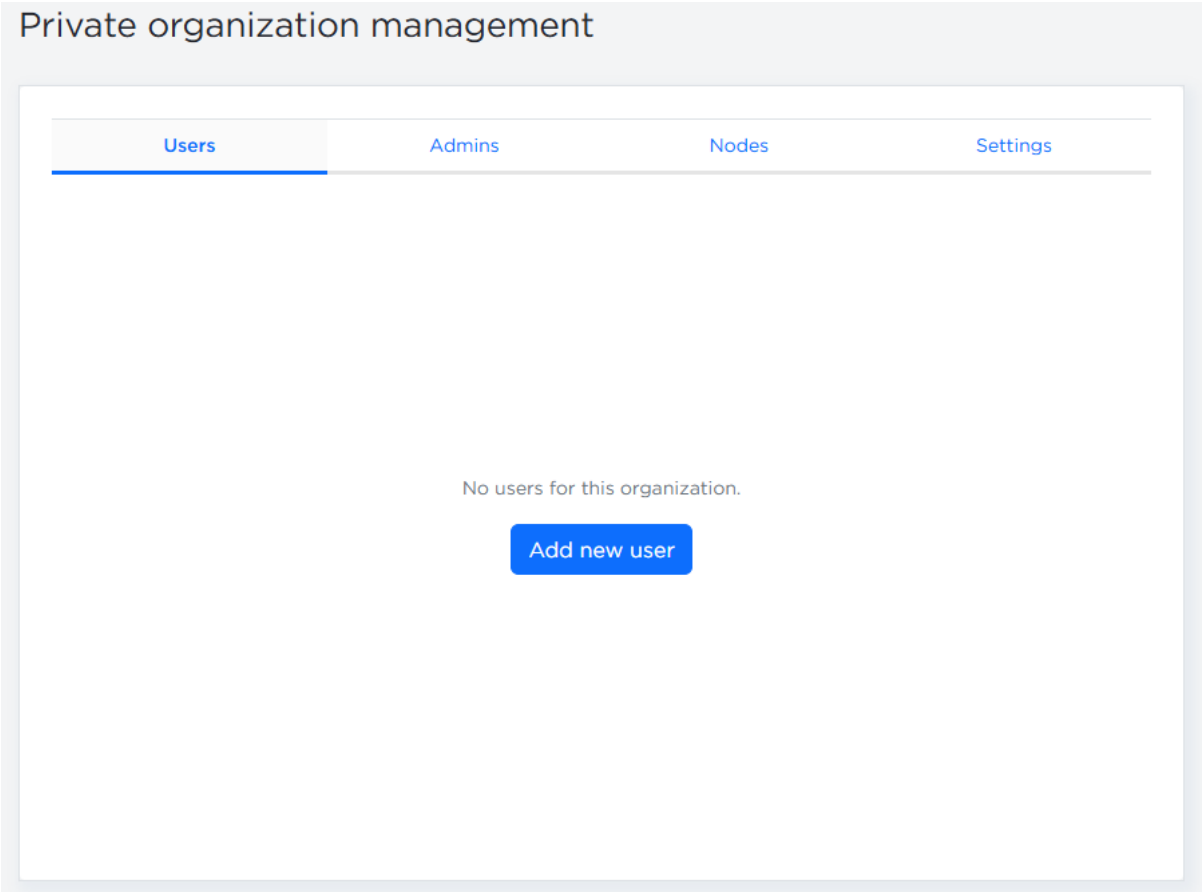
向组织添加用户

与 Zabbix 中的用户类似，您可以在 Zabbix Cloud 中创建用户，并为其分配角色，以控制他们在组织及其节点中的可执行操作。

要向您的组织添加用户：

1. 打开 *All organizations* 页面。

2. 点击特定组织的  设置按钮，进入其管理菜单。



3. 在 *Users* 或 *Admins* 选项卡中，点击 *Add new user*。

4. 输入用户的电子邮件地址并选择角色：*Administrator* 或 *User*。

请注意，选择 *User* 角色会将该用户添加到组织中，但不会向其授予对任何节点的访问权限。

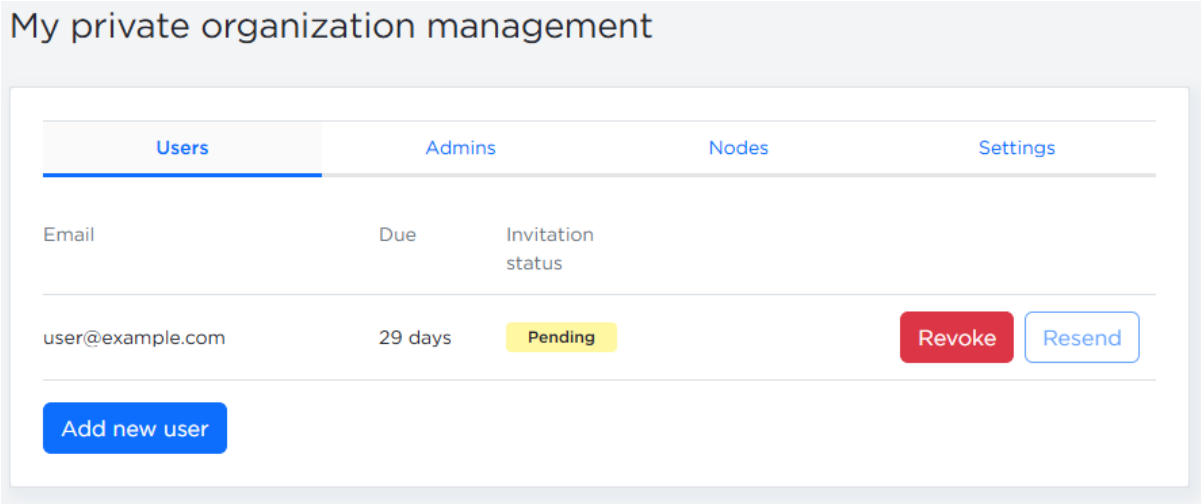
每种角色授予的访问级别不同：

- **User** - 可以访问分配给他们的节点，并可执行不会产生费用的配置任务，例如：
  - 管理访问过滤器
  - 配置加密

  - 设置历史记录/趋势保留期限
  - 安排维护
- **Administrator** - 拥有 User 的所有权限，还可以执行可能产生费用的操作，例如：
  - 创建和恢复备份
  - 升级或降级节点资源（计算层级和磁盘大小）
  - 取消订阅
- **Owner** - 对组织拥有完全控制权。除拥有 Administrator 的所有权限外，Owner 还可以：
  - 创建和销毁节点
  - 添加或移除 User 和 Administrator
  - 重命名或删除组织
  - 管理账单详细信息

Attention:
Owner 的付款方式将用于支付所有服务费用。如果需要银行确认，则必须由 Owner 授权该交易，即使该交易是由 Administrator 发起的。

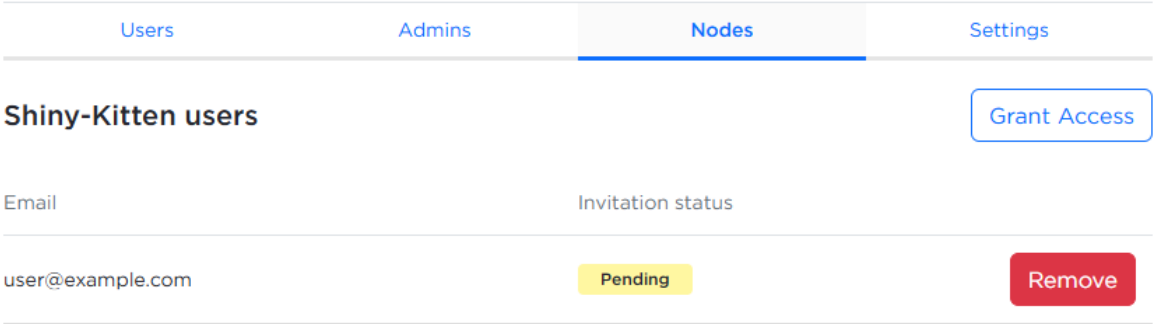
受邀用户将以 *Pending* 状态添加。
您可以撤销邀请或重新发送邀请（每 10 分钟只能重新发送一次——如果达到此限制，按钮将被禁用）。



向节点添加用户

要将用户添加到节点：

1. 转到 *Nodes* 选项卡。
2. 单击 *Grant Access*。
3. 从下拉列表中选择一个已添加到组织的用户。
4. 单击 *Grant access*。



或者，您也可以从*Node configuration*窗口授予访问权限：

1. 单击  设置按钮。
2. 选择 *Add user*。

3. 选择一个用户并单击 *Grant access*。

受邀用户必须使用接收邀请的同一电子邮件地址创建 Zabbix Cloud 账户，以接受邀请。如果用户在该电子邮件地址下已经拥有 Zabbix Cloud 账户，则必须登录并确认邀请。邀请与特定电子邮件地址绑定，无法通过其他账户接受。

Zabbix Cloud 的主要区别

简介

Zabbix Cloud 是一个基于云的平台，允许用户在云中运行独立的 Zabbix 实例（云节点）。

与常规的本地部署 Zabbix 相比：

- 无需安装或维护。
- 云节点由 Zabbix 全面托管。
- 云节点可在几分钟内启动。

该服务采用按需付费模式（提供免费试用）。

对比表　虽然本地部署的 Zabbix 安装与云节点的大多数功能存在重叠，但两者之间也有一些差异。下面提供了汇总对比表。后续章节将详细介绍其中的部分项目。

功能 / 方面	Zabbix Cloud	Zabbix 本地部署
发布周期	与 LTS 版本同步	您可以选择 Zabbix 版本（稳定版、测试版、自定义构建）
易用性	开箱即用的平台，完全由 Zabbix 管理	需要进行安装、配置和维护
漏洞修复	自动执行操作系统/Zabbix 修补	由用户手动修补
HTTPS 证书	开箱即用的有效证书（不使用 Let's Encrypt）	默认使用自签名证书（可配置 Let's Encrypt 或其他 CA）
数据库性能（IN-SERT/UPDATE/SELECT）	自动分区以实现最高速度；不会执行长时间运行的 DELETE	手动分区和清理（可能存在长时间运行的 DELETE 事务）
性能调优	由 Zabbix Cloud 团队预先调优（无变更日志）	可全面查看并控制调优参数
防火墙控制	由 Zabbix Cloud 团队管理；GUI/trapper 端口不会公开暴露	可完全控制防火墙规则和端口暴露
端口 *25* 的出站流量	默认阻止，以防止垃圾邮件和滥用；请改用 SMTP 端口 587 或 465	Zabbix 不会限制；取决于您的网络和托管服务提供商
数据库隔离与保护	高度隔离，不提供直接的数据库/SSH 访问	取决于您的网络和主机安全性
高可用性（*HA*）	运行在高可用云平台上，可使用 Zabbix proxy 提高数据采集的弹性	由用户管理 HA
SNMP traps	仅可通过专用 proxy 使用（无 HA/负载均衡的 trap 接收器）	可直接使用或通过 proxy 使用，支持 HA/负载均衡
SNMP 轮询	不使用 proxy 时较难实现（每台设备都需要 NAT/自定义端口）	原生支持SNMP 轮询；proxy 可选
保留设置	只能通过 UI 配置历史数据/趋势数据/审计（不支持 API 或配置文件）	可在 Zabbix 服务器配置文件中配置，或通过API 配置
自定义脚本	不支持`AlertScriptsPath`、`ExternalScripts`、前端和社区模块	完全支持（脚本路径、模块、集成）
ODBC 监控	仅支持 PostgreSQL（驱动程序 `{postgresql}`）；MariaDB 插件作为占位符提供	支持 PostgreSQL、MySQL、Oracle 等数据库的 ODBC - 可完全配置
限制 *ODBC* 调用	无法实现（仅支持`StartODBCPollers=1`）；繁重的同步查询会阻塞其他工作	可全面调节轮询器数量和调度
SAML 证书上传	不支持	支持通过 UI 或 API 使用
计划报表	必须创建自己的 Email 媒介（不支持脚本媒介）	开箱即用地同时支持脚本和 Email 媒介
主动检查的主机接口	平台分配的 IP 地址（自动创建）	由您管理接口；IP 地址由您控制

功能差异　SNMP trap

SNMP trap 仅通过专用的 Zabbix proxy 提供支持。
如果需要 SNMP trap 监控，则不能对 proxy 使用自动负载均衡或高可用性，因为 SNMP trap 必须发送到单个 IP 地址。

SNMP 轮询

如果没有 proxy，SNMP 轮询要求每台设备都通过 NAT 和自定义端口暴露出来。

保留设置

历史数据、趋势数据和审计日志的保留期在云环境中只能通过 Web 界面进行配置。这些设置无法通过 `zabbix_server.conf` 或 API 进行配置。

不支持按监控项手动覆盖历史数据保留设置（分区由全局统一控制）。云节点 URL 不接受与本地部署安装相同的查询参数。

自定义脚本

Zabbix Cloud 不支持以下自定义脚本类型：

- 告警脚本（`AlertScriptsPath`）
- 外部脚本（`ExternalScripts`）
- Zabbix 服务器上的前端脚本

此外，无法安装由社区开发的前端模块。

ODBC 监控

Zabbix Cloud 仅支持 PostgreSQL 的ODBC 监控。请使用官方 Zabbix ODBC 模板，并在模板中使用以下内容定义连接字符串：

```
Driver={postgresql}
```

还安装了一个 MariaDB 插件，但目前无法正常工作。如果使用，请定义：

```
Driver={mysql}
```

Attention:
目前在 Zabbix Cloud 中，没有已知可用的 Oracle MySQL 8.0 配置。虽然简单查询（例如 SELECT 1）可能成功，但更复杂的查询会返回 SQL_ERROR。

 无法限制 ODBC 调用（仅支持 StartODBCPollers=1）。高负载的同步报表可能会影响性能，并且一次只能运行一条 SELECT。

基础设施访问

Zabbix Cloud 不提供对底层节点的 SSH 访问，也不允许直接连接数据库（例如连接到端口 3306）。所有配置、监控和故障排查活动都必须通过云 UI 或 API 进行，以确保操作系统和数据库层保持隔离并受到保护。

端口 25 上的出站流量

为防止垃圾邮件和滥用，默认情况下，Zabbix Cloud 节点到 TCP 端口 25 的出站连接会被阻止。这会影响：

- 通过端口 25 上的 SMTP 服务器发送邮件的Email 媒体类型。由于 25 是新建 Email 媒体类型时默认的 *SMTP server port*，请将其更改为 587（并将 *Connection security* 设置为 STARTTLS）或 465（SSL/TLS），具体取决于您的 SMTP 服务器支持的配置。无法使用 *Office365 relay* 邮件提供商，因为它会连接到 Microsoft 的端口 25 中继端点。
- Scheduled reports，该功能需要 Email 媒体类型。
- 云节点自身对端口 25 执行的检查，例如 `net.tcp.service[smtp]` simple checks，或网络发现规则中的 SMTP 检查。即使 SMTP 服务正在运行，这些检查也会将其报告为停止；请改为通过您自己网络中的 Zabbix proxy 或 Zabbix agent 执行此类检查。

默认的 *Cloud Email* 媒体类型不受影响。

主动检查接口

主动检查 在云中会自动创建一个主机接口，其 IP 与您的本地网络无关。默认情况下，此 IP 可能是外部 IP。为保持一致性，您可以在创建后手动删除或调整该主机接口。

SAML 证书上传

不支持SAML 身份验证，因为很少有提供商接受未加密或未签名的 SAML 消息。

计划报表

默认的 *Cloud Email* 媒介类型使用脚本传输，因此无法用于计划报表。要通过电子邮件发送报表：

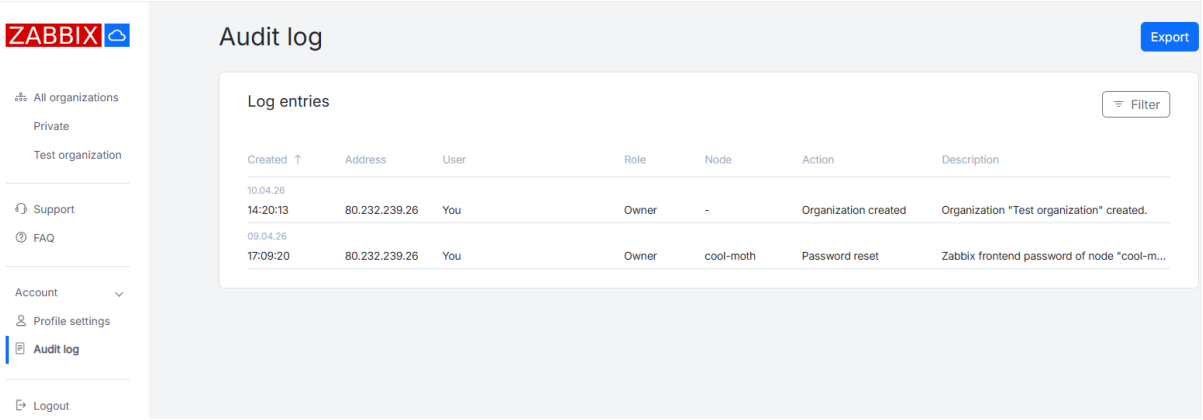
1. 创建新的 Email 媒介类型。
2. 在 用户设置 > 媒介中，将该电子邮件媒介分配给您的用户。
3. 配置计划报表时选择该媒介类型。

Attention:
请勿将端口 25 用作此媒介类型的 *SMTP* 服务器端口；请参阅端口 25 上的出站流量。

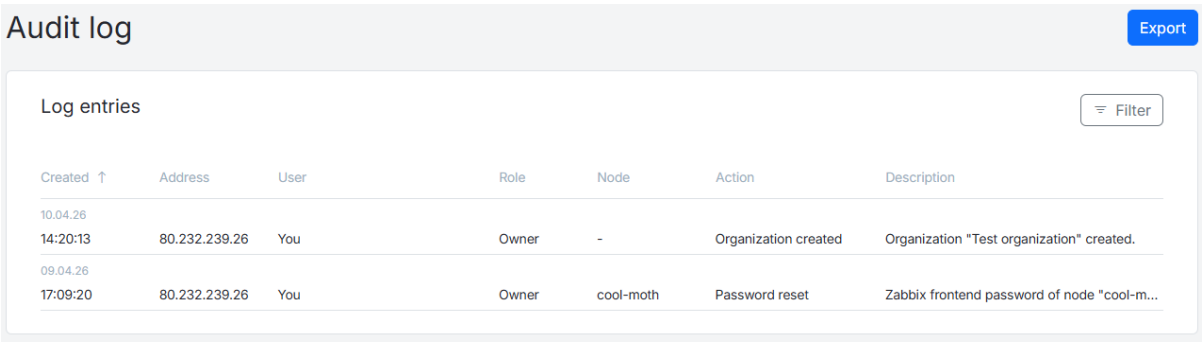
审计日志

概述　在 Zabbix Cloud 中，审计日志会记录用户和系统活动。

登录Zabbix Cloud 账户后，转到 *Account* > *Audit log* 查看审计日志记录。用户可查看的审计日志条目取决于该用户的组织角色。



查看审计日志记录　审计日志记录以表格形式显示。



该表包含以下信息：

Column	Description
Created	日志条目创建的日期和时间。显示的时间对应于 *Account* > *Profile settings* 中 *User timezone* 下拉菜单里选择的用户时区。
User	执行该操作的组织用户 名称。
Role	执行该操作的用户的角色。
Node	与该操作相关的节点名称。
Action	所执行操作的名称。
Description	所执行操作的描述。

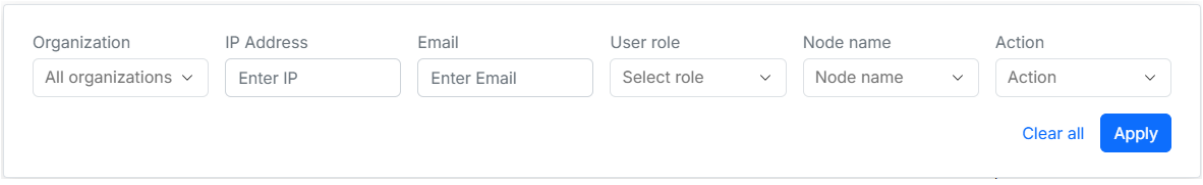
可以点击列名，按升序或降序对记录进行排序。

可使用 *Export* 按钮将已筛选 的审计日志记录导出为 `.csv` 格式。

使用筛选器

单击 *Filter* 按钮打开筛选器。

选择一个或多个筛选参数后，单击 *Apply* 以显示筛选结果。
要清除筛选器，请单击 *Clear all*。



可以根据一个或多个参数筛选记录。
可用的筛选参数如下：

参数	描述
Organization	选择所有组织或特定组织。
IP Address	指定 IP 地址。
Email	指定与用户关联的电子邮件地址。
User role	选择所有角色或特定角色：Owner、Admin 或 User。
Node name	选择所有节点或特定节点。
Action	选择特定的操作类型。

审计日志会记录以下操作：

- 创建节点
- 删除节点
- 升级节点
- 更新节点版本
- 重置密码
- 更新维护
- 更新历史记录设置
- 延长节点到期日期
- 更新*proxy* 配置
- 创建备份
- 删除备份
- 恢复备份
- 更新备份
- 添加访问筛选器
- 更新访问筛选器
- 删除访问筛选器
- 启用加密
- 禁用加密
- 取消订阅
- 续订订阅
- 订阅付款失败
- 付款需要确认
- 创建组织
- 删除组织
- 邀请管理员
- 邀请用户
- 管理员加入
- 用户加入
- 从组织中移除管理员
- 从组织中移除用户
- 将节点添加到组织
- 从组织中移除节点
- 授予节点访问权限
- 撤销节点访问权限
- 更新用户设置
- 创建付款
- 付款完成
- 付款失败
- 收取增值税
- 增值税验证失败
- 更新账单信息
- 更新付款方式
- 移除付款方式
- 创建预付费节点
- 更新预付费节点
- 事件开始
- 事件结束

设置 **Zabbix Cloud proxy**

概述　本页面介绍如何配置部署在 Zabbix Cloud 上的 Zabbix proxy，以从远程环境中的主机收集监控数据。

此过程主要适用于在 Zabbix Cloud 上创建的 Zabbix 服务器和 proxy。

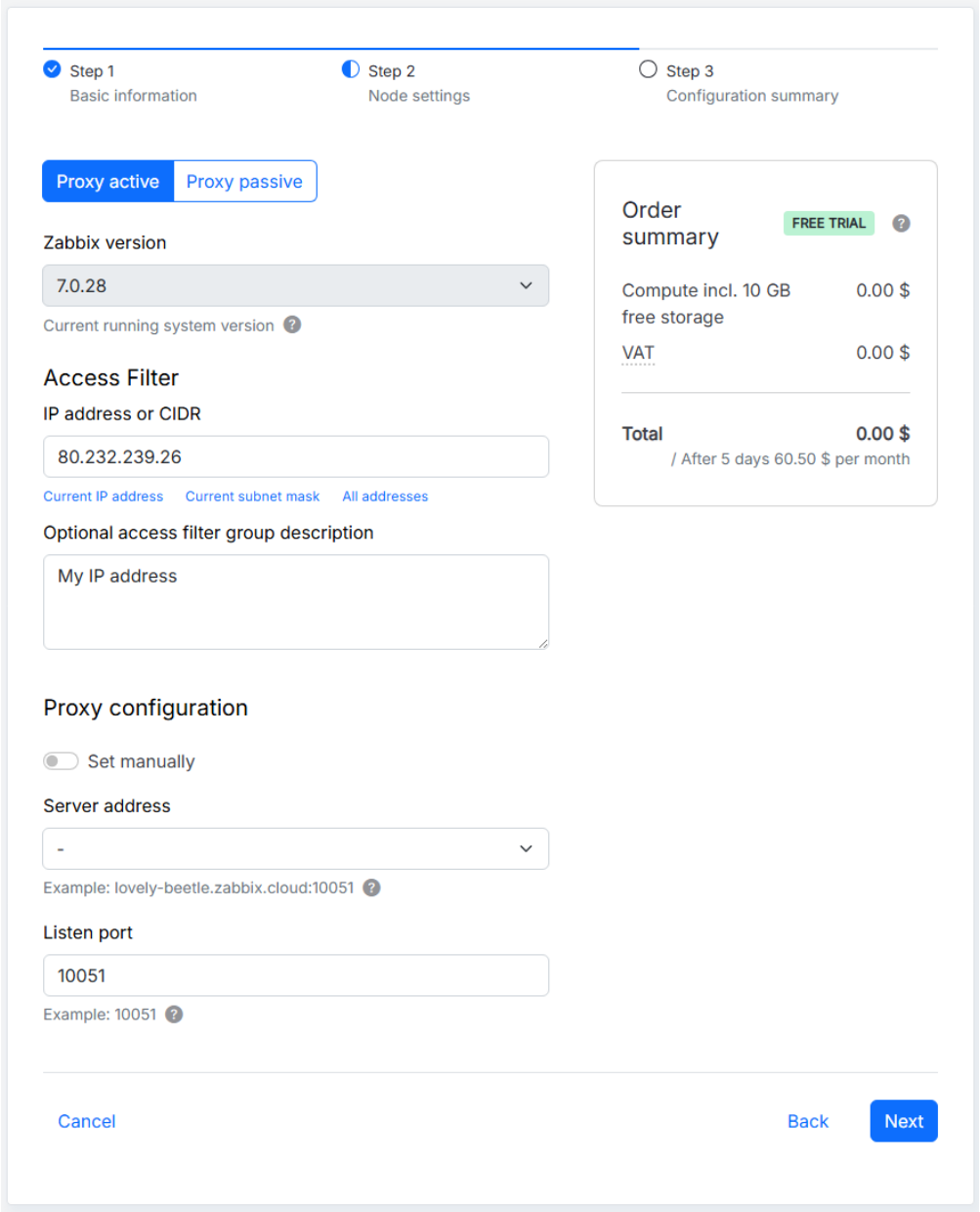
此过程还介绍了 Zabbix 服务器未托管在 Zabbix Cloud 上时所需的配置。

前提条件　已安装、配置并运行 Zabbix 服务器，或 Zabbix Cloud 上的 Zabbix 服务器节点正在运行。

创建 **proxy** 节点　创建一个Zabbix Cloud proxy 节点。

创建节点时：

1. 选择 *Proxy*。
2. 根据 proxy 与 Zabbix 服务器通信的方式，选择 *Proxy active* 或 *Proxy passive*。
3. 配置节点的 IP 地址或 CIDR 访问过滤器（可在创建节点后完成）。
4. 指定 proxy 将连接的 Zabbix 服务器。



5. 完成节点创建过程。

有关创建 proxy 节点的更多信息，请参阅在云中部署 Zabbix。

配置访问过滤器　Zabbix Cloud 使用访问过滤器控制哪些 IP 地址或子网可以连接到云节点。

在节点初始化期间，系统会自动为用于访问节点的 IP 地址创建访问过滤器。这不一定包括受监控主机或其他 Zabbix 组件的地址。请手动添加所需的地址。

有关更多信息，请参阅节点配置 — 访问过滤器。

Proxy 节点

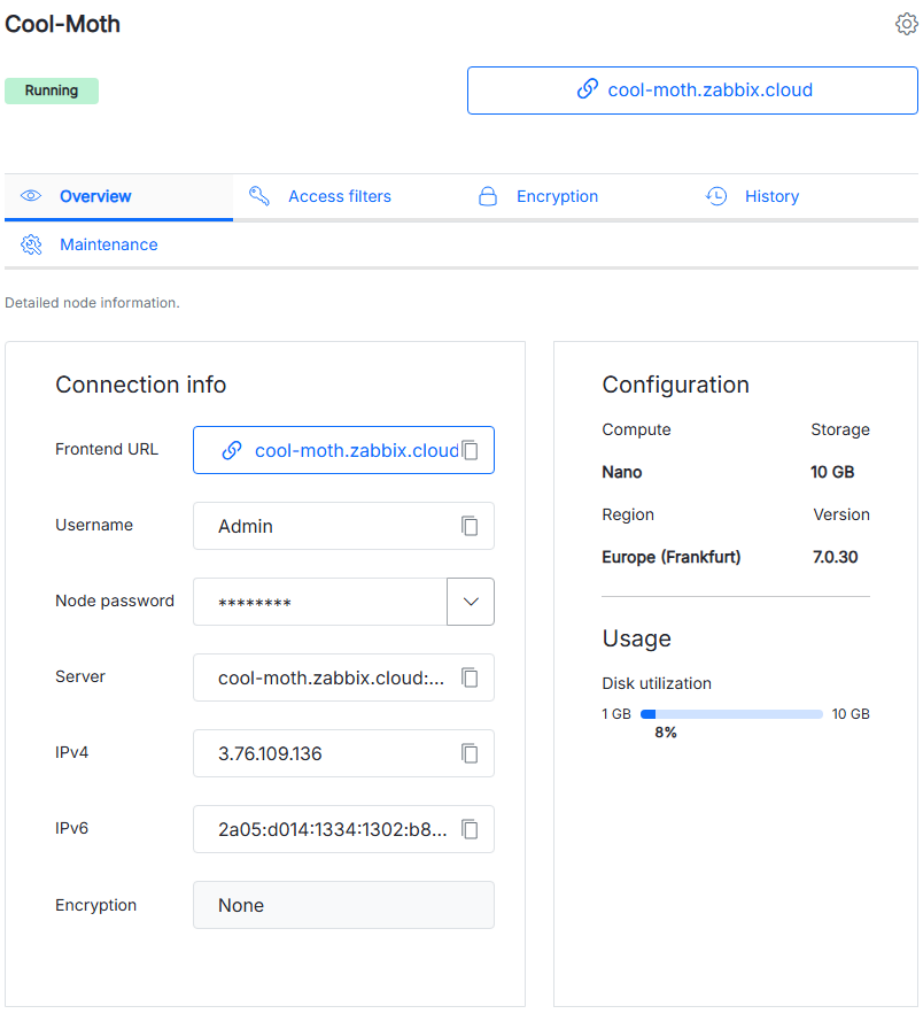
添加预期会连接到 proxy 的 IP 地址或子网。

如果受监控主机使用 agent 主动检查，agent 会发起到 proxy 的连接，因此必须在 proxy 节点的访问过滤器中允许受监控主机的 IP 地址或子网。

如果将 *Proxy Passive* 设置为 proxy 模式，Zabbix 服务器会连接到 proxy，因此必须在 proxy 节点的访问过滤器中允许 Zabbix 服务器的 IP 地址或子网。

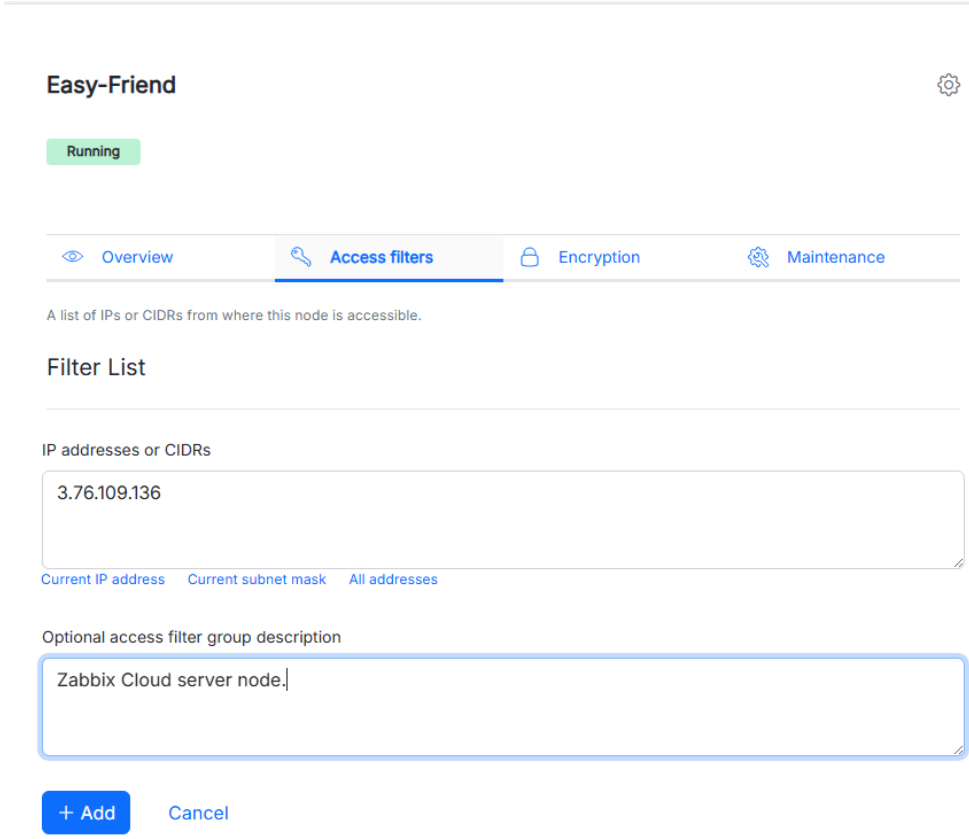
如果选择了 *Proxy passive*，并且使用服务器节点：

1. 导航到服务器节点的 *Overview* 选项卡，并复制该节点的 IPv4 或 IPv6 地址。

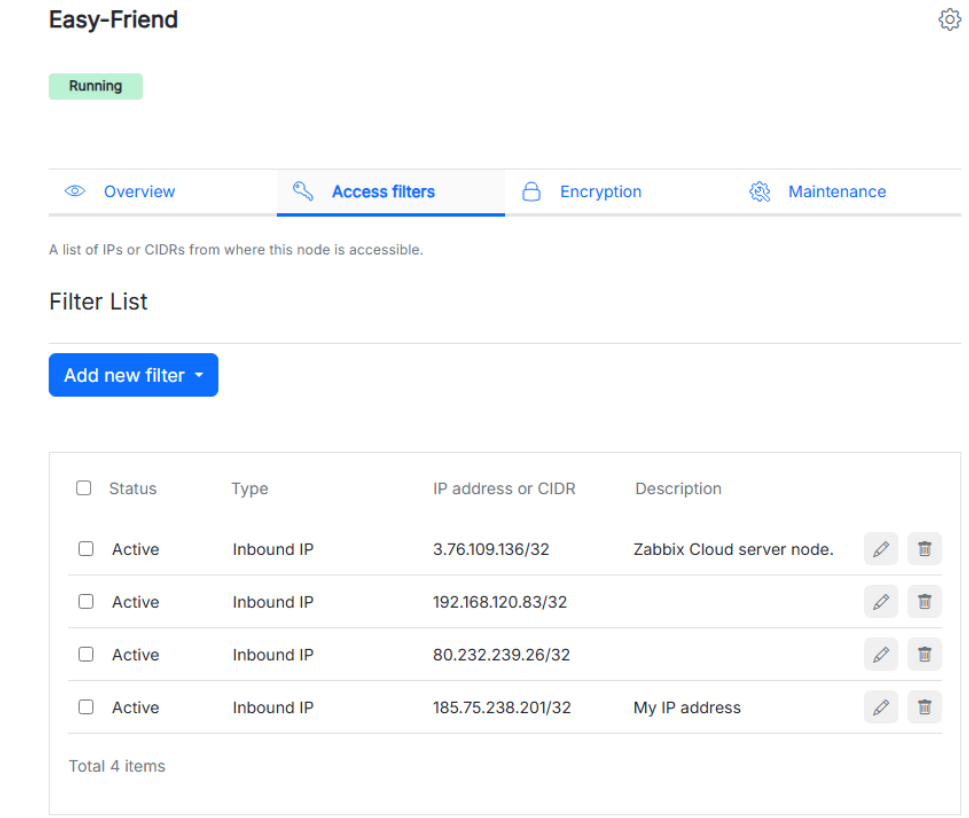


2. 导航到 proxy 节点的 *Access filters* 选项卡，点击 *Add new filter* > *Add*。

3. 将复制的地址粘贴到 *IP addresses or CIDRs* 字段，并在 *Optional access filter group description* 字段中添加描述。



4. 点击 *Add*。



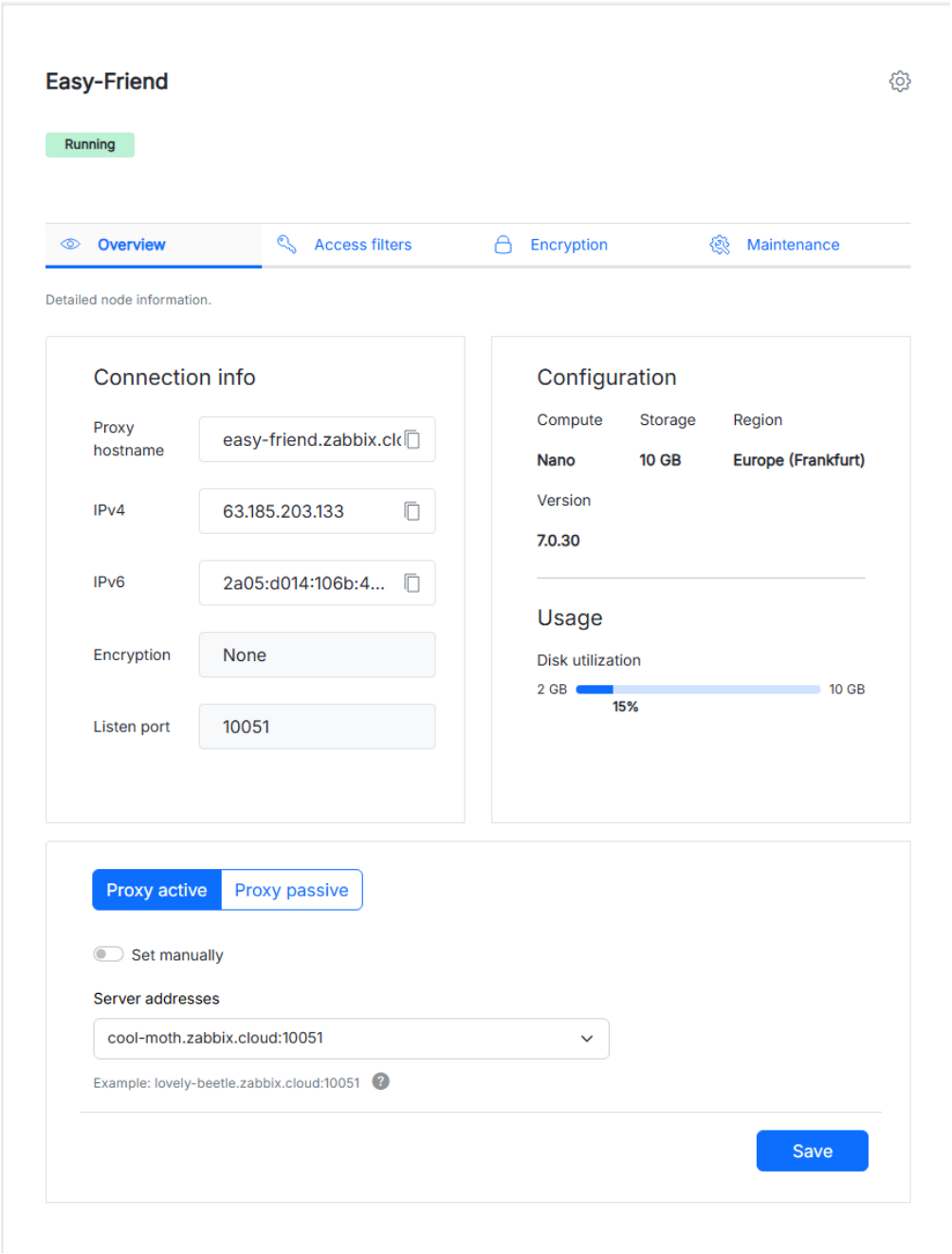
对使用主动检查的 Zabbix agent 的 IP 地址执行步骤 2-4。

服务器节点

如果 Zabbix 服务器托管在 Zabbix Cloud 上，还需要在服务器节点上配置访问过滤器。

对于主动模式下的 proxy，请将 proxy 节点的 IP 地址或 CIDR 添加到服务器节点的访问过滤器中，因为 proxy 会发起与服务器的连接。

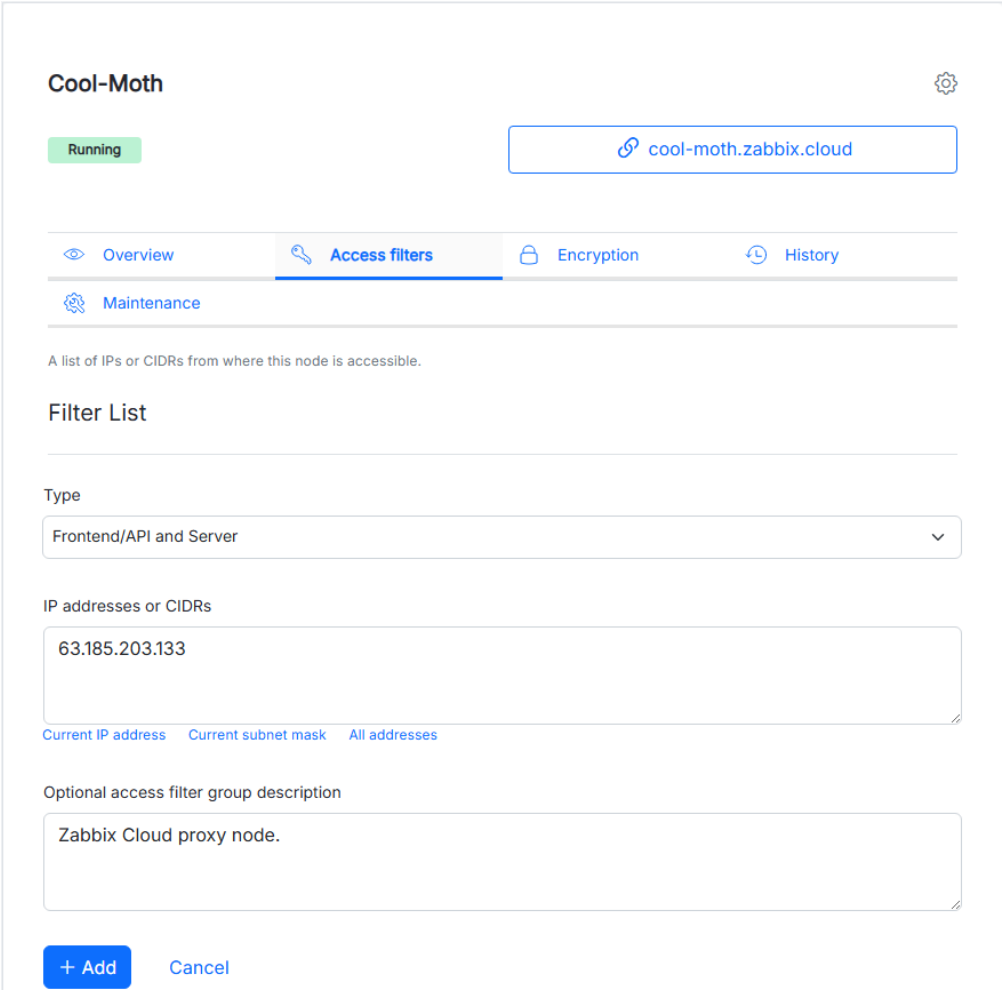
1. 转到 proxy 节点的 *Overview* 选项卡，并复制该节点的 IPv4 或 IPv6 地址。



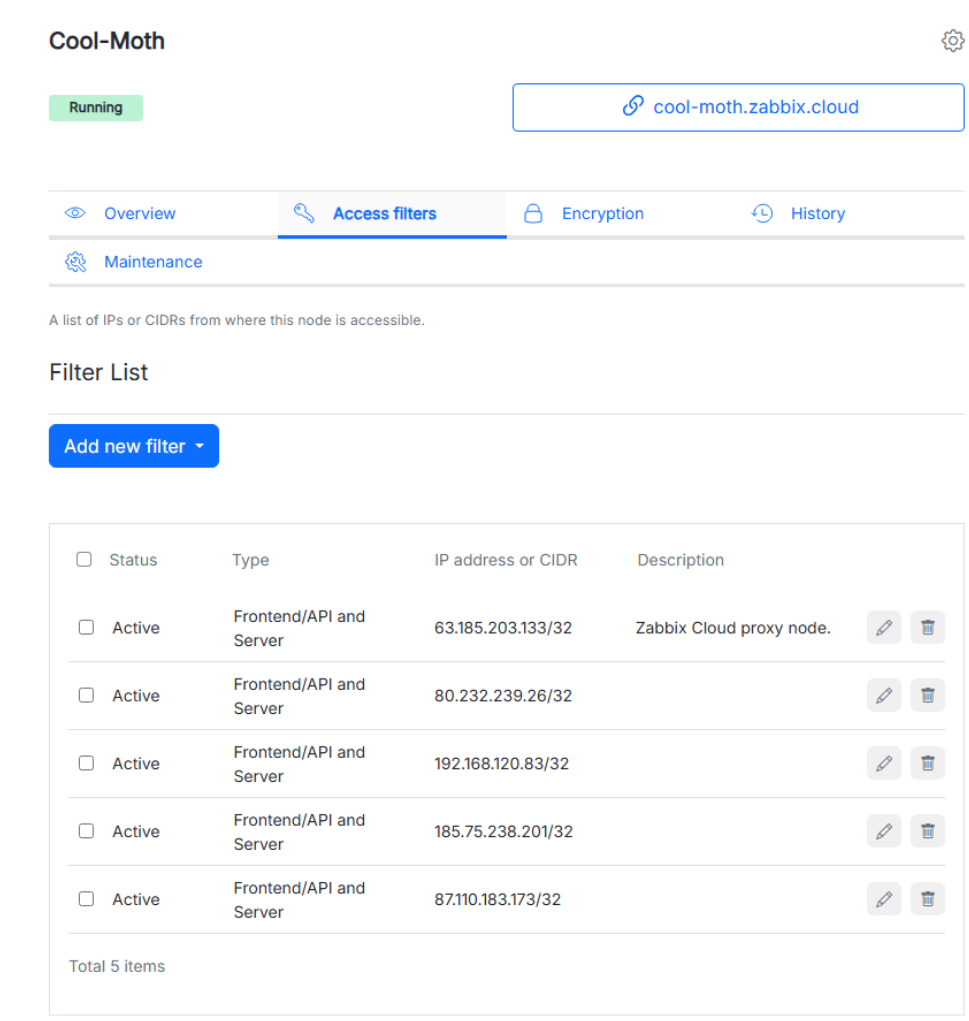
2. 转到服务器节点的 *Access filters* 选项卡，点击 *Add new filter* > *Add*。

3. 将复制的地址粘贴到 *IP addresses or CIDRs* 字段，并在 *Optional access filter group description* 字段中添加描述。

4. 选择适当的过滤器类型。


Cool-Moth
Running
cool-moth.zabbix.cloud
Overview
Access filters
Encryption
History
Maintenance
A list of IPs or CIDRs from where this node is accessible.
Filter List
Type
Frontend/API and Server
IP addresses or CIDRs
63.185.203.133
Current IP address
Current subnet mask
All addresses
Optional access filter group description
Zabbix Cloud proxy node.
+ Add
Cancel

5. 点击 *Add*。



配置 **proxy**　必须在 Zabbix 前端中配置 proxy，才能与 Zabbix 服务器通信。

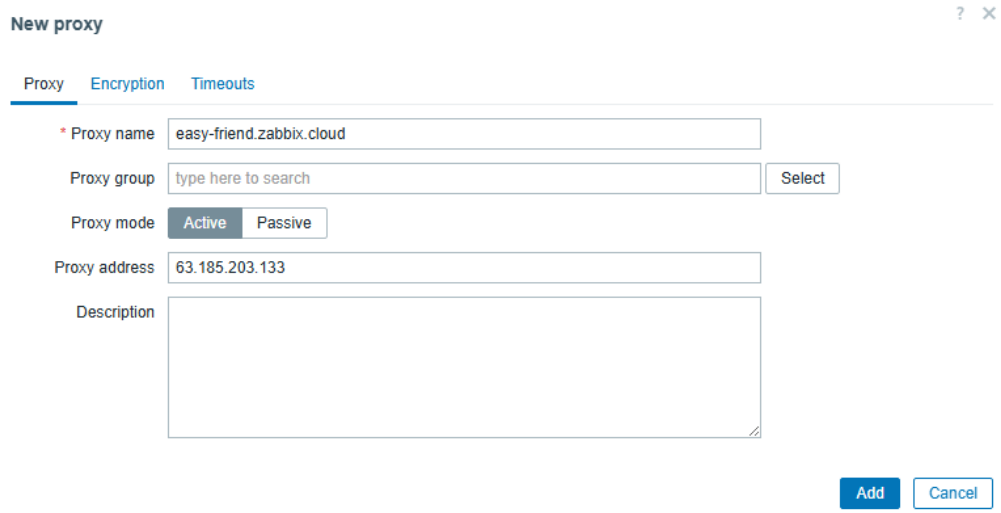
在 Zabbix 前端中打开 *Administration* > *Proxies*，点击 *Create proxy*，然后根据需要配置 proxy：

1. 在 *Proxy name* 字段中添加 proxy 名称。该名称必须与为 proxy 配置的 proxy 主机名匹配。手动输入，或从 proxy 的 *Overview* 选项卡中复制。

2. 选择与创建 proxy 时所选模式对应的 Proxy mode。

3. 对于主动式 proxy，输入 proxy 的 IP 地址或 DNS 名称。

4. 点击 *Add*。



有关更多信息，请参阅使用 proxy 进行分布式监控。

配置受监控的主机 添加应通过 proxy 进行监控的主机。

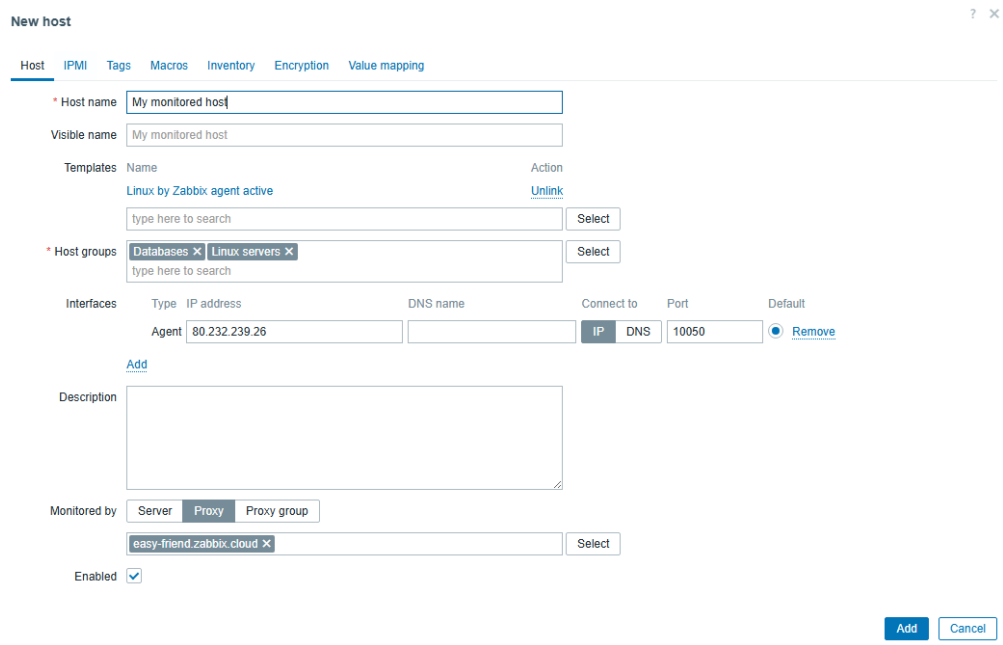
1. 转到 数据采集 > 主机。

2. 创建主机或打开现有主机。

3. 在 监控方中，选择 *Proxy*，然后输入 proxy 名称进行搜索并选择，或者点击 选择按钮打开 proxy 列表并从中选择。

4. 配置主机接口和其他参数。

5. 点击 添加或 更新。



配置为由 proxy 监控的主机会通过该 proxy 发送或接收监控数据，具体取决于监控项类型和 agent 模式。

配置 **Zabbix agent** 当受监控主机使用 Zabbix agent 或 Zabbix agent 2 时，请配置 agent 以便与相应的 Zabbix 组件通信。

配置取决于 agent 使用的是被动检查还是主动检查。

被动 agent 检查

对于被动检查，Zabbix proxy 会向 agent 发起连接并请求监控数据。

打开 Zabbix agent 配置文件，并更新 `Server` 参数，以允许来自 Zabbix Cloud proxy 的 IP 地址的连接。

例如：

```
Server=<proxy_IP>
```

确保被监控主机允许 proxy 通过 agent 端口接收入站 TCP 连接，该端口默认为 10050。

proxy 必须能够通过网络访问被监控主机。

主动 agent 检查

使用主动检查时，agent 会主动与 Zabbix proxy 建立连接，并发送收集到的监控数据。

打开 Zabbix agent 配置文件，并使用 Zabbix Cloud proxy 的地址更新 `ServerActive` 参数。

例如：

`ServerActive=<proxy_IP>:<proxy_listen_port>`

将 `<proxy_IP>` 替换为 Zabbix Cloud proxy 的 IP 地址，将 `<proxy_listen_port>` 替换为创建 proxy 节点时配置的监听端口。默认监听端口为 10051。

如果在 `ServerActive` 中指定了 proxy，agent 会从 proxy 获取主动检查配置，而不是直接从 Zabbix 服务器获取。

请确保被监控主机可以通过配置的监听端口与 proxy 建立出站 TCP 连接。

有关 agent 被动检查和主动检查的更多信息，请参阅被动和主动 agent 检查。

外部 **Zabbix** 服务器　Zabbix Cloud proxy 也可以与部署在 Zabbix Cloud 外部的 Zabbix 服务器通信。

在这种情况下，外部 Zabbix 服务器必须能够根据所选的 proxy 模式与云 proxy 通信。

对于主动模式的 proxy，proxy 会发起与外部 Zabbix 服务器的连接。配置外部 Zabbix 服务器以接受来自 proxy 的连接，并确保 proxy 可以通过网络访问该服务器。

对于被动 proxy，Zabbix 服务器会发起与 proxy 的连接。将外部 Zabbix 服务器的IP 地址或子网添加到 proxy 节点的访问过滤器中，并确保服务器可以通过网络访问 proxy。

外部 Zabbix 服务器不使用 Zabbix Cloud 访问过滤器。请改为在外部环境中配置所需的防火墙和网络规则。

受监控的主机可以位于与外部 Zabbix 服务器相同的环境中、另一个网络中，或 proxy 可访问的网络中。

当 agent 使用主动检查时，将其 `ServerActive` 参数配置为指向 Zabbix Cloud proxy。当 agent 使用被动检查时，将其 `Server` 参数配置为允许来自 proxy 的连接。

验证配置　配置 proxy、agent、访问过滤器和网络连接后，验证 proxy 是否正在与 Zabbix 服务器通信。

1. 转到 管理 > *Proxy*。

2. 检查 proxy 的状态。

3. 转到 监测 > 最新数据，检查是否有可用的最新监控项值。

如果 proxy 未与服务器通信，请检查：

- proxy 模式；
- 服务器地址和端口；
- proxy 访问过滤器；
- 服务器访问过滤器（当服务器托管在 Zabbix Cloud 上时）；
- agent 配置；
- 防火墙和路由规则；
- 在 `Server` 或 `ServerActive` 中配置的地址。

移动应用

移动应用

Zabbix 移动应用可让您通过移动设备监控 Zabbix 环境。查看问题、检查问题详情、更新问题，并在需要关注时接收推送通知。

获取应用（即将推出）

为应用设置 **Zabbix**

设置 **Zabbix bridge adapter**　安装 Zabbix bridge adapter，以便移动设备连接到 Zabbix。

添加设备　在 Zabbix 前端中添加您的移动设备，以便将其连接到 Zabbix 服务器。

设置推送通知　问题发生时，在您的移动设备上接收告警。

了解该应用

应用概览　了解使用 Zabbix 移动应用的基础知识。

安全的移动访问　保护应用、移动设备和 Zabbix bridge adapter，或撤销丢失或遭入侵设备的访问权限。

为应用设置 **Zabbix**

要使用 Zabbix 移动应用（即将推出），您必须先：

1. 设置 Zabbix bridge adapter。
2. 设置推送通知。
3. 在 Zabbix 前端中添加您的移动设备。

以下各节将介绍每个步骤。

设置完成后，请参阅保护移动访问，以确保应用和 Zabbix bridge adapter 的安全。

设置 **Zabbix bridge adapter**　要使您的 Zabbix 服务器能够与移动应用通信，您必须设置Zabbix bridge adapter。

bridge adapter 可以使用软件包安装在与 Zabbix 服务器相同的系统上，也可以安装在其他系统上：

> **Attention:**
> 以下步骤适用于 Ubuntu 24.04。对于其他操作系统，请从 获取 Zabbix 页面获取适用于您操作系统的仓库和软件包安装命令（例如 `wget`、`dpkg`、`apt`），并使用这些命令安装 `zabbix-bridge-adapter` 软件包。

1. 以 root 权限启动新的 shell 会话：

```
sudo -s
```

2. 安装 Zabbix 仓库：

```
wget https://repo.zabbix.com/zabbix/8.0/release/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest_8.
dpkg -i zabbix-release_latest_8.0+ubuntu24.04_all.deb
apt update
```

3. 安装 Zabbix bridge adapter：

```
apt install zabbix-bridge-adapter
```

4. 编辑Zabbix 服务器配置文件：

```
EnableMobileDevices=1
BridgeAdapterURL=https://bridge-adapter.example.com:10054/rpc
```

5. 重启 Zabbix 服务器，例如（参见其他选项）：

```
systemctl restart zabbix-server
```

6. 从报表 > 系统信息复制 Zabbix 服务器 ID；您将在下一步中使用该 ID。

7. 编辑Zabbix bridge adapter 配置文件：

```
ZabbixAPIURL=https://example.com/zabbix/api_jsonrpc.php
ServerId=019fb1f9-22bd-74ea-bc03-51efb8b76aba
DataDir=/var/lib/zabbix-bridge-adapter
```

> **Attention:**
> DataDir 中指定的目录必须具有 0700 权限，其中存储的所有文件必须具有 0600 权限。

8. 查看并接受 Zabbix bridge adapter 的最终用户许可协议（EULA）：

```
zabbix_bridge_adapter license show
sudo -u zabbix-bridge-adapter zabbix_bridge_adapter -c /etc/zabbix/zabbix_bridge_adapter.conf license acce
```

9. 启动 Zabbix bridge adapter：

```
systemctl start zabbix-bridge-adapter
```

> **Attention:**
> 启动时，Zabbix bridge adapter 会使用服务器 ID 将您的 Zabbix 服务器注册到集中托管的 Zabbix Bridge。注册状态存储在`DataDir` 中的状态文件内。如果重新安装 bridge adapter 时状态文件丢失，可能无法注册同一个 Zabbix 服务器。为避免这种情况，请在重新安装 bridge adapter 前备份状态文件，并在安装后将其恢复。

设置推送通知　要在受监控环境中出现需要关注的情况时，在移动设备上接收警报：

1. 创建推送通知媒体类型（或启用默认媒体类型）。
2. 创建操作（或启用默认的向 *Zabbix* 管理员报告问题触发器操作）。
3. 添加设备，以便自动将媒体类型添加到您的用户中（或手动添加用户媒体）。

添加设备　要在设备上使用 Zabbix 移动应用，您必须先在 Zabbix 前端中添加该设备。有关说明，请参阅：

- 用户 > 设备（作为超级管理员，添加您自己或其他用户的移动设备）
- 用户设置 > 设备（作为任何用户，添加您自己的移动设备）

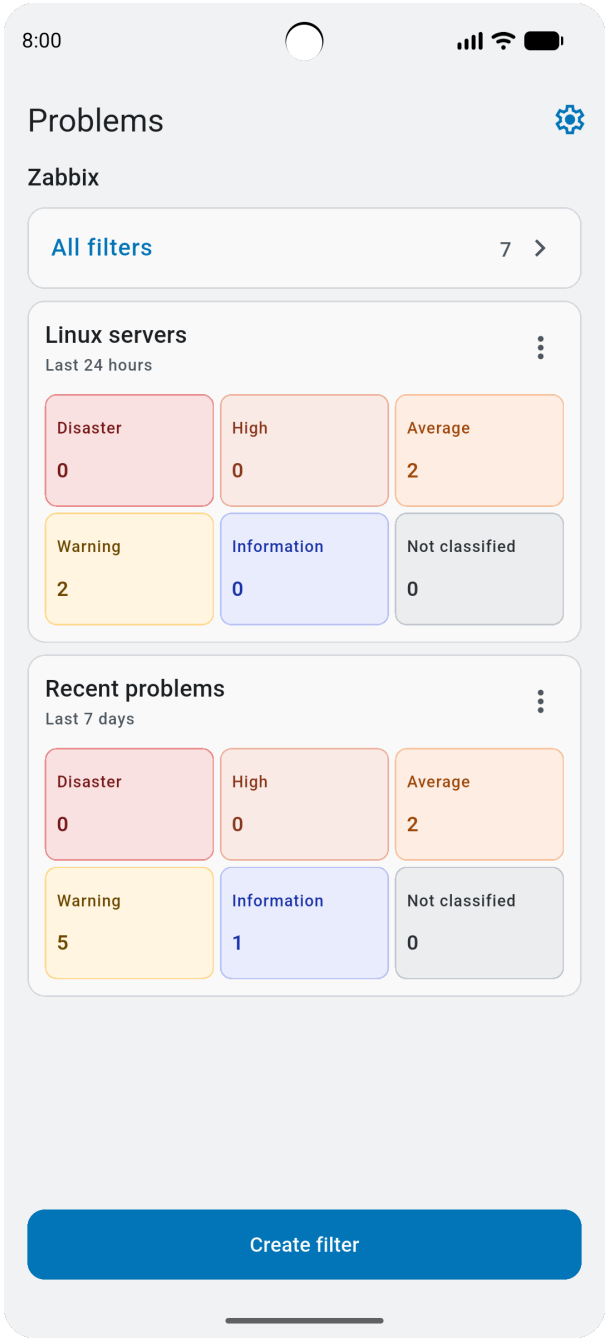
应用概览

Zabbix 移动应用（即将推出）可让您通过移动设备监控 Zabbix 环境。查看问题、检查问题详情、更新问题，并在需要关注时接收推送通知。

本页面概述了移动应用及其使用方法。

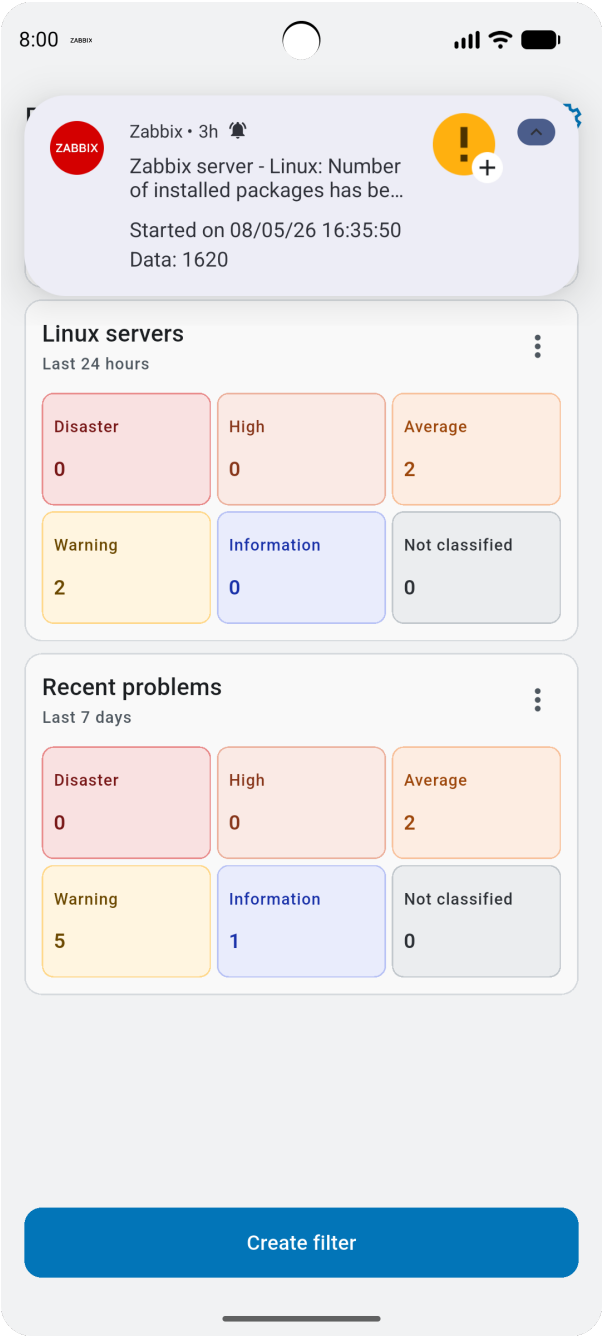
问题　问题屏幕根据您固定的筛选器显示问题概览。该屏幕包含：

- 设置图标 - 点击以查看应用设置。
- 所有筛选器 - 点击以查看和管理筛选器。
- Zabbix 服务器名称（如果安装Zabbix 前端时未配置，则显示为 *Zabbix*）。
- 固定的筛选器 - 点击筛选器名称或严重性以打开问题列表屏幕。
 点击三点菜单以管理筛选器：编辑、复制、取消固定或删除筛选器。
 请注意，固定的筛选器按字母顺序排列，且无法编辑和删除 最近的问题筛选器。
- 创建筛选器 - 点击以创建新的筛选器。



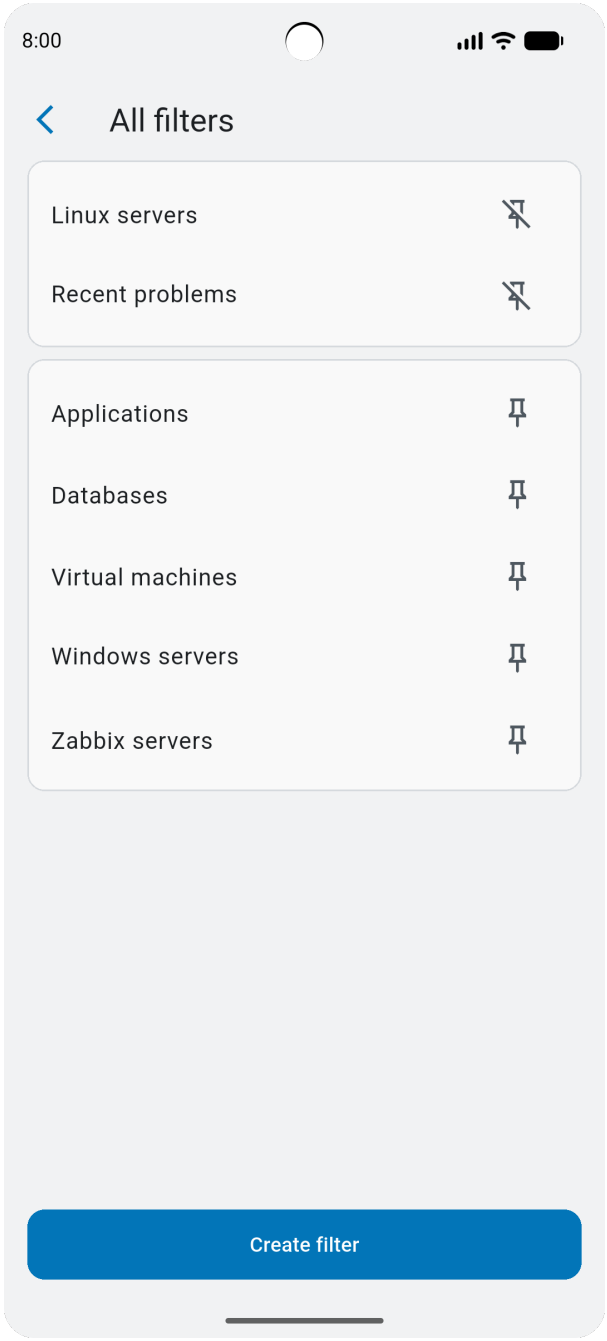
通知　当 Zabbix 服务器检测到问题时，可以向您的移动设备发送通知。通知包含发生问题的主机、名称，以及根据为推送通知 媒介类型配置的消息模板生成的消息。

点击通知以打开问题详情屏幕。



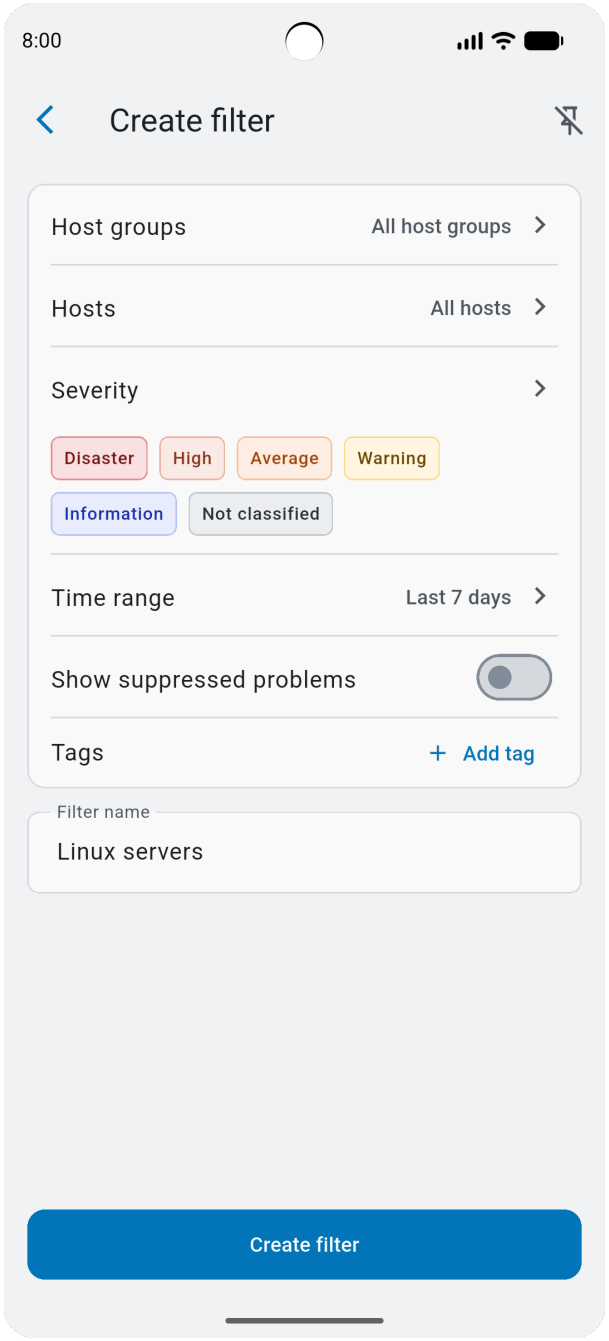
所有筛选器　所有筛选器界面用于管理筛选器。该界面包含：

- 筛选器名称 - 点击以编辑筛选器。
- 图钉图标 - 点击以将筛选器固定到问题界面，或取消固定。
- 创建筛选器 - 点击以创建新的筛选器。



创建筛选器　创建筛选器界面可用于创建筛选器，其选项与监控 > 问题中的选项类似（请参阅详细信息）。您还可以设置时间范围，以指定显示未解决问题的时长。

点击创建筛选器以保存筛选器。新筛选器会固定到问题界面；点击图钉图标可更改此设置。



问题列表　问题列表屏幕显示与筛选条件匹配的问题（例如，最近的问题）。

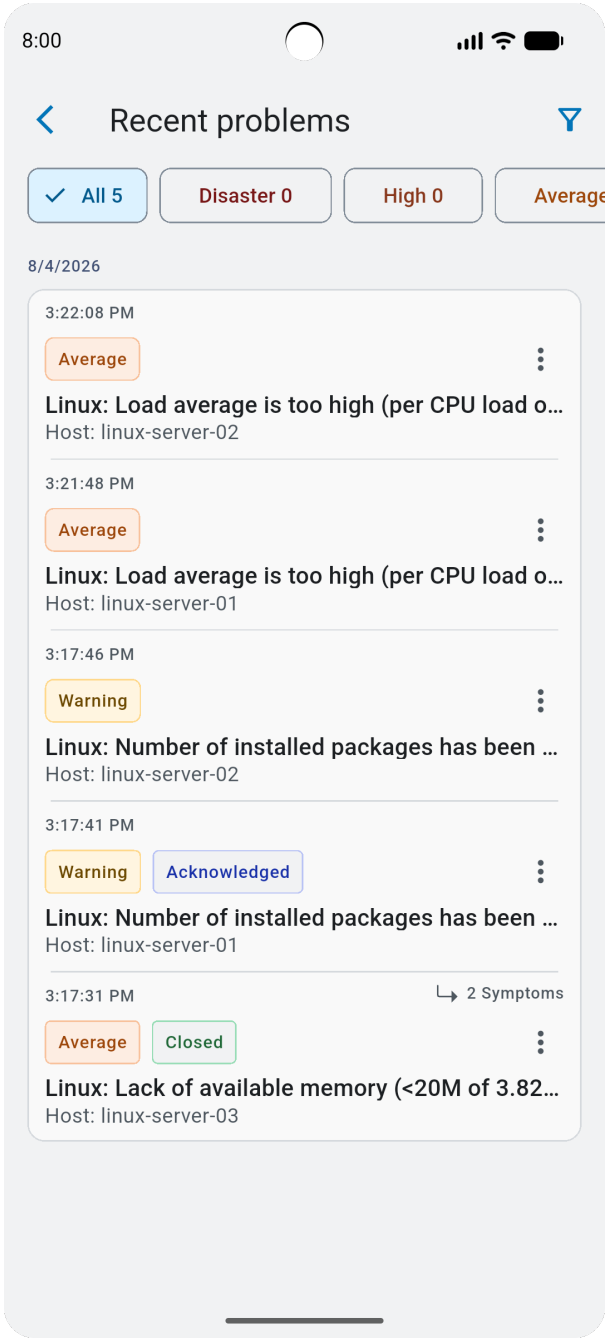
点击筛选图标或问题严重性，可进一步缩小列表范围。

对于每个问题，您可以查看：

- 开始时间和日期。
- 严重性、名称和主机。
- 是否已确认，或最近是否已关闭（自动解决或手动关闭）。
- 是否存在症状 - 点击可查看症状问题列表。

点击某个问题可打开问题详情屏幕。

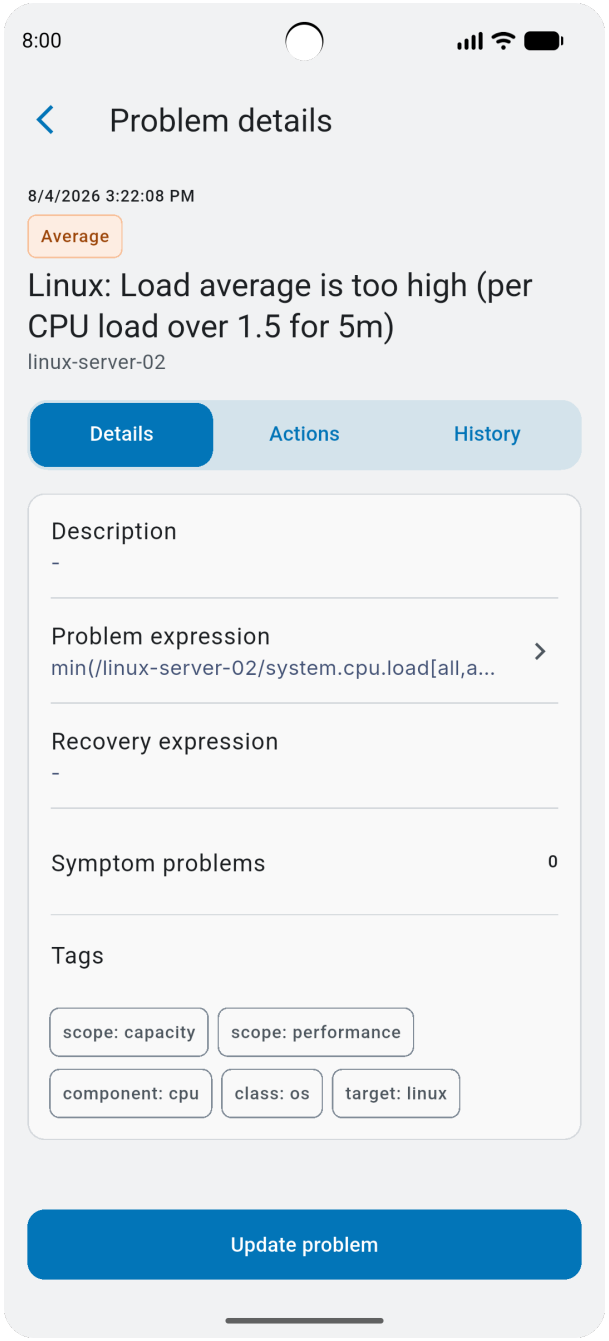
点击三点菜单可快速确认问题，或在您选择的时间内抑制问题。



问题详细信息　问题详细信息屏幕显示问题详细信息、操作和历史记录。

- 详细信息选项卡 - 显示问题描述、表达式、症状问题（点击可查看问题列表）和标签。
- 操作选项卡 - 显示问题时间线：问题何时开始或被确认、是否已发送通知，以及其他操作。
- 历史记录选项卡 - 显示问题持续时间、时长和恢复时间（如果问题仍在持续，则显示 –）。

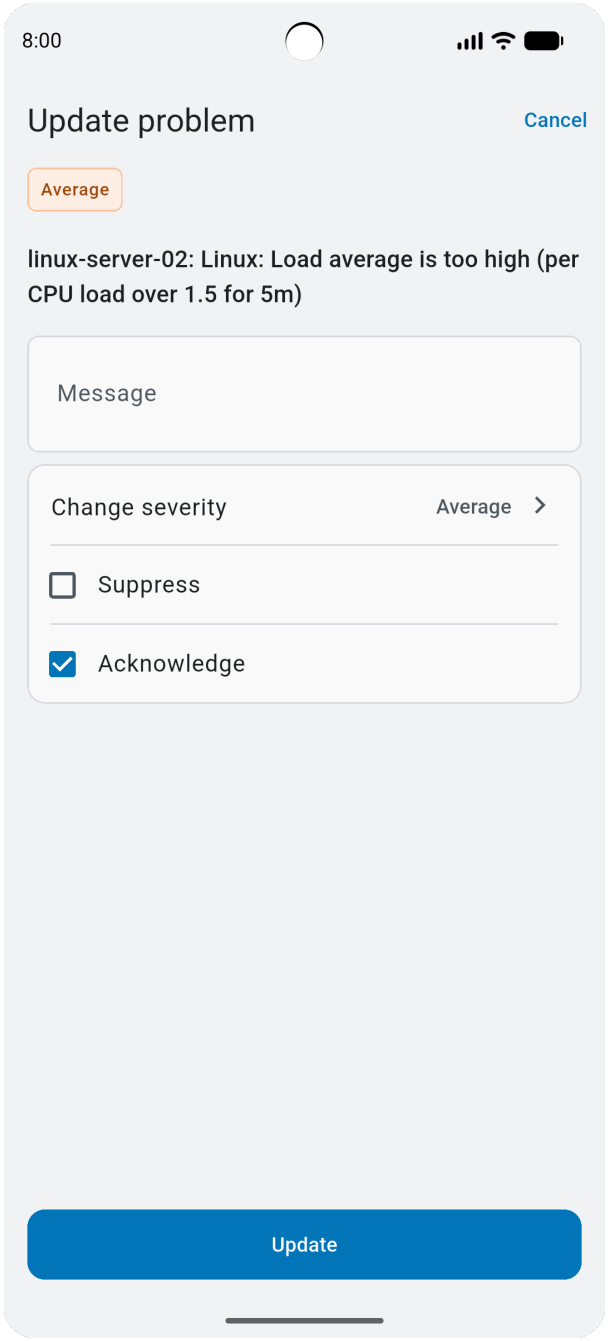
点击更新问题以更新问题。



更新问题　更新问题界面可用于更新问题：

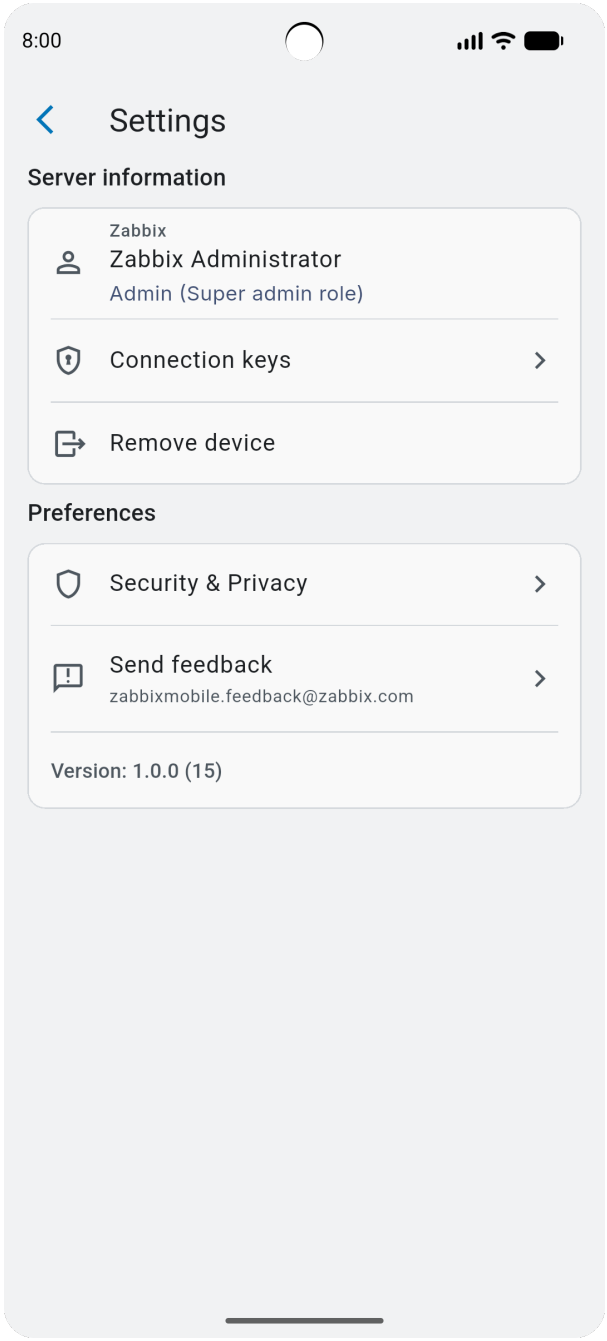
- 添加消息。
- 更改问题严重性。
- 在您选择的时间内抑制问题。
- 确认或取消确认问题。
- 关闭问题。

点击更新以保存更改。



设置　设置屏幕可用于访问应用设置。其中包含：

- Zabbix 服务器名称（如果安装Zabbix 前端时未进行配置，则显示为 *Zabbix*）。
- 添加移动设备的用户的名称、类型和角色。
- 连接密钥 - 打开以查看和复制服务器 ID、设备 ID 及其他详细信息。
- 移除设备 - 点击以从 Zabbix 中移除移动设备。
- 安全与隐私 - 打开以启用应用锁并管理诊断数据共享。
- 发送反馈 - 打开以向 Zabbix 发送反馈。
- 应用版本。



安全的移动访问

本页面介绍了确保 Zabbix 移动应用和 Zabbix bridge adapter 安全的最佳实践。

保护移动设备　为防止他人打开您移动设备上的应用，请在应用的设置 中启用应用锁。

如果设备丢失或遭到入侵，请在 Zabbix 前端中将其移除。这会撤销并移除设备的访问凭据，因此该设备将无法再访问 Zabbix。根据您的用户角色权限，您可以在以下部分移除设备：

- 用户设置 > 设备 - 任何用户都可以移除自己的设备（需要 管理自己的设备权限）。
- 用户 > 设备 - 超级管理员可以移除任何用户的设备（需要 管理用户设备权限）。

您还可以限制移动设备可以访问的 Zabbix 部分。为此，请为用户分配一个仅授予应用所需权限的用户角色。所需的最小权限如下：

部分	配置
访问 *UI* 元素	清除所有复选框。
访问服务	将 对服务的读写访问权限和 对服务的只读访问权限设置为 `None`。
访问模块	清除所有复选框。

部分	配置
访问 *API*	清除 已启用复选框。
访问操作	清除所有复选框。 若要允许用户在应用中更新问题，还需勾选其所需的操作： - 添加问题评论 - 更改严重性 - 确认问题 - 抑制问题 - 关闭问题 - 创建和编辑自己的媒介 请注意，用户可以在应用中看到的问题还取决于其对主机的访问权限。
设备	勾选 已启用复选框。 若要允许用户添加和移除自己的移动设备，还需勾选 管理自己的设备复选框；否则，超级管理员必须在用户 > 设备 中管理其设备。

保护 **Zabbix bridge adapter**　Zabbix bridge adapter 将其数据（状态文件和加密密钥）存储在 Zabbix bridge adapter 配置文件中`DataDir` 参数指定的目录中。确保该目录具有 0700 权限，并且其中存储的所有文件都具有 0600 权限。

Zabbix bridge adapter 将移动设备的请求传递给 Zabbix API。要限制可通过 Zabbix bridge adapter 调用的 API 方法，请使用其`AllowedAPIMethods` 参数。例如，仅允许移动应用所需的最小方法集；如果不希望使用移动应用的某些功能，也可以允许更少的方法（例如，如果移动应用用户不应更新问题，请不要允许使用 `event.acknowledge`）。

开发者中心

开发者中心

在插件、模块和小部件的指导下，开始为 Zabbix 开发自定义扩展。

基于 **Zabbix** 构建

插件　关于如何开发和管理 Zabbix 插件以扩展功能或与外部系统集成的概述。

适用于 **Zabbix** 的 **Python** 库　用于使用官方 `zabbix_utils` 库与 Zabbix API 交互并构建集成的资源。

前端模块　用于构建自定义模块的指南和参考，这些模块可扩展或修改 Zabbix 前端，以适应特定用例。

小组件　对小组件结构和逻辑的细分说明，并提供创建符合您需求的自定义仪表板元素的说明。

创建插件

本页面面向希望为 Zabbix agent 2 开发自己的插件（独立数据采集器）的用户。

自定义插件可在所提供的内置插件和可加载插件之外扩展 Zabbix agent 2 的功能。

每个插件都是一个 *Go* 包，用于定义结构并实现一个或多个接口（*Exporter*、*Configurator*、*Runner*）。有关更多信息，请参阅插件接口和连接图。

本指南将帮助您创建自定义可加载插件。

如需其他指导，还可以参阅以下代码库：

- 示例插件
- 现有可加载插件

您将创建的内容　这是一个分步教程，用于创建一个名为 **MyIP** 的简单可加载插件。
该插件将实现一个单独的监控项键（`myip`），用于返回 Zabbix agent 主机的外部 IP 地址。

第 **1** 步：设置

1. 插件是一个标准的 Go 模块。首先在插件目录中初始化 go.mod 文件，以跟踪插件依赖项：

```
cd path/to/plugins/myip # 切换到您的插件目录
go mod init myip
```

2. 安装必需的依赖项 Zabbix Go SDK（golang.zabbix.com/sdk）：

```
go get golang.zabbix.com/sdk@$LATEST_COMMIT_HASH
```

将 $LATEST_COMMIT_HASH 替换为 golang.zabbix.com/sdk repository 相应发布分支中最新 HEAD 提交的哈希值。例如：

```
go get golang.zabbix.com/sdk@af85407
```

请注意，当前 golang.zabbix.com/sdk 不支持版本控制，但未来可能会有所变化。

可以根据需要使用 go get 安装其他依赖项。

3. 为插件源代码创建一个空的 main.go 文件：

```
touch main.go
```

现在初始设置已完成，插件已准备好进行开发。

步骤 **2**：插件结构　上一步安装的 golang.zabbix.com/sdk 模块提供了插件开发所需的框架，并确保所有插件具有一致的结构。

1. 设置基本执行流程。

首先定义插件的主要执行流程。将以下代码添加到 main.go：

```
package main

func main() {
    err := run()
    if err != nil {
        panic(err)
    }
}

func run() error {
    return nil
}
```

这建立了插件的基本执行流程。稍后，run 函数将包含插件的核心逻辑。

2. 了解插件接口。

Zabbix agent 2 插件应由一个实现 golang.zabbix.com/sdk/plugin 包中接口的结构体表示：

- *Accessor* - 定义所有插件都必须实现的基本方法，例如设置插件名称和处理监控项键超时。
- 以下一个或多个功能插件接口：
  - *Exporter* - 执行轮询并返回一个或多个值、空结果或错误；通常与 *Collector* 接口结合使用。
  - *Collector* - 管理定期数据收集。
  - *Runner* - 定义插件的启动和关闭过程。
  - *Watcher* - 支持独立的指标轮询，绕过 agent 的内部调度程序；适用于基于 trap 或事件驱动的监控。
  - *Configurator* - 定义插件读取和应用其配置设置的方式。

您可以自行实现这些接口，也可以使用 Zabbix Go SDK 提供的默认接口，并根据需要对其进行修改。本教程使用默认实现。

3. 创建插件结构体。

现在，导入 *plugin* 包，并创建一个嵌入 plugin.Base 结构体的 myIP 结构体：

```
import "golang.zabbix.com/sdk/plugin"

type myIP struct {
    plugin.Base
}
```

目前，myIP 结构体已满足 Accessor 接口。本教程稍后将添加用于实现某个功能插件接口（即 Exporter）的方法。

第 **3** 步：定义监控项键值　您的插件需要监控项键值来收集数据，并将其提供给 Zabbix 服务器或 proxy。

1. 导入 *errs* 包以进行错误处理：

```
import "golang.zabbix.com/sdk/errs"
```

2. 在 run() 函数中使用 plugin.RegisterMetrics() 函数注册监控项键值：

```
func run() error {
    p := &myIP{}

    // 注册 `myip` 监控项键值。
    err := plugin.RegisterMetrics(
        p,
        "MyIP",                             // 插件名称
        "myip",                             // 监控项键值名称
        "Returns the host's IP address.", // 监控项键值描述
    )
    if err != nil {
        return errs.Wrap(err, "failed to register metrics")
    }

    return nil
}
```

要注册多个监控项键值，请为每个指标重复使用参数 *metric name* 和 *description*。
例如：

```
plugin.RegisterMetrics(&impl, "Myip", "metric.one", "Metric one description.", "metric.two", "Metric two d
```

第 **4** 步：设置处理程序　处理程序用于促进 agent 与插件之间的通信。

1. 导入 *container* 包：

```
import "golang.zabbix.com/sdk/plugin/container"
```

2. 在 run() 函数中添加用于创建和设置处理程序的代码：

```
func run() error {
    p := &myIP{}

    // 注册 `myip` 监控项键。
    err := plugin.RegisterMetrics(
        p,
        "MyIP",                             // 插件名称
        "myip",                             // 监控项键名称
        " 返回主机的 IP 地址。",             // 监控项键描述
    )
    if err != nil {
        return errs.Wrap(err, "failed to register metrics")
    }

    // 创建新的处理程序。
    h, err := container.NewHandler("MyIP") // 插件名称
    if err != nil {
        return errs.Wrap(err, "failed to create new handler")
    }

    // 设置日志记录，将日志从插件转发到 agent。
    // 可通过 p.Logger.Infof、p.Logger.Debugf 等方式使用。
    p.Logger = h

    // 启动插件执行。
    // 在收到来自 agent 的终止请求之前会一直阻塞。
    err = h.Execute()
    if err != nil {
        return errs.Wrap(err, "failed to execute plugin handler")
```

```
    }

    return nil
}
```

第 **5** 步：实现数据采集 数据采集通过 Exporter 接口完成，该接口描述了 Export 方法：

```
func Export(
  key string,              // 要采集的监控项键值。
  params []string,         // 传递给监控项键值的参数（`myip[arg1, arg2]`）。
  context ContextProvider // 有关监控项键值数据采集的元数据。
) (any, error)
```

1. 导入 HTTP 请求和响应读取所需的软件包：

```
import (
    "io"
    "net/http"
)
```

2. 为 myIP 结构体实现 Export 方法：

```
func (p *myIP) Export(
    key string, params []string, context plugin.ContextProvider,
) (any, error) {
    // 插件可以根据 `key` 参数使用不同的数据采集逻辑。
    // 此实现仅验证所提供的 `key` 是否受支持。
    if key != "myip" {
        return nil, errs.Errorf("unknown item key %q", key)
    }

    // 日志将被转发到 agent 2 日志中。
    p.Infof(
        "received request to handle %q key with %d parameters",
        key,
        len(params),
    )

    // 采集数据并返回。

    resp, err := http.Get("https://api.ipify.org")
    if err != nil {
        return nil, errs.Wrap(err, "failed to get IP address")
    }

    defer resp.Body.Close()

    body, err := io.ReadAll(resp.Body)
    if err != nil {
        return nil, errs.Wrap(err, "failed to read response body")
    }

    return string(body), nil
}
```

第 **6** 步：构建并配置插件

1. 要构建插件，请运行：

```
go mod tidy
go build
```

这应该会在当前目录中创建一个可执行文件 myip。

2. 配置 Zabbix agent 2 使用该插件：

```
echo "Plugins.MyIP.System.Path=$PATH_TO_THE_MYIP_PLUGIN_EXECUTABLE" > /etc/zabbix/zabbix_agent2.d/plugins.
```

将 $PATH_TO_THE_MYIP_PLUGIN_EXECUTABLE 替换为第 5 步中创建的 myip 的路径。

配置参数名中的插件名称（本教程中的 *MyIP*）必须与 *plugin.RegisterMetrics()* 函数中定义的插件名称一致。

3. 要测试插件及其 myip 监控项，请运行：

```
zabbix_agent2 -c /etc/zabbix/zabbix_agent2.conf -t myip
```

输出应包含主机的外部 IP 地址，并且看起来类似于：

```
myip                                            [s|192.0.2.1]
```

至此，您已经为 Zabbix agent 2 创建了一个简单的可加载插件。恭喜！

```go
package main

import (
    "io"
    "net/http"

    "golang.zabbix.com/sdk/errs"
    "golang.zabbix.com/sdk/plugin"
    "golang.zabbix.com/sdk/plugin/container"
)

var _ plugin.Exporter = (*myIP)(nil)

type myIP struct {
    plugin.Base
}

func main() {
    err := run()
    if err != nil {
        panic(err)
    }
}

func run() error {
    p := &myIP{}

    // 注册 `myip` 监控项键。
    err := plugin.RegisterMetrics(
        p,
        "MyIP",                             // 插件名称
        "myip",                             // 监控项键名称
        " 返回主机的 IP 地址。",              // 监控项键描述
    )
    if err != nil {
        return errs.Wrap(err, "failed to register metrics")
    }

    // 创建一个新的处理程序。
    h, err := container.NewHandler("MyIP") // 插件名称
    if err != nil {
        return errs.Wrap(err, "failed to create new handler")
    }

    // 设置日志记录，将插件中的日志转发到 agent。
    // 可通过 p.Logger.Infof、p.Logger.Debugf 等使用。
    p.Logger = h
```

```
    // 启动插件执行。
    // 在收到来自 agent 的终止请求之前会一直阻塞。
    err = h.Execute()
    if err != nil {
        return errs.Wrap(err, "failed to execute plugin handler")
    }

    return nil
}

func (p *myIP) Export(
    key string, params []string, context plugin.ContextProvider,
) (any, error) {
    // 插件可以根据 `key` 参数使用不同的数据采集逻辑。
    // 此实现仅验证提供的 `key` 是否受支持。
    if key != "myip" {
        return nil, errs.Errorf("unknown item key %q", key)
    }

    // 日志将被转发到 agent 2 日志中。
    p.Infof(
        "received request to handle %q key with %d parameters",
        key,
        len(params),
    )

    // 采集数据并返回。

    resp, err := http.Get("https://api.ipify.org")
    if err != nil {
        return nil, errs.Wrap(err, "failed to get IP address")
    }

    defer resp.Body.Close()

    body, err := io.ReadAll(resp.Body)
    if err != nil {
        return nil, errs.Wrap(err, "failed to read response body")
    }

    return string(body), nil
}
```

完整源代码

插件接口

本节介绍可用的插件接口。

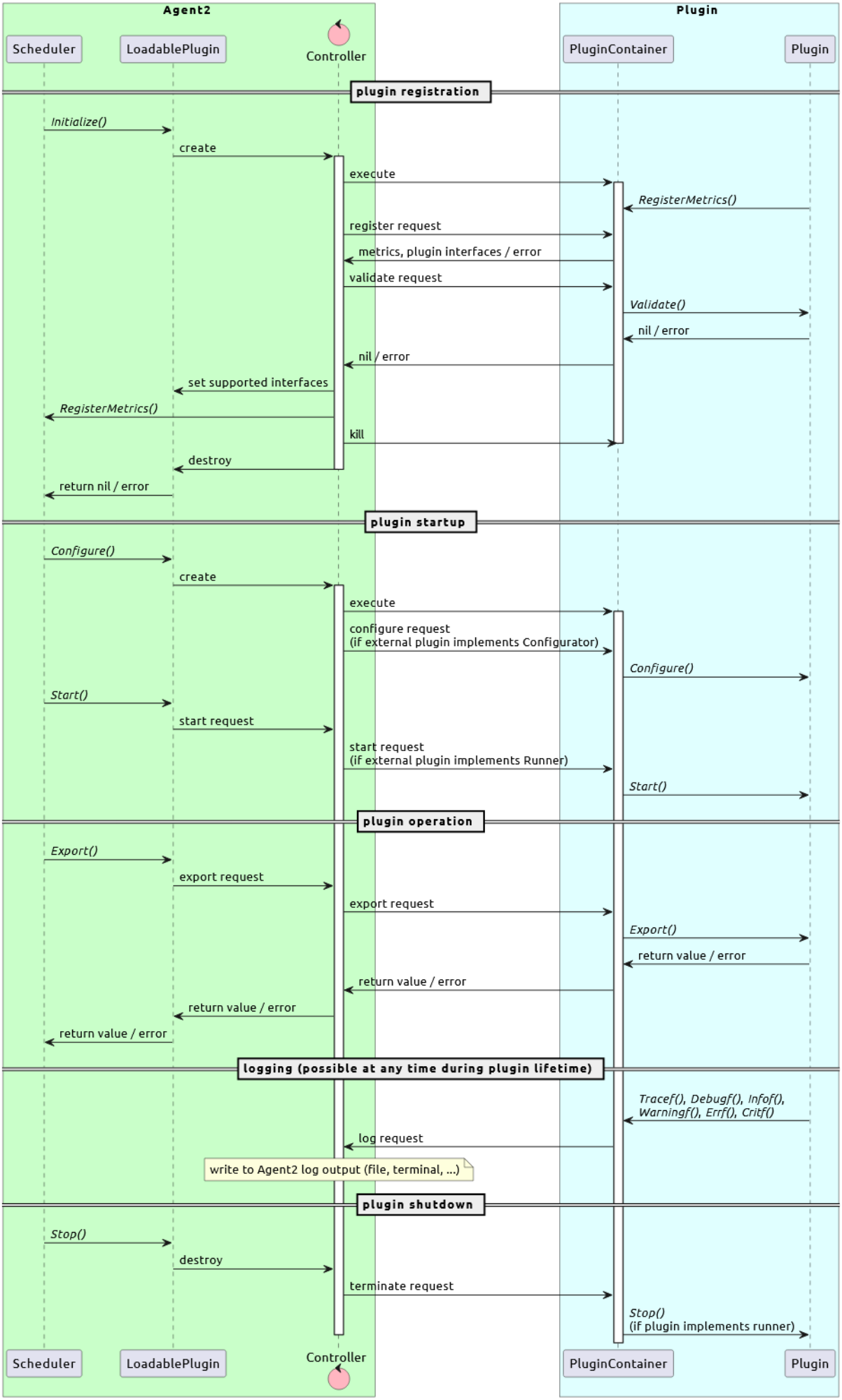
plugin.Exporter *Exporter* 是最简单的接口，它执行轮询并返回一个值（值）、不返回任何值或错误。它接受预解析的项目键、参数和上下文。对所有其他插件接口的访问都是独占的，如果插件已在执行任务，则无法调用任何方法。此外，每个插件最多可同时调用 100 次 *Export()*，可根据每个插件的要求减少。

plugin.Configurator *Configurator* 接口提供来自 Zabbix agent 2 配置文件的插件配置参数。

plugin.Runner *Runner* 接口提供了在插件启动（激活）时执行初始化以及在插件停止（停用）时取消初始化的方法。例如，插件可以通过实现 Runner 接口来启动/停止某些后台 *goroutine*。

连接示意图 Zabbix agent 2 在 Linux 上使用 UNIX 套接字、在 Windows 上使用命名管道与插件进行双向连接。

下方的连接示意图展示了 Zabbix agent 2 与可加载插件之间的通信过程以及指标采集过程。


Agent2
Plugin
Scheduler
LoadablePlugin
Controller
PluginContainer
Plugin
plugin registration
Initialize()
create
execute
RegisterMetrics()
register request
metrics, plugin interfaces / error
validate request
Validate()
nil / error
nil / error
set supported interfaces
RegisterMetrics()
kill
destroy
return nil / error
plugin startup
Configure()
create
execute
configure request
(if external plugin implements Configurator)
Configure()
Start()
start request
start request
(if external plugin implements Runner)
Start()
plugin operation
Export()
export request
export request
Export()
return value / error
return value / error
return value / error
return value / error
logging (possible at any time during plugin lifetime)
Tracef(), Debugf(), Infof(),
Warningf(), Errf(), Critf()
log request
write to Agent2 log output (file, terminal, ...)
plugin shutdown
Stop()
destroy
terminate request
Stop()
(if plugin implements runner)
Scheduler
LoadablePlugin
Controller
PluginContainer
Plugin

Zabbix 的 Python 库

zabbix_utils 是 Zabbix 的官方 Python 库。

使用 zabbix_utils，您可以：

- 使用 Zabbix API 管理 Zabbix 对象（创建主机、更新监控项、获取事件等）。
- 从 Zabbix agent 收集数据（类似于 Zabbix get）。
- 将数据发送到 Zabbix 服务器或 proxy（类似于 Zabbix sender）。

该库同时支持同步和异步脚本执行，因此既可处理简单工作流，也可处理大规模并行操作，例如批量数据采集、批量导出、告警处理以及高效的系统扩展。

该库兼容受支持的 Zabbix 服务器版本和现代 Python 版本。

要开始使用，请先安装该库，然后按照快速入门指南进行操作。

安装

可以使用以下任一方法安装 zabbix_utils。

要求：

- Zabbix 6.0 或更高版本（已在 6.0、7.0、7.2、7.4 上测试）
- Python 3.8 或更高版本（已在 3.8–3.13 上测试）
- aiohttp 异步 HTTP 框架（仅异步模式需要）

Python 包索引 (PyPI)

这是大多数环境中最常用的方法：

```
pip install zabbix_utils
pip install zabbix_utils[async] # 仅用于 async 支持
```

Zabbix 软件仓库

如果您更倾向于通过系统的软件包管理器管理依赖项，请使用此方法。

下载适用于您的 Linux 发行版的 Zabbix 软件包，然后运行以下命令。

在 RHEL 及其衍生版上：

```
dnf install python3-zabbix-utils
dnf install epel-release # 仅用于异步支持
dnf install python3-aiohttp # 仅用于异步支持
```

在 Debian/Ubuntu 及其衍生版上：

```
apt install python3-zabbix-utils
apt install python3-aiohttp # 仅用于异步支持
```

源码（GitHub）

如果您希望使用最新的开发版本，请使用此方法：

```
git clone https://github.com/zabbix/python-zabbix-utils
cd python-zabbix-utils/
pip install -r requirements.txt # 仅用于异步支持
python3 setup.py install
```

要安装异步模式所需的依赖项，您也可以使用上述方法之一。

快速入门指南

在您安装 zabbix_utils 后，您可以在脚本中使用它。

本快速入门指南将向您展示如何：

- 从 Zabbix agent 获取主机名。
- 使用 Zabbix API 创建主机和 trapper 监控项。
- 向该监控项发送一个值。

本指南将逐步介绍脚本，并在引入各个部分时进行说明。完整脚本 位于页面末尾。

本指南还假设您的 Zabbix 服务器、agent 和 API 都在本地运行。

从 Zabbix agent 获取主机名

首先，获取运行 Zabbix agent 的系统的主机名。创建主机时，你将需要使用这个主机名通过 Zabbix API 进行创建。

1. 从 zabbix_utils 导入 `Getter` 类。该类的工作方式类似于Zabbix get，可让你向 Zabbix agent 请求数据。

2. 创建一个 `Getter` 实例，连接到位于 `127.0.0.1`、端口为 `10050` 的本地 Zabbix agent。

3. 对 `Getter` 实例调用 `get()` 方法，从 Zabbix agent 请求主机名。

- `get()` 方法将监控项键值作为参数，并向 Zabbix agent 发送该监控项的请求。
- `get()` 方法返回一个对象，主机名存储在该对象的 `value` 属性中。
- `system.hostname` 是 Zabbix agent 的内置键值，用于返回主机名称。

```
from zabbix_utils import Getter

agent = Getter(host='127.0.0.1', port=10050)
hostname = agent.get('system.hostname').value
```

连接并登录到 Zabbix API

接下来，连接到 Zabbix API。
这样，您的脚本就可以管理 Zabbix 对象，例如主机和监控项。

1. 从 zabbix_utils 导入 `ZabbixAPI` 类。

2. 创建一个 `ZabbixAPI` 实例，并提供您的 Zabbix Web 界面的 URL。

3. 在 `ZabbixAPI` 实例上调用 `login()` 方法，并提供您的用户名和密码。
该账户必须具有访问 Zabbix API 的权限。

4. 在 API 操作完成后，调用 `logout()` 方法以关闭会话。

```
from zabbix_utils import ZabbixAPI

api = ZabbixAPI(url='127.0.0.1/zabbix')
api.login(user='Admin', password='zabbix')

### API operations go here

api.logout()
```

在 Zabbix 中创建主机

现在，您已经从 Zabbix agent 获取了主机名，并且已连接到 Zabbix API，接下来可以在 Zabbix 中创建一个新的主机。

1. 在 `ZabbixAPI` 实例上调用`host.create()` API 方法，并提供主机详细信息：

- `host` - 设置为 `hostname` 变量，其中包含您从 Zabbix agent 获取的主机名。
- `interfaces` - 包含运行在该主机上的 Zabbix agent 的连接详细信息。
- `groups` - 至少包含一个该主机应所属的主机组。

2. 由于 Zabbix API 会返回新创建主机的 ID，请将此 ID 保存到 `host_id` 变量中，以便后续使用。

3. 打印一条消息以确认主机已创建。

```
api.host.create(
    host=hostname,
    interfaces=[{
        'type': 1,
        'main': 1,
        'useip': 1,
        'ip': '127.0.0.1',
        'dns': '',
        'port': '10050',
    }],
```

```
    groups=[{'groupid': '2'}]
)

host_id = host['hostids'][0]

print(f"Host '{hostname}' created with ID {host_id}")
```

在 Zabbix 中创建新的监控项

创建主机后，您可以向其中添加一个监控项。

1. 为要创建的监控项定义一个唯一的键。

2. 在 ZabbixAPI 实例上调用item.create() 方法，并提供监控项详细信息：

- hostid - 设置为 host_id 变量，其中包含您刚刚创建的主机 ID。
- name - 监控项名称。
- key_ - 设置为您刚刚定义的 item_key 变量。
- type - 设置为 2（trapper 监控项），这是接收后续步骤中由脚本发送的值所必需的。
- value_type - 设置为 3（无符号数值），即此监控项存储的数据类型。

3. 由于 Zabbix API 会返回新创建监控项的 ID，请将此 ID 保存到 item_id 变量中，以便后续使用。

4. 打印一条消息以确认监控项已创建。

```
item_key = 'app.myservice.heartbeat'

item = api.item.create(
    hostid=host_id,
    name='App heartbeat',
    key_=item_key,
    type=2,
    value_type=3,
)

item_id = item['itemids'][0]

print(f"Item '{item_key}' created with ID {item_id}")
```

向主机发送值

现在您已经创建了主机和监控项，可以向其发送数据了。

1. 导入 time 类，并在发送数据前等待几秒钟。
这样可以确保 Zabbix 已完全处理您新建的主机和监控项。

2. 从 zabbix_utils 导入 Sender 类。
该类的工作方式类似于Zabbix sender，可让您的脚本向 Zabbix 发送数据。

3. 创建一个 Sender 实例，连接到本地 Zabbix 服务器 127.0.0.1 的 10051 端口。

4. 在 Sender 实例上调用 send_value() 方法，向主机的监控项发送一个值，并提供以下详细信息：

- hostname - 设置为 hostname 变量，其中包含您从 Zabbix agent 获取的主机名。
- item_key - 设置为您之前在脚本中刚定义的 item_key 变量。
- 1 - 要发送的值。

4. 打印一条消息以确认该值已发送。

```
import time
time.sleep(10)

from zabbix_utils import Sender

sender = Sender(server='127.0.0.1', port=10051)

response = sender.send_value(hostname, item_key, 1)

print(f"Sender response: {response}")
```

此代码成功运行后，您应会看到一条成功消息，表明 Zabbix 已接收到您发送的值：

```
Sender response: {"processed": 1, "failed": 0, "total": 1, "time": "0.000151", "chunk": 1}
```

现在，您还可以查看 Zabbix Web 界面（监测 > 最新数据）以查看该值。

完整脚本

下面是整合了所有步骤的完整脚本：获取主机名、使用 Zabbix API 创建主机和监控项，以及向 Zabbix 发送数据。

```
import time
from zabbix_utils import Getter, ZabbixAPI, Sender

### 从 Zabbix agent 获取主机名
agent = Getter(host='127.0.0.1', port=10050)
hostname = agent.get('system.hostname').value

### 连接并登录到 Zabbix API
api = ZabbixAPI(url='127.0.0.1/zabbix')
api.login(user='Admin', password='zabbix')

### 在 Zabbix 中创建主机
host = api.host.create(
    host=hostname,
    interfaces=[{
        'type': 1,
        'main': 1,
        'useip': 1,
        'ip': '127.0.0.1',
        'dns': '',
        'port': '10050',
    }],
    groups=[{'groupid': '2'}]
)

host_id = host['hostids'][0]

print(f" 主机 '{hostname}' 已创建，ID 为 {host_id}")

### 在 Zabbix 中创建监控项
item_key = 'app.myservice.heartbeat'

item = api.item.create(
    hostid=host_id,
    name='App heartbeat',
    key_=item_key,
    type=2,
    value_type=3
)

item_id = item['itemids'][0]

print(f" 监控项 '{item_key}' 已创建，ID 为 {item_id}")

### 从 API 登出
api.logout()

### 等待 Zabbix 处理新主机
time.sleep(10)

### 向主机发送值
sender = Sender(server='127.0.0.1', port=10051)
response = sender.send_value(hostname, item_key, 1)

print(f"Sender 响应：{response}")
```

使用 **Zabbix API**

zabbix_utils 让您能够使用Zabbix API 来管理 Zabbix 对象，包括创建主机、更新监控项、检索事件等。

API 请求可以以同步或异步模式发出：

- 在同步模式下，您的 Python 脚本会发送请求，并在继续执行之前等待响应，这适用于简单、顺序且可预测的操作。
- 在异步模式下，脚本发送请求时不会等待每个响应，从而允许其他操作并行进行；对于响应较慢的请求或大批量数据，这种方式效率更高。

本页中的示例重点介绍同步模式，不过异步模式遵循类似的模式。更多示例可在 zabbix_utils GitHub 仓库中找到。

导入

要使用 zabbix_utils 通过 Zabbix API 进行操作，请在您的脚本中导入 ZabbixAPI 类：

```
from zabbix_utils import ZabbixAPI
```

> **Note:**
> 如果您已经在使用社区 Python 库，通常可以将该导入替换为来自 zabbix_utils 的 ZabbixAPI。

登录

在发出 API 请求之前，您必须创建一个 ZabbixAPI 实例并登录 Zabbix API。

请选择最适合您管理凭据方式的方法。您可以使用用户名和密码，或使用API 令牌。

初始化期间

创建 ZabbixAPI 实例时，您可以一次性提供 Zabbix Web 界面 URL 和您的凭据：

```
### 用户名和密码：
api = ZabbixAPI(url="127.0.0.1/zabbix", user="Admin", password="zabbix")

### API 令牌：
api = ZabbixAPI(url="127.0.0.1/zabbix", token="your_api_token")
```

如果您希望将连接参数分组，也可以将它们作为 Python 字典传递，并使用 Python 的关键字参数语法将其解包：

```
ZABBIX_AUTH = {
    "url": "127.0.0.1/zabbix",
    "user": "Admin",
    "password": "zabbix"
}

api = ZabbixAPI(**ZABBIX_AUTH)
```

使用 `login()`

您可以先仅使用 Zabbix Web 界面 URL 创建一个 ZabbixAPI 实例，然后再使用您的凭据调用 `login()` 方法：

```
### 用户名和密码：
api = ZabbixAPI(url="127.0.0.1/zabbix")
api.login(user="Admin", password="zabbix")

### API 令牌：
api = ZabbixAPI(url="127.0.0.1/zabbix")
api.login(token="your_api_token")
```

使用环境变量

您可以先将凭据存储为环境变量，例如通过终端执行：

```
### 用户名和密码：
export ZABBIX_USER="Admin"
export ZABBIX_PASSWORD="zabbix"

### Token：
export ZABBIX_TOKEN="your_api_token"
```

然后，如果您从同一个终端会话启动脚本，脚本中的 ZabbixAPI 实例只需要 Zabbix Web 界面的 URL：

```
api = ZabbixAPI(url="127.0.0.1/zabbix")
```

您还可以将 URL 存储在环境变量中：

```
### 用户名和密码：
export ZABBIX_USER="Admin"
export ZABBIX_PASSWORD="zabbix"
export ZABBIX_URL="https://127.0.0.1/zabbix"

### Token：
export ZABBIX_TOKEN="your_api_token"
export ZABBIX_URL="https://127.0.0.1/zabbix"
```

当 URL 和凭据都设置为环境变量后，脚本中的 ZabbixAPI 实例将完全不需要任何参数：

```
api = ZabbixAPI()
```

注销

如果您使用用户名和密码登录，请在完成 API 操作后调用 logout() 方法：

```
from zabbix_utils import ZabbixAPI

api = ZabbixAPI(url="127.0.0.1/zabbix")
api.login(user="Admin", password="zabbix")

### API actions go here

api.logout()
```

> **Note:**
> 使用 API 令牌时，无需调用 logout() 方法。

API 请求

登录后，您可以通过调用 Zabbix API 方法参考中描述的方法发起任何 API 请求。

API 方法使用以下格式调用：

```
api_instance.zabbix_object.method(parameters)
```

> **Note:**
> 某些 Zabbix API 方法或对象名称使用了 Python 中的保留关键字（例如 import）。为避免 Python 报错，在 Python 中使用时，请在方法或对象名称后添加下划线（例如 api.configuration.import_）。

例如，要使用host.get 查看主机列表：

```
### 1. 从 zabbix_utils 导入 ZabbixAPI：
from zabbix_utils import ZabbixAPI

### 2. 创建 ZabbixAPI 实例并登录：
api = ZabbixAPI(url="127.0.0.1/zabbix")
api.login(user="Admin", password="zabbix")

### 3. 获取与过滤条件匹配的主机：
hosts = api.host.get(
    filter={
        "host": [
            "Zabbix server",
            "Linux server"
        ]
    }
)

### 4. 遍历返回的主机并打印每个主机的详细信息：
```

```
for host in hosts:
    print(host)

### 5. 从 API 注销以关闭会话：
api.logout()
```

异步模式

异步模式允许您的脚本发送 API 请求时无需等待每个请求完成。当脚本需要执行大量 API 请求，或者某些请求耗时较长时，这可以让脚本更高效。

使用异步模式时，与同步模式相比有以下重要区别：

- 导入 Python 的 asyncio 模块（您必须先安装所需的依赖）。
- 导入 AsyncZabbixAPI，而不是 ZabbixAPI。
- 在 async 函数中编写代码。
- 调用 API 方法时使用 await，包括 login() 和 logout()。

Attention:
创建 AsyncZabbixAPI 实例时，您不能在初始化期间登录。

例如，要在异步模式下使用host.get 查看主机列表：

```
### 1. 为异步模式导入 asyncio，并从 zabbix_utils 导入 AsyncZabbixAPI：
import asyncio
from zabbix_utils import AsyncZabbixAPI

### 2. 定义主 async 函数，所有 API 操作都将在其中运行：
async def main():

    # 3. 创建 AsyncZabbixAPI 实例并登录（必须使用 await）：
    api = AsyncZabbixAPI(url="127.0.0.1")
    await api.login(user="User", password="zabbix")

    # 4. 获取与过滤条件匹配的主机（必须使用 await）：
    hosts = await api.host.get(
        filter={
            "host": [
                "Zabbix server",
                "Linux server"
            ]
        }
    )

    # 5. 遍历返回的主机并打印每个主机的详细信息：
    for host in hosts:
        print(host)

    # 6. 从 API 注销以关闭会话（必须使用 await）：
    await api.logout()

### 7. 使用 asyncio 的事件循环运行 async main() 函数：
asyncio.run(main())
```

示例

以下示例展示了使用 zabbix_utils 库执行常见 Zabbix API 任务的方法。

更多示例可在 zabbix_utils GitHub 仓库的 examples 目录中找到。

从 **Zabbix agent** 收集数据

概述

zabbix_utils 允许您从 Zabbix agent 收集数据（类似于Zabbix get）。

可以使用同步或异步模式收集数据：

- 在同步模式下，您的 Python 脚本会先请求并等待数据返回，然后再继续执行，这适用于简单、顺序性强且可预测的操作。
- 在异步模式下，脚本发出数据请求后不会等待每个响应返回，从而允许其他操作并行进行；对于响应较慢的请求或大批量数据，这种方式效率更高。

本页中的示例主要侧重于同步模式，不过异步模式也遵循类似的模式。更多示例可在 GitHub 仓库中的 zabbix_utils 中找到。

导入

要使用 zabbix_utils 收集监控项值，请在您的 Python 脚本中导入 Getter 类：

```
from zabbix_utils import Getter
```

请求数据

要请求监控项值：

1. 创建一个 Getter 实例，指定 Zabbix agent 的 IP 地址和端口。
2. 在 Getter 实例上调用 get() 方法，指定要检索的监控项键值。

例如，要请求 system.uname 监控项的数据：

```
agent = Getter(host='192.0.2.1', port=10050)
response = agent.get('system.uname')
```

使用非默认 IP

如果运行脚本的服务器有多个 IP 地址，您可以为 Getter 指定一个 source_ip，以便在连接到 Zabbix agent 时使用：

```
agent = Getter(
    host='192.0.2.1',
    port=10050,
    source_ip='10.10.7.1'
)
```

使用超时

您可以为 Getter 设置响应 timeout，以控制脚本在放弃之前应等待 Zabbix agent 响应的时间：

```
agent = Getter(
    host='192.0.2.1',
    port=10050,
    timeout=30
)
```

使用加密

Getter 不包含内置加密支持，但你可以通过使用第三方库创建一个包装器来提供它：

```
def psk_wrapper(sock, tls):
    # ...
    # socket 的 TLS PSK 包装器实现
    # ...

agent = Getter(
    host='192.0.2.1',
    port=10050,
    socket_wrapper=psk_wrapper
)
```

响应

来自 Zabbix agent 的响应由库处理，并作为 AgentResponse 对象返回：

```
print(response)
### {
###     "error": null,
###     "raw": "Linux zabbix_server 5.15.0-3.60.5.1.el9uek.x86_64",
###     "value": "Linux zabbix_server 5.15.0-3.60.5.1.el9uek.x86_64"
### }
```

```
print(response.value)
### Linux zabbix_server 5.15.0-3.60.5.1.el9uek.x86_64

print(response.error)
### None
```

异步模式

异步模式允许你的脚本在不等待每个值返回的情况下收集值。当脚本需要收集大量值，或者某些值的收集耗时较长时，这可以让脚本更高效。

使用异步模式时，与同步模式相比有一些重要区别：

- 导入 Python 的 asyncio 模块（你必须先安装所需依赖）。
- 导入 AsyncGetter，而不是 Getter。
- 将代码写在 async 函数中。
- 调用 get() 方法时使用 await。

例如，要使用异步模式收集单个值：

```
### 1. 为异步模式导入 asyncio，并从 zabbix_utils 导入 AsyncGetter：
import asyncio
from zabbix_utils import AsyncGetter

### 2. 定义主 async 函数，所有数据请求都将在其中执行：
async def main():
    agent = AsyncGetter(host='192.0.2.1', port=10050)

    # 3. 从 Zabbix agent 获取 system.uname 值（必须使用 await）：
    response = await agent.get('system.uname')

    # 4. 打印 Zabbix agent 返回的值：
    print(response.value)

### 5. 使用 asyncio 的事件循环运行 async main() 函数：
asyncio.run(main())
```

向 **Zabbix** 服务器或 **proxy** 发送数据

zabbix_utils 允许您将监控项值发送到 Zabbix 服务器或 proxy 上的trapper 监控项（类似于Zabbix sender）。

您可以发送单个值、多个值，甚至以多个 Zabbix 集群为目标发送数据。

数据可以通过同步或异步模式发送：

- 在同步模式下，您的 Python 脚本会发送值，并在继续之前等待响应；这适用于简单、顺序且可预测的操作。
- 在异步模式下，脚本发送值时不会等待每个响应，从而允许其他操作并行进行；对于较慢的请求或大批量数据，这种方式效率更高。

本页中的示例重点介绍同步模式，不过异步模式遵循类似的模式。更多示例可在 zabbix_utils GitHub 仓库中找到。

导入

要使用 zabbix_utils 发送监控项值，请在您的脚本中导入 Sender 类：

```
from zabbix_utils import Sender
```

如需发送多个值，您还可以导入 ItemValue 类：

```
from zabbix_utils import Sender, ItemValue
```

发送单个值

要发送监控项值：

1. 创建一个 Sender 实例，并指定 Zabbix 服务器或 proxy 的 IP 地址和端口。
2. 在 Sender 实例上调用 send_value() 方法，格式如下：

```
sender_instance.send_value('host', 'item.key', 'value', optional_timestamp, optional_nanoseconds)
```

例如，要将 1 发送到主机 Linux server 上的 service.status trapper 监控项：

```
sender = Sender(server='127.0.0.1', port=10051)
response = sender.send_value('Linux server', 'service.status', 1)
```

使用非默认 IP

如果运行脚本的服务器有多个 IP 地址，您可以为 Sender 指定一个 source_ip，以便在向 Zabbix 服务器或 proxy 发送值时使用：

```
sender = Sender(
    server='127.0.0.1',
    port=10051,
    source_ip='10.10.7.1'
)
```

使用超时

您可以为 Sender 设置响应 timeout，以控制脚本在放弃之前应等待 Zabbix 服务器或 proxy 响应的时长：

```
sender = Sender(
    server='127.0.0.1',
    port=10051,
    timeout=30
)
```

使用 agent 配置文件

您可以让 zabbix_utils 从本地 Zabbix agent 或agent 2 配置文件中读取Server 或ServerActive 参数。在这种情况下，创建 Sender 实例时无需指定连接参数：

```
sender = Sender(
    use_config=True,
    config_path='/etc/zabbix/zabbix_agent2.conf'
)
```

Attention:
如果 ServerActive 包含一个或多个具有多个服务器实例的 Zabbix 集群，Sender 会将数据发送到每个集群中第一个可用的服务器。如果未设置 ServerActive，则使用 Server 中的地址和默认端口（10051）。

使用加密

Sender 不包含内置加密支持，但您可以通过使用第三方库创建一个包装器来提供该功能：

```
def psk_wrapper(sock, tls):
    # ...
    # 套接字的 TLS PSK 包装器实现
    # ...

sender = Sender(
    server='127.0.0.1',
    port=10051,
    socket_wrapper=psk_wrapper
)
```

单个值的响应

由 Zabbix 服务器或 proxy 返回的响应会由库处理，并作为 TrapperResponse 对象返回：

```
print(response)
### {"processed": 1, "failed": 0, "total": 1, "time": "0.000123", "chunk": 1}

print(response.processed)
### 1

print(response.failed)
### 0

print(response.total)
### 1
```

异步模式

异步模式允许您的 Python 脚本在不等待 Zabbix 服务器或 proxy 响应的情况下发送值。
当脚本需要发送大量值，或者某些值发送耗时较长时，这可以让脚本更高效。

与同步模式相比，使用异步模式时有几个重要区别：

- 导入 Python 的 asyncio 模块（您必须先安装所需的依赖）。
- 导入 AsyncSender，而不是 Sender。
- 将代码编写在 async 函数中。
- 调用 send_value() 方法时使用 await。

例如，使用异步模式发送单个值：

```
### 1. 为异步模式导入 asyncio，并从 zabbix_utils 导入 AsyncSender：
import asyncio
from zabbix_utils import AsyncSender

### 2. 定义主 async 函数，所有数据发送操作（必须 await）都将在其中执行：
async def main():
    sender = AsyncSender(server='127.0.0.1', port=10051)
    response = await sender.send_value('Linux server', 'service.status', 1)

    # 3. 打印 Zabbix 服务器或 proxy 返回的响应：
    print(response)

### 4. 使用 asyncio 的事件循环运行 async main() 函数：
asyncio.run(main())
```

发送多个值

要发送多个值：

1. 准备一个 ItemValue 对象数组，每个对象都使用与send_value() 方法相同的格式。
2. 创建一个 Sender 实例，并指定 Zabbix 服务器或 proxy 的 IP 地址和端口。
3. 在 Sender 实例上调用 send() 方法（而不是 send_value()），并指定包含待发送值的对象数组。

例如，要向不同主机发送五个值：

```
items = [
    ItemValue('server-de', 'service.status', 'up', 1770887205, 100),
    ItemValue('server-fr', 'service.status', 'up', 1770887205, 100),
    ItemValue('server-uk', 'service.status', 'up', 1770887205, 100),
    ItemValue('server-nl', 'service.status', 'up', 1770887205, 100),
    ItemValue('server-pl', 'service.status', 'up', 1770887205, 100),
]

sender = Sender(server='127.0.0.1', port=10051)
response = sender.send(items)
```

使用自定义分块大小

如果您需要发送的值超过 trapper 监控项在单个请求中可接受的数量，可以将它们拆分为多个块。

默认情况下，块大小为 250 个值。
您可以在创建 Sender 实例时设置 chunk_size 参数来更改它。

例如，要将五个值分三块发送（2-2-1），请将 chunk_size 参数设置为 2：

```
items = [
    ItemValue('server-de', 'service.status', 'up'),
    ItemValue('server-fr', 'service.status', 'up'),
    ItemValue('server-uk', 'service.status', 'up'),
    ItemValue('server-nl', 'service.status', 'up'),
    ItemValue('server-pl', 'service.status', 'up'),
]

sender = Sender(server='127.0.0.1', port=10051, chunk_size=2)
response = sender.send(items)
```

向多个 Zabbix 集群发送值

要向多个 Zabbix 集群发送值：

1. 准备一个 Zabbix 集群数组。如果某个集群有多个节点，值将被发送到该集群中第一个可用的节点。
2. 创建一个 Sender，并指定你的 Zabbix 集群数组。
3. 在 Sender 实例上调用 send_value() 方法，格式与send_value() 方法相同。

例如，要将一个值发送到每个集群中第一个可用的节点：

```
zabbix_clusters = [
    ['zabbix.cluster1.node1', 'zabbix.cluster1.node2:10051'],
    ['zabbix.cluster2.node1:10051', 'zabbix.cluster2.node2', 'zabbix.cluster2.node3']
]

sender = Sender(clusters=zabbix_clusters)
response = sender.send_value('Linux server', 'service.status', 1)
```

多个值的响应

默认情况下，Sender 会返回一个聚合结果，表示跨所有主机或集群发送值的结果：

```
print(response)
### {"processed": 2, "failed": 0, "total": 2, "time": "0.000108", "chunk": 2}
```

如果您需要更详细的信息，可以使用 response.details 属性查看每个集群和每个块的结果：

```
print(response)
### {"processed": 2, "failed": 0, "total": 2, "time": "0.000108", "chunk": 2}

if response.failed == 0:
    print(f"Value sent successfully in {response.time}")
else:
    print(response.details)
    # {
    #     127.0.0.1:10051: [
    #         {
    #             "processed": 1,
    #             "failed": 0,
    #             "total": 1,
    #             "time": "0.000051",
    #             "chunk": 1
    #         }
    #     ],
    #     zabbix.example.local:10051: [
    #         {
    #             "processed": 1,
    #             "failed": 0,
    #             "total": 1,
    #             "time": "0.000057",
    #             "chunk": 1
    #         }
    #     ]
    # }

    for node, chunks in response.details.items():
        for resp in chunks:
            print(f"Processed {resp.processed} of {resp.total} at {node.address}:{node.port}")
            # Processed 1 of 1 at 127.0.0.1:10051
            # Processed 1 of 1 at zabbix.example.local:10051
```

调试日志

在使用 Zabbix API、从 Zabbix agent 收集数据，或向 Zabbix 服务器或 proxy 发送数据时，如需排查问题，可以启用调试日志记录。

zabbix_utils 库使用 Python 标准的 logging 模块。

要查看该库输出的详细调试消息：

1. 在脚本中导入 Python 的 logging 模块。
2. 配置日志系统，以可读格式输出消息，并将日志级别设置为 DEBUG。

```
import logging
from zabbix_utils import Getter

logging.basicConfig(
    format=u'[%(asctime)s] %(levelname)s %(message)s',
    level=logging.DEBUG
)

agent = Getter(host='127.0.0.1', port=10050)
resp = agent.get('system.uname')

print(resp.value)
```

启用调试日志记录后，将打印有关请求和响应的详细信息，例如：

```
[2023-10-01 12:00:01,587] DEBUG Content of the packet: b'ZBXD\x01\x0c\x00\x00\x00\x00\x00\x00\x00system.un
[2023-10-01 12:00:01,722] DEBUG Zabbix response header: b'ZBXD\x01C\x00\x00\x00C\x00\x00\x00'
[2023-10-01 12:00:01,723] DEBUG Zabbix response body: Linux test_server 5.15.0-3.60.5.1.el9uek.x86_64
[2023-10-01 12:00:01,724] DEBUG Response from [127.0.0.1:10050]: Linux test_server 5.15.0-3.60.5.1.el9uek.
Linux test_server 5.15.0-3.60.5.1.el9uek.x86_64
```

模块

什么是 **PHP** 前端模块？

- 模块是一个实体，具有唯一的 ID、名称、描述、作者和在其清单文件中定义的其他字段，以及位于 Zabbix 前端安装的 *modules* 目录内的单个目录中的 PHP、Javascript 和其他文件（例如，*zabbix/ui/modules*）。
- 模块应符合简单的规则以保证正确运行。
- 必须由管理员在前端安装（解压）并启用模块。

模块的作用

- 通过自定义前端部分添加新功能；
- 创建自定义仪表板小部件类型（请参阅小部件模块）；
- 覆盖或扩展现有功能。

模块不能实现的功能

- 注册新的 API 方法或修改现有的方法。

模块的工作原理

- 在执行操作代码之前，每个 HTTP 请求都会启动一个启用的模块。
- 模块将注册新操作或重新定义现有操作。
- 模块将添加新的前端部分并删除或重新定义现有部分。
- 如果需要，模块将挂接到 onBeforeAction 和 onTerminate 等前端事件。
- 最终通过运行操作代码（默认代码或模块定义的代码）来执行请求的操作。

下一步　无论您是喜欢边做边学还是先阅读指南，这些页面都包含构建您自己的模块所需的信息和步骤：

- 编写第一个模块的分步教程
- 模块文件结构
- 小部件模块细节
- 可重复使用的模块示例

模块文件结构

与模块相关的所有代码都存储在 Zabbix 前端安装的 **modules** 目录内的单个目录中（例如，*zabbix/ui/modules*）。

```
example_module_directory/ （必需）
manifest.json （必需） 元数据和操作定义。
Module.php 模块初始化和事件处理。
action/ 操作控制器文件。
SomethingView.php
SomethingCreate.php
SomethingDelete.php
data_export/
ExportAsXml.php
ExportAsExcel.php
views/ 视图文件。
example.something.view.php
example.something.delete.php
assets/ 视图中使用的任何其他文件。必须在 manifest.json 中指定。
js/ 视图中使用的 JavaScript 文件。
example.something.view.js.php
css/ 视图中使用的 CSS 文件。
example.something.css
image.png 视图中使用的图像。
example.something.file 视图中使用的任何文件。
```

模块文件树

编写模块　一个示例模块的编写过程包括以下步骤（如有可用内容，点击文件或文件夹名称可查看该步骤的更多详细信息）：

1. 在 **zabbix/ui/modules/** 中为该模块创建一个新目录。
2. 添加manifest.json 文件，其中包含模块元数据。
3. 创建 *views* 文件夹，并定义模块视图。
4. 创建 *actions* 文件夹，并定义模块动作。
5. 创建 Module.php（对于仪表板小部件则为 Widget.php）文件，并定义初始化和事件处理规则。
6. 创建 *assets* 文件夹，用于存放 JavaScript 文件（放在 *assets/js* 中）、CSS 样式（放在 *assets/css* 中）或其他附加文件。
7. 确保在 manifest.json 中指定所需的视图、动作和资源文件。
8. 在 Zabbix 前端中注册该模块并开始使用。

Note:
你在创建 manifest.json 文件后即可注册并启用模块。模块启用后，只需刷新 Zabbix 前端，即可立即预览对模块文件所做的所有更改。

manifest.json

任何模块都需要 manifest.json 文件。
该文件必须位于模块的主目录中（例如，*zabbix/ui/modules/module_name/manifest.json*）。

作为最低要求，manifest.json 应指定以下字段：

```
{
    "manifest_version": 2.0,
    "id": "my_ip_address",
    "name": "My IP Address",
    "namespace": "MyIPAddress",
    "version": "1.0"
}
```

manifest.json 中支持的参数（点击参数名称可查看详细说明）：

Parameter	Description	Required
manifest_version	模块的清单版本。	Yes
id	唯一的模块 ID。	
name	将在管理部分显示的模块名称。	
namespace	模块类的 PHP 命名空间。	
version	模块版本。	
type	模块类型。对于 widget，必须设置为 *widget*	Yes for widgets, otherwise no
widget	widget 配置。仅用于 widget。	
actions	要与模块一起注册的操作。	
assets	要包含的 CSS 样式和 JavaScript 文件。	No
author	模块作者。	
config	自定义模块选项的默认值。	
description	模块描述。	
url	指向模块描述的链接。	

manifest_version

模块的清单版本。当前支持的版本为 **2.0**。

类型：Double

示例：

```
“manifest_version”: 2.0
```

id

模块 ID。必须是唯一的。为避免将来的命名冲突，建议使用模块前缀（作者或公司名称，或任何其他名称）。例如，如果模块是课程示例，模块名称为 “我的模块”，则 ID 将为 “example_my_module”。

类型：字符串

示例：

```
“id”:“example_my_module”
```

name

将在管理部分中显示的模块名称。

类型：字符串

示例：

```
“name”:“我的模块”
```

命名空间

模块类的 PHP 命名空间。

类型：字符串

示例：

```
“ namespace”:“ClockWidget”
```

版本

将在管理部分中显示的模块版本。

类型：字符串

示例：

```
“version”:“1.0”
```

type

模块的类型。对于小部件而言是必需的，并且必须等于 “widget”。

类型：String

默认：“module”

示例：

```
“type”:“widget”
```

操作

要向模块注册的操作。

必须为每个操作定义 *class* 对象键，其他操作键是可选的。

类型：对象

如果 *type* 是 模块，则支持的对象键：

- **write.your.action.name** (对象) - 操作名称，应以小写 [a-z] 书写，用点分隔单词。

支持的键：- **class** (字符串；必需) - 操作类名。- **layout** (字符串) - 操作布局。支持的值：*layout.json*、*layout.htmlpage*（默认）、*null*。- **view** (字符串) - 操作视图。

示例：

```
"actions": {
"module.example.list": {
"class": "ExampleList",
"view": "example.list",
"layout": "layout.htmlpage"
}
}
```

如果 *type* 为 *widget*，则支持的对象键：

- **widget.{id}.view** (对象) - 小部件视图的文件和类名。将 **{id}** 替换为小部件的id 值（例如，*widget.example_clock.view*）。支持以下键：
- **class** (字符串；必需) - 小部件视图模式的操作类名，用于扩展默认的 CControllerDashboardWidgetView 类。类源文件必须位于 *actions* 目录中。
- **view** (字符串) - 小部件视图。必须位于 *views* 目录中。如果视图文件是 *widget.view.php*（默认情况下是预期的），则可以省略此参数。如果使用其他名称，请在此处指定。
- **widget.{id}.edit**（对象）- 小部件配置视图的文件名。将 **{id}** 替换为小部件的id 值（例如，*widget.example_clock.edit*）。支持以下键：
- **class**（字符串；必需）- 小部件配置视图模式的操作类名。类源文件必须位于 *actions* 目录中。
- **view**（字符串）- 小部件配置视图。必须位于 *views* 目录中。如果视图文件是 *widget.edit.php*（默认情况下是预期的），则可以省略此参数。如果使用其他名称，请在此处指定。

示例:

```
"actions": {
"widget.tophosts.view": {
"class": "WidgetView"
},
"widget.tophosts.column.edit": {
"class": "ColumnEdit",
"view": "column.edit",
"layout": "layout.json"
}
}
```

资源

要包含的 CSS 样式和 JavaScript 文件。

类型：对象

支持的对象键：

- **css** (数组) - 要包含的 CSS 文件。这些文件必须位于 *assets/css* 中。
- **js** (数组) - 要包含的 JavaScript 文件。这些文件必须位于 *assets/js* 中。

示例：

```
"assets": {
"css": ["widget.css"],
"js": ["class.widget.js"]
}
```

name

将在管理部分中显示的模块名称。

类型：字符串

示例：

```
“name”:“John Smith”
```

config

模块选项的默认值。对象可以包含任何自定义键。如果指定，这些值将在模块注册期间写入数据库。稍后添加的新变量将在第一次调用时写入。之后，只能在数据库中直接更改变量值。

类型：对象

示例：

```
“config”: {
“ username”:“Admin”,
“ password”:“”,
“auth_url”:“https://example.com/auth”
}
```

description

模块描述。

类型：字符串

示例：

```
“description”:“这是一个时钟小部件。”
```

widget

Widget 配置。如果 *type* 设置为 *widget*，则使用。

类型：对象

支持的对象键：

- **name** (字符串) - 用于 widget 列表和默认标题。如果为空，则将使用模块中的 “name” 参数。
- **size** (对象) - 默认 widget 尺寸。支持键：
- *width* (整数) - 默认 widget 宽度。
- *height* (整数) - 默认 widget 高度。
- **form_class** (字符串) - 带有 widget 字段表单的类。必须位于 *includes* 目录中。如果类是 *WidgetForm.php*（默认情况下是预期的），则可以省略此参数。如果使用不同的名称，请在此处指定。
- **js_class** (字符串) - 用于 widget 视图模式的 JavaScript 类的名称，用于扩展默认的 CWidget 类。该类将与仪表板一起加载。类源文件必须位于 assets/js 目录中。另请参阅：assets。
- **use_time_selector** (布尔值) - 确定小部件是否需要仪表板时间选择器。支持的值：*true*、*false*（默认值）。
- **refresh_rate** (整数) - 小部件刷新率（以秒为单位）（默认值：60）。

示例：

```
"widget": {
"name": "",
"size": {
"width": 12,
"height": 5
},
"form_class": "WidgetForm",
"js_class": "CWidget",
"use_time_selector": false,
"refresh_rate": 60
}
```

url

指向模块描述的链接。对于小组件，点击添加小组件 或编辑小组件 窗口中的帮助图标 时，将打开此链接。如果未指定 **url**，点击帮助图标将打开常规的仪表板小组件 页面。

类型: String

示例:

```
"url": "http://example.com"
```

动作 {#devel-modules-file_structure-actions}

动作是一个负责’ 业务逻辑’ 的模块。一个动作一般由一个触发器和一个动作视图组成。

模块可以做以下事情：

- 调用已经在 Zabbix 前端定义的动作。
- 通过自定义动作覆盖默认动作。
- 定义全新的动作。

如果想通过某些自定义行为去覆盖默认动作，请在模块配置中定义具有相同名称的动作。当调用该动作时，模板动作会被执行，而不是 Zabbix 默认动作。

动作文档应该存储在 *actions* 文件夹中。动作需要被指定在manifest.json。

控制器

操作控制器工作流程：

1. 检查 HTTP 请求中传递的所有参数是否有效：
   - 调用控制器的 *checkInput()* 方法；
   - 使用 CNewValidator.php 中定义的验证规则；
   - 调用 *validateInput()* 方法。
2. 检查用户权限。
3. 根据传递的参数准备数据：如果 *checkInput()* 返回 true，Zabbix 会调用控制器的 *doAction()* 方法。
4. 为视图准备 **$data** 数组。使用 *CControllerResponseData* 和 *setResponse()* 方法将响应存储到 **$data** 数组中。

示例：

```
/**
 * 验证输入参数。
 *
 * @return bool
 */
protected function checkInput(): bool {
    $ret = $this->validateInput([
        'status' => 'in '.implode(',', [HOST_STATUS_MONITORED, HOST_STATUS_NOT_MONITORED])
    ]);

    if (!$ret) {
        $this->setResponse(new CControllerResponseFatal());
    }

    return $ret;
}

/**
 * 检查用户权限。
 *
 * @return bool
 */
protected function checkPermissions() {
    return $this->getUserType() >= USER_TYPE_ZABBIX_ADMIN;
}

/**
 * 执行操作并生成响应对象。
```

```
 */
protected function do Action(): void {
    $data = [
        'hosts_count' => API::Host()->get([
            'countOutput' => true,
            'filter' => [
                'status' => $this->getInput('status')
            ]
        ])
    ];

    $this->setResponse(new CControllerResponseData($data));
}
```

Note:
您可以在 Zabbix 源代码中查看可用控制器类的完整列表。

视图

视图文件从控制器接收数据，然后准备其 HTML 外观。

Note:
为前端模块定义视图是可选的，除非该模块是一个 widget。
 仪表板 widget 至少需要两个视图：一个用于编辑模式，另一个用于查看模式（应存储在 *views* 目录中）。

可以在视图中使用预定义的 Zabbix HTML 类（来自 */zabbix/ui/include/classes/html*），也可以添加新的 HTML 和 CSS 类。新类应存储在模块的 *assets* 文件夹中。

示例：

```
...
(new CColHeader(_('Name')))
```

这将添加一个名为 *Name* 的新列，并将表格顶部行设置为与其他 Zabbix 页面相同的样式。

操作视图

这是定义操作视图的参考文件。

```
<?php declared(strict_types = 1);

/**
* @var CView $this
*/

$this->includeJsFile('example.something.view.js.php');

(new CWidget())
->setTitle(_('Something view'))
->addItem(new CDiv($data['name']))
->addItem(new CPartial('module.example.something.reusable', [
'contacts' => $data['contacts']
])
->show();
```

资源

文件夹 *assets* 可能包含任何不属于其他目录的文件和子文件夹。您可以将其用于：

- JavaScript 样式（必须位于 *assets/js* 内）；
- CSS 样式（必须位于 *assets/css* 内）；
- Images；
- Fonts；
- 任何您需要包含的其他内容。

assets/js

assets/js 目录是保留的，应该只包含 JavaScript 文件。要由小部件使用，请在 *manifest.json* 中指定这些文件。

例如：

```
"assets": {
"js": ["class.widget.js"]
}
```

assets/js

assets/js 目录是保留的，应该只包含 JavaScript 文件。要由小部件使用，请在 *manifest.json* 中指定这些文件。

例如：

```
"assets":
   {"css": ["mywidget.css"]}
```

CSS 样式

CSS 文件可以包含自定义属性 `theme`，用于为特定前端主题定义不同的样式。

可用主题及其属性值：

- 蓝色 - [theme='blue-theme']
- 蓝色（经典） - [theme='blue-classic-theme']
- 深色 - [theme='dark-theme']
- 深蓝色 - [theme='dark-blue-theme']
- 深色（经典） - [theme='dark-classic-theme']
- 高对比度浅色 - [theme='hc-light']
- 高对比度深色 - [theme='hc-dark']

示例：

```
.widget {
    background-color: red;
}

[theme='dark-theme'] .widget {
    background-color: green;
}
```

注册新模块

本节介绍如何向 Zabbix 前端添加新模块。

先决条件　继续操作之前，请确保：

- 模块位于 Zabbix 前端安装的 *modules* 目录中（例如，*zabbix/ui/modules*）。
- 模块至少有一个基本版本的manifest.json 文件。
- 您可以访问 Zabbix 中的管理菜单部分（需要超级管理员用户角色类型）。

> **Note:**
> 前端不会安装和识别不兼容的模块。

添加模块　打开 *Administration* > *General* > *Modules* 页面，然后点击 *Scan directory*。



在列表中找到你的模块并将其启用。

要启用模块，请点击 *Disabled* 超链接 - 模块状态将变为 *Enabled*。

点击模块名称可查看该模块的更多信息，例如作者、版本或简短描述（如果在 manifest 中已定义）。

小组件预览　小组件模块一旦添加后，将立即显示在仪表板小组件列表中。

您可以打开一个仪表板，切换到编辑模式，并像往常一样将该小组件添加到仪表板中。

当您对小组件进行一些更改后，请刷新仪表板，以查看该小组件在最新更新后的显示效果。

小部件

小部件是用于仪表板的 Zabbix 前端模块。除非另有说明，所有模块指南也适用于小部件。

但是，小部件与模块明显不同。要构建小部件：

- 在manifest.json 文件 中指定类型 “widget”（“type”：“widget”）；
- 至少包含两个视图：一个用于小部件演示模式，另一个用于小部件配置模式（example.widget.view.php 和 example.widget.edit.php）；
- 以及用于小部件演示 (WidgetView.php) 的控制器；
- 使用并扩展默认的小部件类。

配置

本页介绍可用于创建具有自定义配置字段的小部件配置视图的类。小部件配置视图是小部件的一部分，允许用户为演示 配置小部件参数。

小部件

主要小部件类，扩展所有仪表板小部件的基类 - *CWidget*。

覆盖默认小部件行为所需。

Widget 类应位于小部件的根目录中（例如，*zabbix/ui/modules/my_custom_widget*）。

Widget.php 示例

```
<?php

命名空间 Modules\MyCustomWidget;

使用 Zabbix\Core\CWidget;

class Widget extends CWidget {

public const MY_CONSTANT = 0;

public function getTranslationStrings(): array {
return [
'class.widget.js' => [
'No data' => _('No data')
]
];
}
}
```

WidgetForm

WidgetForm 类扩展了默认类 *CWidgetForm*，并包含一组*CWidgetField* 字段这些字段是定义数据库中的小部件配置存储结构和处理输入验证所必需的。

WidgetForm 类应位于 *includes* 目录中。

如果该类具有不同的名称，则应在 *manifest.json* 文件中的*widget/form_class* 参数中指定该名称。

includes/WidgetForm.php 示例

```
<?php

命名空间 Modules\MyCustomWidget\Includes;

使用 Modules\MyCustomWidget\Widget;
```

```
使用 Zabbix\Widgets\{
CWidgetField,
CWidgetForm
};

使用 Zabbix\Widgets\Fields\{
CWidgetFieldMultiSelectItem,
CWidgetFieldTextBox,
CWidgetFieldColor
};

类 WidgetForm 扩展了 CWidgetForm {

public const DEFAULT_COLOR_PALETTE = [
'FF465C', 'B0AF07', '0EC9AC', '524BBC', 'ED1248', 'D1E754', '2AB5FF', '385CC7', 'EC1594', 'BAE37D',
'6AC8FF', 'EE2B29', '3CA20D', '6F4BBC', '00A1FF', 'F3601B', '1CAE59', '45CFDB', '894BBC', '6D6D6D'
];

公共函数 addFields(): self {
返回 $this
->addField(
(new CWidgetFieldMultiSelectItem('itemid', _('Item')))
->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
->setMultiple(false)
)
->addField(
new CWidgetFieldTextBox('description', _('Description'))
)
->addField(
(new CWidgetFieldColor('chart_color', _('Color')))->setDefault('FF0000')
);
}
}
```

CWidgetFormView

CWidgetFormView 类用于指定 *WidgetForm* 类中定义字段的呈现逻辑，确定这些字段在配置视图中渲染时的外观和行为。

CWidgetFormView 类支持以下方法：

- *addField()* - 接收 *CWidgetFieldView* 类的实例作为参数；每个*CWidgetField* 类都有一个对应的 *CWidgetFieldView* 类，用于小组件配置视图。
- *addFieldset()* - 接收 *CWidgetFieldsGroupView* 类的实例，该实例将字段组合到可折叠容器中。
- *addFieldsGroup()* - 接收 *CWidgetFormFieldsetCollapsibleView* 的实例，该实例以视觉方式（带边框）将字段组合成一个组。
- *includeJsFile()* - 用于向小组件配置视图添加 JavaScript 文件。
- *addJavaScript()* - 用于添加内联 JavaScript，该脚本将在小组件配置视图加载后立即执行。

CWidgetFormView 类应位于 *views* 目录中。

views/widget.edit.php 示例

```
<?php

/**
 * My custom widget form view.
 *
 * @var CView $this
 * @var array $data
 */

use Modules\MyCustomWidget\Includes\WidgetForm;

(new CWidgetFormView($data))
    ->addField(
        (new CWidgetFieldMultiSelectItemView($data['fields']['itemid']))->setPopupParameter('numeric', tru
    )
```

```
    ->addFieldset(
        (new CWidgetFormFieldsetCollapsibleView(_('Advanced configuration')))
            ->addField(
                new CWidgetFieldTextBoxView($data['fields']['description'])
            )
            ->addField(
                new CWidgetFieldColorView($data['fields']['chart_color'])
            )
    )
    ->includeJsFile('widget.edit.js.php')
    ->addJavaScript('my_custom_widget_form.init('.json_encode([
        'color_palette' => WidgetForm::DEFAULT_COLOR_PALETTE
    ]).');')
    ->show();
```

JavaScript

JavaScript 类可用于向小部件配置视图添加动态行为和交互性。例如，您可以初始化在*CWidgetFormView* 类中定义的颜色选择器。

JavaScript 类应随表单一起加载，因此应使用方法 *includeJsFile()* 和 *addJavaScript()* 在*CWidgetFormView* 类中引用它。

在下面的示例中，立即创建了一个单例类实例，并将其存储在 *window.my_custom_widget_form* 名称下。因此，第二次打开表单将重新创建该实例。

JavaScript 类应位于 *views* 目录中。

views/widget.edit.js.php 示例

```
<?php

使用 Modules\MyCustomWidget\Widget;

?>

window.my_custom_widget_form = new class {

init({color_palette}) {
colorPalette.setThemeColors(color_palette);

for (jQuery('.<?= ZBX_STYLE_COLOR_PICKER ?> input') 的 const colorpicker) {
jQuery(colorpicker).colorpicker();
}

const overlay = overlays_stack.getById('widget_properties');

for (['overlay.reload', 'overlay.close'] 的 const event) {
overlay.$dialogue[0].addEventListener(event, () => { jQuery.colorpicker('hide'); });
}
}
};
```

CWidgetField

CWidgetField 类是所有表单字段类（*CWidgetFieldCheckBox*、*CWidgetFieldTextArea*、*CWidgetFieldRadioButtonList* 等）的基类。扩展 *CWidgetField* 的类负责接收、保存和验证小组件配置值。

以下 *CWidgetField* 类可用。

CWidgetField 类	数据库字段类型	描述
CWidgetFieldCheckBox	int32	单个复选框。
CWidgetFieldCheckBoxList	array of int32	单个配置字段下的多个复选框。
CWidgetFieldColor	string	颜色选择字段。
CWidgetFieldDatePicker	string	日期选择字段。
CWidgetFieldIntegerBox	int32	用于输入整数的字段。可用于配置最小值和最大值。
CWidgetFieldLatLng	string	用于输入以逗号分隔的纬度、经度和地图缩放级别的文本框。
CWidgetFieldMultiSelect	-	多选字段的基类，由所有 CWidgetFieldMultiSelect* 类扩展。
CWidgetFieldMultiSelectActionID		用于从 *Alerts* > *Actions* 中定义的操作列表选择操作的多选字段。

CWidgetField 类	数据库字段类型	描述
CWidgetFieldMultiSelectGraph	ID	用于选择自定义图形的多选字段。
CWidgetFieldMultiSelectGraphPrototype	ID	用于选择自定义图形原型的多选字段。
CWidgetFieldMultiSelectGroup	ID	用于选择主机组的多选字段。
CWidgetFieldMultiSelectHost	ID	用于选择主机的多选字段。
CWidgetFieldMultiSelectHostInventory	array of int32	用于选择主机资产记录字段的多选字段。
CWidgetFieldMultiSelectItem	ID	用于选择监控项的多选字段。
CWidgetFieldMultiSelectItemPrototype	ID	用于选择监控项原型的多选字段。
CWidgetFieldMultiSelectMap	ID	用于选择地图的多选字段。
CWidgetFieldMultiSelectMediaType	ID	用于选择媒介类型的多选字段。
CWidgetFieldMultiSelectOverrideHost	ID	用于从仪表板或其他小组件中选择主机数据源的多选字段。
CWidgetFieldMultiSelectService	ID	用于选择服务的多选字段。
CWidgetFieldMultiSelectSla	ID	用于选择 SLA 的多选字段。
CWidgetFieldMultiSelectUser	ID	用于选择用户的多选字段。
CWidgetFieldNavTree	array of (multiple mixed)	存储地图的导航树结构，以供 *Map navigation tree* 小组件使用。
CWidgetFieldNumericBox	string	用于输入浮点数的字段。
CWidgetFieldPatternSelect	-	模式选择字段的基类，由所有 `CWidgetPatternSelect*` 类扩展。
CWidgetFieldPatternSelectHost	string	用于选择主机的多选字段。支持主机名称模式。
CWidgetFieldPatternSelectItem	string	用于选择监控项的多选字段。支持监控项名称模式。
CWidgetFieldRadioButtonList	int32	由一个或多个单选框组成的单选框组。
CWidgetFieldRangeControl	int32	用于选择整数值的滑块。
CWidgetFieldReference	string	存储此小组件在仪表板上的唯一标识符；用于小组件通信。
CWidgetFieldSelect	int32	下拉选择框。
CWidgetFieldSeverities	array of int32	预设了触发器严重性的 *CWidgetFieldCheckBoxList*。
CWidgetFieldSparkline	array of (multiple mixed)	允许配置迷你图表。
CWidgetFieldTags	array of (string, int32, string)	允许配置一个或多个标签过滤器行。
CWidgetFieldTextArea	string	用于输入多行文本的文本区域。
CWidgetFieldTextBox	string	用于输入单行文本的文本框。
CWidgetFieldThresholds	array of (string, string)	允许配置颜色和数值对。
CWidgetFieldTimePeriod	array of string	时间段选择字段。
CWidgetFieldTimeZone	string	包含时区的下拉列表。
CWidgetFieldUrl	string	用于输入 URL 的文本框。

Note:
int32 字段的最大长度为 10 个字符，ID 字段为 20 个字符，string 字段为 65535 个字符。
对于复合类型（例如 array of （string, int32, string)），限制分别应用于每个元素。
可以使用 setMaxLength() 等方法设置字段特定的长度限制。

演示

本页介绍可用于创建小部件演示视图的组件。小部件演示视图是小部件的一部分，它根据其配置 接收数据并将其显示在容器中的仪表板上。

演示视图由三部分组成：

- 小部件操作
- 小部件视图
- JavaScript

小部件操作

小部件操作类 (*WidgetView*) 包含用于在演示视图模式下操作小部件的方法。

大多数小部件操作使用和/或扩展默认控制器类 *CControllerDashboardWidgetView*。

小部件操作类应位于 *actions* 目录中，并在 *manifest.json* 文件中的*actions* 参数 (*actions/widget.{id}.view/class*) 中指定。

actions/WidgetView.php 示例（在 **Zabbix** 原生系统信息 小部件中实现）

```
class WidgetView extends CControllerDashboardWidgetView {

protected function doAction(): void {
$this->setResponse(new CControllerResponseData([
```

```
'name' => $this->getInput('name', $this->widget->getDefaultName()),
'system_info' => CSystemInfoHelper::getData(),
'info_type' => $this->fields_values['info_type'],
'user_type' => CWebUser::getType(),
'user' => [
'debug_mode' => $this->getDebugMode()
]
]));
}
}
```

小部件视图

小部件视图类 (*CWidgetView*) 负责构建小部件展示视图。

小部件视图类应位于 *views* 目录中。

如果包含小部件视图类的文件的名称与默认名称 (*widget.view.php*) 不同，则必须在 *manifest.json* 文件*actions* 参数 (*actions/widget.{id}.view/view*) 中指定它。

views/widget.view.php 示例

```
<?php

/**
* 我的自定义小部件视图。
*
* @var CView $this
* @var array $data
*/

(new CWidgetView($data))
->addItem(
new CTag('h1', true, $data['name'])
)
->show();
```

JavaScript

JavaScript 类负责确定仪表盘小部件的行为，如更新小部件数据、调整小部件大小、显示小部件元素等。

所有 JavaScript 操作都使用和/或扩展了所有仪表盘小部件的基础 JavaScript 类 - *CWidget*。*CWidget* 类包含一组具有默认实现的小部件行为方法。根据小部件的复杂性，这些方法可以直接使用或进行扩展。

CWidget 类包含以下方法：

- 定义小部件生命周期的方法：*onInitialize()*、*onStart()*、*onActivate()*、*onDeactivate()*、*onDestroy()*、*onEdit()*。
- 处理更新和显示小部件数据的方法：*promiseUpdate()*、*getUpdateRequestData()*、*processUpdateResponse(response)*、*processUpdateErrorResponse(error)*、*setContents(response)*。
- 修改小部件外观的方法：*onResize()*、*hasPadding()*。

JavaScript 类应位于 *assets/js* 目录中，并在 *manifest.json* 文件的*assets* (*assets/js*) 参数中指定。

生命周期方法

小部件生命周期方法由仪表板调用，并在小部件存在于仪表板期间的生命周期的不同阶段调用。

onInitialize() 方法定义了仪表盘的初始状态和/或值，但不执行任何 HTML 或数据操作。此方法在创建仪表盘（实例化仪表盘对象）时被调用，通常是通过将仪表盘添加到仪表盘页面或加载仪表盘页面来完成的。

示例：

```
onInitialize() {
    this._time_offset = 0;
    this._interval_id = null;
    this._clock_type = CWidgetClock.TYPE_ANALOG;
    this._time_zone = null;
    this._show_seconds = true;
    this._time_format = 0;
    this._tzone_format = 0;
    this._show = [];
```

```
    this._has_contents = false;
    this._is_enabled = true;
}
```

onStart() 方法定义了仪表盘的 HTML 结构，但不执行任何数据操作。此方法在仪表盘页面首次激活之前被调用，即在仪表盘及其仪表盘完全显示给用户之前。

示例：

```
onStart() {
    this._events.resize = () => {
        const padding = 25;
        const header_height = this._view_mode === ZBX_WIDGET_VIEW_MODE_HIDDEN_HEADER
            ? 0
            : this._header.offsetHeight;

        this._target.style.setProperty(
            '--content-height',
            `${this._cell_height * this._pos.height - padding * 2 - header_height}px`
        );
    }
}
```

onActivate() 方法通过启用自定义事件侦听器（用于响应用户操作）和启动小部件更新周期（用于保持其内容最新）使小部件处于活动状态并具有交互性。当仪表板页面被激活时，即当其完全显示在用户界面中时，将调用此方法。

请注意，在调用 *onActivate()* 方法之前，小部件处于非活动状态（WIDGET_STATE_INACTIVE）。成功调用后，小部件将转换为活动状态（WIDGET_STATE_ACTIVE）。在活动状态下，小部件具有响应性，可侦听事件，定期更新其内容，并可与其他小部件交互。

示例：

```
onActivate() {
this._startClock();

this._resize_observer = new ResizeObserver(this._events.resize);
this._resize_observer.observe(this._target);
}
```

onDeactivate() 方法通过停用自定义事件侦听器并停止小部件更新周期来停止小部件的任何活动和交互。当仪表板页面停用（即切换或删除）或小部件从仪表板页面中删除时，将调用此方法。

请注意，在调用 *onDeactivate()* 方法之前，小部件处于活动状态（WIDGET_STATE_ACTIVE）。成功调用后，小部件将转换为非活动状态（WIDGET_STATE_INACTIVE）。

示例：

```
onDeactivate() {
this._stopClock();
this._resize_observer.disconnect();
}
```

onDestroy() 方法在从仪表板中删除小部件之前执行清理任务，其中包括关闭在小部件初始化期间建立的数据库连接、清理临时数据以释放系统内存并避免资源泄漏、取消注册与调整大小事件或按钮点击相关的事件监听器以防止不必要的事件处理和内存泄漏等。当小部件或包含它的仪表板页面被删除时，将调用此方法。

请注意，在调用 *onDestroy()* 方法之前，处于活动状态 (WIDGET_STATE_ACTIVE) 的小部件始终通过调用 *onDeactivate()* 方法停用。

示例：

```
onDestroy() {
if (this._filter_widget) {
this._filter_widget.off(CWidgetMap.WIDGET_NAVTREE_EVENT_MARK, this._events.mark);
this._filter_widget.off(CWidgetMap.WIDGET_NAVTREE_EVENT_SELECT, this._events.select);
}
}
```

onEdit() 方法定义了仪表板切换到编辑模式时小部件的外观和行为。
当仪表板切换到编辑模式时，将调用此方法；通常是在用户与小部件的 *Edit* 按钮或仪表板的 *Edit dashboard* 按钮交互时触发。

示例：

```
onEdit() {
    this._deactivateGraph();
}
```

更新处理方法

小部件更新处理方法负责从 Zabbix 服务器或任何其他数据源检索更新的数据并将其显示在小部件中。

promiseUpdate() 方法通过检索数据来启动数据更新过程，通常使用 Web 请求或 API 调用。显示仪表板页面时会调用此方法，之后会定期调用，直到仪表板页面切换到另一个仪表板页面。

以下是大多数 Zabbix 原生小部件使用的 *promiseUpdate()* 方法的默认实现示例。在默认实现中，*promiseUpdate()* 方法遵循从服务器检索数据的一般模式。它使用适当的 URL 和请求参数创建一个新的 `Curl` 对象，使用 *fetch()* 方法发送 POST 请求，并使用由 *getUpdateRequestData()* 方法构造的数据对象，并相应地使用 *processUpdateResponse(response)* 或 *processUpdateErrorResponse(error)* 处理响应（或错误响应）。此实现适用于大多数小部件，因为它们通常以 JSON 格式检索数据并以一致的方式处理它。

```
promiseUpdate() {
const curl = new Curl('zabbix.php');

curl.setArgument('action', `widget.${this._type}.view`);

return fetch(curl.getUrl(), {
method: 'POST',
headers: {'Content-Type': 'application/json'},
body: JSON.stringify(this.getUpdateRequestData()),
signal: this._update_abort_controller.signal
})
.then((response) => response.json())
.then((response) => {
if ('error' in response) {
this.processUpdateErrorResponse(response.error);

return;
}

this.processUpdateResponse(response);
});
}
```

getUpdateRequestData() 方法通过从小部件的状态和配置中收集各种属性及其对应的值（小部件标识符、过滤器设置、时间范围等）并构造一个数据对象来准备更新小部件的服务器请求数据，该数据对象表示更新请求中要发送到服务器的必要信息。此方法仅作为默认 *promiseUpdate()* 方法的一部分调用，即在小部件更新过程中调用。

默认实现：

```
getUpdateRequestData() {
return {
templateid: this._dashboard.templateid ?? undefined,
dashid: this._dashboard.dashboardid ?? undefined,
widgetid: this._widgetid ?? undefined,
name: this._name !== '' ? this._name : undefined,
fields: Object.keys(this._fields).length > 0 ? this._fields ：未定义，
view_mode：this._view_mode，
edit_mode：this._is_edit_mode ？1：0，
dynamic_hostid：this._dashboard.templateid !== null || this.supportsDynamicHosts()
？(this._dynamic_hostid ?? 未定义)
：未定义，
...this._contents_size
};
}
```

processUpdateResponse(response) 方法处理更新请求后从服务器收到的响应，并且，如果更新过程成功且没有错误，则使用 *setContents()* 方法清除小部件数据并显示新内容。此方法仅作为默认 *promiseUpdate()* 方法的一部分调用，即在小部件更新过程中调用。

默认实现：

```
processUpdateResponse(response) {
this._setHeaderName(response.name);

this._updateMessages(response.messages);
this._updateInfo(response.info);
this._updateDebug(response.debug);

this.setContents(response);
}
```

processUpdateErrorResponse(error) 方法处理更新请求后从服务器收到的响应（如果响应为错误），并显示错误消息。此方法仅作为默认 *promiseUpdate()* 方法的一部分调用，即在小部件更新过程中调用。

默认实现：

```
processUpdateErrorResponse(error) {
this._updateMessages(error.messages, error.title);
}
```

如果小部件更新过程成功且无错误，则 ***setContents(response)*** 方法将显示小部件内容，这可能包括操作 DOM 元素、更新 UI 组件、应用样式或格式等。此方法仅作为默认 *processUpdateResponse(response)* 方法的一部分调用，即在处理更新请求后从服务器收到的响应的过程中。

默认实现：

```
setContents(response) {
this._body.innerHTML = response.body ?? '';
}
```

展示修改方法

小部件展示修改方法负责修改小部件的外观。

onResize() 方法负责调整小部件的视觉元素以适应新的小部件大小，其中可以包括重新排列元素、调整元素尺寸、文本截断、实现延迟加载以提高调整大小期间的响应能力等。当小部件调整大小时，例如，当用户手动调整小部件大小或调整浏览器窗口大小时，将调用此方法。

示例：

```
onResize() {
if (this.getState() === WIDGET_STATE_ACTIVE) {
this._startUpdating();
}
}
```

hasPadding() 方法负责在小部件配置为显示其标题 时在其底部应用 8px 垂直填充。

当仪表板页面被激活时，即当它成为用户界面中显示的页面时，将调用此方法。

默认实现：

```
hasPadding() {
return this.getViewMode() !== ZBX_WIDGET_VIEW_MODE_HIDDEN_HEADER;
}
```

对于某些小部件，需要使用所有可用的小部件空间来配置，例如，自定义背景颜色。以下是 Zabbix-native *Item value* 小部件中使用的 *hasPadding()* 方法的实现示例。

```
hasPadding() {
return false;
}
```

教程

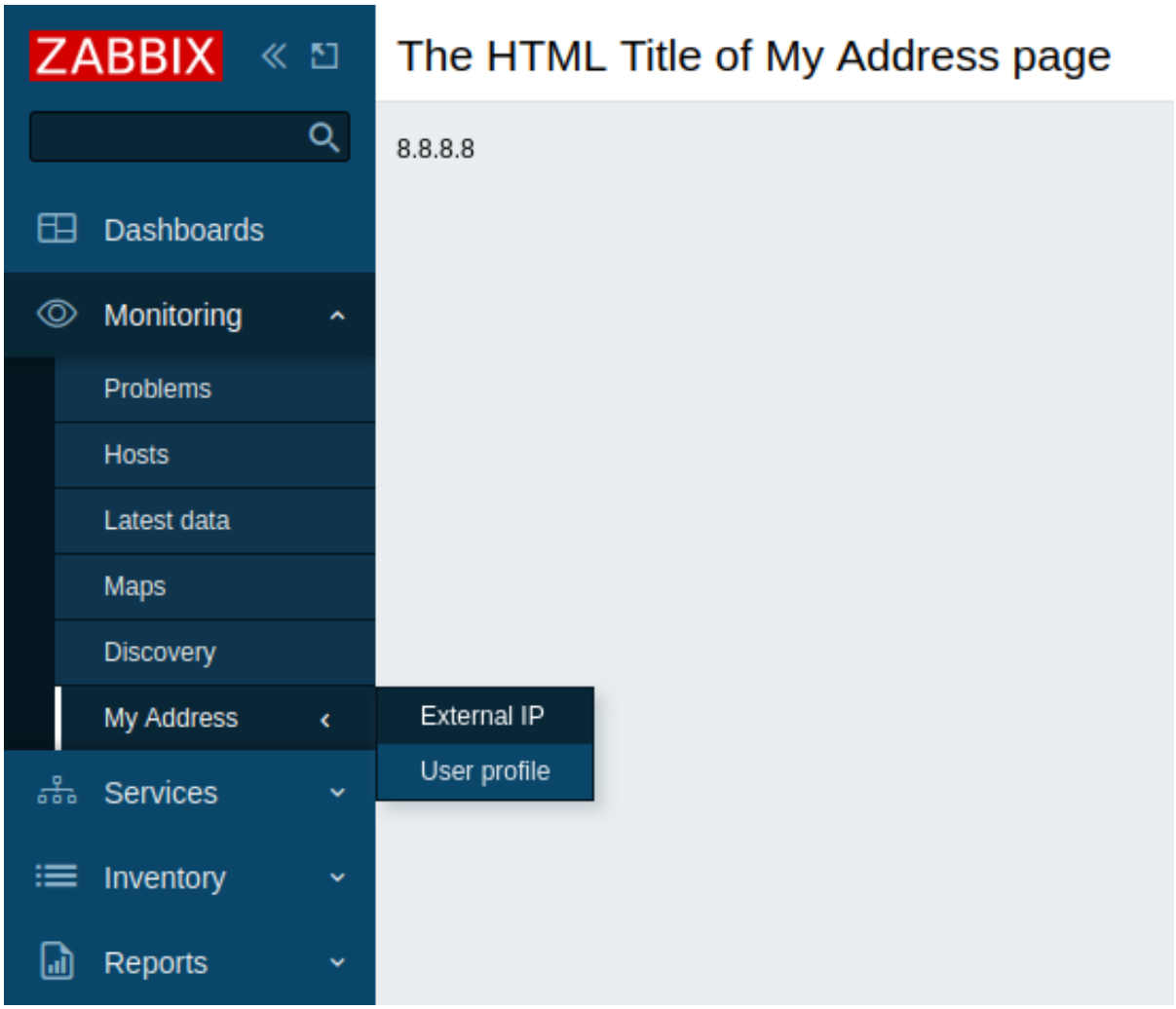
本节包含实用的分步教程，说明如何在 Zabbix 中构建自定义模块 和小部件。

创建模块（教程）

这是一个分步教程，展示了如何创建一个简单的 Zabbix 前端模块。您可以将此模块的所有文件下载为 ZIP 存档：MyAddress.zip。

您将构建的内容

在本教程中，您将首先构建一个前端模块，为菜单新增一个 *My Address* 菜单部分，然后将其转换为一个更高级的前端模块，该模块会向 *https://api.seeip.org* 发起 HTTP 请求，并在新创建的 *My Address* 菜单部分中的新页面上显示响应内容——您计算机的 IP 地址。完成后的模块如下所示：



第一部分 - 新建菜单部分

向 Zabbix 前端添加一个空模块

1. 在 Zabbix 前端安装目录的 *modules* 目录中创建一个 *MyAddress* 目录（例如，*zabbix/ui/modules*）。
2. 创建一个 *manifest.json* 文件，包含基本的模块元数据（请参见支持的参数 说明）。

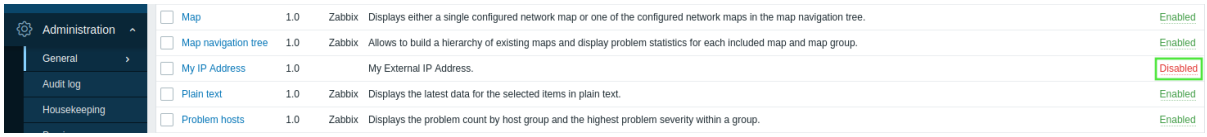
ui/modules/MyAddress/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "my-address",
    "name": "My IP Address",
    "version": "1.0",
    "namespace": "MyAddress",
    "description": "My External IP Address."
}
```

3. 在 Zabbix 前端中，转到 *Administration* > *General* > *Modules* 部分，然后单击 *Scan directory* 按钮。



4. 在列表中找到新模块 *My IP Address*，然后单击”Disabled” 超链接，将模块状态从”Disabled” 更改为”Enabled”（如果列表中没有该模块，请参见troubleshooting 部分）。



该模块现在已在前端中注册。不过，它还不可见，因为你仍然需要定义模块功能。一旦你向模块目录中添加内容，刷新页面后就会立即在 Zabbix 前端中看到更改。

创建菜单部分

1. 在 *MyAddress* 目录中创建一个 *Module.php* 文件。

此文件实现一个新的 *Module* 类，该类继承默认的 *CModule* 类。*Module* 类将把一个新的 *My Address* 菜单部分插入到主菜单中。

setAction() 方法指定在点击该菜单部分时要执行的操作。开始时，你可以使用预定义操作 *userprofile.edit*，它会打开 *User profile* 页面。在本教程的第 III 部分中，你将学习如何创建自定义操作。

ui/modules/MyAddress/Module.php

```
<?php

namespace Modules\MyAddress;

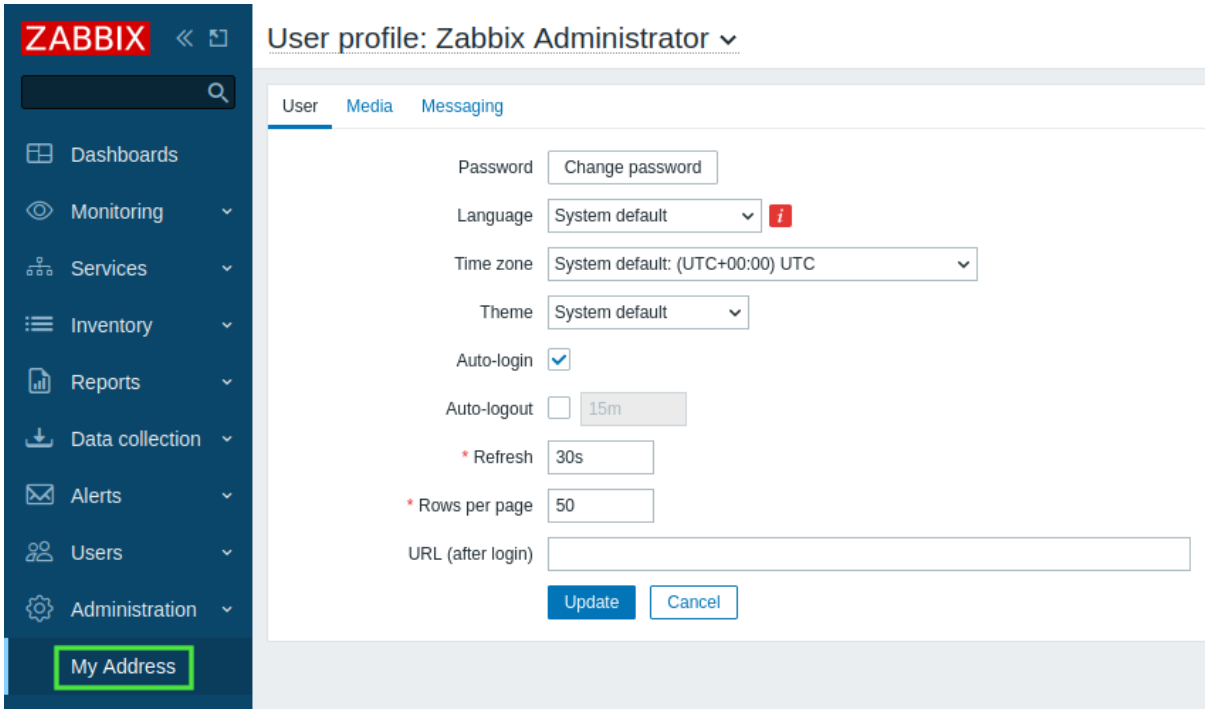
use Zabbix\Core\CModule,
    APP,
    CMenuItem;

class Module extends CModule {

    public function init(): void {
        APP::Component()->get('menu.main')
            ->add((new CMenuItem(_('My Address')))
            ->setAction('userprofile.edit'));
    }
}
```

Note:
你可以将 'userprofile.edit' 替换为其他操作，例如 'charts.view'（打开自定义图形）、'problems.view'（打开 监控 > 问题），或 'report.status'（打开 系统信息报告）。

3. 刷新 Zabbix 前端。现在在 Zabbix 主菜单底部会出现一个新的 *My Address* 部分。点击 *My Address* 可打开 *User profile* 页面。



第二部分 - 菜单部分位置变更

在这一部分中，你将把 *My Address* 菜单部分移动到 *Monitoring* 部分，然后为其添加一个嵌套菜单。
这样，用户就可以从 *Monitoring* > *My Address* 菜单部分访问两个子菜单页面。

1. 打开并编辑 *Module.php* 文件。

ui/modules/MyAddress/Module.php

```
<?php

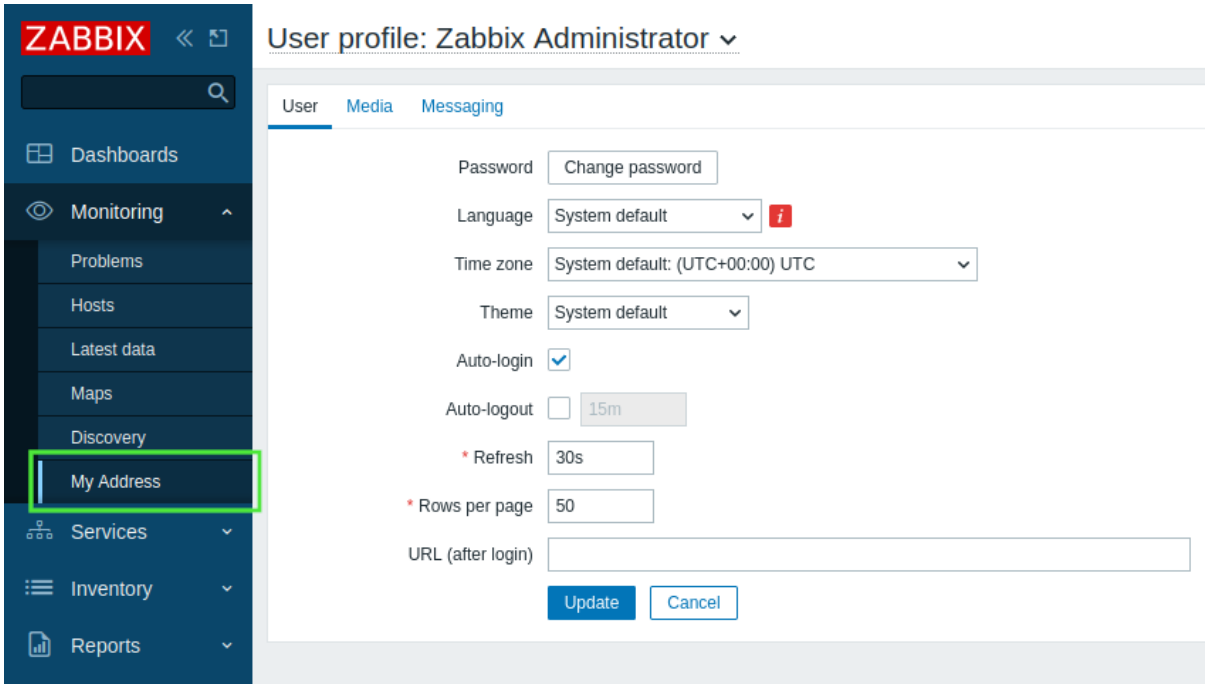
namespace Modules\MyAddress;

use Zabbix\Core\CModule,
    APP,
    CMenuItem;

class Module extends CModule {

    public function init(): void {
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Monitoring'))
            ->getSubmenu()
            ->insertAfter(_('Discovery'),
                (new CMenuItem(_('My Address')))->setAction('userprofile.edit')
            );
    }
}
```

2. 刷新 Zabbix 前端。展开 *Monitoring* 菜单部分，观察 *My address* 部分现在位于 *Discovery* 部分下方。



3. 要向 *My Address* 菜单部分添加嵌套页面，请再次打开并编辑 *Module.php* 文件。

此步骤会创建两个子部分：

- *External IP*，执行一个新的'my.address' 动作，该动作将在后续步骤中定义；
- *User profile*，执行预定义的'userprofile.edit' 动作，以打开 *User profile* 页面。

请注意，对于嵌套菜单，除了前面步骤中使用的类之外，你还需要使用 *CMenu* 类。

ui/modules/MyAddress/Module.php

```
<?php

namespace Modules\MyAddress;

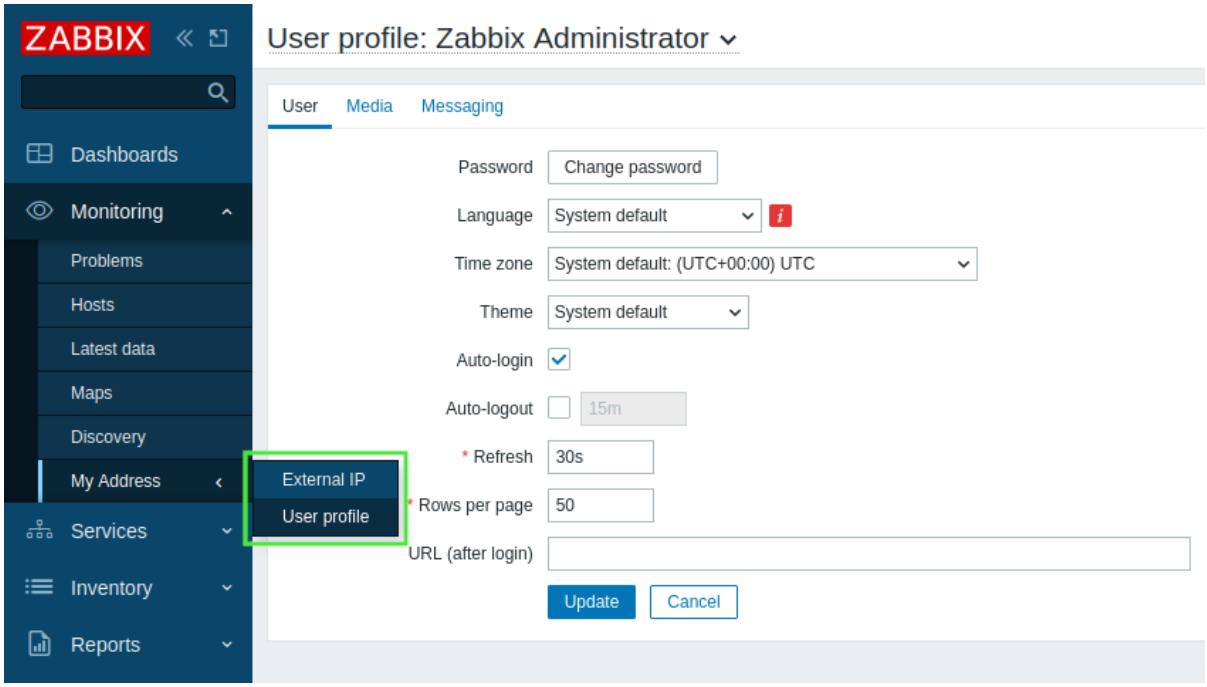
use Zabbix\Core\CModule,
    APP,
```

```
    CMenu,
    CMenuItem;

class Module extends CModule {

    public function init(): void {
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Monitoring'))
            ->getSubmenu()
            ->insertAfter(_('Discovery'),
                (new CMenuItem(_('My Address')))->setSubMenu(
                    new CMenu([
                        (new CMenuItem(_('External IP')))->setAction('my.address'),
                        (new CMenuItem(_('User profile')))->setAction('userprofile.edit')
                    ])
                )
            );
    }
}
```

4. 刷新 Zabbix 前端。观察 *My address* 菜单部分现在包含一个三级菜单，其中有两个页面 - *External IP* 和 *User profile*。



第 III 部分 - 模块 action

一个 action 由两个文件实现 - *actions/MyAddress.php* 和 *views/my.address.php*。***actions/MyAddress.php*** 文件负责业务逻辑的实现，而 ***views/my.address.php*** 文件负责视图。

1. 在 *MyAddress* 目录中创建一个 *actions* 目录。
2. 在 *actions* 目录中创建一个 *MyAddress.php* 文件。

action 逻辑将在 *MyAddress* 类中定义。这个 action 类将实现四个函数：*init()*, *checkInput()*, *checkPermissions()* 和 *doAction()*。当请求该 action 时，Zabbix 前端会调用 *doAction()* 函数。此函数负责模块的业务逻辑。

Attention:
数据必须组织为关联数组。该数组可以是多维的，并且可以包含视图所期望的任何数据。

ui/modules/MyAddress/actions/MyAddress.php

```
<?php

namespace Modules\MyAddress\Actions;

use CController,
```

```
    CControllerResponseData;

class MyAddress extends CController {

    public function init(): void {
        $this->disableCsrfValidation();
    }

    protected function checkInput(): bool {
        return true;
    }

    protected function checkPermissions(): bool {
        return true;
    }

    protected function doAction(): void {
        $data = ['my-ip' => file_get_contents("https://api.seeip.org")];
        $response = new CControllerResponseData($data);
        $this->setResponse($response);
    }
}
```

3. 在 *MyAddress* 目录中创建一个新的 *views* 目录。
4. 在 *views* 目录中创建一个 *my.address.php* 文件，并定义模块视图。

请注意，变量 $data 在视图中可直接使用，无需专门定义。框架会自动将关联数组传递给视图。

ui/modules/MyAddress/views/my.address.php

```
<?php

(new CHtmlPage())
    ->setTitle(_('The HTML Title of My Address Page'))
    ->addItem(new CDiv($data['my-ip']))
    ->show();
```

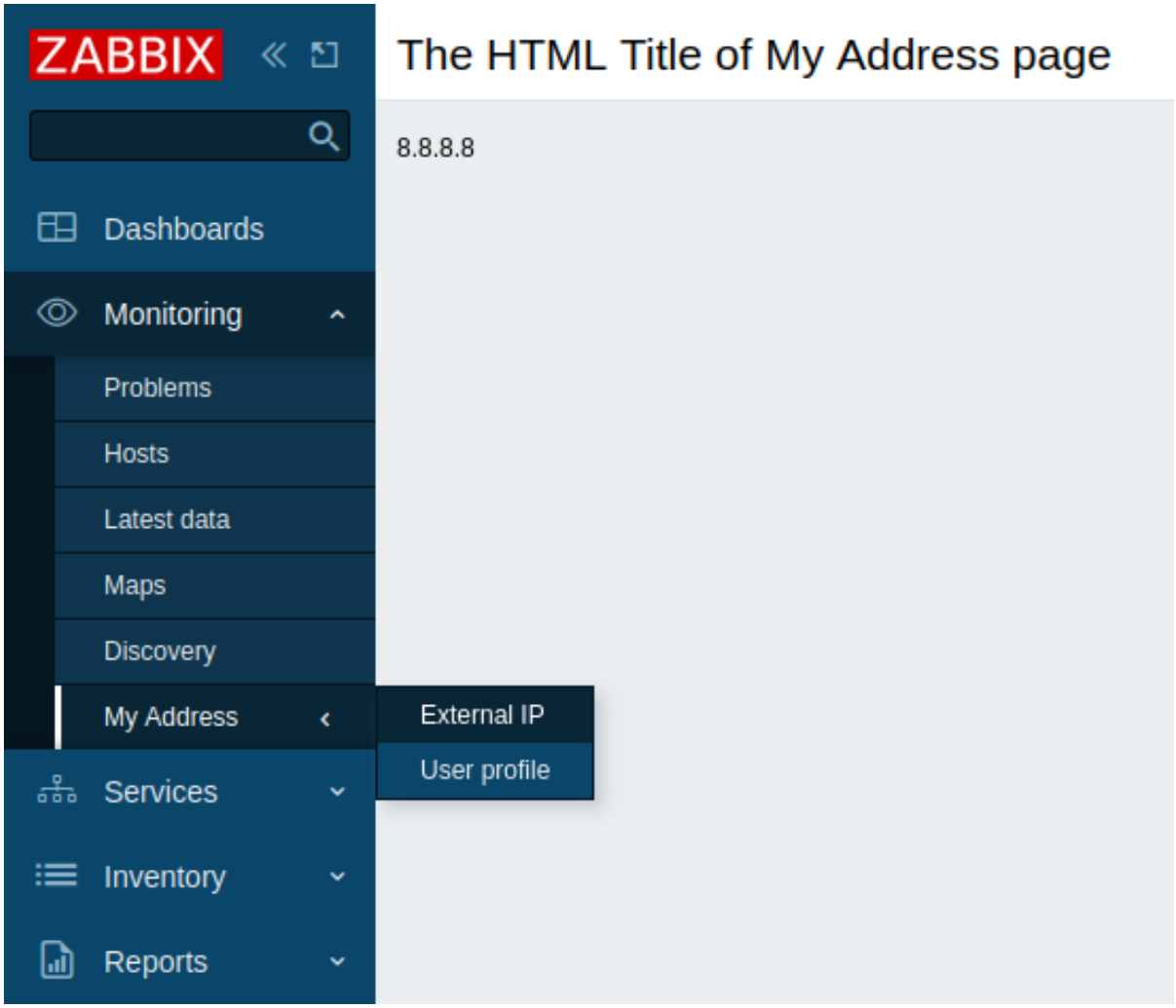
5. 必须在 *manifest.json* 文件中注册该模块 action。打开 *manifest.json* 并添加一个新的 `actions` 对象，其中包含：

- action 键，action 名称使用小写字母 (a-z) 编写，单词之间用点分隔（例如，`my.address`）；
- `my.address` 对象的 `class` 键对应 action 类名（`MyAddress`）；
- `my.address` 对象的 `view` 键对应 action 视图名（`my.address`）。

ui/modules/MyAddress/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "my-address",
    "name": "My IP Address",
    "version": "1.0",
    "namespace": "MyAddress",
    "description": "My External IP Address.",
    "actions": {
        "my.address": {
            "class": "MyAddress",
            "view": "my.address"
        }
    }
}
```

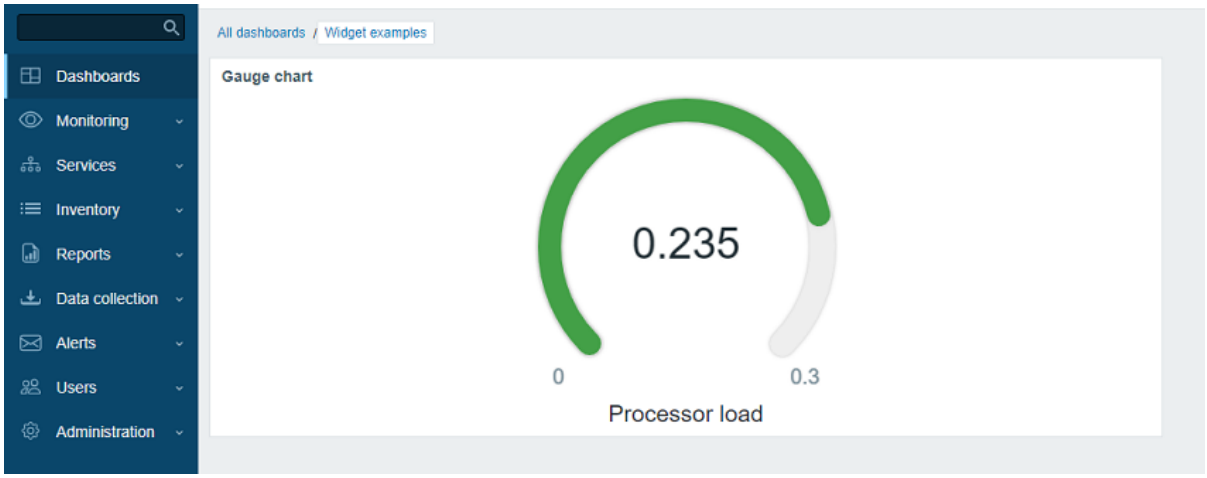
6. 刷新 Zabbix 前端。点击 *My address* > *External IP* 即可查看你电脑的 IP 地址。



创建小部件（教程）

这是一个分步教程，展示了如何创建一个简单的仪表板小部件。您可以将此小部件的所有文件下载为 ZIP 存档：lesson_gauge_chart.zip。

您将构建的内容　在本教程中，您将首先构建一个基础“Hello, world!” 小部件，然后将其转换为一个更高级的小部件，以仪表图的形式显示监控项值。完成后的小部件如下所示：



第一部分 **- “Hello, world!”**　在本部分中，您将学习如何创建所需的最小小部件元素并向 Zabbix 前端添加新小部件。

向 Zabbix 前端添加一个空白小部件

1. 在你的 Zabbix 前端安装的 `modules` 目录中创建一个 `lesson_gauge_chart` 目录（例如，*zabbix/ui/modules*）。

Note:
所有自定义小部件都被视为外部模块，必须添加到你的 Zabbix 前端安装的 `modules` 目录中（例如，*zabbix/ui/modules*）。目录 `zabbix/ui/widgets` 专用于 Zabbix 内置小部件，并会随 Zabbix UI 一起更新。

2. 创建一个包含基本小部件元数据的 `manifest.json` 文件（请参见支持的参数 说明）。

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.1",
    "author": "Zabbix"
}
```

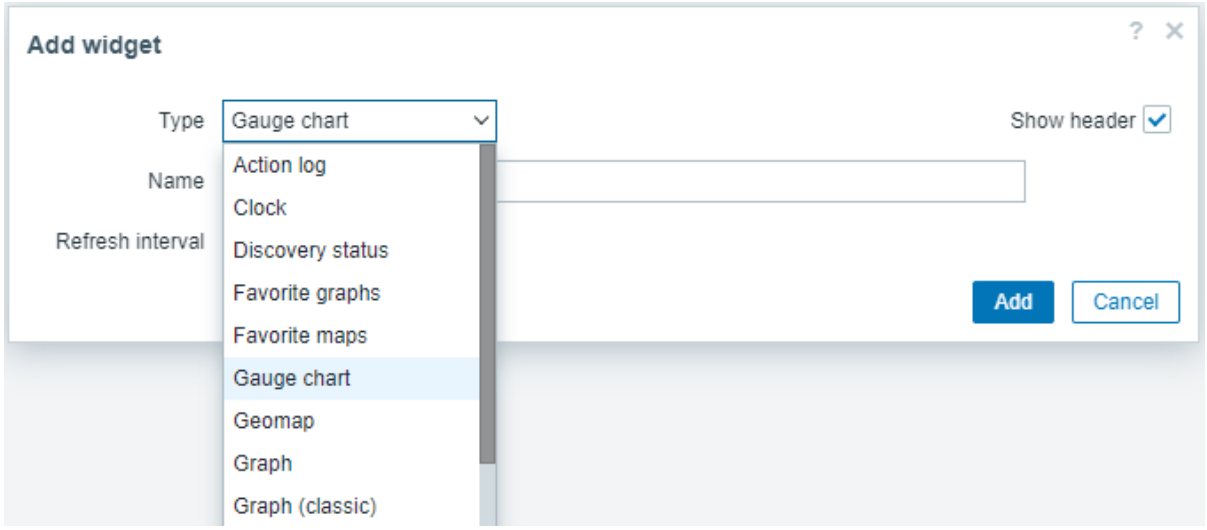
3. 在 Zabbix 前端中，转到 *Administration* > *General* > *Modules* 部分，然后点击 *Scan directory* 按钮。



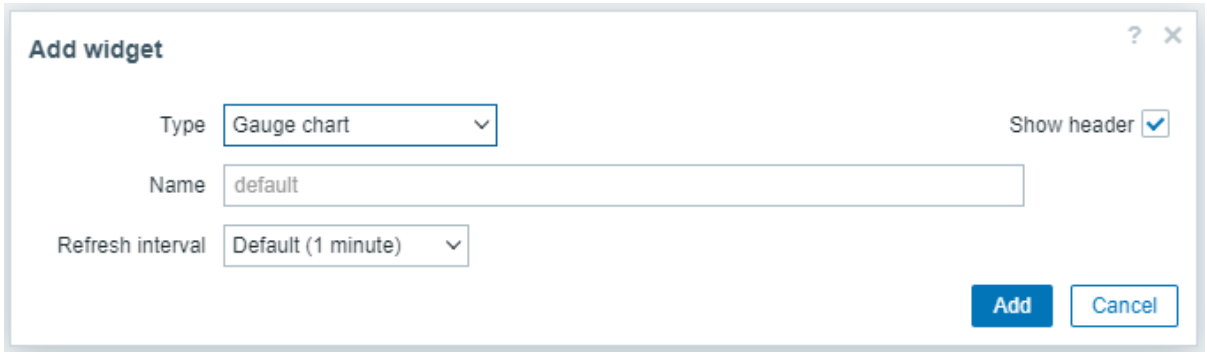
4. 在列表中找到新模块 *Gauge chart*，然后点击”Disabled” 超链接，将模块状态从”Disabled” 更改为”Enabled”（如果列表中未显示该模块，请参见troubleshooting 部分）。



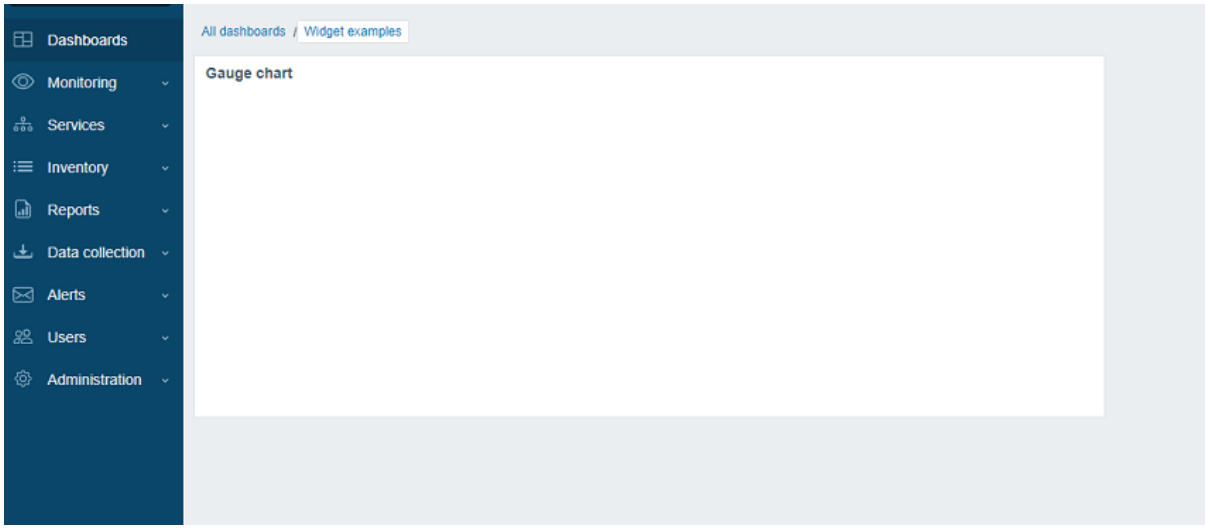
5. 打开一个仪表板，将其切换到编辑模式并添加一个新小部件。在 *Type* 字段中，选择”Gauge chart”。



6. 此时，*Gauge chart* 小部件配置只包含通用小部件字段 *Name* 和 *Refresh interval*。点击 *Add* 将小部件添加到仪表板。



7. 仪表板上应显示一个空白小部件。点击右上角的 *Save changes* 保存仪表板。



添加 widget 视图

Note:
widget 的 **view** 文件应位于 `views` 目录中（对于本教程，路径为 `ui/modules/lesson_gauge_chart/views/`）。如果该文件使用默认名称 `widget.view.php`，则无需在 `manifest.json` 文件中注册。如果文件名不同，请在 `manifest.json` 文件的`actions/widget.lesson_gauge_chart.view` 部分中指定。

1. 在 `lesson_gauge_chart` 目录中创建一个 `views` 目录。
2. 在 `views` 目录中创建一个 `widget.view.php` 文件。

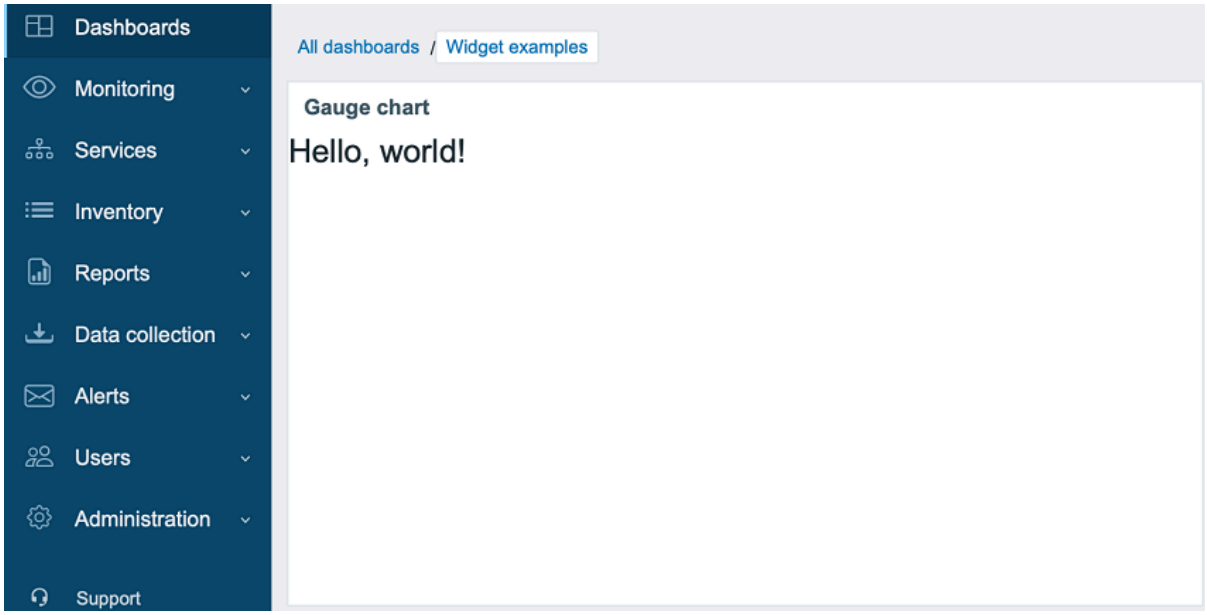
`ui/modules/lesson_gauge_chart/views/widget.view.php`

```
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem(
        new CTag('h1', true, 'Hello, world!')
    )
    ->show();
```

3. 刷新仪表板。现在 *Gauge chart* widget 会显示”Hello, world!”。



第二部分 - 仪表图　向配置视图添加设置并在 widget 视图中使用它们

在本节中，你将学习如何添加 widget 配置字段，并将输入的值作为文本显示在 widget 视图中。

widget 配置由一个表单（Zabbix\Widgets\CWidgetForm）和一个 widget 表单视图（widget.edit.php）组成。
要添加字段(Zabbix\Widgets\CWidgetField)，你需要创建一个 WidgetForm 类，该类将继承 Zabbix\Widgets\CWidgetForm。

表单包含一组不同类型的字段（Zabbix\Widgets\CWidgetField），用于验证用户输入的值。
表单字段（Zabbix\Widgets\CWidgetField）会将每种输入元素类型的值转换为单一格式，以便存储到数据库中。

> **Note:**
> widget 的 **form** 文件应位于 includes 目录中（本教程中为 ui/modules/lesson_gauge_chart/includes/）。
> 如果文件使用默认名称 WidgetForm.php，则无需在 manifest.json 文件中注册。
> 如果文件使用其他名称，请在manifest.json 文件的 widget/form_class 部分中指定它。

1. 在 lesson_gauge_chart 目录中创建一个新的 includes 目录。
2. 在 includes 目录中创建一个 WidgetForm.php 文件。

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Zabbix\Widgets\CWidgetForm;

class WidgetForm extends CWidgetForm {
}
```

3. 向 widget 配置表单添加一个 *Description* 字段。
   这是一个普通文本字段，用户可以输入任意字符集。
   你可以为此使用 CWidgetFieldTextBox 类。

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Zabbix\Widgets\CWidgetForm;

use Zabbix\Widgets\Fields\CWidgetFieldTextBox;

class WidgetForm extends CWidgetForm {

    public function addFields(): self {
```

```
        return $this
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            );
    }
}
```

4. 在 `views` 目录中，创建一个 widget 配置视图文件 `widget.edit.php`，并为新的 *Description* 字段添加一个视图。
   对于 `CWidgetFieldTextBox` 字段类，视图为 `CWidgetFieldTextBoxView`。

`ui/modules/lesson_gauge_chart/views/widget.edit.php`

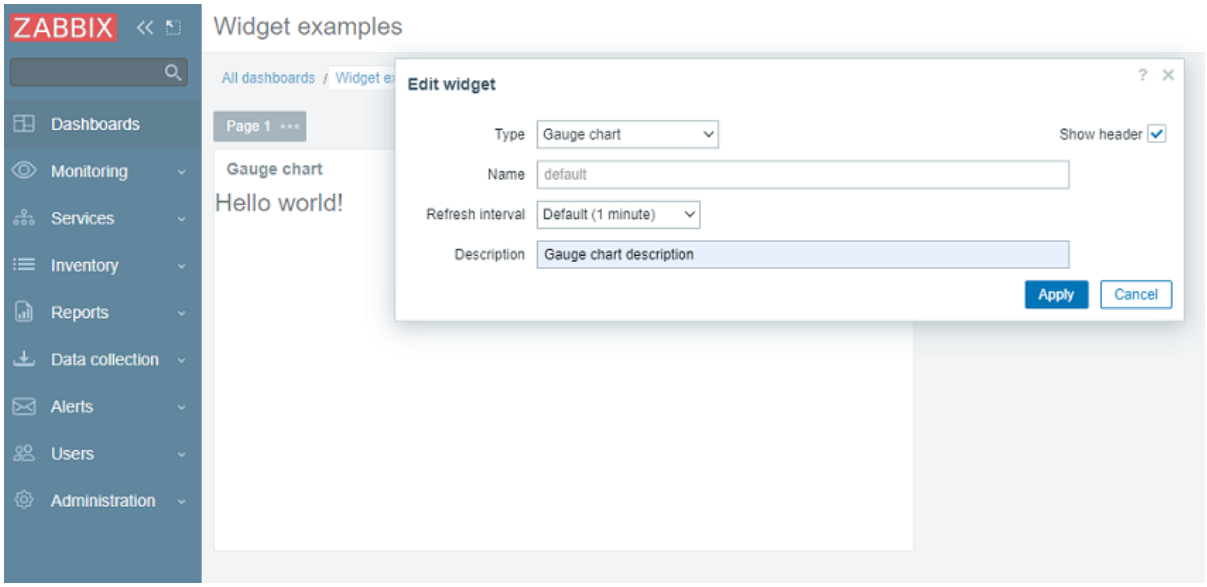
```
<?php

/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetFormView($data))
    ->addField(
        new CWidgetFieldTextBoxView($data['fields']['description'])
    )
    ->show();
```

5. 转到仪表板，并点击 widget 中的齿轮图标以打开 widget 配置表单。

6. widget 配置表单现在包含一个新的 *Description* 文本字段。
   输入任意值，例如 *Gauge chart description*。



7. 在 widget 配置表单中点击 *Apply*。
   然后点击右上角的 *Save changes* 保存仪表板。
   请注意，新的描述在任何地方都不可见，而 widget 仍然显示"Hello, world!"。

要让新的描述显示在 widget 中，需要从数据库中检索 *Description* 字段的值并将其传递给 widget 视图。
为此，你需要创建一个 action 类。

8. 在 `lesson_gauge_chart` 目录中创建一个新的 `actions` 目录。

9. 在 `actions` 目录中创建一个 `WidgetView.php` 文件。
   `WidgetView` action 类将继承 `CControllerDashboardWidgetView` 类。

widget 配置字段的值存储在 action 类的 `$fields_values` 属性中。

`ui/modules/lesson_gauge_chart/actions/WidgetView.php`

```php
<?php

namespace Modules\LessonGaugeChart\Actions;

use CControllerDashboardWidgetView,
    CControllerResponseData;

class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getName()),
            'description' => $this->fields_values['description'],
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

10. 打开 manifest.json，并在 actions/widget.lesson_gauge_chart.view 部分将 WidgetView 注册为 action 类。

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.0",
    "author": "Zabbix",
    "actions": {
        "widget.lesson_gauge_chart.view": {
            "class": "WidgetView"
        }
    }
}
```

11. 现在你可以在 widget 视图中使用包含在 $data['description'] 中的 description 字段值。
    打开 views/widget.view.php，并将静态文本"Hello, world!" 替换为 $data['description']。

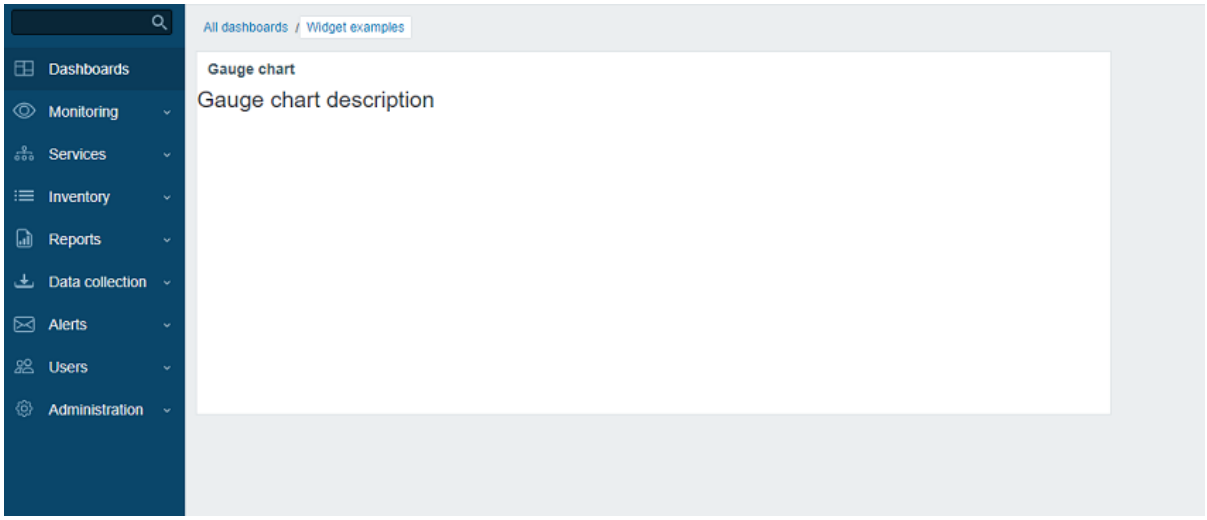
ui/modules/lesson_gauge_chart/views/widget.view.php

```php
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem(
        new CTag('h1', true, $data['description'])
    )
    ->show();
```

12. 刷新仪表板页面。
    现在你应该会看到 widget 描述文本，而不是"Hello, world!"。



通过 API 获取监控项值

该小组件应显示用户所选监控项的最新值。
为此，需要为小组件配置添加选择监控项的能力。

在本节中，你将学习如何在小组件表单中添加监控项选择字段，以及如何在配置视图中添加该字段的可视化部分。
随后，小组件控制器将能够通过 API 请求获取监控项数据及其值。
一旦收到该值，就可以在小组件视图中显示出来。

1. 打开 includes/WidgetForm.php 并添加 CWidgetFieldMultiSelectItem 字段。
   这将允许在配置表单中选择监控项。

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Zabbix\Widgets\{
    CWidgetField,
    CWidgetForm
};

use Zabbix\Widgets\Fields\{
    CWidgetFieldMultiSelectItem,
    CWidgetFieldTextBox
};

/**
 * Gauge chart widget form.
 */
class WidgetForm extends CWidgetForm {

    public function addFields(): self {
        return $this
            ->addField(
                (new CWidgetFieldMultiSelectItem('itemid', _('Item')))
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
                    ->setMultiple(false)
            )
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            );
    }
}
```

2. 打开 views/widget.edit.php 并将该字段的可视组件添加到配置视图中。

ui/modules/lesson_gauge_chart/views/widget.edit.php

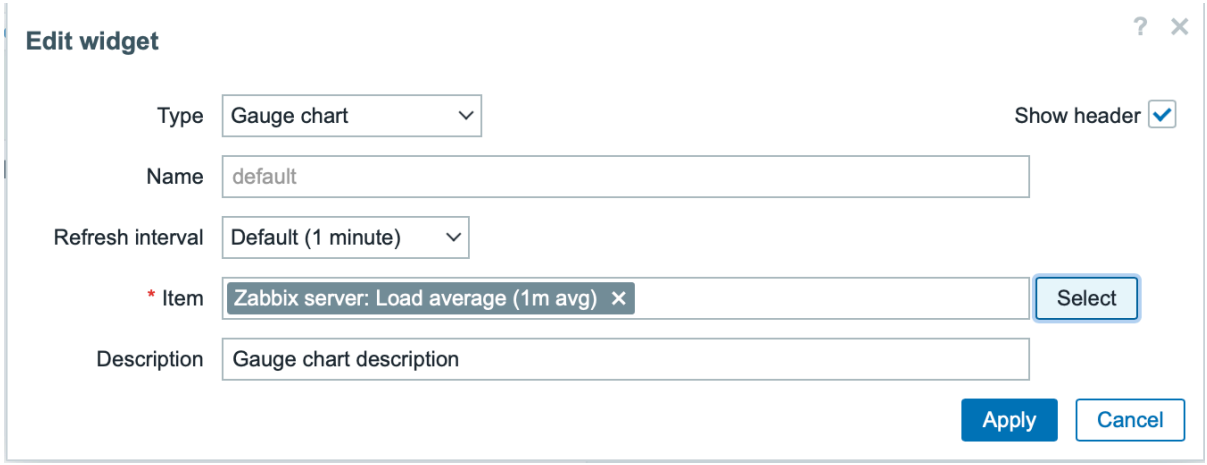
```php
<?php

/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetFormView($data))
    ->addField(
        new CWidgetFieldMultiSelectItemView($data['fields']['itemid'])
    )
    ->addField(
        new CWidgetFieldTextBoxView($data['fields']['description'])
    )
    ->show();
```

3. 返回仪表板，并点击小组件中的齿轮图标以打开小组件配置表单。

4. 现在，小组件配置表单中包含一个新的输入字段 *Item*。
   选择主机"Zabbix server" 和监控项"Load average (1m avg)"。



5. 在小组件配置表单中点击 *Apply*。
   然后点击右上角的 *Save changes* 保存仪表板。

6. 打开并修改 `actions/WidgetView.php`。

从现在开始，监控项 ID 将在小组件控制器中的 `$this->fields_values['itemid']` 中可用。
`doAction()` 控制器方法使用 API 方法`item.get` 获取监控项数据（名称、值类型、单位），并使用 API 方法`history.get` 获取监控项的最新值。

`ui/modules/lesson_gauge_chart/actions/WidgetView.php`

```php
<?php

namespace Modules\LessonGaugeChart\Actions;

use API,
    CControllerDashboardWidgetView,
    CControllerResponseData;

class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $db_items = API::Item()->get([
            'output' => ['itemid', 'value_type', 'name', 'units'],
            'itemids' => $this->fields_values['itemid'],
            'webitems' => true,
            'filter' => [
                'value_type' => [ITEM_VALUE_TYPE_UINT64, ITEM_VALUE_TYPE_FLOAT]
```

```
            ]
        ]);

        $value = null;

        if ($db_items) {
            $item = $db_items[0];

            $history = API::History()->get([
                'output' => API_OUTPUT_EXTEND,
                'itemids' => $item['itemid'],
                'history' => $item['value_type'],
                'sortfield' => 'clock',
                'sortorder' => ZBX_SORT_DOWN,
                'limit' => 1
            ]);

            if ($history) {
                $value = convertUnitsRaw([
                    'value' => $history[0]['value'],
                    'units' => $item['units']
                ]);
            }
        }

        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getName()),
            'value' => $value,
            'description' => $this->fields_values['description'],
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

7. 打开 views/widget.view.php 并将监控项值添加到小组件视图中。

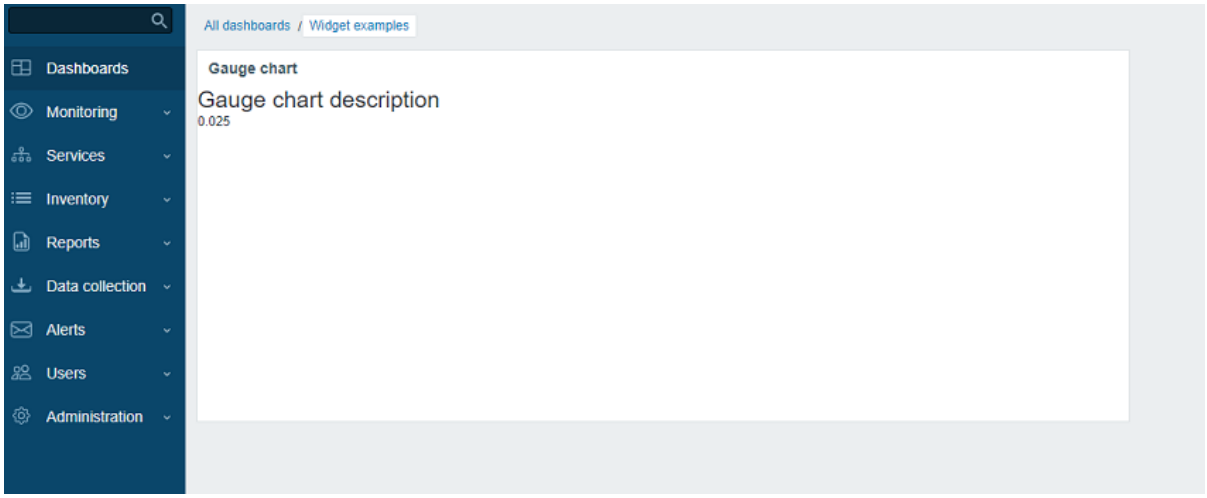
ui/modules/lesson_gauge_chart/views/widget.view.php

```
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem([
        new CTag('h1', true, $data['description']),
        new CDiv($data['value'] !== null ? $data['value']['value'] : _('No data'))
    ])
    ->show();
```

8. 刷新仪表板页面。
   小组件将显示最新的监控项值。



将高级配置设置添加到配置视图

在本节中，你将学习如何添加一个可展开/折叠的 高级配置区域，其中包含可选参数，例如颜色、最小值和最大值、单位，以及前面创建的 *Description* 字段。

1. 在主 widget 目录 `lesson_gauge_chart` 中创建一个 `Widget.php` 文件，以创建一个新的 `Widget` 类。

`Widget` 类将扩展 `CWidget` 基类，以添加/覆盖默认的 widget 设置（在本例中为翻译）。下面提供的 JavaScript 会在缺少数据时显示字符串"No data"。"No data" 字符串存在于 Zabbix UI 翻译文件中。

如果存在任何 widget 常量，建议也在 `Widget` 类中进行定义。

`ui/modules/lesson_gauge_chart/Widget.php`

```
<?php

namespace Modules\LessonGaugeChart;

use Zabbix\Core\CWidget;

class Widget extends CWidget {

    public const UNIT_AUTO = 0;
    public const UNIT_STATIC = 1;

    public function getTranslationStrings(): array {
        return [
            'class.widget.js' => [
                'No data' => _('No data')
            ]
        ];
    }
}
```

2. 打开 `includes/WidgetForm.php`，添加新的字段 *Color*（颜色选择器）、*Min*（数值字段）、*Max*（数值字段）和 *Units*（下拉选择），并为颜色选择器定义默认调色板，以便在后续步骤中使用。

`ui/modules/lesson_gauge_chart/includes/WidgetForm.php`

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Modules\LessonGaugeChart\Widget;

use Zabbix\Widgets\{
    CWidgetField,
    CWidgetForm
};

use Zabbix\Widgets\Fields\{
```

```
    CWidgetFieldColor,
    CWidgetFieldMultiSelectItem,
    CWidgetFieldNumericBox,
    CWidgetFieldSelect,
    CWidgetFieldTextBox
};

/**
 * Gauge chart widget form.
 */
class WidgetForm extends CWidgetForm {

    public const DEFAULT_COLOR_PALETTE = [
        'FF465C', 'B0AF07', '0EC9AC', '524BBC', 'ED1248', 'D1E754', '2AB5FF', '385CC7', 'EC1594', 'BAE37D',
        '6AC8FF', 'EE2B29', '3CA20D', '6F4BBC', '00A1FF', 'F3601B', '1CAE59', '45CFDB', '894BBC', '6D6D6D'
    ];

    public function addFields(): self {
        return $this
            ->addField(
                (new CWidgetFieldMultiSelectItem('itemid', _('Item')))
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
                    ->setMultiple(false)
            )
            ->addField(
                (new CWidgetFieldColor('chart_color', _('Color')))->setDefault('FF0000')
            )
            ->addField(
                (new CWidgetFieldNumericBox('value_min', _('Min')))
                    ->setDefault(0)
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
            )
            ->addField(
                (new CWidgetFieldNumericBox('value_max', _('Max')))
                    ->setDefault(100)
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
            )
            ->addField(
                (new CWidgetFieldSelect('value_units', _('Units'), [
                    Widget::UNIT_AUTO => _x('Auto', 'history source selection method'),
                    Widget::UNIT_STATIC => _x('Static', 'history source selection method')
                ]))->setDefault(Widget::UNIT_AUTO)
            )
            ->addField(
                (new CWidgetFieldTextBox('value_static_units'))
            )
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            );
    }
}
```

3. 打开 views/widget.edit.php，并将字段的可视化组件添加到配置视图中。

ui/modules/lesson_gauge_chart/views/widget.edit.php

```
<?php

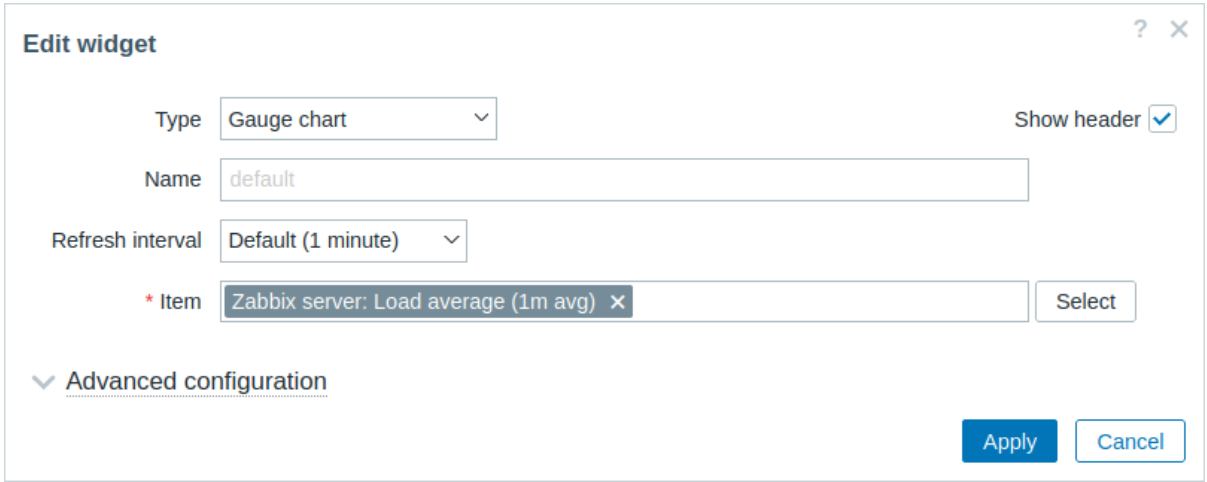
/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */
```

```
$lefty_units = new CWidgetFieldSelectView($data['fields']['value_units']);
$lefty_static_units = (new CWidgetFieldTextBoxView($data['fields']['value_static_units']))
    ->setPlaceholder(_('value'))
    ->setWidth(ZBX_TEXTAREA_TINY_WIDTH);

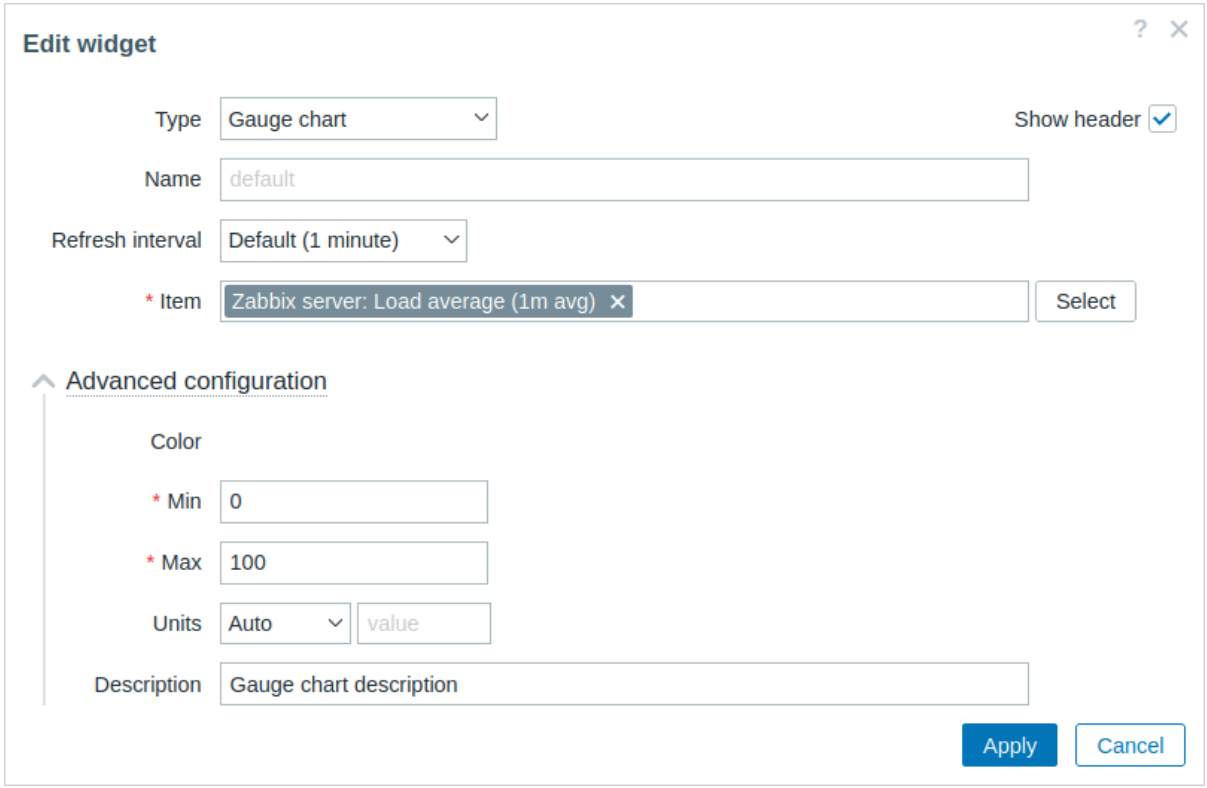
(new CWidgetFormView($data))
    ->addField(
        (new CWidgetFieldMultiSelectItemView($data['fields']['itemid']))
            ->setPopupParameter('numeric', true)
    )
    ->addFieldset(
        (new CWidgetFormFieldsetCollapsibleView(_('Advanced configuration')))
            ->addField(
                new CWidgetFieldColorView($data['fields']['chart_color'])
            )
            ->addField(
                new CWidgetFieldNumericBoxView($data['fields']['value_min'])
            )
            ->addField(
                new CWidgetFieldNumericBoxView($data['fields']['value_max'])
            )
            ->addItem([
                $lefty_units->getLabel(),
                (new CFormField([
                    $lefty_units->getView()->addClass(ZBX_STYLE_FORM_INPUT_MARGIN),
                    $lefty_static_units->getView()
                ]))
            ])
            ->addField(
                new CWidgetFieldTextBoxView($data['fields']['description'])
            )
    )
    ->show();
```

Note:
CWidgetFormView 类的 addField() 方法将 CSS 类字符串作为第二个参数。

4. 返回仪表板，切换到编辑模式，然后点击 widget 中的齿轮图标以打开 widget 配置表单。现在，widget 配置表单包含一个新的可展开/折叠区域 *Advanced configuration*。



5. 展开 *Advanced configuration* 区域以查看其他 widget 配置字段。请注意，字段 *Color* 目前还没有颜色选择器。这是因为颜色选择器必须通过 JavaScript 初始化，这将在下一节中添加 - 向 *widget* 添加 *JavaScript*。



将 JavaScript 添加到 widget

在本节中，你将学习如何添加一个使用 JavaScript 制作的仪表盘图表，用于显示最新值是正常、过高还是过低。

1. 在 `views` 目录中创建 `widget.edit.js.php` 文件。

JavaScript 将负责在配置视图中初始化颜色选择器。

`ui/modules/lesson_gauge_chart/views/widget.edit.js.php`

```
<?php

use Modules\LessonGaugeChart\Widget;

?>

window.widget_lesson_gauge_chart_form = new class {

    init({color_palette}) {
        this._unit_select = document.getElementById('value_units');
        this._unit_value = document.getElementById('value_static_units');

        this._unit_select.addEventListener('change', () => this.updateForm());

        colorPalette.setThemeColors(color_palette);

        for (const colorpicker of jQuery('.<?= ZBX_STYLE_COLOR_PICKER ?> input')) {
            jQuery(colorpicker).colorpicker();
        }

        const overlay = overlays_stack.getById('widget_properties');

        for (const event of ['overlay.reload', 'overlay.close']) {
            overlay.$dialogue[0].addEventListener(event, () => { jQuery.colorpicker('hide'); });
        }

        this.updateForm();
    }

    updateForm() {
```

```
        this._unit_value.disabled = this._unit_select.value == <?= Widget::UNIT_AUTO ?>;
    }
};
```

2. 打开 views/widget.edit.php，并将 widget.edit.js.php 文件中的 JavaScript 添加到配置视图中。
   为此，请使用 includeJsFile() 方法。
   若要添加内联 JavaScript，请使用 addJavaScript() 方法。

ui/modules/lesson_gauge_chart/views/widget.edit.php

```
<?php

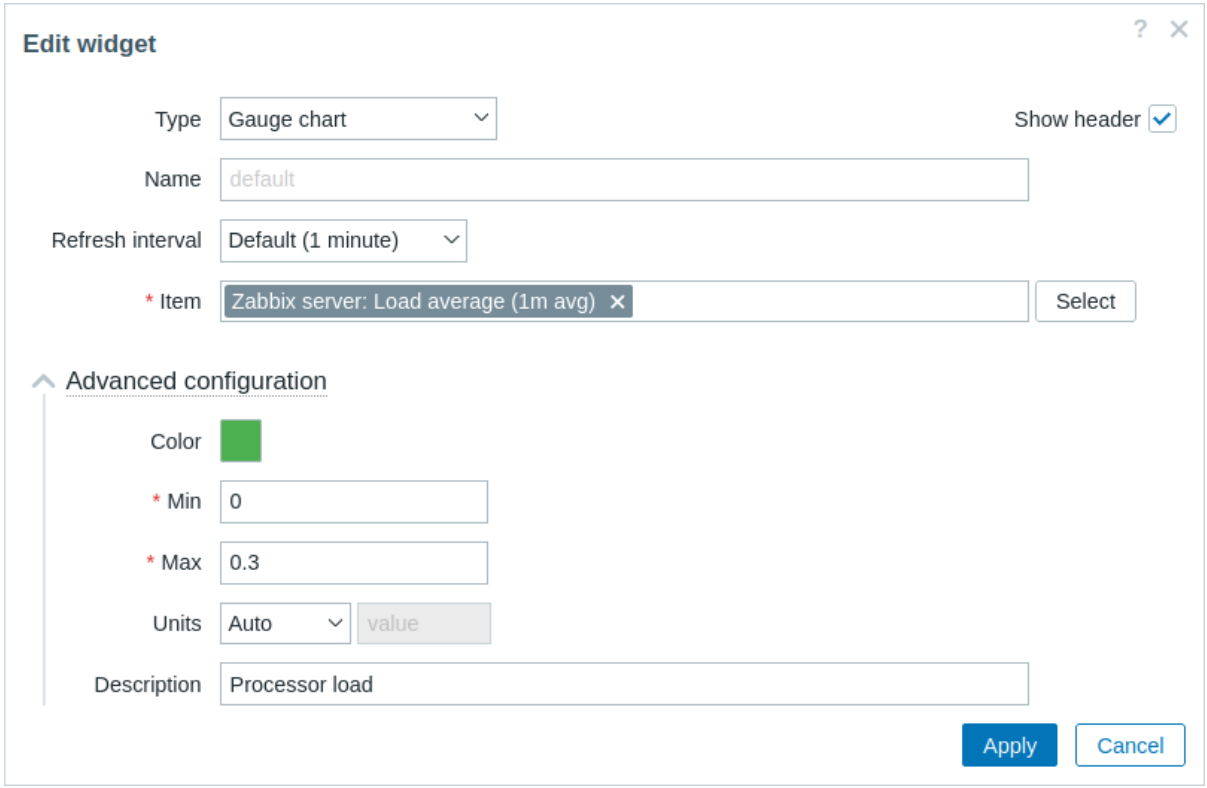
/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */

use Modules\LessonGaugeChart\Includes\WidgetForm;

$lefty_units = new CWidgetFieldSelectView($data['fields']['value_units']);
$lefty_static_units = (new CWidgetFieldTextBoxView($data['fields']['value_static_units']))
    ->setPlaceholder(_('value'))
    ->setWidth(ZBX_TEXTAREA_TINY_WIDTH);

(new CWidgetFormView($data))
    ->addField(
        (new CWidgetFieldMultiSelectItemView($data['fields']['itemid']))
            ->setPopupParameter('numeric', true)
    )
    ->addFieldset(
        (new CWidgetFormFieldsetCollapsibleView(_('Advanced configuration')))
            ->addField(
                new CWidgetFieldColorView($data['fields']['chart_color'])
            )
            ->addField(
                new CWidgetFieldNumericBoxView($data['fields']['value_min'])
            )
            ->addField(
                new CWidgetFieldNumericBoxView($data['fields']['value_max'])
            )
            ->addItem([
                $lefty_units->getLabel(),
                (new CFormField([
                    $lefty_units->getView()->addClass(ZBX_STYLE_FORM_INPUT_MARGIN),
                    $lefty_static_units->getView()
                ]))
            ])
            ->addField(
                new CWidgetFieldTextBoxView($data['fields']['description'])
            )
    )
    ->includeJsFile('widget.edit.js.php')
    ->addJavaScript('widget_lesson_gauge_chart_form.init('.json_encode([
        'color_palette' => WidgetForm::DEFAULT_COLOR_PALETTE
    ], JSON_THROW_ON_ERROR).');')
    ->show();
```

3. 返回仪表板，点击 widget 中的齿轮图标以打开 widget 配置表单。
   然后展开 *Advanced configuration* 部分，查看已初始化的颜色选择器。
   填写各字段的值，并为仪表盘图表选择一种颜色。



4. 在 widget 配置表单中点击 *Apply*。
   然后点击右上角的 *Save changes* 保存仪表板。

5. 打开 `actions/WidgetView.php` 并更新控制器。

现在，`$this->fields_values` 属性包含所有 *Advanced configuration* 字段的值。
完成控制器，以便将配置和所选监控项值传递给 widget 视图。

`ui/modules/lesson_gauge_chart/actions/WidgetView.php`

```
<?php

namespace Modules\LessonGaugeChart\Actions;

use API,
    CControllerDashboardWidgetView,
    CControllerResponseData;

class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $db_items = API::Item()->get([
            'output' => ['itemid', 'value_type', 'name', 'units'],
            'itemids' => $this->fields_values['itemid'],
            'webitems' => true,
            'filter' => [
                'value_type' => [ITEM_VALUE_TYPE_UINT64, ITEM_VALUE_TYPE_FLOAT]
            ]
        ]);

        $history_value = null;

        if ($db_items) {
            $item = $db_items[0];

            $history = API::History()->get([
                'output' => API_OUTPUT_EXTEND,
                'itemids' => $item['itemid'],
                'history' => $item['value_type'],
                'sortfield' => 'clock',
```

```
                'sortorder' => ZBX_SORT_DOWN,
                'limit' => 1
            ]);

            if ($history) {
                $history_value = convertUnitsRaw([
                    'value' => $history[0]['value'],
                    'units' => $item['units']
                ]);
            }
        }

        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getName()),
            'history' => $history_value,
            'fields_values' => $this->fields_values,
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

6. 打开并修改 `views/widget.view.php`。

你需要为仪表盘图表创建一个容器，后续步骤中将在其中绘制图表，以及一个用于描述的容器。

若要将值作为 JSON 对象传递给 JavaScript，请使用 `setVar()` 方法。

`ui/modules/lesson_gauge_chart/views/widget.view.php`

```
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem([
        (new CDiv())->addClass('chart'),
        $data['fields_values']['description']
            ? (new CDiv($data['fields_values']['description']))->addClass('description')
            : null
    ])
    ->setVar('history', $data['history'])
    ->setVar('fields_values', $data['fields_values'])
    ->show();
```

7. 在 `lesson_gauge_chart` 目录中创建一个新的 `assets` 目录。
   该目录将用于存放 JavaScript、CSS，以及可能的其他资源，例如字体或图片。
8. 对于 widget 视图 JavaScript，在 `assets` 目录中创建一个 `js` 目录。
9. 在 `assets/js` 目录中创建 `class.widget.js` 文件。

这个 JavaScript widget 类将扩展所有仪表板 widget 的基础 JavaScript 类 - `CWidget`。

仪表板依赖于 widget 的正确实现，并通过调用相应的 JavaScript 方法向 widget 传递任何相关信息。
仪表板还期望 widget 在发生某些交互时生成事件。
因此，`CWidget` 类包含一组 widget 行为的默认实现方法，这些方法可以通过继承该类进行自定义。

在本例中，需要进行一些自定义，因此将为以下 widget 行为实现自定义逻辑：

- widget 初始化，负责定义 widget 的初始状态（参见 `onInitialize()` 方法）；

- 在 widget 更新过程成功且无错误时显示 widget 内容（即绘制仪表盘图表）（参见 processUpdateResponse(response) 方法以及相关的 _resizeChart() 和 _updatedChart() 方法）
- 调整 widget 大小（参见 onResize() 方法以及相关的 _resizeChart() 方法）

对于仪表盘图表 widget 的其他方面，将使用 widget 行为的默认实现。
要了解有关 CWidget 类的 JavaScript 方法的更多信息，请参见：JavaScript。

由于该 JavaScript 是 widget 视图所必需的，因此应随仪表板页面一起加载。
要启用 JavaScript 加载，你需要在 manifest.json 文件中更新 assets/js 和 js_class 参数，如第 10 步所示。

ui/modules/lesson_gauge_chart/assets/js/class.widget.js

```
class WidgetLessonGaugeChart extends CWidget {

    static UNIT_AUTO = 0;
    static UNIT_STATIC = 1;

    onInitialize() {
        super.onInitialize();

        this._refresh_frame = null;
        this._chart_container = null;
        this._canvas = null;
        this._chart_color = null;
        this._min = null;
        this._max = null;
        this._value = null;
        this._last_value = null;
        this._units = '';
    }

    processUpdateResponse(response) {
        if (response.history === null) {
            this._value = null;
            this._units = '';
        }
        else {
            this._value = Number(response.history.value);
            this._units = response.fields_values.value_units == WidgetLessonGaugeChart.UNIT_AUTO
                ? response.history.units
                : response.fields_values.value_static_units;
        }

        this._chart_color = response.fields_values.chart_color;
        this._min = Number(response.fields_values.value_min);
        this._max = Number(response.fields_values.value_max);

        super.processUpdateResponse(response);
    }

    setContents(response) {
        if (this._canvas === null) {
            super.setContents(response);

            this._chart_container = this._body.querySelector('.chart');
            this._chart_container.style.height =
                `${this._getContentsSize().height - this._body.querySelector('.description').clientHeight}
            this._canvas = document.createElement('canvas');

            this._chart_container.appendChild(this._canvas);

            this._resizeChart();
        }

        this._updatedChart();
```

```
    }

    onResize() {
        super.onResize();

        if (this._state === WIDGET_STATE_ACTIVE) {
            this._resizeChart();
        }
    }

    _resizeChart() {
        const ctx = this._canvas.getContext('2d');
        const dpr = window.devicePixelRatio;

        this._canvas.style.display = 'none';
        const size = Math.min(this._chart_container.offsetWidth, this._chart_container.offsetHeight);
        this._canvas.style.display = '';

        this._canvas.width = size * dpr;
        this._canvas.height = size * dpr;

        ctx.scale(dpr, dpr);

        this._canvas.style.width = `${size}px`;
        this._canvas.style.height = `${size}px`;

        this._refresh_frame = null;

        this._updatedChart();
    }

    _updatedChart() {
        if (this._last_value === null) {
            this._last_value = this._min;
        }

        const start_time = Date.now();
        const end_time = start_time + 400;

        const animate = () => {
            const time = Date.now();

            if (time <= end_time) {
                const progress = (time - start_time) / (end_time - start_time);
                const smooth_progress = 0.5 + Math.sin(Math.PI * (progress - 0.5)) / 2;
                let value = this._value !== null ? this._value : this._min;
                value = (this._last_value + (value - this._last_value) * smooth_progress - this._min) / (t

                const ctx = this._canvas.getContext('2d');
                const size = this._canvas.width;
                const char_weight = size / 12;
                const char_shadow = 3;
                const char_x = size / 2;
                const char_y = size / 2;
                const char_radius = (size - char_weight) / 2 - char_shadow;

                const font_ratio = 32 / 100;

                ctx.clearRect(0, 0, size, size);

                ctx.beginPath();
                ctx.shadowBlur = char_shadow;
```

```
                ctx.shadowColor = '#bbb';
                ctx.strokeStyle = '#eee';
                ctx.lineWidth = char_weight;
                ctx.lineCap = 'round';
                ctx.arc(char_x, char_y, char_radius, Math.PI * 0.749, Math.PI * 2.251, false);
                ctx.stroke();

                ctx.beginPath();
                ctx.strokeStyle = `#${this._chart_color}`;
                ctx.lineWidth = char_weight - 2;
                ctx.lineCap = 'round';
                ctx.arc(char_x, char_y, char_radius, Math.PI * 0.75,
                    Math.PI * (0.75 + (1.5 * Math.min(1, Math.max(0, value)))), false
                    );
                ctx.stroke();

                ctx.shadowBlur = 2;
                ctx.fillStyle = '#1f2c33';
                ctx.font = `${(char_radius * font_ratio)|0}px Arial`;
                ctx.textAlign = 'center';
                ctx.textBaseline = 'middle';
                ctx.fillText(`${this._value !== null ? this._value : t('No data')}${this._units}`,
                    char_x, char_y, size - char_shadow * 4 - char_weight * 2
                );

                ctx.fillStyle = '#768d99';
                ctx.font = `${(char_radius * font_ratio * .5)|0}px Arial`;
                ctx.textBaseline = 'top';

                ctx.textAlign = 'left';
                ctx.fillText(`${this._min}${this._min != '' ? this._units : ''}`,
                    char_weight * .75, size - char_weight * 1.25, size / 2 - char_weight
                );

                ctx.textAlign = 'right';
                ctx.fillText(`${this._max}${this._max != '' ? this._units : ''}`,
                    size - char_weight * .75, size - char_weight * 1.25, size / 2 - char_weight
                );

                requestAnimationFrame(animate);
            }
            else {
                this._last_value = this._value;
            }
        };

        requestAnimationFrame(animate);
    }
}
```

10. 打开 manifest.json 并添加：

- 将文件名 (class.widget.js) 添加到 assets/js 部分的数组中；
- 将类名 (WidgetLessonGaugeChart) 添加到 widget 部分的 js_class 参数中。

现在，WidgetLessonGaugeChart 类将随仪表板自动加载。

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
```

```
    "version": "1.0",
    "author": "Zabbix",
    "actions": {
        "widget.lesson_gauge_chart.view": {
            "class": "WidgetView"
        }
    },
    "widget": {
        "js_class": "WidgetLessonGaugeChart"
    },
    "assets": {
        "js": ["class.widget.js"]
    }
}
```

为小组件添加 CSS 样式

在本节中，你将学习如何添加自定义 CSS 样式，使小组件看起来更美观。

1. 对于小组件样式，在 assets 目录中创建一个新的 css 目录。
2. 在 assets/css 目录中创建 widget.css 文件。要为小组件元素设置样式，请使用选择器 div.dashboard-widget-{widget id}。要为整个小组件配置 CSS，请使用选择器 form.dashboard-widget-{widget id}

ui/modules/lesson_gauge_chart/assets/css/widget.css

```
div.dashboard-widget-lesson_gauge_chart {
    display: grid;
    grid-template-rows: 1fr;
    padding: 0;
}

div.dashboard-widget-lesson_gauge_chart .chart {
    display: grid;
    align-items: center;
    justify-items: center;
}

div.dashboard-widget-lesson_gauge_chart .chart canvas {
    background: white;
}

div.dashboard-widget-lesson_gauge_chart .description {
    padding-bottom: 8px;
    font-size: 1.750em;
    line-height: 1.2;
    text-align: center;
}

.dashboard-grid-widget-hidden-header div.dashboard-widget-lesson_gauge_chart .chart {
    margin-top: 8px;
}
```

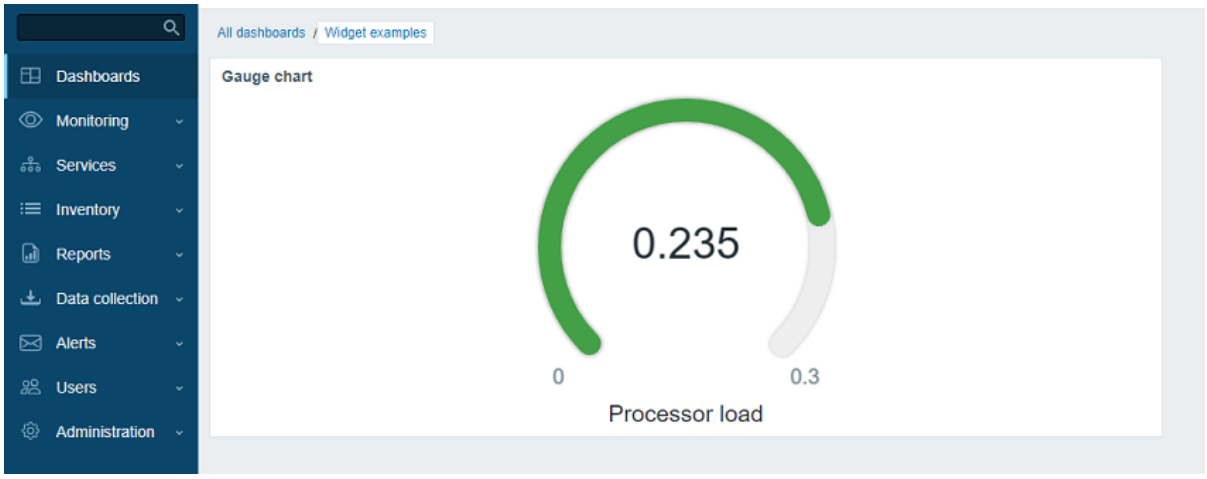
3. 打开 manifest.json，并将 CSS 文件名 (widget.css) 添加到 assets/css 部分的数组中。这将允许 widget.css 中定义的 CSS 样式随仪表板页面一起加载。

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.0",
    "author": "Zabbix",
    "actions": {
```

```
        "widget.lesson_gauge_chart.view": {
            "class": "WidgetView"
        }
    },
    "widget": {
        "js_class": "WidgetLessonGaugeChart"
    },
    "assets": {
        "css": ["widget.css"],
        "js": ["class.widget.js"]
    }
}
```

4. 刷新仪表板页面，查看小组件的完成版本。



示例

本节介绍了示例模块和小部件。您可以将其作为自定义模块的基础使用。

使用模块：

1. 下载 ZIP 压缩包。
2. 将内容解压到 Zabbix 前端安装目录下的 *modules* 目录内的单独文件夹中 (例如，*zabbix/ui/modules*)。
3. 在 Zabbix 前端中注册该模块。

模块示例

- 创建主机组时，将读取权限授予已配置的用户组 - hg_auto_perm.zip

小部件示例

- 最小小部件 - widget_min.zip
- "Hello, world" 仅使用 CSS 的小部件 - hello_world_css.zip
- "Hello, world" 仅使用 JavaScrip 的小部件 - hello_world_js.zip
- "Hello, world" 使用 PHP 的小部件 - hello_world_php.zip

您也可以使用Zabbix 原生窗口小部件 作为样板。

扩展开发的变更

从 **7.4** 到 **8.0** 的变更 ZBXNEXT-10445 `string` 部件字段 的最大长度已从 2048 个字符增加到 65535 个字符。
Zabbix 原生部件中的特定 `string` 字段现在也有各自的长度限制（例如，通过 `setMaxLength()` 函数设置）。

Zabbix 手册页

Zabbix 命令行工具和服务的参考文档，列出了可用命令、选项和使用示例。

zabbix_agent2

```
ZABBIX_AGENT2(8)            系统管理员手册            ZABBIX_AGENT2(8)

名称
       zabbix_agent2 - Zabbix agent 2

概要
       zabbix_agent2 [-c config-file]
       zabbix_agent2 [-c config-file] [-v] -p
       zabbix_agent2 [-c config-file] [-v] -t item-key
       zabbix_agent2 [-c config-file] -R runtime-option
       zabbix_agent2 [-c config-file] -T
       zabbix_agent2 -h
       zabbix_agent2 -V

描述
       zabbix_agent2 是一个用于监控各种服务参数的应用程序。

选项
       -c, --config config-file
              使用备用的 config-file，而不是默认配置文件。

       -R, --runtime-control runtime-option
              根据 runtime-option 执行管理功能。

           运行时控制选项
               userparameter_reload
                   从配置文件重新加载用户参数

               log_level_increase
                   提高日志级别

               log_level_decrease
                   降低日志级别

               help
                   列出可用的运行时控制选项

               metrics
                   列出可用的指标
               version
                   显示版本

       -p, --print
              打印已知监控项并退出。对于每个监控项，要么使用通用默认值，
              要么提供用于测试的特定默认值。这些默认值会以监控项键参数的
              形式列在方括号中。返回值会括在方括号中，并以前缀形式标明返
              回值类型，二者之间用竖线字符分隔。对于用户参数，类型始终为
              t，因为 agent 无法确定所有可能的返回值。显示为可工作的监控项，
              并不保证在 Zabbix 服务器或 zabbix_get 查询正在运行的 agent
              守护进程时也能正常工作，因为权限或环境可能不同。返回值类型
              如下：

           d   带小数部分的数字。

           m   不支持。这可能是由于查询了仅在主动模式下工作的监控项，例
```

如日志监控项，或者需要多个已采集值的监控项。权限问题或错误的用户参数也可能导致不支持状态。

s 文本。最大长度不限。

t 文本。与 s 相同。

u 无符号整数。

-t, --test item-key
测试单个监控项并退出。输出说明请参见 --print。

-v, --verbose
在测试监控项或打印已知监控项时启用详细输出。

-T, --test-config
验证配置文件并退出。

-h, --help
显示此帮助并退出。

-V, --version
输出版本信息并退出。

文件

/usr/local/etc/zabbix_agent2.conf
Zabbix agent 2 配置文件的默认位置（如果未在编译时修改）。

另请参见

文档

zabbix_agentd(8), zabbix_get(1), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8), zabbix_js(1), zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_AGENT2(8)

zabbix_agentd

ZABBIX_AGENTD(8) 系统管理员手册 ZABBIX_AGENTD(8)

名称

zabbix_agentd - Zabbix agent 守护进程

概要

```
zabbix_agentd [-c config-file]
zabbix_agentd [-c config-file] -p
zabbix_agentd [-c config-file] -t item-key
zabbix_agentd [-c config-file] -R runtime-option
zabbix_agentd [-c config-file] -T
zabbix_agentd -h
zabbix_agentd -V
```

描述

zabbix_agentd 是一个用于监控各种服务器参数的守护进程。

选项

-c, --config config-file
使用备用的 config-file，而不是默认配置文件。

-f, --foreground
在前台运行 Zabbix agent。

-R, --runtime-control runtime-option
根据 runtime-option 执行管理功能。

运行时控制选项
userparameter_reload
从配置文件重新加载用户参数

log_level_increase[=target]
提高日志级别；如果未指定 target，则影响所有进程

log_level_decrease[=target]
降低日志级别；如果未指定 target，则影响所有进程

日志级别控制目标
process-type
指定类型的所有进程（主动检查、collector、listener）

process-type,N
进程类型和编号（例如 listener,3）
pid 进程标识符，最大到 65535。对于更大的值，请将
target 指定为 "process-type,N"

-p, --print
打印已知监控项并退出。对于每个监控项，要么使用通用默认值，要么提供用于测试的特定默认值。这些默认值会以监控项键参数的形式列在方括号中。返回值会括在方括号中，并以前缀形式标明返回值类型，两者之间以竖线字符分隔。对于用户参数，类型始终为 t，因为 agent 无法确定所有可能的返回值。显示为可工作的监控项，并不保证在 Zabbix 服务器或 zabbix_get 查询正在运行的 agent 守护进程时一定可用，因为权限或环境可能不同。返回值类型如下：

d 带小数部分的数字。

m 不支持。这可能是由于查询了仅在主动模式下工作的监控项，例如日志监控监控项，或需要多个已采集值的监控项。权限问题或不正确的用户参数也可能导致不支持状态。

s 文本。最大长度不限。

t 文本。与 s 相同。

u 无符号整数。

-t, --test item-key
测试单个监控项并退出。输出说明请参见 --print。

-T, --test-config
验证配置文件并退出。

-h, --help
显示此帮助并退出。

-V, --version
输出版本信息并退出。

文件

/usr/local/etc/zabbix_agentd.conf
Zabbix agent 配置文件的默认位置（如果在编译时未修改）。

另请参见

Documentation

zabbix_get(1), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8), zabbix_js(1), zabbix_agent2(8), zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_AGENTD(8)

zabbix_get

ZABBIX_GET(1) 通用命令手册 ZABBIX_GET(1)

名称

zabbix_get - Zabbix 获取工具

概要

```
zabbix_get -s host-name-or-IP [-p port-number] [-I IP-address] [-t
timeout] -k item-key
zabbix_get -s host-name-or-IP [-p port-number] [-I IP-address] [-t
timeout] --tls-connect cert --tls-ca-file CA-file [--tls-crl-file
CRL-file] [--tls-agent-cert-issuer cert-issuer] [--tls-agent-cert-sub-
ject cert-subject] --tls-cert-file cert-file --tls-key-file key-file
[--tls-cipher13 cipher-string] [--tls-cipher cipher-string] -k item-key
zabbix_get -s host-name-or-IP [-p port-number] [-I IP-address] [-t
timeout] --tls-connect psk --tls-psk-identity PSK-identity
--tls-psk-file PSK-file [--tls-cipher13 cipher-string] [--tls-cipher
cipher-string] -k item-key
zabbix_get -h
zabbix_get -V
```

描述

zabbix_get 是一个命令行工具，用于从 Zabbix agent 获取数据。

选项

-s, --host host-name-or-IP
指定主机的主机名或 IP 地址。

-p, --port port-number
指定运行在该主机上的 agent 端口号。默认值为 10050。

-I, --source-address IP-address
指定源 IP 地址。

-t, --timeout seconds
指定超时时间。有效范围：1-600 秒（默认：30）

-k, --key item-key
指定要获取其值的监控项键。

-P, --protocol value
与 agent 通信时使用的协议。可选值：

auto	使用 JSON 协议连接，如失败则回退并重试纯文本协议（默认）
json	使用 JSON 协议连接
plaintext	使用纯文本协议连接，仅发送监控项键（6.4.x 及更早版本）

--tls-connect value
连接到 agent 的方式。可选值：

unencrypted
不加密连接（默认）

psk 使用 TLS 和预共享密钥连接

cert 使用 TLS 和证书连接

--tls-ca-file CA-file
包含用于对端证书验证的顶级 CA 证书的文件完整路径。

--tls-crl-file CRL-file
包含已吊销证书的文件完整路径。

--tls-agent-cert-issuer cert-issuer
允许的 agent 证书颁发者。

--tls-agent-cert-subject cert-subject
允许的 agent 证书主题。

--tls-cert-file cert-file
包含证书或证书链的文件完整路径。

--tls-key-file key-file
包含私钥的文件完整路径。

--tls-psk-identity PSK-identity
PSK 标识字符串。

--tls-psk-file PSK-file
包含预共享密钥的文件完整路径。

--tls-cipher13 cipher-string
适用于 OpenSSL 1.1.1 或更高版本、用于 TLS 1.3 的密码字符串。覆盖默认的密码套件选择条件。如果 OpenSSL 版本低于 1.1.1，则此选项不可用。

--tls-cipher cipher-string
GnuTLS 优先级字符串（用于 TLS 1.2 及以上）或 OpenSSL 密码字符串（仅用于 TLS 1.2）。覆盖默认的密码套件选择条件。

-h, --help
显示此帮助并退出。

-V, --version
输出版本信息并退出。

退出状态

如果 zabbix_get 成功从 Zabbix agent 获取到值，则退出状态为 0。
如果取值失败，则退出状态为 1。

示例

```
zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]"
zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]"
--tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file
--tls-agent-cert-issuer "CN=Signing CA,OU=IT operations,O=Example
Corp,DC=example,DC=com" --tls-agent-cert-subject "CN=server1,OU=IT op-
erations,O=Example Corp,DC=example,DC=com" --tls-cert-file /home/zab-
bix/zabbix_get.crt --tls-key-file /home/zabbix/zabbix_get.key
zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]"
--tls-connect psk --tls-psk-identity "PSK ID Zabbix agentd"
--tls-psk-file /home/zabbix/zabbix_agentd.psk
```

另请参见

文档

zabbix_agentd(8), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8),
zabbix_js(1), zabbix_agent2(8), zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_GET(1)

zabbix_js

ZABBIX_JS(1) 通用命令手册 ZABBIX_JS(1)

名称

zabbix_js - Zabbix JS 实用工具

概要

```
zabbix_js -s script-file -p input-param [-l log-level] [-t timeout]
zabbix_js -s script-file -i input-file [-l log-level] [-t timeout]
zabbix_js -h
zabbix_js -V
```

描述

zabbix_js 是一个命令行实用工具，可用于测试嵌入式脚本。

选项

-s, --script script-file
指定要执行的脚本文件名。如果文件名指定为 '-'，则脚本将从 stdin 读取。

-p, --param input-param
指定传递给脚本的输入参数。参数将按原样作为字符串传递，并可在脚本中通过 'value' 变量访问。与 -i 互斥。

-i, --input input-file
指定一个包含传递给脚本的输入参数的文件。文件内容将按原样作为字符串传递，并可在脚本中通过 'value' 变量访问。如果文件名指定为 '-'，则输入将从 stdin 读取。与 -p 互斥。

-w, --webdriver url
指定 webdriver URL。

-l, --loglevel log-level
指定日志级别（范围：0-5）。

-t, --timeout timeout
指定超时时间（秒）。有效范围：1-600 秒（默认值：10）

-h, --help
显示此帮助信息并退出。

-V, --version
输出版本信息并退出。

示例

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

另请参阅

Documentation

zabbix_agentd(8), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8),
zabbix_get(1), zabbix_agent2(8), zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_JS(1)

zabbix_proxy

ZABBIX_PROXY(8)　　系统管理员手册　　ZABBIX_PROXY(8)

名称

zabbix_proxy - Zabbix proxy 守护进程

概要

```
zabbix_proxy [-c config-file]
zabbix_proxy [-c config-file] -R runtime-option
zabbix_proxy [-c config-file] -T
zabbix_proxy -h
zabbix_proxy -V
```

描述

zabbix_proxy 是一个守护进程，用于从设备收集监控数据并将其发送到 Zabbix 服务器。

选项

-c, --config config-file
　使用指定的 config-file 替代默认配置文件。

-f, --foreground
　在前台运行 Zabbix proxy。

-R, --runtime-control runtime-option
　根据 runtime-option 执行管理功能。

运行时控制选项

config_cache_reload
　重新加载配置缓存。如果缓存当前正在加载，则忽略。活动的 Zabbix proxy 将连接到 Zabbix 服务

snmp_cache_reload
　重新加载 SNMP 缓存。

housekeeper_execute
　执行 housekeeper。如果 housekeeper 当前正在执行，则忽略。

diaginfo[=section]
　记录指定 section 的内部诊断信息。section 可以是 historycache、preprocessing、locks。默认

prof_enable[=target]
　启用性能分析；如果未指定 target，则影响所有进程

prof_disable[=target]
　禁用性能分析；如果未指定 target，则影响所有进程

log_level_increase[=target]
　提高日志级别；如果未指定 target，则影响所有进程。

log_level_decrease[=target]
　降低日志级别；如果未指定 target，则影响所有进程。

日志级别控制目标

process-type
　指定类型的所有进程（availability manager、browser poller、configuration syncer、data se

process-type,N
　进程类型和编号（例如，poller,3）

pid 进程标识符

性能分析控制目标
process-type
指定类型的所有进程（availability manager、browser poller、configuration syncer、data se

process-type,N
进程类型和编号（例如，history syncer,1）

pid 进程标识符

scope
（rwlock、mutex、processing）可与 process-type 一起使用（例如，history syncer,1,process

-T, --test-config
验证配置文件并退出。

-h, --help
显示此帮助并退出。

-V, --version
输出版本信息并退出。

文件
/usr/local/etc/zabbix_proxy.conf
Zabbix proxy 配置文件的默认位置（如果在编译时未修改）。

另请参阅
文档

zabbix_agentd(8), zabbix_get(1), zabbix_sender(1), zabbix_server(8),
zabbix_js(1), zabbix_agent2(8), zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_PROXY(8)

zabbix_sender

ZABBIX_SENDER(1) 常规命令手册 ZABBIX_SENDER(1)

名称
zabbix_sender - Zabbix sender 工具

概要
zabbix_sender [-v] -z server [-p port] [-I IP-address] [-t timeout] -s
host -k key -o value
zabbix_sender [-v] -z server [-p port] [-I IP-address] [-t timeout] [-s
host] [-T] [-N] [-r] -i input-file
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address]
[-t timeout] [-s host] -k key -o value
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address]
[-t timeout] [-s host] [-T] [-N] [-r] -i input-file
zabbix_sender [-v] -z server [-p port] [-I IP-address] [-t timeout] -s
host --tls-connect cert --tls-ca-file CA-file [--tls-crl-file CRL-file]
[--tls-server-cert-issuer cert-issuer] [--tls-server-cert-subject
cert-subject] --tls-cert-file cert-file --tls-key-file key-file
[--tls-cipher13 cipher-string] [--tls-cipher cipher-string] -k key -o
value
zabbix_sender [-v] -z server [-p port] [-I IP-address] [-t timeout] [-s
host] --tls-connect cert --tls-ca-file CA-file [--tls-crl-file
CRL-file] [--tls-server-cert-issuer cert-issuer]
[--tls-server-cert-subject cert-subject] --tls-cert-file cert-file
--tls-key-file key-file [--tls-cipher13 cipher-string] [--tls-cipher
cipher-string] [-T] [-N] [-r] -i input-file

```
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address]
[-t timeout] [-s host] --tls-connect cert --tls-ca-file CA-file
[--tls-crl-file CRL-file] [--tls-server-cert-issuer cert-issuer]
[--tls-server-cert-subject cert-subject] --tls-cert-file cert-file
--tls-key-file key-file [--tls-cipher13 cipher-string] [--tls-cipher
cipher-string] -k key -o value
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address]
[-t timeout] [-s host] --tls-connect cert --tls-ca-file CA-file
[--tls-crl-file CRL-file] [--tls-server-cert-issuer cert-issuer]
[--tls-server-cert-subject cert-subject] --tls-cert-file cert-file
--tls-key-file key-file [--tls-cipher13 cipher-string] [--tls-cipher
cipher-string] [-T] [-N] [-r] -i input-file
zabbix_sender [-v] -z server [-p port] [-I IP-address] [-t timeout] -s
host --tls-connect psk --tls-psk-identity PSK-identity --tls-psk-file
PSK-file [--tls-cipher13 cipher-string] [--tls-cipher cipher-string] -k
key -o value
zabbix_sender [-v] -z server [-p port] [-I IP-address] [-t timeout] [-s
host] --tls-connect psk --tls-psk-identity PSK-identity --tls-psk-file
PSK-file [--tls-cipher13 cipher-string] [--tls-cipher cipher-string]
[-T] [-N] [-r] -i input-file
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address]
[-t timeout] [-s host] --tls-connect psk --tls-psk-identity PSK-iden-
tity --tls-psk-file PSK-file [--tls-cipher13 cipher-string] [--tls-ci-
pher cipher-string] -k key -o value
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address]
[-t timeout] [-s host] --tls-connect psk --tls-psk-identity PSK-iden-
tity --tls-psk-file PSK-file [--tls-cipher13 cipher-string] [--tls-ci-
pher cipher-string] [-T] [-N] [-r] -i input-file
zabbix_sender -h
zabbix_sender -V
```

描述

zabbix_sender 是一个用于向 Zabbix 服务器或 proxy 发送监控数据的命令行工具。在 Zabbix 服务器上，应创建一个类型为 Zabbix trapper 且具有相应键值的监控项。请注意，只有来自该监控项"允许的主机"字段中指定的主机的传入值才会被接受。

选项

-c, --config config-file
: 使用 config-file。Zabbix sender 从 agentd 配置文件中读取服务器详细信息。默认情况下，Zabbix sender 不读取任何配置文件。仅支持 Hostname、ServerActive、SourceIP、TLSConnect、TLSCAFile、TLSCRLFile、TLSServerCertIssuer、TLSServerCertSubject、TLSCertFile、TLSKeyFile、TLSPSKIdentity 和 TLSPSKFile 参数。通过 HostnameItem 参数定义的 Hostname 不会被读取；在这种情况下，应通过命令行指定主机名（参见 -s 选项）。agent ServerActive 配置参数中定义的所有地址都会用于发送数据。如果向某个地址发送批量数据失败，则不会再向该地址发送后续批次。

-z, --zabbix-server server
: Zabbix 服务器的主机名或 IP 地址。如果主机由 proxy 监控，则应使用 proxy 的主机名或 IP 地址。与 --config 一起使用时，会覆盖 agentd 配置文件中指定的 ServerActive 参数条目。

-p, --port port
: 指定服务器上运行的 Zabbix 服务器 trapper 的端口号。默认值为 10051。与 --config 一起使用时，会覆盖 agentd 配置文件中 ServerActive 参数指定的端口条目。

-I, --source-address IP-address
: 指定源 IP 地址。与 --config 一起使用时，会覆盖 agentd 配置文件中指定的 SourceIP 参数。

-t, --timeout seconds
指定超时时间。有效范围：1-300 秒（默认值：60）

-s, --host host
指定监控项所属的主机名（在 Zabbix 前端中注册的名称）。主机 IP 地址和 DNS 名称不起作用。与 --config 一起使用时，会覆盖 agentd 配置文件中指定的 Hostname 参数。

-k, --key key
指定要向其发送值的监控项键值。

-o, --value value
指定监控项值。

-i, --input-file input-file
从输入文件加载值。将 <input-file> 指定为 - 可从标准输入读取值。文件的每一行包含以空格分隔的以下内容：<hostname> <key> <value>。每个值必须单独占一行。每行必须包含 3 个以空格分隔的条目：<hostname> <key> <value>，其中“hostname”是已在 Zabbix 前端中注册的受监控主机名称，“key”是目标监控项键值，“value”是要发送的值。将 <hostname> 指定为 -，可使用 agent 配置文件或 --host 参数中的主机名。

输入文件中的行示例：

```
"Linux DB3" db.connections 43
```

必须在 Zabbix 前端的监控项配置中正确设置值类型。Zabbix sender 每个连接最多发送 250 个值。输入文件内容必须采用 UTF-8 编码。输入文件中的所有值都会按照从上到下的顺序依次发送。条目必须遵循以下格式规则：

- 支持带引号和不带引号的条目。
- 双引号是引号字符。
- 包含空格的条目必须使用引号。
- 带引号的条目中的双引号和反斜杠字符必须使用反斜杠转义。
- 不支持对不带引号的条目进行转义。
- 带引号的字符串支持换行符转义序列（\n）。
- 条目末尾的换行符转义序列会被删除。

-T, --with-timestamps
此选项只能与 --input-file 选项一起使用。

输入文件的每一行必须包含 4 个以空格分隔的条目：<hostname> <key> <timestamp> <value>。时间戳应采用 Unix 时间戳格式。如果目标监控项有引用它的触发器，则所有时间戳必须按递增顺序排列，否则事件计算将不正确。

输入文件中的行示例：

```
"Linux DB3" db.connections 1429533600 43
```

有关详细信息，请参阅 --input-file 选项。

如果向处于“无数据”维护类型中的主机发送带时间戳的值，该值将被丢弃；但是，可以在维护期已过期时发送带时间戳的值，并且该值会被接受。

-N, --with-ns
       此选项只能与 --with-timestamps 选项一起使用。

       输入文件的每一行必须包含 5 个以空格分隔的条目：
       <hostname> <key> <timestamp> <ns> <value>。

       输入文件中的行示例：

        "Linux DB3" db.connections 1429533600 7402561 43

       有关详细信息，请参阅 --input-file 选项。

-r, --real-time
       值一经接收便逐个发送。读取标准输入时可以使用此选项。

--tls-connect value
       如何连接到服务器或 proxy。取值：

            unencrypted
                        不加密连接（默认值）

            psk   使用 TLS 和预共享密钥连接

            cert   使用 TLS 和证书连接

--tls-ca-file CA-file
       包含用于对等证书验证的顶级 CA 证书的文件的完整路径名。

--tls-crl-file CRL-file
       包含已吊销证书的文件的完整路径名。

--tls-server-cert-issuer cert-issuer
       允许的服务器证书颁发者。

--tls-server-cert-subject cert-subject
       允许的服务器证书主题。

--tls-cert-file cert-file
       包含证书或证书链的文件的完整路径名。

--tls-key-file key-file
       包含私钥的文件的完整路径名。

--tls-psk-identity PSK-identity
       PSK-identity 字符串。

--tls-psk-file PSK-file
       包含预共享密钥的文件的完整路径名。

--tls-cipher13 cipher-string
       OpenSSL 1.1.1 或更高版本中 TLS 1.3 使用的密码字符串。覆盖默认
       的密码套件选择标准。如果 OpenSSL 版本低于 1.1.1，则此选项不可用。

--tls-cipher cipher-string
       GnuTLS 优先级字符串（用于 TLS 1.2 及更高版本）或 OpenSSL 密码
       字符串（仅用于 TLS 1.2）。覆盖默认的密码套件选择标准。

-v, --verbose
       详细模式；使用 -vv 可显示更多详细信息。

-h, --help
       显示此帮助信息并退出。

-V, --version
输出版本信息并退出。

退出状态
如果值已发送且服务器已成功处理所有值，则退出状态为 0。如果数据已发送，但至少一个值处理失败，则退出状态为 2。如果数据发送失败，则退出状态为 1。

示例
zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -k mysql.queries -o 342.45
将 342.45 作为受监控主机的 mysql.queries 监控项的值发送。使用 agent 配置文件中定义的受监控主机和 Zabbix 服务器。

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -s "Monitored Host" -k mysql.queries -o 342.45
使用 agent 配置文件中定义的 Zabbix 服务器，将 342.45 作为 Monitored Host 主机的 mysql.queries 监控项的值发送。

zabbix_sender -z 192.168.1.113 -i data_values.txt
将 data_values.txt 文件中的值发送到 IP 地址为 192.168.1.113 的 Zabbix 服务器。主机名和键值在文件中定义。

echo "- hw.serial.number 1287872261 SQ4321ASDF" | zabbix_sender -c /usr/local/etc/zabbix_agentd.conf -T -i -
从命令行向 agent 配置文件中指定的 Zabbix 服务器发送带时间戳的值。输入数据中的短横线表示也应使用同一配置文件中的主机名。

echo '"Zabbix server" trapper.item ""' | zabbix_sender -z 192.168.1.113 -p 10000 -i -
从命令行向 IP 地址为 192.168.1.113、端口为 10000 的 Zabbix 服务器发送监控项的空值。空值必须使用一对空的双引号表示。

zabbix_sender -z 192.168.1.113 -s "Monitored Host" -k mysql.queries -o 342.45 --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-cert-file /home/zabbix/zabbix_agentd.crt --tls-key-file /home/zabbix/zabbix_agentd.key
使用 TLS 证书，将 342.45 作为 Monitored Host 主机中 mysql.queries 监控项的值发送到 IP 地址为 192.168.1.113 的服务器。

zabbix_sender -z 192.168.1.113 -s "Monitored Host" -k mysql.queries -o 342.45 --tls-connect psk --tls-psk-identity "PSK ID Zabbix agentd" --tls-psk-file /home/zabbix/zabbix_agentd.psk
使用 TLS 和预共享密钥（PSK），将 342.45 作为 Monitored Host 主机中 mysql.queries 监控项的值发送到 IP 地址为 192.168.1.113 的服务器。

另请参阅
文档

zabbix_agentd(8)、zabbix_get(1)、zabbix_proxy(8)、zabbix_server(8)、zabbix_js(1)、zabbix_agent2(8)、zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_SENDER(1)

zabbix_server

ZABBIX_SERVER(8) 系统管理员手册 ZABBIX_SERVER(8)

名称
zabbix_server - Zabbix 服务器守护进程

概要

```
zabbix_server [-c config-file]
zabbix_server [-c config-file] -R runtime-option
zabbix_server [-c config-file] -T
zabbix_server -h
zabbix_server -V
```

描述

zabbix_server 是 Zabbix 软件的核心守护进程。

选项

-c, --config config-file
使用替代的 config-file，而不是默认文件。

-f, --foreground
在前台运行 Zabbix 服务器。

-R, --runtime-control runtime-option
根据 runtime-option 执行管理功能。

运行时控制选项

config_cache_reload
重新加载配置缓存。如果缓存当前正在加载，则忽略。将使用默认配置文件（除非指定了 -c 选项）

snmp_cache_reload
重新加载 SNMP 缓存。

housekeeper_execute
执行 housekeeper。如果 housekeeper 当前正在执行，则忽略。

trigger_housekeeper_execute
执行 trigger housekeeper（移除已删除触发器的故障）。如果 trigger housekeeper 当前正在执

secrets_reload
从 Vault 重新加载密钥。如果配置缓存或密钥当前正在加载，则忽略。

diaginfo[=section]
记录指定 section 的内部诊断信息。section 可以是 historycache、preprocessing、alerting、

prof_enable[=target]
启用性能分析；如果未指定 target，则影响所有进程

prof_disable[=target]
禁用性能分析；如果未指定 target，则影响所有进程

ha_status
显示高可用集群状态。只能在活动节点上执行。

ha_remove_node
ha_remove_node[=name|ID] 根据 ha_status 中列出的名称或 ID 删除高可用节点。只能在活动节点

ha_set_failover_delay[=delay]
设置高可用故障切换延迟。可以使用时间后缀指定延迟。只能在活动节点上执行。

log_level_increase[=target]
提高日志级别；如果未指定 target，则影响所有进程

log_level_decrease[=target]
降低日志级别；如果未指定 target，则影响所有进程

日志级别控制目标

process-type
指定类型的所有进程（alerter、alert manager、alert syncer、availability manager、browser

process-type,N
进程类型和编号（例如，poller,3）

pid 进程标识符

性能分析控制目标
process-type
指定类型的所有进程（alerter、alert manager、alert syncer、availability manager、browser

process-type,N
进程类型和编号（例如，history syncer,1）

pid 进程标识符

scope
（rwlock、mutex、processing）可与 process-type 一起使用（例如，history syncer,1,process

-T, --test-config
验证配置文件并退出。

-h, --help
显示此帮助并退出。

-V, --version
输出版本信息并退出。

文件
/usr/local/etc/zabbix_server.conf
Zabbix 服务器配置文件的默认位置（如果编译时未修改）。

另请参见
文档

zabbix_agentd(8), zabbix_get(1), zabbix_proxy(8), zabbix_sender(1),
zabbix_js(1), zabbix_agent2(8), zabbix_web_service(8)

Zabbix 2022-01-06 ZABBIX_SERVER(8)

zabbix_web_service

ZABBIX_WEB_SERVICE(8) 系统管理员手册 ZABBIX_WEB_SERVICE(8)

名称
zabbix_web_service - Zabbix Web 服务

概要
zabbix_web_service [-c config-file]
zabbix_web_service [-c config-file] -T
zabbix_web_service -h
zabbix_web_service -V

描述
zabbix_web_service 是一个为 Zabbix 组件提供 Web 服务的应用程序。

选项
-c, --config config-file
使用备用的 config-file，而不是默认配置文件。

-T, --test-config
验证配置文件并退出。

-h, --help
显示此帮助信息并退出。

-V, --version
输出版本信息并退出。

文件
/usr/local/etc/zabbix_web_service.conf
Zabbix Web 服务配置文件的默认位置（如果在编译时未修改）。

另请参阅
Documentation

zabbix_agentd(8), zabbix_get(1), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8), zabbix_js(1), zabbix_agent2(8)

Zabbix 2022-01-06 ZABBIX_WEB_SERVICE(8)

版权声明

Zabbix 文档并非依据 AGPL-3.0 许可证发布。
使用 Zabbix 文档须遵守以下条款：

您可以仅为个人使用而制作本文件的纸质副本。
允许转换为其他格式，前提是实际内容不得以任何方式被更改或编辑。
您不得以任何形式或通过任何介质发布或分发本文件，除非您以与 Zabbix 发布方式类似的方式分发本文件（即在 Zabbix 网站上以电子方式提供下载），或者通过 USB 或类似介质分发，但前提是该文档与软件一同发布在同一介质上。
任何其他使用方式，例如分发纸质副本，或在其他出版物中全部或部分使用本文件，均须事先获得 Zabbix 授权代表的书面同意。
对于本文件中未在上述条款中明确授予的任何及所有权利，Zabbix 均予以保留。